사용자 가이드

AWS OpsWorks

API 버전 2013-02-18

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS OpsWorks: 사용자 가이드

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

Table of Contents

란 무엇인가요 AWS OpsWorks? 1
AWS OpsWorks 서비스 1
AWS OpsWorks for Puppet Enterprise 4
OpsWorks for Puppet Enterprise 리전 지원 5
수명 종료 관련 자주 묻는 질문(FAQs) 6
이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까? 6
아무 조치도 취하지 않으면 어떻게 됩니까? 6
신규 고객을 AWS OpsWorks for Puppet Enterprise 수락하고 있습니까? 7
수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까? 7
어떤 수준의 기술 지원을 사용할 수 있습니까 AWS OpsWorks for Puppet Enterprise? 7
저는 현재 OpsWorks for Puppet Enterprise의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요? 7
에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks for Puppet Enterprise? 7
시작하기 7
사전 조건 8
Puppet 마스터 생성 12
구성 완료 23
관리할 노드 추가 27
Puppet Enterprise 콘솔에 로그인 30
선택 사항: CodeCommit 사용 33
CloudFormation에서 Puppet 마스터 만들기 40
사전 조건 40
AWS CloudFormation에서 Puppet Enterprise 마스터 만들기 41
사용자 지정 도메인을 사용하도록 서버 업데이트 46
사전 조건 47
제한 사항 47
사용자 지정 도메인을 사용하도록 서버 업데이트 48
참고 51
태그 작업 51
에서 태그 작동 방식 AWS OpsWorks for Puppet Enterprise 52
OpsWorks for Puppet Enterprise에서 태그 추가 및 관리(콘솔) 54
OpsWorks for Puppet Enterprise(CLI) 에서 태그 추가 및 관리 56
참고 60
서버 백업 및 복원 61

OpsWorks for Puppet Enterprise 서버 백업 ... 61
OpsWorks for Puppet Enterprise 서버 복원 ... 65
시스템 유지 관리 ... 66
시스템 유지 관리 구성 ... 67
요청 시 시스템 유지 관리 시작 ... 69
유지 관리 후 사용자 지정 구성 및 파일 복원 ... 69
자동으로 노드 추가 ... 70
1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성 ... 71
2단계: 무인 연결 스크립트를 사용하여 인스턴스 생성 ... 71
노드 제거 ... 72
참고 ... 73
Puppet 마스터 삭제 ... 73
1단계: 관리되는 노드 연결 해제 ... 74
2단계: 서버 삭제 ... 74
참고 ... 75
Puppet 서버를 Amazon EC2로 마이그레이션 ... 75
1단계: Puppet에 문의하여 라이선스 구매 ... 76
2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기 ... 76
3단계: OpsWorks for Puppet Enterprise 서버의 백업 만들기 ... 76
4단계: 새 EC2 인스턴스 시작 ... 77
5단계: 새 EC2 인스턴스에 Puppet Enterprise를 설치합니다. ... 78
6단계: 새 EC2 인스턴스에서 백업 복원 ... 79
7단계: Puppet 라이선스 구성 ... 80
8단계: 노드 마이그레이션 ... 80
9단계: OpsWorks for Puppet Enterprise 서버를 삭제합니다 ... 82
사용 AWS CloudTrail ... 83
CloudTrail의 OpsWorks for Puppet Enterprise 정보 ... 83
OpsWorks for Puppet Enterprise 로그 파일 항목 이해 ... 84
문제 해결 ... 86
일반적인 문제 해결 팁 ... 87
구체적 오류 해결 ... 87
추가 도움말 및 지원 ... 91
AWS OpsWorks for Chef Automate ... 93
에 대한 리전 지원 AWS OpsWorks for Chef Automate ... 96
서비스 종료 관련 자주 묻는 질문 ... 97
이번 서비스 종료로 인해 기존 사용자는 어떤 영향을 받게 됩니까? ... 97

아무 조치도 취하지 않으면 어떻게 됩니까? 97
어떤 대안으로 전환할 수 있나요? 98
이 서비스는 여전히 신규 고객을 받고 있나요? 98
수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까? 98
어떤 수준의 기술 지원을 이용할 수 있나요? 98
저는 현재 OpsWorks for Chef Automate의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요? 98
내년에 주요 기능이 릴리스될 예정인가요? 99
Chef Automate 2로 업그레이드 99
Chef Automate 2로 업그레이드하기 위한 사전 요구 사항 99
업그레이드 프로세스 정보 100
Chef Automate 2로 업그레이드(콘솔) 100
Chef Automate 2(CLI)로 업그레이드 101
AWS OpsWorks for Chef Automate 서버를 Chef Automate 1로 롤백(CLI) 102
참고 103
시작하기 103
사전 조건 103
Chef Automate 서버 생성 105
구성 완료 및 쿡북 업로드 118
관리할 노드 추가 127
Chef Automate 대시보드에 로그인 133
CloudFormation에서 Chef Automate 서버 만들기 135
사전 조건 136
AWS CloudFormation에서 Chef Automate 서버 만들기 137
사용자 지정 도메인을 사용하도록 서버 업데이트 143
사전 조건 144
제한 사항 47
사용자 지정 도메인을 사용하도록 서버 업데이트 48
참고 51
스타터 키트 재생성 148
를 사용하여 AWS OpsWorks for Chef Automate 스타터 키트 재생성 AWS CLI 149
태그 작업 150
에서 태그 작동 방식 AWS OpsWorks for Chef Automate 151
에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (콘솔) 152
에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (CLI) 154
참고 159

서버 백업 및 복원 159
AWS OpsWorks for Chef Automate 서버 백업 160
AWS OpsWorks for Chef Automate 서버 복원 162
시스템 유지 관리 163
노드가 AWS OpsWorks 인증 기관을 신뢰하도록 보장 165
시스템 유지 관리 구성 166
요청 시 시스템 유지 관리 시작 167
유지 관리 후 사용자 지정 구성 및 파일 복원 168
규정 준수 검사 168
Chef Automate 2.0의 규정 준수 169
Chef Automate 1.x의 규정 준수 176
규정 준수에 대한 업데이트 182
커뮤니티 및 사용자 지정 규정 준수 프로필 183
참고 183
노드 제거 183
관련 항목 184
Chef Automate 서버 삭제 185
1단계: 관리되는 노드 연결 해제 185
2단계: 서버 삭제 185
Chef 자격 증명 재설정 186
사용 AWS CloudTrail 187
AWS OpsWorks for Chef Automate CloudTrail의 정보 188
AWS OpsWorks for Chef Automate 로그 파일 항목 이해 188
문제 해결 190
일반적인 문제 해결 팁 191
구체적 오류 해결 191
추가 도움말 및 지원 198
AWS OpsWorks 구성 관리(CM)의 보안 199
데이터 보호 200
와 통합 AWS Secrets Manager 201
데이터 암호화 202
유휴 데이터 암호화 202
전송 중 데이터 암호화 202
키 관리 202
ID 및 액세스 관리 202
대상 203

자격 증명을 통한 인증 ... 203
정책을 사용하여 액세스 관리 ... 206
AWS OpsWorks CM과 IAM의 작동 방식 ... 208
자격 증명 기반 정책 예제 ... 213
문제 해결 ... 217
AWS 관리형 정책 ... 218
의 교차 서비스 혼동된 대리자 방지 AWS OpsWorks CM ... 226
인터네트워크 트래픽 개인 정보 ... 230
로깅 및 모니터링 ... 230
규정 준수 검증 ... 230
복원성 ... 231
인프라 보안 ... 232
구성 및 취약성 분석 ... 232
보안 모범 사례 ... 233
AWS OpsWorks 스택 ... 235
스택 ... 238
계층 ... 238
레시피 및 수명 주기 이벤트 ... 238
인스턴스 ... 239
앱 ... 240
스택 사용자 지정 ... 240
리소스 관리 ... 241
보안 및 권한 ... 241
모니터링 및 로깅 ... 242
CLI, SDK 및 AWS CloudFormation 템플릿 ... 242
수명 종료 관련 자주 묻는 질문(FAQs) ... 243
이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까? ... 243
신규 고객을 AWS OpsWorks Stacks 수락하고 있습니까? ... 244
기존 스택을 어디로 마이그레이션해야 하나요? ... 244
수명 종료 후에도 기존 Amazon EC2 인스턴스를 유지하려면 어떻게 해야 하나요? ... 244
수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까? ... 245
어떤 수준의 기술 지원을 사용할 수 있나요 AWS OpsWorks Stacks? ... 245
에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks Stacks? ... 245
Systems Manager 애플리케이션 관리자로 애플리케이션 마이그레이션 ... 245
스크립트 작동 방식 ... 246
사전 조건 ... 246

제한 사항 247
시작 248
FAQ 261
문제 해결 272
AWS OpsWorks Stacks Detach in Place 도구 사용 273
프로세스 작동 방식 274
제한 사항 275
시작 276
시작하기 284
리전 지원 285
시작하기: 샘플 286
시작하기: Linux 306
시작하기: Windows 335
시작하기 - 쿡북 367
모범 사례 400
루트 디바이스 스토리지 400
서버 수 최적화 402
권한 관리 405
앱과 쿡북의 관리 및 배포 408
로컬로 쿡북 종속성 패키징 416
스택 420
EC2-Classic에서 스택 마이그레이션 422
새 스택 생성 424
VPC에서 스택 실행 432
스택 업데이트 443
스택 복제 444
스택 명령 실행 446
사용자 지정 JSON 사용 448
스택 삭제 451
계층 455
OpsWorks 계층 기초 456
Elastic Load Balancing 계층 471
Amazon RDS 서비스 계층 475
ECS 클러스터 계층 481
사용자 지정 계층 487
계층별 패키지 설치 488

인스턴스 489
AWS OpsWorks Stacks 인스턴스 사용 490
AWS OpsWorks Stacks 외부에서 생성된 컴퓨팅 리소스 사용 548
인스턴스 구성 편집 593
AWS OpsWorks 스택 인스턴스 삭제 595
SSH를 사용하여 로그인 597
RDP를 사용하여 로그인 600
앱 604
앱 추가 605
앱 배포 612
앱 편집 616
데이터베이스에 연결 617
환경 변수 사용 618
애플리케이션으로 데이터 전달 620
Git 리포지토리 SSH 키 사용 623
사용자 지정 도메인 사용 624
SSL 사용 626
쿡북과 레시피 634
쿡북 리포지토리 635
Chef 버전 638
Ruby 버전 656
사용자 지정 쿡북 설치 657
사용자 지정 쿡북 업데이트 661
레시피 실행 663
리소스 관리 671
스택에 리소스 등록 673
리소스 연결 및 이동 678
리소스 분리 683
리소스 등록 해제 685
Tags 688
스택 수준에서 태그 설정 689
계층 수준에서 태그 설정 691
를 사용하여 태그 관리 AWS CLI 693
태그 제한 694
모니터링 695
Amazon CloudWatch 사용 695

사용 AWS CloudTrail ... 706
Amazon CloudWatch Logs 사용 ... 709
Amazon CloudWatch Events 사용 ... 714
보안 및 권한 ... 715
사용자 권한 관리 ... 716
AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 ... 739
혼동된 대리자 방지 ... 744
EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 ... 748
SSH 액세스 관리 ... 752
보안 업데이트 관리 ... 758
보안 그룹 사용 ... 760
Chef 12 Linux ... 763
개요 ... 764
Chef 12로 전환 ... 765
지원되는 운영 체제 ... 766
지원되는 인스턴스 유형 ... 766
추가 정보 ... 766
데이터 백으로 이전 ... 767
이전 Chef 버전 ... 769
Linux용 Chef 11.10 및 이전 버전 ... 769
다른 AWS 서비스와 함께 AWS OpsWorks Stacks 사용 ... 1181
백엔드 데이터 스토어 사용 ... 1182
ElastiCache Redis ... 1190
Amazon S3 버킷 사용하기 ... 1204
AWS OpsWorks Stacks AWS CodePipeline 에서 사용 ... 1217
AWS OpsWorks Stacks CLI 사용 ... 1277
인스턴스 생성 ... 1279
앱 배포 ... 1282
앱 나열 ... 1283
목록 명령 ... 1284
배포 나열 ... 1286
탄력적 IP 주소 나열 ... 1287
인스턴스 나열 ... 1288
스택 나열 ... 1289
계층 나열 ... 1291
레시피 실행 ... 1295

Install Dependencies 1296
스택 구성 업데이트 1297
디버깅 및 문제 해결 안내서 1298
레시피 디버깅 1299
일반적인 디버깅 및 문제 해결 1315
AWS OpsWorks Stacks 에이전트 CLI 1324
agent_report 1327
get_json 1327
instance_report 1332
list_commands 1333
run_command 1333
show_log 1335
stack_state 1336
AWS OpsWorks Stacks 데이터 백 참조 1338
앱 데이터 백(aws_opsworks_app) 1343
명령 데이터 백(aws_opsworks_command) 1346
Amazon 클러스터 데이터 백(aws_opsworks_ecs_cluster) 1348
Elastic Load Balancing 데이터 백(aws_opsworks_elastic_load_balancer) 1349
인스턴스 데이터 백(aws_opsworks_instance) 1350
계층 데이터 백(aws_opsworks_layer) 1355
Amazon RDS 데이터 백(aws_opsworks_rds_db_instance) 1357
스택 데이터 백(aws_opsworks_stack) 1359
사용자 데이터 백(aws_opsworks_user) 1361
OpsWorks 에이전트 변경 사항 1362
Chef 12 에이전트 릴리스 1362
Chef 11.10 에이전트 릴리스 1365
리소스 1371
참조 설명서, 도구, 지원 리소스 1371
AWS 소프트웨어 개발 키트 1372
오픈 소스 소프트웨어 1372
AWS OpsWorks 문서 기록 1374
이전 업데이트 1381
..... mccclxxxv

란 무엇인가요 AWS OpsWorks?

> **Important**
>
> AWS OpsWorks 서비스가 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks 는 Puppet 또는 Chef. AWS OpsWorks Stacks를 사용하여 클라우드 엔터프라이즈에서 애플리케이션을 구성하고 운영하는 데 도움이 되는 구성 관리 서비스이며, 구성 관리를 위해 Chef 쿡북과 솔루션을 사용할 AWS OpsWorks for Chef Automate 수 있고, OpsWorks for Puppet Enterprise를 사용하면에서 Puppet Enterprise 마스터 서버를 구성할 수 있습니다 AWS. Puppet은 원하는 인프라 상태를 적용하고 온디맨드 작업을 자동화하기 위한 도구 세트를 제공합니다.

AWS OpsWorks 서비스

AWS OpsWorks for Puppet Enterprise

OpsWorks for Puppet Enterprise를 사용하면 AWS관리형 Puppet 마스터 서버를 생성할 수 있습니다. Puppet 마스터 서버는 인프라에서 노드를 관리하고 이들 노드에 대한 정보를 저장하며 Puppet 모듈을 위한 중앙 리포지토리 역할을 합니다. 모듈은 인프라 구성 방법에 대한 지침이 포함된 Puppet 코드 단위로 재사용과 공유가 가능합니다. Puppet Forge에서 커뮤니티 모듈을 다운로드하거나 Puppet 개발 키트를 사용하여 사용자 지정 모듈을 생성한 다음, Puppet Code Manager를 통해 배포를 관리할 수 있습니다.

OpsWorks for Puppet Enterprise는 완전 관리형 Puppet 마스터를 제공합니다. Puppet Master는 애플리케이션을 검사, 제공, 운영 및 미래에 대비할 수 있는 자동화 도구 모음이며, 노드 및 Puppet 활동에 대한 정보를 볼 수 있도록 사용자 인터페이스에 액세스할 수 있습니다. OpsWorks for Puppet Enterprise를 사용하면 Puppet을 사용하여 노드가 Amazon EC2 인스턴스이든 온프레미스 디바이스이든 상관없이 노드가 구성, 배포 및 관리되는 방식을 자동화할 수 있습니다. OpsWorks for Puppet Enterprise 마스터는 소프트웨어 및 운영 체제 구성, 패키지 설치, 데이터베이스 설치, 변경 관리, 정책 시행, 모니터링 및 품질 보증 같은 작업을 처리하여 전체 스택 자동화를 제공합니다.

OpsWorks for Puppet Enterprise가 Puppet Enterprise 소프트웨어를 관리하므로 서버는 사용자가 지정한 시점에 자동 백업이 가능하며, 항상 AWS와 호환되는 최신 버전의 Puppet를 실행하고,

항상 최신 보안 업데이트를 적용합니다. Amazon EC2 Auto Scaling 그룹을 사용하여 자동으로 새 Amazon EC2 노드를 서버와 연결할 수 있습니다.

AWS OpsWorks for Chef Automate

AWS OpsWorks for Chef Automate 를 사용하면 Chef Automate 프리미엄 기능이 포함된 AWS관리형 Chef 서버를 생성하고 Chef DK 및 기타 Chef 도구를 사용하여 관리할 수 있습니다. Chef 서버는 사용자의 환경에서 노드를 관리하고, 노드에 대한 정보를 저장하고, Chef 쿡북의 중앙 리포지토리로 사용됩니다. 쿡북에는 Chef를 사용하여 관리하는 각 노드에서 Chef Infra 클라이언트(`chef-client`) 에이전트가 실행하는 레시피가 포함됩니다. `knife` 및 Test Kitchen과 같은 Chef 도구를 사용하여 AWS OpsWorks for Chef Automate 서비스의 Chef 서버에서 노드와 쿡북을 관리할 수 있습니다.

Chef Automate는 포함된 서버 소프트웨어 패키지로서 지속적 배포 및 규정 준수 점검을 위한 자동화된 워크플로우를 제공합니다. AWS OpsWorks for Chef Automate 는 단일 Amazon Elastic Compute Cloud 인스턴스를 사용하여 Chef Automate, Chef Infra 및 Chef InSpec을 설치하고 관리합니다. 를 사용하면 AWS OpsWorks특정 변경 없이 커뮤니티에서 작성하거나 사용자 지정 Chef 쿡북을 사용할 AWS OpsWorks for Chef Automate수 있습니다.

는 단일 인스턴스에서 Chef Automate 구성 요소를 AWS OpsWorks for Chef Automate 관리하기 때문에 선택한 시간에 서버를 자동으로 백업할 수 있으며,는 항상 최신 마이너 버전의 Chef를 실행하고, 에는 항상 최신 보안 업데이트가 적용됩니다. Amazon EC2 Auto Scaling 그룹을 사용하여 자동으로 새 Amazon EC2 노드를 서버와 연결할 수 있습니다.

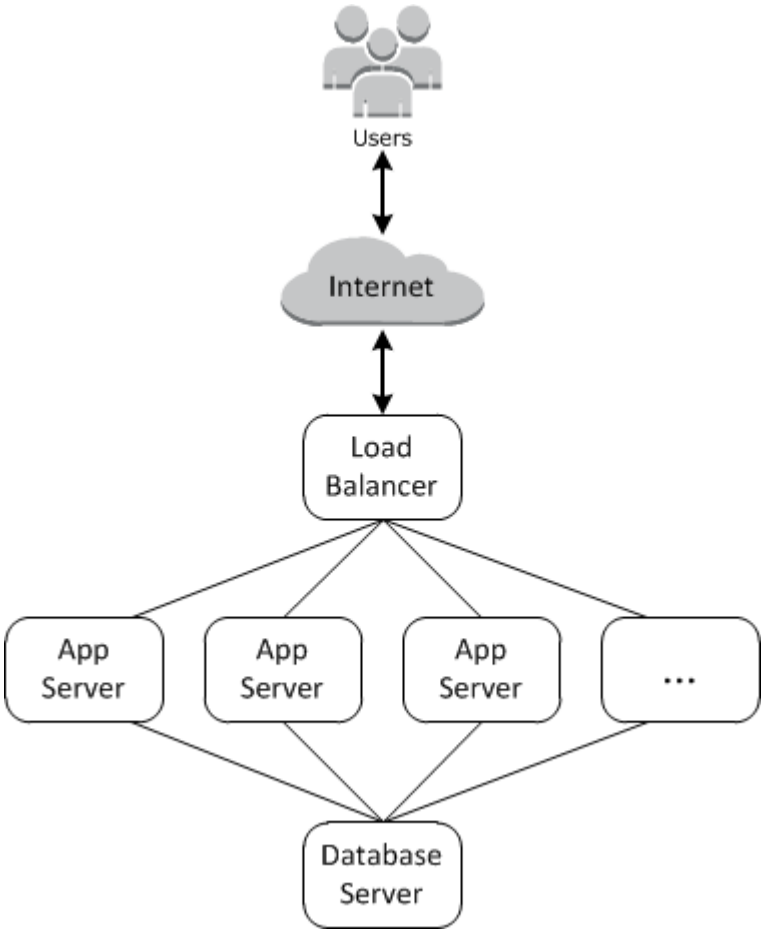
AWS OpsWorks 스택

클라우드 기반 컴퓨팅에는 일반적으로 EC2 인스턴스, Amazon Relational Database Service(RDS) 인스턴스와 같은 AWS 리소스 그룹이 포함됩니다. 예를 들어 웹 애플리케이션에는 일반적으로 애플리케이션 서버, 데이터베이스 서버, 로드 밸런서 등의 리소스가 필요합니다. 이러한 인스턴스 그룹을 일반적으로 스택이라고 합니다.

AWS OpsWorks 원래 서비스인 Stacks는 스택과 애플리케이션을 생성하고 관리하는 간단하고 유연한 방법을 제공합니다. AWS OpsWorks Stacks를 사용하면 스택에 애플리케이션을 배포하고 모니터링할 수 있습니다. 계층이라고 하는 전문화된 그룹으로 클라우드 리소스를 관리할 수 있는 스택을 생성할 수 있습니다. 계층은 애플리케이션에 서비스하거나 데이터베이스 서버를 호스팅하는 등 특정 목적에 사용되는 EC2 인스턴스 집합을 나타냅니다. 계층은 인스턴스에 패키지 설치, 앱 배포, 스크립트 실행과 같은 작업을 처리하기 위해 Chef 레시피에 의존합니다.

이와 달리 AWS OpsWorks for Chef Automate AWS OpsWorks Stacks는 Chef 서버를 요구하거나 생성하지 않습니다. AWS OpsWorks Stacks는 Chef 서버의 일부 작업을 자동으로 수행합니

다. AWS OpsWorks Stacks는 인스턴스 상태를 모니터링하고 필요할 경우 Auto Healing 및 Auto Scaling을 사용하여 새 인스턴스를 프로비저닝합니다. 다음 다이어그램은 간단한 애플리케이션 서버의 예입니다.



AWS OpsWorks for Puppet Enterprise

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

OpsWorks for Puppet Enterprise를 사용하면 몇 분 만에 Puppet Enterprise 마스터를 시작할 수 있으며 가 운영, 백업, 복원 및 소프트웨어 업그레이드를 AWS OpsWorks 처리할 수 있습니다. OpsWorks for Puppet Enterprise를 사용하면 Puppet 마스터를 관리하는 대신 핵심 구성 관리 작업에 집중할 수 있습니다. OpsWorks for Puppet Enterprise를 사용하면 동일한 구성을 사용하여 온프레미스 및 클라우드 인프라를 모두 관리할 수 있기 때문에 하이브리드 환경에서 작업의 규모를 효율적으로 조정할 수 있습니다. Puppet Enterprise 콘솔, AWS Management Console및 AWS CLI를 통해 Puppet 마스터 서버를 손쉽게 관리할 수 있습니다.

Puppet 마스터는 `puppet-agent` 소프트웨어에 특정 노드에 대한 구성 카탈로그를 제공하여 환경에서 노드 구성을 관리하고 Puppet 모듈을 위한 중앙 리포지토리 역할을 합니다. OpsWorks for Puppet Enterprise의 Puppet 마스터는 관리형 노드에 `puppet-agent`을(를) 배포하고 Puppet Enterprise의 프리미엄 기능을 제공합니다.

OpsWorks for Puppet Enterprise 마스터는 Amazon Elastic Compute Cloud 인스턴스에서 실행됩니다. OpsWorks for Puppet Enterprise 서버는 최신 버전의 Amazon Linux(Amazon Linux 2)와 가장 최신 버전의 Puppet Enterprise 마스터 버전, 2019.8.5를 실행하도록 구성되어 있습니다. Puppet Enterprise 2019.8.5 변경 사항에 대한 자세한 내용은 Puppet Enterprise 출시 정보를 참조하세요.

Puppet 소프트웨어의 새 버전이 나오면 AWS 테스트를 통과하는 즉시 서버에서 Puppet Enterprise 버전이 자동으로 업데이트되도록 시스템 유지 관리가 설계되어 있습니다. AWS는 철저한 테스트를 실시하여 Puppet 업그레이드가 프로덕션 환경을 지원하고 기존의 고객 환경을 방해하지 않는지 확인합니다.

지원되는 운영 체제를 실행하는 모든 온프레미스 컴퓨터 또는 EC2 인스턴스를 연결할 수 있고 OpsWorks for Puppet Enterprise 마스터에 네트워크로 액세스할 수 있습니다. `puppet` 에이전트 소프트웨어는 관리하려는 노드의 Puppet 마스터에 의해 설치됩니다.

주제

- OpsWorks for Puppet Enterprise 리전 지원
- AWS OpsWorks for Puppet Enterprise 수명 종료 FAQs
- OpsWorks for Puppet Enterprise 시작하기
- 를 사용하여 AWS OpsWorks for Puppet Enterprise 마스터 생성 AWS CloudFormation
- OpsWorks for Puppet Enterprise 서버를 사용자 지정 도메인을 사용하도록 업데이트
- AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업
- OpsWorks for Puppet Enterprise 서버 백업 및 복원
- OpsWorks for Puppet Enterprise 시스템 유지 관리
- OpsWorks for Puppet Enterprise에서 자동으로 노드 추가
- OpsWorks for Puppet Enterprise 서버를 노드에서 분리
- OpsWorks for Puppet Enterprise 서버 삭제
- OpsWorks for Puppet Enterprise 서버를 Amazon Elastic Compute Cloud(Amazon EC2)로 마이그레이션하는 방법
- 를 사용하여 OpsWorks for Puppet Enterprise API 호출 로깅 AWS CloudTrail
- OpsWorks for Puppet Enterprise 문제 해결

OpsWorks for Puppet Enterprise 리전 지원

다음 리전 엔드포인트는 OpsWorks for Puppet Enterprise 마스터를 지원합니다. OpsWorks for Puppet Enterprise는 Puppet 마스터와 동일한 리전 엔드포인트에서 Puppet 마스터에 연결되는 리소스(예: 인스턴스 프로파일, IAM 사용자, 서비스 역할)를 생성합니다. Puppet 마스터는 VPC에 속해야 합니다. VPC를 생성하여 사용하거나, 기존 VPC를 사용하거나, 기본 VPC를 사용할 수 있습니다.

- US East (Ohio) Region
- 미국 동부(버지니아 북부) 리전
- 미국 서부(캘리포니아 북부) 리전
- US West (Oregon) Region
- 아시아 태평양(도쿄) 리전
- 아시아 태평양(싱가포르) 리전
- 아시아 태평양(시드니) 리전
- Europe (Frankfurt) Region
- Europe (Ireland) Region

AWS OpsWorks for Puppet Enterprise 수명 종료 FAQs

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

주제

- 이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까?
- 아무 조치도 취하지 않으면 어떻게 됩니까?
- 신규 고객을 AWS OpsWorks for Puppet Enterprise 수락하고 있습니까?
- 수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?
- 어떤 수준의 기술 지원을 사용할 수 있습니까 AWS OpsWorks for Puppet Enterprise?
- 저는 현재 OpsWorks for Puppet Enterprise의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요?
- 에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks for Puppet Enterprise?

이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까?

기존 고객은 OpsWorks for Puppet Enterprise의 서비스 종료일인 2024년 3월 31일까지 영향을 받지 않습니다. 서비스 종료일이 지나면 고객은 더 이상 OpsWorks 콘솔 또는 API를 사용하여 서버를 관리할 수 없습니다.

아무 조치도 취하지 않으면 어떻게 됩니까?

2024년 3월 31일부터 OpsWorks 콘솔 또는 API를 사용하여 서버를 관리할 수 없습니다. 이때 당사는 백업 또는 유지 관리와 같은 서버의 지속적인 관리 기능 수행을 중단합니다. 고객에게 미치는 영향을 제한하기 위해 Puppet Enterprise 서버를 백업하는 EC2 인스턴스를 계속 실행할 예정이지만, OpsWorks for Puppet Enterprise 서비스 계약에 더 이상 적용(또는 청구)되지 않으므로 해당 라이선스는 더 이상 유효하지 않습니다. Puppet Enterprise를 사용하여 인프라를 계속 관리하려면 OpsWorks for Puppet Enterprise 서버를 Amazon Elastic Compute Cloud (Amazon EC2)로 마이그레이션하는 방법을 참조하세요.

신규 고객을 AWS OpsWorks for Puppet Enterprise 수락하고 있습니까?

아니요. AWS OpsWorks for Puppet Enterprise 는 더 이상 신규 고객을 받지 않습니다.

수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?

예. API와 콘솔의 수명이 종료되며 2024년 3월 31일부터 모든 지역에서 사용할 수 없게 됩니다. 를 AWS OpsWorks for Puppet Enterprise 사용할 수 AWS 리전 있는 위치 목록은 AWS 리전 서비스 목록을 참조하세요.

어떤 수준의 기술 지원을 사용할 수 있습니까 AWS OpsWorks for Puppet Enterprise?

AWS 는 오늘부터 수명 종료 날짜까지 AWS OpsWorks for Puppet Enterprise 고객이 보유한 것과 동일한 수준의 지원을 계속 제공합니다. 질문이나 우려 사항이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

저는 현재 OpsWorks for Puppet Enterprise의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요?

예외적인 상황이 아니라면 보통은 안 됩니다. 특별한 상황이 있는 경우 AWS re:Post에서 AWS Support 또는 AWS Premium Support를 통해 팀에 연락하여 이에 대한 세부 정보 및 정당화를 문의하면 요청을 검토하겠습니다.

에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks for Puppet Enterprise?

아니요. 서비스 수명이 종료되는 시점이므로 새로운 기능은 출시되지 않을 예정입니다. 하지만 서비스 종료 날짜까지 계속해서 보안을 개선하고 예상대로 서버를 관리할 예정입니다.

OpsWorks for Puppet Enterprise 시작하기

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로

마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

OpsWorks for Puppet Enterprise를 사용하면 AWS에서 Puppet Enterprise 서버를 실행할 수 있습니다. 약 15분 안에 Puppet Enterprise 마스터를 프로비저닝할 수 있습니다.

2021년 5월 3일부터 Puppet Enterprise용 OpsWorks는 일부 Puppet Enterprise 서버 속성을 AWS Secrets Manager에 저장합니다. 자세한 내용은 와 통합 AWS Secrets Manager 단원을 참조하십시오.

다음 안내서는 OpsWorks for Puppet Enterprise에서 첫 번째 Puppet 마스터를 생성하도록 돕습니다.

사전 조건

시작하기 전에 다음과 같은 사전 조건을 완료해야 합니다.

주제

- Puppet 개발 키트 설치
- Puppet Enterprise 클라이언트 도구 설치
- Git 제어 리포지토리 설정
- VPC 설정
- EC2 키 페어 설정(선택 사항)
- 사용자 지정 도메인 사용을 위한 사전 조건(선택 사항)

Puppet 개발 키트 설치

1. Puppet 웹 사이트에서 로컬 컴퓨터의 운영 체제에 맞는 Puppet 개발 키트를 다운로드하세요.
2. Puppet 개발 키트를 설치합니다.
3. 로컬 컴퓨터의 PATH 변수에 Puppet 개발 키트를 추가합니다.

   - Linux 또는 macOS 운영 체제의 경우, Bash 셸에서 다음 명령을 실행하여 PATH 변수에 Puppet 개발 키트를 추가할 수 있습니다.

     ```
     echo 'export PATH=/opt/puppetlabs/pdk/bin/pdk:$PATH' >> ~/.bash_profile && source
      ~/.bash_profile
     ```

- Windows 기반 운영 체제에서는 PowerShell 세션에서 다음 .NET Framework 명령을 사용하거나 시스템 속성에서 액세스할 수 있는 환경 변수 대화 상자에서 다음 .NET Framework 명령을 사용하여 Puppet 개발 키트를 PATH 변수에 추가할 수 있습니다. 다음 명령을 실행하기 위해 관리자 권한으로 PowerShell 세션을 실행해야 할 수 있습니다.

```
[Environment]::SetEnvironmentVariable("Path","new path value","Machine")
```

Puppet Enterprise 클라이언트 도구 설치

Puppet Enterprise(PE) 클라이언트 도구는 워크스테이션에서 Puppet Enterprise 서비스에 액세스할 수 있도록 해주는 명령줄 도구 세트입니다. 이들 도구는 서로 다른 다수의 운영 체제에 설치가 가능하며, Puppet을 사용하여 관리하고 있는 노드에 설치할 수도 있습니다. 도구에서 지원되는 운영 체제와 이들의 설치 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 Installing PE client tools 단원을 참조하세요.

Git 제어 리포지토리 설정

사용자가 Puppet 마스터를 시작할 수 있기 때문에 Puppet 모듈 및 클래스를 저장하고 변경-관리할 수 있도록 Git에 제어 리포지토리를 구성해야 합니다. Puppet Enterprise 마스터 서버를 시작하기 위한 단계에서 Git 리포지토리에 대한 URL과 리포지토리 액세스를 위한 HTTPS 또는 SSH 계정 정보가 필요합니다. Puppet Enterprise 마스터가 사용하게 될 제어 리포지토리를 설정하는 방법에 대한 자세한 정보는 Setting up a control repository 단원을 참조하세요. Puppet의 `control-repo`GitHub 기반 리포지토리 샘플을 위한 readme에서 제어 리포지토리 설정 지침을 찾아볼 수도 있습니다. 제어 리포지토리의 구조는 다음과 유사합니다.

```
### LICENSE
### Puppetfile
### README.md
### environment.conf
### hieradata
#   ### common.yaml
#   ### nodes
#       ### example-node.yaml
### manifests
#   ### site.pp
### scripts
#   ### code_manager_config_version.rb
#   ### config_version.rb
#   ### config_version.sh
```

```
### site
    ### profile
    #   ### manifests
    #       ### base.pp
    #       ### example.pp
    ### role
        ### manifests
            ### database_server.pp
            ### example.pp
            ### webserver.pp
```

CodeCommit를 사용하여 리포지토리 설정

CodeCommit를 사용하여 리포지토리를 새로 생성할 수 있습니다. CodeCommit를 사용하여 제어 리포지토리를 생성하는 방법에 대한 자세한 정보는 본 설명서의 the section called “선택 사항: CodeCommit 사용” 섹션을 참조하세요. CodeCommit에서 Git를 시작하는 방법에 대한 자세한 내용은 AWS CodeCommit 시작하기를 참조하세요. 리포지토리에 대해 OpsWorks for Puppet Enterprise 서버를 승인하려면 `AWSCodeCommitReadOnly` 정책을 IAM 인스턴스 프로파일 역할에 연결합니다.

VPC 설정

OpsWorks for Puppet Enterprise 마스터는 Amazon Virtual Private Cloud에서 운영해야 합니다. 이 서버를 기존 VPC에 추가하거나 기본 VPC를 사용하거나 새 VPC를 생성해 서버를 포함시킬 수 있습니다. Amazon VPC 및 새 VPC 생성 방법에 대한 자세한 내용은 Amazon VPC 시작하기 안내서를 참조하세요.

자체 VPC를 생성하거나 기존 VPC를 사용하는 경우, VPC는 다음과 같은 설정 또는 속성을 가져야 합니다.

- VPC에는 서브넷이 최소 하나 이상 있어야 합니다.

  OpsWorks for Puppet Enterprise 마스터에 공개적으로 액세스할 수 있는 경우 서브넷을 퍼블릭으로 설정하고 퍼블릭 IP 자동 할당을 활성화합니다.
- [DNS 확인]이 활성화되어야 합니다.
- 서브넷에서 [퍼블릭 IP 자동 할당]을 활성화합니다.

EC2 키 페어 설정(선택 사항)

일반적인 Puppet 서버 관리에는 SSH 연결이 필요하지 않거나 권장되지 않습니다. AWS Management Console 및 AWS CLI 명령을 사용하여 Puppet 서버에서 많은 관리 작업을 수행할 수 있습니다.

Puppet Enterprise 웹 기반 콘솔에서 로그인 암호를 변경하려는 경우에는 SSH를 사용하여 서버에 연결하려면 EC2 키 페어가 필요합니다. 기존 키 페어를 사용하거나 새 키 페어를 생성할 수 있습니다. 새 EC2 키 페어 생성 방법에 대한 자세한 정보는 Amazon EC2 키 페어를 참조하세요.

EC2 키 페어가 필요하지 않다면 곧 바로 Puppet Enterprise 마스터를 생성할 수 있습니다.

사용자 지정 도메인 사용을 위한 사전 조건(선택 사항)

고유 도메인에 Puppet Enterprise 마스터를 설정하여 사용자 지정 도메인에서 퍼블릭 엔드포인트를 지정하여 서버의 엔드포인트로 사용할 수 있습니다. 사용자 지정 도메인을 사용하는 경우 이 단원에서 자세히 설명하는 것과 같이 다음이 모두 필요합니다.

주제

- 사용자 지정 도메인 설정
- 인증서 가져오기
- 프라이빗 키 가져오기

사용자 지정 도메인 설정

사용자 지정 도메인에서 Puppet Enterprise 마스터를 실행하려면 서버의 퍼블릭 엔드포인트(예: `https://aws.my-company.com`)가 필요합니다. 사용자 지정 도메인을 지정하는 경우 이전 단원에서 설명한 것과 같이 인증서와 프라이빗 키를 제공해야 합니다.

서버를 만든 후 서버에 액세스하려면 기본 설정 DNS 서비스에 CNAME DNS 레코드를 추가합니다. 이 레코드는 사용자 지정 도메인을 Puppet 마스터 생성 프로세스에 의해 생성된 엔드포인트(서버 `Endpoint`의 속성 값)을 가리켜야 합니다. 서버가 사용자 지정 도메인을 사용하는 경우 생성된 `Endpoint` 값을 사용하여 서버에 액세스할 수 없습니다.

인증서 가져오기

사용자 지정 도메인에 Puppet 마스터를 설정하려면 PEM 형식의 HTTPS 인증서가 필요합니다. 이는 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다. Puppet Enterprise 마스터 생성 워크플로를 완료할 때 이 인증서를 지정하는 경우 사용자 지정 도메인과 프라이빗 키도 제공해야 합니다.

인증서 값에 대한 요구 사항은 다음과 같습니다.

- 자체 서명된 사용자 지정 인증서 또는 전체 인증서 체인을 제공할 수 있습니다.
- 인증서는 유효한 X509 인증서 또는 PEM 형식의 인증서 체인이어야 합니다.

- 인증서는 업로드 시점에 유효해야 합니다. 유효 기간이 시작되기 전(인증서의 `NotBefore` 날짜) 또는 만료된 후(인증서의 `NotAfter` 날짜)에는 인증서를 사용할 수 없습니다.
- 인증서의 일반 이름 또는 SAN(주체 대체 이름)이 있는 경우 사용자 지정 도메인 값과 일치해야 합니다.
- 인증서는 사용자 지정 프라이빗 키 필드의 값과 일치해야 합니다.

프라이빗 키 가져오기

사용자 지정 도메인에 Puppet 마스터를 설정하려면 HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키가 필요합니다. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다. 사용자 지정 프라이빗 키를 지정하는 경우 사용자 지정 도메인과 인증서도 제공해야 합니다.

Puppet Enterprise 마스터 생성

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

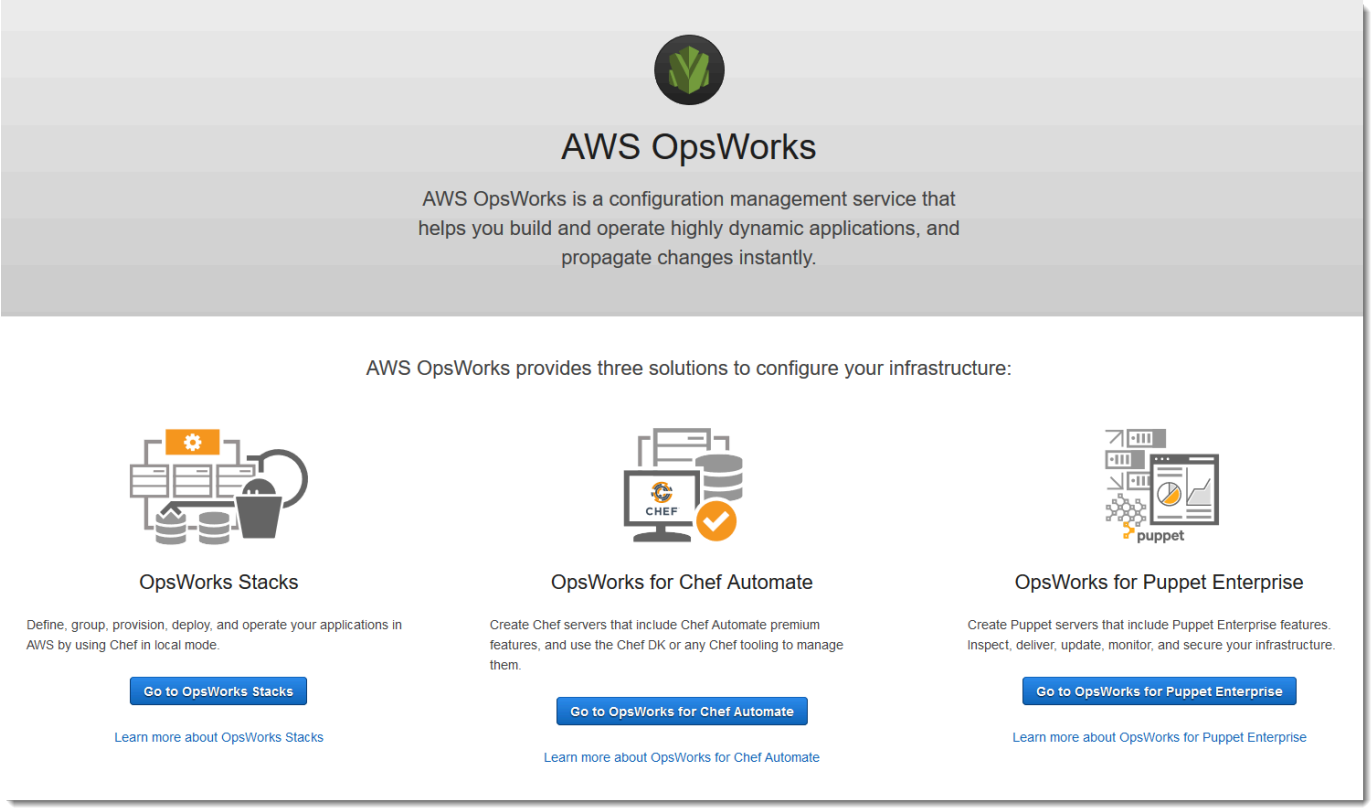
OpsWorks for Puppet Enterprise 콘솔이나 AWS CLI를 사용하여 Puppet 마스터를 생성할 수 있습니다.

주제

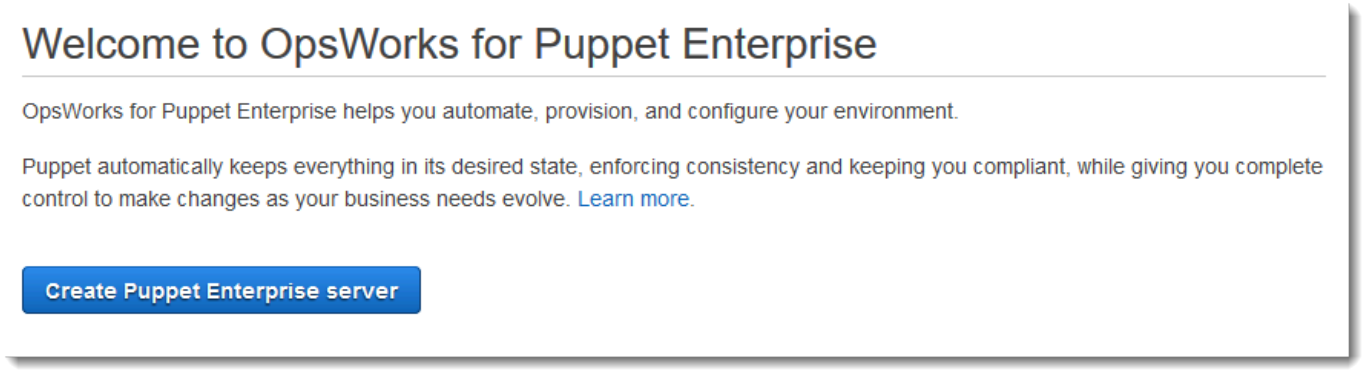
- 를 사용하여 Puppet Enterprise Master 생성 AWS Management Console
- 를 사용하여 Puppet Enterprise Master 생성 AWS CLI

를 사용하여 Puppet Enterprise Master 생성 AWS Management Console

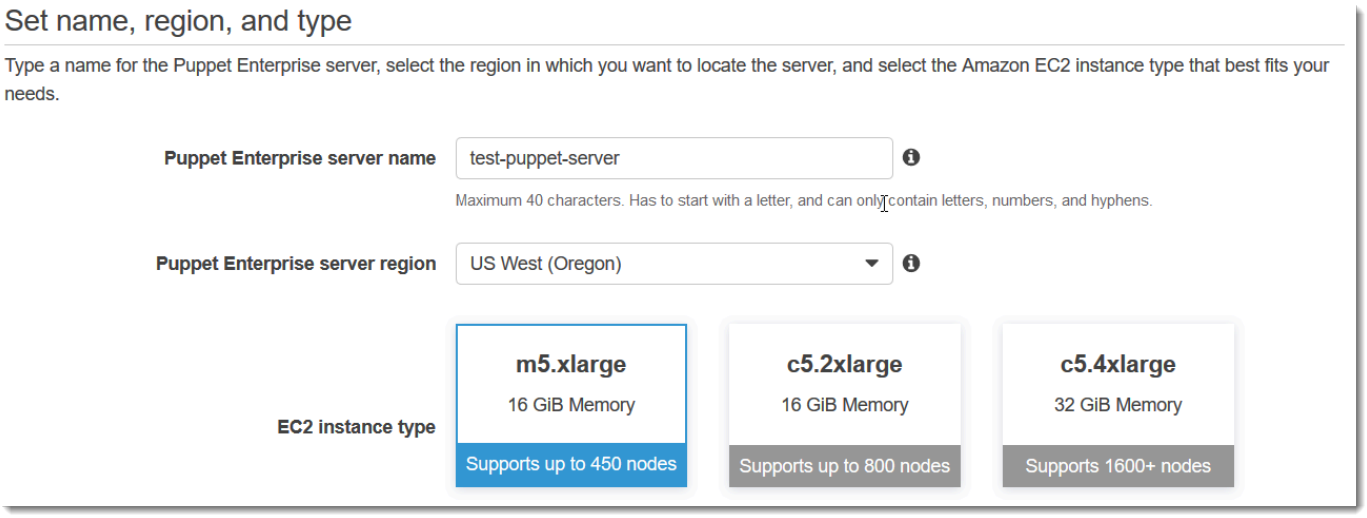
1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks 홈 페이지에서 OpsWorks for Puppet Enterprise로 이동을 선택합니다.



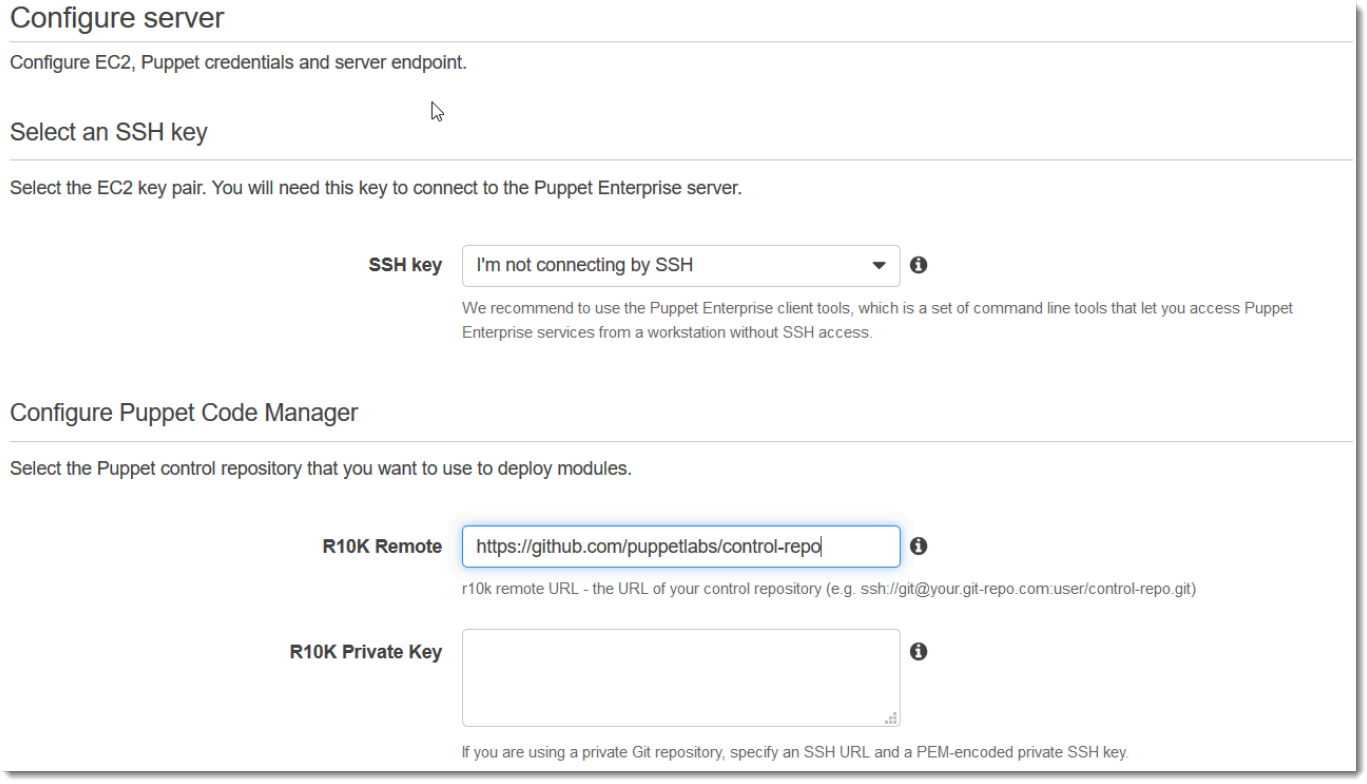
3. OpsWorks for Puppet Enterprise 홈 페이지에서 Puppet Enterprise 서버 생성을 선택합니다.



4. [이름, 리전, 유형 설정] 페이지에서 서버의 이름을 지정합니다. Puppet 마스터 이름은 최대 40자까지 가능하고 반드시 문자로 시작해야 하며, 영숫자와 대시만 포함할 수 있습니다. 지원되는 리전을 선택한 다음 관리하려는 노드 수에 맞는 인스턴스 유형을 선택합니다. 필요한 경우 서버가 생성된 후 인스턴스 유형을 변경할 수 있습니다. 이 안내를 위해 미국 서부(오레곤) 리전에서 m5.xlarge 인스턴스 유형을 생성합니다. 다음을 선택합니다.



5. 키 페어 이름을 지정하지 않으려면 서버 구성 페이지에서 SSH 키 드롭다운 목록의 기본 선택을 그대로 둡니다. Puppet Code Manager 구성 영역의 r10k 원격 필드에 Git 리모컨의 유효한 SSH 또는 HTTPS URL을 지정합니다. r10k 프라이빗 키 필드에서가 r10k 원격 리포지토리에 액세스하는 데 사용할 AWS OpsWorks 수 있는 SSH 프라이빗 키를 붙여 넣습니다. 이 값은 프라이빗 리포지토리를 생성할 때 Git에서 제공되지만, HTTPS 인증을 사용하여 제어 리포지토리에 액세스하는 경우에는 필요하지 않습니다. 다음을 선택합니다.

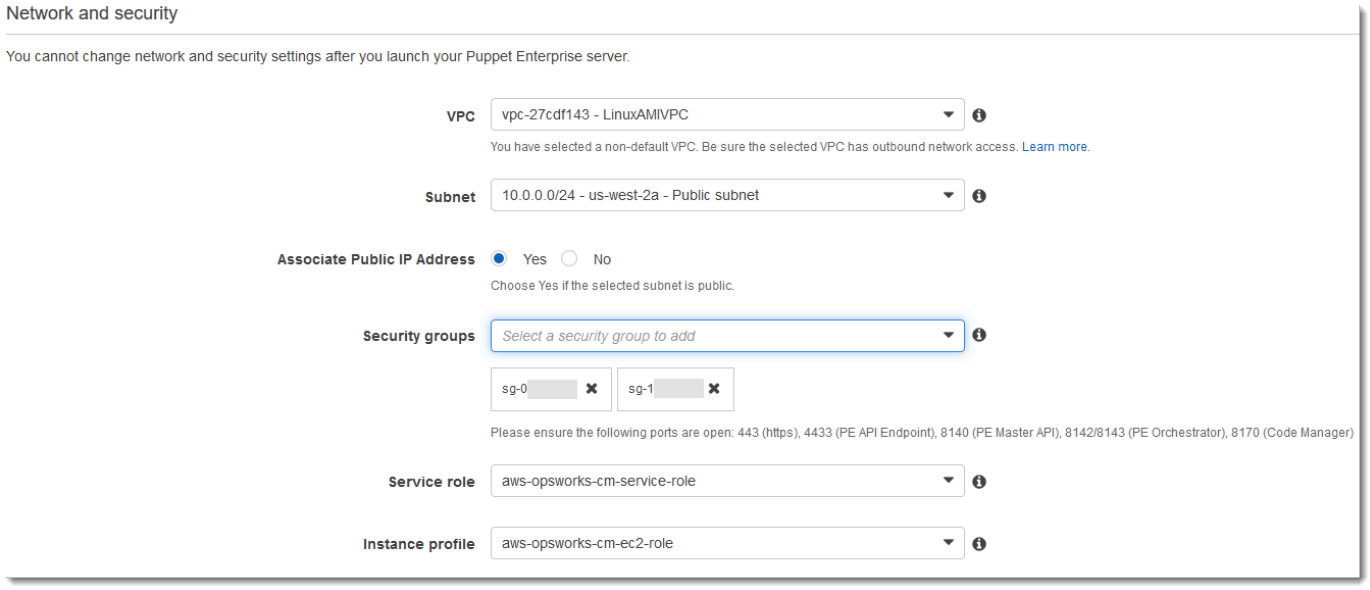


6. 서버 엔드포인트 지정에서 기본값인 자동 생성된 엔드포인트 사용을 그대로 두고 서버가 사용자 지정 도메인에 있는 것을 원하지 않는 한 다음을 선택합니다. 사용자 지정 도메인을 구성하려면 다음 단계로 이동합니다.
7. 사용자 지정 도메인을 사용하려면 서버 엔드포인트 지정에 있는 드롭다운 목록에서 사용자 지정 도메인 사용을 선택합니다.

   a. FQDN(정규화된 도메인 이름)에서 FQDN을 지정합니다. 사용하고자 하는 도메인 이름을 보유해야 합니다.

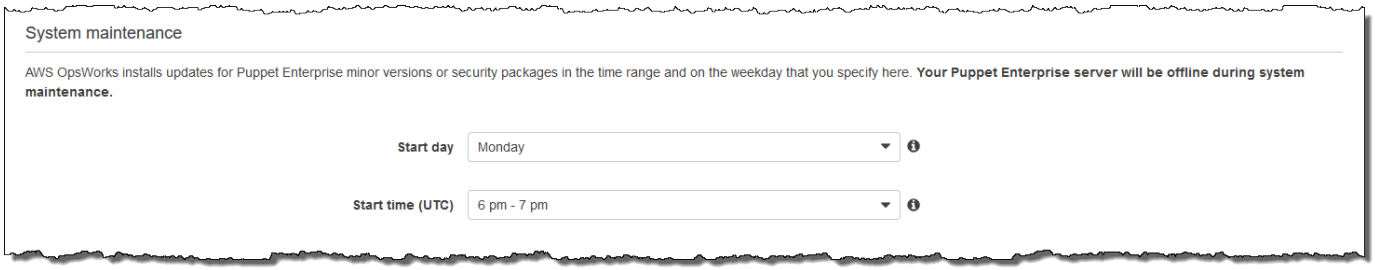
   b. SSL 인증서에서 `-----BEGIN CERTIFICATE-----`로 시작하고 `-----END CERTIFICATE-----`로 끝나는 전체 PEM 형식의 인증서를 붙여넣습니다. SSL 인증서 주체는 이전 단계에서 입력한 FQDN과 일치해야 합니다. 인증서 앞뒤에 있는 여분의 줄을 제거합니다.

   c. SSL 프라이빗 키에서 `-----BEGIN RSA PRIVATE KEY-----`로 시작하고 `-----END RSA PRIVATE KEY-----`로 끝나는 전체 RSA 프라이빗 키를 붙여넣습니다. SSL 프라이빗 키는 이전 단계에서 입력한 SSL 인증서의 퍼블릭 키와 일치해야 합니다. 프라이빗 키 앞뒤에 있는 여분의 줄을 제거합니다. 다음을 선택합니다.
8. 고급 설정 구성 페이지의 네트워크 및 보안 영역에서 VPC, 서브넷 및 하나 이상의 보안 그룹을 선택합니다. 사용하려는 보안 그룹이 아직 없는 경우 보안 그룹, 서비스 역할 및 인스턴스 프로파일을 AWS OpsWorks 생성할 수 있습니다. 서버는 여러 보안 그룹의 멤버일 수 있습니다. 이 페이지를 나간 뒤에는 Puppet 마스터의 네트워크 및 보안 설정을 변경할 수 없습니다.

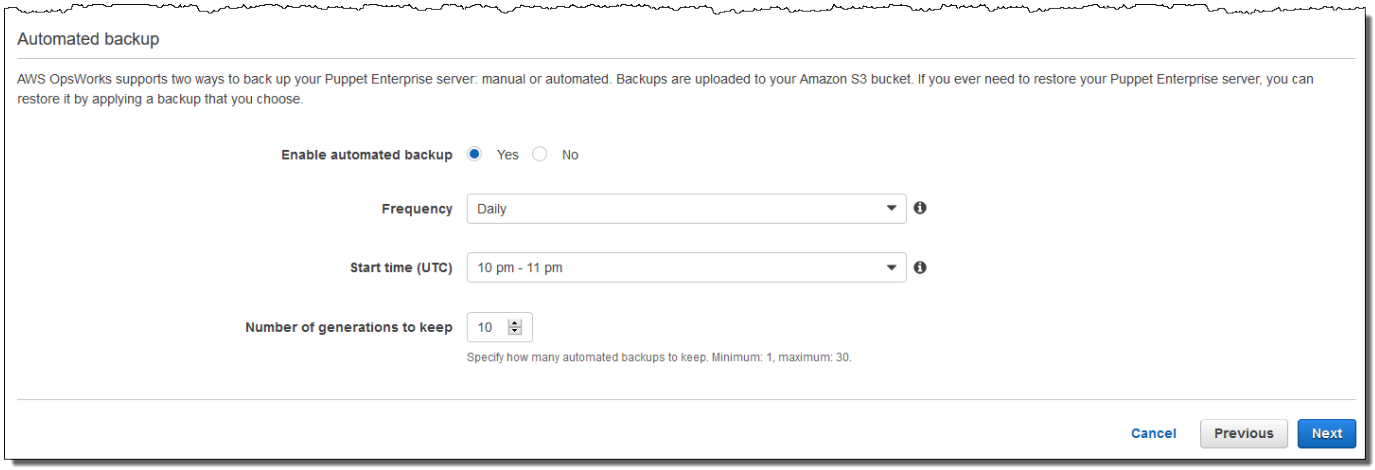


9. [시스템 유지 관리] 섹션에서 시스템 유지 관리를 시작하려는 날짜와 시간을 설정합니다. 시스템 유지 관리 중에는 서버가 오프라인 상태여야 하므로 정규 업무 시간 중 서버에 대한 수요가 낮은 시간을 선택하세요.

유지 관리 기간은 필수 항목입니다. AWS Management Console AWS CLI또는 APIs.



10. 백업을 구성합니다. 기본적으로 자동 백업이 활성화되어 있습니다. 선호하는 빈도와 자동 백업을 시작할 시간을 설정한 다음 Amazon Simple Storage Service에 저장할 백업 세대 수를 설정합니다. 백업은 최대 30개까지 보관할 수 있으며, 최대 개수에 도달하면 OpsWorks for Puppet Enterprise에서 가장 오래된 백업을 삭제해 새 백업을 저장할 공간을 마련합니다.



11. (선택 사항) 태그에서 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같은 서버 및 관련 리소스에 태그를 추가합니다. OpsWorks for Puppet Enterprise 서버를 태그 지정하는 방법에 대한 자세한 내용은 AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업 섹션을 참조하세요.
12. 고급 설정 구성을 마치면 [다음]을 선택합니다.
13. [검토] 페이지에서 선택 사항을 검토합니다. 서버를 생성할 준비가 되면 [시작]을 선택합니다.

    가 Puppet 마스터 AWS OpsWorks 를 생성할 때까지 기다리는 동안 로 이동하여 Starter Kit 스타터 키트를 사용하여 Puppet 마스터 구성 및 Puppet Enterprise 콘솔 자격 증명을 다운로드합니다. 서버가 온라인 상태가 되어 이러한 항목을 다운로드할 때까지 기다리지 마십시오.

    서버 생성이 끝나면 OpsWorks for Puppet Enterprise 홈 페이지에서 Puppet 마스터를 확인할 수 있으며, 상태는 온라인입니다. 서버가 온라인 상태가 되면

`https://your_server_name-randomID.region.opsworks-cm.io` 형식의 URL을 가진 서버 도메인에서 Puppet Enterprise 콘솔을 사용할 수 있습니다.

를 사용하여 Puppet Enterprise Master 생성 AWS CLI

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS CLI 명령을 실행하여 OpsWorks for Puppet Enterprise 마스터 서버를 생성하는 것은 콘솔에서 서버를 생성하는 것과 다릅니다. 사용하려는 기존 역할을 지정하지 않으면 콘솔에서가 서비스 역할 및 보안 그룹을 AWS OpsWorks 생성합니다. 에서 AWS CLI AWS OpsWorks 는 보안 그룹을 지정하지 않지만 자동으로 서비스 역할을 생성하지 않는 경우 사용자를 대신하여 보안 그룹을 생성할 수 있습니다. `create-server` 명령의 일부로 서비스 역할 ARN을 제공해야 합니다. 콘솔에서 AWS OpsWorks 가 Puppet 마스터를 생성하는 동안 Puppet Enterprise 콘솔의 스타터 키트와 로그인 자격 증명을 다운로드합니다. 를 사용하여 OpsWorks for Puppet Enterprise 마스터를 생성할 때는이 작업을 수행할 수 없으므로 새 OpsWorks for Puppet Enterprise 마스터가 온라인 상태가 된 후 JSON 처리 유틸리티를 AWS CLI사용하여 `create-server` 명령 결과에서 로그인 자격 증명과 스타터 키트를 가져옵니다.

로컬 컴퓨터에서 아직를 실행하지 않는 경우 AWS Command Line Interface 사용 설명서의 설치 지침에 AWS CLI 따라를 AWS CLI다운로드하고 설치합니다. 이 단원에서는 `create-server` 명령과 함께 사용할 수 있는 모든 파라미터를 다 설명하지는 않습니다. `create-server` 파라미터에 대한 자세한 내용은 AWS CLI 레퍼런스의 `create-server`를 참조하세요.

1. 사전 조건 단원을 완료해야 합니다. Puppet 마스터를 생성하려면 서브넷 ID가 필요하므로 VPC가 있어야 합니다.
2. 서비스 역할과 인스턴스 프로파일을 생성합니다.
3. 계정에서 서비스 역할의 ARNs 찾아 복사합니다.

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   `list-roles` 명령의 결과에서 다음과 같은 서비스 역할 ARN을 찾아봅니다. 서비스 역할 ARN을 기록해 둡니다. 이러한 값은 Puppet Enterprise 마스터를 만들기 위해 필요합니다.

```
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

4. 해당 계정에서 인스턴스 프로파일의 ARN을 찾아서 복사합니다.

```
aws iam list-instance-profiles --no-paginate
```

`list-instance-profiles` 명령의 결과에서 다음과 같은 인스턴스 프로파일 ARN을 찾아봅니다. 인스턴스 프로파일 ARN을 기록해 둡니다. 이러한 값은 Puppet Enterprise 마스터를 만들기 위해 필요합니다.

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

5. `create-server` 명령을 실행하여 OpsWorks for Puppet Enterprise 마스터를 생성합니다.

   - `--engine` 값은 `Puppet`이고 `--engine-model` 값은 `Monolithic`이며 `--engine-version` 값은 `2019` 또는 `2017`일 수 있습니다.

- 서버 이름은 AWS 계정 내에서 각 리전 내에서 고유해야 합니다. 서버 이름은 문자로 시작해야 하며, 그 이후에는 문자, 숫자 또는 하이픈(-)을 사용할 수 있으며, 최대 길이는 40자입니다.
- 3단계와 4단계에서 복사해 둔 인스턴스 프로파일 ARN 및 서비스 역할 ARN을 사용합니다.
- 유효한 인스턴스 유형은 `m5.xlarge`, `c5.2xlarge` 또는 `c5.4xlarge`입니다. 인스턴스 유형의 사양에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하세요.
- `--engine-attributes` 파라미터는 선택 사항이며, Puppet 관리자 암호를 지정하지 않을 경우 서버 생성 프로세스에서 해당 값이 자동으로 생성됩니다. `--engine-attributes`를 추가하는 경우 Puppet Enterprise 콘솔 웹 페이지에 로그인하기 위한 관리자 암호인 `PUPPET_ADMIN_PASSWORD`를 지정합니다. 암호는 ASCII 문자 8~32자로 되어 있어야 합니다.
- SSH 키 페어는 선택 사항이지만, 콘솔 관리자 암호를 재설정해야 하는 경우 Puppet 마스터에 연결하는 데 도움이 될 수 있습니다. SSH 키 페어 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.
- 사용자 지정 도메인을 사용하려면 명령에 다음 파라미터를 추가합니다. 그렇지 않은 경우 Puppet 마스터 생성 프로세스가 자동으로 엔드포인트를 생성합니다. 사용자 지정 도메인을 구성하려면 세 가지 파라미터 모두가 필요합니다. 이러한 파라미터 사용에 대한 추가 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer를 참조하세요.
  - `--custom-domain` - 서버의 선택적 퍼블릭 엔드포인트(예: `https://aws.my-company.com`).
  - `--custom-certificate` - PEM 형식의 HTTPS 인증서. 값은 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다.
  - `--custom-private-key` - HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다.
- 매주 시스템 유지 관리가 필요합니다. 유효한 값은 `DDD:HH:MM` 형식으로 지정해야 합니다. 지정한 시간은 협정 세계시(UTC)로 표시됩니다. `--preferred-maintenance-window`의 값을 지정하지 않으면 기본값은 화요일, 수요일 또는 금요일의 임의 한 시간입니다.
- `--preferred-backup-window`의 유효 값은 `HH:MM`(매일 백업) 또는 `DDD:HH:MM`(매주 백업) 형식 중 하나로 지정해야 합니다. 지정한 시간은 UTC 형식입니다. 기본값은 임의 일일 시작 시간입니다. 자동 백업을 옵트아웃하려면 대신에 `--disable-automated-backup` 파라미터를 추가합니다.
- `--security-group-ids`에는 공백으로 구분하여 하나 이상의 보안 그룹 ID를 입력합니다.
- `--subnet-ids`에는 서브넷 ID를 입력합니다.

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

다음은 예입니다.

```
aws opsworks-cm create-server --engine "Puppet" --engine-model
 "Monolithic" --engine-version "2019" --server-name "puppet-02" --
instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.xlarge" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' --key-pair "amazon-test"
 --preferred-maintenance-window "Mon:08:00" --preferred-backup-window
 "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-role-arn
 "arn:aws:iam::111122223333:role/service-role/aws-opsworks-cm-service-role" --
subnet-ids subnet-383daa71
```

다음 예제에서는 사용자 정의 도메인을 사용하는 Puppet 마스터를 만듭니다.

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
```

```
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

다음 예제에서는 `Stage: Production` 및 `Department: Marketing` 태그를 추가하는 Puppet 마스터를 만듭니다. OpsWorks for Puppet Enterprise 서버의 태그 추가 및 관리에 대한 자세한 내용은 이 안내서의 AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업 섹션을 참조하세요.

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

6. OpsWorks for Puppet Enterprise는 새 서버를 만드는 데 약 15분이 걸립니다. `create-server` 명령의 출력을 무시하거나 셸 세션을 닫지 마십시오. 다시 표시되지 않는 중요 정보가 출력에 포함되어 있을 수도 있기 때문입니다. `create-server` 결과에서 암호 및 스타터 키트를 가져오려면 다음 단계로 이동합니다.

   서버에서 사용자 지정 도메인을 사용하는 경우 `create-server` 명령 출력에서 `Endpoint` 속성 값을 복사합니다. 다음은 예입니다.

   ```
   "Endpoint": "puppet-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

7. OpsWorks for Puppet Enterprise에서 자동으로 암호를 생성하도록 선택한 경우 jq와 같은 JSON 프로세서를 사용하여 `create-server` 결과에서 사용 가능한 형식으로 암호를 추출할 수 있습니다. jq를 설치한 후에는 다음 명령을 실행하여 Puppet 관리자 암호 및 스타터 키트를 추출할 수 있

습니다. 3단계에서 고유의 암호를 제공하지 않은 경우 추출한 관리자 암호를 안전하면서 편리한 위치에 저장해야 합니다.

```
#Get the Puppet password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_ADMIN_PASSWORD") | .Value'

#Get the Puppet Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

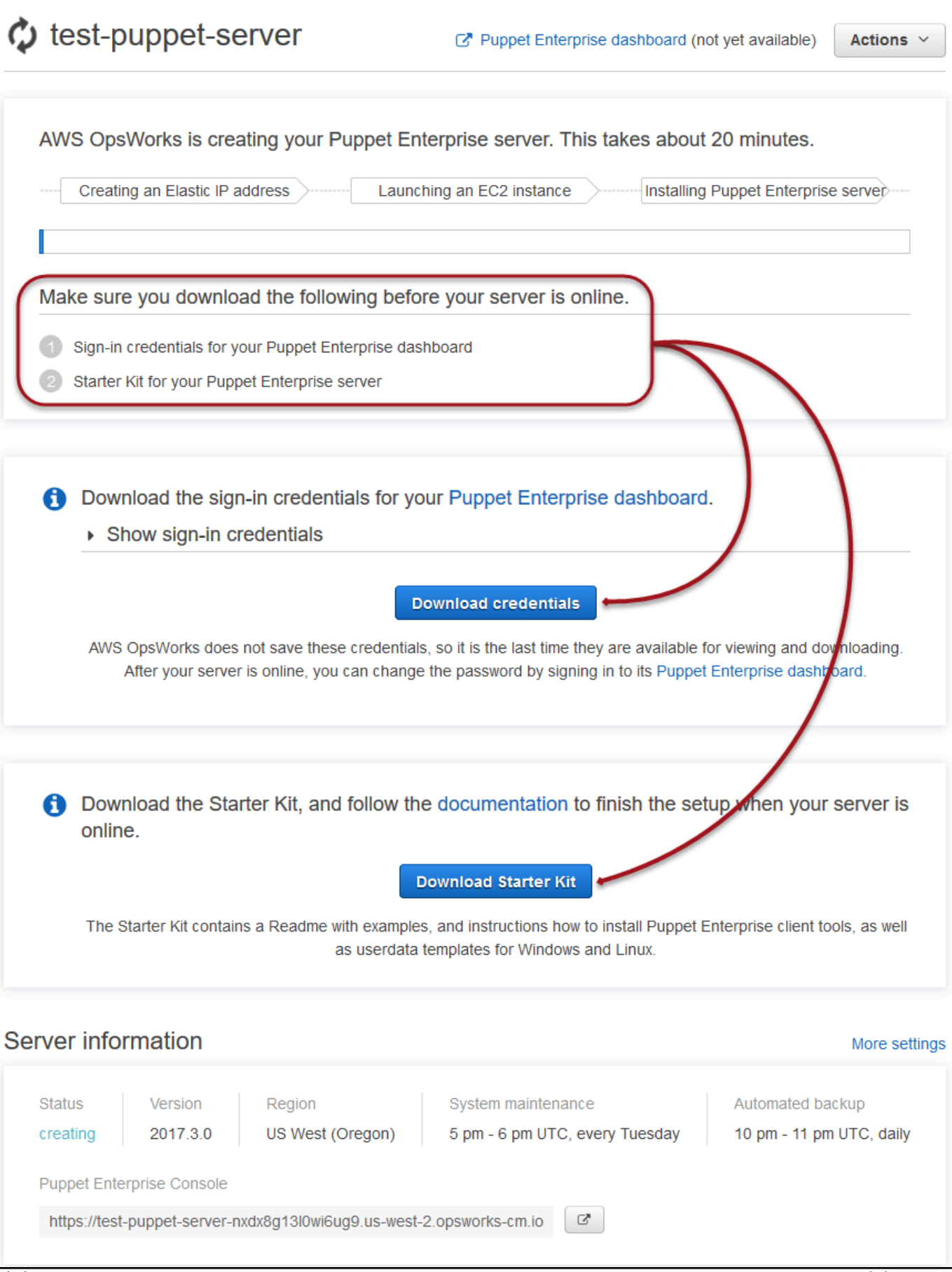
> **Note**
>
> AWS Management Console에서 새 Puppet 마스터 스타터 키트를 다시 생성할 수 없습니다. 를 사용하여 Puppet 마스터를 생성할 때 이전 `jq` 명령을 AWS CLI실행하여 base64로 인코딩된 스타터 키트를 `create-server` 결과에 ZIP 파일로 저장합니다.

8. 사용자 지정 도메인을 사용하지 않는 경우 다음 단계로 이동합니다. 서버에서 사용자 지정 도메인을 사용하는 경우 엔터프라이즈의 DNS 관리 도구에서 CNAME 항목을 만들어 사용자 지정 도메인을 6단계에서 복사한 OpsWorks for Puppet Enterprise 엔드포인트로 지정합니다. 이 단계를 완료해야 사용자 지정 도메인을 사용하여 서버에 연결하거나 로그인할 수 있습니다.
9. 다음 the section called “구성 완료” 단원으로 이동합니다.

스타터 키트를 사용하여 Puppet 마스터 구성

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Puppet 마스터 생성이 아직 진행 중인 상태에서 OpsWorks for Puppet Enterprise 콘솔에서 서버의 속성 페이지가 열립니다. 새 Puppet 마스터를 처음 작업할 때는 속성 페이지에 두 가지 필수 항목을 다운로드하라는 메시지가 표시됩니다. Puppet 서버가 온라인 상태가 되기 전에 이 항목을 다운로드하세요. 새 서버가 온라인 상태가 된 후에는 다운로드 버튼을 사용할 수 없습니다.


test-puppet-server
Puppet Enterprise dashboard (not yet available)
Actions
AWS OpsWorks is creating your Puppet Enterprise server. This takes about 20 minutes.
Creating an Elastic IP address
Launching an EC2 instance
Installing Puppet Enterprise server
Make sure you download the following before your server is online.
1 Sign-in credentials for your Puppet Enterprise dashboard
2 Starter Kit for your Puppet Enterprise server
Download the sign-in credentials for your Puppet Enterprise dashboard.
Show sign-in credentials
Download credentials
AWS OpsWorks does not save these credentials, so it is the last time they are available for viewing and downloading.
After your server is online, you can change the password by signing in to its Puppet Enterprise dashboard.
Download the Starter Kit, and follow the documentation to finish the setup when your server is online.
Download Starter Kit
The Starter Kit contains a Readme with examples, and instructions how to install Puppet Enterprise client tools, as well as userdata templates for Windows and Linux.
Server information
More settings
Status
creating
Version
2017.3.0
Region
US West (Oregon)
System maintenance
5 pm - 6 pm UTC, every Tuesday
Automated backup
10 pm - 11 pm UTC, daily
Puppet Enterprise Console
https://test-puppet-server-nxdx8g13l0wi6ug9.us-west-2.opsworks-cm.io

- Puppet 마스터의 자격 증명으로 로그인. 이러한 자격 증명을 사용하여 대부분의 노드 관리를 수행하는 Puppet Enterprise 콘솔에 로그인합니다.는 이러한 자격 증명을 저장하지 AWS OpsWorks 않습니다.이 자격 증명을 보고 다운로드할 수 있는 마지막 시간입니다. 필요할 경우, 로그인 후 이 자격 증명과 함께 제공되는 암호를 변경할 수 있습니다.
- 스타터 키트.] 스타터 키트에는 설정을 완료하는 방법을 설명하는 정보 및 예제와 Puppet Enterprise 콘솔을 위한 관리자 자격 증명과 함께 README 파일이 포함되어 있습니다. 스타터 키트를 다운로드할 때마다 새 자격 증명이 생성되고 이전 자격 증명은 무효화됩니다.

사전 조건

1. 서버 생성이 계속 진행 중인 상태에서 Puppet 마스터에 대한 로그인 자격 증명을 다운로드하여 안전하고 편리한 위치에 저장합니다.
2. 스타터 키트를 다운로드하고 작업 영역 디렉터리에 스타터 키트 .zip 파일의 압축을 풉니다. 로그인 자격 증명을 누구와도 공유하지 마십시오. 다른 사용자가 Puppet 마스터를 관리하는 경우, 나중에 Puppet Enterprise 콘솔에서 이러한 사용자를 관리자로 추가합니다. Puppet 마스터에 사용자를 추가하는 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 Creating and managing users and user roles 단원을 참조하세요.

Puppet 마스터 인증서 설치

Puppet 마스터에서 작업을 수행하고 관리할 노드를 추가하려면 인증서를 설치해야 합니다. 다음 AWS CLI 명령을 실행하여 설치합니다. 에서는이 작업을 수행할 수 없습니다 AWS Management Console.

```
aws --region region opsworks-cm describe-servers --server-name server_name --query
 "Servers[0].EngineAttributes[?Name=='PUPPET_API_CA_CERT'].Value" --output text
 > .config/ssl/cert/ca.pem
```

단기 토큰 생성

Puppet API를 사용하려면 자체적으로 단기 토큰을 생성해야 합니다. 이 단계에서 Puppet Enterprise 콘솔을 사용할 필요가 없습니다. 다음 명령을 실행하여 토큰을 생성합니다.

기본 토큰 수명은 5분이지만, 이 기본 설정을 변경할 수 있습니다.

```
puppet-access login --config-file .config/puppetlabs/client-tools/puppet-access.conf --
lifetime 8h
```

> **Note**
>
> 기본 토큰 수명이 5분이므로 앞의 예제 명령은 `--lifetime` 파라미터를 추가하여 토큰 수명을 연장합니다. 토큰 수명은 최대 10년까지 설정할 수 있습니다(`10y`). 기본 토크 수명을 변경하는 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 Change the token's default lifetime 단원을 참조하세요.

스타터 키트 Apache 예제 설정

스타터 키트를 다운로드하여 압축을 풀고 나면 샘플로 포함된 `control-repo-example` 폴더에서 예제 브랜치를 사용하여 관리형 노드에 Apache 웹 서버를 구성할 수 있습니다.

스타터 키트에는 `control-repo`와 `control-repo-example` 등 두 개의 `control-repo` 폴더가 포함되어 있습니다. `control-repo` 폴더에는 Puppet GitHub 리포지토리에서 볼 수 있는 것과 변경되지 않은 `production` 브랜치가 포함되어 있습니다. `control-repo-example` 폴더는 테스트 웹 사이트에서 Apache 서버를 설정하기 위한 예제 코드를 포함하는 `production` 브랜치도 가지고 있습니다.

1. `control-repo-example` `production` 브랜치를 Git 원격 작업(Puppet 마스터의`r10k_remote` URL)으로 푸시합니다. 스타터 키트 루트 디렉터리에서 다음을 실행하여 *`r10kRemoteUrl`*을 `r10k_remote` URL로 바꿉니다.

   ```
   cd  control-repo-example
   git remote add origin r10kRemoteUrl
   git push origin production
   ```

   Puppet Code Manager는 Git 브랜치를 환경으로 사용합니다. 기본적으로 모든 노드는 프로덕션 환경에 있습니다.

   > **Important**
   >
   > `master` 브랜치로 푸시하지 마십시오. `master` 브랜치는 Puppet 마스터용으로 예약되어 있습니다.

2. `control-repo-example` 브랜치의 코드를 Puppet 마스터에 배포합니다. 이렇게 하면 Puppet 마스터가 Git 리포지토리에서 Puppet 코드(`r10k_remote`)를 다운로드합니다. 스타터 키트 루트 디렉터리에서 다음을 실행합니다.

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

Amazon EC2에서 생성되는 관리형 노드에 샘플 Apache 구성을 적용할 수 있는 방법에 대한 자세한 정보는 2단계: 무인 연결 스크립트를 사용하여 인스턴스 생성 섹션을 참조하세요.

Puppet 마스터가 관리할 노드 추가

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

주제

- associateNode() API 호출 실행
- 온프레미스 노드 추가 고려 사항
- 추가 정보

노드를 추가하는 권장 방법은 API를 AWS OpsWorks `associateNode()` 사용하는 것입니다. Puppet Enterprise 마스터 서버는 노드가 온프레미스 실제 컴퓨터이든 가상 머신이든 관계 없이, 관리하려는 노드에 Puppet 에이전트 소프트웨어를 설치하는 데 사용하는 리포지토리를 호스팅합니다. 일부 운영 체제의 경우, 시작 프로세스의 일환으로 Puppet 에이전트 소프트웨어가 OpsWorks for Puppet Enterprise 서버에 설치됩니다. 다음 표에는 시작 시 OpsWorks for Puppet Enterprise 서버에서 사용할 수 있는 운영 체제 에이전트가 나와 있습니다.

사전 설치된 운영 체제 에이전트

지원되는 운영 체제	버전
Ubuntu	16.04, 18.04, 20.04
Red Hat Enterprise Linux(RHEL)	6, 7, 8

지원되는 운영 체제	버전
Windows	모든 Puppet 지원 Windows 릴리스의 64비트 에디션

다른 운영 체제의 경우, puppet-agent를 서버에 추가할 수 있습니다. 시스템 유지 관리 시에는 시작 이후에 서버에 추가된 에이전트를 삭제한다는 점에 유의하세요. 삭제된 에이전트를 실행 중이던 대부분의 기존 연결 노드는 계속해서 체크인을 하지만, Debian 운영 체제를 실행 중인 노드들은 보고를 중지할 수 있습니다. OpsWorks for Puppet Enterprise 서버에 에이전트 소프트웨어가 사전 설치되지 않은 운영 체제의 경우에는 노드에 수동으로 puppet-agent를 설치하는 것이 좋습니다. 다른 운영 체제를 가진 노드에서 puppet-agent를 서버에서 가용 상태로 만드는 방법에 대한 자세한 설명은 Puppet Enterprise 설명서의 Installing agents 단원을 참조하세요.

EC2 인스턴스 사용자 데이터를 채워서 자동으로 Puppet 마스터에 노드를 연결하는 방법에 대한 자세한 정보는 OpsWorks for Puppet Enterprise에서 자동으로 노드 추가 단원을 참조하세요.

`associateNode()` API 호출 실행

puppet-agent를 설치하여 노드를 추가하고 나면 노드가 OpsWorks for Puppet Enterprise 서버에 인증서 서명 요청(CSR)을 전송합니다. Puppet 콘솔에서 CSR을 볼 수 있으며, 노드 CSR에 대한 자세한 정보는 Puppet Enterprise 설명서의 Managing certificate signing requests 단원을 참조하세요. OpsWorks for Puppet Enterprise associateNode() API 호출을 실행하면 노드 CSR이 처리되고 서버에 노드가 연결됩니다. 다음은에서이 API 호출을 사용하여 단일 노드 AWS CLI 를 연결하는 방법의 예입니다. 노드가 전송하는 PEM 형식 CSR이 필요하며, Puppet 콘솔에서 이를 얻을 수 있습니다.

```
aws opsworks-cm associate-node --server-name "test-puppet-
server" --node-name "node or instance ID" --engine-attributes
 "Name=PUPPET_NODE_CSR,Value='PEM_formatted_CSR_from_the_node'
```

associateNode()를 사용하여 자동으로 노드를 추가하는 방법에 대한 자세한 정보는OpsWorks for Puppet Enterprise에서 자동으로 노드 추가 단원을 참조하세요.

온프레미스 노드 추가 고려 사항

온프레미스 컴퓨터나 가상 머신에 puppet-agent를 설치한 후에는 두 가지 방법 중 하나를 사용하여 OpsWorks for Puppet Enterprise 마스터와 온프레미스 노드를 연결할 수 있습니다.

- 노드에서 AWS SDK, AWS CLI 또는 AWS Tools for PowerShell 설치를 지원하는 경우 노드를 연결하기 위한 권장 방법으로, associateNode() API 호출을 실행할 수 있습니다. 먼저 OpsWorks for

Puppet Enterprise 마스터를 생성할 때 다운로드하는 스타터 키트에는 태그를 사용하여 역할을 노드에 할당하는 방법이 나와 있습니다. CSR에서 신뢰할 수 있는 정보를 지정하여 Puppet 마스터와 노드를 연결하면서 동시에 태그를 적용할 수 있습니다. 예를 들면 스타터 키트에 포함된 데모 제어 리포지토리는 `pp_role` 태그를 사용하여 역할을 Amazon EC2 인스턴스에 할당하도록 구성되어 있습니다. 태그를 신뢰할 수 있는 정보로서 CSR에 추가하는 방법에 대한 자세한 정보는 Puppet 플랫폼 설명서의 Extension requests (permanent certificate data) 단원을 참조하세요.

- 노드가 AWS 관리 또는 개발 도구를 실행할 수 없는 경우에도 비관리형 Puppet Enterprise 마스터에 등록하는 것과 동일한 방식으로 OpsWorks for Puppet Enterprise 마스터에 등록할 수 있습니다. 이 주제에 언급한 바와 같이, `puppet-agent` 설치 시 CSR이 OpsWorks for Puppet Enterprise 마스터로 전송됩니다. 권한 있는 Puppet 사용자는 CSR에 수동으로 서명하거나 Puppet 마스터에 저장된 `autosign.conf` 파일을 편집하여 CSR의 자동 서명을 구성할 수 있습니다. 자동 서명 구성 및 `autosign.conf` 편집에 대한 자세한 정보는 Puppet 플랫폼 설명서의 SSL 구성: 인증서 요청 자동 서명 단원을 참조하세요.

온프레미스 노드를 Puppet 마스터와 연결하거나 Puppet 마스터가 모든 CSR을 허용하게 하려면 Puppet Enterprise 콘솔에서 다음을 따라하세요. 동작을 제어하는 파라미터는 `puppet_enterprise::profile::master::allow_unauthenticated_ca`입니다.

> **Important**
>
> Puppet 마스터의 자체 서명된 CSR 또는 모든 CSR 허용은 보안상의 이유로 권장하지 않습니다. 기본적으로 인증되지 않은 CSR을 허용하면 Puppet 마스터로 하여금 세계에 접근 가능하게 합니다. 기본적으로 인증서 요청 업로드의 활성화는 Puppet 마스터를 DoS(서비스 거부) 공격에 취약하게 만듭니다.

1. Puppet Enterprise 콘솔에 로그인.
2. 구성을 선택하고, 분류 를 선택하고 PE 마스터를 선택하고 마지막으로 구성 탭을 선택하세요.
3. 분류 탭에서 puppet_enterprise::profile::master 클래스를 찾으십시오.
4. allow_unauthenticated_ca 파라미터의 값을 참으로 설정합니다.
5. 변경 내용을 저장합니다. 귀하의 변경 사항은 다음 Puppet 실행 때 적용됩니다. 변경 사항이 적용될 때까지(그리고 온프레미스 노드가 추가될 때까지) 30분을 기다리시거나 PE 콘솔의 실행 섹션에서 수동으로 Puppet을 실행하시면 됩니다.

추가 정보

OpsWorks for Puppet Enterprise 서버 및 Puppet Enterprise 콘솔 기능을 사용하는 방법에 대해 자세히 알아보려면 Learn Puppet 튜토리얼 사이트를 방문하세요.

Puppet Enterprise 콘솔에 로그인

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Puppet 마스터의 속성 페이지에서 로그인 자격 증명을 다운로드하고 서버가 온라인 상태가 되고 나면 Puppet Enterprise 콘솔에 로그인합니다. 이 안내서에서는 모듈을 포함하고 있는 제어 리포지토리를 지정하고 관리할 노드를 하나 이상 추가하라고 설명했습니다. 이렇게 하면 콘솔에서 에이전트와 노드에 대한 정보를 볼 수 있습니다.

Puppet Enterprise 콘솔 웹 페이지에 연결하려고 하면 Puppet 서버를 관리하는 데 사용하는 클라이언트 컴퓨터에 AWS OpsWorks특정 CA 서명 SSL 인증서를 설치할 때까지 브라우저에 인증서 경고가 표시됩니다. 대시보드 웹 페이지로 가기 전에 경고를 보지 않으려면 로그인 전에 SSL 인증서를 설치하세요.

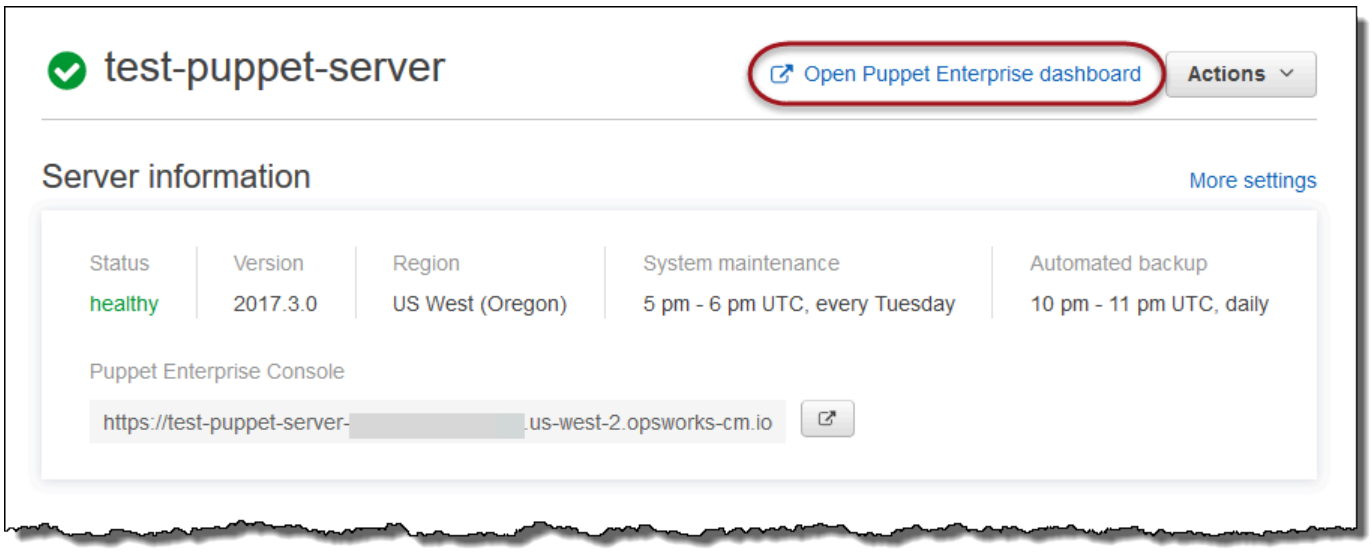
AWS OpsWorks SSL 인증서를 설치하려면

- 시스템과 일치하는 인증서를 선택합니다.

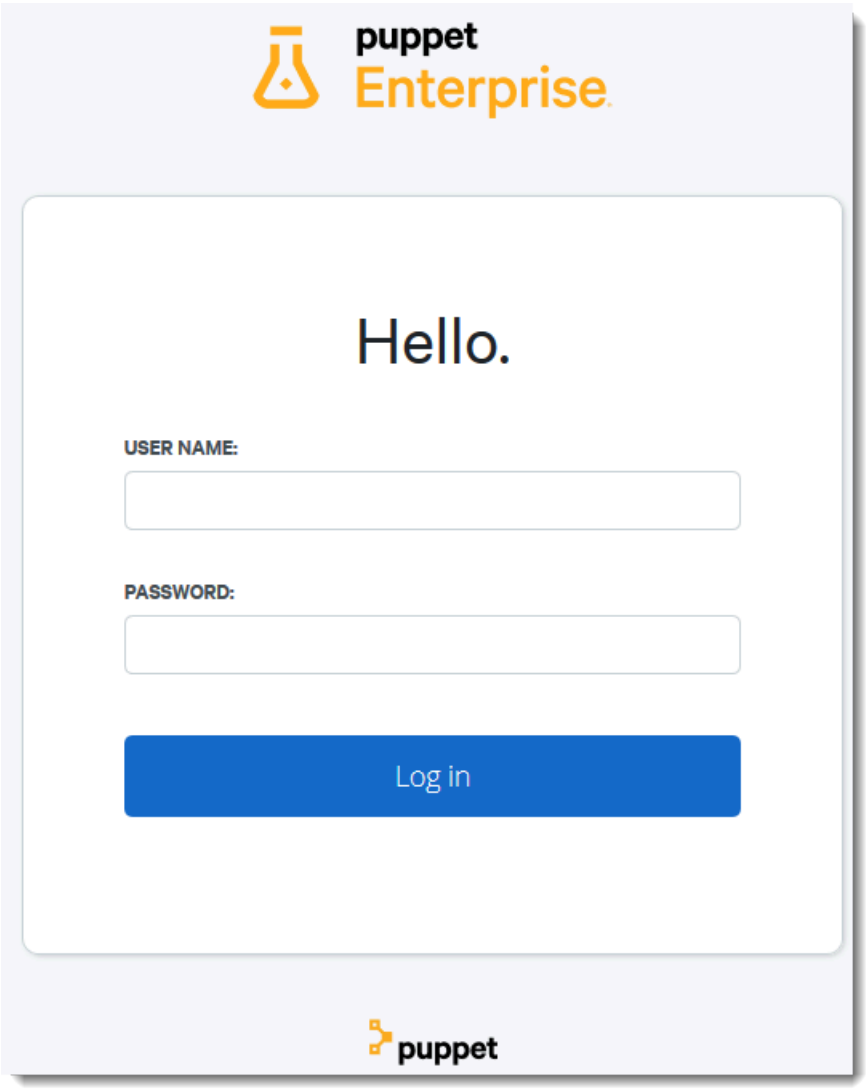
클라이언트 측 SSL 인증서를 설치한 후에는 경고 메시지를 보지 않고 Puppet Enterprise 콘솔에 로그인할 수 있습니다.

Puppet Enterprise 콘솔에 로그인하려면

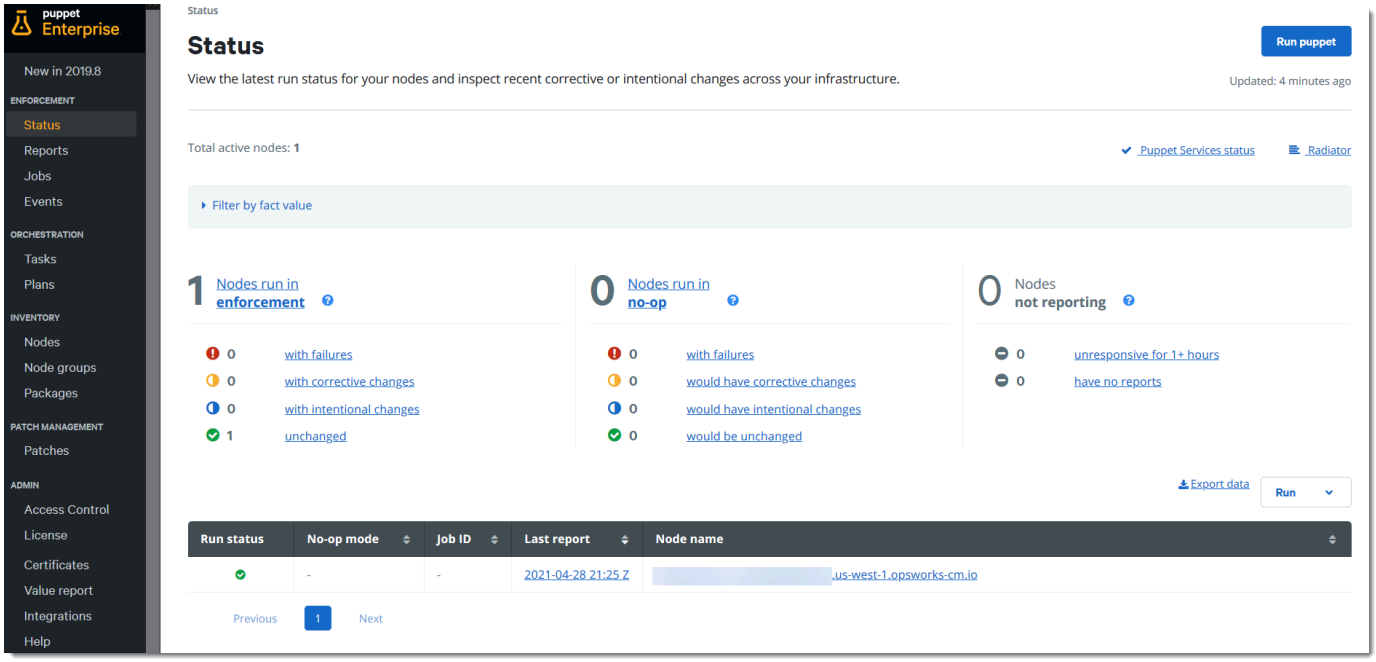
1. 사전 조건에서 다운로드한 Puppet Enterprise 자격 증명의 압축을 풀어 엽니다. 로그인하려면 이 자격 증명이 필요합니다.
2. 에서 Puppet 서버의 속성 페이지를 AWS Management Console엽니다.
3. [속성] 페이지의 오른쪽 위에서 [Puppet Enterprise 콘솔 열기]를 선택합니다.



4. 1단계의 자격 증명으로 로그인합니다.



5. Puppet Enterprise 콘솔에서는 관리 중인 노드에 대한 세부 정보와 모듈 실행 진행 상황 및 이벤트, 노드의 규정 준수 수준 등을 확인할 수 있습니다. Puppet Enterprise 콘솔의 기능과 그 사용 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 노드 관리 섹션을 참조하세요.



노드 그룹화 및 분류

노드에 클래스를 적용하여 원하는 노드 구성을 지정하기 전에 엔터프라이즈에서의 역할과 공통 특성에 따라 노드를 그룹화합니다. 노드의 그룹화 및 분류를 위해서는 다음과 같이 상위 수준의 작업이 필요합니다. PE 콘솔을 사용하여 이러한 작업을 완료할 수 있습니다. 노드를 그룹화 및 분류하는 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 Grouping and classifying nodes 단원을 참조하세요.

1. 노드 그룹을 생성합니다.
2. 생성한 규칙을 적용하여 수동 또는 자동으로 그룹에 노드를 추가합니다.
3. 노드 그룹에 클래스를 할당합니다.

관리자 및 사용자 암호 재설정

Puppet Enterprise 콘솔 로그인에 사용되는 암호를 변경하는 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 콘솔 관리자 암호 재설정 섹션을 참조하세요.

기본적으로 로그인을 10회 시도한 후에는 사용자가 Puppet 콘솔에 액세스할 수 없도록 잠깁니다. 잠금 시 사용자 암호를 재설정하는 방법에 대한 자세한 정보는 Puppet Enterprise 설명서의 Password endpoints 단원을 참조하세요.

선택 사항: Puppet r10k 원격 제어 리포지토리 AWS CodeCommit 로 사용

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

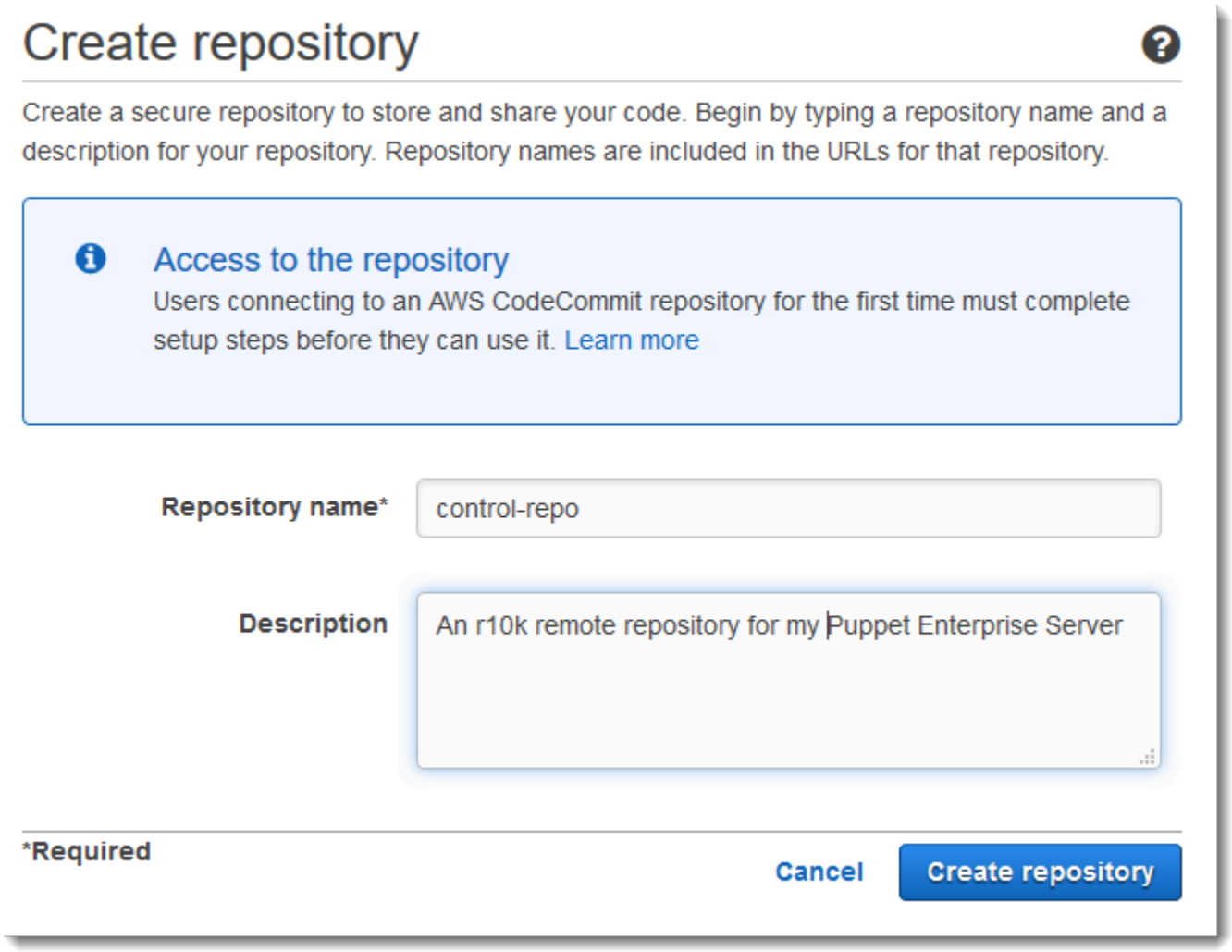
를 사용하여 새 리포지토리를 생성하고 이를 r10k 원격 제어 리포지토리로 AWS CodeCommit 사용할 수 있습니다. 이 섹션의 단계를 완료하고 CodeCommit 리포지토리를 사용하려면 AWSCodeCommitreadonly 관리형 정책에서 제공하는 권한을 가진 사용자가 필요합니다.

주제

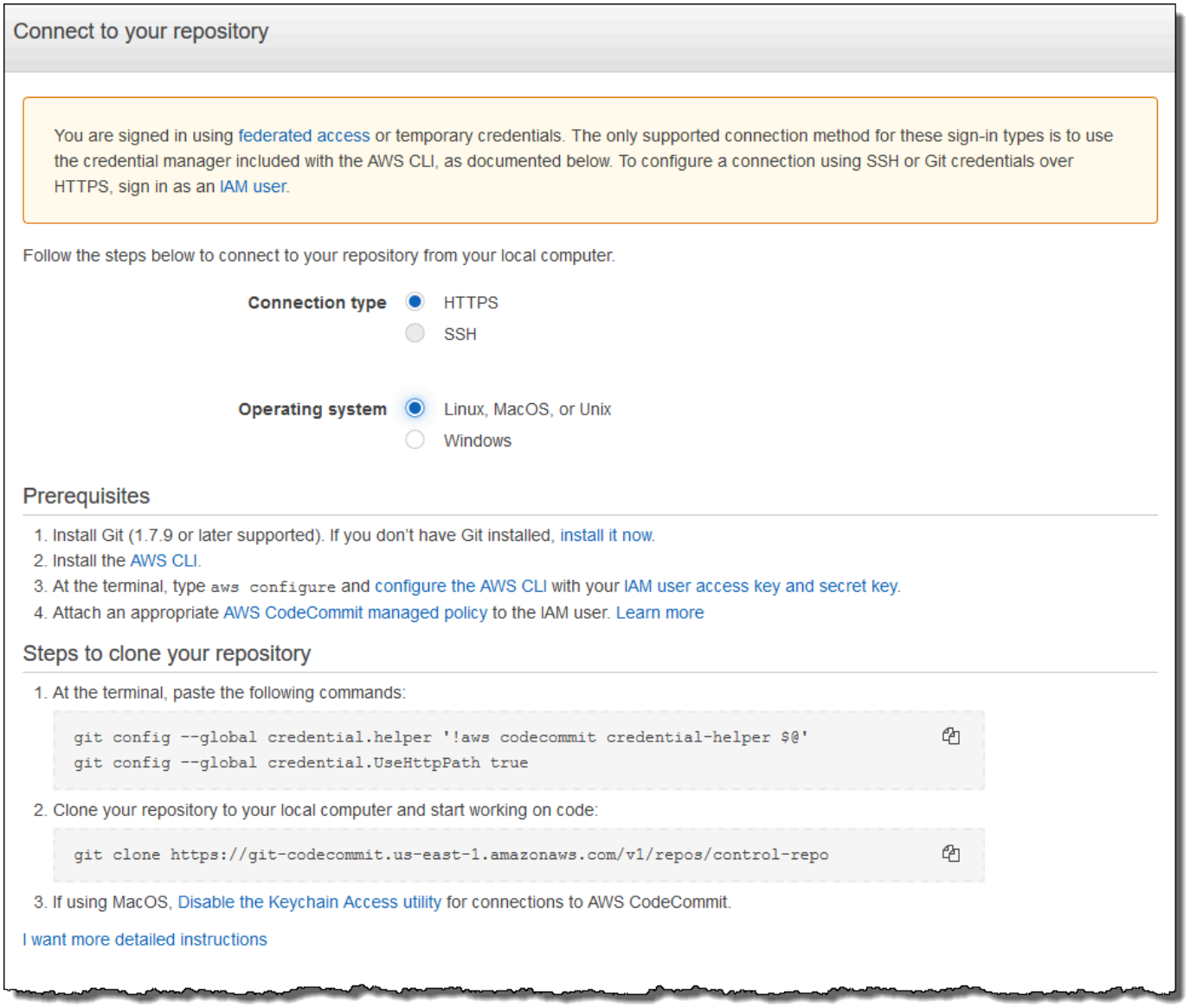
- 1단계: HTTPS 연결 유형에서 CodeCommit를 리포지토리로 사용
- 2단계: (선택 사항) SSH 연결 유형에서 CodeCommit를 리포지토리로 사용

1단계: HTTPS 연결 유형에서 CodeCommit를 리포지토리로 사용

1. CodeCommit 콘솔에서 새 리포지토리를 생성합니다.



2. Amazon SNS 주제 설정을 건너뛰려면 건너뛰기를 선택합니다.
3. [코드] 페이지에서 [리포지토리에 연결]을 선택합니다.
4. 리포지토리에 연결 페이지에서 연결 유형으로 HTTPS를 선택하고 운영 체제를 선택합니다.



리포지토리를 복제하는 단계 영역에서 `git clone` URL은 `https://git-codecommit.`*`region`*`.amazonaws.com/v1/repos/control-repo`와 유사해야 합니다. Puppet 서버 설정에 사용할 수 있도록 이 URL을 편리한 위치에 복사합니다.

5. 리포지토리에 연결 페이지를 닫고 OpsWorks for Puppet Enterprise 서버 설정으로 돌아갑니다.
6. 4단계에서 복사한 URL을 Puppet 마스터 설치 마법사의 자격 증명 구성 페이지에 있는 r10k 원격 문자열 상자에 붙여넣습니다. r10k 프라이빗 키 상자를 비워둡니다. Puppet 마스터의 생성 및 시작을 완료합니다.
7. IAM 콘솔에서 Puppet 마스터의 인스턴스 프로파일 역할에 AWSCodeCommitReadOnly 정책을 연결합니다. 권한 정책을 IAM 역할에 추가하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

8. AWS CodeCommit 사용 설명서의 Git 자격 증명을 사용하는 HTTPS 사용자용 설정의 단계에 따라 기존 `control-repo` 콘텐츠를 새 CodeCommit 리포지토리로 푸시하세요.
9. 이제, the section called “구성 완료”의 지침에 따라 계속 진행하고 스타터 키트를 사용하여 Puppet 마스터에 코드를 배포할 수 있습니다. 다음 명령은 예제입니다.

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

2단계: (선택 사항) SSH 연결 유형에서 CodeCommit를 리포지토리로 사용

SSH 키 페어 인증을 사용하도록 AWS CodeCommit r10k 원격 제어 리포지토리를 구성할 수 있습니다. 이 절차를 시작하기 전에 다음 사전 조건을 완료해야 합니다.

- 앞의 the section called “1단계: HTTPS 연결 유형에서 CodeCommit를 리포지토리로 사용” 단원에서 설명한 것과 같이 HTTPS 제어 리포지토리가 있는 OpsWorks for Puppet Enterprise 서버를 시작해야 합니다. 필수 구성을 Puppet 마스터에 업로드하려면 먼저 이 작업을 완료해야 합니다.
- AWSCodeCommitreadOnly 관리형 정책을 사용하는 사용자가 연결되어 있는지 확인하세요. 사용자를 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.
- SSH 키를 생성하여 사용자와 연결합니다. AWS CodeCommit 사용 설명서의 3단계: Linux, macOS 또는 Unix에서 자격 증명 구성의 `ssh-keygen`을 사용하여 퍼블릭/프라이빗 키 페어를 생성하는 방법에 대한 지침을 따르세요.

1. AWS CLI 세션에서 다음 명령을 실행하여 프라이빗 키 파일 콘텐츠를 AWS Systems Manager 파라미터 스토어에 업로드합니다. OpsWorks for Puppet Enterprise 서버는 이 파라미터를 쿼리하여 필수 인증서 파일을 가져옵니다. *`private_key_file`*을 SSH 프라이빗 키 파일의 경로로 바꿉니다.

```
aws ssm put-parameter --name puppet_user_pk --type String --value
 "`cat private_key_file`"
```

2. Systems Manager 파라미터 스토어 권한을 Puppet 마스터에 추가합니다.
   a. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
   b. 왼쪽 탐색 창에서 역할을 선택합니다.
   c. aws-opsworks-cm-ec2-role을 선택합니다.
   d. 권한 탭에서 정책 연결을 선택합니다.

e. 검색 창에 **AmazonSSMManagedInstanceCore**를 입력합니다.

f. 검색 결과에서 AmazonSSMManagedInstanceCore를 선택합니다.

g. 정책 연결을 선택합니다.

3. 구성 파일 매니페스트를 생성합니다. 스타터 키트에서 제공하는 `control-repo-example` 리포지토리를 사용하는 경우 예제 리포지토리에 표시된 위치에서 다음 파일을 생성합니다. 그렇지 않으면 자체적인 제어 리포지토리 구조에 따라 해당 파일을 생성합니다. *IAM_USER_SSH_KEY* 값을 이 절차의 사전 조건에서 생성한 SSH 키 ID로 바꿉니다.

```
control-repo-example/site/profile/manifests/codecommit.pp
```

```
class profile::codecommit {
  $configfile = @(CONFIGFILE)
      Host git-codecommit.*.amazonaws.com
      User IAM_USER_SSH_KEY
      IdentityFile /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      StrictHostKeyChecking=no
      | CONFIGFILE

  # Replace REGION with the correct region for your server.
  $command = @(COMMAND)
      aws ssm get-parameters \
      --region REGION \
      --names puppet_user_pk \
      --query "Parameters[0].Value" \
      --output text >| /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      | COMMAND

  $dirs = [
            '/opt/puppetlabs/server/data/puppetserver/.ssh',
            '/etc/puppetlabs/puppetserver/ssh',
          ]

  file { $dirs:
    ensure => 'directory',
    group  => 'pe-puppet',
    owner  => 'pe-puppet',
    mode   => '0750',
  }

  file { 'ssh-config':
```

```
    path    => '/opt/puppetlabs/server/data/puppetserver/.ssh/config',
    require => File[$dirs],
    content => $configfile,
    group   => 'pe-puppet',
    owner   => 'pe-puppet',
    mode    => '0600',
  }

  exec { 'download-codecommit-certificate':
    command => $command,
    require => File[$dirs],
    creates => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    path    => '/bin',
    cwd     => '/etc/puppetlabs',
  }

  file { 'private-key-permissions':
    subscribe => Exec['download-codecommit-certificate'],
    path      => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    group     => 'pe-puppet',
    owner     => 'pe-puppet',
    mode      => '0600',
  }
}
```

4. 제어 리포지토리를 CodeCommit로 푸시합니다. 다음 명령을 실행하여 새 매니페스트 파일을 리포지토리로 푸시합니다.

```
git add ./site/profile/manifests/codecommit.pp
git commit -m 'Configuring for SSH connection to CodeCommit'
git push origin production
```

5. 매니페스트 파일을 배포합니다. 다음 명령을 실행하여 업데이트된 구성을 OpsWorks for Puppet Enterprise 서버에 배포합니다. *STARTER_KIT_DIRECTORY*를 Puppet 구성 파일의 경로로 바꿉니다.

```
cd STARTER_KIT_DIRECTORY

puppet-access login --config-file .config/puppetlabs/client-tools/puppet-
access.conf

puppet-code deploy --all --wait \
--config-file .config/puppet-code.conf \
```

```
--token-file .config/puppetlabs/token
```

6. OpsWorks for Chef Automate 서버 분류를 업데이트합니다. 기본적으로 Puppet 에이전트는 노드(마스터 포함)에서 30분마다 실행됩니다. 기다리지 않으려면 Puppet 마스터에서 에이전트를 수동으로 실행할 수 있습니다. 에이전트를 실행하면 새 매니페스트 파일이 선택됩니다.

   a. Puppet Enterprise 콘솔에 로그인.

   b. 분류을 선택합니다.

   c. PE 인프라를 확장합니다.

   d. PE 마스터를 선택합니다.

   e. 구성 탭에서 새 클래스 추가에 **`profile::codecommit`**을 입력합니다.

      새 클래스인 `profile::codecommit`는 `puppet-code deploy`를 실행한 후 즉시 나타나지 않을 수 있습니다. 새 클래스가 나타나지 않으면 이 페이지에서 새로 고침을 선택합니다.

   f. 클래스 추가를 선택한 다음 커밋 1 변경을 선택합니다.

   g. OpsWorks for Puppet Enterprise 서버에서 Puppet 에이전트를 수동으로 실행합니다. 노드를 선택하고, 목록에서 서버를 선택한 다음, Puppet 실행을 선택하고, 실행을 선택합니다.

7. Puppet Enterprise 콘솔에서 HTTPS 대신 SSH를 사용하도록 리포지토리 URL을 변경합니다. 이러한 단계에서 수행하는 구성은 OpsWorks for Puppet Enterprise 백업 및 복원 프로세스 중에 저장되므로, 유지 관리 활동 후 리포지토리 구성을 수동으로 변경할 필요가 없습니다.

   a. 분류를 선택합니다.

   b. PE 인프라를 확장합니다.

   c. PE 마스터를 선택합니다.

   d. 구성 탭에서 `puppet_enterprise::profile::master` 클래스를 찾습니다.

   e. `r10k_remote` 매개 변수 옆의 편집을 선택합니다.

   f. HTTPS URL을 해당 리포지토리의 SSH URL로 바꾼 다음, 커밋 1 변경을 선택합니다.

   g. OpsWorks for Puppet Enterprise 서버에서 Puppet 에이전트를 수동으로 실행합니다. 노드를 선택하고, 목록에서 서버를 선택한 다음, Puppet 실행을 선택하고, 실행을 선택합니다.

를 사용하여 AWS OpsWorks for Puppet Enterprise 마스터 생성 AWS CloudFormation

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 서비스 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks for Puppet Enterprise 를 사용하면에서 Puppet Enterprise 서버를 실행할 수 있습니다 AWS. 약 15분 안에 Puppet Enterprise 마스터를 프로비저닝할 수 있습니다.

2021년 5월 3일부터 Puppet Enterprise용 OpsWorks는 일부 Puppet Enterprise 서버 속성을 AWS Secrets Manager에 저장합니다. 자세한 내용은 와 통합 AWS Secrets Manager 단원을 참조하십시오.

다음 연습에서는 AWS CloudFormation에 스택을 생성하여 OpsWorks for Puppet Enterprise에 Puppet 마스터를 만들어 봅니다.

주제

- 사전 조건
- AWS CloudFormation에서 Puppet Enterprise 마스터 만들기

사전 조건

Puppet 마스터를 새로 만들기 전에 먼저 Puppet 마스터에 액세스하고 관리하는 데 필요한 리소스를 OpsWorks for Puppet Enterprise 외부에서 생성하세요. 자세한 내용은 이 설명서의 시작하기 단원에서 사전 조건를 참조하세요.

사용자 지정 도메인을 사용하는 서버를 만드는 경우 사용자 지정 도메인, 인증서 및 프라이빗 키가 필요합니다. AWS CloudFormation 템플릿에서 이러한 세 파라미터 모두에 값을 지정해야 합니다. `CustomDomain`, `CustomCertificate` 및 `CustomPrivateKey` 매개 변수의 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer 항목을 참조하세요.

AWS CloudFormation 사용 설명서 템플릿 참조의 OpsWorks-CM 섹션을 검토하여 서버를 생성하는 데 사용하는 AWS CloudFormation 템플릿에서 지원되는 값과 필요한 값에 대해 알아봅니다.

AWS CloudFormation에서 Puppet Enterprise 마스터 만들기

이 섹션에서는 AWS CloudFormation 템플릿을 사용하여 OpsWorks for Puppet Enterprise 마스터 서버를 생성하는 스택을 빌드하는 방법을 설명합니다. AWS CloudFormation 콘솔 또는를 사용하여이 작업을 수행할 수 있습니다 AWS CLI. 예제 AWS CloudFormation 템플릿을 사용하여 OpsWorks for Puppet Enterprise 서버 스택을 빌드할 수 있습니다. 예제 템플릿을 서버 이름, IAM 역할, 인스턴스 프로파일, 서버 설명, 백업 보존 수, 유지 관리 옵션 및 옵션 태그로 업데이트하세요. 서버에서 사용자 정의 도메인을 사용하는 경우 AWS CloudFormation 템플릿에서 `CustomDomain`, `CustomCertificate` 및 `CustomPrivateKey` 파라미터에 대한 값을 지정해야 합니다. 이러한 옵션에 대한 자세한 내용은 이 설명서의 시작하기 단원에서the section called “를 사용하여 Puppet Enterprise Master 생성 AWS Management Console” 항목을 참조하세요.

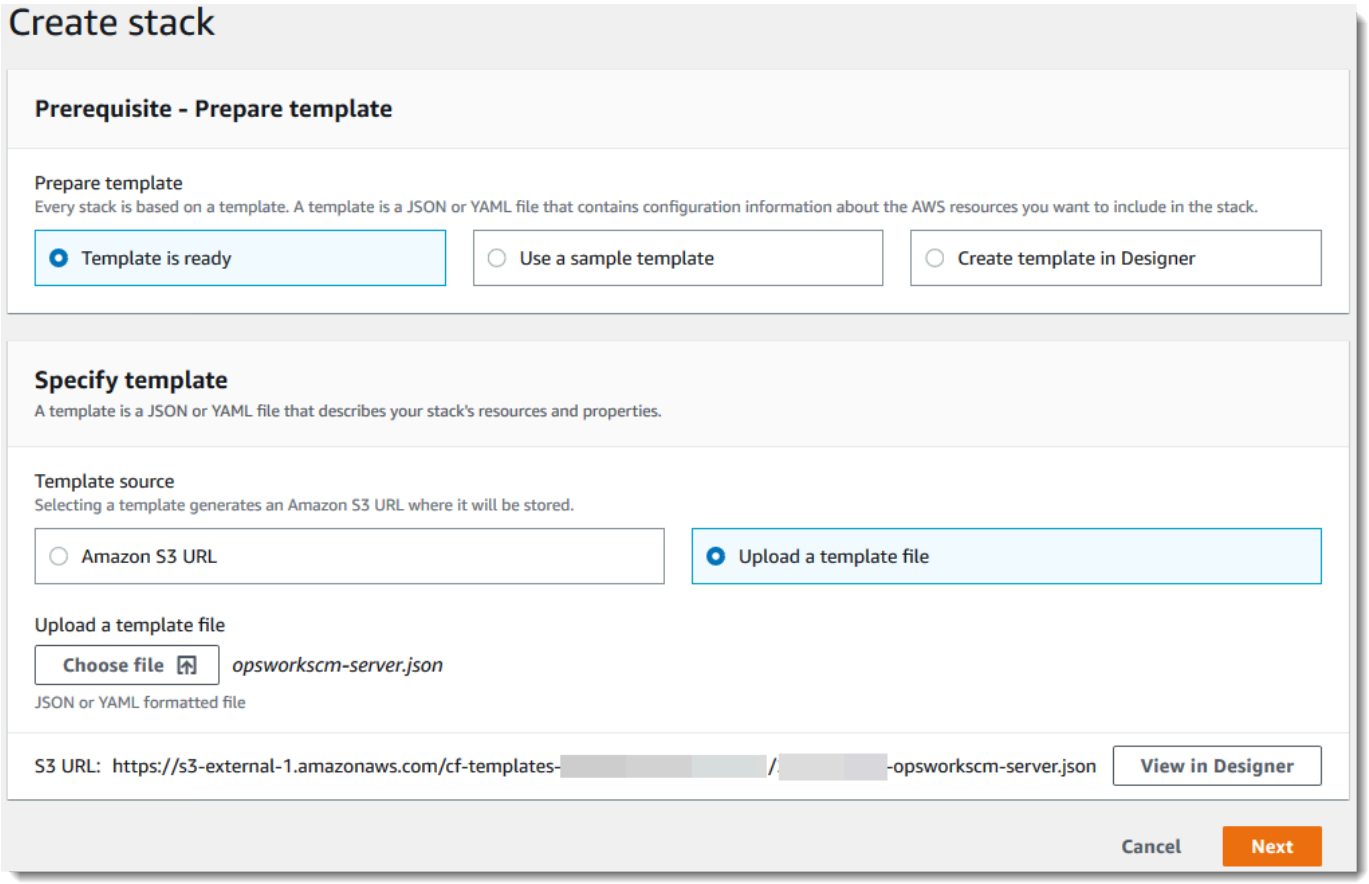
주제

- 를 사용하여 Puppet Enterprise Master 생성 AWS CloudFormation (콘솔)
- 를 사용하여 Puppet Enterprise Master 생성 AWS CloudFormation (CLI)

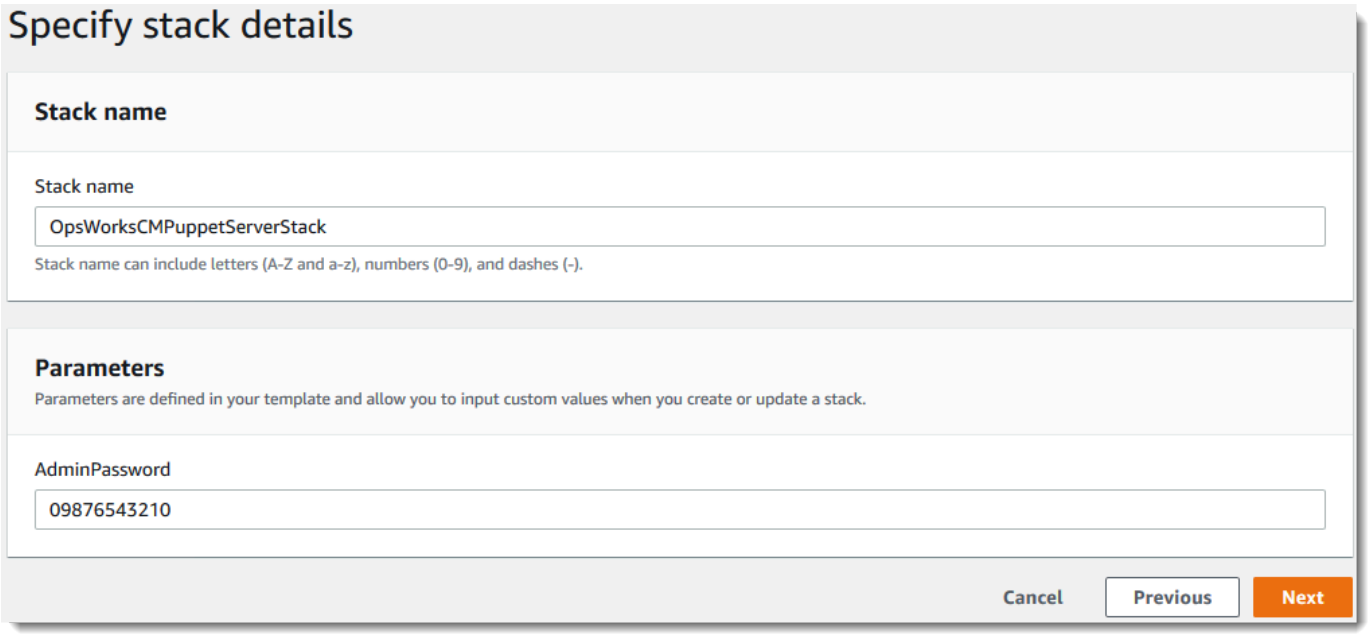
를 사용하여 Puppet Enterprise Master 생성 AWS CloudFormation (콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation AWS CloudFormation 콘솔을 엽니다.
2. AWS CloudFormation 홈 페이지에서 스택 생성을 선택합니다.
3. 사전 조건 - 템플릿 준비에서 예제 AWS CloudFormation 템플릿을 사용하고 있는 경우 템플릿 준비가 완료되었습니다를 선택합니다.
4. 템플릿 지정에서 템플릿의 소스를 선택합니다. 이 연습에서는 템플릿 파일 업로드를 선택하고 Puppet Enterprise 서버를 생성하는 AWS CloudFormation 템플릿을 업로드합니다. 템플릿 파일을 찾은 후 다음을 선택합니다.

   AWS CloudFormation 템플릿은 YAML 또는 JSON 형식일 수 있습니다. 예제 AWS CloudFormation 템플릿을 사용할 수 있습니다. 예제 값을 자신의 값으로 바꿔야 합니다. AWS CloudFormation 템플릿 디자이너를 사용하여 새 템플릿을 빌드하거나 기존 템플릿을 검증할 수 있습니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 디자이너 인터페이스 개요를 참조하세요.



5. 스택 세부 정보 지정 페이지에서 스택의 이름을 입력합니다. 이것은 서버 이름이 아니고 스택 이름입니다. 파라미터에서 Puppet Enterprise 콘솔 웹 페이지에 로그인하기 위한 관리자 암호를 입력합니다. 암호는 ASCII 문자 8~32자로 되어 있어야 합니다. 다음을 선택합니다.



6. 옵션 페이지에서, 스택을 사용하여 생성할 서버에 태그를 추가하고, 템플릿에서 사용할 IAM 역할을 아직 지정하지 않은 경우 리소스 생성을 위한 IAM 역할을 선택할 수 있습니다. 옵션 지정을 마쳤으면 다음을 선택합니다. 롤백 트리거와 같은 고급 옵션에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 스택 옵션 설정을 참조하세요.
7. [검토] 페이지에서 선택 사항을 검토합니다. 서버 스택을 생성할 준비가 되면 생성을 선택합니다.

   가 스택 AWS CloudFormation 을 생성할 때까지 기다리는 동안 스택 생성 상태를 확인합니다. 스택 생성이 실패하면 콘솔에 표시된 오류 메시지를 검토하여 문제를 해결할 수 있습니다. AWS CloudFormation 스택의 오류 문제 해결에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 오류 문제 해결을 참조하세요.

   서버 생성이 끝나면 OpsWorks for Puppet Enterprise 홈 페이지에서 Puppet 마스터를 확인할 수 있으며, 상태는 온라인입니다. 서버가 온라인 상태가 되면 `https://your_server_name-randomID.region.opsworks-cm.io` 형식의 URL을 가진 서버 도메인에서 Puppet Enterprise 콘솔을 사용할 수 있습니다.

   > **Note**
   >
   > 서버에 대한 사용자 지정 도메인, 인증서 및 프라이빗 키를 지정한 경우 OpsWorks for Puppet Enterprise에서 서버에 대해 자동으로 생성한 엔드포인트에 사용자 지정 도메인을 매핑하는 엔터프라이즈 DNS 관리 도구에 CNAME 항목을 만듭니다. 생성된 엔드포인트를 사용자 정의 도메인 값에 매핑할 때까지 서버를 관리하거나 서버의 Puppet Enterprise 관리 웹 사이트에 연결할 수 없습니다.
   > 생성된 엔드포인트 값을 가져오려면 서버가 온라인 상태가 된 후 다음 AWS CLI 명령을 실행합니다.
   >
   > ```
   > aws opsworks describe-servers --server-name server_name
   > ```

를 사용하여 Puppet Enterprise Master 생성 AWS CloudFormation (CLI)

로컬 컴퓨터에서 아직를 실행하지 않는 경우 AWS Command Line Interface 사용 설명서의 설치 지침에 AWS CLI 따라를 AWS CLI다운로드하고 설치합니다. 이 단원에서는 create-stack 명령과 함께 사용할 수 있는 모든 파라미터를 다 설명하지는 않습니다. create-stack 파라미터에 대한 자세한 내용은 AWS CLI 레퍼런스의 create-stack를 참조하세요.

1. OpsWorks for Puppet Enterprise 마스터를 만들기 위한 사전 조건을(를) 완료해야 합니다.

2. 서비스 역할과 인스턴스 프로파일을 생성합니다.

   계정에서 서비스 역할의 ARNs 찾아 복사합니다.

```
aws iam list-roles --path-prefix "/service-role/" --no-paginate
```

   `list-roles` 명령의 결과에서 다음과 비슷한 서비스 역할 및 인스턴스 프로파일을 찾아봅니다. 서비스 역할 및 인스턴스 프로파일의 ARNs을 기록해 두고 Puppet 마스터 서버 스택을 생성하는 데 사용하는 AWS CloudFormation 템플릿에 추가합니다.

```
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
```

```
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. create-stack 명령을 다시 실행하여 OpsWorks for Puppet Enterprise 마스터를 생성합니다.

   - *stack_name*을 스택 이름으로 바꿉니다. 이것은 Puppet 마스터가 아닌 AWS CloudFormation 스택의 이름입니다. Puppet 마스터 이름은 AWS CloudFormation 템플릿의 `ServerName` 값입니다.
   - *template*을 해당 템플릿 파일의 경로로 바꾸고 *yaml ## json* 확장명을 `.yaml` 또는 `.json`으로 적절히 바꿉니다.
   - `--parameters`의 값은 CreateServer API의 엔진 속성에 해당합니다. Puppet의 경우에는 서버를 만들기 위해 다음과 같은 엔진 속성을 사용자가 제공합니다. r10k 엔진 속성은 Puppet 마스터를 코드 리포지토리에 연결하여 서버의 환경 구성을 관리합니다. r10k 엔진 속성에 대한 자세한 내용은 Puppet Enterprise 설명서에서 Managing code with r10k를 참조하세요.
     - `PUPPET_ADMIN_PASSWORD` - Puppet Enterprise 콘솔 웹 페이지에 로그인하기 위한 관리자 암호입니다. 암호는 8-32자의 ASCII 문자여야 하며 대문자, 소문자, 숫자, 특수 문자를 한 개 이상 포함해야 합니다.
     - `PUPPET_R10K_REMOTE` - 제어 리포지토리의 URL입니다(예: ssh://git@your.git-repo.com:user/control-repo.git). r10k 원격을 지정하면 TCP 포트 8170이 열립니다.
     - `PUPPET_R10K_PRIVATE_KEY`. 프라이빗 Git 리포지토리를 사용할 경우 PUPPET_R10K_PRIVATE_KEY를 추가하여 SSH URL과 PEM 인코딩 형식의 프라이빗 SSH 키를 지정합니다.

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=AdminPassword,ParameterValue="password"
```

다음은 예입니다.

```
aws cloudformation create-stack --stack-name "OpsWorksCMPuppetServerStack"
 --template-body file://opsworkscm-puppet-server.json --parameters
 ParameterKey=AdminPassword,ParameterValue="09876543210Ab#"
```

다음 예제에서는 AWS CloudFormation 템플릿에 제공되지 않은 r10k 엔진 속성을 파라미터로 지정합니다. r10k 엔진 속성을 포함하는 예제 템플릿인 `puppet-server-param-attributes.yaml`은 예제 AWS CloudFormation 템플릿에 포함되어 있습니다.

```
aws cloudformation create-stack --stack-name MyPuppetStack --
template-body file://puppet-server-param-attributes.yaml --parameters
 ParameterKey=AdminPassword,ParameterValue="superSecret1%3"
 ParameterKey=R10KRemote,ParameterValue="https://www.yourRemote.com"
 ParameterKey=R10KKey,ParameterValue="$(cat puppet-r10k.pem)"
```

다음 예제는 r10k 엔진 속성과 그 값을 AWS CloudFormation 템플릿에 지정합니다. 명령에서 템플릿 파일만 가리키면 됩니다. `--template-body`의 값으로 지정된 템플릿인 `puppet-server-in-file-attributes.yaml`은 예제 AWS CloudFormation 템플릿에 포함되어 있습니다.

```
aws cloudformation create-stack --stack-name MyPuppetStack --template-body file://
puppet-server-in-file-attributes.yaml
```

4. (선택 사항) 스택 생성 상태를 확인하려면 다음 명령을 실행합니다.

```
aws cloudformation describe-stacks --stack-name stack_name
```

5. 스택 생성을 완료했으면 다음 the section called “구성 완료” 단원으로 진행하세요. 스택 생성이 실패하면 콘솔에 표시된 오류 메시지를 검토하여 문제를 해결할 수 있습니다. AWS CloudFormation 스택의 오류 문제 해결에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 오류 문제 해결을 참조하세요.

OpsWorks for Puppet Enterprise 서버를 사용자 지정 도메인을 사용하도록 업데이트

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 서버의 백업을 사용하여 새 서버를 만들어서 사용자 지정 도메인과 인증서를 사용하도록 기존 OpsWorks for Puppet Enterprise 서버를 업데이트하는 방법을 설명합니다. 기본적으로 백업에서 새 서버를 만든 다음 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하도록 새 서버를 구성하여 기존 =OpsWorks for Puppet Enterprise 2.0 서버를 복사합니다.

주제

- 사전 조건
- 제한 사항
- 사용자 지정 도메인을 사용하도록 서버 업데이트
- 참고

사전 조건

다음은 사용자 지정 도메인과 인증서를 사용하도록 기존 OpsWorks for Puppet Enterprise 서버를 업데이트하는 데 있어서의 요구 사항입니다.

- 업데이트(또는 복사)하려는 서버에서 Puppet Enterprise 2019.8.5를 실행 중이어야 합니다.
- 새 서버를 만드는 데 사용할 백업을 결정합니다. 업데이트하고자 하는 서버에 사용 가능한 백업이 하나 이상 있어야 합니다. OpsWorks for Puppet Enterprise 백업에 대한 자세한 정보는 OpsWorks for Puppet Enterprise 서버 백업 섹션을 참조하세요.
- 백업 소스인 기존 서버를 만드는 데 사용한 서비스 역할 및 인스턴스 프로파일 ARN을 준비합니다.
- AWS CLI의 최신 릴리스를 실행 중이어야 합니다. AWS CLI 도구 업데이트에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 설치를 AWS CLI 참조하세요.

제한 사항

백업에서 새 서버를 생성하여 기존 서버를 업데이트하는 경우 새 서버는 기존 OpsWorks for Puppet Enterprise 서버와 정확히 동일할 수 없습니다.

- AWS CLI 또는 AWS SDKs. AWS Management Console사용을 통해 백업에서 새 서버를 생성할 수 없습니다.
- 새 서버는 계정 내, 그리고 AWS 리전 내에 있는 기존 서버와 동일한 이름을 사용할 수 없습니다. 이름은 백업 소스로 사용한 기존 서버와 달라야 합니다.
- 기존 서버에 연결된 노드는 새 서버에서 관리하지 않습니다. 다음 중 하나를 수행해야 합니다.
  - 둘 이상의 Puppet 마스터에서 노드를 관리할 수 없으므로 다른 노드를 연결합니다.

- 기존 서버(백업 소스)에서 새 서버 및 새 사용자 지정 도메인 엔드포인트로 노드를 마이그레이션합니다. 노드를 마이그레이션하는 방법에 대한 자세한 내용은 Puppet Enterprise 설명서를 참조하세요.

사용자 지정 도메인을 사용하도록 서버 업데이트

기존 Puppet 마스터를 업데이트하려면 `create-server` 명령을 실행하고 백업, 사용자 지정 도메인, 사용자 지정 인증서 및 사용자 지정 프라이빗 키를 지정하는 파라미터를 추가하여 서버의 복사본을 만듭니다.

1. `create-server` 명령에서 지정할 수 있는 서비스 역할 또는 인스턴스 프로파일 ARN이 없는 경우 를 사용하여 Chef Automate 서버 생성 AWS CLI의 1~5단계를 수행하여 사용할 수 있는 서비스 역할 및 인스턴스 프로필을 만듭니다.
2. 아직 수행하지 않은 경우 사용자 지정 도메인이 있는 새 서버의 기반이 될 기존 Puppet 마스터의 백업을 찾습니다. 계정 및 리전의 모든 OpsWorks for Puppet Enterprise 백업에 대한 정보를 표시하려면 다음 명령을 실행합니다. 사용할 백업 ID를 적어 둡니다.

   ```
   aws opsworks-cm --region region name describe-backups
   ```

3. `create-server` 명령을 실행하여 OpsWorks for Puppet Enterprise 서버를 생성합니다.

   - `--engine` 값은 `Puppet`이고, `--engine-model` 값은 `Monolithic`, `--engine-version` 값은 `2019` 또는 `2017`입니다.
   - 서버 이름은 AWS 계정 내에서 각 리전 내에서 고유해야 합니다. 서버 이름은 문자로 시작해야 하며, 그 이후에는 문자, 숫자 또는 하이픈(-)을 사용할 수 있으며, 최대 길이는 40자입니다.
   - 3단계와 4단계에서 복사해 둔 인스턴스 프로파일 ARN 및 서비스 역할 ARN을 사용합니다.
   - 유효한 인스턴스 유형은 `c4.large`, `c4.xlarge` 또는 `c4.2xlarge`입니다. 인스턴스 유형의 사양에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하세요.
   - `--engine-attributes` 파라미터는 선택 사항이며, Puppet 관리자 암호를 지정하지 않을 경우 서버 생성 프로세스에서 해당 값이 자동으로 생성됩니다. `--engine-attributes`를 추가하는 경우 Puppet Enterprise 콘솔 웹 페이지에 로그인하기 위한 관리자 암호인 `PUPPET_ADMIN_PASSWORD`를 지정합니다. 암호는 ASCII 문자 8~32자로 되어 있어야 합니다.
   - SSH 키 페어는 선택 사항이지만, 콘솔 관리자 암호를 재설정해야 하는 경우 Puppet 마스터에 연결하는 데 도움이 될 수 있습니다. SSH 키 페어 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

- 사용자 지정 도메인을 사용하려면 명령에 다음 파라미터를 추가합니다. 그렇지 않은 경우 Puppet 마스터 생성 프로세스가 자동으로 엔드포인트를 생성합니다. 사용자 지정 도메인을 구성하려면 세 가지 파라미터 모두가 필요합니다. 이러한 파라미터 사용에 대한 추가 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer를 참조하세요.
  - `--custom-domain` - 서버의 선택적 퍼블릭 엔드포인트(예: `https://aws.my-company.com`).
  - `--custom-certificate` - PEM 형식의 HTTPS 인증서. 값은 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다.
  - `--custom-private-key` - HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다.
- 매주 시스템 유지 관리가 필요합니다. 유효한 값은 `DDD:HH:MM` 형식으로 지정해야 합니다. 지정한 시간은 협정 세계시(UTC)로 표시됩니다. `--preferred-maintenance-window`의 값을 지정하지 않으면 기본값은 화요일, 수요일 또는 금요일의 임의 한 시간입니다.
- `--preferred-backup-window`의 유효 값은 `HH:MM`(매일 백업) 또는 `DDD:HH:MM`(매주 백업) 형식 중 하나로 지정해야 합니다. 지정한 시간은 UTC 형식입니다. 기본값은 임의 일일 시작 시간입니다. 자동 백업을 옵트아웃하려면 대신에 `--disable-automated-backup` 파라미터를 추가합니다.
- `--security-group-ids`에는 공백으로 구분하여 하나 이상의 보안 그룹 ID를 입력합니다.
- `--subnet-ids`에는 서브넷 ID를 입력합니다.

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

다음 예제에서는 사용자 정의 도메인을 사용하는 Puppet 마스터를 만듭니다.

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
```

```
    --instance-profile-arn "arn:aws:iam::1019881987024:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "c4.large" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::044726508045:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

4. OpsWorks for Puppet Enterprise는 새 서버를 만드는 데 약 15분이 걸립니다. `create-server` 명령의 출력에서 `Endpoint` 속성 값을 복사합니다. 다음은 예입니다.

```
"Endpoint": "puppet-2019-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

`create-server` 명령의 출력을 무시하거나 셸 세션을 닫지 마십시오. 다시 표시되지 않는 중요 정보가 출력에 포함되어 있을 수도 있기 때문입니다. `create-server` 결과에서 암호 및 스타터 키트를 가져오려면 다음 단계로 이동합니다.

5. OpsWorks for Puppet Enterprise에서 자동으로 암호를 생성하도록 선택한 경우 jq와 같은 JSON 프로세서를 사용하여 `create-server` 결과에서 사용 가능한 형식으로 암호를 추출할 수 있습니다. jq를 설치한 후에는 다음 명령을 실행하여 Puppet 관리자 암호 및 스타터 키트를 추출할 수 있습니다. 3단계에서 고유의 암호를 제공하지 않은 경우 추출한 관리자 암호를 안전하면서 편리한 위치에 저장해야 합니다.

```
#Get the Puppet password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_ADMIN_PASSWORD") | .Value'

#Get the Puppet Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

> **Note**
>
> AWS Management Console에서 새 Puppet 마스터 스타터 키트를 다시 생성할 수 없습니다. 를 사용하여 Puppet 마스터를 생성할 때 이전 `jq` 명령을 AWS CLI실행하여 base64로 인코딩된 스타터 키트를 `create-server` 결과에 ZIP 파일로 저장합니다.

6. 또는 `create-server` 명령 결과에서 스타터 키트를 추출하지 않은 경우 OpsWorks for Puppet Enterprise 콘솔의 서버 속성 페이지에서 새 스타터 키트를 다운로드할 수 있습니다.
7. 사용자 지정 도메인을 사용하지 않는 경우 다음 단계로 이동합니다. 서버에서 사용자 지정 도메인을 사용하는 경우 엔터프라이즈의 DNS 관리 도구에서 CNAME 항목을 만들어 사용자 지정 도메인을 4단계에서 복사한 OpsWorks for Puppet Enterprise 엔드포인트로 지정합니다. 이 단계를 완료해야 사용자 지정 도메인을 사용하여 서버에 연결하거나 로그인할 수 있습니다.
8. 서버 생성 프로세스가 완료되면 스타터 키트를 사용하여 Puppet 마스터 구성 단원을 진행합니다.

참고

- 를 사용하여 Puppet Enterprise Master 생성 AWS CLI
- OpsWorks for Puppet Enterprise 서버 백업 및 복원
- AWS OpsWorks CM API 참조의 CreateServer
- AWS CLI 명령 Reference의 create-server

AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

태그는 AWS 리소스를 식별 및 구성하기 위한 메타데이터로 작동하는 단어나 구문입니다. OpsWorks for Puppet Enterprise에서는 리소스에 최대 50개의 사용자 적용 태그가 포함될 수 있습니다. 각 태그는

키와 하나의 값(선택 사항)으로 구성됩니다. OpsWorks for Puppet Enterprise에서는 다음과 같은 리소스에 태그를 적용할 수 있습니다.

- OpsWorks for Puppet Enterprise 서버
- OpsWorks for Puppet Enterprise 서버의 백업

AWS 리소스에 태그를 지정하면 비용을 추적하고, 리소스에 대한 액세스를 제어하고, 작업 자동화를 위한 리소스를 그룹화하거나, 용도 또는 수명 주기 단계별로 리소스를 구성하는 데 도움이 될 수 있습니다. 태그의 이점에 대한 자세한 내용은 AWS Answers의 AWS 태깅 전략 및 AWS 결제 및 비용 관리 사용 설명서의 비용 할당 태그 사용을 참조하세요.

태그를 사용하여 OpsWorks for Puppet Enterprise 서버 또는 백업에 대한 액세스를 제어하려면 AWS Identity and Access Management (IAM)에서 정책 설명을 생성하거나 편집합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 리소스 태그를 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

OpsWorks for Puppet Enterprise 마스터에 태그를 적용하면 마스터의 백업, 백업을 저장하는 Amazon S3 버킷, 마스터의 Amazon EC2 인스턴스,에 저장된 마스터의 보안 암호 AWS Secrets Manager, 마스터가 사용하는 탄력적 IP 주소에도 태그가 적용됩니다. 태그는가 Puppet 마스터를 생성하는 데 AWS OpsWorks 사용하는 AWS CloudFormation 스택에 전파되지 않습니다.

주제

- 에서 태그 작동 방식 AWS OpsWorks for Puppet Enterprise
- OpsWorks for Puppet Enterprise에서 태그 추가 및 관리(콘솔)
- OpsWorks for Puppet Enterprise(CLI) 에서 태그 추가 및 관리
- 참고

에서 태그 작동 방식 AWS OpsWorks for Puppet Enterprise

이 릴리스에서는 AWS OpsWorks CM API 및 AWS Management Console을 사용하여 태그를 추가하고 관리할 수 있습니다. AWS OpsWorks 또한 CM은 EC2 인스턴스, Secrets Manager의 보안 암호, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업을 포함하여 서버에 연결된 AWS 리소스에 서버에 추가하는 태그를 추가하려고 시도합니다.

다음 표는 OpsWorks for Puppet Enterprise에서 태그를 추가하고 관리하는 방법의 개요를 제공합니다.

작업	사용 작업
새 OpsWorks for Puppet Enterprise 서버 또는 수동으로 생성 중인 백업에 태그를 추가합니다.	• Puppet Enterprise 서버 생성을 선택하고 고급 설정 구성 페이지에서 태그를 추가합니다. • 기존 서버의 백업 페이지에서 백업 생성을 선택하고 Puppet Enterprise 서버의 백업 생성 페이지에서 태그를 추가하세요. • `CreateServer` 또는 `CreateBackup` 명령에 `Tags` 파라미터를 추가합니다.
리소스의 태그를 봅니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택합니다. • 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. • `ListTagsForResource` 명령을 실행합니다.
백업이 수동으로 생성되었는지 또는 자동으로 생성되었는지에 관계없이 기존 OpsWorks for Puppet Enterprise 서버 또는 백업에 태그를 추가합니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택한 다음 편집을 선택합니다. • 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. • `TagResource` 명령을 실행합니다.
리소스에서 태그를 삭제합니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택한 다음 편집을 선택합니다. 삭제할 태그 옆에 있는 X를 선택합니다. • 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. 삭제할 태그 옆에 있는 X를 선택합니다. • `UntagResource` 명령을 실행합니다.

`DescribeServers` 및 `DescribeBackups` 응답에는 태그 정보가 포함되지 않습니다. 태그를 표시하려면 `ListTagsForResource` API를 사용합니다.

OpsWorks for Puppet Enterprise에서 태그 추가 및 관리(콘솔)

이 섹션의 절차는 AWS Management Console에서 수행됩니다.

태그를 추가할 경우 태그 키는 비워둘 수 없습니다. 키는 최대 127자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다. 태그 값은 선택 사항입니다. 키는 있지만 값은 없는 태그를 추가할 수 있습니다. 값은 최대 255자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다.

주제

- 새 OpsWorks for Puppet Enterprise 서버에 태그 추가(콘솔)
- 새 백업에 태그 추가(콘솔)
- 기존 서버에서 태그 추가 또는 보기(콘솔)
- 기존 백업에서 태그 추가 또는 보기(콘솔)
- 서버에서 태그 삭제(콘솔)
- 백업에서 태그 삭제(콘솔)

새 OpsWorks for Puppet Enterprise 서버에 태그 추가(콘솔)

1. OpsWorks for Puppet Enterprise 마스터를 만들기 위한 사전 요구 사항을 모두 완료해야 합니다.
2. 를 사용하여 Puppet Enterprise Master 생성 AWS Management Console의 1-8단계를 따릅니다.
3. 자동 백업 설정을 지정한 후 고급 설정 구성 페이지의 태그 영역에 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다. 태그 추가가 완료되면 다음을 선택합니다.
4. 를 사용하여 Puppet Enterprise Master 생성 AWS Management Console의 11단계로 이동하여 새 서버에 대해 선택한 설정을 검토합니다.

새 백업에 태그 추가(콘솔)

1. OpsWorks for Puppet Enterprise 홈 페이지에서 기존 Puppet 마스터를 선택합니다.
2. 서버 세부 정보 페이지의 탐색 창에서 백업을 선택합니다.
3. 백업 페이지에서 백업 생성을 선택합니다.
4. 태그를 추가합니다. 태그 추가가 완료되면 생성을 클릭합니다.

기존 서버에서 태그 추가 또는 보기(콘솔)

1. OpsWorks for Puppet Enterprise 홈 페이지에서 기존 Puppet 마스터를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 태그를 선택하거나 세부 정보 페이지 하단에서 모든 태그 보기를 선택합니다.
3. 태그 페이지에서 편집을 선택합니다.
4. 서버에서 태그를 추가하거나 편집합니다. 작업을 마쳤으면 저장을 선택합니다.

> **Note**
>
> Puppet 마스터에서 태그를 변경하면 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같이 서버와 연결된 리소스의 태그도 변경됩니다.

기존 백업에서 태그 추가 또는 보기(콘솔)

1. OpsWorks for Puppet Enterprise 홈 페이지에서 기존 Puppet 마스터를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 백업을 선택하거나 세부 정보 페이지의 최근 백업 영역에서 모든 백업 보기를 선택합니다.
3. 백업 페이지에서 관리할 백업을 선택한 다음 백업 편집을 선택합니다.
4. 백업에서 태그를 추가하거나 편집합니다. 완료되면 업데이트를 선택합니다.

서버에서 태그 삭제(콘솔)

1. Puppet Enterprise 홈 페이지에서 기존 Puppet 마스터를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 태그를 선택하거나 세부 정보 페이지 하단에서 모든 태그 보기를 선택합니다.
3. 태그 페이지에서 편집을 선택합니다.
4. 태그 옆에 있는 X를 선택하여 태그를 삭제합니다. 작업을 마쳤으면 저장을 선택합니다.

> **Note**
>
> Puppet 마스터에서 태그를 변경하면 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같이 서버와 연결된 리소스의 태그도 변경됩니다.

백업에서 태그 삭제(콘솔)

1. Puppet Enterprise 홈 페이지에서 기존 Puppet 마스터를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 백업을 선택하거나 세부 정보 페이지의 최근 백업 영역에서 모든 백업 보기를 선택합니다.
3. 백업 페이지에서 관리할 백업을 선택한 다음 백업 편집을 선택합니다.
4. 태그 옆에 있는 X를 선택하여 태그를 삭제합니다. 완료되면 업데이트를 선택합니다.

OpsWorks for Puppet Enterprise(CLI) 에서 태그 추가 및 관리

이 섹션의 절차는 AWS CLI에서 수행됩니다. 태그 작업을 시작하기 AWS CLI 전에의 최신 릴리스를 실행해야 합니다. 설치 또는 업데이트에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 설치를 AWS CLI AWS CLI참조하세요.

태그를 추가할 경우 태그 키는 비워둘 수 없습니다. 키는 최대 127자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다. 태그 값은 선택 사항입니다. 키는 있지만 값은 없는 태그를 추가할 수 있습니다. 값은 최대 255자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다.

주제

- 새 OpsWorks for Puppet Enterprise Server에 태그 추가(CLI)
- 새 백업에 태그 추가(CLI)
- 기존 서버 또는 백업에 태그 추가(CLI)
- 리소스 태그 나열(CLI)
- 리소스에서 태그 삭제(CLI)

새 OpsWorks for Puppet Enterprise Server에 태그 추가(CLI)

OpsWorks for Puppet Enterprise 서버를 생성할 때를 사용하여 태그를 AWS CLI 추가할 수 있습니다. 이 절차에서는 서버를 만드는 방법을 자세히 설명하지 않습니다. 를 사용하여 OpsWorks for Puppet Enterprise 서버를 생성하는 방법에 대한 자세한 내용은이 설명서를 사용하여 Puppet Enterprise Master 생성 AWS CLI의 섹션을 AWS CLI참조하세요. 서버에 최대 50개의 태그를 추가할 수 있습니다.

1. OpsWorks for Puppet Enterprise 서버를 만들기 위한 사전 요구 사항을 모두 완료해야 합니다.
2. 를 사용하여 Puppet Enterprise Master 생성 AWS CLI의 1-4단계를 완료합니다.

3. 5단계에서 `create-server` 명령을 실행할 때, 다음 예제와 같이 명령에 `--tags` 파라미터를 추가합니다.

```
aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

   다음은 `create-server` 명령의 태그 부분만 보여주는 예제입니다.

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

4. 를 사용하여 Puppet Enterprise Master 생성 AWS CLI에 설명된 나머지 단계를 완료합니다. 태그가 새 서버에 추가되었는지 확인하려면 이 주제의 리소스 태그 나열(CLI)에 소개된 단계를 따릅니다.

새 백업에 태그 추가(CLI)

OpsWorks for Puppet Enterprise 서버의 수동 백업을 새로 생성할 때를 사용하여 태그를 AWS CLI 추가할 수 있습니다. 이 절차에서는 수동 백업을 만드는 방법을 자세히 설명하지 않습니다. 수동 백업을 생성하는 방법에 대한 자세한 내용은의 "에서 수동 백업을 수행하려면 AWS CLI"을 참조하세요OpsWorks for Puppet Enterprise 서버 백업. 백업에 최대 50개의 태그를 추가할 수 있습니다. 서버에 태그가 있는 경우 새 백업에 서버 태그가 자동으로 지정됩니다.

기본적으로 새 OpsWorks for Puppet Enterprise 서버를 만들면 자동 백업이 활성화됩니다. 이 주제의 기존 서버 또는 백업에 태그 추가(CLI)에서 설명하는 `tag-resource` 명령을 실행하여 자동 백업에 태그를 추가할 수 있습니다.

- 백업을 만들 때 수동 백업에 태그를 추가하려면 다음 명령을 실행합니다. 명령의 태그 부분만 표시됩니다. 전체 `create-backup` 명령의 예는 OpsWorks for Puppet Enterprise 서버 백업의 “ AWS CLI에서 수동 백업 수행하기”를 참조하세요.

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

  다음은 `create-backup` 명령의 태그 부분만 보여주는 예제입니다.

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

기존 서버 또는 백업에 태그 추가(CLI)

`tag-resource` 명령을 실행하여 기존 OpsWorks for Puppet Enterprise 서버 또는 백업에 태그를 추가할 수 있습니다(백업이 자동으로 생성되었든 수동으로 생성되었든 상관없이). 대상 리소스의 Amazon 리소스 번호(ARN)를 지정하여 태그를 추가합니다.

1. 태그를 적용할 리소스의 ARN을 가져오려면 다음과 같이 하세요.

   - 서버의 경우 `describe-servers --server-name` *`server_name`*을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id` *`backup_ID`*를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. `describe-backups --server-name` *`server_name`*을 실행하여 특정 OpsWorks for Puppet Enterprise 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.

   다음 예제는 `describe-servers --server-name opsworks-cm-test` 명령의 결과인 `ServerArn`만 보여줍니다. `ServerArn` 값이 `tag-resource` 명령에 추가되어 서버에 태그를 추가합니다.

   ```
   {
       "Servers": [
           {
               ...
               "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
   opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
           }
       ]
   }
   ```

2. 1단계에서 반환한 ARN을 사용하여 `tag-resource` 명령을 실행합니다.

   ```
   aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   다음은 예입니다.

   ```
   aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production Key=Department,Value=Marketing
   ```

3. 태그가 성공적으로 추가되었는지 확인하려면 다음 절차 리소스 태그 나열(CLI)로 이동합니다.

리소스 태그 나열(CLI)

`list-tags-for-resource` 명령을 실행하여 OpsWorks for Puppet Enterprise 서버 또는 백업에 연결된 태그를 표시할 수 있습니다. 해당 태그를 볼 대상 리소스의 ARN을 지정합니다.

1. 태그를 나열할 리소스의 ARN을 가져오려면 다음과 같이 하세요.
   - 서버의 경우 `describe-servers --server-name` *`server_name`*을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id` *`backup_ID`*를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. `describe-backups --server-name` *`server_name`*을 실행하여 특정 OpsWorks for Puppet Enterprise 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.
2. 1단계에서 반환한 ARN을 사용하여 `list-tags-for-resource` 명령을 실행합니다.

   ```
   aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
   ```

   다음은 예입니다.

   ```
   aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
   ```

   리소스에 태그가 있는 경우 이 명령은 다음과 같은 결과를 반환합니다.

   ```
   {
       "Tags": [
           {
               "Key": "Stage",
               "Value": "Production"
           },
           {
               "Key": "Department",
               "Value": "Marketing"
           }
       ]
   }
   ```

리소스에서 태그 삭제(CLI)

`untag-resource` 명령을 실행하여 OpsWorks for Puppet Enterprise 서버 또는 백업에서 태그를 삭제할 수 있습니다. 리소스가 삭제되면 리소스의 태그도 삭제됩니다. 대상 리소스의 Amazon 리소스 번호(ARN)를 지정하여 태그를 제거합니다.

1. 태그를 제거할 리소스의 ARN을 가져오려면 다음과 같이 하세요.
   - 서버의 경우 `describe-servers --server-name` *`server_name`*을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id` *`backup_ID`*를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. `describe-backups --server-name` *`server_name`*을 실행하여 특정 OpsWorks for Puppet Enterprise 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.
2. 1단계에서 반환한 ARN을 사용하여 `untag-resource` 명령을 실행합니다. 삭제할 태그만 지정합니다.

   ```
   aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   이 예에서 `untag-resource` 명령은 키가 `Stage`이고, 값이 `Production`인 태그만 제거합니다.

   ```
   aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production
   ```
3. 태그가 성공적으로 삭제되었는지 확인하려면 이 주제의 리소스 태그 나열(CLI)에 소개된 단계를 따릅니다.

참고

- 를 사용하여 Puppet Enterprise Master 생성 AWS CLI
- OpsWorks for Puppet Enterprise 서버 백업
- AWS 태깅 전략
- AWS Identity and Access Management 사용 설명서의 AWS 리소스 태그를 사용하여 리소스에 대한 액세스 제어
- AWS 결제 및 비용 관리 사용 설명서의 비용 할당 태그 사용
- AWS OpsWorks CM API 참조에서 CreateBackup

- AWS OpsWorks CM API 참조에서 CreateServer
- AWS OpsWorks CM API 참조의 TagResource
- AWS OpsWorks CM API 참조의 ListTagsForResource
- AWS OpsWorks CM API 참조의 UntagResource

OpsWorks for Puppet Enterprise 서버 백업 및 복원

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 OpsWorks for Puppet Enterprise 서버를 백업하고 복원하는 방법을 설명합니다.

주제

- OpsWorks for Puppet Enterprise 서버 백업
- OpsWorks for Puppet Enterprise 서버를 백업에서 복원

OpsWorks for Puppet Enterprise 서버 백업

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

매일 또는 매주 반복되는 OpsWorks for Puppet Enterprise 서버 백업을 정의하고 서비스가 사용자를 대신하여 Amazon Simple Storage Service(S3)에 백업을 저장하도록 할 수 있습니다. 또는 온디맨드로 수동 백업을 만들 수 있습니다.

백업은 Amazon S3에 저장되므로 추가 비용이 발생합니다. 최대 30개 생성까지 백업 보존 기간을 정의할 수 있습니다. AWS 지원 채널을 사용하여 해당 제한을 변경하도록 서비스 요청을 제출할 수 있습니다. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

OpsWorks for Puppet Enterprise 마스터의 백업에 태그를 추가할 수 있습니다. OpsWorks for Puppet Enterprise 마스터에 태그를 추가한 경우 Puppet 마스터의 자동 백업에 해당 태그가 상속됩니다. 백업에서 태그를 추가하고 관리하는 방법에 대한 자세한 내용은 이 안내서의 AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업 단원을 참조하세요.

주제

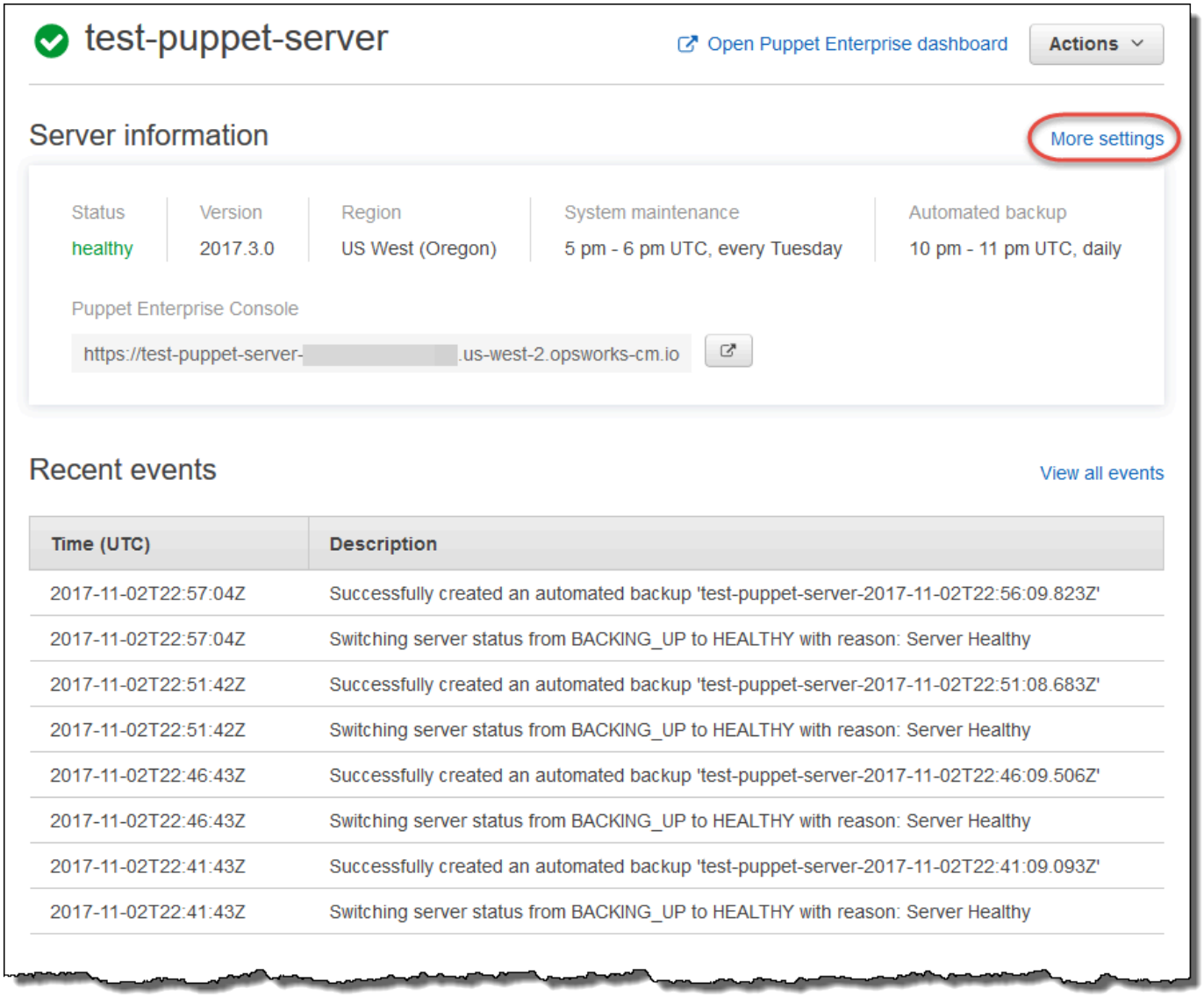
- 백업 자동화
- 수동 백업
- 백업 삭제

백업 자동화

OpsWorks for Puppet Enterprise 서버를 구성할 때는 자동 백업과 수동 백업 중 하나를 선택합니다. OpsWorks for Puppet Enterprise는 설치 마법사의 고급 설정 구성 페이지의 자동 백업 섹션에서 선택한 시간 및 날짜에 자동 백업을 시작합니다. 서버가 온라인 상태가 된 후 서버 속성 페이지에서 다음 단계를 수행하여 백업 설정을 변경할 수 있습니다.

자동화된 백업 설정을 변경하려면

1. 서버 속성 페이지에서 [더 많은 설정]를 선택합니다.



2. 자동 백업을 끄려면 자동 백업 활성화 옵션에서 아니오를 선택합니다. 변경 사항을 저장합니다. 다음 단계로 진행할 필요는 없습니다.
3. [자동 백업] 섹션에서 빈도, 시작 시간 또는 유지할 세대를 변경합니다. 변경 내용을 저장합니다.

수동 백업

에서 언제든지 AWS Management Console또는 create-backup 명령을 실행하여 수동 백업을 AWS CLI 시작할 수 있습니다. 수동 백업은 저장되는 최대 30세대의 자동 백업에 포함되지 않습니다. 최대 10개의 수동 백업이 저장되며 Amazon S3에서 수동으로 삭제해야 합니다.

에서 수동 백업을 수행하려면 AWS Management Console

1. [Puppet Enterprise 서버] 페이지에서 백업하려는 서버를 선택합니다.
2. 서버의 속성 페이지의 왼쪽 탐색 창에서 [백업]을 선택합니다.
3. [백업 생성]을 선택합니다.
4. 화면에서 백업의 [상태] 열에서 녹색 확인 표시가 나타나면 수동 백업이 완료됩니다.

에서 수동 백업을 수행하려면 AWS CLI

OpsWorks for Puppet Enterprise 서버의 새 수동 백업을 만들 때 태그를 추가할 수 있습니다. 수동 백업을 만들 때 태그를 추가하는 방법에 대한 자세한 내용은 새 백업에 태그 추가(CLI) 단원을 참조하세요.

- 수동 백업을 시작하려면 다음 AWS CLI 명령을 실행합니다.

```
aws opsworks-cm --region region name create-backup --server-name "Puppet server
 name" --description "optional descriptive string"
```

백업 삭제

백업을 삭제하면 백업이 저장된 S3 버킷에서 영구적으로 삭제가 됩니다.

에서 백업을 삭제하려면 AWS Management Console

1. [Puppet Enterprise 서버] 페이지에서 백업하려는 서버를 선택합니다.
2. 서버의 속성 페이지의 왼쪽 탐색 창에서 [백업]을 선택합니다.
3. 삭제하려는 백업을 선택한 다음 [백업 삭제]를 선택합니다. 한 번에 하나의 백업만 선택할 수 있습니다.
4. 삭제를 확인하는 메시지가 나타나면 [S3 버킷에 저장된 백업 삭제] 확인란을 선택한 다음 [예, 삭제]를 선택합니다.

에서 백업을 삭제하려면 AWS CLI

- 백업을 삭제하려면 다음 AWS CLI 명령을 실행하여의 값을 삭제하려는 백업의 ID--backup-id로 바꿉니다. 백업 ID는 *ServerName-yyyyMMddHHmmssSSS* 형식입니다. 예를 들어 **puppet-server-20171218132604388**입니다.

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

OpsWorks for Puppet Enterprise 서버를 백업에서 복원

Important

이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용할 수 있는 백업을 찾은 다음 OpsWorks for Puppet Enterprise 서버를 복원하려는 시점을 손쉽게 선택할 수 있습니다. 서버 백업에는 모듈, 클래스, 노드 연결, 데이터베이스 정보(보고서, 사실 정보 등이 포함) 같은 구성-관리 소프트웨어 영구 데이터가 포함되어 있습니다. 서버의 인 플레이스 복원(즉, 기존 OpsWorks for Puppet Enterprise 서버를 새 EC2 인스턴스로 복원)을 수행하면 서버 복원에 사용하는 백업 시점에 등록되었던 노드가 다시 등록되고, 복원에 성공하면 트래픽이 새 인스턴스로 전환됩니다. 복원된 OpsWorks for Puppet Enterprise 서버 상태는 `Healthy`입니다. 새로 만든 OpsWorks for Puppet Enterprise 서버로 복원해도 노드 연결이 유지되지 않습니다. 서버를 복원하면 Puppet 소프트웨어 버전이 업데이트되지 않습니다. 선택한 백업에서 사용할 수 있는 것과 동일한 Puppet 버전 및 구성 관리 데이터가 적용됩니다.

일반적으로 서버 복원은 새 서버를 만드는 것보다 시간이 더 걸리며, 시간은 선택한 백업 크기에 따라 달라집니다. 복원이 완료되면 이전 EC2 인스턴스는 `Running` 또는 `Stopped` 상태로 유지되지만 일시적입니다. 결국 종료됩니다.

이 릴리스에서는를 사용하여 OpsWorks for Puppet Enterprise에서 Puppet 마스터를 복원 AWS CLI 할 수 있습니다.

Note

restore-server 명령을 실행하여 현재 인스턴스 유형을 변경하거나 분실 또는 침해된 SSH 키를 복원 또는 설정할 수도 있습니다.

백업에서 서버를 복원하려면

1. 에서 다음 명령을 AWS CLI실행하여 사용 가능한 백업 목록과 해당 IDs. 사용할 백업 ID를 적어 둡니다. 백업 ID는 *myServerName-yyyyMMddHHmmssSSS* 형식입니다.

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 다음 명령을 실행합니다.

   ```
   aws opsworks-cm --region region name restore-server --backup-id "myServerName-
   yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
    key pair" --server-name "name of Puppet master"
   ```

   다음은 예입니다.

   ```
   aws opsworks-cm --region us-west-2 restore-server --backup-id
    "MyPuppetServer-20161120122143125" --server-name "MyPuppetServer"
   ```

3. 복원이 완료될 때까지 기다리십시오.

OpsWorks for Puppet Enterprise 시스템 유지 관리

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

필수 시스템 유지 관리는 보안 업데이트를 포함하여 최신 AWS테스트 버전의 Puppet Server가 항상 OpsWorks for Puppet Enterprise 서버에서 실행되도록 합니다. 시스템 유지 관리는 일주일에 최소한 1번은 수행해야 합니다. 를 사용하여 원하는 경우 일일 자동 유지 관리를 구성할 AWS CLI수 있습니다. 또한 AWS CLI 를 사용하여 예약된 시스템 유지 관리 외에도 온디맨드로 시스템 유지 관리를 수행할 수 있습니다.

Puppet 소프트웨어의 새 버전이 나오면 AWS 테스트를 통과하는 즉시 서버에서 Puppet 서버 버전이 자동으로 업데이트되도록 시스템 유지 관리가 설계되어 있습니다. AWS가 광범위한 테스트를 실시하

여 Puppet 업그레이드가 프로덕션 환경을 지원하는지, 기존의 고객 환경을 방해하지 않는지 확인하므로 Puppet 소프트웨어 릴리스 후 기존 OpsWorks for Puppet Enterprise 서버에서 애플리케이션을 사용할 수 있게 되기까지 시간이 지연될 수 있습니다. 요구에 따라 사용 가능한 Puppet 소프트웨어 버전을 업데이트하는 방법은 이 주제의 요청 시 시스템 유지 관리 시작 단원을 참조하세요.

시스템 유지 관리는 유지 관리 프로세스의 일부로 수행된 백업에서 새 인스턴스를 시작합니다. 그러면 정기 유지 관리가 진행 중인 성능이 저하되었거나 손상된 Amazon EC2 인스턴스의 위험을 줄일 수 있습니다.

> **Important**
>
> 시스템 유지 관리 시 OpsWorks for Puppet Enterprise 서버에 추가한 모든 파일 또는 사용자 지정 구성이 삭제됩니다. 구성 또는 파일 손실을 복구하는 자세한 방법은 이 주제의 유지 관리 후 사용자 지정 구성 및 파일 복원 단원을 참조하세요.

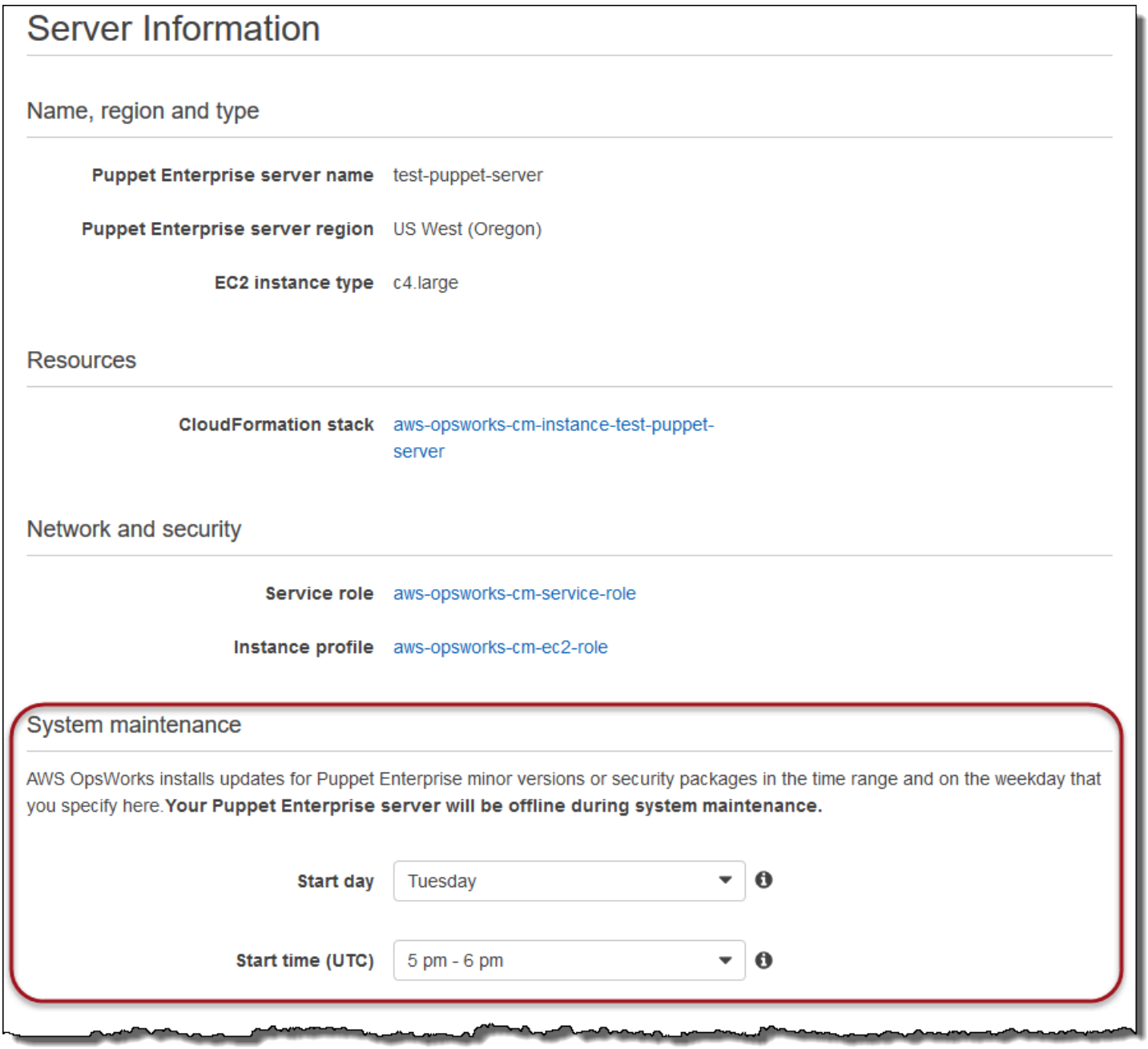
주제

- 시스템 유지 관리 구성
- 요청 시 시스템 유지 관리 시작
- 유지 관리 후 사용자 지정 구성 및 파일 복원

시스템 유지 관리 구성

새 OpsWorks for Puppet Enterprise 서버를 생성하는 경우 시스템 유지 관리를 시작할 날짜 및 시간을 협정 세계시(UTC)로 구성할 수 있습니다. 지정한 시간 동안 유지 관리가 시작됩니다. 시스템 유지 관리 중에는 서버가 오프라인 상태여야 하므로 정규 업무 시간 중 서버에 대한 수요가 낮은 시간을 선택하세요. 유지 관리 진행 중 서버 상태는 `UNDER_MAINTENANCE`입니다.

다음 스크린샷과 같이 서버 설정 페이지의 시스템 유지 관리 영역에서 설정을 변경하여 기존 OpsWorks for Puppet Enterprise 서버의 시스템 유지 관리 설정을 변경할 수도 있습니다.



[시스템 유지 관리] 섹션에서 시스템 유지 관리를 시작하려는 날짜와 시간을 설정합니다.

를 사용하여 시스템 유지 관리 구성 AWS CLI

또한 AWS CLI를 사용하여 시스템 유지 관리 자동 시작 시간을 구성할 수도 있습니다. 를 AWS CLI 사용하면 원하는 경우 3자 평일 접두사를 생략하여 일일 자동 유지 관리를 구성할 수 있습니다.

서버 인스턴스 생성을 위한 요구 사항(예: 인스턴스 유형, 인스턴스 프로파일 ARN 및 서비스 역할 ARN)을 지정한 후`create-server` 명령에서 명령에 `--preferred-maintenance-window` 파

라미터를 추가합니다. 다음 `create-server` 예에서는 `--preferred-maintenance-window`가 `Mon:08:00`으로 설정되어 있습니다. 즉, 매주 월요일 8:00 a.m. UTC에 유지보수가 시작되도록 설정했습니다. UTC 기준입니다.

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2017" --server-name "puppet-06" --instance-profile-arn
 "arn:aws:iam::1119001987000:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "c4.large" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

원하는 경우 `update-server` 명령에서는 `--preferred-maintenance-window` 값만 업데이트할 수 있습니다. 다음 예제에서 유지 관리 기간은 금요일 밤 6:15 p.m으로 설정되어 있습니다. UTC 기준입니다.

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "Fri:18:15"
```

유지 관리 기간의 시작 시간을 매일 세계협정시(UTC) 오후 6:15로 변경하려면 다음 예에서와 같이, 세 자리의 요일 접두사를 생략합니다.

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "18:15"
```

를 사용하여 기본 시스템 유지 관리 기간을 설정하는 방법에 대한 자세한 내용은 create-server 및 update-server를 AWS CLI참조하세요.

요청 시 시스템 유지 관리 시작

구성된 주간 또는 일일 자동 유지 관리 외에 온디맨드 시스템 유지 관리를 시작하려면 다음 AWS CLI 명령을 실행합니다. AWS Management Console에서는 온디맨드 유지 관리를 시작할 수 없습니다.

```
aws opsworks-cm start-maintenance --server-name server_name
```

이 명령에 대한 자세한 내용은 start-maintenance를 참조하세요.

유지 관리 후 사용자 지정 구성 및 파일 복원

시스템 유지 관리 시 OpsWorks for Puppet Enterprise 서버에 추가한 사용자 지정 파일 또는 구성이 삭제 또는 변경될 수 있습니다.

RunCommand 또는 SSH를 사용하여 사용자가 추가한 파일 또는 설정이 유지 관리 실행 후 Puppet 마스터에서 누락된 경우, Amazon 머신 이미지(AMI)를 사용하여 새 Amazon EC2 인스턴스를 시작할 수 있습니다. 서버의 사전 유지 관리 구성에서 빌드된 AMI를 사용할 수 있습니다.

새 인스턴스는 유지 관리 이전의 Puppet 마스터와 상태가 동일하므로 누락된 파일 및 설정이 포함되어 있어야 합니다.

> **Important**
>
> 새 인스턴스는 서버를 복원하는 데 사용할 수 없고 Puppet 마스터로 실행할 수 없습니다. 이 인스턴스는 파일 및 구성 설정을 복원하는 데에만 사용할 수 있습니다.

Amazon EC2 콘솔에서 AMI의 EC2 인스턴스를 시작하려면 시작 마법사를 열고 내 AMI를 선택한 후 서버와 이름이 같은 AMI를 선택합니다. 다른 인스턴스를 시작할 때와 마찬가지로 Amazon EC2 마법사의 단계에 따라 진행합니다.

OpsWorks for Puppet Enterprise에서 자동으로 노드 추가

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 서비스 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는 Amazon Elastic Compute Cloud(Amazon EC2) 노드를 OpsWorks for Puppet Enterprise 서버에 자동으로 추가하는 방법을 설명합니다. Puppet 마스터가 관리할 노드 추가에서 `associate-node` 명령을 사용하여 한 번에 하나씩 Puppet Enterprise 서버에 노드를 추가하는 방법을 배웠습니다. 이 주제의 코드는 무인 방식을 사용하여 자동으로 노드를 추가하는 방법을 보여 줍니다. 권장되는 새 노드의 무인(또는 자동) 연결 방법은 Amazon EC2 사용자 데이터를 구성하는 것입니다. 기본적으로 OpsWorks for Puppet Enterprise 서버는 Ubuntu, Amazon Linux 및 RHEL 노드 운영 체제에서 이미 `puppet-agent`가 지원되고 있습니다.

노드 연결을 끊는 방법에 대한 자세한 내용은 이 안내서의 OpsWorks for Puppet Enterprise 서버를 노드에서 분리와 OpsWorks for Puppet Enterprise API 설명서의 `disassociate-node`를 참조하세요.

1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성

EC2 인스턴스 프로파일로 사용할 AWS Identity and Access Management (IAM) 역할을 생성하고 다음 정책을 IAM 역할에 연결합니다. 이 정책은 노드 등록 중에 `opsworks-cm` API가 EC2 인스턴스와 통신하도록 허용합니다. 인스턴스 프로파일에 대한 자세한 내용은 Amazon EC2 설명서의 인스턴스 프로파일 사용하기를 참조하세요. IAM 역할을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 설명서의 콘솔에서 IAM 역할 생성을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
                "opsworks-cm:DescribeServers",
                "ec2:DescribeTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

2단계: 무인 연결 스크립트를 사용하여 인스턴스 생성

EC2 인스턴스를 생성하려면 스타터 키트에 포함된 사용자 데이터 스크립트를 EC2 인스턴스 지침, Amazon EC2 Auto Scaling 그룹 시작 구성 또는 AWS CloudFormation 템플릿의 `userdata` 섹션에 복사할 수 있습니다. 스크립트는 Ubuntu 및 Amazon Linux 운영 체제를 실행하는 EC2 인스턴스에만 지원됩니다. 사용자 데이터에 스크립트 추가에 대한 자세한 내용은 Amazon EC2 사용 설명서의 시작 시 Linux 인스턴스에서 명령 실행을 참조하세요. 새 노드를 생성하는 가장 쉬운 방법은 Amazon EC2 인스턴스 시작 마법사를 사용하는 것입니다. 본 안내서에서는 OpsWorks for Puppet Enterprise 시작하기에 설명되어 있는 Apache 웹 서버 예제 모듈 설정을 사용합니다.

1. 스타터 키트의 사용자 데이터 스크립트는 `opsworks-cm` API `associate-node` 명령을 실행하여 새 노드를 Puppet 마스터와 연결합니다. 이 릴리스에서는 최신 버전을 아직 실행하지 않는 경우 노드 AWS CLI 에의 현재 up-to-date 버전도 설치합니다. 이 스크립트를 편리한 위치에 `userdata.sh`로 저장합니다.

   기본적으로 새로 등록된 노드의 이름은 인스턴스 ID입니다.

2. EC2 설명서의 인스턴스 시작 단원에 나와 있는 절차를 따릅니다(여기에서 수정). EC2 인스턴스 시작 마법사에서 Amazon Linux AMI를 선택합니다.
3. 인스턴스 세부 정보 구성 페이지에서 IAM 역할로 1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성에서 생성한 역할인 MyPuppetInstanceProfile을 선택합니다.
4. [고급 세부 정보] 영역에서 1단계에서 생성한 `userdata.sh` 스크립트를 업로드합니다.
5. [스토리지 추가] 페이지에서 변경해야 할 사항은 없습니다. [태그 추가]로 이동합니다.

   EC2 인스턴스에 태그를 적용하여 `userdata.sh` 동작을 사용자 지정할 수 있습니다. 이 예제에서는 **`apache_webserver`** 값과 함께 **`pp_role`**, 태그를 추가하여 노드에 `apache_webserver` 역할을 적용합니다.

   노드에서 `pp_role` 값을 설정하면 노드의 에이전트 인증서에 영구 저장된 데이터 값이 설정되어 신뢰할 수 있는 노드 분류가 가능합니다. 자세한 정보는 Puppet 플랫폼 설명서의 Extension requests(permanent certificate data) 단원을 참조하세요.
6. 이 예제에서는 보안 그룹 구성 페이지에서 규칙 추가를 선택한 후 HTTP를 입력하여 Apache 웹 서버에서 8080 포트를 엽니다.
7. [검토 및 시작]을 선택한 다음 [시작]을 선택합니다. 새 노드가 시작되면 스타터 키트 Apache 예제 설정에서 설정한 샘플 모듈의 Apache 구성이 적용됩니다.
8. 새 노드의 퍼블릭 DNS에 연결된 웹 페이지를 열면 Puppet 관리형 Apache 웹 서버에서 호스팅된 웹 사이트가 나타납니다.

OpsWorks for Puppet Enterprise 서버를 노드에서 분리

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 OpsWorks for Puppet Enterprise 서버에 의한 관리에서 관리형 노드를 연결 해제 또는 제거하는 방법을 설명합니다. 이 작업은 명령줄이나 Puppet Enterprise 콘솔에서 수행합니다. OpsWorks for Puppet Enterprise 관리 콘솔에서는 노드를 연결 해제할 수 없습니다. 현재 OpsWorks for Puppet Enterprise API는 여러 노드의 배치 제거를 허용하지 않습니다. 이 섹션에서 사용하는 명령은 노드를 한 번에 하나씩 연결 해제합니다.

Puppet 마스터 서버를 삭제하기 전에 노드 연결을 해제하는 것이 좋습니다. 그래야 노드가 서버에 재연결을 시도하지 않고 계속 작동합니다. 이렇게 하려면 `disassociate-node` AWS CLI 명령을 실행합니다. PE에서 노드를 완전히 삭제하려면 노드 연결을 해제하고 인증서를 취소해야 합니다. 그래야 노드가 Puppet 마스터에 계속해서 체크인을 시도하지 않습니다. 더 이상 Puppet 마스터를 사용하여 노드를 관리하고 싶지 않으면 노드에서 puppet-agent를 제거해야 합니다.

노드 연결 해제

1. 에서 다음 명령을 AWS CLI실행하여 노드 연결을 해제합니다. *Node_name*은 연결 해제할 노드의 이름으로, Amazon EC2 인스턴스의 경우 이 값은 인스턴스 ID입니다. *Server_name*은 노드 연결을 해제하려는 Puppet 마스터의 이름입니다. 두 파라미터는 모두 필수입니다. 기본 리전에 없는 Puppet 마스터에서 노드 연결을 해제하지 않으려는 경우에는 `--region` 파라미터가 필요하지 않습니다.

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

   다음 명령은 예제입니다.

   ```
   aws opsworks-cm --region us-west-2 disassociate-node --node-name
    i-0010zzz00d66zzz90 --server-name opsworkstest
   ```

2. 연결 해제가 완료되었다는 응답 메시지가 표시될 때까지 기다리십시오.

OpsWorks for Puppet Enterprise 서버를 삭제하는 방법에 대한 자세한 정보는 OpsWorks for Puppet Enterprise 서버 삭제 섹션을 참조하세요.

참고

- Puppet Enterprise 설명서의 노드 제거

OpsWorks for Puppet Enterprise 서버 삭제

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이

그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 OpsWorks for Puppet Enterprise 서버를 삭제하는 방법을 설명합니다. 서버를 삭제하면 이벤트, 로그 및 서버에 저장된 모든 모듈도 삭제됩니다. 지원 리소스(Amazon Elastic Compute Cloud 인스턴스, Amazon Elastic Block Store 볼륨 등)도 모든 자동 백업과 함께 삭제됩니다.

서버를 삭제해도 노드가 삭제되지는 않지만 노드는 삭제된 서버에 의해 더 이상 관리되지 않으며 계속 재연결을 시도합니다. 이런 이유로 Puppet 마스터를 삭제하기 전에 관리형 노드의 연결을 해제하는 것이 좋습니다. 이 릴리스에서는 AWS CLI 명령을 실행하여 노드의 연결을 해제할 수 있습니다.

1단계: 관리되는 노드 연결 해제

Puppet 마스터 서버를 삭제하기 전에 노드 연결을 해제해야 노드가 서버에 재연결을 시도하지 않고 계속 작동합니다. 이렇게 하려면 `disassociate-node` AWS CLI 명령을 실행합니다.

노드 연결 해제

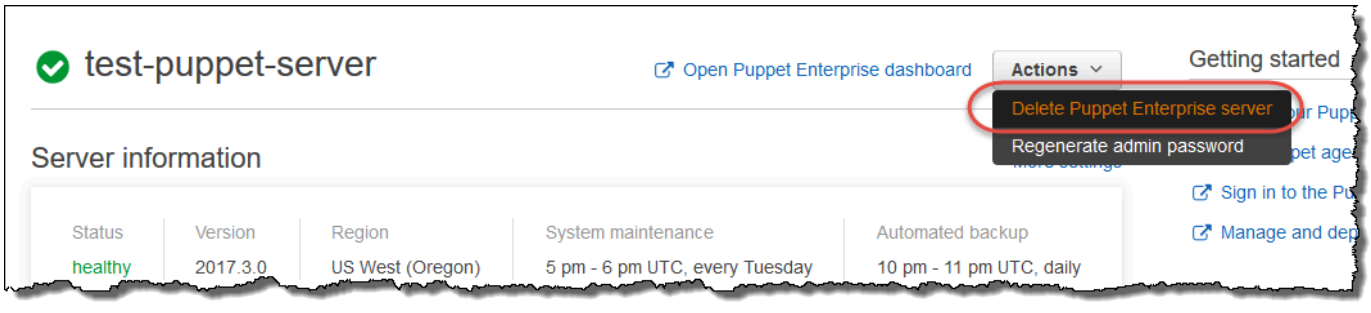
1. 에서 다음 명령을 AWS CLI실행하여 노드 연결을 해제합니다. *`Server_name`*은 노드 연결을 해제하려는 Puppet 마스터의 이름입니다. `--node-name` 값은 인스턴스 ID가 될 수 있습니다.

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 연결 해제가 완료되었다는 응답 메시지가 표시될 때까지 기다리십시오.

2단계: 서버 삭제

1. 대시보드의 서버 타일에서 [작업] 메뉴를 확장합니다.



2. [Puppet Enterprise 서버 삭제]를 선택합니다.

3. 삭제를 확인하는 메시지가 나타나면 연결된 역할 및 리소스를 삭제한다는 확인란을 선택한 다음, [예, 삭제]를 선택합니다.

참고

- OpsWorks for Puppet Enterprise 서버를 노드에서 분리

OpsWorks for Puppet Enterprise 서버를 Amazon Elastic Compute Cloud(Amazon EC2)로 마이그레이션하는 방법

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

아래 지침은 OpsWorks 외부에서 구성 관리 요구 사항에 Puppet Enterprise를 계속 사용하려는 경우 기존 Puppet Enterprise 서버를 Amazon EC2로 마이그레이션하는 방법을 설명합니다.

주제

- 1단계: Puppet에 문의하여 라이선스 구매
- 2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기
- 3단계: OpsWorks for Puppet Enterprise 서버의 백업 만들기
- 4단계: 새 EC2 인스턴스 시작
- 5단계: 새 EC2 인스턴스에 Puppet Enterprise를 설치합니다.
- 6단계: 새 EC2 인스턴스에서 백업 복원
- 7단계: Puppet 라이선스 구성
- 8단계: 노드 마이그레이션
- 9단계: OpsWorks for Puppet Enterprise 서버를 삭제합니다

1단계: Puppet에 문의하여 라이선스 구매

서버를 EC2로 마이그레이션하는 경우 새 인스턴스에는 Puppet 라이선스가 제공되지 않습니다. 라이선스 키를 구입하려면 Puppet 웹 사이트의 지침을 따르세요.

2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기

OpsWorks for Puppet Enterprise 서버를 찾아서 저장합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

   OpsWorks for Puppet Enterprise 서버의 기존 Amazon S3 버킷 이름을 복사합니다. 버킷 이름의 형식은 다음과 같습니다. `aws-opsworks-cm-`*`server-name`*`-`*`random-string`*

2. aws opsworks-cm describe-servers 명령을 실행하여 OpsWorks for Puppet Enterprise 서버를 구성합니다.

   ```
   aws opsworks-cm describe-servers \
      --server-name server-name \
      --region region
   ```

   응답의 `InstanceType`, `KeyPair`, `SubnetIds`, `SecurityGroupIds`. `InstanceProfileArn` 및 `Endpoint` 값을 저장합니다.

3. SSH를 사용하여 기존 OpsWorks for Puppet Enterprise 서버에 연결합니다. EC2 콘솔에서 SSH 대신 세션 관리자를 사용할 수 있습니다.

   다음 명령을 실행합니다.

   ```
   rpm -qa | grep opsworks-cm-puppet-enterprise | cut -d '-' -f 5
   ```

   응답은 Puppet Enterprise 버전(예: 2019.8.10)을 제공합니다. 이 값을 저장합니다.

다음 단계에서 SSH 또는 세션 관리자를 사용합니다.

3단계: OpsWorks for Puppet Enterprise 서버의 백업 만들기

1. 다음 명령을 실행하여 로컬 백업을 만드세요.

```
mkdir /tmp/puppet-backup/
sudo /opt/puppetlabs/bin/puppet-backup create --dir=/tmp/puppet-backup/
```

2. 다음 명령을 실행하여 백업 이름을 저장합니다.

```
ls /tmp/puppet-backup/
PUPPET_BACKUP=$(ls /tmp/puppet-backup/)
```

3. 다음 명령을 실행하여 Amazon S3 버킷에 백업을 업로드합니다. *S3-###* 2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 1단계 값으로 바꿉니다.

```
aws s3 cp /tmp/puppet-backup/PUPPET_BACKUP s3://S3_Bucket/tmp/puppet-backup/
```

PUPPET_BACKUP및 S3_BUCKET 값을 저장합니다. 새 EC2 인스턴스로 해당 값을 가져옵니다.

SSH 또는 세션 관리자 세션을 종료할 수 있습니다.

4단계: 새 EC2 인스턴스 시작

OpsWorks for Puppet Enterprise 서버와 동일한 구성을 사용하여 https://console.aws.amazon.com/ec2/ EC2 콘솔에서 새 EC2 인스턴스를 시작합니다.

파라미터 이름	값
OS	Amazon Linux 2
인스턴스 유형	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계의 `InstanceType` 값입니다.
키 페어 이름	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계의 `KeyPair` 값입니다.
VPC	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계에서 가져온 `SubnetIds` 의 VPC입니다.
서브넷	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계에서 가져온 `SubnetIds` .

파라미터 이름	값
기존 보안 그룹 선택 -> 공통 보안 그룹	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계에서 가져온 `SecurityGroupIds` .
스토리지	최소 120GB.
IAM 인스턴스 프로파일	2단계: OpsWorks for Puppet Enterprise 서버에 대한 세부 정보 가져오기의 2단계에서 가져온 `InstanceProfileArn` .

탄력적 IP를 생성하여 새 인스턴스에 연결하려는 경우 새 인스턴스의 인스턴스 ID를 복사하고 (선택 사항) 4.1단계: 탄력적 IP 생성 및 연결의 단계를 완료하세요.

(선택 사항) 4.1단계: 탄력적 IP 생성 및 연결

탄력적 IP 주소를 사용하면 주소를 계정의 다른 인스턴스에 신속하게 다시 매핑하여 인스턴스나 소프트웨어의 오류를 마스킹할 수 있습니다.

탄력적 IP 주소를 생성 및 연결하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탄력적 IP를 선택합니다.
3. 탄력적 IP 주소 할당를 선택합니다.
4. 탄력적 IP 주소 할당 페이지에서 할당을 선택합니다. 그러면 퍼블릭 IPv4 주소가 생성됩니다.
5. 할당된 IPv4 주소를 복사합니다.
6. 작업에서 탄력적 IP 주소 연결을 선택합니다.
7. 인스턴스의 경우, 새 인스턴스의 인스턴스 ID를 입력합니다.
8. 연결을 선택합니다.

5단계: 새 EC2 인스턴스에 Puppet Enterprise를 설치합니다.

SSH를 사용하여 새 EC2 인스턴스에 연결합니다. EC2 콘솔에서 SSH 대신 세션 관리자를 사용할 수 있습니다.

```
# switch to sudo user
sudo -i
```

```
# Setup environment variables
PUPPET_ENTERPRISE_VERSION=Puppet Enterprise version from step 2.3
hostname Public IPv4 DNS or Custom Domain if available

# Install Puppet Enterprise
curl -JLO https://pm.puppetlabs.com/puppet-enterprise/$PUPPET_ENTERPRISE_VERSION/
puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz
tar -xf puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz

./puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64/puppet-enterprise-installer
```

다음 단계를 위해 SSH 또는 세션 관리자 세션을 열어 둘 수 있습니다.

6단계: 새 EC2 인스턴스에서 백업 복원

```
# Setup environment variables
S3_BUCKET=S3 bucket name from step 2.1
PUPPET_BACKUP=Puppet backup file name from step 3.2

# download backup
aws s3 cp s3://$S3_BUCKET/tmp/puppet-backup/$PUPPET_BACKUP

# Prepare Puppet Enterprise backup to remove OpsWorks metadata
mkdir output
tar -xf $PUPPET_BACKUP -C output/
cd output/
rm -f opt/puppetlabs/facter/facts.d/opsworks.json
tar -cf ../$PUPPET_BACKUP *
cd ..
rm -rf output/

# Restore from backup
PATH=$PATH:/opt/puppetlabs/puppet/bin/
puppet-backup restore $PUPPET_BACKUP
puppet agent -t
```

https://##### ### IPv4에서 복원된 EC2 인스턴스의 Puppet 콘솔에 액세스할 수 있습니다. EC2 콘솔의 인스턴스 세부 정보 페이지에서 퍼블릭 IPv4 DNS를 찾을 수 있습니다. 로그인 자격 증명은 OpsWorks for Puppet Enterprise 서버에 액세스할 때 사용하는 것과 동일한 자격 증명입니다.

다음 단계를 위해 SSH 또는 세션 관리자 세션을 열어 둘 수 있습니다.

7단계: Puppet 라이선스 구성

Puppet 웹 사이트의 단계에 따라 라이선스를 구성하세요.

다음 단계를 위해 SSH 또는 세션 관리자 세션을 열어 둘 수 있습니다.

8단계: 노드 마이그레이션

OpsWorks for Puppet Enterprise 서버에서 지원하는 도메인에는 두 가지 유형이 있습니다.

- BYODC(자체 도메인 및 인증서 사용)
- OpsWorks Endpoint

8.1단계: BYODC의 경우(자체 도메인 및 인증서 사용)

이러한 노드의 경우 DNS 공급자의 사용자 지정 도메인이 새 EC2 인스턴스의 퍼블릭 IPv4 DNS 또는 퍼블릭 IPv4 주소를 가리키도록 지정하기만 하면 됩니다.

8.2단계: OpsWorks 엔드포인트의 경우

OpsWorks 엔드포인트의 경우 Puppet 설명서에서는 노드에서 Puppet 에이전트를 제거한 다음 새로 복원된 Puppet Enterprise 서버를 사용하여 Puppet 에이전트를 설치할 것을 권장합니다.

> **Note**
>
> Puppet에는 에이전트 노드를 이동하는 자동 절차가 없지만 Puppet 커뮤니티 구성원이 자동화된 노드 마이그레이션을 수행하기 위해 Puppet Forge 웹 사이트에 게시한 몇 가지 모듈이 있습니다. 이러한 모듈에는 `pe_migrate` 모듈과 다른 작성자의 두 번째 마이그레이션 모듈이 포함됩니다. Forge 모듈에 명시적으로 명시되어 있지 않는 한 Puppet Forge 웹 사이트의 모듈은 Puppet 또는 OpsWorks에서 지원하지 않습니다. 이러한 모듈은 주의해서 사용하고 널리 사용하기 전에 테스트해 보는 것이 좋습니다.

다음 섹션에서는 Linux 인스턴스에서 Puppet 에이전트를 제거하고 다시 설치하는 단계를 제공합니다.

주제

- 8.2.1단계: Puppet 서버에서 제거 프로그램 복사
- 8.2.2단계: 제거 프로그램을 다운로드하고 노드에서 실행
- 8.2.3단계: 노드에 Puppet 에이전트 재설치

8.2.1단계: Puppet 서버에서 제거 프로그램 복사

에이전트를 제거하기 전에 노드의 IAM 인스턴스 프로필이 S3 ReadOnly 권한을 제공하는지 확인하세요.

다음 명령을 실행하여 제거 프로그램을 Puppet 서버에서 S3 버킷으로 복사합니다.

```
aws s3 cp \
   /opt/puppetlabs/bin/puppet-enterprise-uninstaller \
   s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller
```

명령을 실행한 후 Puppet 서버의 SSH 또는 세션 관리자 세션에서 로그아웃할 수 있습니다.

8.2.2단계: 제거 프로그램을 다운로드하고 노드에서 실행

SSH를 사용하여 노드에 연결합니다. 노드가 EC2 인스턴스인 경우 SSH 대신 EC2 콘솔에서 세션 관리자를 사용할 수 있습니다.

```
sudo -i

S3_BUCKET=aws-opsworks-cm-abcdefg-uuhtyn6messn
aws s3 cp s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller /opt/puppetlabs/bin/
chmod 700 /opt/puppetlabs/bin/puppet-enterprise-uninstaller
/opt/puppetlabs/bin/puppet-enterprise-uninstaller
```

다음 단계를 위해 SSH 또는 세션 관리자 세션을 열어 둘 수 있습니다.

8.2.3단계: 노드에 Puppet 에이전트 재설치

다음 단계를 완료하여 Puppet 에이전트를 노드에 다시 설치합니다.

주제

- 8.2.3.1단계: 올바른 구성으로 Puppet 에이전트 설치
- 8.2.3.2단계: Puppet 콘솔에서 인증서 수락
- 8.2.3.3단계: 노드를 Puppet Enterprise 서버에 체크인합니다.

8.2.3.1단계: 올바른 구성으로 Puppet 에이전트 설치

다음 명령을 실행하여 Puppet 에이전트를 설치합니다.

```
curl -k https://Public_IPv4_DNS:8140/packages/current/install.bash | bash
```

8.2.2.3단계에서 SSH 또는 세션 관리자 세션을 계속 열어 둘 수 있습니다.

8.2.3.2단계: Puppet 콘솔에서 인증서 수락

1. `https://Public_IPv4_DNS`에서 Puppet 서버 콘솔로 이동합니다.
2. 인증서를 선택한 다음 서명되지 않은 인증서를 선택합니다.
3. 수락을 선택하여 Puppet 에이전트의 인증서에 서명합니다.

8.2.3.3단계: 노드를 Puppet Enterprise 서버에 체크인합니다.

노드에서 다음 명령을 실행하여 서버에 체크인합니다.

```
puppet agent -t
```

이제 Puppet 서버의 콘솔에 해당 노드가 표시될 것입니다.

9단계: OpsWorks for Puppet Enterprise 서버를 삭제합니다

OpsWorks 콘솔 또는를 사용하여 OpsWorks for Puppet Enterprise 서버를 AWS CLI 삭제할 수 있습니다.

OpsWorks 콘솔을 사용하여 서버를 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. 탐색 창에서 Puppet Enterprise 서버를 선택합니다.
3. Puppet Enterprise 서버 페이지에서 백업하려는 서버를 선택합니다.
4. 작업에서 Puppet Enterprise 서버 삭제를 선택합니다.

를 사용하여 서버를 삭제하려면 AWS CLI

다음 명령을 실행합니다.

```
aws opsworks-cm delete-server \
   --server-name server-name \
   --region region
```

를 사용하여 OpsWorks for Puppet Enterprise API 호출 로깅 AWS CloudTrail

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

OpsWorks for Puppet Enterprise는 OpsWorks for Puppet Enterprise에서 사용자, 역할 또는 AWS CloudTrail서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 OpsWorks for Puppet Enterprise 콘솔의 호출 및 OpsWorks for Puppet Enterprise API에 대한 코드 호출을 포함하여 OpsWorks for Puppet Enterprise에 대한 모든 API 호출을 이벤트로 캡처합니다. 추적을 생성하면, OpsWorks for Puppet Enterprise 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 전달할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 OpsWorks for Puppet Enterprise에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 OpsWorks for Puppet Enterprise 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. OpsWorks for Puppet Enterprise에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

OpsWorks for Puppet Enterprise에 대한 이벤트를 포함하여 AWS 계정의 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 트레일 생성 개요

- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 OpsWorks for Puppet Enterprise 작업은 CloudTrail에서 로깅되며 OpsWorks for Puppet Enterprise API Reference에 문서화되어 있습니다. 예를 들어 CreateServer, CreateBackup 및 DescribeServers 작업에 대한 호출은 CloudTrail 로그 파일에 항목을 생성합니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청을 했는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

OpsWorks for Puppet Enterprise 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음 예제는 OpsWorks for Puppet Enterprise `CreateServer` 작업에 대한 CloudTrail 로그 항목을 표시합니다.

```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
```

```
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"test-puppet-server",
    "engineModel":"Single",
    "engine":"Puppet",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"test-puppet-server-xxxx8u4390xo6pd9.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Puppet",
        "instanceType":"t2.medium",
        "serverName":"test-puppet-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/test-puppet-
server/8ezz7f6z-e91f-4z10-89z5-8c6219zzz09f",
        "engineModel":"Single",
```

```
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"PUPPET_ADMIN_PASSWORD","value":"*** Redacted ***"},
            {"name":"PUPPET_API_CA_CERT","value":"*** Redacted ***"},
            ],
        "engineVersion":"12.11.1",
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7z64z9-d394-12ug-8081-7zz0386fbcb6",
"eventID":"8z7z18dz-6z90-47bz-87cf-e8346428zzz3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```


OpsWorks for Puppet Enterprise 문제 해결

> **Important**
>
> 이 AWS OpsWorks for Puppet Enterprise 서비스는 2024년 3월 31일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에는 몇 가지 일반적인 OpsWorks for Puppet Enterprise 문제와 이러한 문제의 해결 방법이 제시되어 있습니다.

주제

- 일반적인 문제 해결 팁
- 구체적 오류 해결
- 추가 도움말 및 지원

일반적인 문제 해결 팁

Puppet 마스터를 생성하거나 사용할 수 없는 경우, 오류 메시지나 로그를 보고 문제 해결에 도움을 받을 수 있습니다. 다음 작업은 Puppet 마스터 문제 해결 시 일반적인 첫 단계를 설명합니다. 구체적인 오류와 해결책에 대해서는 이 주제의 구체적 오류 해결 단원을 참조하세요.

- Puppet 마스터가 시작되지 않는 경우, OpsWorks for Puppet Enterprise 콘솔을 사용하여 오류 메시지를 확인합니다 Puppet 마스터 속성 페이지의 상단에는 서버 시작 및 실행과 관련된 오류 메시지가 표시됩니다. 오류는 Puppet 마스터를 생성하는 데 사용되는 OpsWorks for Puppet Enterprise AWS CloudFormation또는 Amazon EC2 서비스에서 발생할 수 있습니다. 속성 페이지에서는 실행 중인 서버에서 발생하는 이벤트도 볼 수 있으며, 여기에 실패 이벤트 메시지가 포함될 수 있습니다.
- EC2 문제를 해결하려면 SSH를 사용하여 서버의 인스턴스에 연결하고 로그를 확인합니다. EC2 인스턴스 로그는 `/var/log/aws/opsworks-cm` 디렉터리에 저장됩니다. 이러한 로그는 OpsWorks for Puppet Enterprise가 Puppet 마스터를 시작하는 동안 명령 출력을 캡처합니다.

구체적 오류 해결

주제

- 서버가 연결 끊김 상태입니다.
- "requested configuration is currently not supported" 메시지와 함께 서버 생성이 실패합니다
- 서버의 Amazon EC2 인스턴스를 생성할 수 없습니다.
- 서비스 역할 오류로 서버가 생성되지 않습니다
- 탄력적 IP 주소 제한 초과
- 무인 노드 연결이 실패합니다
- 시스템 유지 관리 실패

서버가 연결 끊김 상태입니다.

문제: 서버 상태가 연결 끊김으로 표시됩니다.

원인: 외부의 엔터티 AWS OpsWorks 가 OpsWorks for Puppet Enterprise 서버 또는 지원 리소스를 변경할 때 가장 일반적으로 발생합니다.는 백업 생성, 운영 체제 패치 적용 또는 Puppet 업데이트와 같은 유지 관리 작업을 처리하기 위해 연결 손실 상태의 Puppet Enterprise 서버에 연결할 수 AWS OpsWorks 없습니다. 따라서 서버에 중요한 업데이트가 누락되거나 보안 문제가 발생하기 쉽거나 예상대로 작동하지 않을 수 있습니다.

해결 방법: 다음 단계를 수행하여 서버 연결을 복원하세요.

1. 서비스 역할에 필요한 모든 권한이 있는지 확인하세요.

   a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 서비스 역할의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 서비스 역할이 열립니다.

   b. 권한 탭에서 권한 정책 목록에 `AWSOpsWorksCMServiceRole`이(가) 있는지 확인합니다. 목록에 없는 경우 `AWSOpsWorksCMServiceRole` 관리형 정책을 역할에 수동으로 추가하세요.

   c. 신뢰 관계 탭에서 서비스 역할에 `opsworks-cm.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

2. 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.

   a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 인스턴스 프로파일의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 인스턴스 프로파일이 열립니다.

   b. 권한 탭에서 권한 정책 목록에 `AmazonEC2RoleforSSM` 및 `AWSOpsWorksCMInstanceProfileRole`이(가) 둘 다 있는지 확인합니다. 둘 중 하나 또는 둘 다 목록에 없는 경우 이러한 관리형 정책을 역할에 수동으로 추가하세요.

   c. 신뢰 관계 탭에서 서비스 역할에 `ec2.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

3. Amazon EC2 콘솔에서 PupsWorks for Puppet Enterprise 서버 지역과 동일한 지역에 있는지 확인한 다음, 서버에서 사용 중인 EC2 인스턴스를 다시 시작합니다.

   a. 이름이 `aws-opsworks-cm-instance-`*`server-name`*인 EC2 인스턴스를 선택합니다.

   b. 인스턴스 상태 메뉴에서 인스턴스 재부팅을 선택합니다.

   c. 서버가 다시 시작되고 완전히 온라인 상태가 될 때까지 최대 15분이 걸릴 수 있습니다.

4. OpsWorks for Puppet Enterprise 콘솔의 서버 세부 정보 페이지에서 서버 상태가 현재 정상인지 확인합니다.

이전 단계를 수행한 후에도 서버 상태가 여전히 연결 끊김으로 표시되면 다음 중 하나를 시도해 보세요.

- 새 서버를 만들고 원본을 삭제하여 서버를 교체하세요. 현재 서버의 데이터가 중요한 경우 최근 백업에서 서버를 복원하고 데이터가 최신 상태인지 확인한 다음 응답이 없는 원래 서버를 삭제하세요.
- AWS 고객 지원에 문의하세요.

"requested configuration is currently not supported" 메시지와 함께 서버 생성이 실패합니다

문제: Puppet Enterprise 서버를 생성하려고 하지만 "The requested configuration is currently not supported. Please check the documentation for supported configurations."와 비슷한 오류 메시지와 함께 서버 생성이 실패합니다.

원인: Puppet 마스터에 대해 지원되지 않는 인스턴스 유형을 지정했을 수 있습니다. 전용 인스턴스용 VPC처럼 기본이 아닌 테넌시가 있는 VPC에서 Puppet 서버를 생성하는 경우, 지정된 VPC 내부의 모든 인스턴스도 전용 또는 호스트 테넌시여야 합니다. t2와 같은 일부 인스턴스 유형은 기본 테넌시에서만 사용할 수 있기 때문에 Puppet 마스터 인스턴스 유형은 지정한 VPC에서 지원 불가능할 수 있으며, 서버 생성이 실패합니다.

해결 방법: 기본이 아닌 테넌시가 있는 VPC를 선택하는 경우, 전용 테넌시를 지원할 수 있는 m4 인스턴스 유형을 사용하세요.

서버의 Amazon EC2 인스턴스를 생성할 수 없습니다.

문제: 다음과 비슷한 오류 메시지와 함께 서버 생성이 실패했습니다. "The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration."

원인: 가장 가능성 높은 원인은 EC2 인스턴스에 네트워크 액세스 권한이 없기 때문입니다.

해결 방법: 인스턴스에 아웃바운드 인터넷 액세스 권한이 있고 AWS 서비스 에이전트가 명령을 실행할 수 있는지 확인합니다. VPC(단일 퍼블릭 서브넷이 포함된 VPC)에서 [DNS 확인]이 활성화되어 있고, 서브넷에서 [퍼블릭 IP 자동 할당] 설정이 활성화되어 있어야 합니다.

서비스 역할 오류로 서버가 생성되지 않습니다

문제: "Not authorized to perform sts:AssumeRole"이라는 오류 메시지와 함께 서버 생성이 실패합니다.

원인: 이 문제는 사용 중인 서비스 역할에 새 서버를 생성할 수 있는 적절한 권한이 없을 때 발생할 수 있습니다.

해결 방법: OpsWorks for Puppet Enterprise 콘솔을 열고 콘솔을 사용하여 새 서비스 역할과 인스턴스 프로파일 역할을 생성합니다. 자체 서비스 역할을 사용하려는 경우, [AWSOpsWorksCMServiceRole] 정책을 역할에 연결합니다. opsworks-cm.amazonaws.com이 역할의 신뢰 관계에 있는 서비스에 등재되어 있는지 확인하세요. Puppet 마스터에 연결된 서비스 역할에 AWSOpsWorksCMServiceRole 관리형 정책이 연결되어 있는지 확인합니다.

탄력적 IP 주소 제한 초과

문제: "The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached"라는 오류 메시지와 함께 서버 생성이 실패합니다.

원인: 이 문제는 계정이 탄력적 IP(EIP) 주소 최대 수를 사용한 경우에 발생합니다. 기본 EIP 주소 한도는 5입니다.

해결 방법: 기존 EIP 주소를 해제하거나 계정이 적극적으로 사용하지 않는 주소를 삭제하거나 AWS 고객 지원에 문의하여 계정과 연결된 EIP 주소의 제한을 늘릴 수 있습니다.

무인 노드 연결이 실패합니다

문제: 새 Amazon EC2 노드의 무인 또는 자동 연결이 실패합니다. Puppet 마스터에 추가되었어야 할 노드가 Puppet Enterprise 대시보드에 나타나지 않습니다.

원인: 이 문제는 `opsworks-cm` API 호출이 새 EC2 인스턴스와 통신하는 것을 허용하는 인스턴스 프로파일로 IAM 역할이 설정되어 있지 않을 때 발생할 수 있습니다.

해결 방법: OpsWorks for Puppet Enterprise에서 자동으로 노드 추가에 설명된 대로 `AssociateNode` 및 `DescribeNodeAssociationStatus` API 호출이 EC2에서 작동하도록 허용하는 정책을 EC2 인스턴스 프로파일에 연결하세요.

시스템 유지 관리 실패

AWS OpsWorks CM 는 보안 업데이트를 포함하여 최신 AWS테스트를 거친 버전의 Puppet Server가 항상 OpsWorks for Puppet Enterprise 서버에서 실행되도록 매주 시스템 유지 관리를 수행합니다. 어떤 이유로든 시스템 유지 관리가 실패하면에서 실패를 AWS OpsWorks CM 알립니다. 시스템 유지 관리에 대한 자세한 내용은 OpsWorks for Puppet Enterprise 시스템 유지 관리 섹션을 참조하세요.

이 섹션에서는 가능한 실패 원인을 설명하고 해결 방법을 제안합니다.

주제

- 서비스 역할 또는 인스턴스 프로파일 오류로 인해 시스템 유지 관리가 불가능합니다.

서비스 역할 또는 인스턴스 프로파일 오류로 인해 시스템 유지 관리가 불가능합니다.

문제: “STS:AssumeRole 수행 권한이 없음”이라는 오류 메시지 또는 권한에 대한 유사한 오류 메시지와 함께 시스템 유지 관리가 실패합니다.

원인: 사용 중인 서비스 역할 또는 인스턴스 프로파일에 서버에서 시스템 유지 관리를 수행할 수 있는 적절한 권한이 없는 경우 이 문제가 발생할 수 있습니다.

해결 방법: 서비스 역할과 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.

1. 서비스 역할에 필요한 모든 권한이 있는지 확인하세요.
   a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 서비스 역할의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 서비스 역할이 열립니다.
   b. 권한 탭에서 서비스 역할에 `AWSOpsWorksCMServiceRole`이(가) 연결되어 있는지 확인합니다. `AWSOpsWorksCMServiceRole`이(가) 목록에 없는 경우 이 정책을 역할에 추가하세요.
   c. opsworks-cm.amazonaws.com이 역할의 신뢰 관계에 있는 서비스에 등재되어 있는지 확인하세요. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.
2. 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.
   a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 인스턴스 프로파일의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 인스턴스 프로파일이 열립니다.
   b. 권한 탭에서 권한 정책 목록에 `AmazonEC2RoleforSSM` 및 `AWSOpsWorksCMInstanceProfileRole`이(가) 둘 다 있는지 확인합니다. 둘 중 하나 또는 둘 다 목록에 없는 경우 이러한 관리형 정책을 역할에 수동으로 추가하세요.
   c. 신뢰 관계 탭에서 서비스 역할에 `ec2.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

추가 도움말 및 지원

이 주제에 특정 문제가 설명되어 있지 않거나 이 주제의 제안을 시도했지만 여전히 문제가 계속되는 경우, AWS OpsWorks 포럼을 방문하세요.

AWS Support Center를 방문할 수도 있습니다. AWS 지원 센터는 AWS 지원 사례를 생성하고 관리하기 위한 허브입니다. AWS 지원 센터에는 포럼, 기술 FAQs, 서비스 상태 및 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다 AWS Trusted Advisor.

AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

AWS OpsWorks for Chef Automate 를 사용하면 AWS에서 Chef Automate 서버를 실행할 수 있습니다. 몇 분 안에 Chef 서버를 프로비저닝할 수 있으며가 운영, 백업, 복원 및 소프트웨어 업그레이드를 AWS OpsWorks for Chef Automate 처리하도록 할 수 있습니다. AWS OpsWorks for Chef Automate 를 사용하면 Chef 서버를 관리하는 대신 핵심 구성 관리 작업에 집중할 수 있습니다.

Chef Automate 서버는 `chef-client`에게 노드에서 실행할 Chef 레시피를 지시하여 환경 내 노드의 구성을 관리하고, 노드에 대한 정보를 저장하며, Chef 쿡북의 중앙 리포지토리를 제공합니다. AWS OpsWorks for Chef Automate 는 Chef Automate의 프리미엄 기능인 Chef Infra 및 Chef InSpec이 포함된 Chef 서버를 제공합니다.

AWS OpsWorks for Chef Automate 서버는 Amazon Elastic Compute Cloud 인스턴스에서 실행됩니다. AWS OpsWorks for Chef Automate 서버는 최신 버전의 Amazon Linux(Amazon Linux 2)를 실행하도록 구성됩니다. 이 버전의 Chef Automate에 수행된 변경 사항에 대한 자세한 내용은 Chef Automate 출시 정보를 참조하세요. 다음 표에서는 AWS OpsWorks for Chef Automate 서버에 설치된 Chef 구성 요소를 설명합니다.

구성 요소 이름	설명	AWS OpsWorks for Chef Automate 서버에 설치된 버전
Chef Automate	Chef Automate은 연속 배포를 위한 자동화된 워크플로우와 웹 기반 관리 콘솔에서 관리형 노드에 대한 통찰을 제공하는 엔터프라이즈 서버 소프트웨어 패키지입니다. Chef Automate는 Chef Infra를 포함하여 인프라 자동화를 제공하	2.0

구성 요소 이름	설명	AWS OpsWorks for Chef Automate 서버에 설치된 버전
	고, Chef InSpec을 포함하여 보안 및 규정 준수 정보와 적용을 제공하며, Chef Habitat를 포함하여 자동화된 배포를 제공합니다. Chef Automate에 대한 자세한 내용은 Chef 웹 사이트의 Chef Automate를 참조하세요.	
Chef Infra	이전에 Chef Server라고 했던 Chef Infra Server는 Chef Infra Client(`chef-client`) 에이전트를 사용하여 관리형 노드에 구성을 연속으로 적용하여 원하는 상태를 유지합니다. Infra에 대한 자세한 내용은 Chef 웹 사이트의 Chef Infra를 참조하세요.	12.x

구성 요소 이름	설명	AWS OpsWorks for Chef Automate 서버에 설치된 버전
Chef InSpec	Chef InSpec은 소프트웨어 엔지니어, 운영 및 보안 엔지니어 간에 공유할 수 있는 보안 및 규정 준수 규칙을 설명합니다. 규정 준수, 보안 및 기타 정책 요구 사항은 `chef-client` 에이전트가 관리형 노드에 대해 실행할 수 있는 자동화된 테스트에 대한 프레임워크를 구성하여 표준의 일관적인 적용을 보장합니다. InSpec에 대한 자세한 내용은 Chef 웹 사이트의 Chef InSpec을 참조하세요.	3.9.0

AWS OpsWorks for Chef Automate 서버와 관련된 노드에서 지원되는 최소 `chef-client` 버전은 13.x입니다. 최소 14.10.9 또는 가장 최신의 안정적인 `chef-client` 버전을 실행하는 것이 좋습니다.

시스템 유지관리는 Chef 소프트웨어의 새 마이너 버전이 나오면 AWS 테스트를 통과하는 즉시 서버에서 Chef Automate 및 Chef Server의 마이너 버전을 자동으로 업데이트하도록 설계되어 있습니다. AWS가 광범위한 테스트를 실시하여 Chef 업그레이드가 생산 준비가 되었는지, 기존 고객 환경에 지장을 주지 않는지 확인하므로 Chef 소프트웨어 릴리스 후 Chef Automate 서버의 기존 OpsWorks에서 애플리케이션을 사용할 수 있게 되기까지 시간이 지연될 수 있습니다. 또한 시스템 유지 관리를 통해 서버를 최신 버전의 Amazon Linux로 업그레이드할 수 있습니다.

지원되는 운영 체제를 실행하고 AWS OpsWorks for Chef Automate 서버에 대한 네트워크 액세스 권한이 있는 모든 온프레미스 컴퓨터 또는 EC2 인스턴스를 연결할 수 있습니다. 관리할 노드에 지원되는 운영 체제 목록은 Chef 웹사이트를 참조하세요. `chef-client` 에이전트 소프트웨어는 Chef 서버를 사용하여 관리할 노드에 설치됩니다.

주제

- 에 대한 리전 지원 AWS OpsWorks for Chef Automate
- AWS OpsWorks for Chef Automate 수명 종료 FAQs

- AWS OpsWorks for Chef Automate 서버를 Chef Automate 2로 업그레이드
- 시작하기 AWS OpsWorks for Chef Automate
- 를 사용하여 AWS OpsWorks for Chef Automate 서버 생성 AWS CloudFormation
- 사용자 지정 도메인을 사용하도록 AWS OpsWorks for Chef Automate 서버 업데이트
- AWS OpsWorks for Chef Automate 서버의 스타터 키트 재생성
- AWS OpsWorks for Chef Automate 리소스에서 태그 작업
- AWS OpsWorks for Chef Automate 서버 백업 및 복원
- 의 시스템 유지 관리 AWS OpsWorks for Chef Automate
- 의 규정 준수 스캔 AWS OpsWorks for Chef Automate
- AWS OpsWorks for Chef Automate 서버에서 노드 연결 해제
- AWS OpsWorks for Chef Automate 서버 삭제
- Chef Automate 대시보드 자격 증명 재설정
- 를 사용하여 AWS OpsWorks for Chef Automate API 호출 로깅 AWS CloudTrail
- 문제 해결 AWS OpsWorks for Chef Automate

에 대한 리전 지원 AWS OpsWorks for Chef Automate

다음 리전 엔드포인트는 Chef 서버와 동일한 리전 엔드포인트에서 인스턴스 프로파일, 사용자 및 서비스 역할과 같이 Chef 서버와 연결된 리소스를 server AWS OpsWorks for Chef Automate . AWS OpsWorks for Chef Automate create합니다. Chef 서버가 VPC에 속해야 합니다. VPC를 생성하여 사용하거나, 기존 VPC를 사용하거나, 기본 VPC를 사용할 수 있습니다.

- US East (Ohio) Region
- 미국 동부(버지니아 북부) 리전
- 미국 서부(캘리포니아 북부) 리전
- US West (Oregon) Region
- 아시아 태평양(도쿄) 리전
- 아시아 태평양(싱가포르) 리전
- 아시아 태평양(시드니) 리전
- Europe (Frankfurt) Region

- Europe (Ireland) Region

AWS OpsWorks for Chef Automate 수명 종료 FAQs

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다.

주제

- 이번 서비스 종료로 인해 기존 사용자는 어떤 영향을 받게 됩니까?
- 아무 조치도 취하지 않으면 어떻게 됩니까?
- 어떤 대안으로 전환할 수 있나요?
- 이 서비스는 여전히 신규 고객을 받고 있나요?
- 수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?
- 어떤 수준의 기술 지원을 이용할 수 있나요?
- 저는 현재 OpsWorks for Chef Automate의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요?
- 내년에 주요 기능이 릴리스될 예정인가요?

이번 서비스 종료로 인해 기존 사용자는 어떤 영향을 받게 됩니까?

기존 고객은 OpsWorks for Puppet Enterprise의 서비스 종료일인 2024년 5월 5일까지 영향을 받지 않습니다. 서비스 종료일이 지나면 고객은 더 이상 OpsWorks 콘솔 또는 API를 사용하여 서버를 관리할 수 없습니다.

아무 조치도 취하지 않으면 어떻게 됩니까?

2024년 5월 5일부터 OpsWorks 콘솔 또는 API를 사용하여 서버를 관리할 수 없습니다. 이때 당사는 백업 또는 유지 관리와 같은 서버의 지속적인 관리 기능 수행을 중단합니다. 고객에게 미치는 영향을 제한하기 위해 Chef Automate 서버를 백업하는 모든 실행 중인 EC2 인스턴스는 그대로 둘 것입니다. 하지만 Chef와의 OpsWorks for Chef Automate 서비스 계약에 따라 사용이 더 이상 적용(또는 청구)되지 않으므로 라이선스는 더 이상 유효하지 않습니다. 새 라이선스를 받으려면 Chef에 문의해야 합니다.

Chef에 문의할 때는 OpsWorks for Chef Automate의 기존 고객이며 OpsWorks에서 전환하고 있다고 알려주세요.

어떤 대안으로 전환할 수 있나요?

AWS 및 Progress Chef는 완전 관리형 Chef Automate 서비스의 이점을 계속 활용할 수 있도록 새 Chef SaaS 제품으로 마이그레이션하는 것이 좋습니다. Chef SaaS를 시작하려면 Chef에 문의하여 Chef SaaS 계정을 설정하고 데이터와 노드를 전환하는 방법에 대한 설명서를 받을 수 있습니다.

사용자가 제어하는 AWS 계정의 EC2 인스턴스에서 Chef Automate를 실행하려는 이유로 Chef SaaS가 사용자의 요구 사항을 충족하지 못하는 경우 Chef는 AWS Marketplace BYOL(Bring Your Own License) 모델 및 EC2에서 자체 호스팅을 비롯한 여러 옵션을 제공합니다. 이러한 전환을 실행하는 방법에 대한 자세한 내용은 Progress Chef에 문의할 수 있습니다.

이 서비스는 여전히 신규 고객을 받고 있나요?

아니요. AWS OpsWorks Chef Automate는 더 이상 신규 고객을 받지 않습니다.

수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?

예. API와 콘솔의 수명이 종료되어 2024년 5월 5일부터 모든 AWS 리전에서 사용할 수 없게 됩니다. Chef Automate를 사용할 수 AWS 리전 있는 AWS OpsWorks 에 대한 자세한 내용은 AWS 리전 서비스 목록을 참조하세요.

어떤 수준의 기술 지원을 이용할 수 있나요?

AWS 는 오늘부터 수명 종료 날짜까지 고객이 보유한 것과 동일한 수준의 OpsWorks for Chef Automate 지원을 계속 제공할 것입니다. 질문이나 우려 사항이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다. 전환 지원을 받으려면 고객이 Progress Chef에 문의하는 것이 좋습니다.

저는 현재 OpsWorks for Chef Automate의 고객이며 이전에 이 서비스를 사용하지 않았던 계정으로 서버를 시작해야 합니다. 제가 이 작업을 수행할 수 있나요?

예외적인 상황이 아니라면 보통은 안 됩니다. 특별한 상황이 있는 경우 AWS re:Post에서 AWS Support 또는 AWS Premium Support를 통해 팀에 연락하여 이에 대한 세부 정보 및 정당화를 문의하면 요청을 검토하겠습니다.

내년에 주요 기능이 릴리스될 예정인가요?

아니요. 서비스 수명이 종료되는 시점이므로 새로운 기능은 출시되지 않을 예정입니다. 하지만 서비스 종료 날짜까지 계속해서 보안을 개선하고 예상대로 서버를 관리할 예정입니다.

AWS OpsWorks for Chef Automate 서버를 Chef Automate 2로 업그레이드

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

Chef Automate 2로 업그레이드하기 위한 사전 요구 사항

시작하기 전에 Chef Automate 2가 추가하는 새로운 기능과 Chef Automate 2가 지원하지 않는 기능을 이해하세요. Chef Automate 2의 새로운 기능 및 지원되지 않는 기능에 대한 자세한 내용은 Chef 웹사이트에서 Chef Automate 2 설명서를 참조하세요.

Chef Auto 1을 실행하는 서버는 2019년 11월 1일 이후에 유지 관리를 최소 1회 이상 성공적으로 실행했어야 업그레이드할 수 있습니다.

AWS OpsWorks for Chef Automate 서버의 유지 관리 작업과 마찬가지로 업그레이드 중에 서버가 오프라인 상태입니다. 업그레이드 프로세스 중에 최대 3시간의 가동 중지 시간을 계획해야 합니다.

Chef Automate 대시보드 웹사이트에서 이 서버에 대한 로그인 자격 증명이 필요합니다. 업그레이드가 완료되면 Chef Automate 대시보드에 로그인하여 노드 및 구성 정보가 변경되지 않았는지 확인해야 합니다.

> **Important**
>
> AWS OpsWorks for Chef Automate 서버를 Chef Automate 2로 업그레이드할 준비가 되면 여기의 지침만 사용하여 업그레이드하세요. 는 백업 생성과 같은 많은 업그레이드 프로세스를 AWS OpsWorks for Chef Automate 자동화하므로 Chef 웹 사이트의 업그레이드 지침을 따르지 마십시오.

업그레이드 프로세스 정보

업그레이드 프로세스 중에는 업그레이드를 시작하기 전과 업그레이드를 완료한 후에 서버가 백업됩니다. 다음 백업이 생성됩니다.

- Chef Automate 1(버전 12.17.33)을 계속 실행 중일 때 서버의 백업.
- 업그레이드가 완료되고 서버가 Chef Automate 2(버전 2019-08)를 실행 중일 때 서버의 백업.

업그레이드 프로세스는 Chef Automate 1을 실행할 때 서버가 사용했던 Amazon EC2 인스턴스를 종료합니다. Chef Automate 2 서버를 실행하기 위해 새 인스턴스가 생성됩니다.

Chef Automate 2로 업그레이드(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 AWS OpsWorks for Chef Automate를 선택합니다.
3. 서버를 선택하여 해당 속성 페이지를 봅니다. 페이지 상단의 파란색 배너는 서버를 Chef Auto 2로 업그레이드할 수 있는지 여부를 나타냅니다.

   > **Note**
   >
   > Chef Auto 1을 실행하는 서버는 2019년 11월 1일 이후에 유지 관리를 최소 1회 이상 성공적으로 실행했어야 업그레이드할 수 있습니다.

4. 서버를 업그레이드할 수 있는 경우 업그레이드 시작을 선택합니다.
5. 업그레이드에 최대 3시간이 소요됩니다. 업그레이드 프로세스 중에 속성 페이지에 서버 상태가 유지 관리 중으로 표시됩니다.
6. 업그레이드가 완료되면 속성 페이지에 Automate 2로 업그레이드 완료와 유지 관리 완료라는 두 개의 메시지가 표시됩니다. 서버 상태는 정상이어야 합니다.
7. 기존 자격 증명을 사용하여 Chef Automate 대시보드에 로그인하고 노드가 올바르게 보고되는지 확인합니다.

Chef Automate 2(CLI)로 업그레이드

1. (선택 사항) 업그레이드할 수 있는 AWS OpsWorks for Chef Automate 서버가 확실하지 않은 경우 다음 명령을 실행합니다. 기본 AWS 리전과 다른 AWS 리전의 AWS OpsWorks for Chef Automate 서버를 나열하려면 `--region` 파라미터를 추가해야 합니다.

   ```
   aws opsworks-cm describe-servers
   ```

   결과에서 `CHEF_MAJOR_UPGRADE_AVAILABLE` 속성의 `true` 값을 찾습니다. 이는 서버를 Chef Auto 2로 업그레이드할 수 있음을 나타냅니다. 업그레이드할 수 있는 AWS OpsWorks for Chef Automate 서버의 이름을 기록해 둡니다.

2. *`server_name`*을 AWS OpsWorks for Chef Automate 서버 이름으로 바꾸어 다음 명령을 실행합니다. 일상적인 시스템 유지 관리를 수행하는 대신 Chef Automate 2로 업그레이드하려면 명령과 같이 `CHEF_MAJOR_UPGRADE` 엔진 속성을 추가합니다. 대상 서버가 기본 AWS 리전에 없는 경우 `--region` 파라미터를 추가합니다. 명령당 하나의 서버만 업그레이드할 수 있습니다.

   ```
   aws opsworks-cm start-maintenance --server-name server_name --engine-attributes
    Name=CHEF_MAJOR_UPGRADE,Value=true --region region
   ```

   가 어떤 이유로든 서버를 업그레이드할 AWS OpsWorks for Chef Automate 수 없는 경우이 명령으로 인해 검증 예외가 발생합니다.

3. 업그레이드에 최대 3시간이 소요됩니다. 다음 명령을 실행하여 업그레이드 상태를 주기적으로 확인할 수 있습니다.

   ```
   aws opsworks-cm describe-servers --server-name server_name
   ```

   결과에서 `Status` 값을 찾습니다. `Status`가 `UNDER_MAINTENANCE`면 업그레이드가 아직 진행 중임을 의미합니다. 성공적으로 업그레이드하면 다음과 유사한 메시지가 반환됩니다.

   ```
   2019/10/24 00:27:56 UTC          Successfully upgraded to Automate 2.
   2019/10/23 23:50:38 UTC          Upgrading Chef server from Automate 1 to Automate
    2
   ```

   업그레이드에 실패한 경우는 자동으로 서버를 Chef Automate 1로 AWS OpsWorks for Chef Automate 롤백합니다.

업그레이드가 성공했지만 서버가 업그레이드 이전과 동일하게 작동하지 않는 경우(예: 관리형 노드가 보고되지 않는 경우) 수동으로 서버를 롤백할 수 있습니다. 수동 롤백 정보는 AWS OpsWorks for Chef Automate 서버를 Chef Automate 1로 롤백(CLI) 단원을 참조하세요.

AWS OpsWorks for Chef Automate 서버를 Chef Automate 1로 롤백(CLI)

업그레이드 프로세스가 실패하면가 AWS OpsWorks for Chef Automate 자동으로 서버를 Chef Automate 1로 롤백합니다. 업그레이드가 성공했지만 서버가 업그레이드 전과 동일하게 작동하지 않는 경우를 사용하여 수동으로 Chef Automate 1로 AWS OpsWorks for Chef Automate 서버를 롤백할 수 있습니다 AWS CLI.

1. 다음 명령을 실행하여 업그레이드를 시도하기 전에 서버에서 수행한 마지막 백업의 `BackupId`를 표시합니다. 서버가 기본 AWS 리전과 다른 AWS 리전에 있는 경우 `--region` 파라미터를 추가합니다.

   ```
   aws opsworks-cm describe-backups server_name
   ```

   백업 ID는 *`ServerName-yyyyMMddHHmmssSSS`* 형식입니다. 결과에서 다음 Chef Automate 1 속성을 찾습니다.

   ```
   "Engine": "Chef"
   "EngineVersion": "12.17.33"
   ```

2. 1단계에서 반환한 백업 ID를 `--backup-id`의 값으로 사용하여 다음 명령을 실행합니다.

   ```
   aws opsworks-cm restore-server --server-name server_name --backup-id ServerName-
   yyyyMMddHHmmssSSS
   ```

   서버에 저장된 데이터 양에 따라 서버를 복원하는 데 20분에서 3시간이 소요됩니다. 복원 작업 중에 서버의 상태는 `RESTORING`입니다. 이 상태는의 서버 속성 페이지에 표시되고 describe-servers 명령 결과에 AWS Management Console반환됩니다.

3. 복원이 완료되면 복원 완료 메시지가 콘솔에 표시됩니다. AWS OpsWorks for Chef Automate 서버가 온라인 상태이며 업그레이드 프로세스를 시작하기 전과 동일합니다.

참고

- 의 시스템 유지 관리 AWS OpsWorks for Chef Automate
- 백업에서 AWS OpsWorks for Chef Automate 서버 복원
- AWS OpsWorks API 참조의 DescribeServers
- AWS OpsWorks API 참조의 StartMaintenance

시작하기 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

AWS OpsWorks for Chef Automate 를 사용하면에서 Chef Automate 서버를 실행할 수 있습니다 AWS. 약 15분 안에 Chef 서버를 프로비저닝할 수 있습니다.

2021년 5월 3일부터 일부 Chef Automate 서버 속성을에 AWS OpsWorks for Chef Automate 저장합니다 AWS Secrets Manager. 자세한 내용은 와 통합 AWS Secrets Manager 단원을 참조하십시오.

다음 연습은에서 첫 번째 Chef 서버를 생성하는 데 도움이 됩니다 AWS OpsWorks for Chef Automate.

사전 조건

시작하기 전에 다음과 같은 사전 조건을 완료해야 합니다.

주제

- VPC 설정
- 사용자 지정 도메인 사용을 위한 사전 조건(선택 사항)
- EC2 키 페어 설정(선택 사항)

VPC 설정

AWS OpsWorks for Chef Automate 서버는 Amazon Virtual Private Cloud에서 작동해야 합니다. 이 서버를 기존 VPC에 추가하거나 기본 VPC를 사용하거나 새 VPC를 생성해 서버를 포함시킬 수 있습니다. Amazon VPC 및 새 VPC 생성 방법에 대한 자세한 내용은 Amazon VPC 시작하기 안내서를 참조하세요.

자체 VPC를 생성하거나 기존 VPC를 사용하는 경우, VPC는 다음과 같은 설정 또는 속성을 가져야 합니다.

- VPC에는 서브넷이 최소 하나 이상 있어야 합니다.

  AWS OpsWorks for Chef Automate 서버에 공개적으로 액세스할 수 있는 경우 서브넷을 퍼블릭으로 설정하고 퍼블릭 IP 자동 할당을 활성화합니다.
- [DNS 확인]이 활성화되어야 합니다.
- 서브넷에서 [퍼블릭 IP 자동 할당]을 활성화합니다.

사용자 지정 도메인 사용을 위한 사전 조건(선택 사항)

고유 도메인에 Chef Automate 서버를 설정하여 사용자 지정 도메인에서 퍼블릭 엔드포인트를 지정하여 서버의 엔드포인트로 사용할 수 있습니다. 사용자 지정 도메인을 사용하는 경우 이 단원에서 자세히 설명하는 것과 같이 다음이 모두 필요합니다.

주제

- 사용자 지정 도메인 설정
- 인증서 가져오기
- 프라이빗 키 가져오기

사용자 지정 도메인 설정

사용자 지정 도메인에서 Chef Automate 서버를 실행하려면 서버의 퍼블릭 엔드포인트(예: `https://aws.my-company.com`)가 필요합니다. 사용자 지정 도메인을 지정하는 경우 이전 단원에서 설명한 것과 같이 인증서와 프라이빗 키를 제공해야 합니다.

서버를 만든 후 서버에 액세스하려면 기본 설정 DNS 서비스에 CNAME DNS 레코드를 추가합니다. 이 레코드는 사용자 지정 도메인을 Chef Automate 서버 생성 프로세스에 의해 생성된 엔드포인트(서버 `Endpoint`의 속성 값)을 가리켜야 합니다. 서버가 사용자 지정 도메인을 사용하는 경우 생성된 `Endpoint` 값을 사용하여 서버에 액세스할 수 없습니다.

인증서 가져오기

사용자 지정 도메인에 Chef Automate 서버를 설정하려면 PEM 형식의 HTTPS 인증서가 필요합니다. 이는 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다. Chef Automate 서버 생성 워크플로를 완료할 때 이 인증서를 지정하는 경우 사용자 지정 도메인과 프라이빗 키도 제공해야 합니다.

인증서 값에 대한 요구 사항은 다음과 같습니다.

- 자체 서명된 사용자 지정 인증서 또는 전체 인증서 체인을 제공할 수 있습니다.
- 인증서는 유효한 X509 인증서 또는 PEM 형식의 인증서 체인이어야 합니다.
- 인증서는 업로드 시점에 유효해야 합니다. 유효 기간이 시작되기 전(인증서의 `NotBefore` 날짜) 또는 만료된 후(인증서의 `NotAfter` 날짜)에는 인증서를 사용할 수 없습니다.
- 인증서의 일반 이름 또는 SAN(주체 대체 이름)이 있는 경우 사용자 지정 도메인 값과 일치해야 합니다.
- 인증서는 사용자 지정 프라이빗 키 필드의 값과 일치해야 합니다.

프라이빗 키 가져오기

사용자 지정 도메인에 Chef Automate 서버를 설정하려면 HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키가 필요합니다. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다. 사용자 지정 프라이빗 키를 지정하는 경우 사용자 지정 도메인과 인증서도 제공해야 합니다.

EC2 키 페어 설정(선택 사항)

Chef 서버 관리에는 일반적으로 SSH 연결이 필요하거나 권장되지 않습니다. `knife` 명령을 사용하여 Chef 서버에서 대부분의 관리 작업을 수행할 수 있습니다.

Chef Automate 대시보드 로그인 암호를 분실했거나 변경하려는 경우, SSH를 사용하여 서버에 연결하려면 EC2 키 페어가 필요합니다. 기존 키 페어를 사용하거나 새 키 페어를 생성할 수 있습니다. 새 EC2 키 페어 생성 방법에 대한 자세한 정보는 Amazon EC2 키 페어를 참조하세요.

EC2 키 페어가 필요하지 않다면 Chef 서버를 생성할 준비가 되었습니다.

Chef Automate 서버 생성

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레

이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

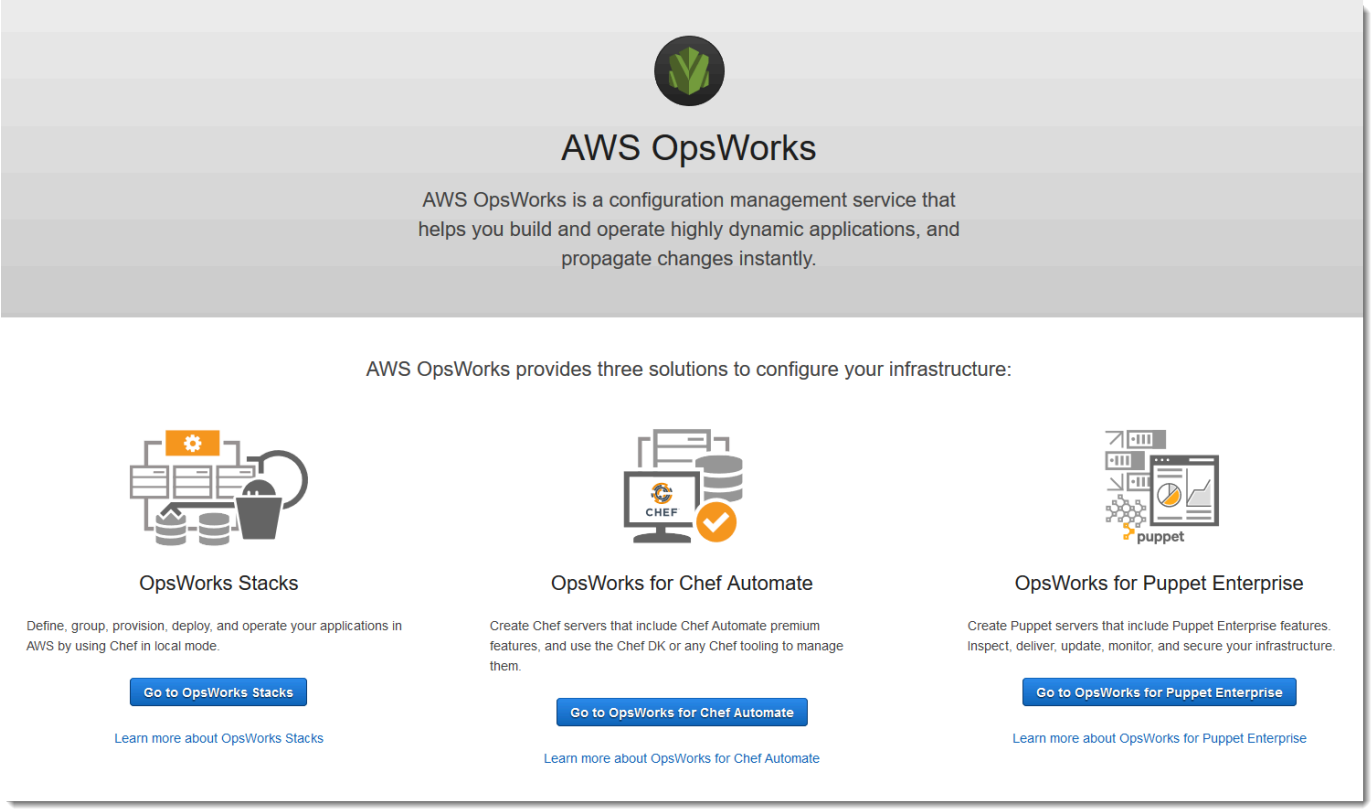
AWS OpsWorks for Chef Automate 콘솔 또는를 사용하여 Chef 서버를 생성할 수 있습니다 AWS CLI.

주제

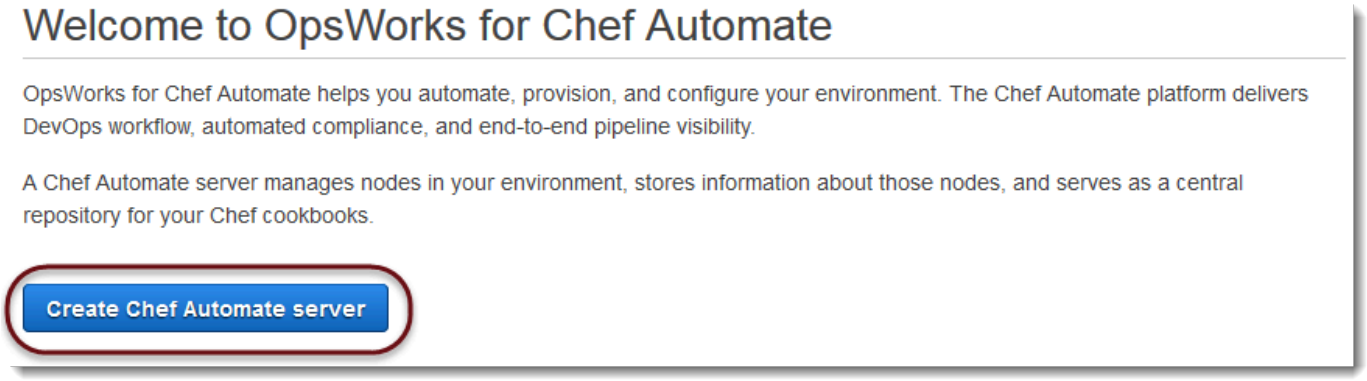
- 에서 Chef Automate 서버 생성 AWS Management Console
- 를 사용하여 Chef Automate 서버 생성 AWS CLI

에서 Chef Automate 서버 생성 AWS Management Console

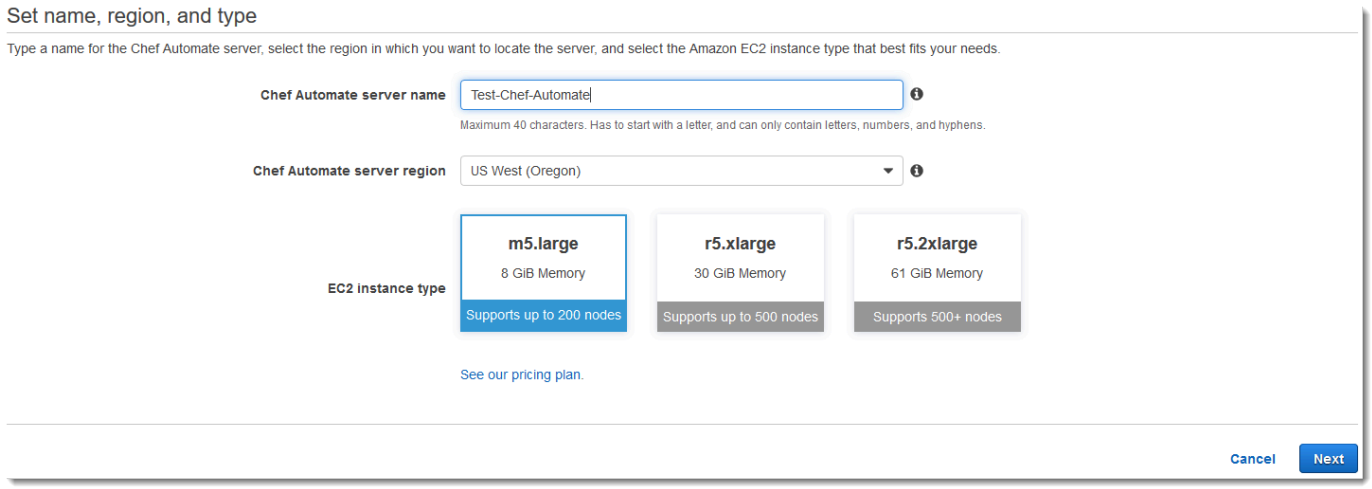
1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks 홈 페이지에서 OpsWorks for Chef Automate로 이동을 선택합니다.



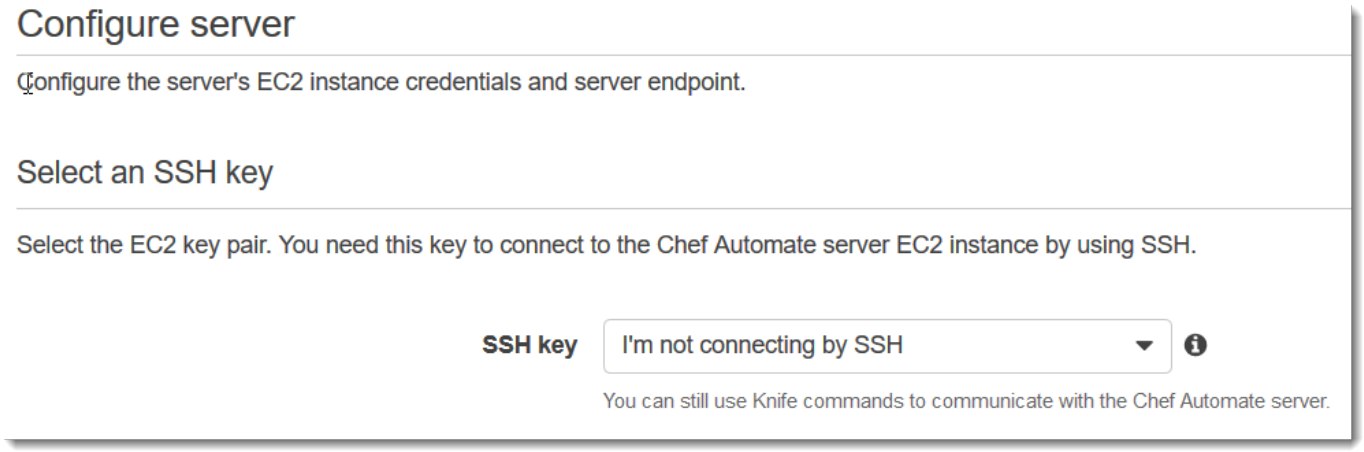
3. AWS OpsWorks for Chef Automate 홈 페이지에서 Chef Automate 서버 생성을 선택합니다.



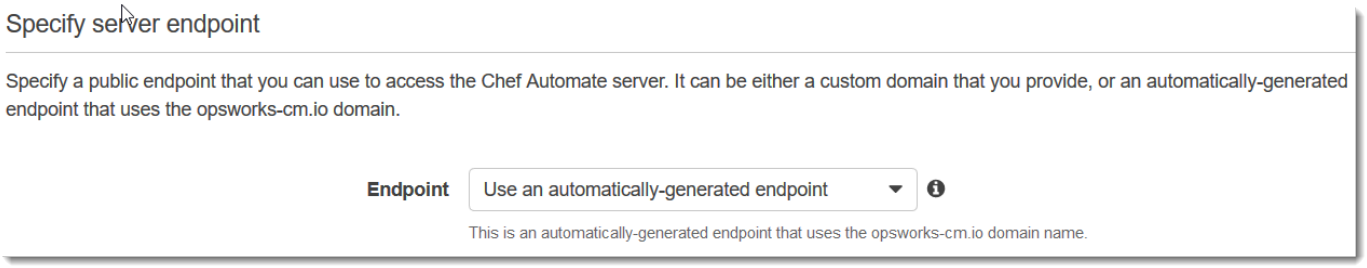
4. [이름, 리전, 유형 설정] 페이지에서 서버의 이름을 지정합니다. Chef 서버 이름은 40자 이하여야 하며 영숫자와 대시만 포함할 수 있습니다. 지원되는 리전을 선택한 다음 관리하려는 노드 수에 맞는 인스턴스 유형을 선택합니다. 필요한 경우 서버가 생성된 후 인스턴스 유형을 변경할 수 있습니다. 이 안내를 위해 미국 서부(오레곤) 리전에 m5.large 인스턴스 유형을 생성합니다. 다음을 선택합니다.



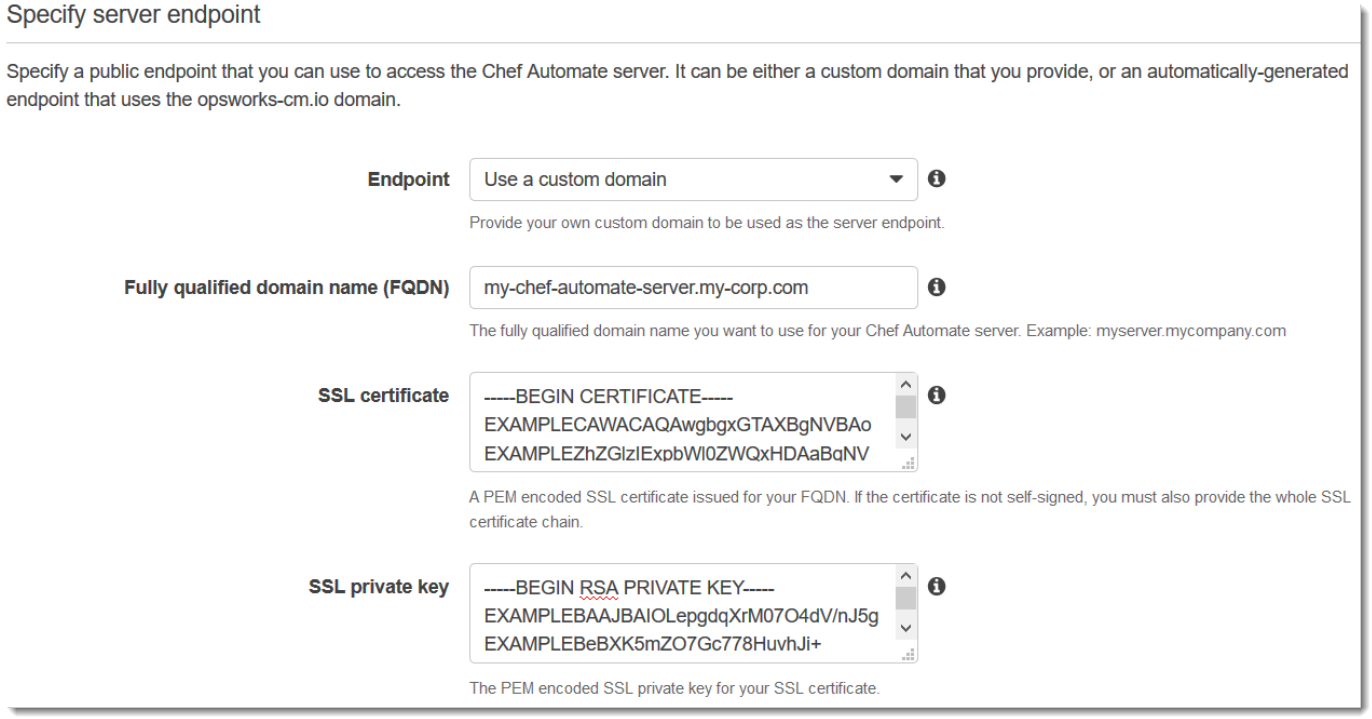
5. 키 페어 이름을 지정하지 않으려면 서버 구성 페이지에서 SSH 키 드롭다운 목록의 기본 선택을 그대로 둡니다.



6. 서버 엔드포인트 지정에서 기본값인 자동 생성된 엔드포인트 사용을 그대로 두고 서버가 사용자 지정 도메인에 있는 것을 원하지 않는 한 다음을 선택합니다. 사용자 지정 도메인을 구성하려면 다음 단계로 이동합니다.



7. 사용자 지정 도메인을 사용하려면 서버 엔드포인트 지정에 있는 드롭다운 목록에서 사용자 지정 도메인 사용을 선택합니다.



a. FQDN(정규화된 도메인 이름)에서 FQDN을 지정합니다. 사용하고자 하는 도메인 이름을 보유해야 합니다.

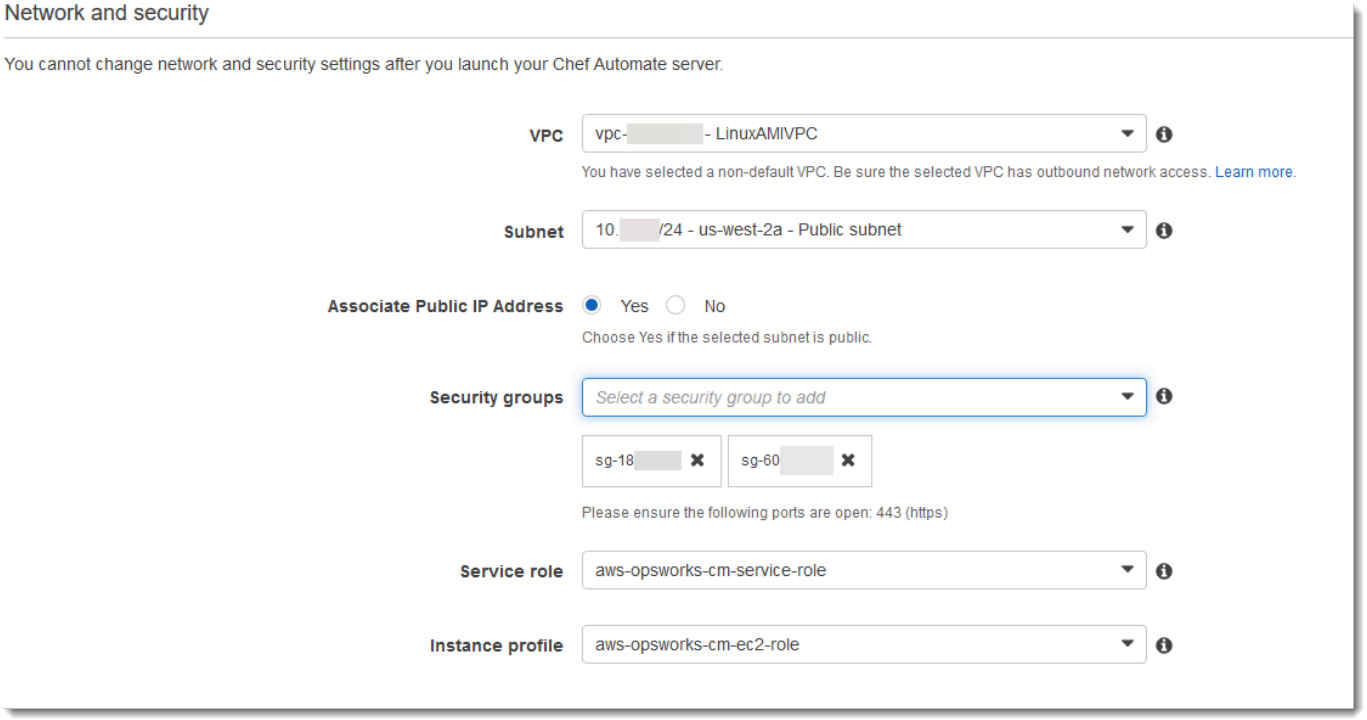
b. SSL 인증서에서 `-----BEGIN CERTIFICATE-----`로 시작하고 `-----END CERTIFICATE-----`로 끝나는 전체 PEM 형식의 인증서를 붙여넣습니다. SSL 인증서 주체는 이전 단계에서 입력한 FQDN과 일치해야 합니다.

c. SSL 프라이빗 키에서 `-----BEGIN RSA PRIVATE KEY-----`로 시작하고 `-----END RSA PRIVATE KEY-----`로 끝나는 전체 RSA 프라이빗 키를 붙여넣습니다. SSL 프라이빗 키는 이전 단계에서 입력한 SSL 인증서의 퍼블릭 키와 일치해야 합니다. 다음을 선택합니다.

8. 고급 설정 구성 페이지의 네트워크 및 보안 영역에서 VPC, 서브넷 및 하나 이상의 보안 그룹을 선택합니다. 다음은 VPC에 대한 요구 사항입니다.

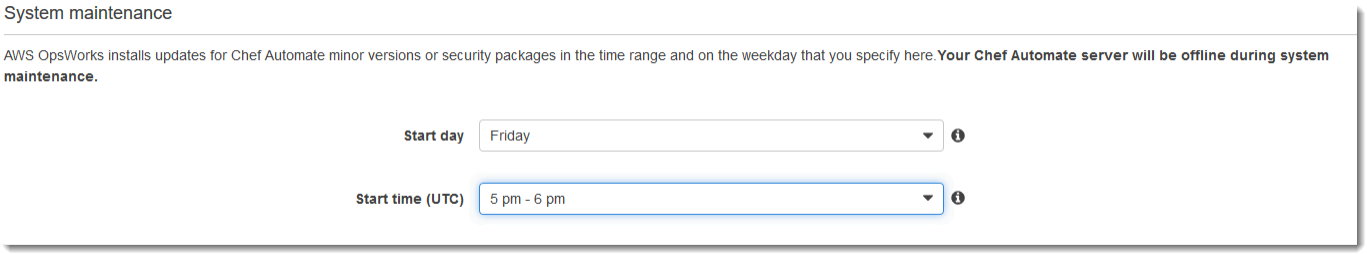
   - VPC에는 하나 이상의 퍼블릭 서브넷이 있어야 합니다.
   - DNS 확인이 활성화되어야 합니다.
   - 퍼블릭 서브넷에서 퍼블릭 IP 자동 할당이 활성화되어야 합니다.

   AWS OpsWorks 사용하려는 보안 그룹, 서비스 역할 및 인스턴스 프로파일이 아직 없는 경우에서 사용자를 대신하여 보안 그룹, 서비스 역할 및 인스턴스 프로파일을 생성할 수 있습니다. 서버는 여러 보안 그룹의 멤버일 수 있습니다. 이 페이지를 나간 뒤에는 Chef 서버의 네트워크 및 보안 설정을 변경할 수 없습니다.

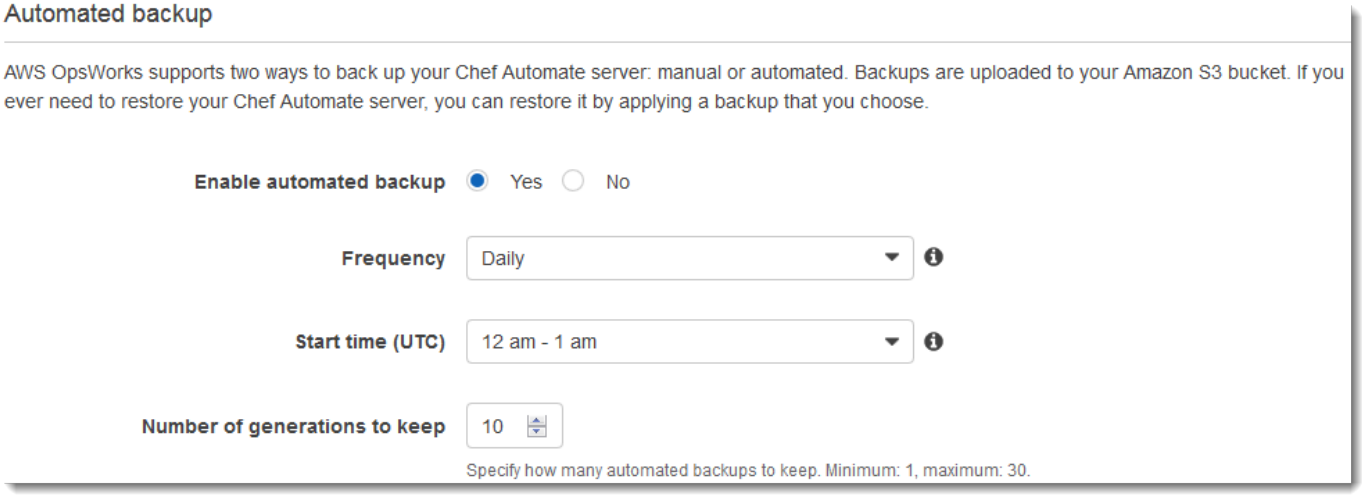


9. [시스템 유지 관리] 섹션에서 시스템 유지 관리를 시작하려는 날짜와 시간을 설정합니다. 시스템 유지 관리 중에는 서버가 오프라인 상태여야 하므로 정규 업무 시간 중 서버에 대한 수요가 낮은 시간을 선택하세요. 연결된 노드는 유지 관리가 완료될 때까지 `pending-server` 상태로 전환됩니다.

   유지 관리 기간은 필수 항목입니다. AWS Management Console AWS CLI또는 APIs.



10. 백업을 구성합니다. 기본적으로 자동 백업이 활성화되어 있습니다. 선호하는 빈도와 자동 백업을 시작할 시간을 설정한 다음 Amazon Simple Storage Service에 저장할 백업 세대 수를 설정합니다. 최대 30개의 백업이 보관됩니다. 최대값에 도달하면는 가장 오래된 백업을 AWS OpsWorks for Chef Automate 삭제하여 새 백업을 위한 공간을 확보합니다.



11. (선택 사항) 태그에서 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같은 서버 및 관련 리소스에 태그를 추가합니다. AWS OpsWorks for Chef Automate 서버 태그 지정에 대한 자세한 내용은 섹션을 참조하세요AWS OpsWorks for Chef Automate 리소스에서 태그 작업.
12. 고급 설정 구성을 마치면 [다음]을 선택합니다.
13. [검토] 페이지에서 선택 사항을 검토합니다. 서버를 생성할 준비가 되면 [시작]을 선택합니다.

    AWS OpsWorks 가 Chef 서버를 생성할 때까지 기다리는 동안 로 이동하여 Starter Kit Starter Kit를 사용하여 Chef 서버 구성 및 Chef Automate 대시보드 자격 증명을 다운로드합니다. 서버가 온라인 상태가 되어 이러한 항목을 다운로드할 때까지 기다리지 마십시오.

    서버 생성이 끝나면 AWS OpsWorks for Chef Automate 홈 페이지에서 Chef 서버를 확인할 수 있으며, 상태는 온라인입니다. 서버가 온라인 상태가 되면 `https://your_server_name-random.region.opsworks-cm.io` 형식의 URL을 가진 서버 도메인에서 Chef Automate 대시보드를 사용할 수 있습니다.

를 사용하여 Chef Automate 서버 생성 AWS CLI

AWS CLI 명령을 실행하여 AWS OpsWorks for Chef Automate 서버를 생성하는 것은 콘솔에서 서버를 생성하는 것과 다릅니다. 사용하려는 기존 역할을 지정하지 않으면 콘솔에서가 서비스 역할 및 보안 그룹을 AWS OpsWorks 생성합니다. 에서 보안 그룹을 지정하지 않고 서비스 역할을 자동으로 AWS CLI AWS OpsWorks 생성하지 않는 경우에서 보안 그룹을 생성할 수 있습니다. `create-server` 명령의 일부로 서비스 역할 ARN을 제공해야 합니다. 콘솔에서 AWS OpsWorks 가 Chef Automate 서버를 생성하는 동안 Chef Automate 시작 키트와 Chef Automate 대시보드의 로그인 자격 증명을 다운로드합니다. 를 AWS OpsWorks for Chef Automate 사용하여 서버를 생성할 때는이 작업을 수행할 수 없으므로 JSON 처리 유틸리티를 AWS CLI사용하여 새 AWS OpsWorks for Chef Automate 서버가 온라인

상태가 된 후 `create-server` 명령 결과에서 로그인 자격 증명과 스타터 키트를 가져옵니다. 또는 새 AWS OpsWorks for Chef Automate 서버가 온라인 상태가 된 후 콘솔에서 새 스타터 키트 및 새 로그인 자격 증명 집합을 생성할 수 있습니다.

로컬 컴퓨터에서 아직를 실행하지 않는 경우 AWS Command Line Interface 사용 설명서의 설치 지침에 AWS CLI 따라를 AWS CLI다운로드하고 설치합니다. 이 단원에서는 `create-server` 명령과 함께 사용할 수 있는 모든 파라미터를 다 설명하지는 않습니다. `create-server` 파라미터에 대한 자세한 내용은 AWS CLI 레퍼런스의 `create-server`를 참조하세요.

1. 사전 요구 사항(특히 VPC 설정)을 완료하거나 사용할 기존 VPC가 있어야 합니다. Chef Automate 서버를 생성하려면 서브넷 ID가 필요합니다.
2. 또는 OpenSSL을 사용하여 Chef 중심 키를 생성하고 나서 키를 로컬 컴퓨터의 안전하고 편리한 파일에 저장한다. 이 중심 키는 `create-server` 명령에서 제공하지 않는 경우 서버 생성 프로세스의 일부로서 자동으로 생성됩니다. 이 단계를 건너뛰려는 경우 `create-server` 명령의 결과에서 Chef Automate 중심 키를 대신 가져올 수 있습니다. Chef Automate 중심 키 값은 RSA 키 페어의 퍼블릭 절반이므로, 다음 명령을 사용하여 중심 키를 생성하도록 선택하는 경우 `-pubout` 파라미터를 포함해야 합니다. 자세한 정보는 6단계를 참조하세요.

   ```
   umask 077
   openssl genrsa -out "pivotal" 2048
   openssl rsa -in "pivotal" -pubout
   ```

3. 서비스 역할과 인스턴스 프로파일을 생성합니다.
4. 계정에서 서비스 역할의 ARNs 찾아 복사합니다.

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   `list-roles` 명령의 결과에서 다음과 같은 서비스 역할 ARN을 찾아봅니다. 서비스 역할 ARN을 기록해 둡니다. 이러한 값은 Chef Automate 서버를 만드는 데 필요합니다.

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
   ```

```
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

5. 해당 계정에서 인스턴스 프로파일의 ARN을 찾아서 복사합니다.

```
aws iam list-instance-profiles --no-paginate
```

`list-instance-profiles` 명령의 결과에서 다음과 같은 인스턴스 프로파일 ARN을 찾아봅니다. 인스턴스 프로파일 ARN을 기록해 둡니다. 이러한 값은 Chef Automate 서버를 만드는 데 필요합니다.

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
```

```
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

6. `create-server` 명령을 실행하여 AWS OpsWorks for Chef Automate 서버를 생성합니다.

   - `--engine` 값은 `ChefAutomate`, `--engine-model` 값은 `Single`, 그리고 `--engine-version` 값은 `12`입니다.
   - 서버 이름은 AWS 계정 내에서 각 리전 내에서 고유해야 합니다. 서버 이름은 문자로 시작해야 하며, 그 이후에는 문자, 숫자 또는 하이픈(-)을 사용할 수 있으며, 최대 길이는 40자입니다.
   - 4단계와 5단계에서 복사해 둔 인스턴스 프로파일 ARN 및 서비스 역할 ARN을 사용합니다.
   - 유효한 인스턴스 유형은 `m5.large`, `r5.xlarge` 또는 `r5.2xlarge`입니다. 인스턴스 유형의 사양에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하세요.
   - `--engine-attributes` 파라미터는 선택 사항이며, 하나 또는 두 값을 모두 지정하지 않을 경우 서버 생성 프로세스에서 해당 값이 자동으로 생성됩니다. `--engine-attributes`를 추가할 경우 2단계에서 생성한 `CHEF_AUTOMATE_PIVOTAL_KEY` 값, `CHEF_AUTOMATE_ADMIN_PASSWORD` 또는 두 가지 모두를 지정합니다.

CHEF_AUTOMATE_ADMIN_PASSWORD의 값을 설정하지 않으면 암호가 생성되어 create-server 응답의 일부로서 반환됩니다. 또한 콘솔에서 스타터 키트를 다시 다운로드할 수도 있습니다. 그러면 이 암호가 다시 생성됩니다. 암호 길이는 최소 8자이며, 최대 32자입니다. 암호는 문자, 숫자 및 특수 문자(!/@#$%^+=_)를 포함할 수 있습니다. 암호에는 소문자, 대문자, 숫자 및 특수 문자가 각각 1개 이상 포함되어야 합니다.

- SSH 키 페어는 선택 사항이지만, Chef Automate 대시보드 관리자 암호를 재설정해야 하는 경우 Chef Automate 서버에 연결하는 데 도움이 될 수 있습니다. SSH 키 페어 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.
- 사용자 지정 도메인을 사용하려면 명령에 다음 파라미터를 추가합니다. 그렇지 않은 경우 Chef Automate의 서버 생성 프로세스가 자동으로 엔드포인트를 생성합니다. 사용자 지정 도메인을 구성하려면 세 가지 파라미터 모두가 필요합니다. 이러한 파라미터 사용에 대한 추가 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer를 참조하세요.
  - --custom-domain - 서버의 선택적 퍼블릭 엔드포인트(예: https://aws.my-company.com).
  - --custom-certificate - PEM 형식의 HTTPS 인증서. 값은 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다.
  - --custom-private-key - HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다.
- 매주 시스템 유지 관리가 필요합니다. 유효한 값은 DDD:HH:MM 형식으로 지정해야 합니다. 지정한 시간은 협정 세계시(UTC)로 표시됩니다. --preferred-maintenance-window의 값을 지정하지 않으면 기본값은 화요일, 수요일 또는 금요일의 임의 한 시간입니다.
- --preferred-backup-window의 유효 값은 HH:MM(매일 백업) 또는 DDD:HH:MM(매주 백업) 형식 중 하나로 지정해야 합니다. 지정한 시간은 UTC 형식입니다. 기본값은 임의 일일 시작 시간입니다. 자동 백업을 옵트아웃하려면 대신에 --disable-automated-backup 파라미터를 추가합니다.
- --security-group-ids에는 공백으로 구분하여 하나 이상의 보안 그룹 ID를 입력합니다.
- --subnet-ids에는 서브넷 ID를 입력합니다.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
```

```
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID
```

다음은 예입니다.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-
model "Single" --engine-version "12" --server-name "automate-06" --
instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.large" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 --key-pair "amazon-test" --preferred-maintenance-window "Mon:08:00" --preferred-
backup-window "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-
role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-service-role"
 --subnet-ids subnet-300aaa00
```

다음 예제에서는 사용자 지정 도메인을 사용하는 Chef Automate 서버를 만듭니다.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00
```

다음 예제에서는 `Stage: Production` 및 `Department: Marketing` 태그를 추가하는 Chef Automate 서버를 만듭니다. AWS OpsWorks for Chef Automate 서버에서 태그를 추가하고 관리

하는 방법에 대한 자세한 내용은이 설명서AWS OpsWorks for Chef Automate 리소스에서 태그 작업의 섹션을 참조하세요.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-test-chef-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

7. AWS OpsWorks for Chef Automate 새 서버를 생성하는 데 약 15분이 걸립니다. `create-server` 명령의 출력을 무시하거나 셸 세션을 닫지 마십시오. 다시 표시되지 않는 중요 정보가 출력에 포함되어 있을 수도 있기 때문입니다. `create-server` 결과에서 암호 및 스타터 키트를 가져오려면 다음 단계로 이동합니다.

   서버에서 사용자 지정 도메인을 사용하는 경우 `create-server` 명령 출력에서 `Endpoint` 속성 값을 복사합니다. 다음은 예입니다.

   ```
   "Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

8. 에서 키와 암호를 AWS OpsWorks for Chef Automate 생성하도록 선택한 경우 jq와 같은 JSON 프로세서를 사용하여 `create-server` 결과에서 사용 가능한 형식으로 추출할 수 있습니다. jq를 설치한 후에는 다음 명령을 실행하여 중심 키, Chef Automate 대시보드 관리자 암호 및 스타터 키트를 추출할 수 있습니다. 4단계에서 고유의 중심 키 및 암호를 제공하지 않은 경우 추출한 중심 키 및 관리자 암호를 안전하면서 편리한 위치에 저장해야 합니다.

   ```
   #Get the Chef password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'
   ```

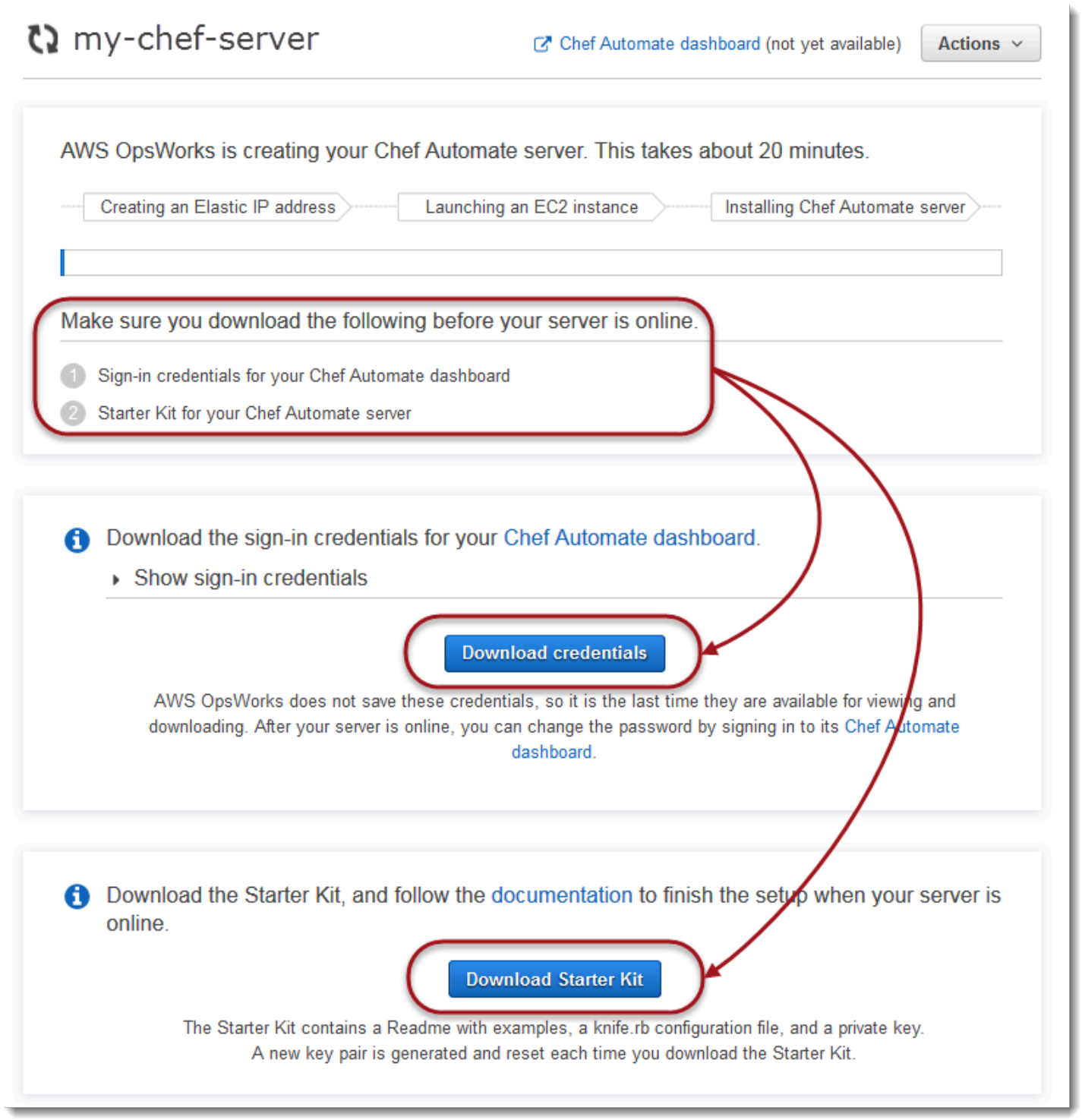
```
#Get the Chef Pivotal Key:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

#Get the Chef Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

9. 선택적으로 `create-server` 명령 결과에서 스타터 키트를 추출하지 않은 경우 AWS OpsWorks for Chef Automate 콘솔의 서버 속성 페이지에서 새 스타터 키트를 다운로드할 수 있습니다. 새 스타터 키트를 다운로드하면 Chef Automate 대시보드 관리자 암호를 재설정합니다.
10. 사용자 지정 도메인을 사용하지 않는 경우 다음 단계로 이동합니다. 서버에서 사용자 지정 도메인을 사용하는 경우 엔터프라이즈의 DNS 관리 도구에 CNAME 항목을 생성하여 사용자 지정 도메인이 7단계에서 복사한 AWS OpsWorks for Chef Automate 엔드포인트를 가리키도록 합니다. 이 단계를 완료해야 사용자 지정 도메인을 사용하여 서버에 연결하거나 로그인할 수 있습니다.
11. 서버 생성 프로세스가 완료되면 the section called “구성 완료 및 쿡북 업로드” 단원을 진행합니다.

Starter Kit를 사용하여 Chef 서버 구성

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

Chef 서버 생성이 아직 진행 중인 동안 AWS OpsWorks for Chef Automate 콘솔에서 Properties 페이지를 엽니다. 새 Chef 서버로 처음 작업할 때는 필요한 2가지 항목을 다운로드하라는 메시지가 Properties 페이지에 표시됩니다. Chef 서버가 온라인 상태가 되기 전에 이 항목을 다운로드하세요. 새 서버가 온라인 상태가 된 후에는 다운로드 버튼을 사용할 수 없습니다.



- Chef 서버의 자격 증명으로 로그인. 이러한 자격 증명을 사용하여 Chef Automate 대시보드에 로그인합니다. 여기서 워크플로 및 규정 준수 스캔과 같은 Chef Automate 프리미엄 기능을 사용합니다. 는 이러한 자격 증명을 저장하지 AWS OpsWorks 않습니다.이 자격 증명을 보고 다운로드할 수 있는

마지막 시간입니다. 필요할 경우, 로그인 후 이 자격 증명과 함께 제공되는 암호를 변경할 수 있습니다.

- 스타터 키트. 스타터 키트에는 예제가 있는 README 파일, `knife.rb` 구성 파일, 주 사용자 또는 중심 사용자용 프라이빗 키가 포함되어 있습니다. Starter Kit를 다운로드할 때마다 새 키 페어가 생성되고 예전 키가 리셋됩니다.

Starter Kit .zip 파일에는 새 서버에서만 작동하는 자격 증명 외에도 AWS OpsWorks for Chef Automate 모든 서버에서 작동하는 Chef 리포지토리의 간단한 예가 포함되어 있습니다. Chef 리포지토리에 쿡북, 역할, 구성 파일 및 Chef로 노드를 관리하기 위한 그 밖의 아티팩트를 저장할 수 있습니다. 이 리포지토리는 Git 같은 버전 제어 시스템에 저장하고 소스 코드로 취급하는 것이 좋습니다. Git에서 추적되는 Chef 리포지토리 설정 방법에 대한 내용과 예제는 Chef 설명서의 chef-repo에 대하여를 참조하세요.

사전 조건

1. 서버 생성이 계속 진행 중이면 Chef 서버에 대한 로그인 자격 증명을 다운로드해 안전하고 편리한 위치에 저장합니다.
2. 스타터 키트를 다운로드하고 작업 영역 디렉터리에 스타터 키트 .zip 파일의 압축을 풉니다. 스타터 키트의 프라이빗 키는 공유하지 마십시오. 다른 사용자가 Chef 서버를 관리하는 경우, 나중에 Chef Automate 대시보드에서 이러한 사용자를 관리자로 추가합니다.
3. Chef 서버와 노드를 관리하는 데 사용할 컴퓨터에 Chef Workstation(이전에는 Chef Development Kit 또는 Chef DK로 알려짐)을 다운로드하여 설치합니다. 이 `knife` 유틸리티는 Chef Workstation의 일부입니다. Chef 웹 사이트에서 Chef Workstation 설치 지침 단원을 참조하세요.

스타터 키트 콘텐츠 탐색

스타터 키트에는 다음과 같은 콘텐츠가 있습니다.

- `cookbooks/` - 생성하는 쿡북의 디렉터리. `cookbooks/`폴더에는 Chef Supermarket 웹 사이트의 `nginx` 쿡북에 따라 달라지는 래퍼 쿡북인 `opsworks-webserver` 쿡북이 들어 있습니다. `cookbooks/` 디렉토리에서 쿡북 종속성을 사용할 수 없는 경우 `Policyfile.rb`은 기본적으로 Chef 슈퍼마켓을 보조 소스로 사용합니다.
- `Policyfile.rb` - 노드에 대한 정책이 되는 쿡북, 종속성 및 속성을 정의하는 Ruby 기반 정책 파일.
- `userdata.sh` 및 `userdata.ps1` - Chef Automate 서버를 시작한 후 사용자 데이터 파일을 사용하여 노드를 자동으로 연결할 수 있습니다. `userdata.sh`는 Linux기반 노드 부트스트래핑용이고 `userdata.ps1`은 Windows 기반 노드용입니다.

- `Berksfile` - Berkshelf 및 `berks` 명령을 사용하여 쿡북과 쿡북의 종속성을 업로드하려는 경우 이 파일을 사용할 수 있습니다. 이 연습에서는 `Policyfile.rb` 및 Chef 명령을 사용하여 쿡북, 종속성 및 속성을 업로드합니다.
- `README.md`, 스타터 키트를 사용하여 Chef Automate 서버를 처음으로 설정하는 방법을 설명하는 Markdown 기반 파일입니다.
- `.chef`는 knife 구성 파일(`knife.rb`)과 보안 인증 키 파일(.pem)이 포함된 숨은 디렉터리입니다.
  - `.chef/knife.rb` - knife 구성 파일(`knife.rb`). `knife.rb` 파일은 Chef의 `knife` 도구 작업이 AWS OpsWorks for Chef Automate 서버에 대해 실행되도록 구성됩니다.
  - `.chef/ca_certs/opsworks-cm-ca-2020-root.pem` - AWS OpsWorks가 제공하는, 인증 기관(CA)이 서명한 SSL 프라이빗 키. 이 키를 사용하여 서버는 서버가 관리하는 노드의 Chef Infra 클라이언트 에이전트에게 인증할 수 있습니다.

Chef 리포지토리 설정

Chef 리포지토리에는 몇 개의 디렉터리가 포함되어 있습니다. Starter Kit의 모든 디렉터리에는 디렉터리의 용도와 Chef를 통한 시스템 관리에 디렉터리를 사용하는 방법을 설명하는 README 파일이 포함되어 있습니다. 다음 두 가지 방법으로 Chef 서버에 쿡북을 설치할 수 있습니다. `knife` 명령을 실행하거나, Chef 명령을 실행하여 지정된 쿡북을 다운로드하고 설치하는 정책 파일(`Policyfile.rb`)을 서버에 업로드하는 것입니다. 이 안내서에서는 Chef 명령과 `Policyfile.rb`를 사용하여 쿡북을 서버에 설치합니다.

1. 로컬 컴퓨터에 쿡북을 저장할 디렉터리를 만듭니다(예: `chef-repo`). 이 리포지토리에 쿡북, 역할 및 기타 파일을 추가한 후에는 버전이 지정된 안전한 시스템(예: CodeCommit, Git 또는 Amazon S3)에 이러한 항목을 업로드 또는 저장하는 것이 좋습니다.
2. `chef-repo` 디렉터리에서 다음 디렉터리를 생성합니다.
   - `cookbooks/` - 쿡북을 저장합니다.
   - `roles/` - `.rb` 또는 `.json` 형식으로 역할을 저장합니다.
   - `environments/` - `.rb` 또는 `.json` 형식으로 환경을 저장합니다.

Policyfile.rb를 사용하여 원격 소스에서 쿡북 가져오기

이 단원에서는 `Policyfile.rb`를 편집하여 쿡북을 지정한 다음, Chef 명령을 실행하여 파일을 서버에 업로드하고 쿡북을 설치합니다.

1. 스타터 키트에서 `Policyfile.rb`를 봅니다. 기본적으로, `Policyfile.rb`에는 `opsworks-webserver` 래퍼 쿡북이 포함되며, 이 쿡북은 Chef Supermarket 웹 사이트에서 사용할 수 있는 nginx 쿡북에 의존합니다. `nginx` 쿡북은 관리형 노드에서 웹 서버를 설치하고 구성합니다. 관리형 노드에 Chef Infra 클라이언트 에이전트를 설치하는 필수 `chef-client` 쿡북도 지정됩니다.

   또한 `Policyfile.rb`는 노드에서 규정 준수 검사를 설정하는 데 사용할 수 있는 선택적 Chef Audit 쿡북을 가리킵니다. 규정 준수 검사를 설정하고 관리형 노드의 규정 준수 결과를 가져오는 방법에 대한 자세한 내용은 의 규정 준수 스캔 AWS OpsWorks for Chef Automate 단원을 참조하세요. 규정 준수 검사 및 감사를 지금 바로 구성하지 않으려는 경우 `'audit'`를 `run_list` 섹션에서 삭제하고, 파일 끝에서 `audit` 쿡북 속성을 지정하지 마십시오.

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information about the Policyfile feature, visit

# https://docs.chef.io/policyfile.html


# A name that describes what the system you're building with Chef does.

name 'opsworks-demo-webserver'

# The cookbooks directory is the preferred source for external cookbooks

default_source :chef_repo, "cookbooks/" do |s|

  s.preferred_for "nginx", "windows", "chef-client", "yum-epel", "seven_zip",
                  "build-essential", "mingw", "ohai", "audit", "logrotate", "cron"

end
# Alternative source
default_source :supermarket


# run_list: chef-client runs these recipes in the order specified.

run_list  'chef-client',
          'opsworks-webserver',
          'audit'
# add 'ssh-hardening' to your runlist to fix compliance issues detected by the ssh-
baseline profile
```

```
# Specify a custom source for a single cookbook:

cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'


# Policyfile defined attributes

# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

다음은 nginx 웹 서버만 지금 구성하려는 경우 audit 쿡북과 속성이 없는 Policyfile.rb의 예제입니다.

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with Chef does.
name 'opsworks-demo-webserver'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list  'chef-client',
          'opsworks-webserver'

# Specify a custom source for a single cookbook:
cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'
```

Policyfile.rb를 변경하는 경우 파일을 저장해야 합니다.

2. Policyfile.rb에 정의된 쿡북을 다운로드하고 설치합니다.

```
chef install
```

모든 쿡북은 쿡북의 `metadata.rb` 파일에 버전이 지정되어 있습니다. 쿡북을 변경할 때마다 쿡북의 `metadata.rb`에 있는 쿡북 버전을 올려야 합니다.

3. 규정 준수 검사를 구성하도록 선택하고 `audit` 쿡북 정보를 정책 파일에 보관한 경우 정책 `opsworks-demo`를 서버에 푸시합니다.

```
chef push opsworks-demo
```

4. 3단계를 완료한 경우 정책 설치를 확인합니다. 다음 명령을 실행합니다.

```
chef show-policy
```

결과는 다음과 비슷해야 합니다.

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 이제 노드를 Chef Automate 서버에 추가하고 부트스트래핑할 준비가 되었습니다. 에서 노드 자동 추가 AWS OpsWorks for Chef Automate의 단계를 수행하여 노드 연결을 자동화하거나, 노드를 개별적으로 추가의 단계를 수행하여 한 번에 한 개의 노드를 추가할 수 있습니다.

(대체 방법) Berkshelf를 사용하여 원격 소스에서 쿡북 가져오기

Berkshelf는 쿡북과 쿡북의 종속성을 관리하기 위한 도구입니다. `Policyfile.rb` 대신 Berkshelf를 사용하여 쿡북을 로컬 스토리지에 설치하려는 경우 이전 단원 대신 이 단원의 절차를 사용하세요. Chef 서버에 사용할 쿡북과 버전을 지정하고 업로드할 수 있습니다. 스타터 키트에는 쿡북을 나열하는 데 사용할 수 있는 `Berksfile`이라는 파일이 포함되어 있습니다.

1. 시작하려면 `chef-client` 쿡북을 포함된 Berksfile에 추가합니다. `chef-client` 쿡북은 Chef Automate 서버에 연결하는 각 노드에서 Chef Infra 클라이언트 에이전트 소프트웨어를 구성합니다. 이 쿡북에 대해 자세히 알아보려면 Chef Supermarket에서 Chef Client Cookbook 단원을 참조하세요.
2. 텍스트 편집기를 사용하여 웹 서버 애플리케이션을 설치하는 다른 쿡북을 Berksfile에 추가합니다(예: Apache 웹 서버를 설치하는 `apache2` 쿡북). Berksfile은 다음과 유사합니다.

```
source 'https://supermarket.chef.io'
cookbook 'chef-client'
```

```
cookbook 'apache2'
```

3. 로컬 컴퓨터에 쿡북을 다운로드하고 설치합니다.

```
berks install
```

4. Chef 서버에 쿡북을 업로드합니다.

   Linux에서 다음 명령을 실행합니다.

```
SSL_CERT_FILE='.chef/ca_certs/opsworks-cm-ca-2020-root.pem' berks upload
```

   Windows의 PowerShell 세션에서 다음 Chef Workstation 명령을 실행합니다. 이 명령을 실행하기 전에 PowerShell의 실행 정책을 `RemoteSigned`로 설정해야 합니다. `chef shell-init`를 추가하여 Chef Workstation 유틸리티 명령을 PowerShell에 사용할 수 있도록 합니다.

```
$env:SSL_CERT_FILE="ca_certs\opsworks-cm-ca-2020-root.pem"
chef shell-init berks upload
Remove-Item Env:\SSL_CERT_FILE
```

5. Chef Automate 서버에서 현재 사용할 수 있는 쿡북 목록을 표시하여 쿡북이 설치되었는지 확인합니다. 다음 `knife` 명령을 실행하여 이 작업을 수행할 수 있습니다.

   AWS OpsWorks for Chef Automate 서버로 관리할 노드를 추가할 준비가 되었습니다.

```
knife cookbook list
```

(선택 사항) 사용자 지정 도메인을 사용하도록 **knife** 구성

Chef Automate 서버에서 사용자 지정 도메인을 사용하는 경우 서버의 인증서 체인을 서명한 루트 CA의 PEM 인증서 또는 인증서가 자체 서명된 경우 서버 PEM 인증서를 추가해야 할 수 있습니다. `ca_certs`는 Chef `knife` 유틸리티에서 신뢰할 수 있는 CA(인증 기관)가 포함된 `chef/`의 하위 디렉토리입니다.

사용자 지정 도메인을 사용하지 않거나 운영 체제에서 신뢰할 수 있는 루트 CA에서 사용자 지정 인증서가 서명된 경우 이 섹션을 건너뛸 수 있습니다. 그렇지 않은 경우 다음 단계에서 설명한 것과 같이 Chef Automate 서버 SSL 인증서를 신뢰하도록 `knife`를 구성합니다.

1. 다음 명령을 실행합니다.

```
knife ssl check
```

결과가 다음과 유사한 경우 이 절차의 나머지 부분을 건너뛰고 다음으로 이동합니다. Chef 서버가 관리할 노드 추가.

```
Connecting to host my-chef-automate-server.my-corp.com:443
          Successfully verified certificates from 'my-chef-automate-server.my-
corp.com'
```

다음과 유사한 오류 메시지가 나타나면 다음 단계로 이동하세요.

```
Connecting to host my-chef-automate-server.my-corp.com:443
          ERROR: The SSL certificate of my-chef-automate-server.my-corp.com could
 not be verified.
          ...
```

2. `knife ssl fetch`를 실행하여 AWS OpsWorks for Chef Automate 서버의 인증서를 신뢰합니다. 또는 서버의 루트 CA PEM 형식 인증서를 `knife ssl check`의 출력에서 `trusted_certs_dir` 값인 디렉토리로 수동 복사할 수 있습니다. 기본적으로 이 디렉토리는 스타터 키트의 `.chef/ca_certs/`에 있습니다. 출력은 다음과 같을 것입니다.

```
WARNING: Certificates from my-chef-automate-server.my-corp.com will be fetched and
 placed in your trusted_cert
          directory (/Users/username/starterkit/.chef/../.chef/ca_certs).

          Knife has no means to verify these are the correct certificates. You
 should
          verify the authenticity of these certificates after downloading.

          Adding certificate for my-chef-automate-server in /Users/users/
starterkit/.chef/../.chef/ca_certs/servv-aqtswxu20swzkjgz.crt
          Adding certificate for MyCorp_Root_CA in /Users/users/
starterkit/.chef/../.chef/ca_certs/MyCorp_Root_CA.crt
```

3. `knife ssl check`를 다시 실행합니다. 출력은 다음과 같을 것입니다.

```
Connecting to host my-chef-automate-server.my-corp.com:443
          Successfully verified certificates from 'my-chef-automate-server.my-
corp.com'
```

Chef Automate 서버에 knife를 사용할 준비가 되었습니다.

Chef 서버가 관리할 노드 추가

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

chef-client 에이전트는 서버에 연결된, 노드라고 하는 물리적 또는 가상 컴퓨터에서 Chef 레시피를 실행합니다. 지원되는 운영 체제를 노드가 실행 중이라면 온프레미스 컴퓨터나 인스턴스를 Chef 서버에 연결할 수 있습니다. 노드를 Chef 서버에 등록하면 이러한 노드에 chef-client 에이전트 소프트웨어가 설치됩니다.

노드를 추가하는 다음 방법을 사용할 수 있습니다.

- Chef 서버가 관리할 수 있도록 EC2 인스턴스를 추가하거나 부트스트랩하는 knife 명령을 실행하여 메모를 개별적으로 추가합니다. 자세한 내용은 노드를 개별적으로 추가을 참조하세요.
- 스크립트를 사용하여 노드를 Chef 서버와 무인 연결하여 노드를 자동으로 추가합니다. 스타터 키트의 코드는 무인 방식을 사용하여 자동으로 노드를 추가하는 방법을 보여줍니다. 자세한 내용은 에서 노드 자동 추가 AWS OpsWorks for Chef Automate 섹션을 참조하세요.

주제

- 노드를 개별적으로 추가
- 에서 노드 자동 추가 AWS OpsWorks for Chef Automate

노드를 개별적으로 추가

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레

이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 섹션에서는 Chef 서버가 관리할 수 있도록 EC2 인스턴스를 추가하거나 부트스트랩하는 `knife` 명령을 실행하는 방법을 설명합니다.

AWS OpsWorks for Chef Automate 서버와 관련된 노드에서 지원되는 최소 `chef-client` 버전은 13.x입니다. 안정적인 최신 `chef-client` 버전을 실행하는 것이 좋습니다.

주제

- (선택 사항) Chef Automate 서버 루트 CA의 URL 지정
- 지원되는 운영 체제
- knife를 사용하여 노드 추가

(선택 사항) Chef Automate 서버 루트 CA의 URL 지정

서버에서 사용자 지정 도메인과 인증서를 사용하는 경우 서버의 루트 CA PEM 형식 인증서를 가져오는 데 사용할 수 있는 퍼블릭 URL을 사용하여 userdata 스크립트에서 `ROOT_CA_URL` 변수를 편집해야 할 수 있습니다. 다음 AWS CLI 명령은 루트 CA를 Amazon S3 버킷에 업로드하고 1시간 동안 사용할 수 있는 미리 서명된 URL을 생성합니다.

1. 루트 CA PEM 형식 인증서를 S3에 업로드합니다.

```
aws s3 cp ROOT_CA_PEM_FILE_PATH s3://bucket_name/
```

2. 한 시간(이 예에서는 3600초) 동안 루트 CA를 다운로드하는 데 사용할 수 있는 미리 서명된 URL을 생성합니다.

```
aws s3 presign s3://bucket_name/ROOT_CA_PEM_FILE_NAME --expires-in 3600
```

3. userdata 스크립트에서 미리 서명된 URL의 값을 사용하여 `ROOT_CA_URL` 변수를 편집합니다.

지원되는 운영 체제

현재 노드에 지원되는 운영 체제 목록은 Chef 웹 사이트를 참조하세요.

knife를 사용하여 노드 추가

knife-ec2 플러그인은 Chef Workstation에 포함되어 있습니다. knife-ec2에 더 익숙하다면 knife bootstrap 대신 사용하여 새 EC2 인스턴스를 프로비저닝하고 부트스트랩할 수 있습니다. 그렇지 않은 경우, 새 EC2 인스턴스를 시작한 다음 이 섹션의 단계에 따르십시오.

관리할 노드를 추가하려면

1. 다음 knife bootstrap 명령을 실행합니다. 이 명령은 Chef 서버에서 관리할 노드에 EC2 인스턴스를 부트스트랩합니다. nginx 단원에서 설치한 the section called “Policyfile.rb를 사용하여 원격 소스에서 쿡북 가져오기” 쿡북의 레시피를 실행하도록 Chef 서버에 지시합니다. knife bootstrap 명령을 실행하여 노드를 추가하는 자세한 방법은 Chef 설명서의 노드 부트스트랩 단원을 참조하세요.

   다음 표에는 이 단계의 knife 명령에서 노드 운영 체제에 사용할 수 있는 유효한 사용자 이름이 나와 있습니다. root 및 ec2-user를 둘 다 사용할 수 없으면 AMI 공급자에게 문의하세요. Linux 기반 인스턴스에 연결하는 자세한 방법은 AWS 설명서의 SSH를 사용하여 Linux 인스턴스에 연결 단원을 참조하세요.

   노드 운영 체제의 사용자 이름에 유효한 값

운영 체제	유효한 사용자 이름
Amazon Linux	ec2-user
Red Hat Enterprise Linux 5	root 또는 ec2-user
Ubuntu	ubuntu
Fedora	fedora 또는 ec2-user
SUSE Linux	root 또는 ec2-user

```
knife bootstrap INSTANCE_IP_ADDRESS -N INSTANCE_NAME -x USER_NAME --sudo --run-list
 "recipe[nginx]"
```

2. 다음 명령을 실행하고, *INSTANCE_NAME*을 방금 추가한 인스턴스의 이름으로 바꿔 새 노드가 추가되었는지 확인합니다.

```
knife client show INSTANCE_NAME
```

```
knife node show INSTANCE_NAME
```

에서 노드 자동 추가 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 주제에서는 Amazon Elastic Compute Cloud(Amazon EC2) 노드를 Chef 서버에 자동으로 추가하는 방법을 설명합니다. 스타터 키트의 코드는 무인 방식을 사용하여 자동으로 노드를 추가하는 방법을 보여줍니다. 권장되는 새 노드의 무인(또는 자동) 연결 방법은 Chef Client 쿡북을 구성하는 것입니다. 스타터 키트의 `userdata` 스크립트를 사용할 수 있으며, 노드에 적용할 쿡북을 사용하여 `userdata` 스크립트의 `run_list` 섹션 또는 `Policyfile.rb`를 변경할 수 있습니다. `chef-client` 에이전트를 실행하기 전에 Chef Client 쿡북을 Chef 서버에 설치한 다음 아래 샘플 명령과 같이 예를 들면 HTTPD 역할을 사용하여 서비스 모드에서 `chef-client` 에이전트를 설치합니다.

```
chef-client -r "chef-client,role[httpd]"
```

Chef 서버와 통신하기 위해서는 `chef-client` 에이전트 소프트웨어에 클라이언트 노드의 퍼블릭 키에 대한 액세스 권한이 있어야 합니다. Amazon EC2에서 퍼블릭-프라이빗 키 페어를 생성한 다음 노드 이름을 사용하여 퍼블릭 키를 API 호출에 AWS OpsWorks `associate-node` 전달할 수 있습니다. 스타터 키트에 포함된 스크립트는 조직 이름, 서버 이름, 서버 엔드포인트를 자동으로 수집합니다. 이렇게 하면 노드가 Chef 서버에 연결되며, 노드에서 실행되는 `chef-client` 에이전트는 프라이빗 키 일치 후에 서버와 통신할 수 있습니다.

AWS OpsWorks for Chef Automate 서버와 관련된 노드에서 지원되는 최소 `chef-client` 버전은 13.x입니다. 안정적인 최신 `chef-client` 버전을 실행하는 것이 좋습니다.

노드의 연결을 해제하는 방법에 대한 자세한 내용은이 설명서AWS OpsWorks for Chef Automate 서버에서 노드 연결 해제의 및 AWS OpsWorks for Chef Automate API 설명서`disassociate-node`의 섹션을 참조하세요.

주제

- 지원되는 운영 체제
- 1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성
- 2단계: Chef Client 쿡북 설치
- 3단계: 무인 연결 스크립트를 사용하여 인스턴스 생성
- 그 밖의 chef-client 반복 실행 자동화 방법
- 관련 항목

지원되는 운영 체제

현재 노드에 지원되는 운영 체제 목록은 Chef 웹 사이트를 참조하세요.

1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성

EC2 인스턴스 프로파일로 사용할 AWS Identity and Access Management (IAM) 역할을 생성하고 다음 정책을 IAM 역할에 연결합니다. 이 정책은 노드 등록 중에 AWS OpsWorks for Chef Automate (`opsworks-cm`) API가 EC2 인스턴스와 통신하도록 허용합니다. 인스턴스 프로파일에 대한 자세한 내용은 Amazon EC2 설명서의 인스턴스 프로파일 사용하기를 참조하세요. IAM 역할을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 설명서의 콘솔에서 IAM 역할 생성을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

2단계: Chef Client 쿡북 설치

아직 수행하지 않은 경우, (대체 방법) Berkshelf를 사용하여 원격 소스에서 쿡북 가져오기의 단계에 따라 Berksfile 또는 `Policyfile.rb` 파일이 Chef Client 쿡북을 참조하고 쿡북을 설치하는지 확인합니다.

3단계: 무인 연결 스크립트를 사용하여 인스턴스 생성

1. EC2 인스턴스를 생성하려면 스타터 키트의 `userdata` 스크립트를 EC2 인스턴스 지침, Amazon EC2 Auto Scaling 그룹 시작 구성 또는 AWS CloudFormation 템플릿의 `userdata` 섹션으로 복사할 수 있습니다. 사용자 데이터에 스크립트 추가에 대한 자세한 내용은 Amazon EC2 사용 설명서의 시작 시 Linux 인스턴스에서 명령 실행을 참조하세요.

   이 스크립트는 `opsworks-cm` API `associate-node` 명령을 실행하여 새 노드를 Chef 서버와 연결합니다.

   기본적으로 새로 등록된 노드의 이름은 인스턴스 ID인데 `userdata` 스크립트에서 `NODE_NAME` 변수의 값을 수정하여 이 이름을 변경할 수 있습니다. Chef 콘솔 UI에서는 현재 조직 이름을 변경할 수 없기 때문에 `CHEF_AUTOMATE_ORGANIZATION`은 `default`로 설정된 상태로 둡니다.
2. EC2 설명서의 인스턴스 시작 단원에 나와 있는 절차를 따릅니다(여기에서 수정). EC2 인스턴스 시작 마법사에서 Amazon Linux AMI를 선택합니다.
3. 인스턴스 세부 정보 구성 페이지에서 1단계: 인스턴스 프로파일로 사용할 IAM 역할 생성에서 IAM 역할로 생성한 역할을 선택합니다.
4. Advanced Details(고급 정보) 영역에서, 이 절차 앞부분에서 생성한 `userdata.sh` 스크립트를 업로드합니다.
5. [스토리지 추가] 페이지에서 변경해야 할 사항은 없습니다. [태그 추가]로 이동합니다.
6. 이 예제에서는 보안 그룹 구성 페이지에서 규칙 추가를 선택한 후 HTTP를 입력하여 Apache 웹 서버에서 443 포트와 80 포트를 엽니다.
7. [검토 및 시작]을 선택한 다음 [시작]을 선택합니다. 새 노드가 시작되면 노드는 `RUN_LIST` 파라미터에 지정한 레시피에 지정된 구성을 적용합니다.
8. 선택 사항: `nginx` 쿡북을 실행 목록에 추가한 경우, 새 노드의 퍼블릭 DNS에 연결된 웹 페이지를 열면 nginx 웹 서버가 호스팅하는 웹 사이트가 표시되어야 합니다.

그 밖의 **`chef-client`** 반복 실행 자동화 방법

달성이 더 어렵고 권장되지는 않지만 이 주제의 스크립트는 독립 실행형 인스턴스 사용자 데이터의 일부로만 실행하거나, AWS CloudFormation 템플릿을 사용하여 새 인스턴스 사용자 데이터에 추가하거나, 스크립트를 정기적으로 실행하도록 `cron` 작업을 구성하거나, 서비스 `chef-client` 내에서 실행할 수 있습니다. 하지만 다른 자동화 기법에는 몇 가지 단점이 있기 때문에 Chef Client Cookbook 방법을 사용하는 것이 좋습니다.

`chef-client`에 제공할 수 있는 파라미터의 완전한 목록은 Chef 설명서를 참조하세요.

관련 항목

다음 AWS 블로그 게시물은 Auto Scaling 그룹을 사용하거나 여러 계정 내에서 노드를 Chef Automate 서버와 자동으로 연결하는 방법에 대한 자세한 정보를 제공합니다.

- Using AWS OpsWorks for Chef Automate to Manage EC2 Instances with Auto Scaling
- OpsWorks for Chef Automate - 서로 다른 계정에서 노드 자동 구성

Chef Automate 대시보드에 로그인

Important

AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

Chef 서버의 Properties 페이지에서 로그인 자격 증명을 다운로드하고 서버가 온라인 상태가 된 후 Chef Automate 대시보드에 로그인합니다. 이 안내서에서는 먼저 쿡북을 업로드하고 관리할 노드를 하나 이상 추가하라고 설명했습니다. 이렇게 하면 대시보드에서 쿡북과 노드에 대한 정보를 볼 수 있습니다.

대시보드 웹 페이지에 연결하려고 하면 Chef 서버를 관리하는 데 사용하는 클라이언트 컴퓨터에 AWS OpsWorks특정 CA 서명 SSL 인증서를 설치할 때까지 브라우저에 인증서 경고가 표시됩니다. 대시보드 웹 페이지로 가기 전에 경고를 보지 않으려면 로그인 전에 SSL 인증서를 설치하세요.

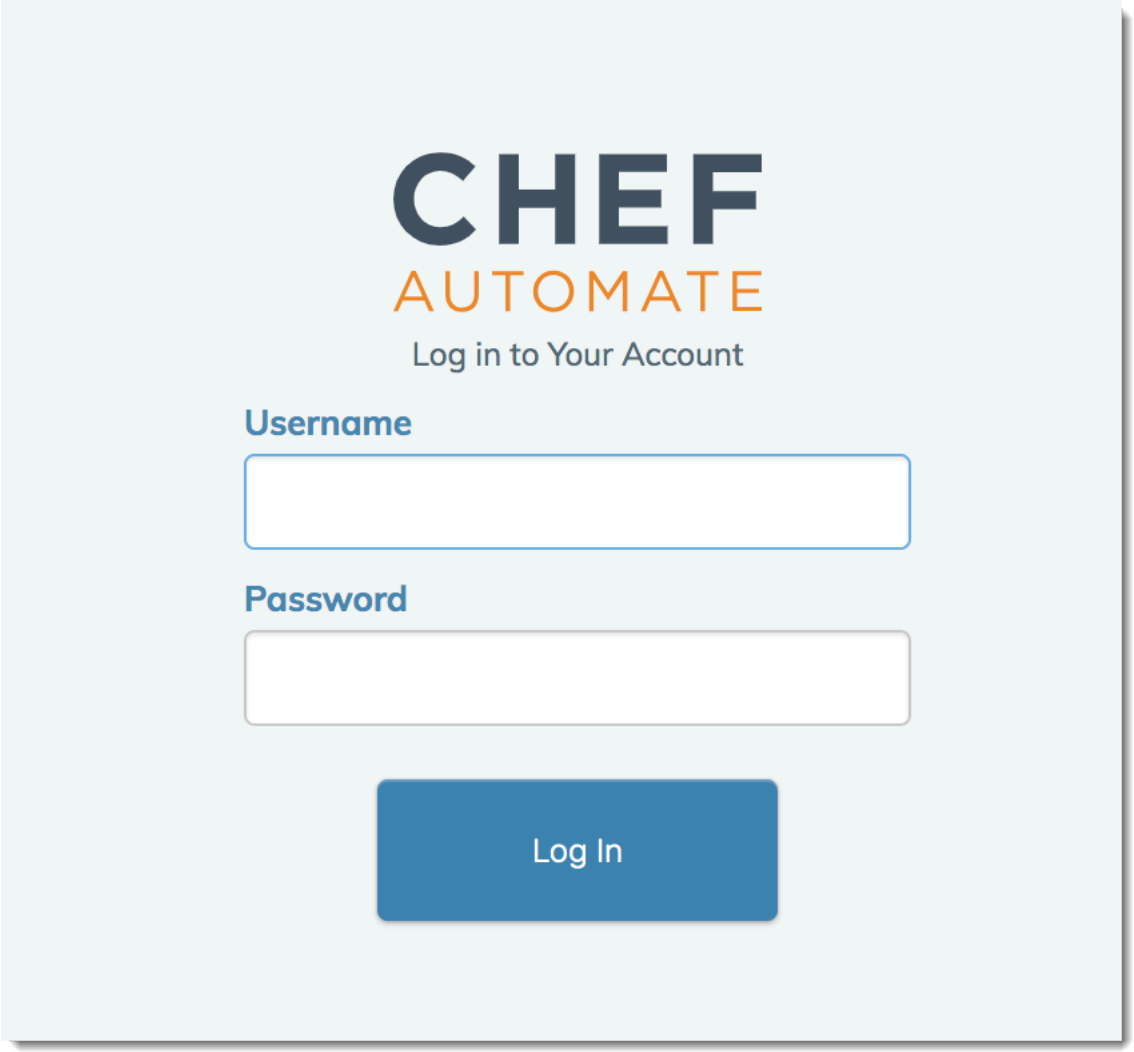
클라이언트 측 SSL 인증서를 설치한 후에는 경고 메시지를 보지 않고 Chef Automate 대시보드에 로그인할 수 있습니다.

Note

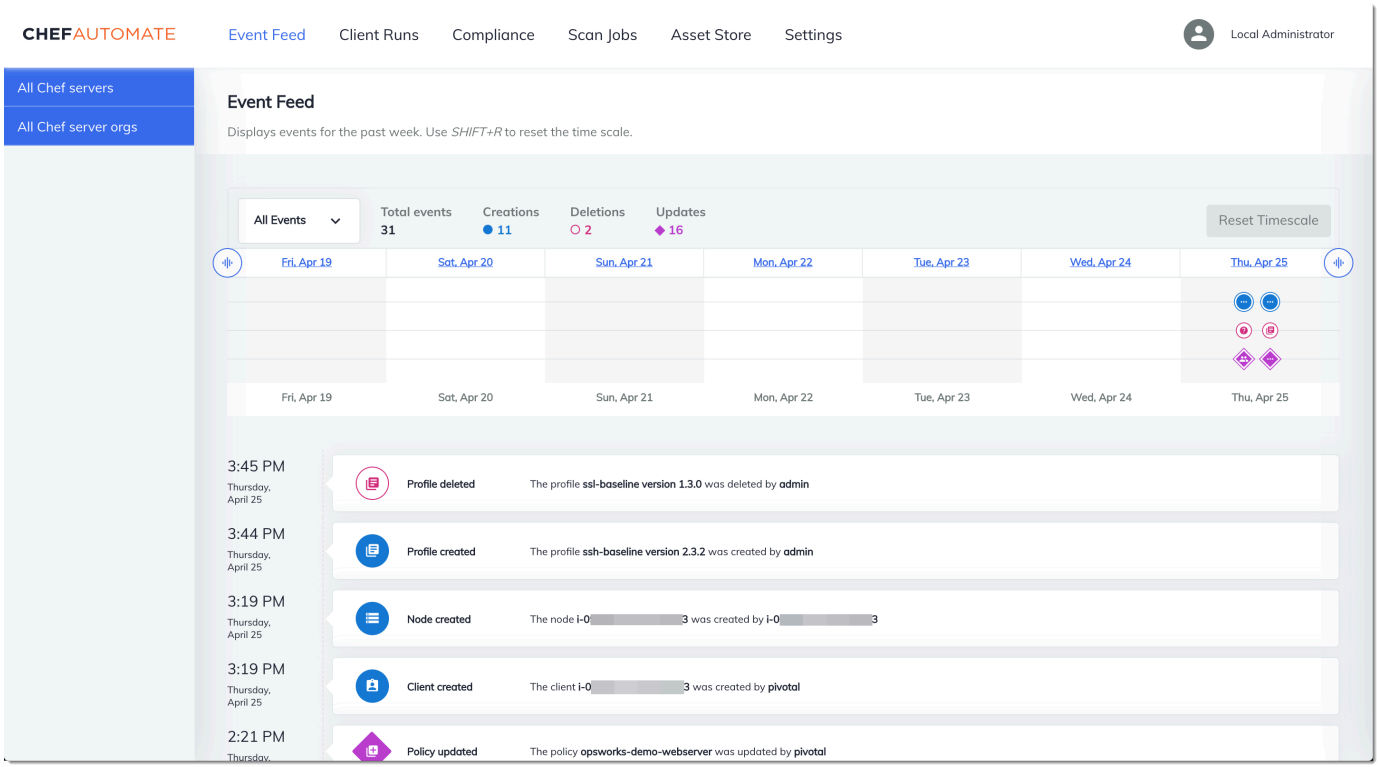
Ubuntu 및 Linux Mint 운영 체제의 Google Chrome 사용자는 로그인이 어려울 수 있습니다. 이러한 운영 체제에서 Chef Automate 대시보드에 로그인하고 대시보드를 사용하려면 Mozilla Firefox나 기타 브라우저를 사용하는 것이 좋습니다. Windows나 MacOS에서의 Google Chrome 사용에는 알려진 문제가 없습니다.

Chef Automate 대시보드에 로그인하려면

1. 사전 조건 단원에서 다운로드한 Chef Automate 자격 증명의 압축을 풀어 엽니다. 로그인하려면 이 자격 증명이 필요합니다.
2. Chef 서버의 [속성] 페이지를 엽니다.
3. [속성] 페이지의 오른쪽 위에서 [Chef Automate 대시보드 열기]를 선택합니다.
4. 1단계의 자격 증명으로 로그인합니다.



5. Chef Automate 대시보드에서는 부트스트랩한 노드에 대한 자세한 정보, 쿡북 실행 진행 상황, 이벤트, 노드의 규정 준수 수준 등을 확인할 수 있습니다. Chef Automate 대시보드의 기능과 이러한 기능의 사용 방법에 대한 자세한 정보는 Chef Automate 설명서를 참조하세요.



Note

Chef Automate 대시보드 로그인에 사용하는 암호를 변경하는 방법은 Chef Automate 대시보드 자격 증명 재설정 단원을 참조하세요.

를 사용하여 AWS OpsWorks for Chef Automate 서버 생성 AWS CloudFormation

Important

AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

AWS OpsWorks for Chef Automate 를 사용하면에서 Chef Automate 서버를 실행할 수 있습니다 AWS. 약 15분 안에 Chef Automate 서버를 프로비저닝할 수 있습니다.

2021년 5월 3일부터 일부 Chef Automate 서버 속성을에 AWS OpsWorks for Chef Automate 저장합니다 AWS Secrets Manager. 자세한 내용은 와 통합 AWS Secrets Manager 단원을 참조하십시오.

다음 연습은에서 스택을 생성 AWS OpsWorks for Chef Automate 하여에서 서버를 생성하는 데 도움이 됩니다 AWS CloudFormation.

주제

- 사전 조건
- AWS CloudFormation에서 Chef Automate 서버 만들기

사전 조건

Chef Automate 서버를 새로 만들기 전에 먼저 Chef 서버에 액세스하고 관리하는 데 필요한 리소스를 AWS OpsWorks for Chef Automate 외부에서 생성하세요. 자세한 내용은 이 설명서의 시작하기 단원에서 사전 조건를 참조하세요.

AWS CloudFormation 사용 설명서 템플릿 참조의 OpsWorks-CM 섹션을 검토하여 서버를 생성하는 데 사용하는 AWS CloudFormation 템플릿에서 지원되는 값과 필요한 값에 대해 알아봅니다.

사용자 지정 도메인을 사용하는 서버를 만드는 경우 사용자 지정 도메인, 인증서 및 프라이빗 키가 필요합니다. AWS CloudFormation 템플릿에서 이러한 세 파라미터 모두에 값을 지정해야 합니다. `CustomDomain`, `CustomCertificate` 및 `CustomPrivateKey` 매개 변수의 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer 항목을 참조하세요.

`CHEF_AUTOMATE_ADMIN_PASSWORD` 엔진 속성에 대한 암호 값을 생성합니다. 암호 길이는 최소 8자이며, 최대 32자입니다. 암호는 문자, 숫자 및 특수 문자 (`!/@#$%^+=_`)를 포함할 수 있습니다. 암호에는 소문자, 대문자, 숫자 및 특수 문자가 각각 1개 이상 포함되어야 합니다. AWS CloudFormation 템플릿에서 또는 스택을 생성할 때 `CHEF_AUTOMATE_ADMIN_PASSWORD` 파라미터 값으로이 암호를 지정합니다.

Chef Automate 서버 생성을 시작하기 전에 base64로 인코딩된 RSA 키 페어를 생성합니다 AWS CloudFormation. 이 페어의 퍼블릭 키는 CreateServer API의 `CHEF_AUTOMATE_PIVOTAL_KEY` Chef 전용 엔진 속성인 의 값입니다. 이 키는 AWS CloudFormation 콘솔 또는의 create-stack 명령에서 파라미터 값으로 제공됩니다 AWS CLI. 이 키를 생성할 때 다음 방법을 사용하는 것이 좋습니다.

- Linux 기반 컴퓨터에서는 다음 OpenSSL 명령을 실행하여 이 키를 생성할 수 있습니다.

```
openssl genrsa -out pivotal_key_file_name.pem 2048
```

그런 다음, 페어의 RSA 퍼블릭 키 부분을 파일로 내보냅니다. 퍼블릭 키가 CHEF_AUTOMATE_PIVOTAL_KEY의 값이 됩니다.

```
openssl rsa -in pivotal_key_file_name.pem -pubout -out public.pem -outform PEM
```

- Windows 기반 컴퓨터에서는 PuTTYgen 유틸리티를 사용하여 base64 인코딩 RSA 키 페어를 생성할 수 있습니다. 자세한 내용은 SSH.com.에서 PuTTYgen - Windows에서 PuTTY용 키 생성을 참조하세요.

AWS CloudFormation에서 Chef Automate 서버 만들기

이 섹션에서는 AWS CloudFormation 템플릿을 사용하여 AWS OpsWorks for Chef Automate 서버를 생성하는 스택을 빌드하는 방법을 설명합니다. AWS CloudFormation 콘솔 또는를 사용하여이 작업을 수행할 수 있습니다 AWS CLI. 예제 AWS CloudFormation 템플릿을 사용하여 AWS OpsWorks for Chef Automate 서버 스택을 빌드할 수 있습니다. 예제 템플릿을 서버 이름, IAM 역할, 인스턴스 프로파일, 서버 설명, 백업 보존 수, 유지 관리 옵션 및 옵션 태그로 업데이트하세요. 서버에서 사용자 지정 도메인을 사용하는 경우 AWS CloudFormation 템플릿에서 CustomDomain, CustomCertificate및 CustomPrivateKey 파라미터 값을 지정해야 합니다. AWS CloudFormation 템플릿에서 CHEF_AUTOMATE_ADMIN_PASSWORD 및 CHEF_AUTOMATE_PIVOTAL_KEY 엔진 속성과 해당 값을 지정하거나 속성만 제공한 다음 AWS CloudFormation 스택 생성 마법사 또는 create-stack 명령에서 속성 값을 지정할 수 있습니다. 이러한 속성에 대한 자세한 내용은 이 설명서의 시작하기 단원에서the section called “에서 Chef Automate 서버 생성 AWS Management Console” 항목을 참조하세요.

주제

- AWS CloudFormation 을 사용하여 Chef Automate 서버 만들기(콘솔)
- AWS CloudFormation 를 사용하여 Chef Automate 서버 만들기(CLI)

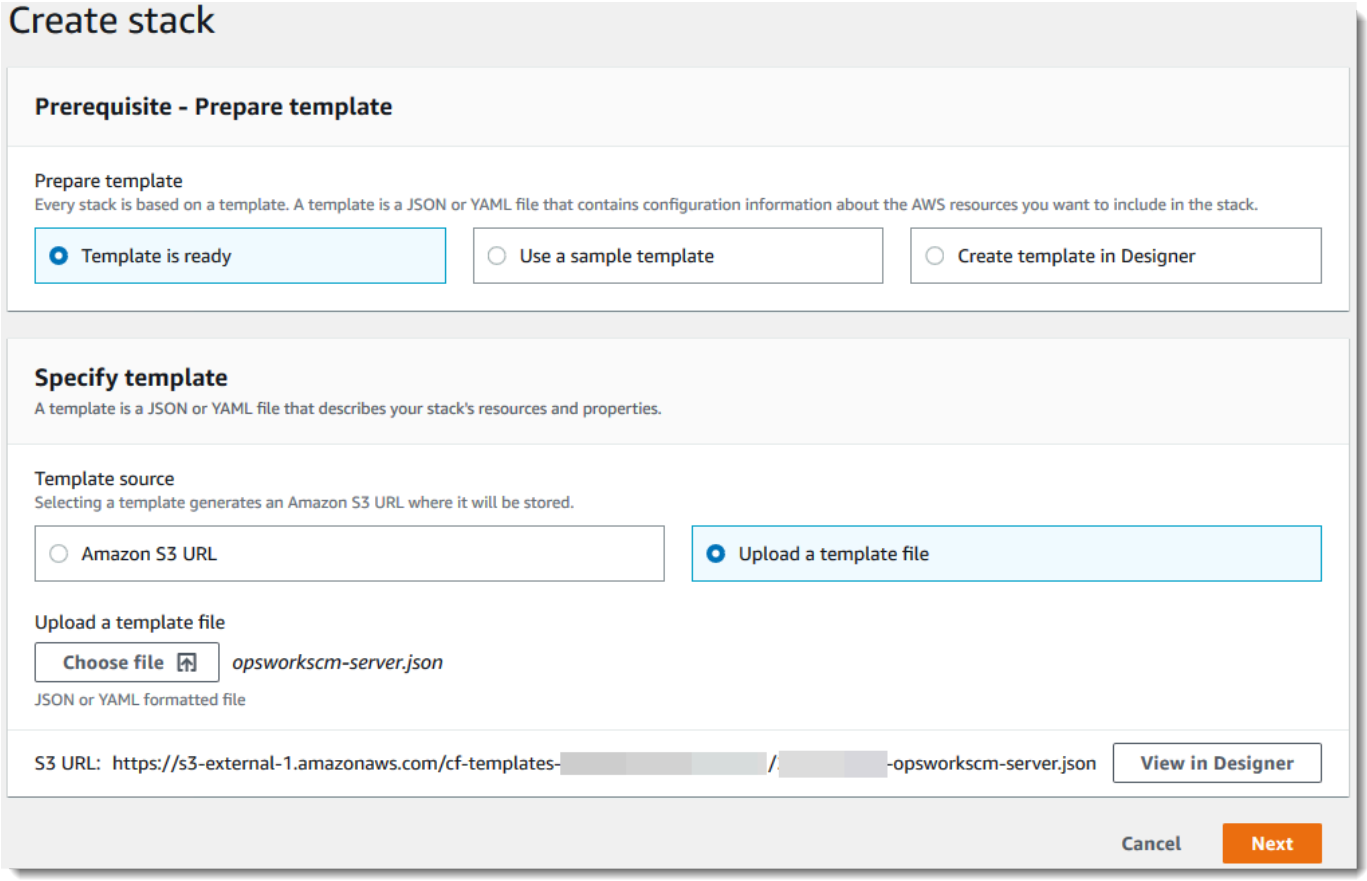
AWS CloudFormation 을 사용하여 Chef Automate 서버 만들기(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation AWS CloudFormation 콘솔을 엽니다.
2. AWS CloudFormation 홈 페이지에서 스택 생성을 선택합니다.

3. 사전 조건 - 템플릿 준비에서 예제 AWS CloudFormation 템플릿을 사용하는 경우 템플릿 준비 완료를 선택합니다.

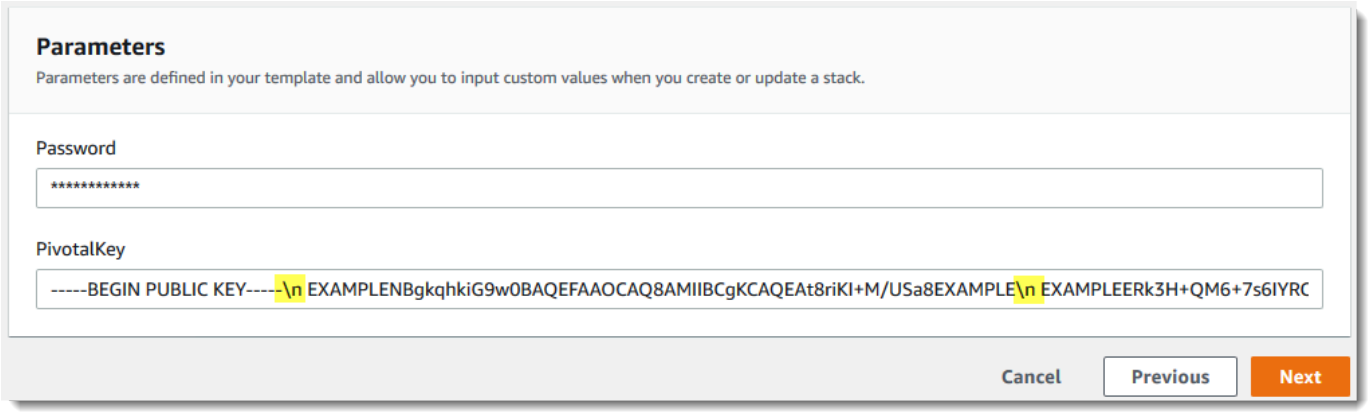
4. 템플릿 지정에서 템플릿의 소스를 선택합니다. 이 연습에서는 템플릿 파일 업로드를 선택하고 Chef Automate 서버를 생성하는 AWS CloudFormation 템플릿을 업로드합니다. 템플릿 파일을 찾은 후 다음을 선택합니다.

   AWS CloudFormation 템플릿은 YAML 또는 JSON 형식일 수 있습니다. 예제 AWS CloudFormation 템플릿을 사용할 수 있습니다. 예제 값을 자신의 값으로 바꿔야 합니다. AWS CloudFormation 템플릿 디자이너를 사용하여 새 템플릿을 빌드하거나 기존 템플릿을 검증할 수 있습니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 디자이너 인터페이스 개요를 참조하세요.



5. 스택 세부 정보 지정 페이지에서 스택의 이름을 입력합니다. 이것은 서버 이름이 아니고 스택 이름입니다. the section called “사전 조건”에서 만든 값을 매개변수 영역에 붙여넣습니다. 암호에 암호를 입력합니다.

   PivotalKey에 RSA 키 파일의 콘텐츠를 붙여 넣습니다. AWS CloudFormation 콘솔에서 다음 스크린샷과 같이 중심 키 값의 각 줄 끝에 줄 바꿈(**\n**) 문자를 추가해야 합니다. 다음을 선택합니다.



6. 스택 옵션 구성 페이지에서, 스택을 사용하여 생성할 서버에 태그를 추가하고, 템플릿에서 사용할 IAM 역할을 아직 지정하지 않은 경우 리소스 생성을 위한 IAM 역할을 선택할 수 있습니다. 옵션 지정을 마쳤으면 다음을 선택합니다. 롤백 트리거와 같은 고급 옵션에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 스택 옵션 설정을 참조하세요.
7. [검토] 페이지에서 선택 사항을 검토합니다. 서버 스택을 생성할 준비가 되면 스택 생성을 선택합니다.

   가 스택 AWS CloudFormation 을 생성할 때까지 기다리는 동안 스택 생성 상태를 확인합니다. 스택 생성이 실패하면 콘솔에 표시된 오류 메시지를 검토하여 문제를 해결할 수 있습니다. AWS CloudFormation 스택의 오류 문제 해결에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 오류 문제 해결을 참조하세요.

   서버 생성이 끝나면 AWS OpsWorks for Chef Automate 홈 페이지에서 Chef Automate 서버를 확인할 수 있으며, 상태는 온라인입니다. 서버의 속성 페이지에서 새로운 스타터 키트와 Chef Automate 대시보드 자격 증명을 생성합니다. 서버가 온라인 상태가 되면 `https://your_server_name-randomID.region.opsworks-cm.io` 형식의 URL을 가진 서버 도메인에서 Chef Automate 대시보드를 사용할 수 있습니다.

   > **Note**
   >
   > 서버에 대한 사용자 지정 도메인, 인증서 및 프라이빗 키를 지정한 경우 엔터프라이즈의 DNS 관리 도구에서 사용자 지정 도메인을 서버에 대해 AWS OpsWorks for Chef Automate 자동으로 생성된 엔드포인트에 매핑하는 CNAME 항목을 생성합니다. 생성된 엔드포인트를 사용자 지정 도메인 값으로 매핑할 때까지 서버를 관리하거나 서버에 대한 Chef Automate 대시보드에 연결할 수 없습니다.
   > 생성된 엔드포인트 값을 가져오려면 서버가 온라인 상태가 된 후 다음 AWS CLI 명령을 실행합니다.

```
aws opsworks describe-servers --server-name server_name
```

AWS CloudFormation 를 사용하여 Chef Automate 서버 만들기(CLI)

로컬 컴퓨터에서 아직를 실행하지 않는 경우 AWS Command Line Interface 사용 설명서의 설치 지침에 AWS CLI 따라를 AWS CLI다운로드하고 설치합니다. 이 단원에서는 create-stack 명령과 함께 사용할 수 있는 모든 파라미터를 다 설명하지는 않습니다. create-stack 파라미터에 대한 자세한 내용은 AWS CLI 레퍼런스의 create-stack를 참조하세요.

1. AWS OpsWorks for Chef Automate 서버 생성에 대한 사전 조건를 완료해야 합니다.
2. 서비스 역할과 인스턴스 프로파일을 생성합니다.

   계정에서 서비스 역할의 ARNs 찾아 복사합니다.

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   `list-roles` 명령의 결과에서 다음과 비슷한 서비스 역할 및 인스턴스 프로파일을 찾아봅니다. 서비스 역할 및 인스턴스 프로파일의 ARNs을 기록해 두고 서버 스택을 생성하는 데 사용하는 AWS CloudFormation 템플릿에 추가합니다.

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
       "Path": "/service-role/",
       "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
   ```

```
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. create-stack 명령을 다시 실행하여 AWS OpsWorks for Chef Automate 서버를 생성합니다.

   - *stack_name*을 스택 이름으로 바꿉니다. Chef Automate 서버가 아닌 AWS CloudFormation 스택의 이름입니다. Chef Automate 서버 이름은 AWS CloudFormation 템플릿`ServerName`의 값입니다.
   - *template*을 해당 템플릿 파일의 경로로 바꾸고 *yaml ## json* 확장명을 `.yaml` 또는 `.json`으로 적절히 바꿉니다.
   - `--parameters`의 값은 CreateServer API의 엔진 속성에 해당합니다. Chef의 경우, 서버를 생성하기 위해 사용자가 제공하는 엔진 속성은 the section called "사전 조건"에서 설명한 유틸리티를 사용하여 생성한 base64 인코딩 형식의 RSA 퍼블릭 키 `CHEF_AUTOMATE_PIVOTAL_KEY`와, 8-32자로 생성한 암호인 `CHEF_AUTOMATE_ADMIN_PASSWORD`입니다. `CHEF_AUTOMATE_ADMIN_PASSWORD`에 대한 자세한 내용은 를 사용하여 Chef Automate 서버 생성 AWS CLI 섹션을 참조하세요. 예제에 나와 있는 것처럼, `PivotalKey` 파라미터의 값으로 해당 값을 포함하는 PEM 파일에 대한 포인터를 제공할 수 있습니다. `CHEF_AUTOMATE_ADMIN_PASSWORD` 및의 값이 템플릿에 지정`CHEF_AUTOMATE_PIVOTAL_KEY`되지 않은 경우 AWS CLI 명령에 값을 제공해야 합니다.

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=PivotalKey,ParameterValue="base64_encoded_RSA_public_key_value"
```

다음은 CHEF_AUTOMATE_ADMIN_PASSWORD 및 CHEF_AUTOMATE_PIVOTAL_KEY 속성의 샘플 값을 보여 주는 예제입니다. AWS CloudFormation 템플릿에서 이러한 속성에 대한 값을 지정하지 않은 경우 유사한 명령을 실행합니다.

```
aws cloudformation create-stack --stack-name "OpsWorksCMChefServerStack"
 --template-body file://opsworkscm-server.yaml --parameters
 ParameterKey=PivotalKey,ParameterValue="$(openssl rsa -in "pivotalKey.pem" -
pubout)" ParameterKey=Password,ParameterValue="SuPer\$ecret890"
```

4. 스택 생성이 완료되면 AWS OpsWorks for Chef Automate 콘솔에서 새 서버의 속성 페이지를 열고 스타터 키트를 다운로드합니다. 새 스타터 키트를 다운로드하면 Chef Automate 대시보드 관리자 암호를 재설정합니다.
5. 서버에서 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용할 경우 (선택 사항) 사용자 지정 도메인을 사용하도록 knife 구성의 구성 knife.rb 단계를 수행한 다음 7단계로 이동하세요.
6. 새로운 서버에서 knife 명령을 사용하여 Chef knife.rb 구성 파일 설정을 업데이트합니다. 스타터 키트에는 knife.rb 예제 파일이 포함되어 있습니다. 다음 예제에서는 사용자 지정 도메인을 사용하지 않는 서버에서의 knife.rb 설정 방법을 보여줍니다. 사용자 지정 도메인을 사용하는 경우 (선택 사항) 사용자 지정 도메인을 사용하도록 knife 구성에서 knife 구성 지침을 참조하세요.

   - *ENDPOINT*를 서버의 엔드포인트 값으로 바꿉니다. 이것은 스택 생성 작업 출력의 일부입니다. 다음 명령을 실행하여 엔드포인트를 확인할 수 있습니다.

     ```
     aws cloudformation describe-stacks --stack-name stack_name
     ```

   - client_key 구성의 *key_pair_file.pem#* 서버를 만드는 데 사용한 CHEF_AUTOMATE_PIVOTAL_KEY이 들어 있는 PEM 파일 이름으로 바꾸세요.

     ```
     base_dir = File.join(File.dirname(File.expand_path(__FILE__)), '..')

     log_level                :info
     log_location             STDOUT
     node_name                'pivotal'
     ```

```
client_key               File.join(base_dir, '.chef', 'key_pair_file.pem')
syntax_check_cache_path  File.join(base_dir, '.chef', 'syntax_check_cache')
cookbook_path            [File.join(base_dir, 'cookbooks')]

chef_server_url          'ENDPOINT/organizations/default'
ssl_ca_file              File.join(base_dir, '.chef', 'ca_certs', 'opsworks-cm-
ca-2020-root.pem')
trusted_certs_dir        File.join(base_dir, '.chef', 'ca_certs')
```

7. 서버 생성 프로세스가 완료되면 the section called “구성 완료 및 쿡북 업로드” 단원을 진행합니다. 스택 생성이 실패하면 콘솔에 표시된 오류 메시지를 검토하여 문제를 해결할 수 있습니다. AWS CloudFormation 스택의 오류 문제 해결에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 오류 문제 해결을 참조하세요.

사용자 지정 도메인을 사용하도록 AWS OpsWorks for Chef Automate 서버 업데이트

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 섹션에서는 AWS OpsWorks for Chef Automate 서버 백업을 사용하여 새 서버를 생성하여 사용자 지정 도메인 및 인증서를 사용하도록 기존 서버를 업데이트하는 방법을 설명합니다. 기본적으로 백업에서 새 서버를 생성한 다음 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하도록 새 서버를 구성하여 기존 AWS OpsWorks for Chef Automate 2.0 서버를 복사합니다.

주제

- 사전 조건
- 제한 사항
- 사용자 지정 도메인을 사용하도록 서버 업데이트
- 참고

사전 조건

다음은 사용자 지정 도메인 및 인증서를 사용하도록 기존 AWS OpsWorks for Chef Automate 서버를 업데이트하기 위한 요구 사항입니다.

- 업데이트(또는 복사)하려는 서버에서 Chef Automate 2.0을 실행 중이어야 합니다.
- 새 서버를 만드는 데 사용할 백업을 결정합니다. 업데이트하고자 하는 서버에 사용 가능한 백업이 하나 이상 있어야 합니다. 의 백업에 대한 자세한 내용은 섹션을 AWS OpsWorks for Chef Automate참조하세요AWS OpsWorks for Chef Automate 서버 백업.
- 백업 소스인 기존 서버를 만드는 데 사용한 서비스 역할 및 인스턴스 프로파일 ARN을 준비합니다.
- AWS CLI의 최신 릴리스를 실행 중이어야 합니다. AWS CLI 도구 업데이트에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 설치를 AWS CLI 참조하세요.

제한 사항

백업에서 새 서버를 생성하여 기존 서버를 업데이트할 때 새 서버는 기존 AWS OpsWorks for Chef Automate 서버와 정확히 동일할 수 없습니다.

- AWS CLI 또는 AWS SDKs. AWS Management Console사용을 통해 백업에서 새 서버를 생성할 수 없습니다.
- 새 서버는 계정 내, 그리고 AWS 리전 내에 있는 기존 서버와 동일한 이름을 사용할 수 없습니다. 이름은 백업 소스로 사용한 기존 서버와 달라야 합니다.
- 기존 서버에 연결된 노드는 새 서버에서 관리하지 않습니다. 다음 중 하나를 수행해야 합니다.
  - 둘 이상의 Chef Automate 서버에서 노드를 관리할 수 없으므로 다른 노드를 연결합니다.
  - 기존 서버(백업 소스)에서 새 서버 및 새 사용자 지정 도메인 엔드포인트로 노드를 마이그레이션합니다. 노드를 마이그레이션하는 방법에 대한 자세한 내용은 Chef 설명서를 참조하세요.

사용자 지정 도메인을 사용하도록 서버 업데이트

기존 Chef Automate 2.0 서버를 업데이트하려면 `create-server` 명령을 실행하고 백업, 사용자 지정 도메인, 사용자 지정 인증서 및 사용자 지정 프라이빗 키를 지정하는 파라미터를 추가하여 서버의 복사본을 만듭니다.

1. `create-server` 명령에서 지정할 수 있는 서비스 역할 또는 인스턴스 프로파일 ARN이 없는 경우 를 사용하여 Chef Automate 서버 생성 AWS CLI의 1~5단계를 수행하여 사용할 수 있는 서비스 역할 및 인스턴스 프로필을 만듭니다.

2. 아직 수행하지 않은 경우 사용자 지정 도메인이 있는 새 서버의 기반이 될 기존 Chef Automate 2.0 서버의 백업을 찾습니다. 다음 명령을 실행하여 계정 및 리전의 모든 AWS OpsWorks for Chef Automate 백업에 대한 정보를 표시합니다. 사용할 백업 ID를 적어 둡니다.

```
aws opsworks-cm --region region name describe-backups
```

3. `create-server` 명령을 실행하여 AWS OpsWorks for Chef Automate 서버를 생성합니다.

   - `--engine` 값은 `ChefAutomate`, `--engine-model` 값은 `Single`, 그리고 `--engine-version` 값은 `12`입니다.
   - 서버 이름은 AWS 계정 내에서 각 리전 내에서 고유해야 합니다. 서버 이름은 문자로 시작해야 하며, 그 이후에는 문자, 숫자 또는 하이픈(-)을 사용할 수 있으며, 최대 길이는 40자입니다.
   - 1단계의 인스턴스 프로파일 ARN 및 서비스 역할 ARN을 사용합니다.
   - 유효한 인스턴스 유형은 `m5.large`, `r5.xlarge` 또는 `r5.2xlarge`입니다. 인스턴스 유형의 사양에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하세요.
   - `--engine-attributes` 파라미터는 선택 사항이며, 하나 또는 두 값을 모두 지정하지 않을 경우 서버 생성 프로세스에서 해당 값이 자동으로 생성됩니다. `--engine-attributes`를 추가할 경우 2단계에서 생성한 `CHEF_AUTOMATE_PIVOTAL_KEY` 값, `CHEF_AUTOMATE_ADMIN_PASSWORD` 또는 두 가지 모두를 지정합니다.

     `CHEF_AUTOMATE_ADMIN_PASSWORD`의 값을 설정하지 않으면 암호가 생성되어 `create-server` 응답의 일부로서 반환됩니다. 또한 콘솔에서 스타터 키트를 다시 다운로드할 수도 있습니다. 그러면 이 암호가 다시 생성됩니다. 암호 길이는 최소 8자이며, 최대 32자입니다. 암호는 문자, 숫자 및 특수 문자(`!/@#$%^+=_`)를 포함할 수 있습니다. 암호에는 소문자, 대문자, 숫자 및 특수 문자가 각각 1개 이상 포함되어야 합니다.
   - SSH 키 페어는 선택 사항이지만, Chef Automate 대시보드 관리자 암호를 재설정해야 하는 경우 Chef Automate 서버에 연결하는 데 도움이 될 수 있습니다. SSH 키 페어 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.
   - 사용자 지정 도메인을 사용하려면 명령에 다음 파라미터를 추가합니다. 그렇지 않은 경우 Chef Automate의 서버 생성 프로세스가 자동으로 엔드포인트를 생성합니다. 사용자 지정 도메인을 구성하려면 세 가지 파라미터 모두가 필요합니다. 이러한 파라미터 사용에 대한 추가 요구 사항에 대한 자세한 내용은 AWS OpsWorks CM API 참조의 CreateServer를 참조하세요.
     - `--custom-domain` - 서버의 선택적 퍼블릭 엔드포인트(예: `https://aws.my-company.com`).
     - `--custom-certificate` - PEM 형식의 HTTPS 인증서. 값은 자체 서명된 단일 인증서 또는 인증서 체인일 수 있습니다.

  - `--custom-private-key` - HTTPS를 사용하여 서버에 연결하기 위한 PEM 형식의 프라이빗 키. 프라이빗 키는 암호화해서는 안 되며 암호로 보호할 수 없습니다.
- 매주 시스템 유지 관리가 필요합니다. 유효한 값은 `DDD:HH:MM` 형식으로 지정해야 합니다. 지정한 시간은 협정 세계시(UTC)로 표시됩니다. `--preferred-maintenance-window`의 값을 지정하지 않으면 기본값은 화요일, 수요일 또는 금요일의 임의 한 시간입니다.
- `--preferred-backup-window`의 유효 값은 `HH:MM`(매일 백업) 또는 `DDD:HH:MM`(매주 백업) 형식 중 하나로 지정해야 합니다. 지정한 시간은 UTC 형식입니다. 기본값은 임의 일일 시작 시간입니다. 자동 백업을 옵트아웃하려면 대신에 `--disable-automated-backup` 파라미터를 추가합니다.
- `--security-group-ids`에는 공백으로 구분하여 하나 이상의 보안 그룹 ID를 입력합니다.
- `--subnet-ids`에는 서브넷 ID를 입력합니다.
- `--backup-id`에는 2단계에서 복사한 백업의 ID를 입력합니다.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID --backup-id backup_ID
```

다음 예제에서는 사용자 지정 도메인을 사용하는 Chef Automate 서버를 만듭니다.

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
```

```
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --backup-id MyChefServer-20191004122143125
```

4. AWS OpsWorks for Chef Automate 새 서버를 생성하는 데 약 15분이 걸립니다. `create-server` 명령의 출력에서 `Endpoint` 속성 값을 복사합니다. 다음은 예입니다.

```
"Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

`create-server` 명령의 출력을 무시하거나 셸 세션을 닫지 마십시오. 다시 표시되지 않는 중요 정보가 출력에 포함되어 있을 수도 있기 때문입니다. `create-server` 결과에서 암호 및 스타터 키트를 가져오려면 다음 단계로 이동합니다.

5. 키와 암호를 AWS OpsWorks for Chef Automate 생성하도록 선택한 경우 jq와 같은 JSON 프로세서를 사용하여 `create-server` 결과에서 사용 가능한 형식으로 추출할 수 있습니다. jq를 설치한 후에는 다음 명령을 실행하여 중심 키, Chef Automate 대시보드 관리자 암호 및 스타터 키트를 추출할 수 있습니다. 3단계에서 고유의 중심 키 및 암호를 제공하지 않은 경우 추출한 중심 키 및 관리자 암호를 안전하면서 편리한 위치에 저장해야 합니다.

```
#Get the Chef password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

#Get the Chef Pivotal Key:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

#Get the Chef Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

6. 선택적으로 `create-server` 명령 결과에서 스타터 키트를 추출하지 않은 경우 AWS OpsWorks for Chef Automate 콘솔의 서버 속성 페이지에서 새 스타터 키트를 다운로드할 수 있습니다. 새 스타터 키트를 다운로드하면 Chef Automate 대시보드 관리자 암호를 재설정합니다.

7. 엔터프라이즈의 DNS 관리 도구에 CNAME 항목을 생성하여 4단계에서 복사한 AWS OpsWorks for Chef Automate 엔드포인트를 사용자 지정 도메인을 가리킵니다. 이 단계를 완료해야 서버에 연결하거나 로그인할 수 있습니다.
8. 서버 생성 프로세스가 완료되면 the section called “구성 완료 및 쿡북 업로드” 단원을 진행합니다.

참고

- 를 사용하여 Chef Automate 서버 생성 AWS CLI
- 백업에서 AWS OpsWorks for Chef Automate 서버 복원
- AWS OpsWorks CM API 참조의 CreateServer
- AWS CLI 명령 Reference의 create-server

AWS OpsWorks for Chef Automate 서버의 스타터 키트 재생성

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

용 스타터 키트에는 예제가 포함된 README 파일, `knife.rb` 구성 파일 및 기본 또는 중추적 사용자를 위한 프라이빗 키가 AWS OpsWorks for Chef Automate 포함되어 있습니다. Starter Kit를 다운로드할 때마다 새 키 페어가 생성되고 예전 키가 리셋됩니다. 다음 두 가지 방법 중 하나로 AWS OpsWorks for Chef Automate 서버의 스타터 키트를 재생성할 수 있습니다.

- AWS OpsWorks 콘솔의 AWS OpsWorks for Chef Automate 서버 세부 정보 페이지의 작업 메뉴에서 이전 피벗 키를 재생성하고 재설정할지 여부를 확인하라는 메시지가 표시됩니다.
- 에서 명령을 실행합니다 AWS CLI.

스타터 키트 사용 방법에 대한 자세한 내용은 Starter Kit를 사용하여 Chef 서버 구성 섹션을 참조하세요.

를 사용하여 AWS OpsWorks for Chef Automate 스타터 키트 재생성 AWS CLI

> **Note**
>
> 스타터 키트를 다시 생성할 때 Chef Automate 서버의 인증 키 페어도 재생성 및 재설정하고 현재 키 페어를 삭제합니다.

update-server-engine-attributes 명령을 실행하여 스타터 키트를 재생성하세요. AWS CLI 세션에서 다음 명령을 실행합니다. 서버 이름을 `--server-name`의 값으로 지정합니다. 자체 퍼블릭 키를 `CHEF_AUTOMATE_PIVOTAL_KEY`의 값으로 설정하려면 `--attribute-value`에서 퍼블릭 키 값을 지정하세요. 그렇지 않으면 `--attribute-value`를 null로 설정합니다.

```
aws opsworks-cm update-server-engine-attributes \
   --server-name server_name \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value your_public_key
```

다음 명령은 서버 관리자가 사용하려는 퍼블릭 키 값을 지정하는 예입니다.

```
aws opsworks-cm update-server-engine-attributes \
   --server-name your-test-server \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value "-----BEGIN PUBLIC KEY-----ExamplePublicKey-----END PUBLIC
 KEY-----"
```

다음 명령은가 퍼블릭 키를 AWS OpsWorks for Chef Automate 재생성할 수 있도록 하는 예입니다.

```
aws opsworks-cm update-server-engine-attributes \
   --server-name your-test-server \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value null
```

이 명령의 출력은 서버 및 base64로 인코딩된 ZIP 파일에 대한 정보입니다. ZIP 파일에는 README, 구성 파일 및 필수 RSA 프라이빗 키가 포함된 Chef 스타터 키트가 포함되어 있습니다. 이 파일을 저장하고 압축을 해제한 다음 파일 콘텐츠의 압축을 푼 디렉토리로 변경합니다. 이 디렉터리에서 `knife` 명령을 실행할 수 있습니다.

AWS OpsWorks for Chef Automate 리소스에서 태그 작업

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

태그는 AWS 리소스를 식별 및 구성하기 위한 메타데이터로 작동하는 단어나 구문입니다. 에서 리소스 AWS OpsWorks for Chef Automate는 최대 50개의 사용자 적용 태그를 가질 수 있습니다. 각 태그는 키와 하나의 값(선택 사항)으로 구성됩니다. AWS OpsWorks for Chef Automate에서 태그를 적용할 수 있는 리소스 유형은 다음과 같습니다.

- AWS OpsWorks for Chef Automate 서버
- AWS OpsWorks for Chef Automate 서버 백업

AWS 리소스에 태그를 지정하면 비용을 추적하고, 리소스에 대한 액세스를 제어하고, 작업 자동화를 위한 리소스를 그룹화하거나, 용도 또는 수명 주기 단계별로 리소스를 구성하는 데 도움이 될 수 있습니다. 태그의 이점에 대한 자세한 내용은 AWS Answers의 AWS 태깅 전략 및 AWS 결제 및 비용 관리 사용 설명서의 비용 할당 태그 사용을 참조하세요.

태그를 사용하여 AWS OpsWorks for Chef Automate 서버 또는 백업에 대한 액세스를 제어하려면 AWS Identity and Access Management (IAM)에서 정책 설명을 생성하거나 편집합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 리소스 태그를 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

AWS OpsWorks for Chef Automate 서버에 태그를 적용하면 서버의 백업, 백업을 저장하는 Amazon S3 버킷, 서버의 Amazon EC2 인스턴스, 저장된 서버의 보안 암호 AWS Secrets Manager, 서버에서 사용하는 탄력적 IP 주소에도 태그가 적용됩니다. 태그는가 서버를 생성하는 데 AWS OpsWorks 사용하는 AWS CloudFormation 스택에 전파되지 않습니다.

주제

- 에서 태그 작동 방식 AWS OpsWorks for Chef Automate
- 에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (콘솔)
- 에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (CLI)

- 참고

에서 태그 작동 방식 AWS OpsWorks for Chef Automate

이 릴리스에서는 AWS OpsWorks CM API 및 AWS Management Console을 사용하여 태그를 추가하고 관리할 수 있습니다. AWS OpsWorks 또한 CM은 EC2 인스턴스, Secrets Manager의 보안 암호, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업을 포함하여 서버에 연결된 AWS 리소스에 서버에 추가하는 태그를 추가하려고 시도합니다. 다음 표는 AWS OpsWorks for Chef Automate에서 태그를 추가하고 관리하는 방법의 개요를 제공합니다.

작업	사용 작업
수동으로 생성하는 새 AWS OpsWorks for Chef Automate 서버 또는 백업에 태그를 추가합니다.	• Chef Automate 서버 생성을 선택하고 고급 설정 구성 페이지에서 태그를 추가합니다. • 기존 서버의 백업 페이지에서 백업 생성을 선택하고 Chef Automate 2 서버의 백업 생성 페이지에서 태그를 추가하세요. • CreateServer 또는 CreateBackup 명령에 Tags 파라미터를 추가합니다.
리소스의 태그를 봅니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택합니다. • 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. • ListTagsForResource 명령을 실행합니다.
백업이 수동으로 생성되었는지 아니면 자동으로 생성되었는지에 관계없이 기존 AWS OpsWorks for Chef Automate 서버 또는 백업에 태그를 추가합니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택한 다음 편집을 선택합니다. • 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. • TagResource 명령을 실행합니다.
리소스에서 태그를 삭제합니다.	• 서버 세부 정보 페이지의 탐색 창에서 태그를 선택한 다음 편집을 선택합니다. 삭제할 태그 옆에 있는 X를 선택합니다.

작업	사용 작업
	• 서버의 백업 페이지에서 백업을 선택한 다음 백업 편집을 선택합니다. 삭제할 태그 옆에 있는 X를 선택합니다. • UntagResource 명령을 실행합니다.

DescribeServers 및 DescribeBackups 응답에는 태그 정보가 포함되지 않습니다. 태그를 표시하려면 ListTagsForResource API를 사용합니다.

에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (콘솔)

이 섹션의 절차는 AWS Management Console에서 수행됩니다.

태그를 추가할 경우 태그 키는 비워둘 수 없습니다. 키는 최대 127자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다. 태그 값은 선택 사항입니다. 키는 있지만 값은 없는 태그를 추가할 수 있습니다. 값은 최대 255자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다.

주제

- 새 AWS OpsWorks for Chef Automate 서버에 태그 추가(콘솔)
- 새 백업에 태그 추가(콘솔)
- 기존 서버에서 태그 추가 또는 보기(콘솔)
- 기존 백업에서 태그 추가 또는 보기(콘솔)
- 서버에서 태그 삭제(콘솔)
- 백업에서 태그 삭제(콘솔)

새 AWS OpsWorks for Chef Automate 서버에 태그 추가(콘솔)

1. AWS OpsWorks for Chef Automate 서버를 생성하기 위한 모든 사전 조건을 완료해야 합니다.
2. Chef Automate 서버 생성의 1-10단계를 따릅니다.
3. 자동 백업 설정을 지정한 후 고급 설정 구성 페이지의 태그 영역에 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다. 태그 추가가 완료되면 다음을 선택합니다.
4. Chef Automate 서버 생성의 13단계로 이동하여 새 서버에 대해 선택한 설정을 검토합니다.

새 백업에 태그 추가(콘솔)

1. AWS OpsWorks for Chef Automate 홈 페이지에서 기존 Chef Automate 서버를 선택합니다.
2. 서버 세부 정보 페이지의 탐색 창에서 백업을 선택합니다.
3. 백업 페이지에서 백업 생성을 선택합니다.
4. 태그를 추가합니다. 태그 추가가 완료되면 생성을 클릭합니다.

기존 서버에서 태그 추가 또는 보기(콘솔)

1. AWS OpsWorks for Chef Automate 홈 페이지에서 기존 Chef Automate 서버를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 태그를 선택하거나 세부 정보 페이지 하단에서 모든 태그 보기를 선택합니다.
3. 태그 페이지에서 편집을 선택합니다.
4. 서버에서 태그를 추가하거나 편집합니다. 작업을 마쳤으면 저장을 선택합니다.

> **Note**
>
> Chef Automate 서버에서 태그를 변경하면 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같이 서버와 연결된 리소스의 태그도 변경됩니다.

기존 백업에서 태그 추가 또는 보기(콘솔)

1. AWS OpsWorks for Chef Automate 홈 페이지에서 기존 Chef Automate 서버를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 백업을 선택하거나 세부 정보 페이지의 최근 백업 영역에서 모든 백업 보기를 선택합니다.
3. 백업 페이지에서 관리할 백업을 선택한 다음 백업 편집을 선택합니다.
4. 백업에서 태그를 추가하거나 편집합니다. 완료되면 업데이트를 선택합니다.

서버에서 태그 삭제(콘솔)

1. AWS OpsWorks for Chef Automate 홈 페이지에서 기존 Chef Automate 서버를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 태그를 선택하거나 세부 정보 페이지 하단에서 모든 태그 보기를 선택합니다.

3. 태그 페이지에서 편집을 선택합니다.
4. 태그 옆에 있는 X를 선택하여 태그를 삭제합니다. 작업을 마쳤으면 저장을 선택합니다.

> **Note**
>
> Chef Automate 서버에서 태그를 변경하면 EC2 인스턴스, 탄력적 IP 주소, 보안 그룹, S3 버킷 및 백업과 같이 서버와 연결된 리소스의 태그도 변경됩니다.

백업에서 태그 삭제(콘솔)

1. AWS OpsWorks for Chef Automate 홈 페이지에서 기존 Chef Automate 서버를 선택하여 세부 정보 페이지를 엽니다.
2. 탐색 창에서 백업을 선택하거나 세부 정보 페이지의 최근 백업 영역에서 모든 백업 보기를 선택합니다.
3. 백업 페이지에서 관리할 백업을 선택한 다음 백업 편집을 선택합니다.
4. 태그 옆에 있는 X를 선택하여 태그를 삭제합니다. 완료되면 업데이트를 선택합니다.

에서 태그 추가 및 관리 AWS OpsWorks for Chef Automate (CLI)

이 섹션의 절차는 AWS CLI에서 수행됩니다. 태그 작업을 시작하기 AWS CLI 전에의 최신 릴리스를 실행해야 합니다. 설치 또는 업데이트에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 설치를 AWS CLI AWS CLI참조하세요.

태그를 추가할 경우 태그 키는 비워둘 수 없습니다. 키는 최대 127자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다. 태그 값은 선택 사항입니다. 키는 있지만 값은 없는 태그를 추가할 수 있습니다. 값은 최대 255자로서 유니코드 문자, 숫자, 구분 기호 또는 특수 문자(+ - = . _ : / @)만 포함할 수 있습니다.

주제

- 새 AWS OpsWorks for Chef Automate 서버에 태그 추가(CLI)
- 새 백업에 태그 추가(CLI)
- 기존 서버 또는 백업에 태그 추가(CLI)
- 리소스 태그 나열
- 리소스에서 태그 삭제

새 AWS OpsWorks for Chef Automate 서버에 태그 추가(CLI)

AWS OpsWorks for Chef Automate 서버를 생성할 때 AWS CLI 를 사용하여 태그를 추가할 수 있습니다. 이 절차에서는 서버를 만드는 방법을 자세히 설명하지 않습니다. 이 설명서의 섹션을 사용하여 AWS OpsWorks for Chef Automate 서버를 생성하는 방법에 대한 자세한 내용은 섹션을 AWS CLI 참조를 사용하여 Chef Automate 서버 생성 AWS CLI하세요. 서버에 최대 50개의 태그를 추가할 수 있습니다.

1. AWS OpsWorks for Chef Automate 서버를 생성하기 위한 모든 사전 조건을 완료해야 합니다.
2. 를 사용하여 Chef Automate 서버 생성 AWS CLI의 1-5단계를 완료합니다.
3. 6단계에서 `create-server` 명령을 실행할 때, 다음 예제와 같이 명령에 `--tags` 파라미터를 추가합니다.

   ```
   aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
    Key=Key2,Value=Value2
   ```

   다음은 `create-server` 명령의 태그 부분만 보여주는 예제입니다.

   ```
   aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
    Key=Department,Value=Marketing
   ```

4. 를 사용하여 Chef Automate 서버 생성 AWS CLI에 설명된 나머지 단계를 완료합니다. 태그가 새 서버에 추가되었는지 확인하려면 이 주제의 리소스 태그 나열에 소개된 단계를 따릅니다.

새 백업에 태그 추가(CLI)

서버의 새 수동 백업 AWS CLI 을 생성할 때를 사용하여 태그를 추가할 수 있습니다 AWS OpsWorks for Chef Automate . 이 절차에서는 수동 백업을 만드는 방법을 자세히 설명하지 않습니다. 수동 백업을 생성하는 방법에 대한 자세한 내용은의 "에서 수동 백업을 수행하려면 AWS CLI"을 참조하세요AWS OpsWorks for Chef Automate 서버 백업. 백업에 최대 50개의 태그를 추가할 수 있습니다. 서버에 태그가 있는 경우 새 백업에 서버 태그가 자동으로 지정됩니다.

기본적으로 새 AWS OpsWorks for Chef Automate 서버를 생성하면 자동 백업이 활성화됩니다. 이 주제의 기존 서버 또는 백업에 태그 추가(CLI)에서 설명하는 `tag-resource` 명령을 실행하여 자동 백업에 태그를 추가할 수 있습니다.

- 백업을 만들 때 수동 백업에 태그를 추가하려면 다음 명령을 실행합니다. 명령의 태그 부분만 표시됩니다. 전체 `create-backup` 명령의 예는 AWS OpsWorks for Chef Automate 서버 백업의 "AWS CLI에서 수동 백업 수행하기"를 참조하세요.

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

다음은 `create-backup` 명령의 태그 부분만 보여주는 예제입니다.

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

기존 서버 또는 백업에 태그 추가(CLI)

`tag-resource` 명령을 실행하여 기존 AWS OpsWorks for Chef Automate 서버 또는 백업에 태그를 추가할 수 있습니다(백업이 자동으로 생성되었든 수동으로 생성되었든 상관없이). 대상 리소스의 Amazon 리소스 번호(ARN)를 지정하여 태그를 추가합니다.

1. 태그를 적용할 리소스의 ARN을 가져오려면 다음과 같이 하세요.

   - 서버의 경우 `describe-servers --server-name` *`server_name`*을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id` *`backup_ID`*를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. 를 실행`describe-backups --server-name` *`server_name`*하여 특정 AWS OpsWorks for Chef Automate 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.

   다음 예제는 `describe-servers --server-name opsworks-cm-test` 명령의 결과인 `ServerArn`만 보여줍니다. `ServerArn` 값이 `tag-resource` 명령에 추가되어 서버에 태그를 추가합니다.

```
{
    "Servers": [
        {
            ...
            "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
        }
    ]
```

```
}
```

2. 1단계에서 반환한 ARN을 사용하여 `tag-resource` 명령을 실행합니다.

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

다음은 예입니다.

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. 태그가 성공적으로 추가되었는지 확인하려면 다음 절차 리소스 태그 나열로 이동합니다.

리소스 태그 나열

`list-tags-for-resource` 명령을 실행하여 AWS OpsWorks for Chef Automate 서버 또는 백업에 연결된 태그를 표시할 수 있습니다. 해당 태그를 볼 대상 리소스의 ARN을 지정합니다.

1. 태그를 나열할 리소스의 ARN을 가져오려면 다음과 같이 하세요.

   - 서버의 경우 `describe-servers --server-name server_name`을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id backup_ID`를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. 를 실행`describe-backups --server-name server_name`하여 특정 AWS OpsWorks for Chef Automate 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.

2. 1단계에서 반환한 ARN을 사용하여 `list-tags-for-resource` 명령을 실행합니다.

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

다음은 예입니다.

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

리소스에 태그가 있는 경우 이 명령은 다음과 같은 결과를 반환합니다.

```
{
```

```
    "Tags": [
        {
            "Key": "Stage",
            "Value": "Production"
        },
        {
            "Key": "Department",
            "Value": "Marketing"
        }
    ]
}
```

리소스에서 태그 삭제

`untag-resource` 명령을 실행하여 AWS OpsWorks for Chef Automate 서버 또는 백업에서 태그를 삭제할 수 있습니다. 리소스가 삭제되면 리소스의 태그도 삭제됩니다. 대상 리소스의 Amazon 리소스 번호(ARN)를 지정하여 태그를 제거합니다.

1. 태그를 제거할 리소스의 ARN을 가져오려면 다음과 같이 하세요.

   - 서버의 경우 `describe-servers --server-name` *`server_name`*을 실행합니다. 명령의 결과에 서버 ARN이 표시됩니다.
   - 백업의 경우 `describe-backups --backup-id` *`backup_ID`*를 실행합니다. 명령 결과에 백업 ARN이 표시됩니다. 를 실행`describe-backups --server-name` *`server_name`*하여 특정 AWS OpsWorks for Chef Automate 서버의 모든 백업에 대한 정보를 표시할 수도 있습니다.

2. 1단계에서 반환한 ARN을 사용하여 `untag-resource` 명령을 실행합니다. 삭제할 태그만 지정합니다.

   ```
   aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   이 예에서 `untag-resource` 명령은 키가 `Stage`이고, 값이 `Production`인 태그만 제거합니다.

   ```
   aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production
   ```

3. 태그가 성공적으로 삭제되었는지 확인하려면 이 주제의 리소스 태그 나열에 소개된 단계를 따릅니다.

참고

- 를 사용하여 Chef Automate 서버 생성 AWS CLI
- AWS OpsWorks for Chef Automate 서버 백업
- AWS 태깅 전략
- AWS Identity and Access Management 사용 설명서의 AWS 리소스 태그를 사용하여 리소스에 대한 액세스 제어
- AWS 결제 및 비용 관리 사용 설명서의 비용 할당 태그 사용
- AWS OpsWorks CM API 참조에서 CreateBackup
- AWS OpsWorks CM API 참조에서 CreateServer
- AWS OpsWorks CM API 참조의 TagResource
- AWS OpsWorks CM API 참조의 ListTagsForResource
- AWS OpsWorks CM API 참조의 UntagResource

AWS OpsWorks for Chef Automate 서버 백업 및 복원

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 섹션에서는 AWS OpsWorks for Chef Automate 서버를 백업 및 복원하는 방법과 백업을 삭제하는 방법을 설명합니다.

주제

- AWS OpsWorks for Chef Automate 서버 백업
- 백업에서 AWS OpsWorks for Chef Automate 서버 복원

AWS OpsWorks for Chef Automate 서버 백업

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

일일 또는 주간 반복 AWS OpsWorks for Chef Automate 서버 백업을 정의하고 서비스가 사용자를 대신하여 Amazon Simple Storage Service(Amazon S3)에 백업을 저장하도록 할 수 있습니다. 또는 온디맨드로 수동 백업을 만들 수 있습니다.

백업은 Amazon S3에 저장되므로 추가 비용이 발생합니다. 최대 30개 생성까지 백업 보존 기간을 정의할 수 있습니다. AWS 지원 채널을 사용하여 해당 제한을 변경하도록 서비스 요청을 제출할 수 있습니다. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

AWS OpsWorks for Chef Automate 서버의 백업에 태그를 추가할 수 있습니다. AWS OpsWorks for Chef Automate 서버에 태그를 추가한 경우 서버의 자동 백업에 해당 태그가 상속됩니다. 백업에서 태그를 추가하고 관리하는 방법에 대한 자세한 내용은 이 안내서의 AWS OpsWorks for Chef Automate 리소스에서 태그 작업 단원을 참조하세요.

주제

- 백업 자동화
- 수동 백업
- 백업 삭제

백업 자동화

AWS OpsWorks for Chef Automate 서버를 구성할 때 자동 또는 수동 백업을 선택합니다.는 설정의 고급 설정 구성 페이지의 자동 백업 섹션에서 선택한 시간과 날짜에 자동 백업을 AWS OpsWorks for Chef Automate 시작합니다. 서버가 온라인 상태가 된 후 Chef Automate 서버 홈 페이지의 서버 타일 또는 서버의 Properties 페이지에서 다음 단계를 수행하여 백업 설정을 변경할 수 있습니다.

자동화된 백업 설정을 변경하려면

1. Chef 서버 홈 페이지의 서버 타일에 있는 작업 메뉴에서 설정 변경을 선택합니다.
2. 자동 백업을 끄려면 자동 백업 활성화 옵션에서 아니오를 선택합니다. 변경 사항을 저장합니다. 다음 단계로 진행할 필요는 없습니다.
3. [자동 백업] 섹션에서 빈도, 시작 시간 또는 유지할 세대를 변경합니다. 변경 내용을 저장합니다.

수동 백업

에서 언제든지 AWS Management Console또는 create-backup 명령을 실행하여 수동 백업을 AWS CLI 시작할 수 있습니다. 수동 백업은 저장되는 최대 30회 생성 자동 백업에 포함되지 않습니다. 수동 백업은 최대 10개 저장되며 Amazon S3에서 수동으로 삭제해야 합니다.

AWS OpsWorks for Chef Automate 서버의 수동 백업을 새로 생성할 때 태그를 추가할 수 있습니다. 수동 백업을 만들 때 태그를 추가하는 방법에 대한 자세한 내용은 새 백업에 태그 추가(CLI) 단원을 참조하세요.

에서 수동 백업을 수행하려면 AWS Management Console

1. [Chef Automate 서버] 페이지에서 백업하려는 서버를 선택합니다.
2. 서버의 속성 페이지의 왼쪽 탐색 창에서 [백업]을 선택합니다.
3. [백업 생성]을 선택합니다.
4. 화면에서 백업의 [상태] 열에서 녹색 확인 표시가 나타나면 수동 백업이 완료됩니다.

에서 수동 백업을 수행하려면 AWS CLI

- 수동 백업을 시작하려면 다음 AWS CLI 명령을 실행합니다.

```
aws opsworks-cm --region region name create-backup --server-name "Chef server name"
 --description "optional descriptive string"
```

백업 삭제

백업을 삭제하면 백업이 저장된 S3 버킷에서 영구적으로 삭제가 됩니다.

에서 백업을 삭제하려면 AWS Management Console

1. [Chef Automate 서버] 페이지에서 백업하려는 서버를 선택합니다.
2. 서버의 속성 페이지의 왼쪽 탐색 창에서 [백업]을 선택합니다.
3. 삭제하려는 백업을 선택한 다음 [백업 삭제]를 선택합니다. 한 번에 하나의 백업만 선택할 수 있습니다.
4. 삭제를 확인하는 메시지가 나타나면 [S3 버킷에 저장된 백업 삭제] 확인란을 선택한 다음 [예, 삭제]를 선택합니다.

에서 백업을 삭제하려면 AWS CLI

- 백업을 삭제하려면 다음 AWS CLI 명령을 실행하여를 삭제하려는 백업의 ID--backup-id로 바꿉니다. 백업 ID는 *ServerName-yyyyMMddHHmmssSSS* 형식입니다. 예를 들어 **test-chef-server-20171218132604388**입니다.

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

백업에서 AWS OpsWorks for Chef Automate 서버 복원

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

사용 가능한 백업을 탐색한 후 AWS OpsWorks for Chef Automate 서버를 복원할 시점을 선택할 수 있습니다. 서버 백업에는 구성-관리 소프트웨어 영구 데이터(쿡북, 등록된 노드 등)만 포함됩니다. 서버의 현재 위치 복원(즉, 기존 AWS OpsWorks for Chef Automate 서버를 새 EC2 인스턴스로 복원)을 수행하면 서버를 복원하는 데 사용하는 백업 시점에 등록된 노드가 다시 등록되고 복원에 성공하고 복원된 AWS OpsWorks for Chef Automate 서버 상태가 인 경우 트래픽이 새 인스턴스로 전환됩니다Healthy. 새로 만든 AWS OpsWorks for Chef Automate 서버로 복원해도 노드 연결이 유지되지 않습니다. 서버를 복원하면 Chef 소프트웨어 마이너 버전이 업데이트되지 않습니다. 선택한 백업에서 사용할 수 있는 것과 동일한 Chef 버전 및 구성 관리 데이터가 적용됩니다.

일반적으로 서버 복원은 새 서버를 만드는 것보다 시간이 더 걸리며, 시간은 선택한 백업 크기에 따라 달라집니다. 복원이 완료되면 이전 EC2 인스턴스는 Running 또는 Stopped 상태로 유지되지만 일시적입니다. 결국 종료됩니다.

이 릴리스에서는를 사용하여에서 Chef 서버를 복원 AWS CLI 할 수 있습니다 AWS OpsWorks for Chef Automate.

> **Note**
>
> restore-server 명령을 실행하여 현재 인스턴스 유형을 변경하거나 분실 또는 침해된 SSH 키를 복원 또는 설정할 수도 있습니다.

백업에서 서버를 복원하려면

1. 에서 다음 명령을 AWS CLI실행하여 사용 가능한 백업 목록과 해당 IDs. 사용할 백업 ID를 적어 둡니다. 백업 ID는 *myServerName-yyyyMMddHHmmssSSS* 형식입니다.

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 다음 명령을 실행합니다.

   ```
   aws opsworks-cm --region region name restore-server --backup-id "myServerName-
   yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
    key pair" --server-name "name of Chef server"
   ```

   다음은 예입니다.

   ```
   aws opsworks-cm --region us-west-2 restore-server --backup-id
    "MyChefServer-20161120122143125" --server-name "MyChefServer"
   ```

3. 복원이 완료될 때까지 기다리십시오.

의 시스템 유지 관리 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션

하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

필수 시스템 유지 관리는 보안 업데이트를 포함하여 Chef Server 및 Chef Automate Server의 최신 마이너 버전이 항상 AWS OpsWorks for Chef Automate 서버에서 실행되도록 합니다. 시스템 유지 관리는 일주일에 최소한 1번은 수행해야 합니다. 를 사용하여 원하는 경우 일일 자동 유지 관리를 구성할 AWS CLI수 있습니다. 또한 AWS CLI 를 사용하여 예약된 시스템 유지 관리 외에도 온디맨드로 시스템 유지 관리를 수행할 수 있습니다.

시스템 유지관리는 Chef 소프트웨어의 새 마이너 버전이 나오면 AWS 테스트를 통과하는 즉시 서버에서 Chef Automate 및 Chef Server의 마이너 버전을 자동으로 업데이트하도록 설계되어 있습니다. AWS가 광범위한 테스트를 실시하여 Chef 업그레이드가 생산 준비가 되었는지, 기존 고객 환경에 지장을 주지 않는지 확인하므로 Chef 소프트웨어 릴리스 후 Chef Automate 서버의 기존 OpsWorks에서 애플리케이션을 사용할 수 있게 되기까지 시간이 지연될 수 있습니다. 요구에 따라 사용 가능한 Chef 소프트웨어의 마이너 버전을 업데이트하는 방법은 이 주제의 요청 시 시스템 유지 관리 시작 단원을 참조하세요.

시스템 유지 관리는 유지 관리 프로세스의 일부로 수행된 백업에서 새 인스턴스를 시작합니다. 그러면 정기 유지 관리가 진행 중인 성능이 저하되었거나 손상된 Amazon EC2 인스턴스의 위험을 줄일 수 있습니다.

Important

시스템 유지 관리 시 AWS OpsWorks for Chef Automate 서버에 추가한 모든 파일 또는 사용자 지정 구성이 삭제됩니다. 구성 또는 파일 손실을 복구하는 자세한 방법은 이 주제의 유지 관리 후 사용자 지정 구성 및 파일 복원 단원을 참조하세요.

주제

- 노드가 AWS OpsWorks 인증 기관을 신뢰하도록 보장
- 시스템 유지 관리 구성
- 요청 시 시스템 유지 관리 시작
- 유지 관리 후 사용자 지정 구성 및 파일 복원

노드가 AWS OpsWorks 인증 기관을 신뢰하도록 보장

> **Note**
>
> AWS OpsWorks for Chef Automate 서버에서 사용자 지정 도메인 및 인증서를 사용하는 경우 이 섹션의 단계는 필요하지 않습니다.

AWS OpsWorks for Chef Automate 서버로 관리하는 노드는 인증서를 사용하여 서버로 인증해야 합니다. 시스템 유지 관리 중에는 서버 인스턴스를 AWS OpsWorks 교체하고 AWS OpsWorks 인증 기관(CA)을 통해 새 인증서를 다시 생성합니다. 유지 관리가 완료된 후 관리형 노드와의 통신을 자동으로 복원하려면 노드가 스타터 키트와 함께 제공되고에서 지원하는 리전에서 호스팅되는 AWS OpsWorks CA를 신뢰해야 합니다 AWS OpsWorks for Chef Automate. AWS OpsWorks CA를 사용하여 노드와 서버 간의 신뢰를 설정하면 유지 관리 후 노드가 새 서버 인스턴스에 다시 연결됩니다. 에 설명된 EC2 `userdata` 스크립트를 사용하여 EC2 노드를 추가하는 경우 에서 노드 자동 추가 AWS OpsWorks for Chef Automate노드는 AWS OpsWorks CA를 신뢰하도록 이미 구성되어 있습니다.

- Linux 기반 노드의 경우 CA의 S3 버킷 위치는 `https://opsworks-cm-${REGION}-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`입니다. 신뢰할 수 AWS OpsWorks 있는 CA는 경로에 저장되어야 합니다`/etc/chef/opsworks-cm-ca-2020-root.pem`.
- Windows 기반 노드의 경우 CA의 S3 버킷 위치는 `https://opsworks-cm-$env:AWS_REGION-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`입니다. AWS OpsWorks CA는 루트 Chef 폴더에 저장해야 합니다. 예를 들면 다음과 같습니다. `C:\chef\opsworks-cm-ca-2020-root.pem`

두 경로에서 리전 변수는 다음 중 하나로 해석됩니다.

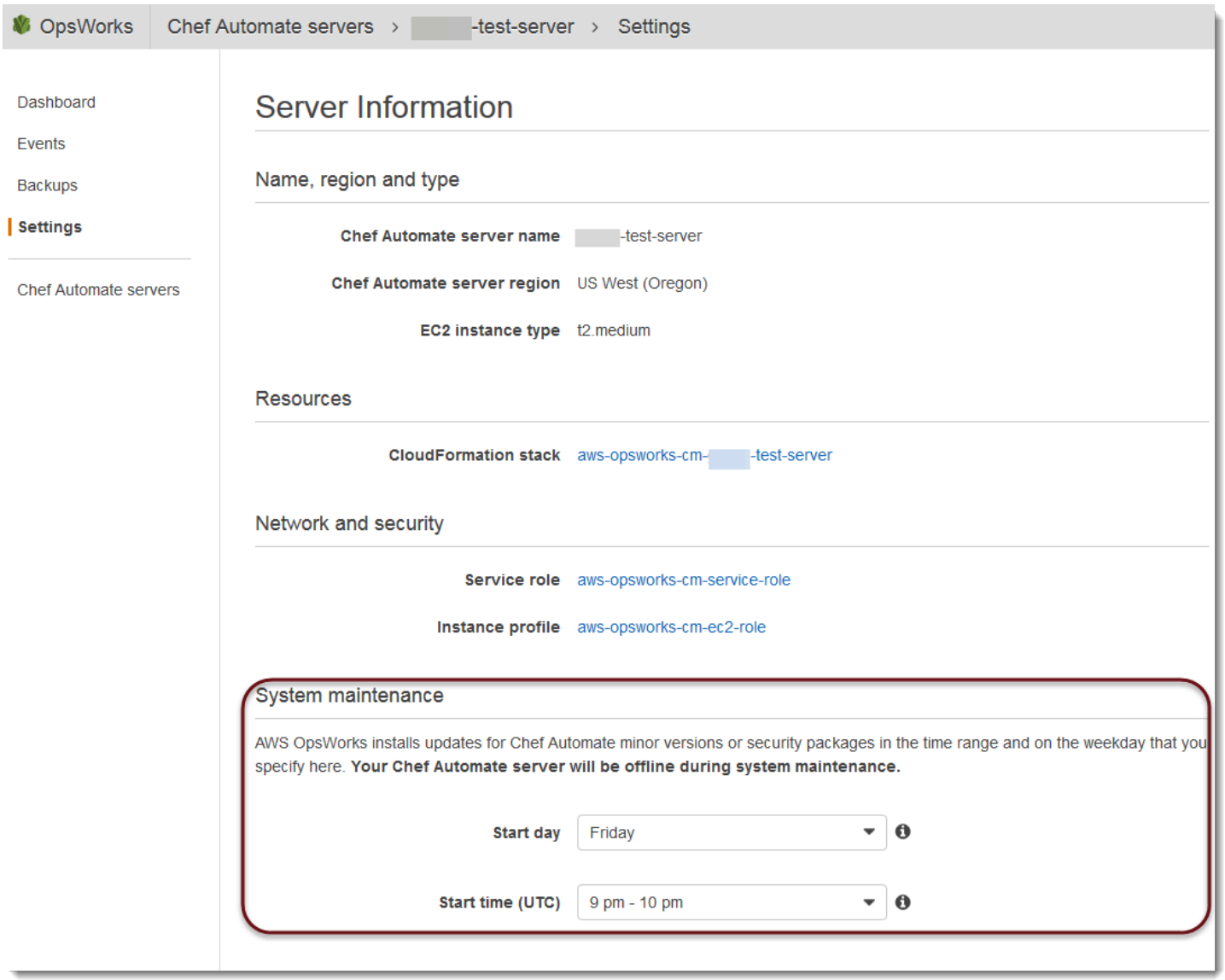
- `us-east-2`
- `us-east-1`
- `us-west-1`
- `us-west-2`
- `ap-northeast-1`
- `ap-southeast-1`
- `ap-southeast-2`

- eu-central-1
- eu-west-1

시스템 유지 관리 구성

새 AWS OpsWorks for Chef Automate 서버를 생성할 때 시스템 유지 관리가 시작되도록 협정 세계시(UTC)로 평일 및 시간을 구성할 수 있습니다. 지정한 시간 동안 유지 관리가 시작됩니다. 시스템 유지 관리 중에는 서버가 오프라인 상태여야 하므로 정규 업무 시간 중 서버에 대한 수요가 낮은 시간을 선택하세요. 유지 관리 진행 중 서버 상태는 UNDER_MAINTENANCE입니다.

다음 스크린샷과 같이 AWS OpsWorks for Chef Automate 서버 설정 페이지의 시스템 유지 관리 영역에서 설정을 변경하여 기존 서버의 시스템 유지 관리 설정을 변경할 수도 있습니다.



[시스템 유지 관리] 섹션에서 시스템 유지 관리를 시작하려는 날짜와 시간을 설정합니다.

를 사용하여 시스템 유지 관리 구성 AWS CLI

또한 AWS CLI를 사용하여 시스템 유지 관리 자동 시작 시간을 구성할 수도 있습니다. 를 AWS CLI 사용하면 원하는 경우 3자 평일 접두사를 생략하여 일일 자동 유지 관리를 구성할 수 있습니다.

서버 인스턴스 생성을 위한 요구 사항(예: 인스턴스 유형, 인스턴스 프로파일 ARN 및 서비스 역할 ARN)을 지정한 후`create-server` 명령에서 명령에 `--preferred-maintenance-window` 파라미터를 추가합니다. 다음 `create-server` 예에서는 `--preferred-maintenance-window`가 `Mon:08:00`으로 설정되어 있습니다. 즉, 매주 월요일 8:00 a.m. UTC에 유지보수가 시작되도록 설정했습니다. UTC 기준입니다.

```
aws opsworks-cm create-server --engine "Chef" --engine-model "Single" --
engine-version "12" --server-name "automate-06" --instance-profile-arn
 "arn:aws:iam::1019881987024:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "t2.medium" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

원하는 경우 `update-server` 명령에서는 `--preferred-maintenance-window` 값만 업데이트할 수 있습니다. 다음 예제에서 유지 관리 기간은 금요일 밤 6:15 p.m으로 설정되어 있습니다. UTC 기준입니다.

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "Fri:18:15"
```

유지 관리 기간의 시작 시간을 매일 세계협정시(UTC) 오후 6:15로 변경하려면 다음 예에서와 같이, 세 자리의 요일 접두사를 생략합니다.

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "18:15"
```

를 사용하여 기본 시스템 유지 관리 기간을 설정하는 방법에 대한 자세한 내용은 create-server 및 update-server를 AWS CLI참조하세요.

요청 시 시스템 유지 관리 시작

온디맨드 시스템 유지 관리를 시작하려면 구성된 주간 또는 일일 자동 유지 관리 외에 다음 AWS CLI 명령을 실행합니다. AWS Management Console에서는 온디맨드 유지 관리를 시작할 수 없습니다.

```
aws opsworks-cm start-maintenance --server-name server_name
```

이 명령에 대한 자세한 내용은 start-maintenance를 참조하세요.

유지 관리 후 사용자 지정 구성 및 파일 복원

시스템 유지 관리는 AWS OpsWorks for Chef Automate 서버에 추가한 사용자 지정 파일 또는 구성을 삭제하거나 변경할 수 있습니다.

RunCommand 또는 SSH를 사용하여 사용자가 추가한 파일 또는 설정이 유지 관리 실행 후 Chef 서버에서 누락된 경우 Amazon Machine Image(AMI)를 사용하여 새 Amazon EC2 인스턴스를 시작할 수 있습니다. 서버의 사전 유지 관리 구성에서 빌드된 AMI를 사용할 수 있습니다.

새 인스턴스는 유지 관리 이전의 Chef 서버 상태와 동일하므로 누락된 파일 및 설정이 포함되어 있어야 합니다.

> **Important**
>
> 새 인스턴스는 서버를 복원하는 데 사용할 수 없고 Chef 서버로 실행할 수 없습니다. 이 인스턴스는 파일 및 구성 설정을 복원하는 데에만 사용할 수 있습니다.

Amazon EC2 콘솔에서 AMI의 EC2 인스턴스를 시작하려면 시작 마법사를 열고 내 AMI를 선택한 후 서버와 이름이 같은 AMI를 선택합니다. 다른 인스턴스를 시작할 때와 마찬가지로 Amazon EC2 마법사의 단계에 따라 진행합니다.

의 규정 준수 스캔 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

규정 준수 검사를 사용하면 규칙이라고도 하는 사전 정의된 정책을 기반으로 인프라에서 관리형 노드의 규정 준수를 추적할 수 있습니다. 규정 준수 보기를 통해 애플리케이션의 취약성 및 규칙 미준수 구

성을 정기적으로 감사할 수 있습니다. Chef는 규정 준수 검사에서 사용할 수 있는 100개 이상의 사전 정의된 규정 준수 프로필(특정 노드 구성에 적용되는 규칙 모음)을 제공합니다. Chef InSpec 언어를 사용하여 고유의 사용자 지정 프로필을 생성할 수도 있습니다.

서버가 Chef Automate 2.0을 아직 실행하지 않는 경우 Audit 쿡북을 설치하여 Chef Compliance를 수동으로 설정할 수 있습니다.

> **Note**
>
> AWS OpsWorks for Chef Automate 서버와 연결된 노드에서 지원되는 Chef Infra 클라이언트 에이전트 소프트웨어(`chef-client`)의 최소 버전은 13.x입니다. 최신의 안정적인 `chef-client` 버전 또는 최소 14.10.9을 실행하는 것이 좋습니다.

주제

- Chef Automate 2.0의 규정 준수
- Chef Automate 1.x의 규정 준수
- 규정 준수에 대한 업데이트
- 커뮤니티 및 사용자 지정 규정 준수 프로필
- 참고

Chef Automate 2.0의 규정 준수

AWS OpsWorks for Chef Automate 서버가 Chef Automate 2.0을 실행하는 경우이 섹션의 절차를 사용하여 Chef Compliance를 설정합니다.

Chef Automate 2.0을 사용하여 규정 준수 검사 작업 실행

Chef Automate 2.0에는 이전에 수동 설정과 쿡북 구성이 필요했던 Chef InSpec 규정 준수 검사 기능이 포함되어 있습니다. Chef Automate 2.0을 실행하는 AWS OpsWorks for Chef Automate 서버에서 스캔 작업을 실행할 수 있습니다. 작업을 즉시(한 번) 실행하거나, 나중에 실행하도록 예약하거나, 매일 또는 두 시간에 한 번과 같은 지정된 간격으로 실행하도록 예약할 수 있습니다. 검사 작업의 결과는 규정 준수 보고에 전송됩니다. Chef Automate 대시보드에서 규정 준수 검사 결과를 보고 조치를 취할 수 있습니다. 규정 준수 탭을 열고 보고서를 보려면 Chef Automate 대시보드의 검사 작업 탭에서 관리형 노드 열의 오른쪽에 있는 보고서를 선택합니다.

관리형 노드에서 검사 작업을 실행하려면 다음이 있어야 합니다.

- 네임스페이스에 설치된 최소 하나의 규정 준수 프로필.
- 수동으로 추가된 최소 하나의 대상 노드, 또는 자동으로 추가된 EC2 인스턴스.

에서 AWS OpsWorks for Chef Automate스캔 작업은 다음 대상에서 지원됩니다.

- 수동으로 추가된 노드
- `aws-ec2` 인스턴스
- AWS 리전

검사 작업을 실행하는 방법에 대한 자세한 내용은 Chef 설명서의 Chef Automate 검사 작업을 참조하세요.

(선택 사항, Chef Automate 2.0) Audit 쿡북을 사용하여 규정 준수 설정

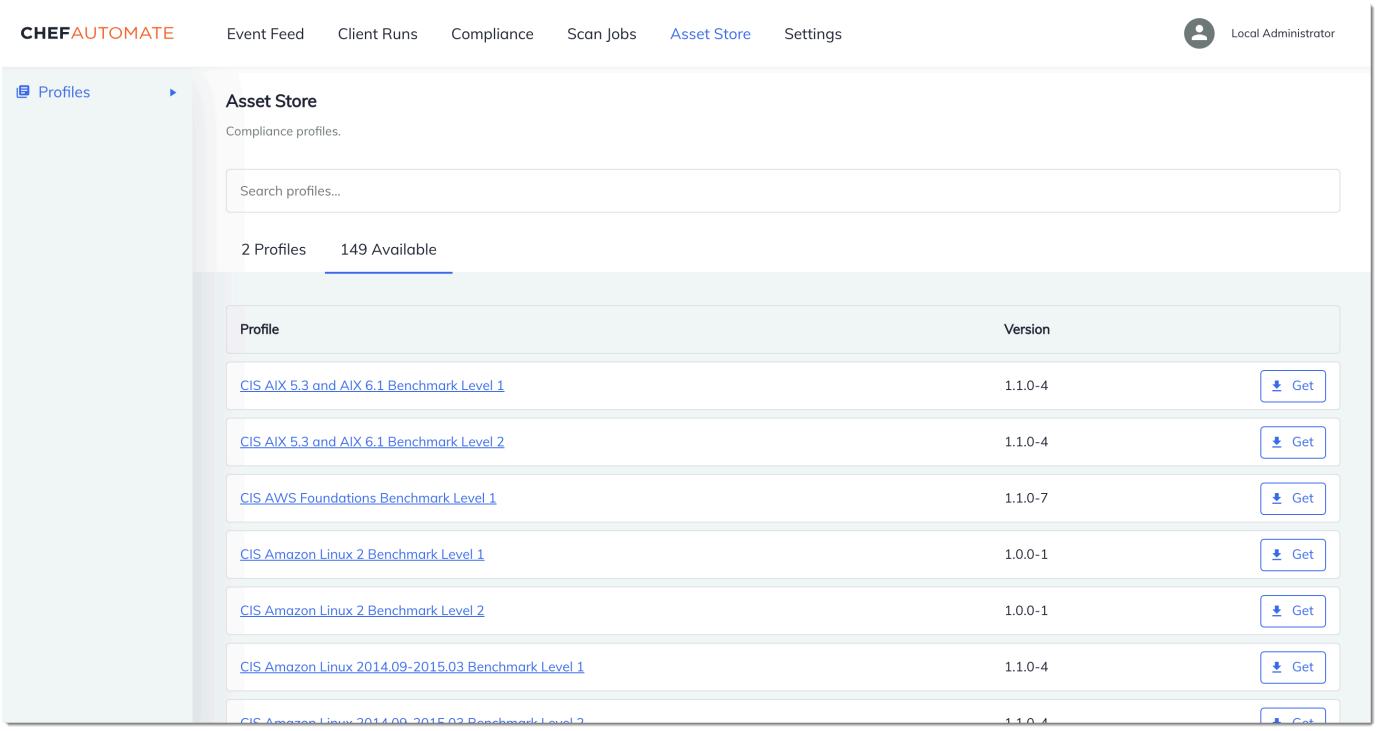
모든 서버에서 규정 준수를 구성할 수 AWS OpsWorks for Chef Automate 있습니다. AWS OpsWorks for Chef Automate 서버를 시작한 후, Chef Automate 대시보드에서 프로필을 설치하거나 `Policyfile.rb` 정책 파일에서 Audit 쿡북 속성에 원하는 프로필을 추가합니다. 미리 채워진 `Policyfile.rb` 파일이 스타터 키트에 포함되어 있습니다.

프로파일을 감사 쿡북의 속성으로 사용하여 `Policyfile.rb`을 편집한 후에는 `chef push` 명령을 실행하여 `Policyfile.rb`에 지정된 감사 쿡북 및 기타 쿡북을 Chef Automate 서버에 업로드합니다. Audit 쿡북을 설치하면 Chef에서 생성된 오픈 소스 테스트 및 감사 프레임워크인 Chef InSpec용 Gem도 설치됩니다. Chef Automate 2.0의 경우 Audit 쿡북의 버전 7.1.0 이상을 선택합니다. InSpec Gem은 버전 2.2.102 이상이어야 합니다.

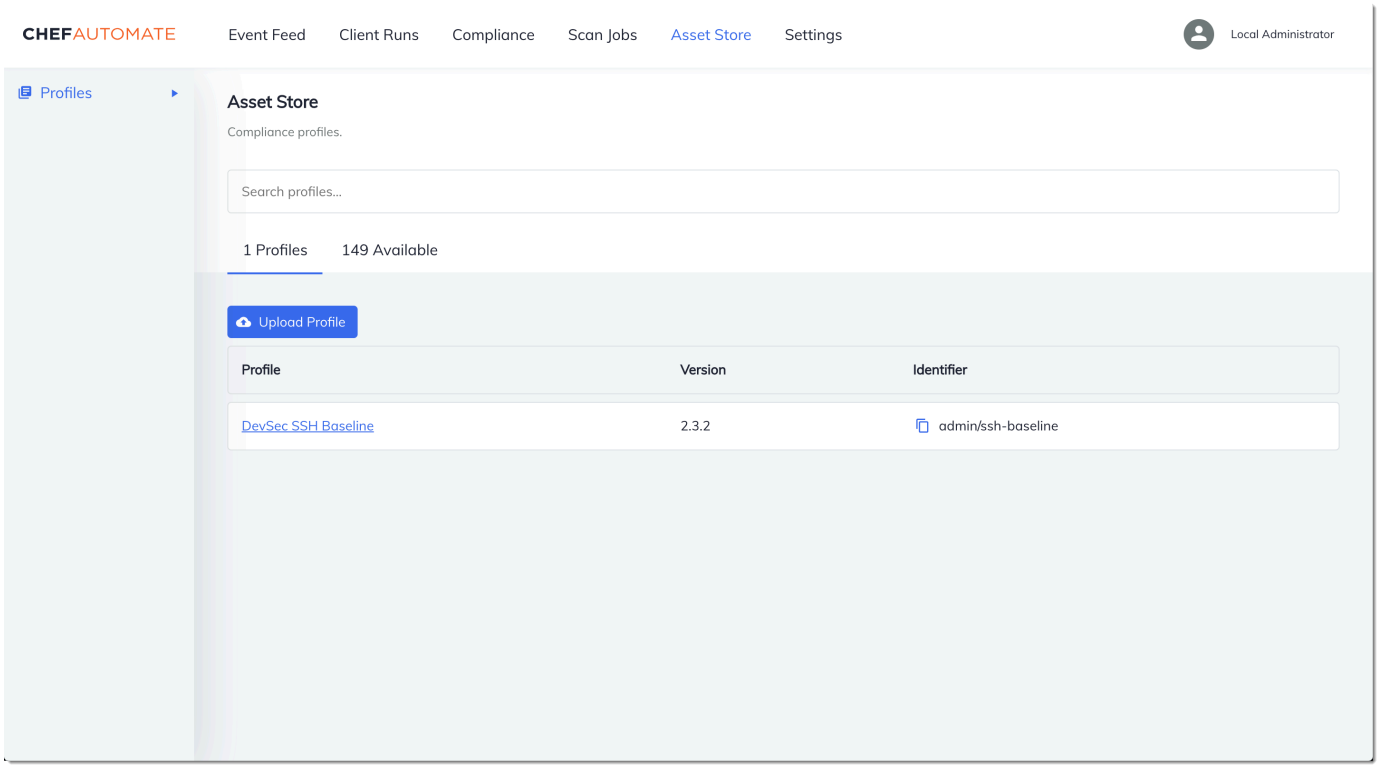
이 단원에서는 `opsworks-audit` 쿡북을 사용하는 방법을 설명합니다. Audit 쿡북은 Chef Automate 서버에서 지정된 프로필을 다운로드하고, DevSec SSH Baseline 프로필에 대해 노드를 평가하며, 모든 `chef-client` 실행에 대해 규정 준수 검사의 결과를 보고합니다.

규정 준수 프로필을 설치하려면

1. 아직 시작하지 않은 경우 Chef Automate 웹 기반 대시보드에 로그인합니다. AWS OpsWorks for Chef Automate 서버를 생성하면서 스타터 키트를 다운로드할 때 받은 자격 증명을 사용합니다.
2. Chef Automate 대시보드에서 자산 스토어 탭을 선택합니다.



3. 사용 가능 탭을 선택하여 사전 정의된 프로필을 봅니다.
4. 프로필 목록을 찾아봅니다. 하나 이상의 관리형 노드의 운영 체제 및 구성과 일치하는 프로필을 선택합니다. 프로필의 대상 위반 및 기본 규칙 코드에 대한 설명을 포함하여 프로필에 대한 세부 정보를 보려면 프로필 항목의 오른쪽에 있는 [>]를 선택합니다. 여러 프로필을 선택할 수 있습니다. 스타터 키트에서 예제를 설정하고 있는 경우 DevSec SSH Baseline을 선택합니다.



5. Chef Automate 서버에 선택한 프로필을 설치하려면 [가져오기]를 선택합니다.
6. 프로필을 설치한 후에는 Chef Automate 대시보드의 프로필 탭에 프로필이 표시됩니다.

`Policyfile.rb`를 사용하여 쿡북을 설치하려면

1. 스타터 키트에서 `Policyfile.rb`를 보고 Audit 쿡북에 대한 속성이 `['profiles']`에서 `ssh-baseline` 프로필을 지정하는지 확인합니다.

```
# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

2. `Policyfile.rb`에 정의된 쿡북을 다운로드하고 설치합니다.

```
chef install
```

모든 쿡북은 쿡북의 `metadata.rb` 파일에 버전이 지정되어 있습니다. 쿡북을 변경할 때마다 쿡북의 `metadata.rb`에 있는 쿡북 버전을 올려야 합니다.

3. `Policyfile.rb`에 정의된 `opsworks-demo` 정책을 서버에 푸시합니다.

```
chef push opsworks-demo
```

4. 정책 설치를 확인합니다. 다음 명령을 실행합니다.

```
chef show-policy
```

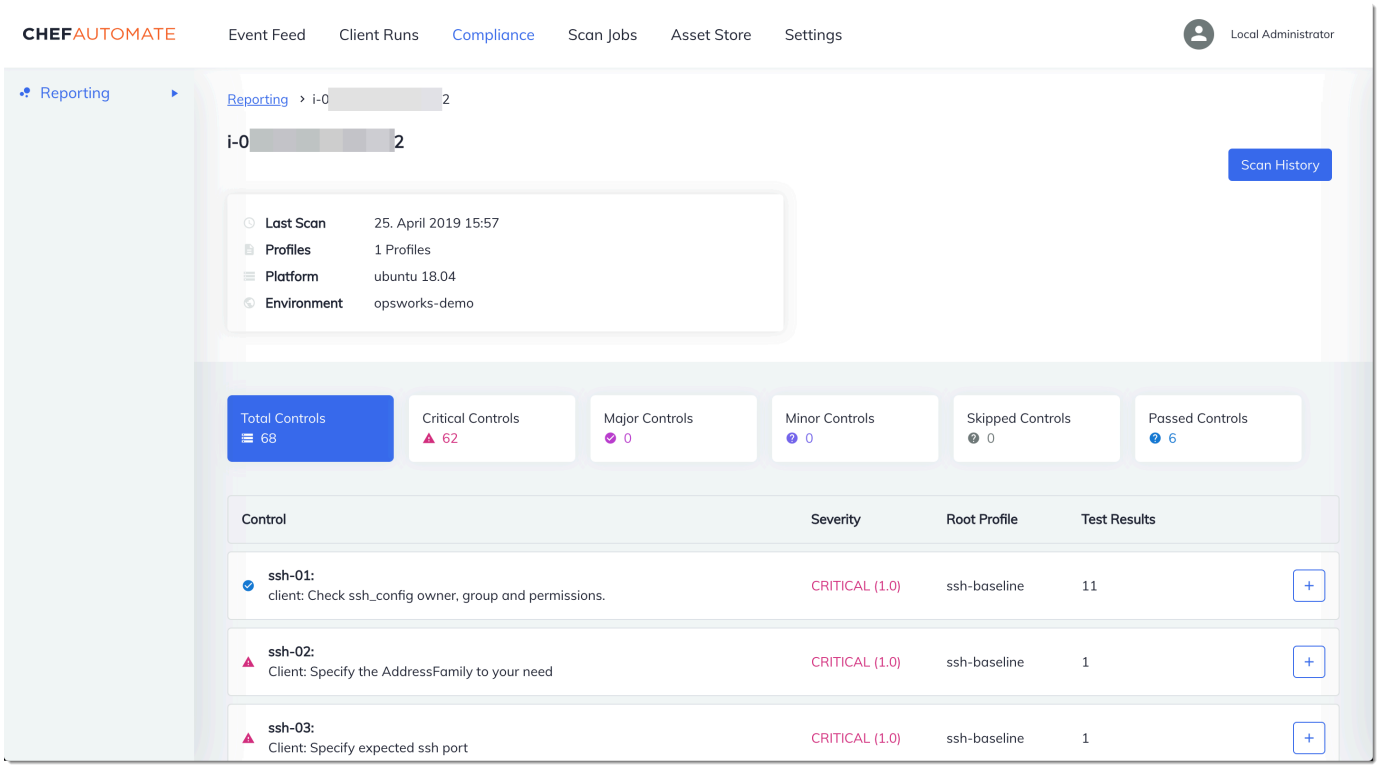
결과는 다음과 비슷해야 합니다.

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 관리할 서버에 노드를 추가하지 않았으면 추가합니다. 첫 번째 노드를 AWS OpsWorks for Chef Automate 서버에 연결하려면이 스타터 키트에 포함된 `userdata.sh` 스크립트를 사용합니다. `AssociateNode` API를 AWS OpsWorks 사용하여 노드를 서버에 연결합니다.

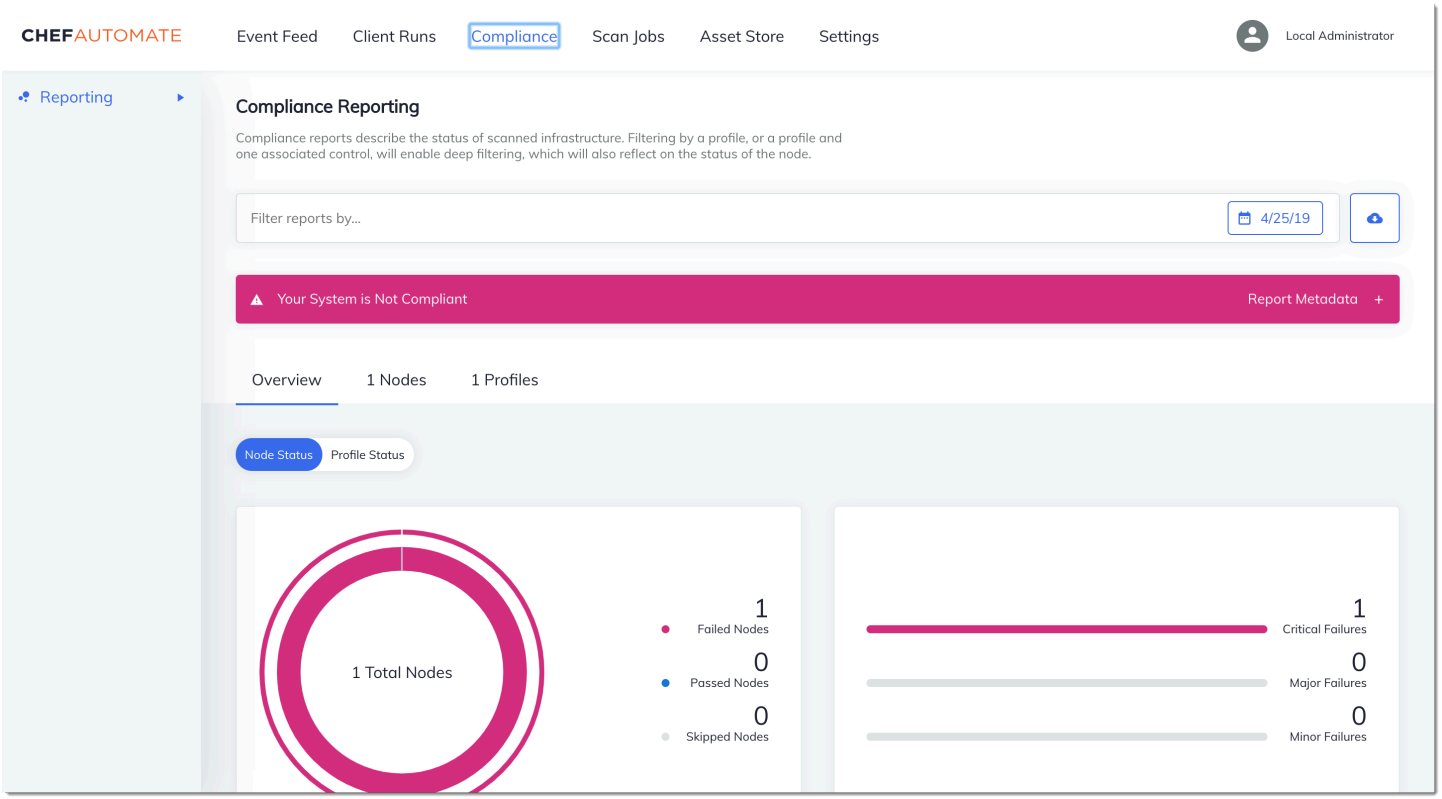
에서 노드 자동 추가 AWS OpsWorks for Chef Automate의 단계를 수행하여 노드 연결을 자동화하거나, 노드를 개별적으로 추가의 단계를 수행하여 한 번에 한 개의 노드를 추가할 수 있습니다.

6. 노드의 실행 목록을 업데이트하면 다음 실행 시 `chef-client` 에이전트가 지정된 레시피를 실행합니다. 이 작업은 기본적으로 1800초(30분)마다 수행됩니다. 실행 후 Chef Automate 대시보드의 규정 준수 탭에서 규정 준수 결과를 보고 조치를 취할 수 있습니다.

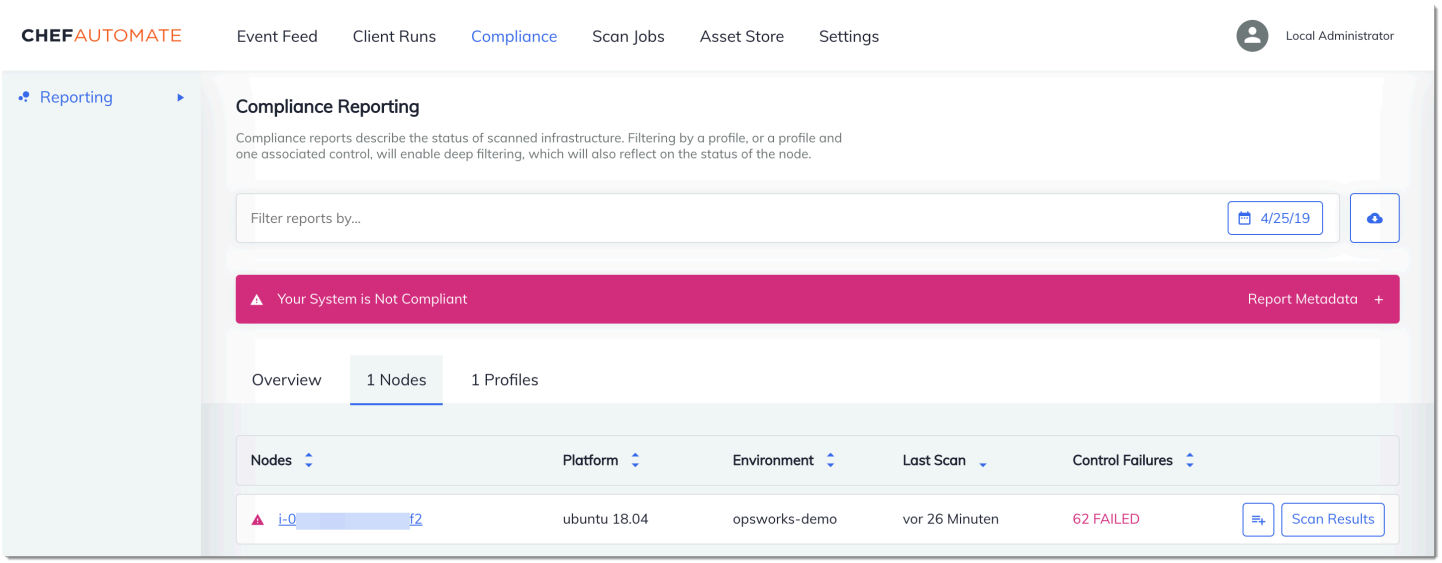


규정 준수 검사 실행

노드 실행 목록을 구성한 후 발생하는 에이전트가 처음 실행된 직후 Chef Automate 대시보드에 규정 준수 검사 결과가 표시됩니다.



Chef Automate 대시보드에서 [규정 준수] 탭을 선택합니다. 왼쪽 탐색 창에서 [보고]를 선택합니다. [프로필] 탭을 선택하고 [결과 스캔]을 선택한 다음 스캔 장애가 있는 노드를 선택하여 노드가 실패한 규칙에 대해 자세히 알아봅니다.



새 노드는 아직 DevSec SSH Baseline 프로파일의 모든 규칙을 충족하지는 못하기 때문에 일반적으로 규정 미준수 검사 결과가 표시됩니다. 커뮤니티 기반 프로젝트인 DevSec Hardening Framework는 DevSec SSH Baseline 프로파일의 규칙을 위반하는 문제를 수정하는 쿡북을 제공합니다.

(선택 사항) 규정 미준수 결과 해결

스타터 키트에는 오픈 소스 쿡북 `ssh-hardening`이 들어 있으며, 이 쿡북을 실행하여 DevSec SSH Baseline 프로파일에 대한 규정 미준수 결과를 수정할 수 있습니다.

> **Note**
>
> `ssh-hardening` 쿡북은 DevSec SSH Baseline 규칙을 준수하도록 노드를 변경합니다. 프로덕션 노드에서 이 쿡북을 실행하기 전에 Chef Automate 콘솔에서 DevSec SSH Baseline 프로파일에 대한 정보를 검토하여 쿡북이 검사하는 규칙 위반을 확인하세요. 오픈 소스 `ssh-hardening` 쿡북을 프로덕션 노드에서 실행하기 전에 쿡북에 대한 정보를 검토하세요.

`ssh-hardening` 쿡북을 실행하려면

1. 텍스트 편집기에서 `ssh-hardening` 쿡북을 `Policyfile.rb`의 실행 목록에 추가합니다. `Policyfile.rb` 실행 목록은 다음과 일치해야 합니다.

   ```
   run_list  'chef-client', 'opsworks-webserver', 'audit', 'ssh-hardening'
   ```

2. `Policyfile.rb`를 업데이트하고 AWS OpsWorks for Chef Automate 서버에 푸시합니다.

   ```
   chef update Policyfile.rb
      chef push opsworks-demo
   ```

3. `opsworks-demo` 정책과 관련된 노드가 실행 목록을 자동으로 업데이트하고 다음 번 `chef-client` 실행 시 `ssh-hardening` 쿡북을 적용합니다.

   `chef-client` 쿡북을 사용하기 때문에 노드는 정기적으로(기본 30분 간격) 체크인됩니다. 다음 체크인에서 `ssh-hardening` 쿡북이 실행되고 DevSec SSH Baseline 프로파일 규칙을 충족하도록 노드 보안이 강화됩니다.

4. `ssh-hardening` 쿡북을 처음 실행한 후 규정 준수 검사가 다시 실행될 때까지 30분 정도 기다립니다. Chef Automate 대시보드에서 결과를 확인합니다. DevSec SSH Baseline 검사를 처음 실행할 때 발생한 규정 미준수 결과는 해결해야 합니다.

Chef Automate 1.x의 규정 준수

AWS OpsWorks for Chef Automate 서버가 Chef Automate 1.x를 실행하는 경우 이 섹션의 절차를 사용하여 Chef 규정 준수를 설정합니다.

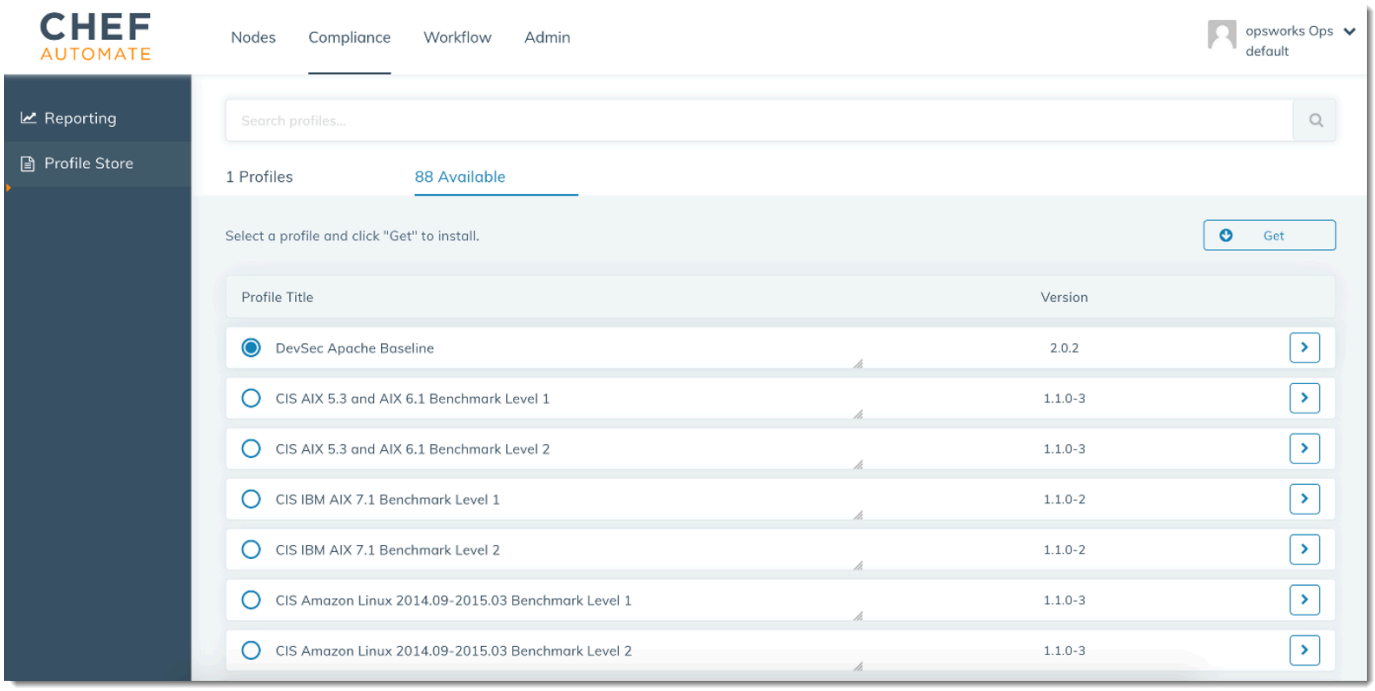
(선택 사항, Chef Automate 1.x) Chef Compliance 설정

모든 AWS OpsWorks for Chef Automate 서버에서 Chef Compliance를 구성할 수 있습니다. AWS OpsWorks for Chef Automate 서버를 시작한 후 Chef Automate 대시보드의 프로필에서 실행할 프로필을 선택합니다. 프로필을 설치한 후 `berks` 명령을 실행하여 Audit 쿡북을 Chef Automate 서버에 업로드합니다. Audit 쿡북을 설치하면 Chef가 제작한 오픈 소스 테스트 프레임워크인 InSpec에 대한 Gem도 설치되므로, 자동화된 테스트를 배포 파이프라인의 모든 단계에 통합할 수 있습니다. Chef Automate 1.x의 경우 Audit 쿡북의 버전 5.0.1 이상을 선택합니다. InSpec Gem은 버전 1.24.0 이상이어야 합니다.

AWS OpsWorks for Chef Automate 스타터 키트에는 적절한 버전의 Chef's Audit 쿡북을 다운로드하여 설치하는 래퍼 쿡북 `opsworks-audit`가 포함되어 있습니다. `opsworks-audit` 쿡북은 `chef-client` 에이전트에게 DevSec SSH Baseline 프로파일을 기준으로 노드를 평가하도록 지시합니다. 이 프로파일은 이 항목 뒷부분에 나오는 Chef's Compliance 콘솔에서 설치합니다. 쿡북을 사용하여 원하는 기본 설정으로 Compliance를 설정할 수 있습니다. 이 단원에서는 `opsworks-audit` 쿡북을 사용하는 방법을 설명합니다.

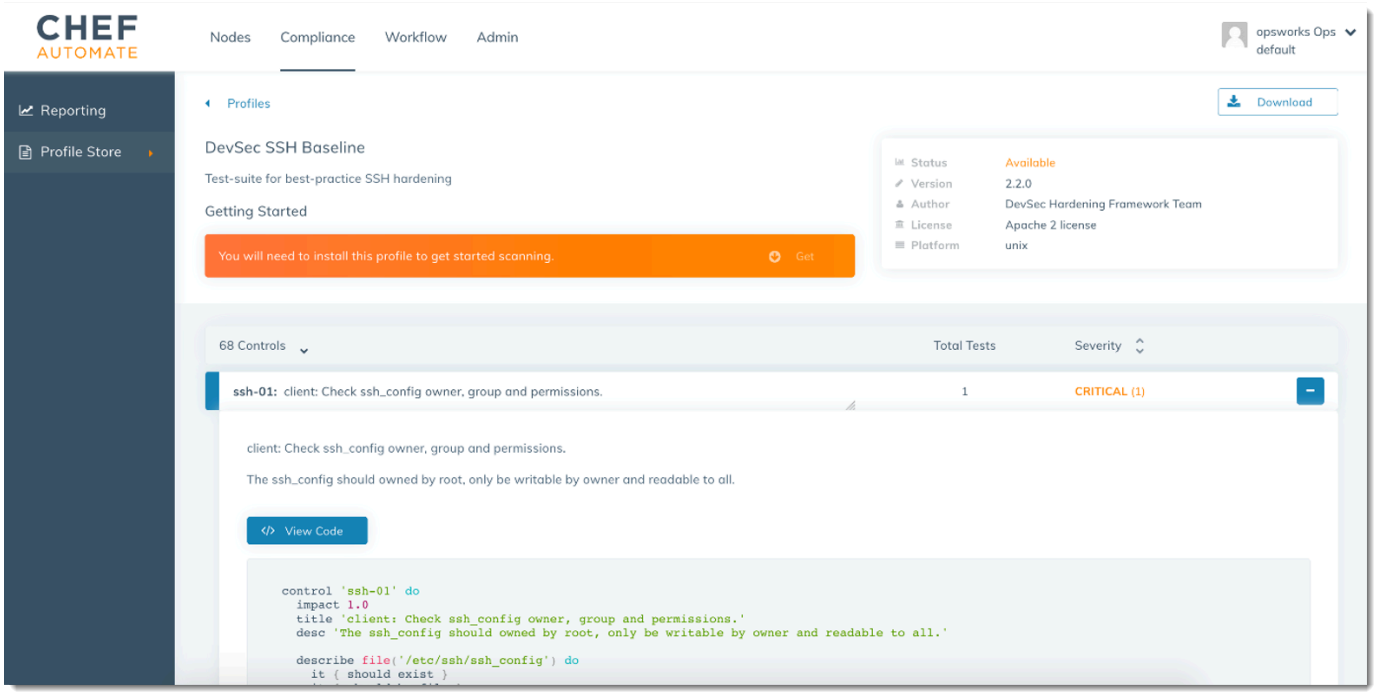
규정 준수 프로필을 설치하려면

1. 아직 시작하지 않은 경우 Chef Automate 웹 기반 대시보드에 로그인합니다. AWS OpsWorks for Chef Automate 서버를 생성할 때 스타터 키트를 다운로드할 때 받은 자격 증명을 사용합니다.
2. Chef Automate 대시보드에서 [규정 준수] 탭을 선택합니다.



3. 왼쪽 탐색 모음에서 [프로필 스토어]를 선택한 다음 [사용 가능]을 선택하여 사전 정의된 프로필을 확인합니다.

4. 프로필 목록을 찾아봅니다. 하나 이상의 관리형 노드의 운영 체제 및 구성과 일치하는 프로필을 선택합니다. 프로필의 대상 위반 및 기본 규칙 코드에 대한 설명을 포함하여 프로필에 대한 세부 정보를 보려면 프로필 항목의 오른쪽에 있는 [>]를 선택합니다. 여러 프로필을 선택할 수 있습니다.



5. Chef Automate 서버에 선택한 프로필을 설치하려면 [가져오기]를 선택합니다.

6. 다운로드가 완료되면 다음 절차로 이동합니다.

opsworks-audit 쿡북을 설치하고 설정하려면

1. 이 단계는 선택 사항이지만, 6단계에서 노드 실행 목록에 레시피를 추가할 때 시간을 절약할 수 있습니다. AWS OpsWorks for Chef Automate 서버를 생성할 때 다운로드한 스타터 키트에 포함된 `roles/opsworks-example-role.rb` 파일을 편집합니다. 다음 행을 추가합니다. `ssh-hardening` 쿡북과 레시피를 추가하여 규정 준수 검사 실행 후 규정 미준수 노드를 해결하는 것은 선택 사항이기 때문에 마지막 행은 주석 처리되어 있습니다.

```
run_list(
     "recipe[chef-client]",
     "recipe[apache2]",
     "recipe[opsworks-audit]"
     # "recipe[ssh-hardening]"
```

```
        )
```

2. 텍스트 편집기를 사용하여 원하는 쿡북을 Berksfile에 지정합니다. 스타터 키트에는 샘플 Berksfile이 제공됩니다. 이 예제에서는 Chef Infra 클라이언트(`chef-client`) 쿡북, `apache2` 쿡북, `opsworks-audit` 쿡북을 설치합니다. Berksfile은 다음과 유사합니다.

```
source 'https://supermarket.chef.io
     cookbook 'chef-client'
     cookbook 'apache2', '~> 5.0.1'
     cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0'
```

   모든 쿡북은 쿡북의 `metadata.rb` 파일에 버전이 지정되어 있습니다. 쿡북을 변경할 때마다 쿡북의 `metadata.rb`에 있는 쿡북 버전을 올려야 합니다.

3. 다음 명령을 실행하여 쿡북을 로컬 컴퓨터나 작업 컴퓨터의 `cookbooks` 폴더로 다운로드하고 설치합니다.

```
berks vendor cookbooks
```

4. 다음 명령을 실행하여 벤더 쿡북을 AWS OpsWorks for Chef Automate 서버로 업로드합니다.

```
knife upload .
```

5. 다음 명령을 실행하여 서버에서 현재 사용 가능한 쿡북 목록을 조회함으로써 `opsworks-audit` 쿡북의 설치를 확인합니다.

```
knife cookbook list
```

6. 관리할 서버에 노드를 추가하지 않았으면 추가합니다. 에서 노드 자동 추가 AWS OpsWorks for Chef Automate의 단계를 수행하여 노드 연결을 자동화하거나, 노드를 개별적으로 추가의 단계를 수행하여 한 번에 한 개의 노드를 추가할 수 있습니다. 노드의 실행 목록을 편집하여 1단계에서 지정한 `opsworks-example-role` 역할을 추가합니다. 이 예제에서는 `RUN_LIST` 스크립트에서 `userdata` 속성을 편집하여 노드 연결을 자동화합니다.

```
RUN_LIST="role[opsworks-example-role]"
```

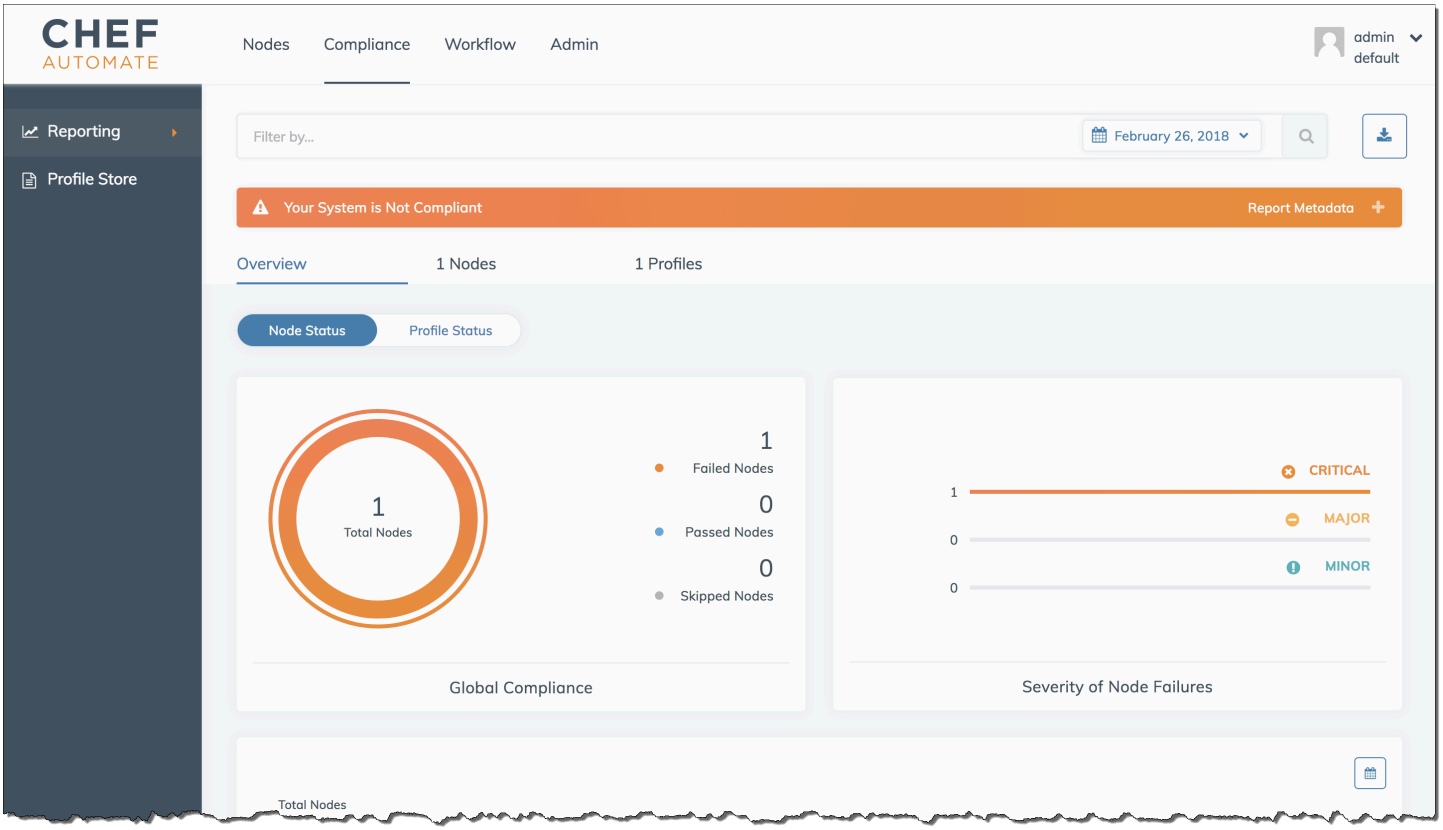
   1단계를 건너뛰고 역할을 설정하지 않은 경우 실행 목록에 개별 레시피 이름을 추가합니다. 변경 사항을 저장하고 3단계: 무인 연결 스크립트를 사용하여 인스턴스 생성의 단계를 따라 userdata 스크립트를 Amazon EC2 인스턴스에 적용합니다.

```
RUN_LIST="recipe[chef-client],recipe[apache2],recipe[opworks-audit]"
```

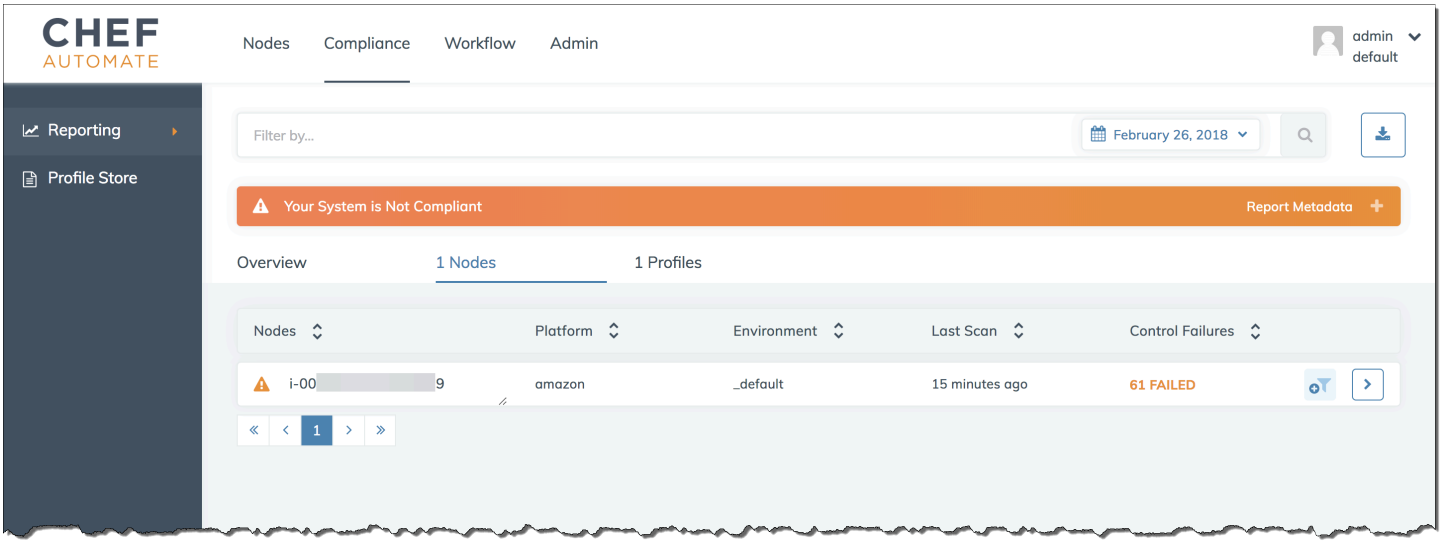
7. 노드의 실행 목록을 업데이트하면 다음 실행 시 `chef-client` 에이전트가 지정된 레시피를 실행합니다. 이 작업은 기본적으로 1800초(30분)마다 수행됩니다. 실행 후 Chef Automate 대시보드에 규정 준수 결과가 표시됩니다.

규정 준수 검사 실행

노드 실행 목록을 구성한 후 발생하는 에이전트 데몬이 처음 실행된 직후 Chef Automate 대시보드에서 규정 준수 검사 결과가 표시됩니다.



Chef Automate 대시보드에서 [규정 준수] 탭을 선택합니다. 왼쪽 탐색 창에서 [보고]를 선택합니다. [프로필] 탭을 선택하고 [스캔 결과]를 선택한 다음 스캔 장애가 있는 노드를 선택하여 노드가 실패한 규칙에 대해 자세히 알아봅니다.



새 노드는 아직 DevSec SSH Baseline 프로파일의 모든 규칙을 충족하지는 못하기 때문에 일반적으로 규정 미준수 검사 결과가 표시됩니다. 커뮤니티 기반 프로젝트인 DevSec Hardening Framework는 DevSec SSH Baseline 프로파일의 규칙을 위반하는 문제를 수정하는 쿡북을 제공합니다.

(선택 사항) 규정 미준수 결과 해결

스타터 키트에는 오픈 소스 쿡북 `ssh-hardening`이 들어 있으며, 이 쿡북을 실행하여 DevSec SSH Baseline 프로파일에 대한 규정 미준수 결과를 수정할 수 있습니다.

> **Note**
>
> `ssh-hardening` 쿡북은 DevSec SSH Baseline 규칙을 준수하도록 노드를 변경합니다. 프로덕션 노드에서 이 쿡북을 실행하기 전에 Chef Automate 콘솔에서 DevSec SSH Baseline 프로파일에 대한 정보를 검토하여 쿡북이 검사하는 규칙 위반을 확인하세요. 오픈 소스 `ssh-hardening` 쿡북을 프로덕션 노드에서 실행하기 전에 쿡북에 대한 정보를 검토하세요.

`ssh-hardening` 쿡북을 실행하려면

1. 텍스트 편집기에서 `ssh-hardening` 쿡북을 Berksfile에 추가합니다. Berksfile은 다음과 유사합니다.

```
source 'https://supermarket.chef.io'
     cookbook 'chef-client'
     cookbook 'apache2', '~> 5.0.1'
     cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0' #
 optional
```

```
    cookbook 'ssh-hardening'
```

2. 다음 명령을 실행하여 `ssh-hardening` 쿡북을 로컬 쿡북 폴더로 다운로드한 후 AWS OpsWorks for Chef Automate 서버에 업로드합니다.

```
berks vendor cookbooks
knife upload .
```

3. `ssh-hardening`의 1단계와 6단계에서 설명한 대로 [opsworks-audit 쿡북을 설치하고 설정하려면] 레시피를 노드 실행 목록에 추가합니다.

   `opsworks-example-role.rb` 파일을 업데이트하는 경우 다음 명령을 실행하여 변경 사항을 서버에 업로드합니다.

```
knife upload .
```

   실행 목록을 직접 업데이트하는 경우 다음 명령을 실행하여 변경 사항을 업로드합니다. 노드 이름은 일반적으로 인스턴스 ID입니다.

```
knife node run_list add <node name> 'recipe[ssh-hardening]'
```

4. `chef-client` 쿡북을 사용하기 때문에 노드는 정기적으로(기본 30분 간격) 체크인됩니다. 다음 체크인에서 `ssh-hardening` 쿡북이 실행되고 DevSec SSH Baseline 프로파일 규칙을 충족하도록 노드 보안이 강화됩니다.

5. `ssh-hardening` 쿡북을 처음 실행한 경우 규정 준수 검사 실행을 위해 30분 기다리십시오. Chef Automate 대시보드에서 결과를 확인합니다. DevSec SSH Baseline 검사를 처음 실행할 때 발생한 규정 미준수 결과는 해결해야 합니다.

규정 준수에 대한 업데이트

AWS OpsWorks for Chef Automate 서버에서 규정 준수 기능은 예약된 [시스템 유지 관리에] 의해 자동으로 업데이트됩니다. Chef Automate 및 Chef Infra Server의 업데이트된 릴리스가 AWS OpsWorks for Chef Automate 서버에서 사용 가능하게 되면, 서버에서 실행 중인 Audit 쿡북 및 Chef InSpec Gem의 지원 버전을 확인하고 업데이트해야 할 수 있습니다. AWS OpsWorks for Chef Automate 서버에 이미 설치한 프로필은 유지 관리의 일부로 업데이트되지 않습니다.

커뮤니티 및 사용자 지정 규정 준수 프로필

Chef에는 현재 100개 이상의 규정 준수 검사 프로필이 포함되어 있습니다. 커뮤니티 및 사용자 지정 프로필을 목록에 추가한 다음, 포함된 프로필과 마찬가지로 해당 프로필을 기반으로 규정 준수 검사를 다운로드하고 실행할 수 있습니다. 커뮤니티 기반 규정 준수 프로필은 Chef Supermarket에서 사용할 수 있습니다. 사용자 지정 프로필은 검사 규칙을 지정하는 컨트롤 폴더가 포함된 Ruby 기반 프로그램입니다.

참고

- Chef Compliance 공지사항 블로그 게시물
- Chef Automate Compliance 온라인 교육
- Chef InSpec 웹 사이트
- Chef InSpec 자습서

AWS OpsWorks for Chef Automate 서버에서 노드 연결 해제

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 섹션에서는 AWS OpsWorks for Chef Automate 서버의 관리에서 관리형 노드의 연결을 해제하거나 제거하는 방법을 설명합니다. 이 작업은 명령줄에서 수행되므로 AWS OpsWorks for Chef Automate 관리 콘솔에서 노드 연결을 해제할 수 없습니다. 현재 AWS OpsWorks for Chef Automate API는 여러 노드의 배치 제거를 허용하지 않습니다. 이 섹션에서 사용하는 명령은 노드를 한 번에 하나씩 연결 해제합니다.

Chef 서버 삭제 전에 서버에서 노드 연결을 해제하는 것이 좋습니다. 그래야 노드가 서버에 재연결을 시도하지 않고 계속 작동합니다. 이렇게 하려면 `disassociate-node` AWS CLI 명령을 실행합니다.

노드 연결 해제

1. 에서 다음 명령을 AWS CLI실행하여 노드 연결을 해제합니다. *Node_name*은 연결 해제할 노드의 이름으로, Amazon EC2 인스턴스의 경우 이 값은 인스턴스 ID입니다. *Server_name*은 노드를 연결 해제할 Chef 서버의 이름입니다. `--engine-attributes`는 기본 `CHEF_AUTOMATE_ORGANIZATION` 이름을 지정합니다. 이러한 세 개 파라미터가 모두 필요합니다.

   기본 리전에 없는 Chef 서버에서 노드를 연결 해제하지 않으려는 경우에는 `--region` 파라미터가 필요 없습니다.

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-
   name Node_name --server-name Server_name --engine-attributes
    "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
   ```

   다음 명령은 예제입니다.

   ```
   aws opsworks-cm --region us-west-2 disassociate-node --node-name
    i-0010zzz00d66zzz90 --server-name opsworkstest --engine-attributes
    "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
   ```

2. 연결 해제가 완료되었다는 응답 메시지가 표시될 때까지 기다리십시오.

   AWS OpsWorks for Chef Automate 서버에서 노드를 성공적으로 연결 해제한 후에도 Chef Automate 대시보드에 계속 표시될 수 있습니다. 기본적으로 Chef는 노드 상태 정보에 대한 보존 기간을 적용하므로 며칠 후 노드를 자동으로 삭제합니다.

AWS OpsWorks for Chef Automate 서버를 삭제하는 방법에 대한 자세한 내용은 섹션을 참조하세요AWS OpsWorks for Chef Automate 서버 삭제.

관련 항목

다음 AWS 블로그 게시물은 Auto Scaling 그룹을 사용하거나 여러 계정 내에서 노드를 Chef Automate 서버와 자동으로 연결하는 방법에 대한 자세한 정보를 제공합니다.

- Using AWS OpsWorks for Chef Automate to Manage EC2 Instances with Auto Scaling
- OpsWorks for Chef Automate - 서로 다른 계정에서 노드 자동 구성

AWS OpsWorks for Chef Automate 서버 삭제

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 섹션에서는 AWS OpsWorks for Chef Automate 서버를 삭제하는 방법을 설명합니다. 서버를 삭제하면 서버의 이벤트, 로그 및 서버에 저장된 모든 쿡북도 삭제됩니다. 지원 리소스(Amazon Elastic Compute Cloud 인스턴스, Amazon Elastic Block Store 볼륨 등)도 모든 자동 백업과 함께 삭제됩니다.

서버를 삭제해도 노드가 삭제되지는 않지만 노드는 삭제된 서버에 의해 더 이상 관리되지 않으며 계속 재연결을 시도합니다. 이런 이유로 Chef 서버를 삭제하기 전에 관리되는 노드의 연결을 해제하는 것이 좋습니다. 이 릴리스에서는 AWS CLI 명령을 실행하여 노드의 연결을 해제할 수 있습니다.

1단계: 관리되는 노드 연결 해제

Chef 서버 삭제 전에 서버에서 노드 연결을 해제해야 노드가 서버에 재연결을 시도하지 않고 계속 작동합니다. 이렇게 하려면 `disassociate-node` AWS CLI 명령을 실행합니다.

노드 연결 해제

1. 에서 다음 명령을 AWS CLI실행하여 노드 연결을 해제합니다. *`Server_name`*은 노드를 연결 해제할 Chef 서버의 이름입니다.

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 연결 해제가 완료되었다는 응답 메시지가 표시될 때까지 기다리십시오.

2단계: 서버 삭제

1. 대시보드의 서버 타일에서 [작업] 메뉴를 확장합니다.
2. [서버 삭제]를 선택합니다.
3. 삭제 확인 메시지가 표시되면 [예]를 선택합니다.

Chef Automate 대시보드 자격 증명 재설정

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

Chef Automate 대시보드 로그인에 사용하는 암호는 주기적으로 변경하는 것이 좋습니다. 이 섹션에 표시된 Amazon EC2 Systems Manager AWS CLI 명령을 사용하여 Chef Automate 대시보드 암호를 분실한 경우 변경할 수도 있습니다. 사용하는 명령은 Chef Automate 서버가 Chef Automate 버전 1을 실행하는지 아니면 버전 2를 실행하는지에 따라 다릅니다.

1. Chef 서버의 인스턴스 ID를 반환하려면 다음 페이지에서 AWS Management Console 를 엽니다.

   https://console.aws.amazon.com/ec2/v2/home?region=*region_of_your_server*#Instances:search=aws-opsworks-cm-*server_name*

   예를 들어 미국 서부(오레곤) 리전에서 MyChefServer라는 이름의 Chef 서버의 콘솔 URL은 다음과 같습니다.

   https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:search=aws-opsworks-cm-MyChefServer

   암호를 변경하는 데 필요하므로 콘솔에 표시된 인스턴스 ID를 적어 둡니다.

2. Chef Automate 대시보드 로그인 비밀번호를 재설정하려면 서버에서 Chef Automate 1을 실행하는지 Chef Automate 2를 실행하는지에 따라 다음 AWS CLI 명령 중 하나를 실행하세요. *enterprise_name*은 엔터프라이즈 또는 기업 이름으로, *user_name*은 서버 관리자의 사용자 이름으로, *new_password*는 사용하려는 암호로, *region_name*은 서버가 위치한 리전으로 바꿉니다. 엔터프라이즈 이름을 지정하지 않으면 엔터프라이즈 이름은 `default`가 됩니다. 기본적으로 *enterprise_name*은 `default`입니다(항상 프로비저닝되는 조직의 이름). *user_name*의 경우 만 라는 사용자를 AWS OpsWorks for Chef Automate 생성합니다`admin`. 새 암호를 적어 두고 찾기 쉽고 안전한 위치에 보관합니다.

   Chef Automate 1의 경우:

```
aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
 password" --instance-ids "instance_id"
--parameters commands="sudo delivery-ctl reset-
password enterprise_name user_name new_password" --region region_name --output text
```

Chef Automate 2의 경우:

```
aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
 password" --instance-ids "instance_id"
--parameters commands="sudo chef-automate iam admin-access restore new_password" --
region region_name --output text
```

3. 출력 텍스트(이 경우에는 명령 ID)에 암호 변경이 완료되었다고 표시될 때까지 기다립니다.

를 사용하여 AWS OpsWorks for Chef Automate API 호출 로깅 AWS CloudTrail

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

AWS OpsWorks for Chef Automate 는 IAM 자격 증명 또는 AWS CloudTrail의 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다 AWS OpsWorks for Chef Automate. CloudTrail은 AWS OpsWorks for Chef Automate 콘솔의 호출 및 API에 대한 코드 호출을 포함하여에 대한 모든 API 호출을 이벤트 AWS OpsWorks for Chef Automate 로 캡처합니다. AWS OpsWorks for Chef Automate APIs 추적을 생성하면 이벤트를 포함하여 CloudTrail 이벤트를 Amazon S3 버킷으로 지속적으로 전송할 수 있습니다 AWS OpsWorks for Chef Automate. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여에 수행된 요청, 요청이 수행된 AWS OpsWorks for Chef Automate IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 설명은 AWS CloudTrail 사용자 가이드를 참조하십시오.

AWS OpsWorks for Chef Automate CloudTrail의 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. 에서 활동이 발생하면 AWS OpsWorks for Chef Automate해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

에 대한 이벤트를 포함하여 AWS 계정의 이벤트를 지속적으로 기록하려면 추적을 AWS OpsWorks for Chef Automate생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 AWS OpsWorks for Chef Automate 작업은 CloudTrail에서 로깅되며 AWS OpsWorks for Chef Automate API 참조에 문서화됩니다. 예를 들어 CreateServer, CreateBackup 및 DescribeServers 작업에 대한 호출은 CloudTrail 로그 파일에 항목을 생성합니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청을 했는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

AWS OpsWorks for Chef Automate 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일

요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음 예제는 AWS OpsWorks for Chef Automate `CreateServer` 작업에 대한 CloudTrail 로그 항목을 표시합니다.


```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"OpsChef-test-server",
    "engineModel":"Single",
    "engine":"Chef",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
```

```
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"OpsChef-test-server-thohsgreckcnwgz3.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Chef",
        "instanceType":"t2.medium",
        "serverName":"OpsChef-test-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/OpsChef-test-
server/8epp7f6z-e91f-4z10-89z5-8c6219cdb09f",
        "engineModel":"Single",
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"CHEF_STARTER_KIT","value":"*** Redacted ***"},
            {"name":"CHEF_PIVOTAL_KEY","value":"*** Redacted ***"},
            {"name":"CHEF_DELIVERY_ADMIN_PASSWORD","value":"*** Redacted ***"}],
        "engineVersion":"12.11.1",
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7f64f9-d394-12ug-8081-7bb0386fbcb6",
"eventID":"8r7b18df-6c90-47be-87cf-e8346428cfc3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```


문제 해결 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate는 2024년 5월 5일에 만료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 기존 고객은 Chef SaaS 또는 대체 솔루션으로 마이그레이션하

는 것이 좋습니다. 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

이 주제에는 몇 가지 일반적인 AWS OpsWorks for Chef Automate 문제와 해당 문제에 대해 제안된 해결책이 포함되어 있습니다.

주제

- 일반적인 문제 해결 팁
- 구체적 오류 해결
- 추가 도움말 및 지원

일반적인 문제 해결 팁

Chef 서버를 생성 또는 사용할 수 없는 경우, 오류 메시지나 로그를 보고 문제 해결에 도움을 받을 수 있습니다. 다음 작업은 Chef 서버 문제 해결 시 일반적인 첫 단계를 설명합니다. 구체적인 오류와 해결책에 대해서는 이 주제의 구체적 오류 해결 단원을 참조하세요.

- Chef 서버가 시작되지 않는 경우 AWS OpsWorks for Chef Automate 콘솔을 사용하여 오류 메시지를 볼 수 있습니다. Chef 서버 세부 정보 페이지 상단에는 서버 시작 및 실행과 관련된 오류 메시지가 표시됩니다. 오류는 Chef 서버를 생성하는 데 사용되는 서비스 AWS OpsWorks for Chef Automate AWS CloudFormation또는 Amazon EC2에서 발생할 수 있습니다. 세부 정보 페이지에서는 실행 중인 서버에서 발생하는 이벤트도 볼 수 있으며, 여기에 실패 이벤트 메시지가 포함될 수 있습니다.
- EC2 문제를 해결하려면 SSH를 사용하여 서버의 인스턴스에 연결하고 로그를 확인합니다. EC2 인스턴스 로그는 `/var/log/aws/opsworks-cm` 디렉터리에 저장됩니다. 이러한 로그는가 Chef 서버를 AWS OpsWorks for Chef Automate 시작하는 동안 명령 출력을 캡처합니다.

구체적 오류 해결

주제

- 서버가 연결 끊김 상태입니다.
- Chef Automate 대시보드에서 관리형 노드가 Missing 열에 나타납니다
- Chef 볼트를 생성할 수 없고 knife vault 명령이 오류와 함께 실패합니다
- "requested configuration is currently not supported" 메시지와 함께 서버 생성이 실패합니다
- Chef Automate 대시보드에서 추가된 조직 이름을 Chef 서버가 인식하지 못합니다

- 서버의 Amazon EC2 인스턴스를 생성할 수 없습니다.
- 서비스 역할 오류로 서버가 생성되지 않습니다
- 탄력적 IP 주소 제한 초과
- Chef Automate 대시보드에 로그인할 수 없습니다
- 무인 노드 연결이 실패합니다
- 시스템 유지 관리 실패

서버가 연결 끊김 상태입니다.

문제: 서버 상태가 연결 끊김으로 표시됩니다.

원인: 외부의 엔터티 AWS OpsWorks 가 AWS OpsWorks for Chef Automate 서버 또는 지원 리소스를 변경할 때 가장 일반적으로 발생합니다.는 백업 생성, 운영 체제 패치 적용 또는 Chef Automate 업데이트와 같은 유지 관리 작업을 처리하기 위해 연결 손실 상태의 Chef Automate 서버에 연결할 AWS OpsWorks 수 없습니다. 따라서 서버에 중요한 업데이트가 누락되거나 보안 문제가 발생하기 쉽거나 예상대로 작동하지 않을 수 있습니다.

해결 방법: 다음 단계를 수행하여 서버 연결을 복원하세요.

1. 서비스 역할에 필요한 모든 권한이 있는지 확인하세요.
    a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 서비스 역할의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 서비스 역할이 열립니다.
    b. 권한 탭에서 권한 정책 목록에 `AWSOpsWorksCMServiceRole`이(가) 있는지 확인합니다. 목록에 없는 경우 `AWSOpsWorksCMServiceRole` 관리형 정책을 역할에 수동으로 추가하세요.
    c. 신뢰 관계 탭에서 서비스 역할에 `opsworks-cm.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.
2. 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.
    a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 인스턴스 프로파일의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 인스턴스 프로파일이 열립니다.
    b. 권한 탭에서 권한 정책 목록에 `AmazonEC2RoleforSSM` 및 `AWSOpsWorksCMInstanceProfileRole`이(가) 둘 다 있는지 확인합니다. 둘 중 하나 또는 둘 다 목록에 없는 경우 이러한 관리형 정책을 역할에 수동으로 추가하세요.

   c. 신뢰 관계 탭에서 서비스 역할에 `ec2.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

3. Amazon EC2 콘솔에서 AWS OpsWorks for Chef Automate 서버 리전과 동일한 리전에 있는지 확인한 다음 서버에서 사용 중인 EC2 인스턴스를 다시 시작합니다.

   a. *### aws-opsworks-cm-instance- server-name*인 EC2 인스턴스를 선택합니다.

   b. 인스턴스 상태 메뉴에서 인스턴스 재부팅을 선택합니다.

   c. 서버가 다시 시작되고 완전히 온라인 상태가 될 때까지 최대 15분이 걸릴 수 있습니다.

4. AWS OpsWorks for Chef Automate 콘솔의 서버 세부 정보 페이지에서 서버 상태가 정상인지 확인합니다.

이전 단계를 수행한 후에도 서버 상태가 여전히 연결 끊김으로 표시되면 다음 중 하나를 시도해 보세요.

- 새 서버를 만들고 원본을 삭제하여 서버를 교체하세요. 현재 서버의 데이터가 중요한 경우 최근 백업에서 서버를 복원하고 데이터가 최신 상태인지 확인한 다음 응답이 없는 원래 서버를 삭제하세요.
- AWS 고객 지원에 문의하세요.

Chef Automate 대시보드에서 관리형 노드가 Missing 열에 나타납니다

문제: 관리형 노드가 Chef Automate 대시보드의 [Missing] 열에 나타납니다.

원인: 노드가 12시간 이상 Chef Automate 서버에 연결되지 않고 `chef-client`가 노드에서 실행될 수 없으면 노드는 12시간 전의 상태에서 변경되고 Chef Automate 대시보드의 [누락] 열로 이동합니다.

해결 방법: 노드가 온라인 상태인지 확인합니다. `knife node show` *node_name* `--run-list`를 실행하여 `chef-client`가 노드에서 실행될 수 있는지 확인하거나 `knife node show -l` *node_name*을 실행하여 노드에 관한 모든 정보를 표시해 보십시오. 노드가 오프라인 상태이거나 네트워크에서 분리되었을 수 있습니다.

Chef 볼트를 생성할 수 없고 **`knife vault`** 명령이 오류와 함께 실패합니다

문제: `knife vault` 명령을 실행하여 Chef Automate 서버에서 볼트(예: 도메인을 조인하는 Windows 기반 노드용 자격 증명을 저장하는 볼트)를 생성하려 하고 있습니다. 이 명령은 다음과 비슷한 오류 메시지를 반환합니다.

```
WARN: Auto inflation of JSON data is deprecated. Please pass in the class to inflate or
 use #edit_hash (CHEF-1)
at /opt/chefdk/embedded/lib/ruby/2.3.0/forwardable.rb:189:in `edit_data'.Please see
 https://docs.chef.io/deprecations_json_auto_inflate.html
for further details and information on how to correct this problem.
WARNING: pivotal not found in users, trying clients.
ERROR: ChefVault::Exceptions::AdminNotFound: FATAL: Could not find pivotal in users or
 clients!
```

`knife user list`를 원격으로 실행하면 중심 사용자가 반환되지 않지만 Chef Automate 서버에서 로컬로 `chef-server-ctl user-show` 명령을 실행하면 결과에서 중심 사용자를 볼 수 있습니다. 다시 말해 `knife vault` 명령은 중심 사용자를 찾을 수 없지만 존재한다는 것은 알고 있습니다.

원인: 중심 사용자는 Chef에서 수퍼유저로 간주되고 모든 권한을 갖지만 AWS OpsWorks for Chef Automate에서 사용되는 `default` 조직을 비롯한 어떤 조직에도 속하지 않습니다. `knife user list` 명령은 Chef 구성 내 현재 조직에 있는 모든 사용자를 반환합니다. `chef-server-ctl user-show` 명령은 조직에 상관없이 중심 사용자를 비롯한 모든 사용자를 반환합니다.

해결 방법: 이 문제를 해결하려면 `knife opc`를 실행하여 기본 조직에 중심 사용자를 추가합니다.

먼저 knife-opc 플러그인을 설치해야 합니다.

```
chef gem install knife-opc
```

플러그인을 설치한 후 다음 명령을 실행하여 기본 조직에 중심 사용자를 추가합니다.

```
knife opc org user add default pivotal
```

`knife user list`를 실행하면 중심 사용자가 기본 조직에 속하는지 확인할 수 있습니다. `pivotal`이 결과에서 나열되어야 합니다. 그런 다음 `knife vault`를 다시 실행해 보십시오.

"requested configuration is currently not supported" 메시지와 함께 서버 생성이 실패합니다

문제: Chef Automate 서버를 생성하려고 하지만 "The requested configuration is currently not supported. Please check the documentation for supported configurations."와 비슷한 오류 메시지와 함께 서버 생성이 실패합니다.

원인: Chef Automate 서버에 대해 지원되지 않는 인스턴스 유형을 지정했을 수 있습니다. 전용 인스턴스용 VPC처럼 기본이 아닌 테넌시가 있는 VPC에서 Chef Automate 서버를 생성하는 경우, 지정된

VPC 내부의 모든 인스턴스도 전용 또는 호스트 테넌시여야 합니다. t2와 같은 일부 인스턴스 유형은 기본 테넌시에서만 사용할 수 있기 때문에 Chef Automate 서버 인스턴스 유형은 지정한 VPC에서 지원 불가능할 수 있으며, 서버 생성이 실패합니다.

해결 방법: 기본이 아닌 테넌시가 있는 VPC를 선택하는 경우, 전용 테넌시를 지원할 수 있는 m4 인스턴스 유형을 사용하세요.

Chef Automate 대시보드에서 추가된 조직 이름을 Chef 서버가 인식하지 못합니다

문제: Chef Automate 대시보드에 새 Workflow 조직 이름을 추가하거나 무인 노드 연결 스크립트에서 `"default"` 아닌 `CHEF_AUTOMATE_ORGANIZATION` 값을 지정했지만 노드 연결이 실패합니다. AWS OpsWorks for Chef Automate 서버가 새 조직 이름을 인식하지 못합니다.

원인: Workflow 조직 이름과 Chef 서버 조직 이름이 다릅니다. 웹 기반 Chef Automate 대시보드에서 새 Workflow 조직을 생성할 수 있지만 Chef 서버 조직 이름은 불가능합니다. Chef Automate 대시보드는 기존 Chef 서버 조직을 보는 데만 사용할 수 있습니다. Chef Automate 대시보드에서 생성하는 새 조직은 Workflow 조직이며, Chef 서버에 의해 인식되지 않습니다. 새 조직 이름은 노드 연결 스크립트에서 지정하여 생성할 수 없습니다. 조직이 먼저 Chef 서버에 추가되지 않은 상태에서 노드 연결 스크립트에서 조직 이름을 언급하면 노드 연결이 실패합니다.

해결 방법: Chef 서버에서 인식되는 새 조직을 생성하려면 `knife opc org create` 명령을 사용하거나 `chef-server-ctl org-create`를 실행하세요.

서버의 Amazon EC2 인스턴스를 생성할 수 없습니다.

문제: 다음과 비슷한 오류 메시지와 함께 서버 생성이 실패했습니다. "The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration."

원인: 가장 가능성 높은 원인은 EC2 인스턴스에 네트워크 액세스 권한이 없기 때문입니다.

해결 방법: 인스턴스에 아웃바운드 인터넷 액세스 권한이 있고 AWS 서비스 에이전트가 명령을 실행할 수 있는지 확인합니다. VPC(단일 퍼블릭 서브넷이 포함된 VPC)에서 [DNS 확인]이 활성화되어 있고, 서브넷에서 [퍼블릭 IP 자동 할당] 설정이 활성화되어 있어야 합니다.

서비스 역할 오류로 서버가 생성되지 않습니다

문제: "Not authorized to perform sts:AssumeRole"이라는 오류 메시지와 함께 서버 생성이 실패합니다.

원인: 이 문제는 사용 중인 서비스 역할에 새 서버를 생성할 수 있는 적절한 권한이 없을 때 발생할 수 있습니다.

해결 방법: AWS OpsWorks for Chef Automate 콘솔을 열고 콘솔을 사용하여 새 서비스 역할과 인스턴스 프로파일 역할을 생성합니다. 자체 서비스 역할을 사용하려는 경우, [AWSOpsWorksCMServiceRole] 정책을 역할에 연결합니다. opsworks-cm.amazonaws.com이 역할의 신뢰 관계에 있는 서비스에 등재되어 있는지 확인하세요. Chef 서버에 연결된 서비스 역할에 AWSOpsWorksCMServiceRole 관리형 정책이 연결되어 있는지 확인합니다.

탄력적 IP 주소 제한 초과

문제: "The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached"라는 오류 메시지와 함께 서버 생성이 실패합니다.

원인: 이 문제는 계정이 탄력적 IP(EIP) 주소 최대 수를 사용한 경우에 발생합니다. 기본 EIP 주소 한도는 5입니다.

해결 방법: 기존 EIP 주소를 해제하거나 계정이 적극적으로 사용하지 않는 주소를 삭제하거나 AWS 고객 지원에 문의하여 계정과 연결된 EIP 주소의 제한을 늘릴 수 있습니다.

Chef Automate 대시보드에 로그인할 수 없습니다

문제: Chef Automate 대시보드에 다음과 비슷한 오류가 표시됩니다. "Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://myserver-name.region.opsworks-cm.io/api/v0/e/default/verify-token. (Reason: CORS header 'Access-Control-Allow-Origin' missing)". 이 오류는 "The User Id / Password combination entered is incorrect"와도 비슷할 수 있습니다.

원인: Chef Automate 대시보드는 FQDN을 명시적으로 설정하며, 관련 URL을 허용하지 않습니다. 이 때는 Chef 서버의 IP 주소를 사용하여 로그인할 수 없고 서버의 DNS 이름을 사용해야만 로그인할 수 있습니다.

해결 방법: Chef 서버의 IP 주소가 아니라 DNS 이름 항목만 사용하여 Chef Automate 대시보드에 로그인하세요. Chef Automate 대시보드 자격 증명 재설정에 설명된 대로 AWS CLI 명령을 실행하여 Chef Automate 대시보드 자격 증명을 재설정해 볼 수도 있습니다.

무인 노드 연결이 실패합니다

문제: 새 Amazon EC2 노드의 무인 또는 자동 연결이 실패합니다. Chef 서버에 추가되었어야 할 노드가 Chef Automate 대시보드에 나타나지 않고 `knife client show` 또는 `knife node show` 명령의 결과에 나열되지 않습니다.

원인: 이 문제는 opsworks-cm API 호출이 새 EC2 인스턴스와 통신하는 것을 허용하는 인스턴스 프로파일로 IAM 역할이 설정되어 있지 않을 때 발생할 수 있습니다.

해결 방법: 에서 노드 자동 추가 AWS OpsWorks for Chef Automate에 설명된 대로 AssociateNode 및 DescribeNodeAssociationStatus API 호출이 EC2에서 작동하도록 허용하는 정책을 EC2 인스턴스 프로파일에 연결하세요.

시스템 유지 관리 실패

AWS OpsWorks CM 는 매주 시스템 유지 관리를 수행하여 보안 업데이트를 포함한 최신 마이너 버전의 Chef Server 및 Chef Automate Server가 항상 AWS OpsWorks for Chef Automate 서버에서 실행되도록 합니다. 어떤 이유로든 시스템 유지 관리가 실패하면에서 실패를 AWS OpsWorks CM 알립니다. 시스템 유지 관리에 대한 자세한 내용은 의 시스템 유지 관리 AWS OpsWorks for Chef Automate 섹션을 참조하세요.

이 섹션에서는 가능한 실패 원인을 설명하고 해결 방법을 제안합니다.

주제

- 서비스 역할 또는 인스턴스 프로파일 오류로 인해 시스템 유지 관리가 불가능합니다.

서비스 역할 또는 인스턴스 프로파일 오류로 인해 시스템 유지 관리가 불가능합니다.

문제: “STS:AssumeRole 수행 권한이 없음”이라는 오류 메시지 또는 권한에 대한 유사한 오류 메시지와 함께 시스템 유지 관리가 실패합니다.

원인: 사용 중인 서비스 역할 또는 인스턴스 프로파일에 서버에서 시스템 유지 관리를 수행할 수 있는 적절한 권한이 없는 경우 이 문제가 발생할 수 있습니다.

해결 방법: 서비스 역할과 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.

1. 서비스 역할에 필요한 모든 권한이 있는지 확인하세요.

    a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 서비스 역할의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 서비스 역할이 열립니다.

    b. 권한 탭에서 서비스 역할에 AWSOpsWorksCMServiceRole이(가) 연결되어 있는지 확인합니다. AWSOpsWorksCMServiceRole이(가) 목록에 없는 경우 이 정책을 역할에 추가하세요.

    c. opsworks-cm.amazonaws.com이 역할의 신뢰 관계에 있는 서비스에 등재되어 있는지 확인하세요. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

2. 인스턴스 프로파일에 필요한 모든 권한이 있는지 확인하세요.

   a. 서버 설정 페이지의 네트워크 및 보안에서 서버가 사용하는 인스턴스 프로파일의 링크를 선택합니다. 그러면 IAM 콘솔에서 볼 수 있는 인스턴스 프로파일이 열립니다.

   b. 권한 탭에서 권한 정책 목록에 `AmazonEC2RoleforSSM` 및 `AWSOpsWorksCMInstanceProfileRole`이(가) 둘 다 있는지 확인합니다. 둘 중 하나 또는 둘 다 목록에 없는 경우 이러한 관리형 정책을 역할에 수동으로 추가하세요.

   c. 신뢰 관계 탭에서 서비스 역할에 `ec2.amazonaws.com` 서비스가 사용자를 대신하여 역할을 맡도록 신뢰하는 신뢰 정책이 있는지 확인합니다. 역할에 신뢰 정책을 사용하는 방법에 대한 자세한 내용은 역할 수정(콘솔) 또는 AWS 보안 블로그 게시물, IAM 역할에 신뢰 정책을 사용하는 방법을 참조하세요.

추가 도움말 및 지원

이 주제에 특정 문제가 설명되어 있지 않거나 이 주제의 제안을 시도했지만 여전히 문제가 계속되는 경우, AWS OpsWorks 포럼을 방문하세요.

AWS Support Center를 방문할 수도 있습니다. AWS 지원 센터는 지원 사례를 생성하고 관리하기 AWS 위한 허브입니다. AWS 지원 센터에는 포럼, 기술 FAQs, 서비스 상태 및 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다 AWS Trusted Advisor.

AWS OpsWorks 구성 관리(CM)의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 클라우드에서 AWS AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. AWS OpsWorks CM에 적용되는 규정 준수 프로그램에 대해 알아보려면 규정 준수 프로그램 제공 범위 내AWS 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS OpsWorks CM을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 AWS OpsWorks CM을 구성하는 방법을 보여줍니다. 또한 AWS OpsWorks CM 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- AWS OpsWorks CM의 데이터 보호
- 데이터 암호화
- AWS OpsWorks CM에 대한 자격 증명 및 액세스 관리
- 인터네트워크 트래픽 개인 정보
- AWS OpsWorks CM에서 로깅 및 모니터링
- AWS OpsWorks CM에 대한 규정 준수 확인
- AWS OpsWorks CM의 복원력
- AWS OpsWorks CM의 인프라 보안
- AWS OpsWorks CM의 구성 및 취약성 분석
- AWS OpsWorks CM의 보안 모범 사례

AWS OpsWorks CM의 데이터 보호

AWS 공동 책임 모델 AWS OpsWorks 구성 관리의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션과 내부의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 OpsWorks CM 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

OpsWorks CM 서버의 이름은 암호화되지 않습니다.

OpsWorks CM은 AWS OpsWorks for Chef Automate 및 AWS OpsWorks for Puppet Enterprise 서버를 생성하고 유지 관리하는 과정에서 다음과 같은 고객 데이터를 수집합니다.

- OpsWorks for Puppet Enterprise의 경우 Puppet Enterprise가 Puppet 마스터와 관리형 노드 간의 통신을 활성화하기 위해 사용하는 프라이빗 키를 수집합니다.
- 의 경우 AWS OpsWorks for Chef Automate사용자 지정 도메인을 사용하는 경우 서비스에 연결하는 인증서에 대한 프라이빗 키를 수집합니다. 사용자 지정 도메인으로 Chef Automate 서버를 생성할 때 제공하는 프라이빗 키는 서버를 통해 전달됩니다.

OpsWorks CM 서버는 Chef Cookbook 또는 Puppet Enterprise 모듈과 같은 구성 코드를 저장합니다. 이 코드는 서버 백업에 저장되지만는 액세스할 수 AWS 없습니다. 이 콘텐츠는 암호화되며 AWS 계정의 관리자만 액세스할 수 있습니다. 소스 리포지토리에 권장되는 프로토콜을 사용하여 Chef 또는 Puppet 구성 코드를 보호하는 것이 좋습니다. 예를 들어 의 리포지토리에 대한 권한을 제한 AWS CodeCommit하거나 GitHub 웹 사이트의 GitHub 리포지토리 보안 지침을 따를 수 있습니다.

OpsWorks CM은 서비스를 유지 관리하거나 고객 로그를 유지하기 위해 고객이 제공한 콘텐츠를 사용하지 않습니다. OpsWorks CM 서버에 대한 로그는 계정에 Amazon S3 버킷으로 저장됩니다. OpsWorks CM 서버에 연결하는 사용자의 IP 주소는 AWS에 의해 로깅됩니다.

와 통합 AWS Secrets Manager

2021년 5월 3일부터 OpsWorks CM에서 새 서버를 생성할 때 OpsWorks CM은 서버의 보안 암호를 AWS Secrets Manager에 저장합니다. 새 서버의 경우 다음 속성이 Secrets Manager에 암호로 저장됩니다.

- Chef Automate 서버
  - HTTPS 프라이빗 키(사용자 지정 도메인을 사용하지 않는 서버만 해당)
  - Chef Automate 관리자 비밀번호 (CHEF_AUTOMATE_ADMIN_PASSWORD)
- Puppet Enterprise 마스터
  - HTTPS 프라이빗 키(사용자 지정 도메인을 사용하지 않는 서버만 해당)
  - Puppet 관리자 비밀번호 (PUPPET_ADMIN_PASSWORD)
  - 원격 Puppet r10k(PUPPET_R10K_REMOTE)

사용자 지정 도메인을 사용하지 않는 기존 서버의 경우 Chef Automate와 Puppet Enterprise 서버 모두에 대해 Secrets Manager에 저장되는 유일한 보안 암호는 HTTPS 프라이빗 키입니다. 이는 자동 주간 시스템 유지 관리 중에 생성되기 때문입니다.

OpsWorks CM은 보안 정보를 Secrets Manager에 자동으로 저장하며, 이 동작은 사용자가 구성할 수 없습니다.

데이터 암호화

AWS OpsWorks CM은 승인된 AWS 사용자와 AWS OpsWorks CM 서버 간의 서버 백업 및 통신을 암호화합니다. 그러나 AWS OpsWorks CM 서버의 루트 Amazon EBS 볼륨은 암호화되지 않습니다.

유휴 데이터 암호화

AWS OpsWorks CM 서버 백업은 암호화됩니다. 그러나 AWS OpsWorks CM 서버의 루트 Amazon EBS 볼륨은 암호화되지 않습니다. 이는 사용자가 구성할 수 없습니다.

전송 중 데이터 암호화

AWS OpsWorks CM은 TLS 암호화와 함께 HTTP를 사용합니다. AWS OpsWorks 사용자가 서명된 인증서를 제공하지 않는 경우 CM은 기본적으로 자체 서명된 인증서를 사용하여 서버를 프로비저닝하고 관리합니다. CA(인증 기관)에서 서명한 인증서를 사용하는 것이 좋습니다.

키 관리

AWS Key Management Service 고객 관리형 키와 AWS 관리형 키는 현재 AWS OpsWorks CM에서 지원되지 않습니다.

AWS OpsWorks CM에 대한 자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 누가 OpsWorks CM 리소스를 사용하도록 인증되고(로그인됨)권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- 대상
- 자격 증명을 통한 인증
- 정책을 사용하여 액세스 관리
- AWS OpsWorks CM과 IAM의 작동 방식
- AWS OpsWorks CM 자격 증명 기반 정책 예제
- AWS OpsWorks CM 자격 증명 및 액세스 문제 해결
- AWSAWS OpsWorks 구성 관리를 위한 관리형 정책
- 의 교차 서비스 혼동된 대리자 방지 AWS OpsWorks CM

대상

AWS Identity and Access Management (IAM) 사용 방법은 OpsWorks CM에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - OpsWorks CM 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 OpsWorks CM 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. OpsWorks CM의 기능에 액세스할 수 없는 경우 AWS OpsWorks CM 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 - 회사에서 OpsWorks CM 리소스를 책임지고 있는 경우 OpsWorks CM에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 OpsWorks CM 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 OpsWorks CM에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 AWS OpsWorks CM과 IAM의 작동 방식 섹션을 참조하세요.

IAM 관리자 - IAM 관리자라면 OpsWorks CM에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 OpsWorks CM 자격 증명 기반 정책 예제를 보려면 AWS OpsWorks CM 자격 증명 기반 정책 예제섹션을 참조하세요.

자격 증명을 통한 인증

인증은 자격 증명 AWS 으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를

사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 AWS 계정 테루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 한 사람 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

> **Warning**
>
> IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
  - 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 대상에 액세스할 수 있는 사용자를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

자격 증명 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지

를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

OpsWorks CM에서는 IAM에서 생성하고 사용자, 역할 또는 그룹에 연결하는 사용자 지정 정책을 지원합니다.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

OpsWorks CM은 리소스 기반 정책을 지원하지 않습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

OpsWorks CM은 ACL을 사용하지 않습니다.

기타 정책 유형

OpsWorks CM은 다음과 같은 다른 정책 유형을 지원하지 않습니다.

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유한 여러 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. Organizations 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 가 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS OpsWorks CM과 IAM의 작동 방식

IAM을 사용하여 AWS OpsWorks CM에 대한 액세스를 관리하기 전에 AWS OpsWorks CM에서 사용할 수 있는 IAM 기능을 이해해야 합니다. AWS OpsWorks CM 및 기타 AWS 서비스가 IAM에서 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM에서 작동하는 서비스를 참조하세요.

주제

- AWS OpsWorks CM 자격 증명 기반 정책
- AWS OpsWorks CM 및 리소스 기반 정책
- AWS OpsWorks CM 태그를 기반으로 한 권한 부여

- AWS OpsWorks CM IAM 역할

AWS OpsWorks CM 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부된 작업 및 리소스와 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. AWS OpsWorks CM은 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

AWS OpsWorks CM에서는 사용자 지정 정책 설명을 사용자, 역할 또는 그룹에 연결할 수 있습니다.

작업

IAM 자격 증명 기반 정책의 `Action` 요소는 정책에 따라 허용되거나 거부되는 특정 작업에 대해 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 이 작업은 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에서 사용됩니다.

AWS OpsWorks CM의 정책 작업은 작업 앞에 접두사를 사용합니다`opsworks-cm:`. 예를 들어, API 작업으로 AWS OpsWorks CM 서버를 생성할 수 있는 권한을 부여하려면 해당 정책에 `opsworks-cm:CreateServer` 작업을 포함합니다. 정책 설명에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. AWS OpsWorks CM은 이 서비스로 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "opsworks-cm:action1",
      "opsworks-cm:action2"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "opsworks-cm:Describe*"
```

와일드카드를 사용하여 정책 설명에서 여러 작업을 허용하는 경우 권한이 부여된 서비스 또는 사용자에 대해서만 이러한 작업을 허용해야 합니다.

AWS OpsWorks CM 작업 목록을 보려면 IAM 사용 설명서의 AWS OpsWorks에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

리소스

Resource 요소는 작업이 적용되는 객체를 지정합니다. 문에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. ARN을 사용하거나 문이 모든 리소스에 적용됨을 표시하는 와일드카드(*)를 사용하여 리소스를 지정합니다.

DescribeServers 또는 DescribeBackups API 작업을 실행하여 AWS OpsWorks CM 서버 또는 백업의 Amazon 리소스 번호(ARN)와 해당 리소스에 대한 기본 리소스 수준 정책을 가져올 수 있습니다.

AWS OpsWorks CM 서버 리소스의 ARN 형식은 다음과 같습니다.

```
arn:aws:opsworks-cm:{Region}:${Account}:server/${ServerName}/${UniqueId}
```

AWS OpsWorks CM 백업 리소스의 ARN 형식은 다음과 같습니다.

```
arn:aws:opsworks-cm:{Region}:${Account}:backup/${ServerName}-{Date-and-Time-Stamp-of-Backup}
```

ARN 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARNs) 및 AWS 서비스 네임스페이스를 참조하세요.

예를 들어, 문에서 test-chef-automate Chef Automate 서버를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/test-chef-automate/EXAMPLE-d1a2bEXAMPLE"
```

특정 계정에 속하는 모든 AWS OpsWorks CM 서버를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/*"
```

다음 예제에서는 AWS OpsWorks CM 서버 백업을 리소스로 지정합니다.

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:backup/test-chef-automate-server-2018-05-20T19:06:12.399Z"
```

리소스 생성 작업과 같은 일부 AWS OpsWorks CM 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

```
"Resource": "*"
```

다양한 API 작업에는 여러 리소스가 관여합니다. 단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": [
      "resource1",
      "resource2"
```

AWS OpsWorks CM 리소스 유형 및 해당 ARNs의 목록을 보려면 IAM 사용 설명서의 AWS OpsWorks CM에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 IAM 사용 설명서의 AWS OpsWorks CM에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

조건 키

AWS OpsWorks CM에는 정책 설명의 Condition 요소에 사용할 수 있는 서비스별 컨텍스트 키가 없습니다. 모든 서비스에 사용할 수 있는 글로벌 컨텍스트 키 목록을 보려면 IAM 정책 참조의 AWS 글로벌 조건 컨텍스트 키를 참조하세요. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같음, 미만 등 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 빌드할 수 있습니다.

한 문에서 여러 Condition 요소를 지정하거나 단일 Condition 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 AND 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 OR 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, 사용자에게 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그 섹션을 참조하십시오.

예시

AWS OpsWorks CM 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS OpsWorks CM 자격 증명 기반 정책 예제.

AWS OpsWorks CM 및 리소스 기반 정책

AWS OpsWorks CM은 리소스 기반 정책을 지원하지 않습니다.

리소스 기반 정책은 지정된 보안 주체가 리소스에 대해 수행할 수 있는 작업 및 관련 조건을 지정하는 JSON 정책 문서입니다.

AWS OpsWorks CM 태그를 기반으로 한 권한 부여

AWS OpsWorks CM 리소스에 태그를 연결하거나 요청에서 태그를 AWS OpsWorks CM에 전달할 수 있습니다. 태그를 기반으로 액세스를 제어하려면 `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. AWS OpsWorks CM 리소스 태그 지정에 대한 자세한 내용은이 설명서AWS OpsWorks for Puppet Enterprise 리소스에서 태그 작업의 AWS OpsWorks for Chef Automate 리소스에서 태그 작업 또는 섹션을 참조하세요.

AWS OpsWorks CM IAM 역할

IAM 역할은 AWS 계정 내에서 특정 권한이 있는 엔터티입니다.

AWS OpsWorks CM은 두 가지 역할을 사용합니다.

- 사용자의 AWS 계정 내에서 작동할 수 있는 AWS OpsWorks CM 서비스 권한을 부여하는 서비스 역할입니다. OpsWorks CM에서 제공하는 기본 서비스 역할을 사용하는 경우 이 역할의 이름은 `aws-opsworks-cm-service-role`입니다.
- AWS OpsWorks CM 서비스가 OpsWorks CM API를 호출하도록 허용하는 인스턴스 프로파일 역할입니다. 이 역할은 Amazon S3 및 AWS CloudFormation 에 백업을 위한 서버 및 S3 버킷을 생성할 수 있는 액세스 권한을 부여합니다. OpsWorks CM에서 제공하는 기본 인스턴스 프로파일을 사용하는 경우 이 인스턴스 프로파일 역할의 이름은 `aws-opsworks-cm-ec2-role`입니다.

AWS OpsWorks CM은 서비스 연결 역할을 사용하지 않습니다.

AWS OpsWorks CM에서 임시 자격 증명 사용

AWS OpsWorks CM은 임시 자격 증명 사용을 지원하며에서 해당 기능을 상속합니다 AWS Security Token Service.

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

서비스 연결 역할

AWS OpsWorks CM은 서비스 연결 역할을 사용하지 않습니다.

서비스 연결 역할을 사용하면 AWS 서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수 없습니다.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

AWS OpsWorks CM은 두 가지 역할을 사용합니다.

- 사용자의 AWS 계정 내에서 작동할 수 있는 AWS OpsWorks CM 서비스 권한을 부여하는 서비스 역할입니다. OpsWorks CM에서 제공하는 기본 서비스 역할을 사용하는 경우 이 역할의 이름은 `aws-opsworks-cm-service-role`입니다.
- AWS OpsWorks CM 서비스가 OpsWorks CM API를 호출하도록 허용하는 인스턴스 프로파일 역할입니다. 이 역할은 Amazon S3 및 AWS CloudFormation 에 백업을 위한 서버 및 S3 버킷을 생성할 수 있는 액세스 권한을 부여합니다. OpsWorks CM에서 제공하는 기본 인스턴스 프로파일을 사용하는 경우 이 인스턴스 프로파일 역할의 이름은 `aws-opsworks-cm-ec2-role`입니다.

AWS OpsWorks CM에서 IAM 역할 선택

AWS OpsWorks CM에서 서버를 생성할 때 AWS OpsWorks CM이 사용자를 대신하여 Amazon EC2에 액세스할 수 있도록 허용하는 역할을 선택해야 합니다. 이전에 서비스 역할을 생성한 경우 AWS OpsWorks CM은 선택할 수 있는 역할 목록을 제공합니다. 역할을 지정하지 않으면 OpsWorks CM에서 역할을 생성할 수 있습니다. Amazon EC2 인스턴스 시작 및 중지에 대한 액세스를 허용하는 역할을 선택하는 것이 중요합니다. 자세한 내용은 Chef Automate 서버 생성 또는 Puppet Enterprise 마스터 생성을 참조하세요.

AWS OpsWorks CM 자격 증명 기반 정책 예제

기본적으로 사용자 또는 역할에는 AWS OpsWorks CM 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console AWS CLI또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 필요로 하는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 IAM 자격 증명 권한을 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예시 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

AWS OpsWorks CM에서 정책을 사용자에게 할당하여 사용자가 AWS Management Console 또는 `AWSOpsWorksCMServiceRole`를 사용하여 Chef Automate 또는 Puppet Enterprise 서버를 생성하고 관리할 수 있습니다 AWS CLI.

주제

- 정책 모범 사례
- 사용자가 자신이 권한을 볼 수 있도록 허용
- 태그를 기반으로 AWS OpsWorks CM 서버 보기

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 OpsWorks CM 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 이동 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 조건을 사용하여 AWS 서비스와 같은 특정를 통해 사용되는 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하

여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.

- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

사용자가 자신이 권한을 볼 수 있도록 허용

이 예제는 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여 줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함되어 있습니다.

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
                   "iam:ListPolicyVersions",
```

```
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

태그를 기반으로 AWS OpsWorks CM 서버 보기

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 AWS OpsWorks CM 서버 및 백업에 대한 액세스를 제어할 수 있습니다. 이 예제에서는 AWS OpsWorks CM 서버를 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 그러나 AWS OpsWorks CM 서버 태그에 해당 사용자의 사용자 이름 값이 Owner 있는 경우에만 권한이 부여됩니다. 이 정책은 콘솔에서 이 작업을 완료하는 데 필요한 권한도 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListServersInConsole",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "*"
        },
        {
            "Sid": "ViewServerIfOwner",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "arn:aws:opsworks-cm:region:master-account-ID:server/server-
name",
            "Condition": {
                "StringEquals": {"opsworks-cm:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

이 정책을 계정의 사용자에게 연결할 수 있습니다. 라는 사용자가 AWS OpsWorks CM 서버를 보려고 richard-roe 하면 서버에 Owner=richard-roe 또는 태그를 지정해야 합니다owner=richard-roe. 그렇지 않으면 액세스가 거부됩니다. 조건 키 이름은 대소문자를 구분하지 않기 때문에 조건 태그

키 Owner는 Owner 및 owner 모두와 일치합니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

AWS OpsWorks CM 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 IAM 작업에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다. AWS OpsWorks CM 관련 문제 해결 정보는 문제 해결 AWS OpsWorks for Chef Automate 및 섹션을 참조하세요OpsWorks for Puppet Enterprise 문제 해결.

주제

- AWS OpsWorks CM에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행할 권한이 없음
- 내 AWS 계정 외부의 사람이 내 AWS OpsWorks CM 리소스에 액세스하도록 허용하고 싶습니다.

AWS OpsWorks CM에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 지원을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 사용자가 콘솔을 사용하여 AWS OpsWorks CM 서버에 대한 세부 정보를 보mateojackson려고 하지만 opsworks-cm:DescribeServers 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 opsworks-cm:DescribeServers on resource: test-chef-automate-server
```

이 경우 Mateo는 opsworks-cm:DescribeServers 작업을 사용하여 test-chef-automate-server 리소스에 액세스하도록 허용하는 정책을 업데이트할 것을 관리자에게 요청합니다.

iam:PassRole을 수행할 권한이 없음

iam:PassRole 태스크를 수행할 권한이 없다는 오류가 수신되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다. 역할을 OpsWorks CM으로 전달하도록 허용하는 정책을 업데이트하도록 관리자에게 요청합니다.

일부 AWS 서비스를 사용하면 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor(이)라는 사용자가 콘솔을 사용하여 OpsWorks CM에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 태스크를 수행하려면 서비스에 서비스 역할이 부여한 권한이 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary는 iam:PassRole 태스크를 수행하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

내 AWS 계정 외부의 사람이 내 AWS OpsWorks CM 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- AWS OpsWorks CM은 둘 이상의 계정에서 사용자에게 AWS OpsWorks CM 서버를 관리할 수 있는 액세스 권한을 부여하는 것을 지원합니다.
- 소유한 AWS 계정에서 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른 AWS 계정의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 타사 AWS 계정에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 소유 AWS 계정에 대한 액세스 권한 제공을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하십시오.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

AWSAWS OpsWorks 구성 관리를 위한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은

일반적인 사용 사례를 다루며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트가 기존 권한을 손상시키지 않습니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: **AWSOpsWorksCMServiceRole**

`AWSOpsWorksCMServiceRole`을(를) IAM 엔티티에 연결할 수 있습니다. 또한 OpsWorks CM은 OpsWorks CM이 사용자를 대신하여 작업을 수행할 수 있도록 하는 서비스 역할에 이 정책을 연결합니다.

이 정책은 OpsWorks CM 관리자가 OpsWorks CM 서버 및 백업을 생성, 관리 및 삭제할 수 있는 ## 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `opsworks-cm` — 주체가 기존 서버를 삭제하고 유지 관리 실행을 시작할 수 있습니다.
- `acm` — 사용자가 OpsWorks CM 서버에 연결할 수 있는 AWS Certificate Manager 에서 인증서를 주체가 삭제하거나 가져올 수 있습니다.
- `cloudformation` - 보안 주체가 OpsWorks CM 서버를 생성, 업데이트 또는 삭제할 때 OpsWorks CM이 AWS CloudFormation 스택을 생성하고 관리할 수 있도록 허용합니다.
- `ec2` — 주체가 OpsWorks CM 서버를 생성, 업데이트 또는 삭제할 때 OpsWorks CM이 Amazon Elastic Compute Cloud 인스턴스를 시작, 프로비저닝, 업데이트 및 종료할 수 있도록 허용합니다.
- `iam` — OpsWorks CM 서버를 만들고 관리하는 데 필요한 서비스 역할을 OpsWorks CM에서 생성할 수 있습니다.
- `tag` — 주체가 서버 및 백업을 포함한 OpsWorks CM 리소스에서 태그를 적용하고 제거할 수 있습니다.

- `s3` — OpsWorks CM이 서버 백업을 저장하기 위한 Amazon S3 버킷을 생성하고, 주요 요청 시 S3 버킷의 객체를 관리(예: 백업 삭제)하고, 버킷을 삭제할 수 있습니다.
- `secretsmanager` — OpsWorks CM이 Secrets Manager 암호를 생성 및 관리하고 암호에서 태그를 적용하거나 제거할 수 있습니다.
- `ssm` — OpsWorks CM이 OpsWorks CM 서버인 인스턴스에서 Systems Manager 실행 명령을 사용할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket*"
            ],
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteObject",
                "s3:DeleteBucket",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:GetBucketTagging",
                "s3:PutBucketTagging"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "tag:UntagResources",
                "tag:TagResources"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
```

```
            ],
            "Action": [
                "ssm:DescribeInstanceInformation",
                "ssm:GetCommandInvocation",
                "ssm:ListCommandInvocations",
                "ssm:ListCommands"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ssm:SendCommand"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:*::document/*",
                "arn:aws:s3:::amzn-s3-demo-bucket*"
            ],
            "Action": [
                "ssm:SendCommand"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateImage",
                "ec2:CreateSecurityGroup",
```

```
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteSnapshot",
                "ec2:DeregisterImage",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSubnets",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:RunInstances",
                "ec2:StopInstances"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ec2:TerminateInstances",
                "ec2:RebootInstances"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:opsworks-cm:*:*:server/*"
            ],
            "Action": [
                "opsworks-cm:DeleteServer",
                "opsworks-cm:StartMaintenance"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/aws-opsworks-cm-*"
            ],
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateStack"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/aws-opsworks-cm-*",
                "arn:aws:iam::*:role/service-role/aws-opsworks-cm-*"
            ],
            "Action": [
                "iam:PassRole"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "acm:DeleteCertificate",
                "acm:ImportCertificate"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:TagResource",
                "secretsmanager:UntagResource"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteTags",
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:elastic-ip/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

AWS 관리형 정책: **AWSOpsWorksCMInstanceProfileRole**

AWSOpsWorksCMInstanceProfileRole을(를) IAM 엔티티에 연결할 수 있습니다. 또한 OpsWorks CM은 OpsWorks CM이 사용자를 대신하여 작업을 수행할 수 있도록 허용하는 서비스 역할에 이 정책을 연결합니다.

이 정책은 OpsWorks CM 서버로 사용되는 Amazon EC2 인스턴스가 AWS CloudFormation 및에서 정보를 가져오 AWS Secrets Manager고 Amazon S3 버킷에 서버 백업을 저장할 수 있도록 허용하는 ## 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- acm — OpsWorks CM 서버 EC2 인스턴스가 사용자가 OpsWorks CM 서버에 연결할 수 있도록 하는 AWS Certificate Manager 에서 인증서를 받을 수 있도록 허용합니다.
- cloudformation - OpsWorks CM 서버 EC2 인스턴스가 인스턴스 생성 또는 업데이트 프로세스 중에 AWS CloudFormation 스택에 대한 정보를 가져오고 상태에 AWS CloudFormation 대한 신호를 에 보낼 수 있도록 허용합니다.
- s3 — OpsWorks CM 서버 EC2 인스턴스가 S3 버킷의 서버 백업을 업로드 및 저장하고, 필요한 경우 업로드를 중지 또는 롤백하고, S3 버킷에서 백업을 삭제할 수 있습니다.
- secretsmanager — OpsWorks CM 서버 EC2 인스턴스가 OpsWorks CM 관련 Secrets Manager 암호 값을 가져올 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket*",
            "Effect": "Allow"
        },
        {
            "Action": "acm:GetCertificate",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Effect": "Allow"
        }
    ]
}
```

AWS 관리형 정책에 대한 OpsWorks CM 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 OpsWorks CM의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 OpsWorks CM 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWSOpsWorksCMInstanceProfileRole — 업데이트된 관리형 정책	OpsWorks CM은 OpsWorks CM 서버로 사용되는 EC2 인스턴스가 CloudFormation 및 Secrets Manager와 정보를 공유하고 백업을 관리할 수 있도록 허용하는 관리형 정책을 업데이트했습니다. 이 `opsworks-cm!` 변경으로 인해 Secrets Manager 암호의 리소스 이름에 이 추가되어 OpsWorks CM이 암호를 소유할 수 있게 되었습니다.	2021년 4월 23일
AWSOpsWorksCMServiceRole - 업데이트된 관리형 정책	OpsWorks CM은 OpsWorks CM 관리자가 OpsWorks CM 서버 및 백업을 생성, 관리 및 삭제할 수 있도록 허용하는 관리형 정책을 업데이트했습니다. 이 변경으로 인해 Secrets Manager 암호의 리소스 이름에 `opsworks-cm!` 이 추가되어 OpsWorks CM이 암호를 소유할 수 있게 되었습니다.	2021년 4월 23일
OpsWorks CM에서 변경 내용 추적 시작	OpsWorks CM은 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 4월 23일

의 교차 서비스 혼동된 대리자 방지 AWS OpsWorks CM

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 위장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스

권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS OpsWorks CM 제공하는 권한을 제한하는 것이 좋습니다. 만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 전역 조건 컨텍스트 키를 모두 사용해야 합니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 aws:SourceArn 값을 모두 사용하는 경우, aws:SourceAccount 값 및 aws:SourceArn 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn를 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

aws:SourceArn의 값은 OpsWorks CM Chef 또는 Puppet 서버의 ARN이어야 합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 AWS OpsWorks CM 서버의 전체 ARN이 포함된 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 전체 ARN을 모를 경우 또는 여러 서버 ARN을 지정하는 경우 ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예: arn:aws:*servicename*:*:*123456789012*:*.

다음 섹션에서는의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 AWS OpsWorks CM 방지하는 방법을 보여줍니다.

에서 혼동된 대리자 악용 방지 AWS OpsWorks CM

이 섹션에서는 혼동된 대리자 악용을 방지하는 방법을 설명하고 액세스에 사용하는 IAM 역할에 연결할 수 있는 권한 정책의 예를 AWS OpsWorks CM포함합니다 AWS OpsWorks CM. 최상의 보안을 위해, IAM 역할이 다른 서비스와 가지는 신뢰 관계에 aws:SourceArn 및 aws:SourceAccount 조건 키를 추가하는 것이 좋습니다. 신뢰 관계를 통해는 역할을 AWS OpsWorks CM 수임하여 AWS OpsWorks CM 서버를 생성하거나 관리하는 데 필요한 다른 서비스에서 작업을 수행할 수 있습니다.

신뢰 관계를 편집하여 **aws:SourceArn** 및 **aws:SourceAccount** 조건 키를 추가하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 검색 상자에서 액세스에 사용하는 역할을 검색합니다 AWS OpsWorks CM. AWS 관리형 역할은 입니다aws-opsworks-cm-service-role.

4. 역할의 요약 페이지에서 신뢰 관계 탭을 선택합니다.
5. 신뢰 관계 탭에서 신뢰 관계 편집을 선택합니다.
6. 정책 문서에서 aws:SourceArn 또는 aws:SourceAccount 조건 키 중 하나 이상을 정책에 추가합니다. aws:SourceArn를 사용하여 교차 서비스(예: AWS Certificate Manager 및 Amazon EC2)와 특정 AWS OpsWorks CM 서버 간의 신뢰 관계를 제한 AWS OpsWorks CM 합니다. 이는 더 제한적입니다. aws:SourceAccount를 추가하여 교차 서비스와 특정 계정의 서버 간의 신뢰 관계를 제한 AWS OpsWorks CM 합니다. 이는 덜 제한적입니다. 다음은 예입니다. 두 조건 키를 모두 사용하는 경우 계정 ID가 동일해야 한다는 점에 유의하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-
opsworks-server/EXAMPLEabcd-1234-efghEXAMPLE-ID"
        }
      }
    }
  ]
}
```

7. 조건 키 추가를 완료했으면 신뢰 정책 업데이트를 선택합니다.

다음은 aws:SourceArn 및를 사용하여 AWS OpsWorks CM 서버에 대한 액세스를 제한하는 역할의 추가 예입니다aws:SourceAccount.

주제

- 예: 특정 리전의 AWS OpsWorks CM 서버 액세스
- 예: 두 개 이상의 서버 ARN을 aws:SourceArn에 추가

예: 특정 리전의 AWS OpsWorks CM 서버 액세스

다음 역할 신뢰 관계 문은 미국 동부(오하이오) 리전()의 모든 AWS OpsWorks CM 서버에 액세스합니다us-east-2. 지역은 aws:SourceArn의 ARN 값으로 지정되지만 서버 ID 값은 와일드카드 (*) 라는 점에 유의하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/*"
        }
      }
    }
  ]
}
```

예: 두 개 이상의 서버 ARN을 **aws:SourceArn**에 추가

다음 예제에서는 계정 ID 123456789012의 두 AWS OpsWorks CM 서버 배열에 대한 액세스를 제한합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-chef-
server/unique_ID",
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-puppet-
server/unique_ID"
           ]
       }
      }
    }
  ]
}
```

인터네트워크 트래픽 개인 정보

AWS OpsWorks CM은 AWS HTTPS 또는 TLS 암호화를 사용하는 HTTP에서 일반적으로 사용하는 것과 동일한 전송 보안 프로토콜을 사용합니다.

AWS OpsWorks CM에서 로깅 및 모니터링

AWS OpsWorks CM은 모든 API 작업을 CloudTrail에 기록합니다. 자세한 정보는 다음의 주제를 참조하세요.

- 를 사용하여 OpsWorks for Puppet Enterprise API 호출 로깅 AWS CloudTrail
- 를 사용하여 AWS OpsWorks for Chef Automate API 호출 로깅 AWS CloudTrail

AWS OpsWorks CM에 대한 규정 준수 확인

AWS OpsWorks CM은 다음과 같은 규정 준수 프로그램 및 규정을 지원합니다.

- 신용카드 업계(PCI)
- HIPAA(미국 건강 보험 양도 및 책임에 관한 법)
- AWS 시스템 및 조직 제어(SOC) 1, 2 및 3
- 일반 데이터 보호 규정(GDPR)

타사 감사자는 여러 규정 준수 프로그램의 일환으로 AWS OpsWorks CM의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 AWS Artifact 트의 보고서 다운로드를 참조하세요.

AWS OpsWorks CM 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수 백서 설계 -이 백서에서는 기업이를 사용하여 HIPAA 준수 애플리케이션을 AWS 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 업계 및 위치에 적용될 수 있습니다.
- AWS Config -이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

AWS OpsWorks CM의 복원력

AWS OpsWorks CM은 서버를 생성할 때 기본적으로 서버의 일일 백업을 활성화합니다. 백업은 암호화되어 Amazon S3 버킷에 저장됩니다. 기본적으로 이 버킷은 서버를 생성한 계정에서만 액세스할 수 있습니다. 다른 사용자 계정에 대한 버킷 액세스를 추가하거나 재량에 따라 Amazon S3에 교차 리전 백업을 구성할 수 있습니다. Chef 및 Puppet 두 제품 모두 AWS OpsWorks CM 서버와 관리형 노드 간의 트래픽을 암호화하므로 교차 지역 암호화를 지원합니다.

AWS OpsWorks CM은 고가용성(HA) 구성을 지원하지 않습니다.

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 기반으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이는 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS OpsWorks CM에서 서버를 백업하고 복원하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- OpsWorks for Puppet Enterprise 서버 백업 및 복원
- AWS OpsWorks for Chef Automate 서버 백업 및 복원

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS OpsWorks CM의 인프라 보안

관리형 서비스인 AWS OpsWorks 구성 관리는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 OpsWorks CM에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

AWS OpsWorks CM은 프라이빗 링크 또는 VPC 프라이빗 엔드포인트를 지원하지 않습니다.

AWS OpsWorks CM은 리소스 기반 정책을 지원하지 않습니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM으로 작업하는AWS 서비스를 참조하세요.

AWS OpsWorks CM의 구성 및 취약성 분석

AWS OpsWorks CM은 AWS OpsWorks CM 서버에서 실행 중인 운영 체제에 대해 정기적인 커널 및 보안 업데이트를 수행합니다. 사용자는 현재 날짜로부터 최대 2주 동안 자동 업데이트가 발생할 기간을 설정할 수 있습니다. AWS OpsWorks CM은 Chef 및 Puppet Enterprise 마이너 버전의 자동 업데이트를 푸시합니다. 에 대한 업데이트 구성에 대한 자세한 내용은이 설명서의 시스템 유지 관리(Chef)를

AWS OpsWorks for Chef Automate참조하세요. OpsWorks for Puppet Enterprise에 대한 업데이트 구성에 대한 자세한 내용은 이 안내서의 시스템 유지 관리(Puppet)를 참조하세요.

AWS OpsWorks CM의 보안 모범 사례

AWS OpsWorks CM은 모든 AWS 서비스와 마찬가지로 자체 보안 정책을 개발하고 구현할 때 고려해야 할 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주세요.

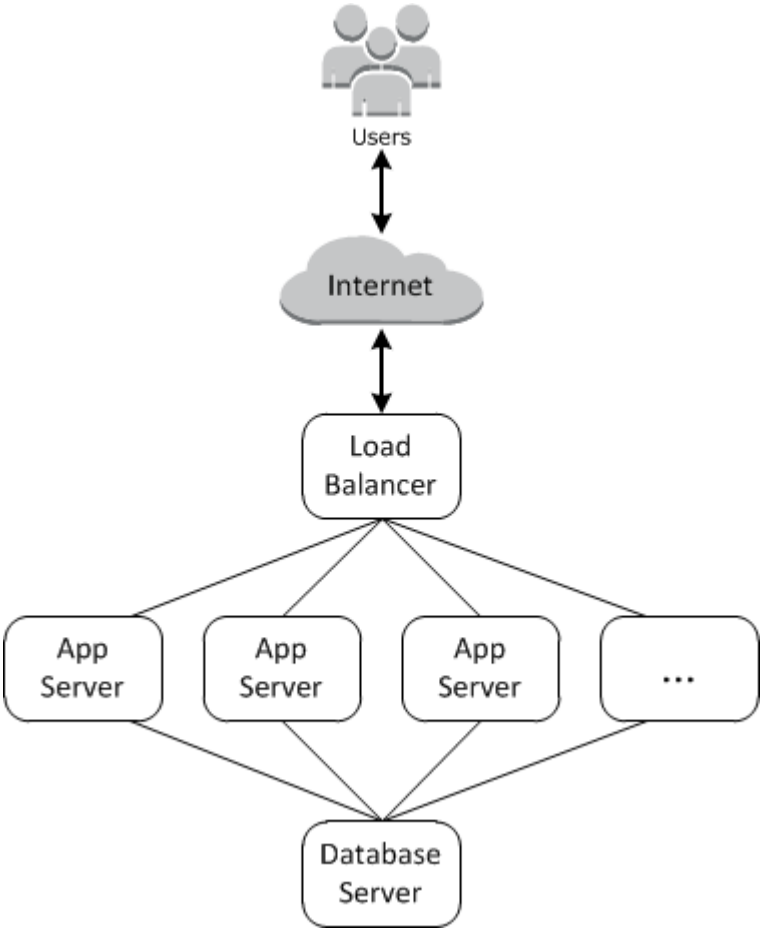
- 스타터 키트와 다운로드한 로그인 자격 증명을 보호합니다. 새 AWS OpsWorks CM 서버를 생성하거나 AWS OpsWorks CM 콘솔에서 새 Starter Kit 및 자격 증명을 다운로드할 때 이러한 항목을 하나 이상의 인증 요소가 필요한 안전한 위치에 저장합니다. 자격 증명은 서버에 대한 관리자 수준의 액세스를 제공합니다.
- 구성 코드를 보호합니다. 소스 리포지토리에 권장되는 프로토콜을 사용하여 Chef 또는 Puppet 구성 코드(Cookbook 및 모듈)를 보호합니다. 예를 들어 AWS CodeCommit의 리포지토리로 권한을 제한하거나 GitHub 웹 사이트의 지침에 따라 GitHub 리포지토리를 보호할 수 있습니다.
- CA가 서명한 인증서를 사용하여 노드에 연결합니다. AWS OpsWorks CM 서버에 노드를 등록하거나 부트스트래핑할 때 자체 서명된 인증서를 사용할 수 있지만 모범 사례로 CA 서명 인증서를 사용합니다. CA(인증 기관)에서 서명한 인증서를 사용하는 것이 좋습니다.
- Chef 또는 Puppet 관리 콘솔 로그인 자격 증명을 다른 사용자와 공유하지 마십시오. 관리자는 Chef 또는 Puppet 콘솔 웹 사이트의 각 사용자에 대해 별도의 사용자를 만들어야 합니다.
  - Chef Automate에서 사용자 관리
  - Puppet Enterprise에서 사용자 관리
- 자동 백업 및 시스템 유지 관리 업데이트를 구성합니다. AWS OpsWorks CM 서버에서 자동 유지 관리 업데이트를 구성하면 서버에서 최신 보안 관련 운영 체제 업데이트를 실행하는 데 도움이 됩니다. 자동 백업을 구성하면 재해 복구를 용이하게 하고 사고 또는 장애 발생 시 복원 시간을 단축할 수 있습니다. AWS OpsWorks CM 서버 백업을 저장하는 Amazon S3 버킷에 대한 액세스를 제한합니다. 모든 사용자에게 액세스 권한을 부여하지 마십시오. 필요에 따라 다른 사용자에게 읽기 또는 쓰기 액세스 권한을 개별적으로 부여하거나, IAM에서 해당 사용자에 대한 보안 그룹을 생성하고 보안 그룹에 액세스 권한을 할당합니다.
  - 시스템 유지 관리(Chef)
  - 시스템 유지 관리(Puppet)
  - AWS OpsWorks for Chef Automate 서버 백업 및 복원
  - OpsWorks for Puppet Enterprise 서버 백업 및 복원

- AWS Identity and Access Management 사용 설명서의 첫 번째 IAM 위임 사용자 및 사용자 그룹 생성
- Amazon 심플 스토리지 서비스 개발자 가이드의 Amazon S3 보안 모범 사례

AWS OpsWorks 스택

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

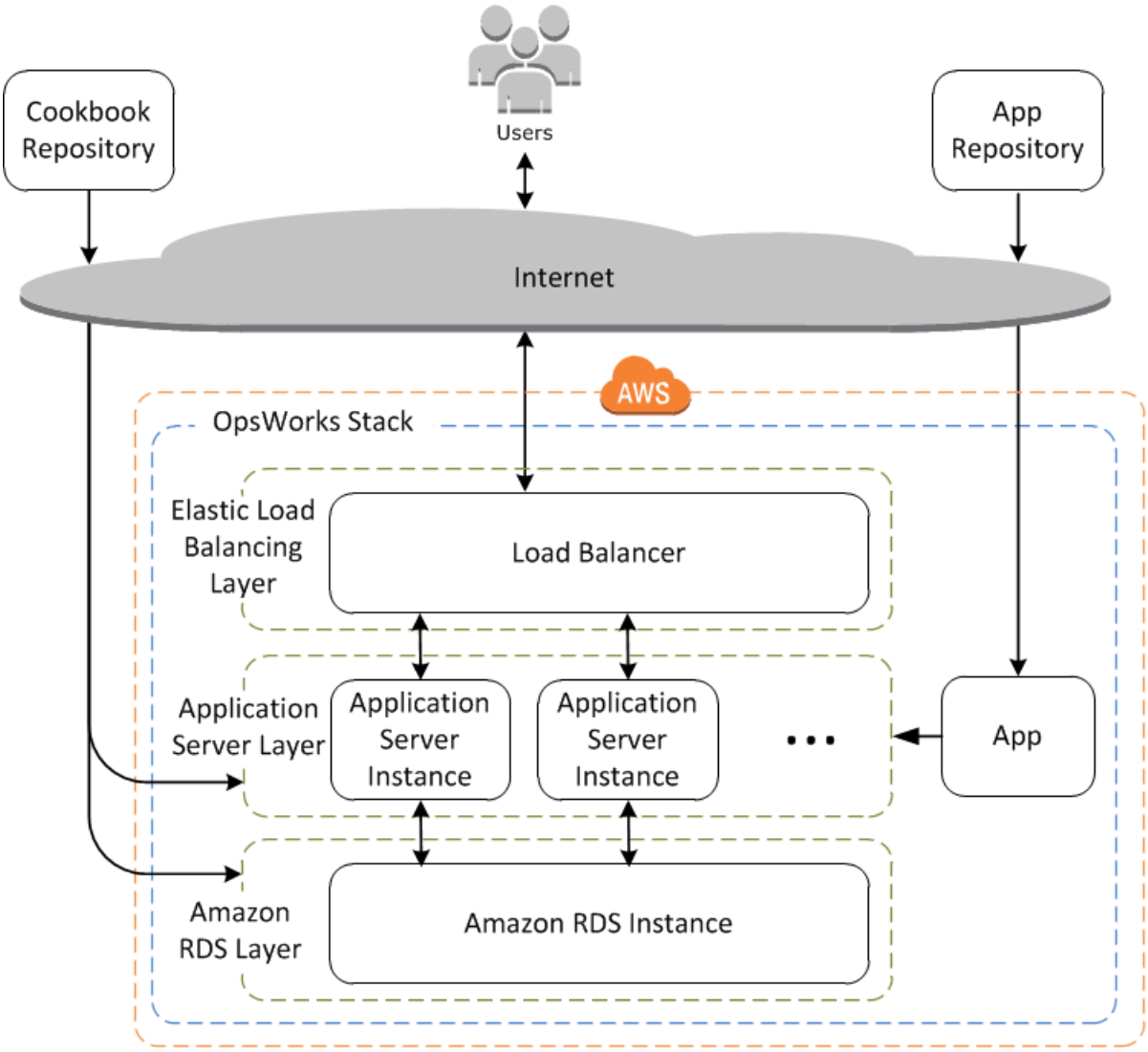
클라우드 기반 컴퓨팅에는 일반적으로 Amazon EC2 인스턴스와 Amazon Relational Database Service(RDS) 인스턴스 같은 AWS 리소스 그룹이 필포함되며, 이들은 집단적으로 생성하고 관리해야 합니다. 예를 들어 웹 애플리케이션에는 일반적으로 애플리케이션 서버, 데이터베이스 서버, 로드 밸런서 등등이 필요합니다. 이 인스턴스 그룹을 보통 스택이라고 하는데, 간단한 애플리케이션 서버 스택은 다음과 같이 보입니다.



인스턴스를 생성하고 필요한 패키지를 설치하는 것 외에도 일반적으로 애플리케이션 서버에 애플리케이션 분산, 스택 성능 모니터링, 보안 및 권한 관리 등을 수행할 방법이 필요합니다.

AWS OpsWorks Stacks는 스택과 애플리케이션을 생성하고 관리하는 간단하고 유연한 방법을 제공합니다.

다음은 AWS OpsWorks Stacks에서 기본 애플리케이션 서버 스택이 어떻게 보일 수 있는지 보여줍니다. Elastic Load Balancing 로드 밸런서 뒤에서 실행되는 애플리케이션 서버 그룹과 백엔드 Amazon RDS 데이터베이스 서버로 구성됩니다.



비교적 단순하지만이 스택에는 모든 주요 AWS OpsWorks Stacks 기능이 표시됩니다. 이 기능들이 조합되는 방식은 다음과 같습니다.

주제

- 스택

- 계층
- 레시피 및 수명 주기 이벤트
- 인스턴스
- 앱
- 스택 사용자 지정
- 리소스 관리
- 보안 및 권한
- 모니터링 및 로깅
- CLI, SDK 및 AWS CloudFormation 템플릿
- AWS OpsWorks Stacks 수명 종료 FAQs
- AWS OpsWorks Stacks 애플리케이션을 AWS Systems Manager Application Manager로 마이그레이션
- AWS OpsWorks Stacks Detach in Place 도구 사용
- AWS OpsWorks Stacks 시작하기
- AWS OpsWorks Stacks 모범 사례
- 스택
- 계층
- 인스턴스
- 앱
- 쿡북과 레시피
- 리소스 관리
- Tags
- 모니터링
- 보안 및 권한
- AWS OpsWorks Chef 12 Linux에 대한 Stacks 지원
- AWS OpsWorks Stacks에서 이전 Chef 버전 지원
- 다른 AWS 서비스와 함께 AWS OpsWorks Stacks 사용
- AWS OpsWorks Stacks CLI 사용
- 디버깅 및 문제 해결 안내서
- AWS OpsWorks Stacks 에이전트 CLI
- AWS OpsWorks Stacks 데이터 백 참조

- OpsWorks 에이전트 변경 사항

스택

스택은 코어 AWS OpsWorks Stacks 구성 요소입니다. 기본적으로 Amazon EC2 인스턴스, Amazon RDS 데이터베이스 인스턴스 등 공통의 목적을 갖고 함께 논리적으로 관리해야 하는 AWS 리소스를 위한 컨테이너입니다. 스택은 이러한 리소스를 그룹으로 관리하는 데 도움이 되며, 인스턴스의 운영 체제와 AWS 리전 같은 일부 기본적인 구성 설정도 정의합니다. 일부 스택 구성 요소를 직접적인 사용자 상호 작용에서 격리하려면 스택을 VPC에서 실행하면 됩니다.

계층

하나 이상의 계층을 추가하여 스택의 구성 요소를 정의합니다. 계층은 애플리케이션에 서비스하거나 데이터베이스 서버를 호스팅하는 등 특정 목적에 사용되는 Amazon EC2 인스턴스 집합을 나타냅니다.

패키지의 기본 구성을 수정하고 추가 패키지 설치 등등의 작업을 수행하는 Chef 레시피를 추가하여 계층을 사용자 지정하거나 확장할 수 있습니다.

모든 스택에 대해 AWS OpsWorks Stacks에는 다음 AWS 서비스를 나타내는 서비스 계층이 포함됩니다.

- Amazon Relational Database Service
- Elastic Load Balancing
- Amazon Elastic Container Service

계층을 통해 어떤 패키지가 설치되고, 어떻게 패키지를 구성하며, 애플리케이션이 어떻게 배포되는지 등을 완전히 제어할 수 있습니다.

레시피 및 수명 주기 이벤트

계층은 인스턴스에 패키지 설치, 앱 배포, 스크립트 실행 등의 작업을 처리하기 위해 Chef 레시피에 의존합니다. 주요 AWS OpsWorks Stacks 기능 중 하나는 각 인스턴스에서 적절한 시간에 지정된 레시피 세트를 자동으로 실행하는 설정, 구성, 배포, 배포 취소 및 종료와 같은 수명 주기 이벤트 세트입니다.

각 계층에는 각각의 수명 주기 이벤트에 할당되어 해당 이벤트와 계층을 위해 다양한 작업을 처리하는 레시피 집합이 있을 수 있습니다. 예를 들어 웹 서버 계층에 속하는 인스턴스가 부팅을 완료한 후 AWS OpsWorks Stacks는 다음을 수행합니다.

1. 웹 서버 설치 및 구성과 같은 작업을 수행할 수 있는 계층의 설정 레시피를 실행합니다.
2. 리포지토리에서 인스턴스로 계층의 애플리케이션을 배포하고 서비스 재시작 같은 관련 작업을 수행하는 계층의 Deploy 레시피를 실행합니다.
3. 각 인스턴스가 필요에 따라 구성을 조정해 새 인스턴스를 수용할 수 있도록 스택의 모든 인스턴스에서 Configure 레시피를 실행합니다.

   예를 들어 로드 밸런서를 실행하는 인스턴스에서 Configure 레시피는 로드 밸런서의 구성을 수정하여 새 인스턴스를 포함시킬 수 있습니다.

인스턴스가 여러 계층에 속하는 경우 AWS OpsWorks Stacks는 각 계층에 대해 레시피를 실행하므로 PHP 애플리케이션 서버와 MySQL 데이터베이스 서버를 지원하는 인스턴스를 보유할 수 있습니다.

레시피를 구현한 경우 각 레시피를 적절한 계층 및 이벤트에 할당할 수 있으며 AWS OpsWorks Stacks는 적절한 시간에 자동으로 레시피를 실행합니다. 레시피는 언제든지 수동으로 실행할 수도 있습니다.

인스턴스

인스턴스는 Amazon EC2 인스턴스와 같은 단일 컴퓨팅 리소스를 나타냅니다. 인스턴스는 운영 체제와 크기 같은 리소스의 기본적 구성을 정의합니다. 탄력적 IP 주소 또는 Amazon EBS 볼륨 등 그 밖의 구성 설정은 인스턴스의 계층에 의해 정의됩니다. 계층의 레시피는 패키지 설치 및 구성, 앱 배포와 같은 작업을 수행하여 구성을 완료합니다.

AWS OpsWorks Stacks를 사용하여 인스턴스를 생성하고 계층에 추가할 수 있습니다. 인스턴스를 시작하면 AWS OpsWorks Stacks는 인스턴스 및 해당 계층에서 지정한 구성 설정을 사용하여 Amazon EC2 인스턴스를 시작합니다. Amazon EC2 인스턴스 부팅이 완료된 후 AWS OpsWorks Stacks는 인스턴스와 서비스 간 통신을 처리하고 수명 주기 이벤트에 응답하여 적절한 레시피를 실행하는 에이전트를 설치합니다.

AWS OpsWorks Stacks는 다음과 같은 인스턴스 유형을 지원합니다. 이러한 인스턴스 유형은 시작 및 중지되는 방식을 특징으로 합니다.

- 24/7 인스턴스는 수동으로 시작되며 중지할 때까지 실행됩니다.
- 시간 기반 인스턴스는 지정된 일별 및 주별 일정에 따라 AWS OpsWorks Stacks에서 실행됩니다.

  이 인스턴스를 통해 스택은 자동으로 인스턴스를 조정하여 예측 가능한 사용 패턴을 수용할 수 있습니다.
- 로드 기반 인스턴스는 CPU 사용률과 같은 지정된 로드 지표를 기반으로 AWS OpsWorks Stacks에 의해 자동으로 시작 및 중지됩니다.

이 인스턴스를 통해 스택은 자동으로 인스턴스 수를 조정하여 수신 트래픽의 변동을 수용할 수 있습니다. 로드 기반 인스턴스는 Linux 기반 스택에서만 사용할 수 있습니다.

AWS OpsWorks Stacks는 인스턴스 자동 복구를 지원합니다. 에이전트가 서비스와의 통신을 중지하면 AWS OpsWorks Stacks는 인스턴스를 자동으로 중지했다가 다시 시작합니다.

Linux 기반 컴퓨팅 리소스를 AWS OpsWorks Stacks 외부에서 생성된 스택에 통합할 수도 있습니다.

- Amazon EC2 콘솔, CLI 또는 API를 사용하여 직접 생성한 Amazon EC2 인스턴스.
- 가상 머신에서 실행되는 인스턴스를 비롯하여 자체 하드웨어에서 실행되는 온프레미스 인스턴스.

이러한 인스턴스 중 하나를 등록한 후에는 Stacks 인스턴스가 되며 AWS OpsWorks Stacks로 AWS OpsWorks 생성한 인스턴스와 거의 동일한 방식으로 인스턴스를 관리할 수 있습니다.

앱

애플리케이션과 관련 파일은 Amazon S3 버킷과 같은 리포지토리에 저장합니다. 각각의 애플리케이션은 애플리케이션 유형을 지정하고 리포지토리에서 인스턴스로 애플리케이션을 배포하는 데 필요한 정보(예: 리포지토리 URL 및 암호)가 포함된 앱으로 표시됩니다. 앱을 배포할 때 AWS OpsWorks Stacks는 스택의 인스턴스에서 Deploy 레시피를 실행하는 Deploy 이벤트를 트리거합니다.

앱은 다음 방법으로 배포할 수 있습니다.

- 자동 - 인스턴스를 시작하면 AWS OpsWorks Stacks가 인스턴스의 Deploy 레시피를 자동으로 실행합니다.
- 수동 - 새 앱이 있거나 기존 앱을 업데이트하려는 경우 온라인 인스턴스의 Deploy 레시피를 수동으로 실행할 수 있습니다.

일반적으로 AWS OpsWorks Stacks가 전체 스택에서 Deploy 레시피를 실행하도록 하면 다른 계층의 인스턴스가 구성을 적절하게 수정할 수 있습니다. 하지만 예컨대 새 앱을 모든 앱 서버 인스턴스에 배포하기 전에 테스트하려는 경우, 인스턴스의 하위 집합으로 배포를 제한할 수 있습니다.

스택 사용자 지정

AWS OpsWorks Stacks는 특정 요구 사항을 충족하도록 계층을 사용자 지정하는 다양한 방법을 제공합니다.

- 다양한 구성 설정을 나타내는 속성을 재정의하거나 구성 파일을 생성하는 데 사용되는 템플릿을 재정의하여 AWS OpsWorks Stacks가 패키지를 구성하는 방법을 수정할 수 있습니다.
- 자체 레시피를 제공하여 스크립트 실행이나 비표준 패키지 설치 및 구성과 같은 작업을 수행하도록 기존 계층을 확장할 수 있습니다.

모든 스택에는 최소한의 레시피 집합으로만 시작하는 하나 이상의 계층이 포함될 수 있습니다. 패키지 설치, 앱 배포 등의 작업을 처리하도록 레시피를 구현하여 계층에 기능을 추가합니다. 사용자 지정 레시피와 관련 파일을 하나 이상의 쿡북에 패키징하고 Amazon S3 또는 Git 같은 리포지토리에 쿡북을 저장합니다.

레시피를 수동으로 실행할 수 있지만 AWS OpsWorks Stacks를 사용하면 다섯 가지 수명 주기 이벤트 세트를 지원하여 프로세스를 자동화할 수도 있습니다.

- 설정은 새 인스턴스가 성공적으로 부팅된 후 새 인스턴스에서 발생합니다.
- Configure는 인스턴스가 온라인 상태에 진입하거나 온라인 상태에서 나갈 때 스택의 모든 인스턴스에서 발생합니다.
- Deploy는 앱을 배포할 때 발생합니다.
- Undeploy는 앱을 삭제할 때 발생합니다.
- Shutdown은 인스턴스를 중지할 때 발생합니다.

각 계층에서 각 이벤트에 할당할 수 있는 레시피의 수에는 제한이 없습니다. 계층의 인스턴스에서 수명 주기 이벤트가 발생하면 AWS OpsWorks Stacks는 연결된 레시피를 실행합니다. 예를 들어 앱 서버 인스턴스에서 Deploy 이벤트가 발생하면 AWS OpsWorks Stacks는 계층의 Deploy 레시피를 실행하여 앱을 다운로드하거나 관련 작업을 수행합니다.

리소스 관리

탄력적 IP 주소와 같은 다른 AWS 리소스를 스택에 통합할 수 있습니다. AWS OpsWorks Stacks 콘솔 또는 API를 사용하여 스택에 리소스를 등록하고, 등록된 리소스를 인스턴스에 연결하거나 인스턴스에서 분리하고, 한 인스턴스에서 다른 인스턴스로 리소스를 이동할 수 있습니다.

보안 및 권한

AWS OpsWorks Stacks는 AWS Identity and Access Management (IAM)과 통합되어 다음을 포함하여 사용자가 AWS OpsWorks Stacks에 액세스하는 방법을 제어하는 강력한 방법을 제공합니다.

- 계층과 인스턴스 같은 스택 리소스를 생성할 수 있는지 여부 혹은 SSH나 RDP를 사용하여 스택의 Amazon EC2 인스턴스에 연결할 수 있는지 여부 등 개별 사용자가 각 스택과 상호 작용할 수 있는 방법.
- AWS OpsWorks Stacks가 사용자를 대신하여 Amazon EC2 인스턴스와 같은 AWS 리소스와 상호 작용하는 방법.
- AWS OpsWorks Stacks 인스턴스에서 실행되는 앱이 Amazon S3 버킷과 같은 AWS 리소스에 액세스하는 방법.
- 사용자의 퍼블릭 SSH 키와 RDP 암호를 관리하고 인스턴스에 연결하는 방법.

모니터링 및 로깅

AWS OpsWorks Stacks는 스택을 모니터링하고 스택 및 레시피 관련 문제를 해결하는 데 도움이 되는 몇 가지 기능을 제공합니다. 모든 스택:

- AWS OpsWorks Stacks는 Linux 스택에 대한 사용자 지정 CloudWatch 지표 세트를 제공하며,이 지표는 사용자의 편의를 위해 모니터링 페이지에 요약되어 있습니다.

  AWS OpsWorks Stacks는 Windows 스택에 대한 표준 CloudWatch 지표를 지원합니다. 지표는 CloudWatch 콘솔로 모니터링할 수 있습니다.
- AWS 계정에서 AWS OpsWorks Stacks에 의해 또는 Stacks를 대신하여 수행된 API 호출을 기록하는 CloudTrail 로그입니다.
- 스택에서 모든 이벤트를 나열하는 이벤트 로그.
- 어떤 레시피가 실행되었고 어떤 오류가 발생했는지 등 각 인스턴스에서 각 수명 주기에 무엇이 발생했는지 상세히 기록하는 Chef 로그.

Linux 기반 스택에는 스택의 인스턴스에 대한 세부 모니터링 데이터를 수집하고 표시하는 데 사용할 수 있는 Ganglia 마스터 계층도 포함될 수 있습니다.

CLI, SDK 및 AWS CloudFormation 템플릿

콘솔 외에도 AWS OpsWorks Stacks는 모든 작업을 수행하는 데 사용할 수 있는 여러 언어에 대한 명령줄 인터페이스(CLI) 및 SDKs도 지원합니다. 다음 기능을 고려하세요.

- AWS OpsWorks Stacks CLI는 AWS CLI의 일부이며 명령줄에서 모든 작업을 수행하는 데 사용할 수 있습니다.

AWS CLI는 여러 AWS 서비스를 지원하며, Windows, Linux 또는 OS X 시스템에 설치할 수 있습니다.

- AWS OpsWorks Stacks는 Windows PowerShell용 AWS 도구에 포함되어 있으며 Windows PowerShell 명령줄에서 모든 작업을 수행하는 데 사용할 수 있습니다.
- AWS OpsWorks Stacks SDK는 Java, JavaJavaScriptScript(브라우저 기반 및 Node.js), .NET, PHP, Python(boto) 또는 Ruby에서 구현된 애플리케이션에서 사용할 수 있는 AWS SDKs에 포함되어 있습니다.

AWS CloudFormation 템플릿을 사용하여 스택을 프로비저닝할 수도 있습니다. 몇 가지 예제는 AWS OpsWorks 조각을 참조하세요.

AWS OpsWorks Stacks 수명 종료 FAQs

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다.

주제

- 이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까?
- 신규 고객을 AWS OpsWorks Stacks 수락하고 있습니까?
- 기존 스택을 어디로 마이그레이션해야 하나요?
- 수명 종료 후에도 기존 Amazon EC2 인스턴스를 유지하려면 어떻게 해야 하나요?
- 수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?
- 어떤 수준의 기술 지원을 사용할 수 있나요 AWS OpsWorks Stacks?
- 에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks Stacks?

이번 서비스 종료로 인해 기존 고객은 어떤 영향을 받게 됩니까?

기존 고객은 서비스 종료일인 2024년 5월 26일까지 영향을 받지 않습니다. AWS OpsWorks Stacks 2024년 5월 26일 이후 고객은 OpsWorks 콘솔, API, CLI, CloudFormation 리소스를 사용할 수 없게 됩니다.

신규 고객을 AWS OpsWorks Stacks 수락하고 있습니까?

아니요. AWS OpsWorks Stacks 는 더 이상 신규 고객을 받지 않으며 현재 기존 고객만 새 스택을 생성할 수 있습니다.

기존 스택을 어디로 마이그레이션해야 하나요?

AWS OpsWorks Stacks 다음 기능을 활용할 수 AWS Systems Manager 있는 로 워크로드를 마이그레이션하는 것이 좋습니다.

- 모던 Chef 버전
- SSM 에이전트
- Application Load Balancer
- Auto Scaling 을 통한 향상된 조정 기능
- EC2 시작 템플릿을 사용하여 원하는 호스트 특성을 정의할 수 있습니다.
- 최신 인스턴스 유형
- 최신 EBS 볼륨 유형

Systems Manager에 대한 자세한 내용은 AWS Systems Manager 사용 설명서를 참조하세요. 로 마이그레이션에 대한 자세한 내용은 섹션을 AWS Systems Manager참조하세요. AWS OpsWorks Stacks 애플리케이션을 AWS Systems Manager Application Manager로 마이그레이션

수명 종료 후에도 기존 Amazon EC2 인스턴스를 유지하려면 어떻게 해야 하나요?

수명 종료 날짜에 도달하면 Amazon EC2 인스턴스는 계정에 남아 있지만 더 이상 OpsWorks Stacks 서비스를 사용하여 인스턴스를 제어하고 관리할 수 없습니다.

AWS OpsWorks Stacks Detach in Place 도구를 사용하여 OpsWorks Stacks 서비스에서 OpsWorks 인스턴스를 분리할 수 있습니다. 분리 후 Amazon EC2 AWS Systems Manager또는 EC2 호환 접근 방식을 사용하여 인스턴스를 구성하고 관리할 수 있습니다. 자세한 내용은 AWS OpsWorks Stacks Detach in Place 도구 사용 단원을 참조하십시오.

수명 종료가 AWS 리전 동시에 모든에 영향을 미칩니까?

예. OpsWorks 콘솔, API, CLI 및 CloudFormation 리소스는 2024년 5월 26일에 모두 AWS 리전 동시에 중단됩니다. 를 AWS OpsWorks Stacks 사용할 수 AWS 리전 있는 위치 목록은 AWS 리전 서비스 목록을 참조하세요.

어떤 수준의 기술 지원을 사용할 수 있나요 AWS OpsWorks Stacks?

AWS 는 오늘부터 수명 종료 날짜까지 AWS OpsWorks Stacks 고객에게 동일한 수준의 지원을 계속 제공합니다. 질문이나 우려 사항이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의할 수 있습니다.

에 대한 새로운 기능 릴리스가 있나요 AWS OpsWorks Stacks?

아니요. 서비스 수명이 종료되는 시점이므로 새로운 기능은 출시되지 않을 예정입니다. 하지만 서비스 종료일까지 계속해서 보안을 개선하고 Amazon EC2 인스턴스를 예상대로 관리할 예정입니다.

AWS OpsWorks Stacks 애플리케이션을 AWS Systems Manager Application Manager로 마이그레이션

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다.

이제 마이그레이션 스크립트를 AWS Systems Manager사용하여의 기능인 Application Manager로 AWS OpsWorks Stacks 애플리케이션을 마이그레이션할 수 있습니다. Stacks 애플리케이션을 Systems Manager Application Manager로 마이그레이션하면 Graviton과 같은 새로운 Amazon EC2 인스턴스 유형 AWS OpsWorks Stacks, gp3와 같은 새로운 Amazon Elastic Block Store(EBS) 볼륨, 새로운 운영 체제, Auto Scaling 그룹과의 통합, 애플리케이션 로드 밸런서와 같이에서 사용할 수 없는 AWS 기능을 사용할 수 있습니다.

이번 릴리스부터 이제 Systems Manager Application Manager에서 사용할 수 있는 새 인스턴스 탭을 사용하여 마이그레이션된 인스턴스에서 작업을 모니터링하고 실행할 수 있습니다. 인스턴스 탭을 사용하여 한 곳에서 여러 AWS 인스턴스를 볼 수 있습니다. 이 탭을 사용하면 인스턴스 상태에 대한

정보를 보고 문제를 해결할 수 있습니다. 인스턴스 탭을 사용하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 애플리케이션 인스턴스를 사용하여 작업하기를 참조하세요.

주제

- 스크립트 작동 방식
- 사전 조건
- 제한 사항
- 시작
- FAQ
- 문제 해결

스크립트 작동 방식

AWS OpsWorks 는 CloudFormation 템플릿을 사용하여 AWS OpsWorks Stacks 애플리케이션을 Systems Manager Application Manager로 마이그레이션하기 위해 실행할 수 있는 스크립트를 제공합니다. 스크립트는 기존 OpsWorks 계층에 대한 정보를 가져오고 스크립트의 `--provision-application` 매개 변수 값에 따라 애플리케이션의 클론을 프로비저닝하거나 AWS CloudFormation 를 사용하여 수정할 수 있는 스타터 CloudFormation 템플릿을 제공합니다.

사전 조건

- AWS CLI 가 설치 및 구성되어 있는지 확인합니다. 설치에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 의 최신 버전 설치 또는 업데이트를 AWS CLI AWS CLI참조하세요.

  > **Note**
  >
  > 를 구성하지 않으려면를 사용하여 명령을 실행할 수도 AWS CLI있습니다 AWS CloudShell. CloudShell을 사용하는 방법에 대한 자세한 내용은 AWS CloudShell 사용 설명서의 AWS CloudShell사용을 참조하세요.

- Python 버전 3.6 이상이 설치되어 있거나 Amazon Machine Image(AMI)와 함께 제공되는지 확인하세요.
- 운영 체제가 지원되는지 확인하세요. 다음 운영 체제에서 마이그레이션 스크립트를 다운로드하고 실행할 수 있습니다.
  - Amazon Linux and Amazon Linux 2

- Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS
- Red Hat Enterprise Linux 8
- Windows Server 2019, Windows 10 Enterprise

> **Note**
>
> Windows Server 2022는 지원되지 않습니다.

제한 사항

새로운 OpsWorks 아키텍처는 아키텍처와 다릅니다 AWS OpsWorks Stacks. 이 섹션에서는 이 아키텍처의 알려진 제한 사항에 대해 설명합니다.

다음은 새 OpsWorks 아키텍처에서 지원되지 않습니다.

- Windows 및 CentOS 인스턴스에서 Chef 레시피 실행
- 내장 Chef 11 계층 및 Berkshelf
- Chef 속성 및 데이터 백
- 온프레미스 인스턴스
- EC2에서 가져온 인스턴스
- 사용자 지정 운영 체제 패키지 목록 설치 지원 안 됨
- 앱은 지원되거나 마이그레이션되지 않습니다.

다음은 제한 사항과 함께 지원됩니다.

- 마이그레이션 스크립트는 EBS 볼륨 정보를 복제하지만 볼륨에 포함된 마운트 지점과 실제 데이터는 제외합니다.
- 시간 기반 및 로드 기반 크기 조정 인스턴스는 마이그레이션되지만 이러한 인스턴스와 관련된 조정 규칙은 마이그레이션되지 않습니다. Auto Scaling 을 수정하여 비슷한 결과를 얻을 수 있습니다.
- OpsWorks 콘솔의 스택 권한 페이지에 정의된 IAM 엔티티는 만들어지거나 생성되지 않습니다.
- 마이그레이션 스크립트는 Systems Manager에서 단일 계층 애플리케이션만 프로비저닝할 수 있습니다. 예를 들어 동일한 스택의 두 계층에 대해 스크립트를 두 번 실행하면 Systems Manager에서 서로 다른 두 개의 애플리케이션을 얻게 됩니다.

시작

마이그레이션 스크립트인 `stack_exporter.py`는 로컬에서 또는 EC2 인스턴스에서 실행할 수 있는 Python 스크립트입니다. 스크립트를 실행하기 전에 모든 사전 요구 사항이 충족되었는지 확인하세요. 필수 조건에 대한 자세한 사항은 사전 조건에서 확인하세요.

다음 섹션의 단계는 OpsWorks 스택을 Systems Manager 애플리케이션 관리자로 마이그레이션하는 방법을 보여줍니다.

주제

- 1단계: 스크립트 실행을 위한 환경 준비
- 2단계: 마이그레이션 스크립트 다운로드
- 3단계: 스크립트를 실행할 환경 설정
- 4단계: 스크립트 실행
- 5단계: CloudFormation 스택 프로비저닝
- 6단계: 프로비저닝된 리소스 검토
- 7단계: 인스턴스 시작
- 8단계: 인스턴스 검토
- 9단계: Systems Manager 애플리케이션 관리자를 사용하여 인스턴스에서 작업을 모니터링하고 실행합니다.

1단계: 스크립트 실행을 위한 환경 준비

운영 체제에 적합한 명령을 실행하여 환경을 준비합니다.

주제

- Amazon Linux 2
- Amazon Linux
- Ubuntu 18.04, 20.04, 22.04
- Red Hat Enterprise Linux 8
- Windows Server 2019, Windows 10 Enterprise

Amazon Linux 2

```
sudo su
```

```
python3 -m pip install pipenv
PATH="$PATH:/usr/local/bin"
yum update
yum install git
```

Amazon Linux

```
sudo su
PATH="$PATH:/usr/local/bin"
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
yum update
yum list | grep python3
yum install python36 // Any python version
yum install git
```

Python 버전 3.6의 경우 다음 명령도 실행하세요.

```
python3 -m pip install pipenv==2022.4.8
```

Python 버전 3.7 이상에서는 다음 명령도 실행합니다.

```
python3 -m pip install pipenv
```

Ubuntu 18.04, 20.04, 22.04

```
sudo su
export PATH="${HOME}/.local/bin:$PATH"
apt-get update
apt install python3-pip
apt-get install git // if git is not installed
python3 -m pip install --user pipenv==2022.4.8
```

Red Hat Enterprise Linux 8

```
sudo su
sudo dnf install python3
PATH="$PATH:/usr/local/bin"
yum update
yum install git
python3 -m pip install pipenv==2022.4.8
```

Windows Server 2019, Windows 10 Enterprise

> **Note**
>
> Windows Server 2019의 경우 Python 버전 3.6.1 이상을 설치합니다.

```
pip install pipenv
```

아직 설치되지 않은 경우 Git를 다운로드하여 설치합니다.

Git을 쿡북 소스로 사용하는 경우 Windows에서 스크립트를 실행하기 전에 Git 서버를 known_hosts 파일에 추가하세요. PowerShell을 사용하여 다음과 같은 함수를 생성할 수 있습니다.

```
function add_to_known_hosts($server){
    $new_host=$(ssh-keyscan $server 2> $null)
    $existing_hosts=''
    if (!(test-path "$env:userprofile\.ssh")) {
        md "$env:userprofile\.ssh"
    }
    if ((test-path "$env:userprofile\.ssh\known_hosts")) {
        $existing_hosts=Get-Content "$env:userprofile\.ssh\known_hosts"
    }
    $host_added=0
    foreach ($line in $new_host) {
        if (!($existing_hosts -contains $line)) {
            Add-Content -Path "$env:userprofile\.ssh\known_hosts" -Value $line
            $host_added=1
    }
   }
   if ($host_added) {
       echo "$server has been added to known_hosts."
   } else {
       echo "$server already exists in known_hosts."
   }
}
```

그런 다음 함수를 실행할 때 Git 서버(예: github.com, git-codecommit.*repository_region*.amazonaws.com)를 제공할 수 있습니다.

```
add_to_known_hosts "myGitServer"
```

2단계: 마이그레이션 스크립트 다운로드

다음 명령을 실행하여 마이그레이션 스크립트와 모든 관련 파일이 포함된 zip 파일을 다운로드합니다.

```
aws s3api get-object \
    --bucket export-opsworks-stacks-bucket-prod-us-east-1 \
    --key export_opsworks_stacks_script.zip export_opsworks_stacks_script.zip
```

Linux를 사용하는 경우 다음 명령을 사용하여 압축 해제 유틸리티를 설치합니다.

```
sudo apt-get install unzip
sudo yum install unzip
```

운영 체제에 맞는 명령을 사용하여 파일을 압축 해제합니다.

Linux의 경우 다음 명령을 사용합니다.

```
unzip export_opsworks_stacks_script.zip
```

Windows의 경우 PowerShell의 `Expand-Archive` 명령을 사용합니다.

```
Expand-Archive -LiteralPath PathToZipFile -DestinationPath PathToDestination
```

파일의 압축을 풀고 나면 다음과 같은 디렉토리와 파일을 사용할 수 있습니다.

- README.md
- 라이선스
- INFO
- requirements.txt
- 템플릿/
  - OpsWorksCFNTemplate.yaml
  - MountEBSVolumes.yaml
- opsworks/
- cloudformation/
- instances_tab/
- cfn_stack_deployer.py
- s3.py

- stack_exporter_context.py
- stack_exporter.py

3단계: 스크립트를 실행할 환경 설정

다음 명령을 사용하여 스크립트를 실행할 환경을 설정합니다.

```
pipenv install -r requirements.txt
pipenv shell
```

> **Note**
>
> 현재 스크립트는 Application Manager에서 단일 계층 애플리케이션만 프로비저닝할 수 있습니다. 예를 들어 동일한 스택의 두 계층에 대해 스크립트를 두 번 실행하면 이 스크립트는 Application Manager에서 서로 다른 두 개의 애플리케이션을 만듭니다.

환경을 설정한 후 스크립트 파라미터를 검토하세요. `python3 stack_exporter.py --help`명령을 실행하여 마이그레이션 스크립트에 사용할 수 있는 옵션을 볼 수 있습니다.

파라미터	설명	필수	유형	기본값
`--layer-id`	이 OpsWorks 계층 ID에 대한 CloudFormation 템플릿을 내보냅니다.	예	문자열	
`--region`	OpsWorks 스택의 AWS 리전입니다. OpsWorks 스택 리전과 API 엔드포인트 리전이 다른 경우 스택 리전을 사용하세요. OpsWorks 스택의 다른 리소스 부분(예: EC2 인스턴스 및 서브넷)과 동일한 리전입니다.	아니요	문자열	us-east-1
`--provision-`	기본적으로 스크립트는 CloudFormation 템플릿에서 내보낸 애플리케이션을 프로비저	아니요	불	TRUE

파라미터	설명	필수	유형	기본값
`application`	닝합니다. 이 파라미터를 값이 FALSE인 스크립트에 전달하면 CloudFormation 템플릿의 프로비저닝을 건너뛸 수 있습니다.			
`--launch-template`	이 파라미터는 기존 시작 템플릿을 사용할지 또는 새 시작 템플릿을 생성할지 정의합니다. 권장 인스턴스 속성을 사용하거나 온라인 인스턴스와 일치하는 인스턴스 속성을 사용하는 새 시작 템플릿을 만들 수 있습니다. 유효한 값으로는 다음이 포함됩니다. • `RECOMMENDED` - OpsWorks 스택의 OS 및 c5.large 인스턴스 크기에 대해 최신 AMI의 인스턴스 특성을 사용합니다. • `MATCH_LAST_INSTANCE` - 사용 가능한 최신 온라인 인스턴스 특성을 사용합니다. • *`LaunchTemplateID`* / [*`LaunchTemplateVersion]`* - 기존 시작 템플릿을 사용합니다. 경우에 따라 템플릿 버전을 제공할 수 있습니다. 템플릿 버전을 제공하지 않는 경우 스크립트는 기본 버전을 사용합니다.	아니요	문자열	`RECOMMENDED`

파라미터	설명	필수	유형	기본값
--system-updates	인스턴스 부팅 시 커널 및 패키지 업데이트 수행 여부를 정의합니다. 유효한 값으로는 다음이 포함됩니다. • ALL_UPDATES - 인스턴스 부팅 시 커널 및 패키지에 대한 시스템 업데이트를 수행합니다. • NO_UPDATES - 인스턴스 부팅 시 시스템 업데이트를 수행하지 않습니다. • MATCH_LAYER_SETTINGS - OpsWorks 계층 또는 인스턴스의 InstallUpdatesOnBoot 속성을 사용하여 시스템 업데이트 설치 여부를 결정합니다.	아니요	문자열	ALL_UPDATES
--http-username	사용자 지정 쿡북이 포함된 HTTP 아카이브를 인증하는 데 사용되는 사용자 이름을 저장하는 Systems Manager SecureString 매개변수의 이름입니다.	아니요	문자열	
--http-password	사용자 지정 쿡북이 포함된 HTTP 아카이브를 인증하는 데 사용되는 비밀번호를 저장하는 Systems Manager SecureString 매개변수의 이름입니다.	아니요	문자열	

파라미터	설명	필수	유형	기본값
`--repo-private-key`	사용자 지정 쿡북이 포함된 저장소를 인증하는 데 사용되는 SSH 키를 저장하는 Systems Manager `SecureString` 매개변수의 이름입니다. 리포지토리가 GitHub에 있는 경우 새 `Ed25519` SSH 키를 생성해야 합니다. 새 `Ed25519` SSH 키를 생성하지 않으면 GitHub 리포지토리에 대한 연결이 실패합니다.	아니요	문자열	
`--lb-type`	기존 로드 밸런서를 마이그레이션할 때 생성할 로드 밸런서의 유형(있는 경우) 유효한 값으로는 다음이 포함됩니다. • `ALB` (Application Load Balancer) • `Classis` (Classic Load Balancer) • `None` (로드 밸런서를 만들고 싶지 않은 경우)	아니요	문자열	`ALB`
`--lb-access-logs-path`	기존 S3 버킷의 경로와 로드 밸런서 액세스 로그를 저장하기 위한 접두사. S3 버킷 및 로드 밸런서는 동일한 리전에 있어야 합니다. 값을 제공하지 않고 `--lb-type` 파라미터 값이 `None`로 설정된 경우 스크립트는 새 S3 버킷과 접두사를 생성합니다. 이 접두사에 적합한 버킷 정책이 있는지 확인하세요.	아니요	문자열	

파라미터	설명	필수	유형	기본값
`--enable-instance-protection`	TRUE로 설정하면 스크립트가 Auto Scaling에 대한 사용자 지정 종료 정책(Lambda 함수)을 생성합니다. `protected_instance` 태그가 있는 EC2 인스턴스는 스케일 인 이벤트로부터 보호됩니다. 스케일 인 이벤트로부터 보호하려는 각 EC2 인스턴스에 `protected_instance` 태그를 추가합니다.	아니요	불	FALSE
`--command-logs-bucket`	AWS ApplyChefRecipe 및 MountEBSVolumes 로그를 저장할 기존 S3 버킷의 이름입니다. 값을 제공하지 않으면 스크립트는 새 S3 버킷을 생성합니다.	아니요	문자열	`aws-opsworks-application-manager-logs-`*account-id*
`--custom-json-bucket`	사용자 지정 JSON을 저장할 기존 S3 버킷의 이름입니다. 값을 제공하지 않으면 스크립트는 새 S3 버킷을 생성합니다.	아니요	문자열	`aws-apply-chef-application-manager-transition-data-`*account-id*

참고:

- 프라이빗 GitHub 리포지토리를 사용하는 경우 SSH용 새 Ed25519 호스트 키를 생성해야 합니다. 이는 GitHub가 SSH에서 지원되는 키를 변경하고 암호화되지 않은 Git 프로토콜을 제거했기 때문입니다. Ed25519 호스트 키에 대한 자세한 내용은 GitHub 블로그 게시물 GitHub의 Git 프로토콜 보안 개선을 참조하세요. 새 Ed25519 호스트 키를 생성한 후 SSH 키에 대한 Systems Manager SecureString 매개변수를 생성하고 SecureString 매개변수 이름을 `--repo-private-key` 매개변수 값으로 사용합니다. Systems Manager SecureString 파라미터 생성 방법에 대한 자

세한 내용은 AWS Systems Manager 사용 설명서의 SecureString 파라미터 (AWS CLI) 생성 또는 Systems Manager 파라미터 생성(콘솔)을 참조하세요.

- `--http-username`, `--http-password` 및 `--repo-private-key` 파라미터는 Systems Manager `SecureString` 파라미터의 이름입니다. 마이그레이션 스크립트는 `AWS-ApplyChefRecipes` 문서를 실행할 때 이러한 매개 변수를 사용합니다.
- `--http-username` 파라미터의 값은 `--http-password` 파라미터의 값도 지정해야 합니다.
- `--http-username` 파라미터의 값은 `--http-password` 파라미터의 값도 지정해야 합니다.
- `--http-password`과 `--repo-private-key` 모두에 값을 설정하지 마세요. SSH 키 (`--repo-private-key`)의 Systems Manager `SecureString` 매개 변수 이름 또는 리포지토리 사용자 이름 (`--http-username`) 및 암호(`--http-password`)를 제공하세요.

4단계: 스크립트 실행

`python3 stack_exporter.py`를 실행하면 애플리케이션을 프로비저닝하거나 `--provision-application` 매개변수 값을 `FALSE`로 설정하여 시작 템플릿을 생성할 수 있습니다.

예 1: Systems Manager 애플리케이션 관리자 애플리케이션 프로비저닝

다음 명령은 기존 OpsWorks 계층에 대한 정보를 가져오고 최신 OpsWorks 아키텍처를 사용하여 애플리케이션을 프로비저닝합니다. 그러면 스택에 구성된 Chef 버전과 유사한 결과를 얻을 수 있습니다. 스크립트는 CloudFormation을 사용하여 Auto Scaling 과 같은 모든 필수 리소스를 프로비저닝한 다음 Systems Manager 애플리케이션 관리자에 애플리케이션을 등록합니다.

및 *## ID*를 OpsWorks 스택 및 계층의 값으로 바꿉니다.

```
python3 stack_exporter.py \
     --layer-id layer-id \
     --region stack-region
```

예 2: 템플릿 생성

다음 명령은 기존 OpsWorks 계층에 대한 정보를 가져오고 CloudFormation 템플릿을 생성합니다. 템플릿을 프로비저닝하면 Chef 14를 사용하는 것과 비슷한 결과를 얻을 수 있습니다. 이 예에서는 `--provision-application` 파라미터가 `FALSE`로 설정되어 있기 때문에 리소스가 프로비저닝되지 않습니다.

및 *## ID*를 OpsWorks 스택 및 계층의 값으로 바꿉니다.

```
python3 stack_exporter.py \
```

```
    --layer-id layer-id \
    --region stack-region \
    --provision-application FALSE
```

명령을 실행한 후 Systems Manager의 Application Manager 템플릿 라이브러리에서 템플릿을 검토하고 템플릿을 프로비저닝할 수도 있습니다. 템플릿 라이브러리 보기에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 템플릿 라이브러리 사용을 참조하세요.

5단계: CloudFormation 스택 프로비저닝

> **Note**
>
> 스크립트의 `--provision-application` 파라미터를 `FALSE`로 설정한 경우에만 이 단계를 완료하면 됩니다.

값이 `FALSE`인 `--provision-application` 파라미터를 지정하면 스크립트 출력에 CloudFormation 템플릿의 이름과 URL이 제공됩니다. 이 템플릿은 기존 OpsWorks 스택 및 계층의 대체 제안 양식을 나타냅니다.

애플리케이션 관리자 템플릿 라이브러리(권장)를 사용하거나 CloudFormation을 사용하여 템플릿을 프로비저닝할 수 있습니다. 템플릿 라이브러리 사용에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 템플릿 라이브러리 사용을 참조하세요.

6단계: 프로비저닝된 리소스 검토

이제 프로비저닝된 리소스를 검토할 준비가 되었습니다.

1. AWS CloudFormation 콘솔을 사용하여 프로비저닝된 스택의 리소스를 검토합니다.
   a. https://console.aws.amazon.com/cloudformation AWS CloudFormation 콘솔을 열고 스택을 선택합니다.
   b. 스택 페이지에서 스택을 선택한 다음 리소스 탭을 선택합니다.
   c. 리소스 탭에서 스택에 대해 나열된 스택의 리소스를 검토하세요. 리소스 목록에는 Auto Scaling 콘솔에서 검토할 수 있는 EC2 Auto Scaling 그룹이 포함되어 있습니다 AWS CLI. 또는
2. Systems Manager 애플리케이션 관리자를 사용하여 애플리케이션에 대한 리소스를 검토하세요.
   a. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.

b. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.

c. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다. 애플리케이션 관리자가 개요 탭을 엽니다.

d. 리소스 탭을 선택합니다. 리소스 탭에는 OpsWorks 스택 및 계층용으로 마이그레이션된 모든 리소스가 표시됩니다. 애플리케이션 이름은 OpsWorks 스택의 이름을 포함하며 *app-stack-name-suffix* 형식으로 지정됩니다. 여기서 *###*는 스택 ID의 처음 6자를 나타냅니다. Application Manager에서 리소스를 보는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 애플리케이션 리소스 보기를 참조하세요.

7단계: 인스턴스 시작

인스턴스를 프로비저닝하고 나면 인스턴스를 테스트할 준비가 된 것입니다. 현재로서는 실행 중인 인스턴스가 없습니다.

인스턴스를 온라인으로 전환하려면 Auto Scaling 의 `Min`, `Max` 및 `Desired capacity` 값을 애플리케이션에 적합한 숫자로 조정하세요. 처음에는 이 값을 1로 설정하여 단일 인스턴스를 온라인 상태로 만들고 인스턴스가 사용자 지정 Chef 레시피 실행을 포함하여 예상되는 모든 작업을 수행하는지 확인하는 것이 좋습니다.

8단계: 인스턴스 검토

인스턴스를 시작한 후 예상대로 실행되는지 확인합니다.

1. 스크립트의 `--command-logs-bucket` 파라미터로 지정된 S3 버킷에 있는 Chef `startup`와 `terminate` 로그를 검토하세요. 기본적으로 로그는 `aws-opsworks-application-manager-logs-`*`account-id`*로 이름이 지정된 버킷에 저장됩니다.

   a. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

   b. 로그가 들어 있는 버킷을 선택합니다.

   c. `ApplyChefRecipes` 접두사로 이동하여 로그를 확인하세요.

2. Application Load Balancer 연결 및 상태를 확인합니다.

   로드 밸런서에 대한 액세스 로그를 보려면 다음 단계를 수행하세요. 스크립트의 `--lb-access-logs-path` 파라미터를 사용하여 로드 밸런서 액세스 로그를 저장할 S3 버킷을 지정할 수 있습니다.

a. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

b. S3 버킷을 선택한 다음 로그가 포함된 접두사로 이동합니다.

3. 인스턴스가 모든 Auto Scaling 및 Application Load Balancer 상태 점검(구성한 경우)을 통과하는지 확인합니다.

   새 인스턴스 탭에서 Auto Scaling 상태에 대한 정보를 볼 수 있습니다.

   a. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.

   b. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.

   c. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다.

   d. 목록에서 애플리케이션을 선택합니다. 애플리케이션 관리자가 개요 탭을 엽니다.

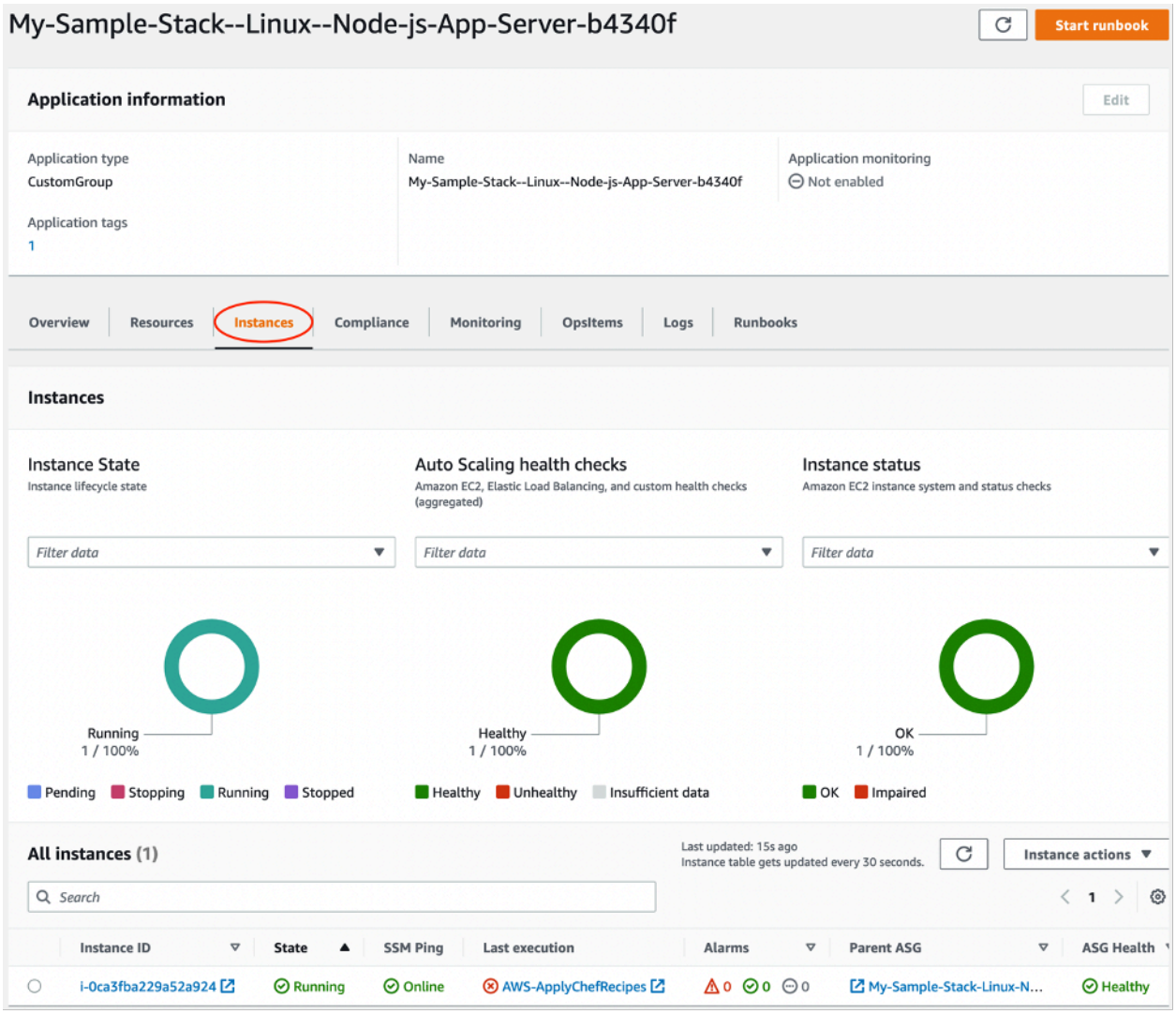
   e. 인스턴스 탭을 선택하여 Auto Scaling 상태에 대한 정보를 확인합니다.

Chef 레시피가 성공적으로 실행되는지 확인한 후 Auto Scaling 용량을 줄여 인스턴스를 종료할 수 있습니다. 사용자 지정 종료 레시피가 있는 경우 레시피가 예상대로 작동하는지 확인하세요.

9단계: Systems Manager 애플리케이션 관리자를 사용하여 인스턴스에서 작업을 모니터링하고 실행합니다.

이제 애플리케이션 관리자 페이지의 새 인스턴스 탭을 사용하여 인스턴스에서 작업을 모니터링하고 실행할 수 있습니다. 인스턴스 탭을 사용하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 애플리케이션 인스턴스를 사용하여 작업하기를 참조하세요.

인스턴스 탭을 사용하여 한 곳에서 여러 AWS 인스턴스를 볼 수 있습니다. 이 탭을 사용하면 인스턴스 상태에 대한 정보를 보고 문제를 해결할 수 있습니다.



인스턴스 탭을 보려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.
3. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다.
4. 목록에서 애플리케이션을 선택합니다. 애플리케이션 관리자가 개요 탭을 엽니다.
5. 인스턴스 상태 및 EC2 상태에 대한 정보를 보려면 인스턴스 탭을 선택합니다.

FAQ

다음 FAQ는 몇 가지 일반적인 질문에 대한 답변을 제공합니다.

주제

- 마이그레이션할 수 있는 AWS OpsWorks Stacks 버전은 무엇입니까?
- 마이그레이션된 인스턴스에서 사용할 수 있는 Chef 버전은 무엇입니까?
- 마이그레이션할 수 있는 리포지토리 유형은 무엇입니까?
- 프라이빗 Git 리포지토리를 계속 사용할 수 있나요?
- 인스턴스에 액세스하는 데 사용할 수 있는 SSH 키는 무엇입니까?
- 인스턴스가 자동으로 확장 및 축소되는 이유는 무엇입니까?
- Auto Scaling을 끌 수 있나요?
- 시작된 EC2 인스턴스에서 커널 및 패키지 업데이트를 수행할 수 있습니까?
- 내 인스턴스의 EBS 볼륨에 데이터가 없는 이유는 무엇입니까?
- 시작 템플릿에 설명된 EBS 볼륨이 마운트되지 않는 이유는 무엇입니까?
- Chef 레시피와 마운트 EBS 볼륨 로그는 어디에서 찾을 수 있나요?
- 마이그레이션 스크립트의 디버그 로그는 어디에서 찾을 수 있나요?
- 마이그레이션 스크립트가 CloudFormation 템플릿 버전 관리를 지원합니까?
- 여러 계층을 마이그레이션할 수 있나요?
- SecureString 파라미터를 어떻게 생성합니까?
- 새 Auto Scaling 의 인스턴스를 종료 이벤트로부터 보호하려면 어떻게 해야 합니까?
- 마이그레이션 스크립트에서 사용할 수 있는 로드 밸런서는 무엇입니까?
- 사용자 지정 쿡북 구성 레시피가 마이그레이션되었나요?
- 새로 생성한 인스턴스에서 배포 및 배포 취소 레시피를 실행할 수 있습니까?
- 내 Auto Scaling 이 포함되는 서브넷을 변경할 수 있습니까?

마이그레이션할 수 있는 AWS OpsWorks Stacks 버전은 무엇입니까?

Chef 11.10 및 Chef 12, Amazon Linux 2, Ubuntu 및 Red Hat Enterprise Linux 7 스택만 마이그레이션할 수 있습니다.

마이그레이션된 인스턴스에서 사용할 수 있는 Chef 버전은 무엇입니까?

마이그레이션된 인스턴스는 Chef 버전 11~14를 사용할 수 있습니다.

> **Note**
>
> Windows 스택 마이그레이션은 지원되지 않습니다.

마이그레이션할 수 있는 리포지토리 유형은 무엇입니까?

S3, Git 및 HTTP 리포지토리 유형을 마이그레이션할 수 있습니다.

프라이빗 Git 리포지토리를 계속 사용할 수 있나요?

예, 프라이빗 Git 리포지토리를 계속 사용할 수 있습니다.

프라이빗 GitHub 리포지토리를 사용하는 경우 SSH용 새 `Ed25519` 호스트 키를 생성해야 합니다. 이는 GitHub가 SSH에서 지원되는 키를 변경하고 암호화되지 않은 Git 프로토콜을 제거했기 때문입니다. `Ed25519` 호스트 키에 대한 자세한 내용은 GitHub 블로그 게시물 GitHub의 Git 프로토콜 보안 개선을 참조하세요. 새 `Ed25519` 호스트 키를 생성한 후 이 SSH 키에 대한 Systems Manager `SecureString` 매개변수를 생성하고 매개변수 이름을 `--repo-private-key` 매개변수 값으로 사용합니다. Systems Manager *`SecureString`* 파라미터를 생성하는 방법에 대한 자세한 내용은AWS Systems Manager 사용 설명서의 SecureString 파라미터 생성(AWS CLI)을 참조하세요.

다른 Git 리포지토리 유형의 경우 이 SSH 키에 대한 Systems Manager `SecureString` 매개 변수를 만들고 이 매개 변수 이름을 스크립트의 `--repo-private-key` 매개 변수 값으로 사용합니다.

인스턴스에 액세스하는 데 사용할 수 있는 SSH 키는 무엇입니까?

스크립트를 실행하면 스크립트가 스택에 구성된 SSH 키와 인스턴스를 마이그레이션합니다. SSH 키를 사용하여 인스턴스에 액세스할 수 있습니다. 스택과 인스턴스에 SSH 키가 제공되는 경우 스크립트는 스택의 키를 사용합니다. 어떤 SSH 키를 사용해야 할지 잘 모르겠으면 EC2 콘솔(https://console.aws.amazon.com/ec2/)에서 인스턴스를 확인하세요. EC2 콘솔의 세부 정보 페이지에는 인스턴스의 SSH 키가 표시됩니다.

인스턴스가 자동으로 확장 및 축소되는 이유는 무엇입니까?

Auto Scaling은 오토 스케일링의 조정 규칙에 따라 인스턴스를 조정합니다. 그룹의 최소, 최대 및 원하는 용량 값을 설정할 수 있습니다. Auto Scaling 은 이러한 값을 업데이트하면 그에 따라 용량을 자동으로 조정합니다.

Auto Scaling을 끌 수 있나요?

오토 스케일링의 최소, 최대 및 원하는 용량 값을 같은 수로 설정하여 Auto Scaling을 끌 수 있습니다. 예를 들어, 항상 10개의 인스턴스를 사용하려는 경우 최소, 최대 및 원하는 용량 값을 10으로 설정합니다.

시작된 EC2 인스턴스에서 커널 및 패키지 업데이트를 수행할 수 있습니까?

기본적으로 커널 및 패키지 업데이트는 EC2 인스턴스가 부팅될 때 발생합니다. 다음 단계를 사용하여 시작된 EC2 인스턴스에서 커널 또는 패키지 업데이트를 수행하세요. 예를 들어 배포 또는 구성 레시피를 실행한 후 업데이트를 적용할 수 있습니다.

1. EC2 인스턴스에 연결합니다.
2. 다음 `perform_upgrade` 함수를 생성하고 인스턴스에서 실행합니다.

```
perform_upgrade() {
    #!/bin/bash
    if [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
     sudo yum -y update
    elif [ -e '/etc/debian_version' ]; then
     sudo apt-get update
     sudo apt-get dist-upgrade -y
    fi
}
perform_upgrade
```

3. 커널과 패키지를 업데이트한 후에는 EC2 인스턴스를 재부팅해야 할 수 있습니다. 재부팅이 필요한지 확인하려면 다음 `reboot_if_required` 함수를 만들고 EC2 인스턴스에서 실행하세요.

```
reboot_if_required () {
 #!/bin/bash
 if [ -e '/etc/debian_version' ]; then
   if [ -f /var/run/reboot-required ]; then
     echo "reboot is required"
   else
     echo "reboot is not required"
   fi
 elif [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
  export LC_CTYPE=en_US.UTF-8
  export LC_ALL=en_US.UTF-8
  LATEST_INSTALLED_KERNEL=`rpm -q --last kernel | perl -X -pe 's/^kernel-(\S+).*/
$1/' | head -1`
  CURRENTLY_USED_KERNEL=`uname -r`
  if [ "${LATEST_INSTALLED_KERNEL}" != "${CURRENTLY_USED_KERNEL}" ];then
     echo "reboot is required"
  else
     echo "reboot is not required"
  fi
```

```
 fi
}
reboot_if_required
```

4. `reboot_if_required` 결과를 `reboot is required` 메시지로 실행하는 경우 EC2 인스턴스를 재부팅하세요. `reboot is not required` 메시지를 받은 경우 EC2 인스턴스를 재부팅할 필요가 없습니다.

내 인스턴스의 EBS 볼륨에 데이터가 없는 이유는 무엇입니까?

스크립트를 실행하면 스크립트가 EBS 볼륨의 구성을 마이그레이션하여 OpsWorks 스택 및 계층을 위한 대체 아키텍처를 생성합니다. 스크립트는 실제 인스턴스나 인스턴스에 포함된 데이터를 마이그레이션하지 않습니다. 스크립트는 계층 수준에서 EBS 볼륨의 구성만 마이그레이션하고 빈 EBS 볼륨을 시작된 EC2 인스턴스에 연결합니다.

다음 단계에 따라 이전 인스턴스의 EBS 볼륨에서 데이터를 가져오세요.

1. 이전 인스턴스 EBS 볼륨의 스냅샷을 만듭니다. EBS 스냅샷 생성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 EBS 스냅샷 만들기를 참조하세요.
2. 스냅샷으로 볼륨 생성 스냅샷에서 볼륨을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 스냅샷에서 볼륨 생성을 참조하세요.
3. 생성한 볼륨을 인스턴스에 연결합니다. 볼륨 연결에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 Amazon EBS 볼륨 연결을 참조하세요.

시작 템플릿에 설명된 EBS 볼륨이 마운트되지 않는 이유는 무엇입니까?

EBS 볼륨과 함께 `--launch-template` 파라미터에 시작 템플릿 ID를 제공하면 스크립트가 EBS 볼륨을 연결하지만 볼륨을 마운트하지는 않습니다. 스크립트가 시작한 EC2 인스턴스에 대해 생성한 `MountEBSVolumes` RunCommand 문서를 실행하여 연결된 EBS 볼륨을 마운트할 수 있습니다.

`--launch-template` 파라미터를 설정하지 않으면 스크립트가 템플릿을 생성하고 Auto Scaling 이 새 EC2 인스턴스를 시작하면 Auto Scaling 이 자동으로 EBS 볼륨을 연결한 다음 `SetupAutomation` 명령을 실행하여 연결된 볼륨을 계층 설정에 구성된 마운트 포인트에 마운트합니다.

Chef 레시피와 마운트 EBS 볼륨 로그는 어디에서 찾을 수 있나요?

OpsWorks는 `--command-logs-bucket` 파라미터 값을 제공하여 지정할 수 있는 S3 버킷으로 로그를 전송합니다. 기본 S3 버킷 이름의 형식은 다음과 같습니다. `aws-opsworks-stacks-`

`application-manager-logs-`*`account-id`* Chef 레시피 로그는 `ApplyChefRecipes` 접두사에 저장됩니다. 마운트 EBS 볼륨 로그는 `MountEBSVolumes` 접두사에 저장됩니다. 스택에서 마이그레이션되는 모든 계층은 동일한 S3 버킷으로 로그를 전송합니다.

> **Note**
>
> - S3 버킷의 수명 주기 구성에는 30일 후에 로그를 삭제하는 규칙이 포함되어 있습니다. 로그를 30일 이상 보관하려면 S3 버킷의 수명 주기 구성에서 규칙을 업데이트해야 합니다.
> - 현재 OpsWorks는 셰프 `setup`와 `terminate` 레시피만 기록합니다.

마이그레이션 스크립트의 디버그 로그는 어디에서 찾을 수 있나요?

스크립트는 `aws-opsworks-stacks-transition-logs-`*`account-id`*로 이름이 지정된 버킷에 디버그 로그를 저장합니다. 마이그레이션한 레이어 이름과 일치하는 폴더 아래의 S3 버킷의 `migration_script` 폴더에서 디버그 로그를 찾을 수 있습니다.

마이그레이션 스크립트가 CloudFormation 템플릿 버전 관리를 지원합니까?

이 스크립트는 마이그레이션하려는 계층 또는 스택을 대체하는 CloudFormation 유형의 Systems Manager 문서를 생성합니다. 매개 변수가 같더라도 스크립트를 다시 실행하면 이전에 내보낸 계층 템플릿의 새 버전을 내보냅니다. 템플릿 버전은 스크립트 로그와 동일한 S3 버킷에 저장됩니다.

여러 계층을 마이그레이션할 수 있나요?

스크립트의 `--layer-id` 매개변수가 단일 계층으로 전달됩니다. 여러 계층을 마이그레이션하려면 스크립트를 다시 실행하고 다른 `--layer-id`로 전달합니다.

동일한 OpsWorks 스택에 속하는 계층은 애플리케이션 관리자의 동일한 애플리케이션 아래에 나열됩니다.

1. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.
3. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다.
4. 애플리케이션을 선택합니다. 애플리케이션 이름은 `app-`*`stack-name`*`-`*`first-six-characters-stack-id`*로 시작합니다.
5. 앱으로 시작하는 최상위 요소에는 OpsWorks 스택에 해당하는 모든 구성 요소가 표시됩니다. 여기에는 OpsWorks 계층에 해당하는 구성 요소가 포함됩니다.

6. 계층에 해당하는 구성 요소를 선택하여 계층의 리소스를 확인합니다. OpsWorks 계층을 나타내는 구성 요소는 사용자 지정 애플리케이션 섹션에서도 개별 애플리케이션으로 볼 수 있습니다.

`SecureString` 파라미터를 어떻게 생성합니까?

Systems Manager를 사용하여 `SecureString` 매개변수를 생성할 수 있습니다. Systems Manager `SecureString` 파라미터 생성 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Systems Manager 파라미터 생성(AWS CLI) 또는 Systems Manager 파라미터 생성(콘솔)을 참조하세요.

`SecureString` 매개변수를 `--http-username`, `--http-password` 또는 `--repo-private-key` 매개변수의 값으로 제공해야 합니다.

새 Auto Scaling 의 인스턴스를 종료 이벤트로부터 보호하려면 어떻게 해야 합니까?

`--enable-instance-protection` 파라미터를 `TRUE`로 설정하고 종료 이벤트로부터 보호하려는 각 EC2 인스턴스에 `protected_instance` 태그 키를 추가하여 인스턴스를 보호할 수 있습니다. `--enable-instance-protection` 파라미터를 `TRUE`로 설정하고 `protected_instance` 태그 키를 추가하면 스크립트가 새 Auto Scaling 에 사용자 지정 종료 정책을 추가하고 `ReplaceUnhealthy` 프로세스를 일시 중단합니다. `protected_instance` 태그 키가 있는 인스턴스는 다음 종료 이벤트로부터 보호됩니다.

- 스케일 인 이벤트
- 인스턴스 새로 고침
- 리밸런싱
- 인스턴스 최대 수명
- 리스팅 인스턴스 종료 허용
- 비정상 인스턴스의 종료 및 교체

> **Note**
>
> 보호하려는 인스턴스에 `protected_instance` 태그 키를 설정해야 합니다. 키는 대/소문자를 구분합니다. 해당 태그 키가 있는 모든 인스턴스는 태그 값에 관계없이 보호됩니다.
> 사용자 지정 종료 정책의 실행 시간을 줄이려면 `default_sample_size` 함수 코드 변수의 값을 업데이트하여 Lambda 함수가 보호된 인스턴스를 필터링하는 데 사용하는 기본 인스턴스 수를 늘릴 수 있습니다. 기본값은 15입니다. `default_sample_size`를 늘리면 Lambda 함수

에 할당된 메모리를 늘려야 할 수 있으며, 이로 인해 Lambda 함수의 비용이 증가할 수 있습니다. AWS Lambda 요금에 대한 자세한 내용은 AWS Lambda 요금을 참조하세요.

마이그레이션 스크립트에서 사용할 수 있는 로드 밸런서는 무엇입니까?

스크립트는 세 가지 로드 밸런서 옵션을 제공합니다.

- (권장) 새 Application Load Balancer를 생성합니다. 기본적으로 스크립트는 새 Application Load Balancer를 생성합니다. `--lb-type` 파라미터를 `ALB`로 설정할 수도 있습니다. Application Load Balancer에 대한 자세한 내용은 Elastic Load Balancing 사용 설명서의 Application Load Balancer란 무엇인가?를 참조하세요.
- Application Load Balancer가 옵션이 아닌 경우 `--lb-type` 파라미터를 `Classic`로 설정하여 Classic Load Balancer를 생성하세요. 이 옵션을 선택하면 OpsWorks 계층에 연결된 기존 Classic Load Balancer가 애플리케이션과 분리되어 유지됩니다. Application Load Balancer에 대한 자세한 내용은 Elastic Load Balancing 사용 설명서의 Classic Load Balancer란 무엇인가?를 참조하세요.
- `--lb-type` 파라미터를 `None`로 설정하여 기존 로드 밸런서를 연결할 수 있습니다.

> **Important**
>
> AWS OpsWorks Stacks 계층에 새 Elastic Load Balancing 로드 밸런서를 생성하는 것이 좋습니다. 기존 Elastic Load Balancing 로드 밸런서를 사용하려는 경우 먼저 해당 로드 밸런서가 다른 목적으로 사용되고 있지 않고 연결된 인스턴스가 없는지 확인해야 합니다. 로드 밸런서가 계층에 연결되면 OpsWorks가 기존 인스턴스를 모두 제거하고 로드 밸런서를 해당 계층의 인스턴스만 처리하도록 구성합니다. 로드 밸런서를 계층에 연결한 후 Elastic Load Balancing 콘솔 또는 API를 사용하여 로드 밸런서의 구성을 수정하는 것은 기술적으로는 가능하지만 변경 사항이 영구적이지 않으므로 이렇게 하면 안 됩니다.

Auto Scaling 에 기존 OpsWorks 계층 로드 밸런서를 연결하는 방법

1. `--lb-type` 파라미터를 `None`로 설정한 상태로 마이그레이션 스크립트를 실행합니다. 값이 `None`로 설정된 경우 스크립트는 로드 밸런서를 복제하거나 생성하지 않습니다.
2. 스크립트가 CloudFormation 스택을 배포한 후 Auto Scaling `Min` `Max` 그룹과 `Desired capacity` 값을 업데이트한 다음 애플리케이션을 테스트합니다.
3. 스크립트 출력에 표시된 `Link to the template`을 선택합니다. 터미널을 닫은 경우 다음 단계에 따라 템플릿에 액세스하세요.

a. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.

b. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.

c. CloudFormation 스택을 선택한 다음 템플릿 라이브러리를 선택합니다.

d. 내가 소유를 선택하고 템플릿을 찾으세요.

4. CloudFormation 템플릿의 작업 메뉴에서 편집을 선택합니다.

5. CloudFormation 템플릿의 `ApplicationAsg` 리소스 섹션 내에서 `LabelBalancerNames` 속성을 업데이트합니다.

```
ApplicationAsg:
   DependsOn: CustomTerminationLambdaPermission
   Properties:
   #(other properties in ApplicationAsg to remain unchanged)
      LoadBalancerNames:
        - load-balancer-name
      HealthCheckType: ELB
```

6. Auto Scaling 인스턴스 상태 확인에서 로드 밸런서의 상태 확인도 사용하도록 하려면 아래 `HealthCheckType` 섹션을 삭제하고 `ELB`를 입력하세요. EC2 상태 확인만 필요한 경우 템플릿을 변경할 필요가 없습니다.

7. 변경 내용을 저장합니다. 저장하면 템플릿의 새 기본 버전이 생성됩니다. 계층에 대한 스크립트를 처음으로 실행하고 콘솔에서 변경 내용을 처음으로 저장한 경우 새 버전은 2입니다.

8. 작업에서 스택 프로비저닝을 선택합니다.

9. 템플릿의 기본 버전을 사용하려 한다는 것을 확인합니다. 기존 스택 선택이 선택되어 있는지 확인하고 업데이트할 CloudFormation 스택을 선택하세요.

10. 검토 및 프로비저닝 페이지가 표시될 때까지 각 후속 페이지에서 다음을 선택합니다. 검토 및 프로비저닝 페이지에서 가 사용자 지정 이름으로 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 인정하고 선택한 템플릿의 변경으로 인해 AWS CloudFormation 가 기존 AWS 리소스를 업데이트하거나 제거할 수 있음을 이해합니다.

11. [스택 프로비저닝]을 선택합니다.

업데이트를 롤백해야 하는 경우 다음 단계를 수행합니다.

1. 작업을 선택한 다음 스택 프로비저닝을 선택합니다.

2. 기존 버전 중 하나를 선택한 다음 이전 템플릿 버전을 선택합니다.

3. 기존 스택 선택을 선택하고 업데이트할 CloudFormation 스택을 선택합니다.

사용자 지정 쿡북 구성 레시피가 마이그레이션되었나요?

설정 이벤트 중에는 사용자 지정 쿡북 구성을 실행할 수 없습니다. 이 스크립트는 사용자 지정 쿡북 구성 레시피를 마이그레이션하고 Systems Manager 자동화 런북을 자동으로 생성합니다. 그러나 레시피를 수동으로 실행해야 합니다.

구성 레시피를 실행하려면 다음 단계를 수행합니다.

1. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.
3. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다.
4. 애플리케이션을 선택합니다. 애플리케이션 이름은 `app-`*`stack-name`*로 시작합니다.
5. 리소스를 선택한 다음 구성 런북을 선택합니다.
6. 자동화 실행을 선택합니다.
7. 구성 레시피를 실행할 인스턴스 ID를 선택한 다음 [실행]을 선택합니다.

새로 생성한 인스턴스에서 배포 및 배포 취소 레시피를 실행할 수 있습니까?

스크립트는 계층 구성에 따라 세 가지 가능한 자동화 런북을 만들 수 있습니다.

- 설정
- 구성
- Terminate

또한 스크립트는 `AWS-ApplyChefRecipes Run Command` 문서에 대한 입력 값을 포함하는 다음과 같은 Systems Manager 매개변수를 만들 수 있습니다.

- 설정
- 배포
- 구성
- Undeploy
- Terminate

스케일 아웃 이벤트가 발생하면 설치 자동화 런북이 자동으로 실행됩니다. 여기에는 원본 OpsWorks 계층에서 사용자 지정 쿡북 레시피를 설정하고 배포하는 작업이 포함됩니다. 스케일 인 이벤트가 발생하면 자동화 종료 런북이 자동으로 실행됩니다. 자동화 종료 런북에는 원본 OpsWorks 계층의 종료 레시피가 포함되어 있습니다.

배포 취소를 실행하거나 레시피를 수동으로 구성하는 경우 다음 단계를 수행합니다.

1. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.
3. 애플리케이션 섹션에서 사용자 지정 애플리케이션을 선택합니다.
4. 애플리케이션을 선택합니다. 애플리케이션 이름은 `app-`*`stack-name`*`-`*`first-six-characters-stack-id`*로 시작합니다. 애플리케이션 관리자가 개요 탭을 엽니다.
5. 리소스를 선택한 다음 자동화 구성 런북을 선택합니다.
6. 자동화 실행을 선택합니다.
7. `applyChefRecipesPropertiesParameter`자동화 런북 입력 매개변수의 경우 올바른 Systems Manager 매개변수를 참조하세요. Systems Manager 매개변수 이름은 형식 `/ApplyChefRecipes-Preset/`*`OpsWorks-stack-name`*`-`*`OpsWorks-layer-name`*`-`*`first-six-characters-stack-id`*`/`*`event`* (실행하려는 리시피에 따라 *`###`* 값이 `Configure`, `Deploy` 또는 `Undeploy`임)을 따릅니다
8. 레시피를 실행할 인스턴스 ID를 선택하고 [실행]을 선택합니다.

내 Auto Scaling 이 포함되는 서브넷을 변경할 수 있습니까?

기본적으로 Auto Scaling 은 OpsWorks 스택 VPC의 모든 서브넷 전체를 포괄합니다. 확장할 서브넷을 업데이트하려면 다음 단계를 수행하세요.

1. 스크립트 출력에 표시된 `Link to the template`을 선택합니다. 터미널을 닫은 경우 다음 단계에 따라 템플릿에 액세스하세요.
   a. https://console.aws.amazon.com/systems-manager/ Systems Manager 콘솔을 엽니다.
   b. 탐색 창에서 [애플리케이션 관리자]를 선택합니다.
   c. CloudFormation 스택을 선택한 다음 템플릿 라이브러리를 선택합니다.
   d. 내가 소유를 선택하고 템플릿을 찾으세요.
2. 작업에서 스택 프로비저닝을 선택합니다.

3. 기본 템플릿을 사용하려 한다는 것을 확인합니다. 기존 스택 선택을 선택하고 업데이트할 CloudFormation 스택을 선택합니다.

   > **Note**
   >
   > `--provision-application` 파라미터를 FALSE로 설정한 상태에서 스크립트를 실행한 경우 새 CloudFormation 스택을 생성해야 합니다.

4. `SubnetIDs` 파라미터에는 Auto Scaling 에 포함시킬 서브넷 ID를 쉼표로 구분한 목록을 제공하세요.
5. 검토 및 프로비저닝 페이지가 표시될 때까지 다음을 선택합니다.
6. 검토 및 프로비저닝 페이지에서 가 사용자 지정 이름으로 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 인정함을 선택하고 선택한 템플릿의 변경 AWS CloudFormation 으로 인해가 기존 AWS 리소스를 업데이트하거나 제거할 수 있음을 이해합니다.
7. [스택 프로비저닝]을 선택합니다.

문제 해결

이 섹션에는 몇 가지 일반적인 문제와 이러한 문제의 해결 방법이 제시되어 있습니다.

주제

- 제공된 보안 주체가 유효하지 않습니다.
- Auto Scaling 보호 인스턴스가 활성화되어 있는 경우 CloudFormation 스택을 삭제할 수 없습니다.
- 기존 S3 버킷 및 접두사를 제공할 때 액세스 거부 오류가 발생했습니다.

제공된 보안 주체가 유효하지 않습니다.

문제: 제공한 주체가 유효하지 않다는 오류 메시지가 나타납니다.

원인: 이는 Auto Scaling 에 서비스 역할이 없기 때문에 발생합니다.

해결 방법: 오류가 발생한 리전에 Auto Scaling 을 생성하세요. Auto Scaling 을 생성하면 사용자 지정 종료 정책에 필요한 서비스 연결 역할이 생성됩니다.

Auto Scaling 보호 인스턴스가 활성화되어 있는 경우 CloudFormation 스택을 삭제할 수 없습니다.

문제: `--enable-instance-protection` 파라미터가 `TRUE`로 설정되어 있고 Auto Scaling 의 일부 EC2 인스턴스가 `protected_instance` 태그 키로 보호되므로 AWS CloudFormation 스택이 완전히 삭제되지 않습니다.

원인: EC2 인스턴스에는 종료 이벤트로부터 인스턴스를 보호하는 `protected_instance` 태그 키가 있습니다.

해결 방법: EC2 인스턴스에서 `protected_instance` 태그 키를 제거합니다. 이렇게 하면 Auto Scaling 을 축소할 수 있습니다. Auto Scaling 그룹이 축소되면 AWS CloudFormation 스택을 삭제할 수 있습니다.

기존 S3 버킷 및 접두사를 제공할 때 액세스 거부 오류가 발생했습니다.

문제: 기존 S3 버킷과 접두사를 제공하면 `AccessDenied` 오류가 발생합니다.

원인: S3 버킷 정책은 로드 밸런서 로그를 버킷으로 전송하는 데 필요한 권한을 제공하지 않습니다.

해결 방법: 스크립트가 로드 밸런서 액세스 로그를 버킷에 전송할 수 있도록 S3 버킷 정책을 업데이트하세요. 버킷 정책을 업데이트하는 방법에 대한 자세한 내용은 Elastic Load Balancing: Application Load Balancer 사용 설명서의 Application Load Balancer에 대한 액세스 로그 활성화를 참조하세요.

AWS OpsWorks Stacks Detach in Place 도구 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다.

이 섹션에서는 AWS OpsWorks Stacks Detach in Place 도구를 사용하여 OpsWorks Stacks 서비스에서 OpsWorks 인스턴스를 분리하는 방법을 설명합니다.

분리한 인스턴스는에 남아 있지만 AWS 계정 OpsWorks를 사용하여 더 이상 인스턴스를 관리할 수 없습니다. 대신 Amazon EC2 AWS Systems Manager또는 EC2 호환 접근 방식을 사용하여 인스턴스를 구성하고 관리합니다.

높은 수준에서 분리 프로세스에는 다음 단계가 포함됩니다.

1. 도구는 검증 검사를 수행하여 리소스가 분리될 준비가 되었는지 확인합니다.
2. 도구는 OpsWorks 스택에서 사용자 지정 JSON을 내보내고 Amazon S3에 객체로 저장합니다.
3. 이 도구는 각 OpsWorks Stacks 수명 주기 이벤트를 나타내는 Systems Manager Automation 문서를 생성합니다.
4. 이 도구는 분리되는 모든 인스턴스에 대해 AWS Service Catalog AppRegistry 카탈로그를 생성하고 OpsWorks 계층에서 모든 Elastic Load Balancing(ELB) 로드 밸런서를 분리합니다.
5. 마지막으로 도구는 Amazon Relational Database Service(Amazon RDS) 인스턴스를 포함한 다른 리소스를 분리하고 등록 취소합니다.

프로세스 작동 방식

In Place 분리 도구는 계층 분리를 진행하기 전에 인스턴스를 확인하고 구성하는 일련의 단계를 안내하는 다음과 같은 3가지 명령과 같은 마법사 경험을 제공합니다.

Command	설명
`handle-prerequisites`	이 명령은 계층의 모든 인스턴스가 분리에 적합한지 여부를 분석하고 사전 조건을 해결합니다. 인스턴스는 OpsWorks에서 정상 상태여야 하며, 시간 또는 로드 기반 오토 스케일러를 가질 수 없고, 최신 OpsWorks 에이전트 버전이 설치되어 있어야 합니다. 또한 명령은 모든 인스턴스에 SSM 에이전트를 지원하는 데 필요한 권한이 있는지, 최신 SSM 에이전트 버전이 설치되어 있는지 확인합니다. 명령은 SSM 에이전트가 없는 경우 설치하고 최신 버전을 사용하지 않는 경우 SSM 에이전트를 업데이트합니다. 명령은 필요한 권한도 추가합니다.
`detach`	이 명령은 지정된 계층의 모든 OpsWorks 인스턴스를 분리합니다.

Command	설명
	먼저 명령은 사전 조건 확인을 실행하여 계층이 분리에 적합한지 확인합니다. 사전 조건을 해결하지 않으려면 강제로 분리할 수 있는 옵션이 제공됩니다. 다음으로 명령은 OpsWorks 태그 지정 APIs를 통해 또는 계층 및 스택에서 태그 전파를 통해 인스턴스에 추가된 모든 태그가 보존됨을 나타냅니다. 분리가 완료된 후 관련 EC2 APIs를 사용하여 이러한 태그를 제거할 수 있습니다. 그런 다음 명령은 Chef 관련 구성을 SSM 파라미터로 내보낼지 확인합니다. 계층에 Classic Load Balancer가 연결되어 있는 경우 명령은 가동 중지 시간을 방지하기 위해 로드 밸런서를 분리할 수 있는지 묻습니다.
`cleanup`	이 명령은 계정에서 OpsWorks의 모든 개체를 삭제합니다. 인스턴스를 종료하고 모든 스택을 삭제합니다. 이는 계정을 정리하기 위한 마지막 단계로 더 이상 필요하지 않은 리소스에 사용해야 합니다. **Note** `cleanup` 명령을 실행하기 전에 며칠 동안 새 설정을 실행하는 것이 좋습니다. 이렇게 하면 필요한 경우 스택에서 필요한 구성을 즉시 사용할 수 있습니다.

제한 사항

현장 분리 도구의 주요 목적은 OpsWorks Stacks 인스턴스를 안전하게 분리하는 것입니다. 이 섹션에서는 도구의 제한 사항을 요약합니다.

- Windows SSM 에이전트 - SSM 에이전트가 인스턴스에 설치되지 않은 경우 수동으로 설치해야 합니다. 에이전트가 최신 버전으로 업데이트되지 않은 경우에도 마찬가지입니다.
- 시간/로드 Auto Scaling 인스턴스 - 분리 도구는 Auto Scaling이 활성화된 인스턴스를 지원하지 않습니다. 분리하려는 인스턴스에서 Auto Scaling을 비활성화해야 합니다.
- 권한 - 분리 도구는 OpsWorks 콘솔의 권한 페이지에 지정된 IAM 엔터티를 생성하거나 생성하지 않습니다.
- 앱 - 분리 도구는 OpsWorks 외부에서 앱을 생성하거나 생성하지 않습니다.

시작

1단계: 사전 조건이 충족되었는지 확인

인플레이스 분리 도구의 3가지 명령은 모두 Python 스크립트입니다.이 스크립트는 로컬, EC2 인스턴스 또는를 사용하여 실행할 수 있습니다AWS CloudShell.

AWS CloudShell 는 브라우저 기반 셸로, 인기 있는 도구(예: AWS CLI 및 Python)가 사전 설치된 selected AWS 리전. AWS CloudShell com의 리소스에 대한 명령줄 액세스 권한을 AWS 제공합니다. 를 사용할 때는 콘솔에 로그인할 때 사용하는 것과 AWS CloudShell동일한 자격 증명을 사용합니다.

이 연습에서는 사용 중이라고 가정합니다 AWS CloudShell.

2단계: 스크립트 다운로드

1. 다음 명령을 실행하여 마이그레이션 스크립트와 모든 관련 파일이 포함된 zip 파일을 다운로드합니다.

   ```
   aws s3api get-object \
   --bucket detach-in-place-bucket-prod-us-east-1 \
   --key detach_in_place_script.zip detach_in_place_script.zip
   ```

2. 다음 명령을 실행하여 파일의 압축을 풉니다.

   ```
   unzip detach_in_place_script.zip
   ```

   파일의 압축을 푼 후 다음 파일을 사용할 수 있습니다.

   - README.md
   - 라이선스

- INFO
- requirements.txt
- TODO.py

3. 필요한 경우 다음 명령을 실행pipenv하여를 설치합니다.

```
pip install pipenv
```

3단계: 스크립트 실행

먼저 다음 명령을 실행하여 스크립트를 실행할 수 있도록 환경을 설정합니다.

```
pipenv install -r requirements.txt
pipenv shell
```

그런 다음 스크립트 파라미터를 검토합니다.

Command	파라미터	설명	형식	필수	기본값
handle-prerequisites	--layer-id	분리하려는 계층의 ID입니다.	String	예	-
	--region	OpsWorks 스택의 리전입니다. OpsWorks 스택 리전과 API 엔드포인트 리전이 다른 경우 스택 리전을 사용하세요. OpsWorks 스택의 다른 리소스 부분(예: EC2 인스턴스 및 서브넷)과 동일한 리전입니다.	String	No	us-east-1
detach	--layer-id	분리하려는 계층의 ID입니다.	String	예	-
	--batch-size	계층에서 분리할 인스턴스 수(예: 5).	String	No	-

Command	파라미터	설명	형식	필수	기본값
	`--region`	OpsWorks 스택의 리전입니다. OpsWorks 스택 리전과 API 엔드포인트 리전이 다른 경우 스택 리전을 사용하세요. OpsWorks 스택의 다른 리소스 부분(예: EC2 인스턴스 및 서브넷)과 동일한 리전입니다.	String	No	us-east-1
`cleanup`	`--stack-id`	삭제하려는 스택의 ID입니다.	String	No	상호 배타적, 계층 ID 또는 스택 ID를 지정해야 합니다.
	`--layer-id`	삭제하려는 계층의 ID	String	No	
	`--region`	OpsWorks 스택의 리전입니다. OpsWorks 스택 리전과 API 엔드포인트 리전이 다른 경우 스택 리전을 사용하세요. OpsWorks 스택의 다른 리소스 부분(예: EC2 인스턴스 및 서브넷)과 동일한 리전입니다.	String	No	us-east-1

다음과 같이 옵션을 사용하여 `cleanup` 명령을 실행하여 `detach`, `handle-prerequisites` 및 명령에 사용할 수 있는 `--help` 옵션을 볼 수 있습니다.

```
python3 layer_detacher.py detach --help
python3 layer_detacher.py handle-prerequisites --help
python3 layer_detacher.py cleanup --help
```

이제 시작할 준비가 되었습니다. 다음 예제에서는 다양한 사용 사례에 대해 명령을 실행하는 방법을 보여줍니다.

예시:

- 예제 1: 계층이 모든 사전 요구 사항을 충족하고 분리할 수 있는지 확인
- 예제 2: 계층의 모든 인스턴스 분리
- 예제 3: 계층의 모든 인스턴스를 일괄적으로 분리
- 예제 4: 계층의 모든 리소스 정리 및 계층 삭제
- 예제 5: 스택의 모든 리소스 정리 및 스택 삭제

예제 1: 계층이 모든 사전 요구 사항을 충족하고 분리할 수 있는지 확인

다음 명령은 OpsWorks 계층(및 포함된 인스턴스)에 대한 정보를 읽고 다음 사전 조건이 충족되는지 확인합니다.

- 모든 인스턴스가 온라인 상태입니다.
- 로드/시간 Auto Scaling 인스턴스가 없습니다.
- 모든 인스턴스에는 최신 OpsWorks 에이전트가 있습니다.
- 모든 인스턴스에는 최신 SSM 에이전트가 설치 및 구성되어 있습니다.
- 모든 인스턴스에는 SSH 키 페어가 있습니다.
- 모든 인스턴스는 정확히 하나의 계층에 속합니다.

```
python3 layer_detacher.py handle-prerequisites \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

예제 2: 계층의 모든 인스턴스 분리

다음 명령은 계층의 모든 인스턴스를 반복하고, 인스턴스가 사전 조건을 충족하는지 확인하고, 사전 조건을 충족하는 모든 인스턴스를 병렬로 분리하려고 시도합니다. 하나 이상의 사전 조건이 충족되지 않으면 명령은 나머지 규정 미준수 인스턴스에 대해 강제 분리 옵션을 제공합니다.

인스턴스를 분리하기 전에 명령은 다음을 수행합니다.

1. 사용자 지정 JSON을 저장하고 S3에 업로드합니다.
2. 계층의 모든 OpsWorks 수명 주기 이벤트에 대한 SSM 자동화 문서를 생성하고 자동화 문서에 대한 실행 로그를 S3에 업로드합니다.
3. 분리할 모든 인스턴스에 대해 AppRegistry 애플리케이션을 생성합니다. 애플리케이션에는 분리된 모든 인스턴스와 리소스를 포함하는 리소스 그룹이 연결되어 있습니다. 리소스에는 수명 주기 이벤

트 및 사용자 지정 Chef 레시피에 대한 정보를 포함하는 SSM 자동화 문서 및 SSM 파라미터가 포함됩니다.

4. Classic Load Balancer가 있는 경우 계층에서 분리합니다.

이 명령은 OpsWorks 리소스만 수정합니다. EC2 인스턴스의 상태는 동일하게 유지됩니다.

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

예제 3: 계층의 모든 인스턴스를 일괄적으로 분리

다음 명령은 이전 예제와 동일합니다. 유일한 차이점은 인스턴스를 배치로 분리한다는 것입니다.

이 명령은 OpsWorks 리소스만 수정합니다. EC2 인스턴스의 상태는 동일하게 유지됩니다.

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
--region opsworks-stack-region \
--batch-size 5
```

예제 4: 계층의 모든 리소스 정리 및 계층 삭제

다음 명령은 계층의 모든 리소스를 반복하고 삭제합니다. 더 자세히 설명하면 OpsWorks 및 EC2의 모든 인스턴스를 중지 및 삭제하고, 로드 밸런서를 분리하고, Amazon RDS 인스턴스, 탄력적 IPs. 리소스를 정리하면 계층이 삭제됩니다.

이 명령은 OpsWorks 리소스 및 EC2 인스턴스를 삭제합니다. EC2 인스턴스를 그대로 유지하려면 `detach` 명령을 사용하기 전에 `cleanup` 명령을 사용합니다. 이렇게 하면 `cleanup` 명령이 나머지 리소스를 모두 삭제합니다.

```
python3 layer_detacher.py cleanup \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

예제 5: 스택의 모든 리소스 정리 및 스택 삭제

다음 명령은 모든 계층을 반복한 다음 각 계층의 리소스를 반복합니다. 각 계층에 대해 명령은 OpsWorks 및 EC2의 모든 인스턴스를 중지 및 삭제하고, 로드 밸런서를 분리하며, Amazon RDS 인스

턴스, 탄력적 IPs. 그러면 명령이 계층을 삭제합니다. 이 스택에 속하는 모든 계층에서 동일한 프로세스가 수행됩니다. 마지막으로 모든 계층이 삭제되면 스택이 제거됩니다.

이 명령은 OpsWorks 리소스 및 EC2 인스턴스를 삭제합니다. EC2 인스턴스를 그대로 유지하려면 `detach` 명령을 사용하기 전에 `cleanup` 명령을 사용합니다. 이렇게 하면 `cleanup` 명령이 나머지 리소스를 모두 삭제합니다.

```
python3 layer_detacher.py cleanup \
--stack-id opsworks-stack-id \
--region opsworks-stack-region
```

4단계: OpsWorks에서 분리한 후 리소스 계속 운영

`detach` 명령을 실행한 후 도구는 분리된 계층에 해당하는 새 AWS Service Catalog AppRegistry 애플리케이션을 생성합니다. 애플리케이션 이름은 형식을 따릅니다*layer-name*`---`*layer-id*. 또한 태그를 추가하여 분리된 계층과 일치하는 애플리케이션을 `OpsWorksLayerId` 고유하게 식별합니다.

이 애플리케이션에 새 AWS 리소스(예: 새 EC2 인스턴스)를 추가하려면 다음 중 하나를 수행할 수 있습니다.

1. AppRegistry 애플리케이션의 고유한 애플리케이션 태그로 리소스에 태그를 지정합니다.

   태그 키: `awsApplication`

   값: `arn:aws:resource-groups:`*region*`:`*account-id*`:group/`*application-name*`/`*application-id*`>`
2. associate-resource 명령을 실행합니다.

또한 각 AppRegistry 애플리케이션에 대해 리소스 그룹이 생성됩니다. 리소스 그룹에는 다음 태그가 포함되어 있습니다.

태그 키	값
`EnableAWSServiceCatalogAppRegistry`	`TRUE`
`aws:servicecatalog:applicationName`	*application-name*

태그 키	값
aws:servicecatalog:applicationId	*application-id*
aws:servicecatalog:applicationArn	arn:aws:servicecatalog: *region*:*account-id* :/applications/ *application-id*

분리 후 작업 수행

다음 표에서는 분리 후 작업을 수행하는 방법에 대한 정보를 제공합니다.

작업	설명
수명 주기 이벤트 실행	detach 명령을 실행한 후 옵션을 선택하면 스크립트는 OpsWorks 수명 주기 이벤트 5개와 일치하는 자동화 문서 5개를 생성합니다. 각 자동화 문서의 이름은 형식을 따릅니다*layer-id*_*lifecycle-event*_automation_document . Systems Manager에서 OpsWorks 동작을 시뮬레이션하려면 프로비저닝, EC2 인스턴스 종료 또는 레시피 배포/제거 시 자동화 실행을 수동으로 트리거해야 합니다.
사용자 지정 JSON 업데이트	스택 및 계층에 대한 사용자 지정 JSON은 분리 중에 지정된 S3 버킷 또는 생성된 새 S3 버킷에 저장됩니다. JSON 파일에 대해 저장된 파일 이름은 다음과 같습니다. • layercustomjson.json • stackcustomjson.json

작업	설명
수명 주기 이벤트에 대한 실행 목록 변경	각 수명 주기 이벤트의 실행 목록은 해당 자동화 문서에 정의되어 있습니다. 실행 목록을 변경하려면 AppRegistry 애플리케이션에서 자동화 문서를 찾아 `RunList` 파라미터를 수정합니다. 자동화 문서가 트리거`AWS-ApplyChefRecipes` 하는가 OpsWorks와 동일한 소스를 지원하므로 레시피 및 쿡북을 업데이트하는 프로세스는 변경되지 않습니다.
자동 복구/자동 조정 관리	인스턴스를 분리하면 OpsWorks 에이전트가 제거됩니다. 에이전트가 없으면 OpsWorks는 비정상 인스턴스를 자동으로 복구하거나 교체할 수 없으며 플릿을 자동으로 확장할 수도 없습니다. 자동 조정 및 실패한 인스턴스 교체를 계속하려면 Amazon EC2 Auto Scaling 그룹을 생성합니다. 그룹은 Amazon EC2가 교체가 필요한 비정상 인스턴스를 감지하면 원하는 용량을 유지하기 위해 새 인스턴스를 시작합니다.
Load Balancer 관리	계층이 Classic Load Balancer를 사용하는 경우 `detach` 명령은 인스턴스 등록을 취소하기 전에 Classic Load Balancer를 분리합니다. 이는 분리 과정에서 모든 ELB 인스턴스 연결이 Amazon EC2에 보존되어 가동 중지 시간이 0이 되도록 하기 위한 것입니다. 프로세스가 완료되면 EC2에서 ELB를 관리할 수 있습니다.

작업	설명
인스턴스에 연결	`handle-prerequisites` 또는 `detach` 명령을 실행하면 두 가지 검사가 수행됩니다. • SSM 에이전트의 버전 및 권한 • SSH 키 또한 명령은 세션 관리자를 사용하여 인스턴스에 연결할 수 있도록 SSM 에이전트를 업데이트하고 필요한 권한을 추가하는 옵션을 제공합니다. SSH 키가 있는 경우 인스턴스에 SSH를 입력할 수도 있습니다.

Systems Manager Application Manager 인스턴스 탭 사용

분리 후에는 Application Manager 인스턴스 탭에서 인스턴스를 보고 관리할 수 있습니다.

인스턴스 탭은 애플리케이션의 상태, 상태 및 마지막 명령 상태와 같은 애플리케이션의 EC2 인스턴스에 대한 집계 정보를 제공합니다. 이 탭을 사용하면 명령 기록, 경보 상태, Systems Manager 에이전트 상태 등과 같은 개별 인스턴스에 대한 자세한 정보를 볼 수 있습니다. 또한 인스턴스 탭은 Chef 레시피를 적용하거나, 인스턴스를 시작 또는 중지하거나, Auto Scaling 그룹에서 인스턴스를 추가 또는 제거하는 기능과 같은 다양한 작업을 제공합니다.

AWS OpsWorks Stacks 시작하기

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 특정 목적을 충족하는 스택을 생성하기 위해 혼합하고 일치시킬 수 있는 풍부한 사용자 지정 가능 구성 요소 세트를 제공합니다. 신규 사용자에게 도전 과제는 이러한 구성 요소

를 유효한 스택으로 결합하고 효과적으로 스택을 관리하는 방법을 이해하는 것입니다. 다음과 같이 시작할 수 있습니다.

다음을 수행하려는 경우	다음 연습을 완료:
최대한 빨리 샘플 스택을 생성	시작하기: 샘플
Linux 기반 스택을 사용하여 실험	시작하기: Linux
Windows 기반 스택을 사용하여 실험	시작하기: Windows
자체 Chef 쿡북을 생성하는 방법을 알아봅니다.	시작하기 - 쿡북

Amazon EC2 인스턴스 또는 자체 하드웨어에서 실행되는 온프레미스 인스턴스와 같은 기존 컴퓨팅 리소스가 있는 경우 스택을 사용하여 생성한 인스턴스와 함께 스택에 통합할 수 있습니다. AWS OpsWorks 그런 다음 AWS OpsWorks Stacks를 사용하여 모든 관련 인스턴스를 생성 방식에 관계없이 그룹으로 관리할 수 있습니다.

리전 지원

전역적으로 AWS OpsWorks Stacks에 액세스할 수 있으며 전역적으로 인스턴스를 생성하고 관리할 수도 있습니다. 사용자는 AWS GovCloud(미국 서부) 및 중국(베이징) AWS 리전을 제외한 모든 리전에서 AWS OpsWorks Stacks 인스턴스를 시작하도록 구성할 수 있습니다. AWS OpsWorks Stacks를 사용하려면 인스턴스가 다음 AWS OpsWorks Stacks 인스턴스 서비스 API 엔드포인트 중 하나에 연결할 수 있어야 합니다.

리소스는 생성된 리전에서만 관리할 수 있습니다. 한 리전 엔드포인트에서 생성되는 리소스는 다른 리전 엔드포인트에서 사용하거나 복제할 수 없습니다. 인스턴스를 다음 리전 중 하나에서 시작할 수 있습니다.

- US East (Ohio) Region
- 미국 동부(버지니아 북부) 리전
- US West (Oregon) Region
- US West (N. California) Region
- 캐나다(중부) 리전(API만 해당, AWS Management Console에서 생성된 스택에는 사용할 수 없음)
- Asia Pacific (Mumbai) Region

- Asia Pacific (Singapore) Region
- 아시아 태평양(시드니) 리전
- 아시아 태평양(도쿄) 리전
- Asia Pacific (Seoul) Region
- Europe (Frankfurt) Region
- Europe (Ireland) Region
- Europe (London) Region
- Europe (Paris) Region
- South America (São Paulo) Region

샘플 스택 시작하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 연습에서는 AWS OpsWorks Stacks를 사용하여 코드를 작성하지 않고 몇 번의 마우스 클릭만으로 샘플 Node.js 애플리케이션 환경을 빠르게 생성하는 방법을 보여줍니다. 마치고 나면 Chef 12를 실행하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, Node.js HTTP 서버 및 Twitter와 상호 작용하고 웹 페이지에 의견을 남기는 데 사용할 수 있는 웹 앱을 갖게 됩니다.

> **Note**
>
> 이 연습을 완료하면 c3.large 유형의 인스턴스가 자동으로 생성되므로 AWS 프리 티어에서는 이 안내나 AWS OpsWorks Stacks의 샘플 스택 생성 도구를 사용할 수 없습니다. VPC에서 샘플 스택 생성 도구를 사용하여 t2.medium 인스턴스를 생성해도 VPC를 AWS 프리 티어에서 현재 사용할 수 없습니다.

1단계: 사전 조건 완료

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

안내서를 시작하기 전에 다음 설정 단계를 완료해야 합니다. 이러한 설정 단계에는 AWS 계정 가입, 관리 사용자 생성, AWS OpsWorks Stacks에 액세스 권한 할당이 포함됩니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 서비스 액세스 권한 할당

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요. AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

서비스 액세스 권한 할당

역할 또는 사용자에게 `AWSOpsWorks_FullAccess` 및 `AmazonS3FullAccess` 권한을 추가하여 AWS OpsWorks Stacks 서비스(및 AWS OpsWorks Stacks가 의존하는 관련 서비스)에 대한 액세스를 활성화합니다.

권한 추가에 대한 자세한 내용은 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

이제 모든 설정 단계를 마쳤으므로 이 안내서를 시작할 수 있습니다.

2단계: 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 단계에서는 AWS OpsWorks Stacks 콘솔을 사용하여 스택을 생성합니다. 스택은 공통 용도가 있고 함께 관리하려는 인스턴스(예: Amazon EC2 인스턴스) 및 관련 AWS 리소스의 모음입니다. (자세한 내용은 스택 섹션을 참조하세요.) 이 안내서에는 인스턴스가 하나만 있습니다.

이 단계를 시작하기 전에 의 사전 요구 사항을 충족해야 합니다.

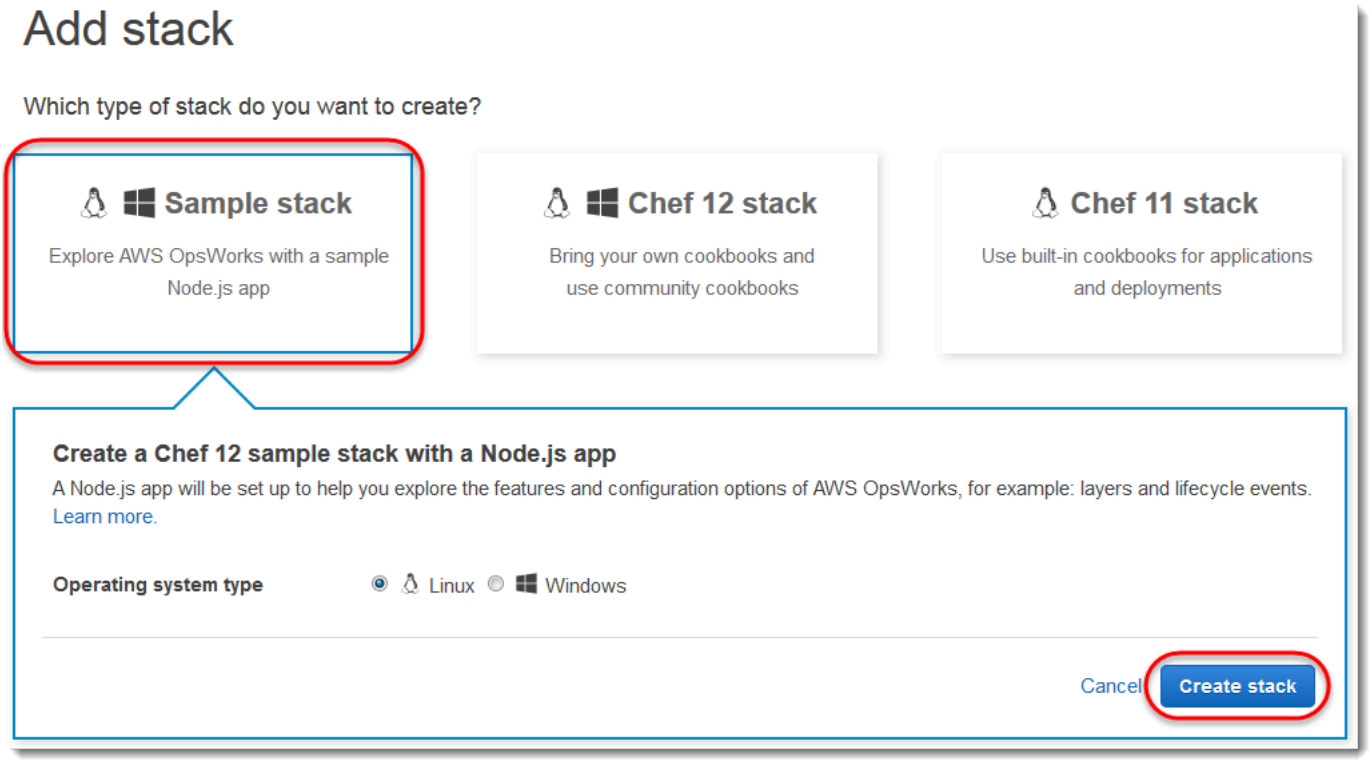
스택을 생성하는 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. 해당하는 경우 다음 중 하나를 수행하세요.

   - AWS OpsWorks 스택 시작 페이지가 표시되면 첫 번째 스택 추가 또는 첫 번째 AWS OpsWorks 스택 추가를 선택합니다(두 선택 모두 동일한 작업을 수행함). [스택 추가] 페이지가 표시됩니다.

- [OpsWorks 대시보드] 페이지가 표시되면 [스택 추가]를 선택합니다. [스택 추가] 페이지가 표시됩니다.

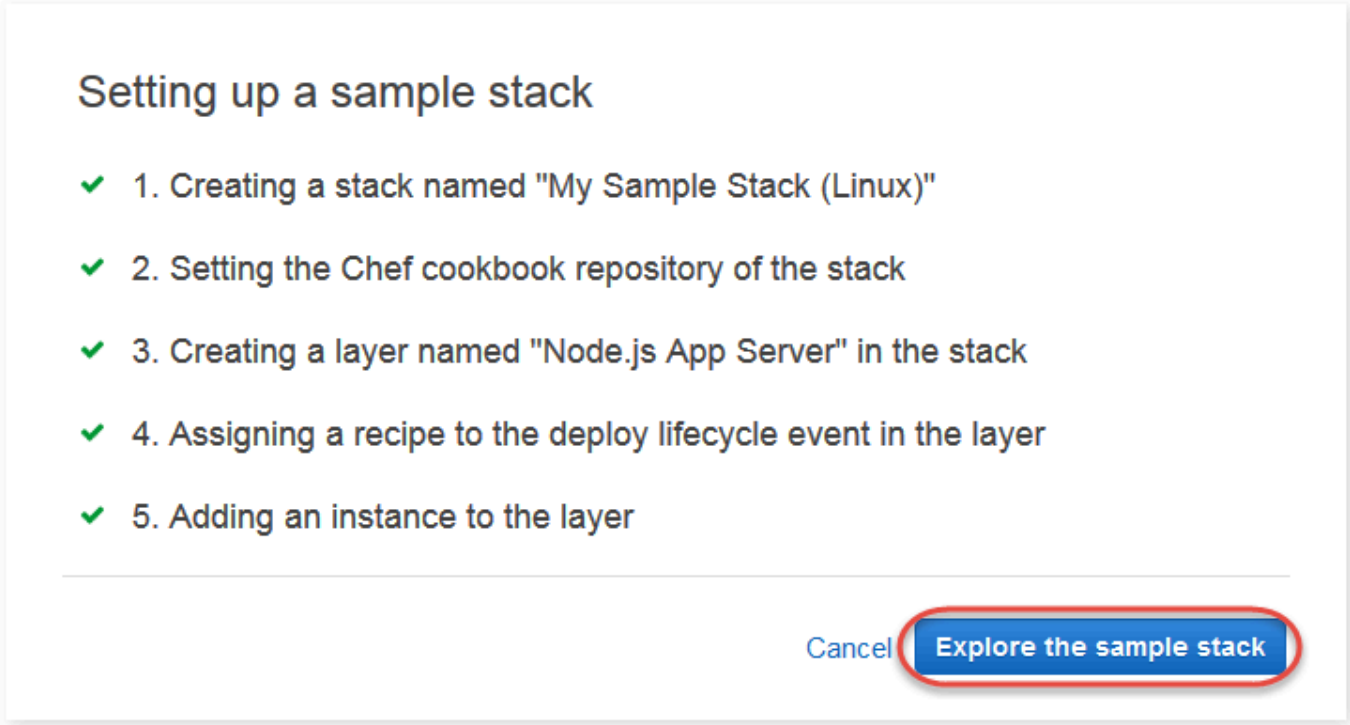
3. 표시된 [스택 추가] 페이지에서 [샘플 스택]이 선택되어 있지 않으면 이 옵션를 선택합니다.
4. 운영 체제 유형으로 이미 Linux를 선택한 상태에서 스택 생성을 선택합니다.



5. AWS OpsWorks Stacks는 My Sample Stack(Linux)이라는 스택을 생성합니다. AWS OpsWorks 또한 Stacks는 앱을 스택에 배포하는 데 필요한 모든 구성 요소를 추가합니다.

   - 계층: 인스턴스 세트의 청사진으로, 인스턴스의 설정, 리소스, 설치된 패키지, 보안 그룹 같은 정보를 지정합니다. (자세한 내용은 계층 섹션을 참조하세요.) 이 계층의 이름은 Node.js 앱 서버입니다.
   - 인스턴스: 이 경우에는 Amazon Linux 2 EC2 인스턴스입니다. 인스턴스에 대한 자세한 정보는 인스턴스 단원을 참조하세요. 이 인스턴스의 호스트 이름은 nodejs-server1입니다.
   - app: 인스턴스에서 실행되는 코드입니다. 앱에 대한 자세한 정보는 앱 단원을 참조하세요. 이 앱의 이름은 Node.js Sample App입니다.

6. AWS OpsWorks Stacks가 스택을 생성한 후 샘플 스택 탐색을 선택하여 내 샘플 스택(Linux) 페이지를 표시합니다(이 연습을 여러 번 완료하면 내 샘플 스택(Linux) 뒤에 2 또는 3과 같은 순차적 번호가 있을 수 있음).



다음 단계에서는 인스턴스를 시작하고 앱을 인스턴스에 배포합니다.

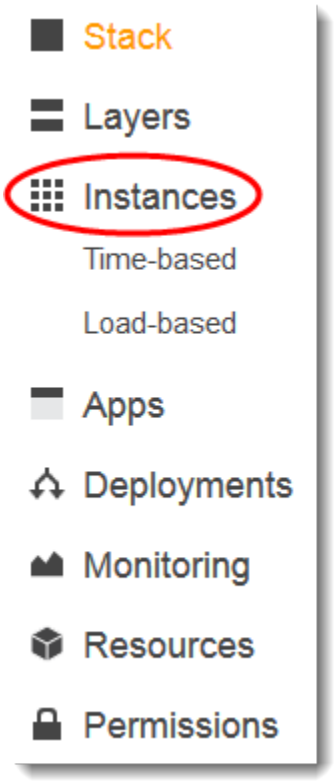
3단계: 인스턴스 시작 및 앱 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

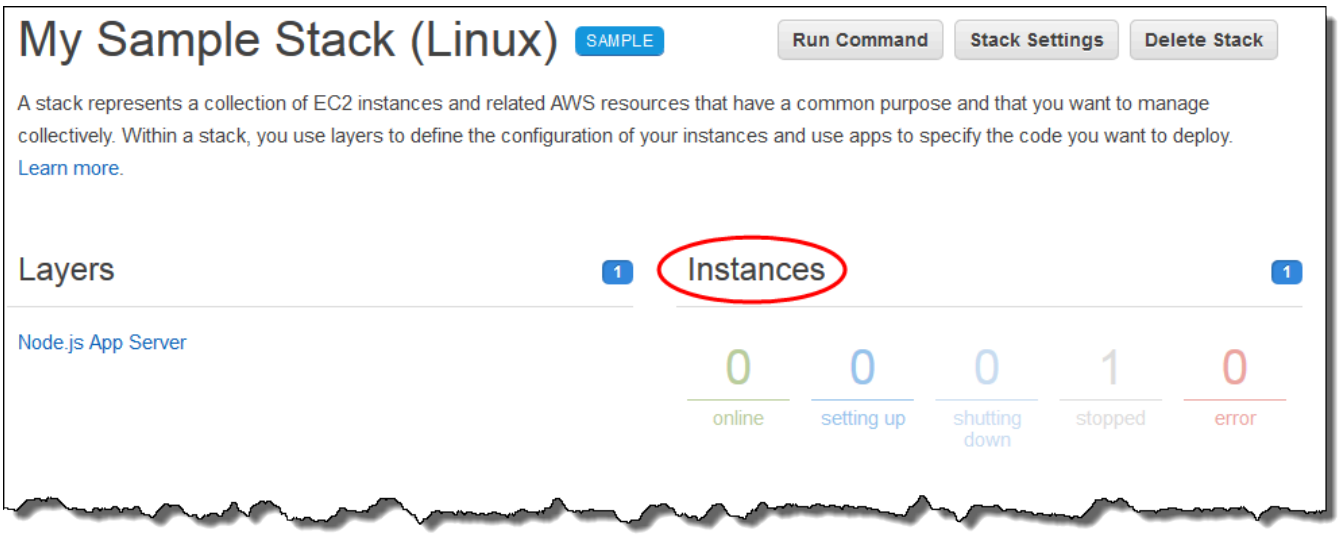
이제 인스턴스와 앱이 준비되었으므로 인스턴스를 시작하고 여기에 앱을 배포합니다.

인스턴스를 시작하고 앱을 배포하려면

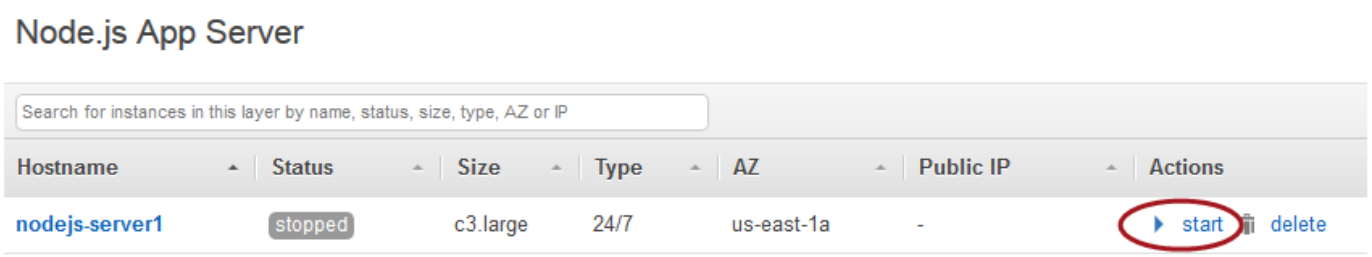
1. 다음 중 하나를 수행합니다.
   - 서비스 탐색 창에서 [인스턴스]를 선택합니다.



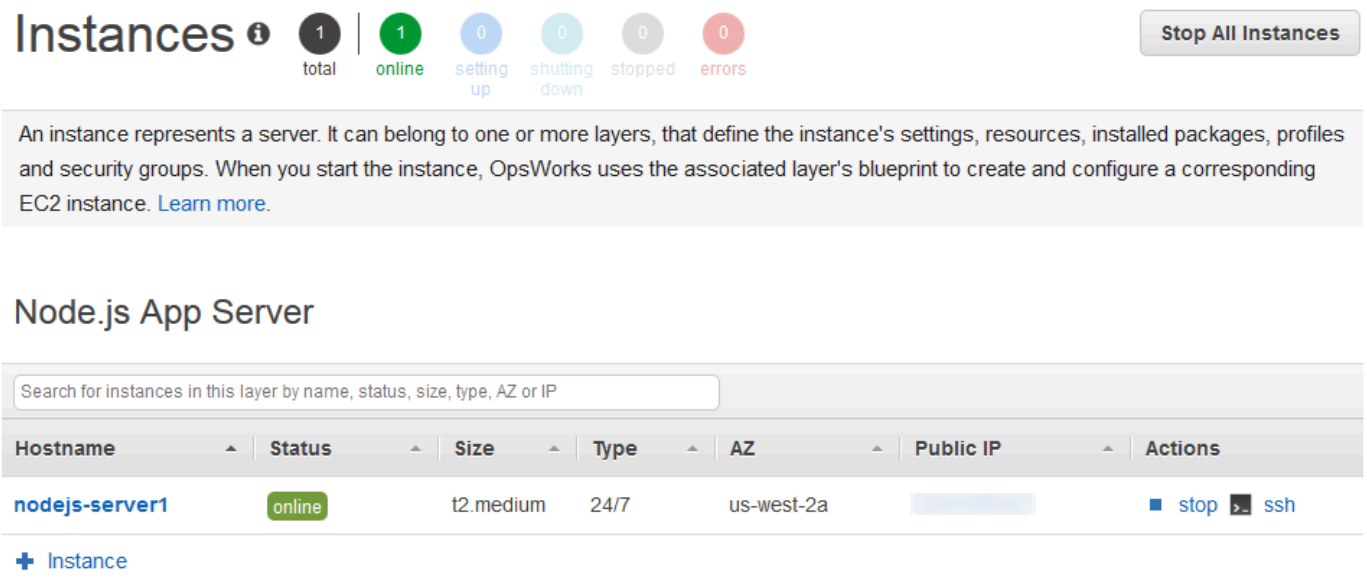
- [내 샘플 스택(Linux)] 페이지에서 [인스턴스]를 선택합니다.



2. [인스턴스] 페이지의 [Node.js 앱 서버]에서 [nodejs-server1]에 대해 [시작]을 선택합니다.



3. [온라인] 원이 밝은 녹색으로 변할 때까지 진행하지 마십시오. 실패 메시지가 표시되면 디버깅 및 문제 해결 안내서 단원을 참조하세요.
4. 인스턴스가 설정되면 AWS OpsWorks Stacks는 앱을 인스턴스에 배포합니다.
5. 결과가 다음 스크린샷과 비슷해야 계속 진행할 수 있습니다(실패 메시지가 표시되면 디버깅 및 문제 해결 안내서 단원 참조).



이제 인스턴스가 시작되고 여기에 앱이 배포되었습니다.

다음 단계에서는 인스턴스에서 앱을 테스트합니다.

4단계: 인스턴스에서 배포된 앱 테스트

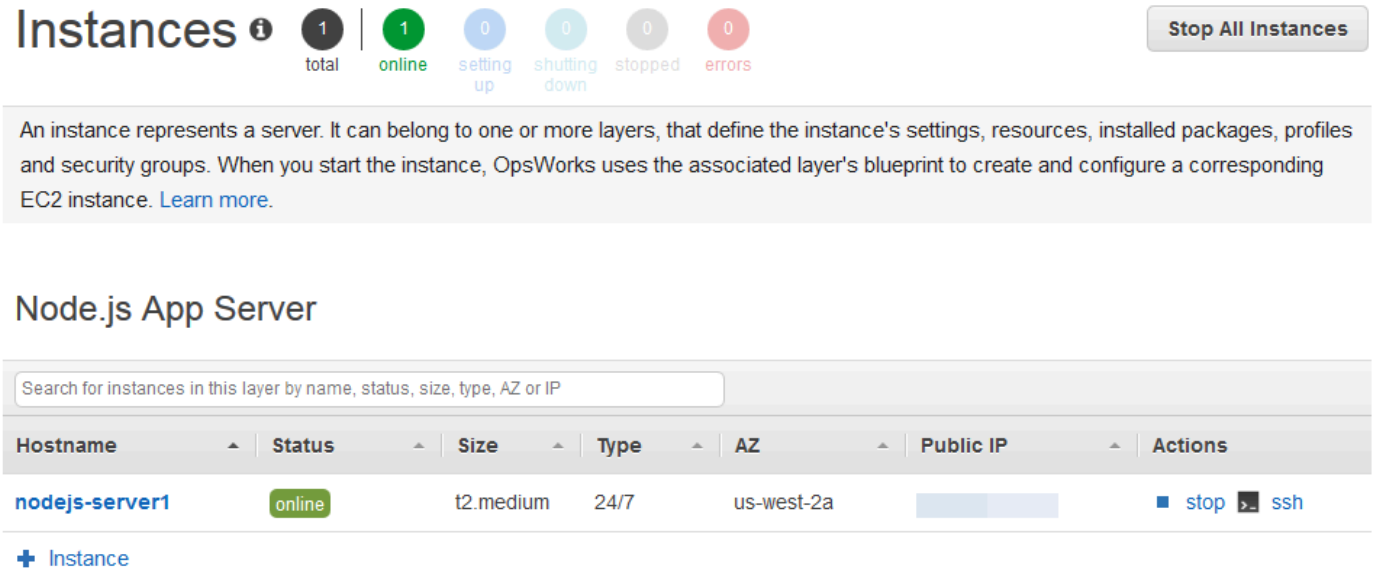
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에서 앱 배포 결과를 테스트합니다.

인스턴스에서 배포를 테스트하려면

1. 이전 단계에서 [인스턴스] 페이지가 열린 상태로 [Node.js 앱 서버], [nodejs-server1], [퍼블릭 IP]에 IP 주소를 선택합니다.



2. 성공을 알리는 웹 페이지의 [댓글 남기기] 텍스트 상자에 댓글을 입력한 다음 [보내기]를 선택하여 앱을 테스트합니다. 앱이 웹 페이지에 사용자의 댓글을 추가합니다. 원하는 수만큼 계속해서 댓글을 남기고 [보내기]를 선택합니다.

3. Twitter 계정이 있으면 [트윗] 또는 [팔로우 @AWSOpsWorks]를 선택하고 화면에 표시되는 지침에 따라 앱을 트윗하거나 @AWSOpsWorks를 팔로우하세요.

이제 인스턴스에 배포된 앱을 성공적으로 테스트했습니다.

나머지 단계에서 AWS OpsWorks 는 Stacks 콘솔을 사용하여 스택 및 해당 구성 요소의 설정을 탐색할 수 있습니다. 다음 단계에서는 스택의 설정을 살펴볼 수 있습니다.

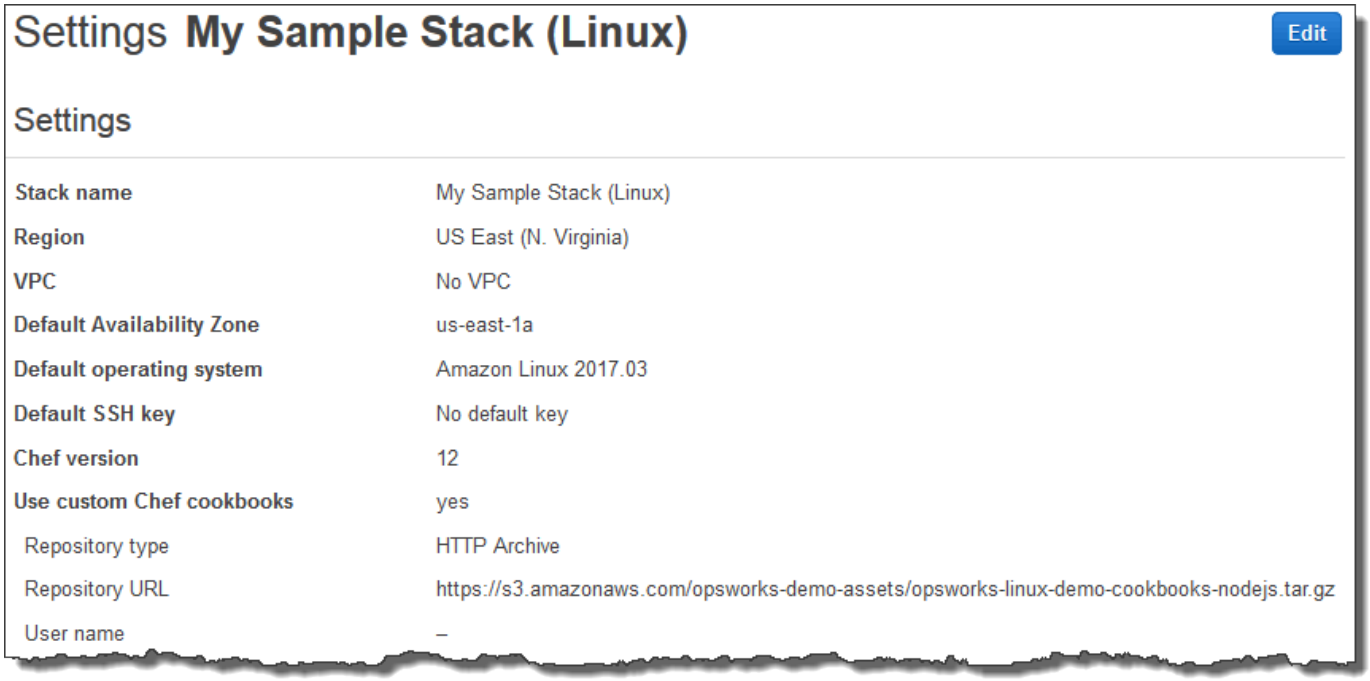
5단계: 스택 설정 탐색

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 스택을 설정하는 방법을 살펴봅니다.

스택의 설정을 표시하려면

1. 서비스 탐색 모음에서 [스택]을 선택합니다. [내 샘플 스택(Linux)] 페이지가 표시됩니다.
2. [스택 설정]를 선택합니다. [내 샘플 스택 설정(Linux)] 페이지가 표시됩니다.



여러 설정에 대해 자세히 알아보려면 [편집]을 선택한 후 마우스로 각 설정을 가리키십시오. (일부 설정은 화면에 설명이 표시되지 않습니다.) 이러한 설정에 대한 자세한 내용은 새 스택 생성 섹션을 참조하세요.

이 안내서에 사용된 Chef 쿡북을 살펴보려면 GitHub에서 opsworks-linux-demo-cookbooks-nodejs 리포지토리를 여십시오.

다음 단계에서는 계층의 설정을 살펴볼 수 있습니다.

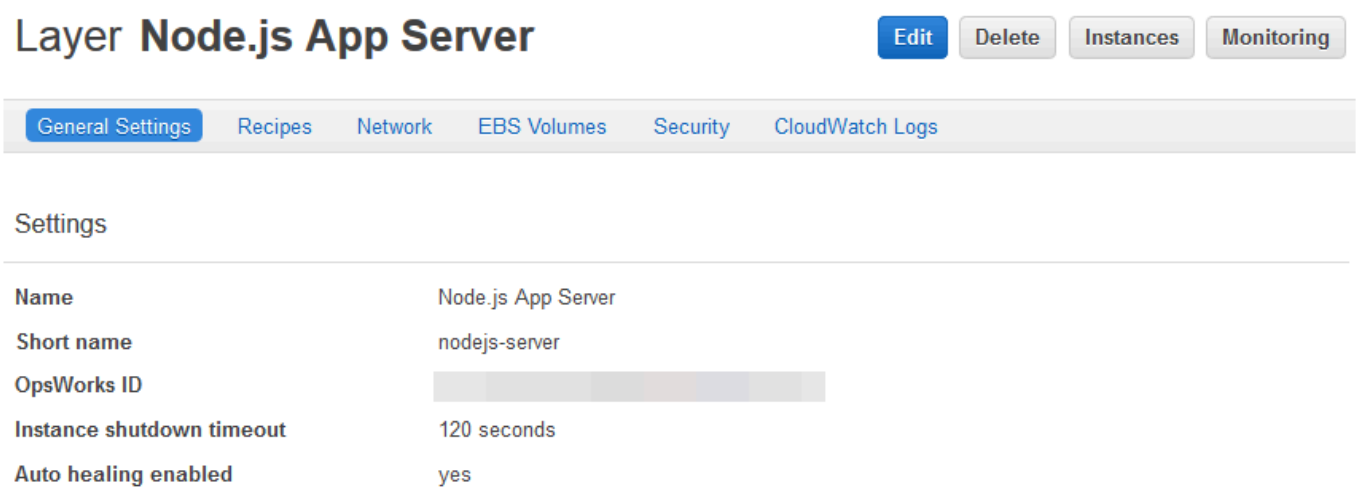
6단계: 계층 설정 탐색

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 계층을 설정하는 방법을 살펴봅니다.

계층의 설정을 표시하려면

1. 서비스 탐색 창에서 [계층]를 선택합니다. [계층] 페이지가 표시됩니다.
2. [Node.js 앱 서버]를 선택합니다. [계층 Node.js 앱 서버] 페이지가 표시됩니다. 계층의 설정을 보려면 [일반 설정], [레시피], [네트워크], [EBS 볼륨] 및 [보안]을 선택합니다.



여러 설정에 대해 자세히 알아보려면 [편집]을 선택한 후 마우스로 각 설정을 가리키십시오. (일부 설정은 화면에 설명이 표시되지 않습니다.) 이러한 설정에 대한 자세한 내용은 OpsWorks 계층의 구성 편집 섹션을 참조하세요.

다음 단계에서는 인스턴스의 설정 및 로그를 살펴볼 수 있습니다.

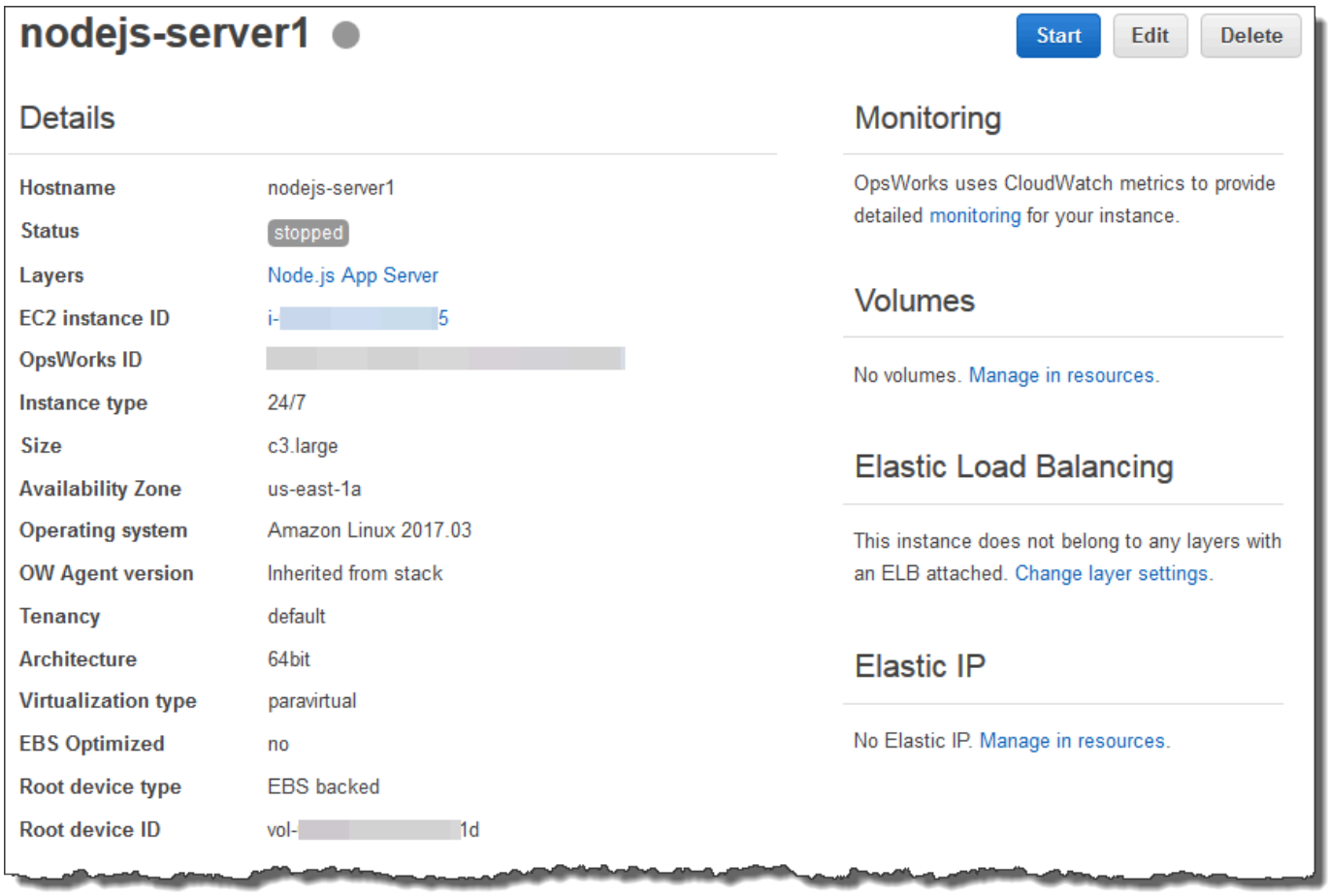
7단계: 인스턴스 설정 및 로그 탐색

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

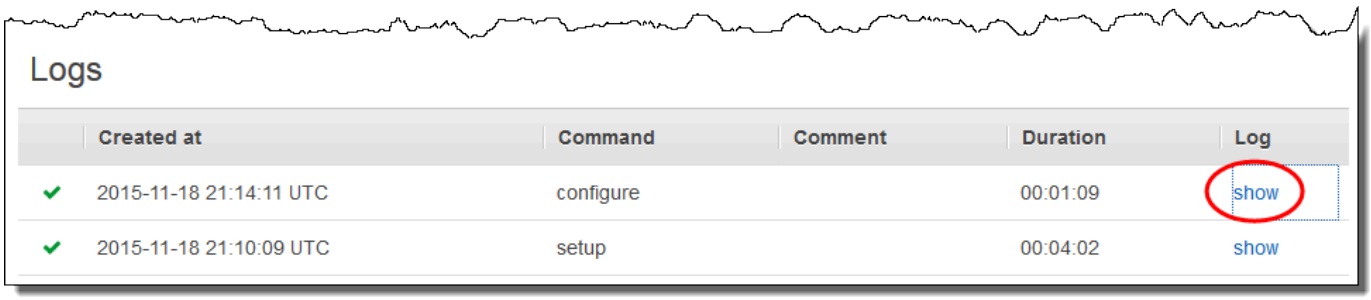
AWS OpsWorks Stacks가 인스턴스를 시작하는 데 사용한 설정을 검사합니다. AWS OpsWorks Stacks가 생성한 인스턴스 로그를 검사할 수도 있습니다.

인스턴스의 설정 및 로그를 표시하려면

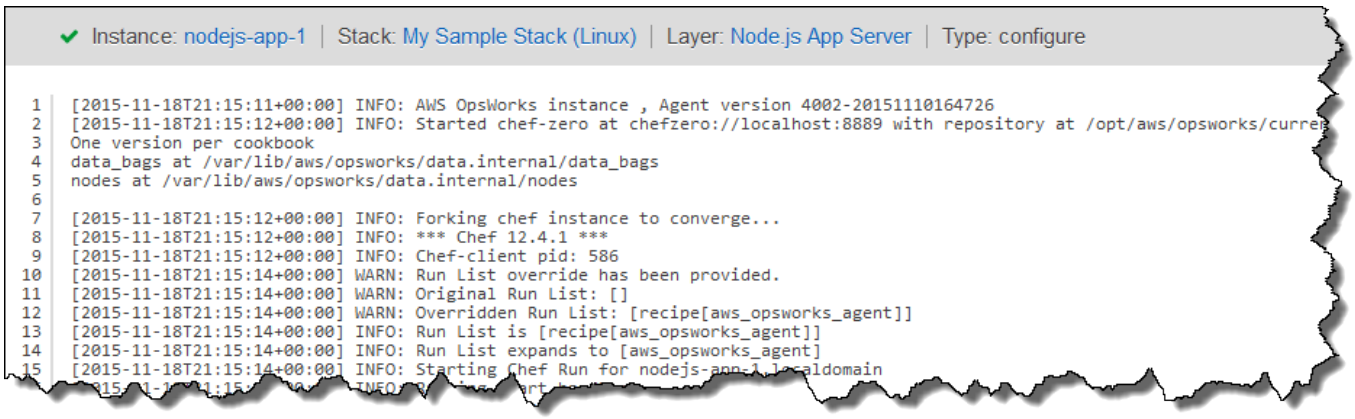
1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. [Node.js 앱 서버]에 대해 [nodejs-server1]을 선택합니다. 인스턴스의 속성 페이지가 표시됩니다.



3. 인스턴스 로그를 살펴보려면 [로그] 섹션의 [로그]에서 [표시]를 선택합니다.



4. AWS OpsWorks Stacks는 별도의 웹 브라우저 탭에 로그를 표시합니다.



인스턴스 설정 중 일부의 의미를 자세히 알아보려면 [nodejs-server1] 페이지로 돌아가서 [중지]를 선택하고 확인 메시지가 나타나면 [중지]를 선택합니다. 상태가 중지 중에서 중지됨으로 변경된 후 편집을 선택한 다음 각 설정에 마우스를 갖다 댑니다. (일부 설정은 화면에 설명이 표시되지 않습니다.) 이러한 설정에 대한 자세한 내용은 계층에 인스턴스 추가 섹션을 참조하세요.

설정 검토가 완료되었으면 [시작]을 선택하여 인스턴스를 다시 시작하고 [상태]가 [온라인]으로 전환될 때까지 기다립니다. 그렇지 않으면 인스턴스가 중지 상태를 유지하므로 나중에 앱을 테스트할 수 없게 됩니다.

> **Note**
>
> 인스턴스에 로그인하여 더 자세히 살펴보려면 먼저 퍼블릭 SSH 키(ssh-keygen 또는 PuTTYgen과 같은 도구를 사용하여 생성할 수 있음)에 대한 정보를 AWS OpsWorks Stacks에 제공한 다음 내 샘플 스택(Linux) 스택에 대한 권한을 설정하여 사용자가 인스턴스에 로그인할 수 있도록 해야 합니다. 지침은 사용자의 퍼블릭 SSH 키 등록 및 SSH를 사용하여 로그인 단원을 참조하세요.

다음 단계에서는 앱의 설정을 살펴볼 수 있습니다.

8단계: 앱 설정 탐색

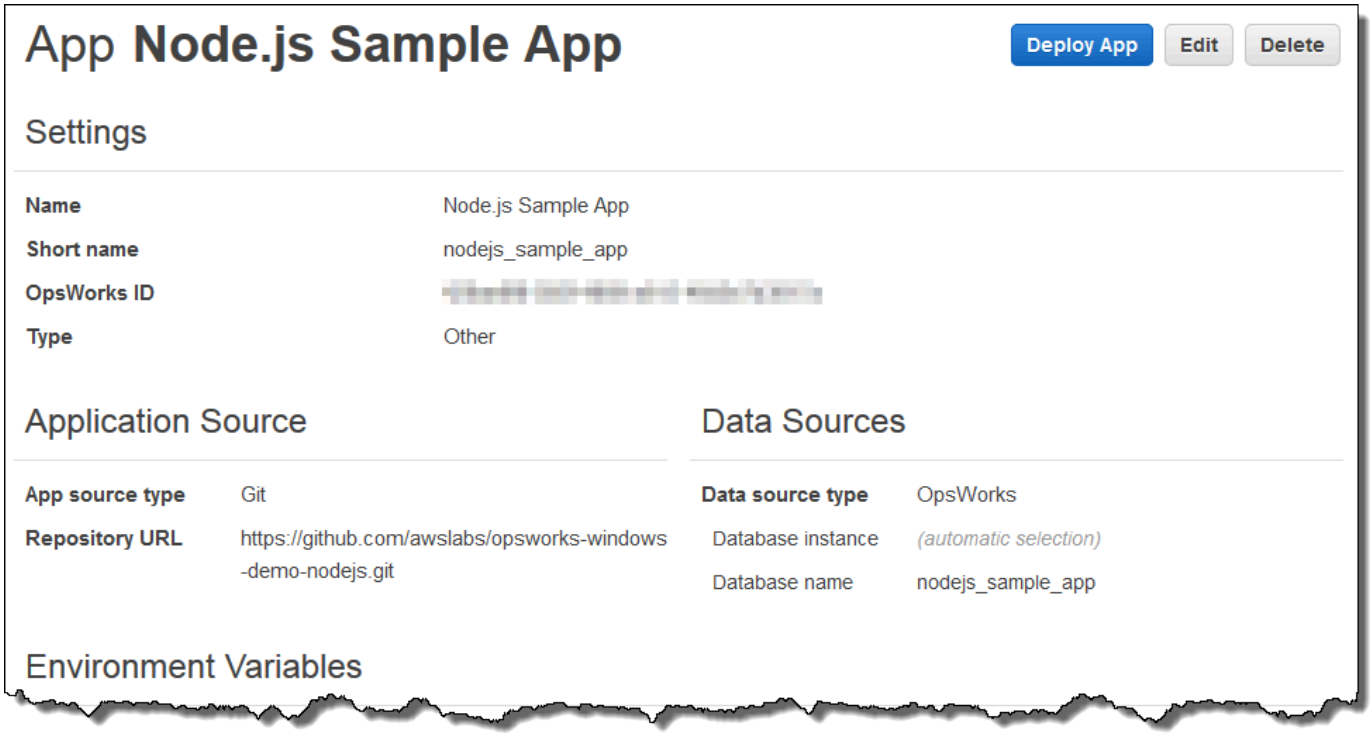
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 앱에 사용한 설정을 검사합니다.

앱의 설정을 표시하려면

1. 서비스 탐색 창에서 [앱]을 선택합니다. [앱] 페이지가 표시됩니다.
2. [Node.js 샘플 앱]을 선택합니다. [앱 Node.js 샘플 앱] 페이지가 표시됩니다.



설정 중 일부의 의미를 자세히 알아보려면 [편집]을 선택한 후 마우스로 각 설정을 가리키십시오. (일부 설정은 화면에 설명이 표시되지 않습니다.) 이러한 설정에 대한 자세한 정보는 앱 추가 단원을 참조하세요.

다음 단계에서는 계층 모니터링 보고서를 살펴볼 수 있습니다.

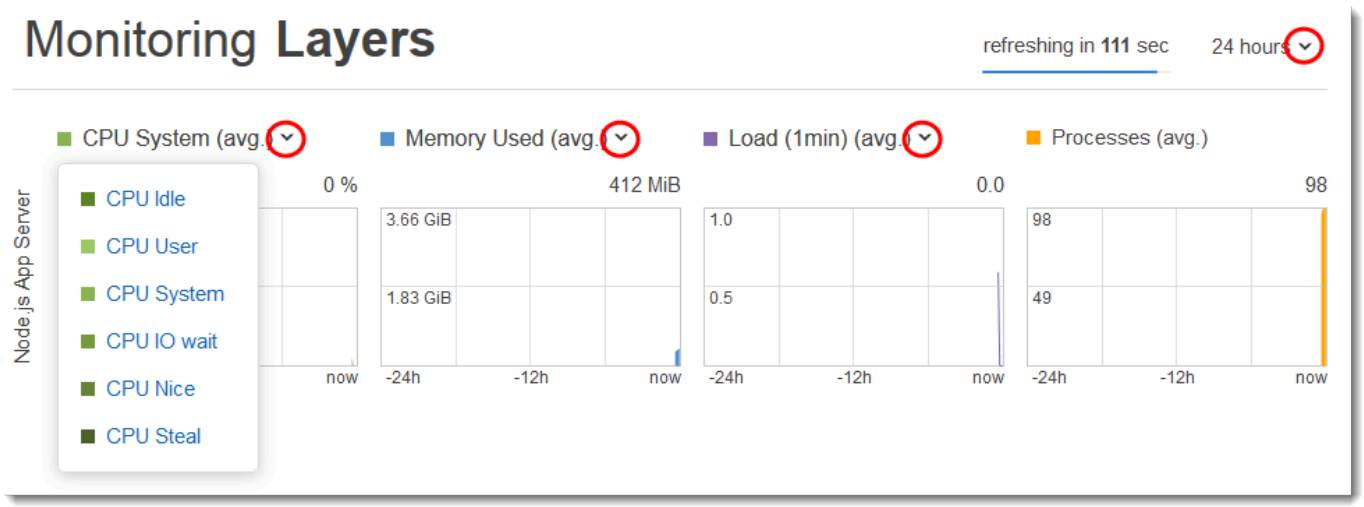
9단계: 계층 모니터링 보고서 탐색

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 계층의 컴퓨팅 성능에 대해 생성하는 보고서를 검토합니다.

계층 모니터링 보고서를 표시하려면

1. 서비스 탐색 창에서 [모니터링]을 선택합니다. [모니터링 계층] 페이지가 표시됩니다.
2. 다른 보기를 더 살펴보려면 [CPU], [메모리], [로드] 및 시간 옆의 화살표를 선택합니다.



이들 보고서 및 기타 보고서에 대한 자세한 정보는 Amazon CloudWatch 사용 및 모니터링 단원을 참조하세요.

다음 단계에서는 추가 스택 설정을 살펴볼 수 있습니다.

10단계: 추가 스택 설정 탐색

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 단계에서는 추가 스택 설정을 살펴볼 수 있습니다.

AWS OpsWorks Stacks는 별도의 배포를 실행하지 않았고, 추가 리소스를 프로비저닝하지 않았으며, 이 스택의 일부로 추가 권한을 조정하지 않았으므로 배포 및 명령, 리소스 및 권한 페이지에는 관심이 없습니다. 어쨌든 이러한 설정을 보려면 서비스 탐색 창에서 [배포], [리소스] 및 [권한]을 각각 선택합니다. 이러한 페이지의 의미를 자세히 알아보려면 앱 배포, 리소스 관리 및 사용자 권한 관리 단원을 참조하세요.

다음 단계에서는이 연습에 사용한 AWS 리소스를 정리할 수 있습니다. 이 다음 단계는 선택 사항입니다. AWS OpsWorks Stacks에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용할 수 있습니다. 그러나 이러한 AWS 리소스를 주변에 보관하면 AWS 계정에 요금이 계속 부과될 수 있습니다. 나중에 사용할 수 있도록 이러한 AWS 리소스를 계속 유지하려면 이제이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

11단계(선택 사항): 정리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 계정에 추가 요금이 발생하지 않도록 인스턴스 및 AWS OpsWorks Stacks 스택을 포함하여이 연습에 사용된 앱과 AWS 리소스를 삭제할 수 있습니다. (자세한 내용은 AWS OpsWorks 요금을 참조하세요.) 그러나 AWS OpsWorks Stacks에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용하고 싶을 수 있습니다. 이러한 AWS 리소스를 계속 사용하려면 이제이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

이번 연습에서 생성한 리소스에 보관된 콘텐츠는 개인 식별 정보를 포함할 수 있습니다. 이 정보를 AWS에 저장하고 싶지 않다면, 이 주제에서 설명하는 다음 단계를 따르십시오.

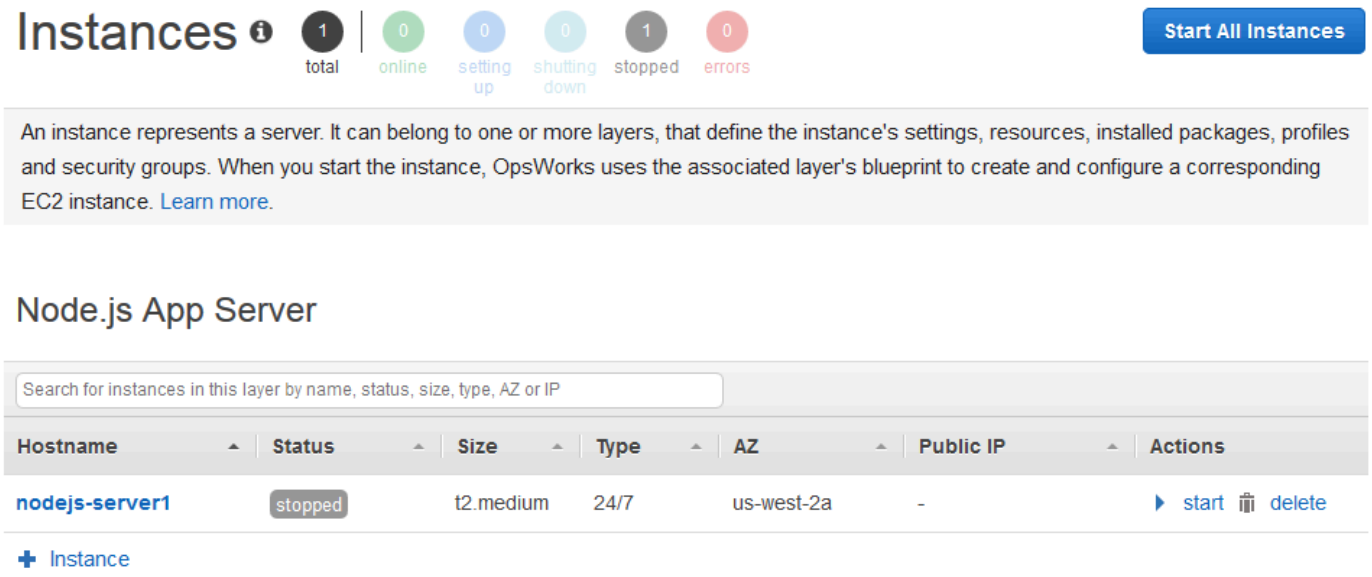
스택에서 앱을 삭제하려면

1. 서비스 탐색 창에서 [앱]을 선택합니다. [앱] 페이지가 표시됩니다.
2. [Node.js 샘플 앱]의 [작업]에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다. 앱이 삭제되면 [앱 없음] 메시지가 표시됩니다.

스택의 인스턴스를 삭제하려면

1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. [Node.js 앱 서버]에 있는 [nodejs-server1]의 [작업]에서 [중지]를 선택합니다. 확인 메시지가 표시되면 [중지]를 선택합니다.

   이 프로세스는 몇 분 정도 걸릴 수 있습니다. AWS OpsWorks Stacks가 완료되면 다음 결과가 표시됩니다.



3. 작업에 대해 삭제를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다. 인스턴스가 삭제되면 [인스턴스 없음] 메시지가 표시됩니다.

스택을 삭제하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [내 샘플 스택(Linux)] 페이지가 표시됩니다.

2. [스택 삭제]를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다. 스택이 삭제되고 [OpsWorks 대시보드] 페이지가 표시됩니다.

선택적으로 다른 AWS 서비스 및 EC2 인스턴스에 액세스하는 데 재사용하지 않으려는 경우이 연습에 사용한 사용자 및 Amazon EC2 키 페어를 삭제할 수 있습니다. 지침은 IAM 사용자 및 Amazon EC2 키 페어 및 Linux 인스턴스 삭제를 참조하세요.

이제 이 안내서를 성공적으로 완료했습니다. 자세한 내용은 다음 단계 단원을 참조하십시오.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제이 연습을 완료했으므로 AWS OpsWorks Stacks 사용에 대해 자세히 알아볼 수 있습니다.

- Stacks를 사용하여 직접이 AWS OpsWorks 스택을 수동으로 다시 생성하는 방법을 연습합니다. 시작하기: Linux을(를) 참조하세요.
- Stacks가이 연습에 사용한 AWS OpsWorks 쿡북과 앱을 살펴봅니다. 시작하기: Linux 안내서의 자세히 알아보기: 이 안내서에서 사용한 쿡북 살펴보기 및 자세히 알아보기: 이 안내서에서 사용한 앱 살펴보기 단원을 참조하세요.
- Windows 인스턴스에서 AWS OpsWorks Stacks 사용을 연습합니다. 시작하기: Windows을(를) 참조하세요.
- 새 스택 생성 학습을 통해 스택에 대해 자세히 알아봅니다.
- OpsWorks 계층의 구성 편집를 통해 계층에 대해 자세히 알아봅니다.
- 계층에 인스턴스 추가를 통해 인스턴스에 대해 자세히 알아봅니다.
- 앱 배포를 통해 앱에 대해 자세히 알아봅니다.
- 쿡북과 레시피 단원에 대해 자세히 알아보세요.
- 자체 쿡북을 생성합니다. 시작하기 - 쿡북을(를) 참조하세요.
- 보안 및 권한을 사용하여 스택에 대한 액세스를 제어하는 방법을 알아봅니다.

Linux 스택 시작하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 연습에서는 AWS OpsWorks Stacks를 사용하여 Node.js 애플리케이션 환경을 생성하는 방법을 알아봅니다. 마치고 나면 Chef 12를 실행하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, Node.js HTTP 서버 및 Twitter와 상호 작용하고 웹 페이지에 의견을 남기는 데 사용할 수 있는 웹 앱을 갖게 됩니다.

Chef는 EC2 인스턴스 같은 서버의 구성 및 유지 관리와 이러한 서버에서의 앱 배포 및 유지 관리 방식을 위한 타사 프레임워크입니다. Chef에 익숙하지 않은 경우이 연습을 완료한 후 AWS OpsWorks Stacks가 제공하는 모든 기능을 최대한 활용할 수 있도록 Chef에 대해 자세히 알아보는 것이 좋습니다. (자세한 정보는 Learn Chef 웹 사이트를 참조하세요.)

AWS OpsWorks Stacks는 Amazon Linux, Ubuntu Server, CentOS, Red Hat Enterprise Linux의 네 가지 Linux 배포를 지원합니다. 이 연습에서는 Ubuntu Server를 사용합니다. AWS OpsWorks Stacks는 Windows Server에서도 작동합니다. Windows Server 스택에 대해 동등한 연습이 있지만 먼저이 연습을 완료하여 반복되지 않는 AWS OpsWorks Stacks 및 Chef에 대한 기본 개념을 알아보는 것이 좋습니다. 이 안내서를 완료한 후 시작하기: Windows 안내서를 참조하세요.

주제

- 1단계: 사전 조건 완료
- 2단계: 스택 생성
- 3단계: 스택에 계층 추가
- 4단계: 인스턴스에 배포할 앱 지정
- 5단계: 인스턴스 시작
- 6단계: 인스턴스에 앱 배포
- 7단계: 인스턴스에 배포된 앱 테스트
- 8단계(선택 사항): 정리
- 다음 단계

- 자세히 알아보기: 이 안내서에서 사용한 쿡북 살펴보기
- 자세히 알아보기: 이 안내서에서 사용한 앱 살펴보기

1단계: 사전 조건 완료

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

안내서를 시작하기 전에 다음 설정 단계를 완료합니다. 이러한 설정 단계에는 AWS 계정 가입, 관리 사용자 생성, AWS OpsWorks Stacks에 액세스 권한 할당이 포함됩니다.

이미 시작하기: 샘플 안내서를 마쳤다면 이 안내서의 사전 조건을 충족한 것이며, 2단계: 스택 생성 단원으로 건너뛸 수 있습니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 서비스 액세스 권한 할당

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스

권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

서비스 액세스 권한 할당

역할 또는 사용자에게 `AWSOpsWorks_FullAccess` 및 `AmazonS3FullAccess` 권한을 추가하여 AWS OpsWorks Stacks 서비스(및 AWS OpsWorks Stacks가 의존하는 관련 서비스)에 대한 액세스를 활성화합니다.

권한 추가에 대한 자세한 내용은 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

이제 모든 설정 단계를 마쳤으므로 이 안내서를 시작할 수 있습니다.

2단계: 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 콘솔을 사용하여 스택을 생성합니다. 스택은 공통 용도가 있고 함께 관리하려는 인스턴스 및 관련 AWS 리소스의 모음입니다. (자세한 내용은 스택 섹션을 참조하세요.) 이 안내서에는 인스턴스가 하나만 있습니다.

시작에 앞서 아직 완료하지 않았다면 사전 조건을 완료하세요.

스택을 생성하는 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. 해당하는 경우 다음 중 하나를 수행하세요.
   - AWS OpsWorks 스택 시작 페이지가 표시되면 첫 번째 스택 추가 또는 첫 번째 AWS OpsWorks 스택 추가를 선택합니다(두 선택 모두 동일한 작업을 수행함). [스택 추가] 페이지가 표시됩니다.
   - [OpsWorks 대시보드] 페이지가 표시되면 [스택 추가]를 선택합니다. [스택 추가] 페이지가 표시됩니다.
3. [스택 추가] 페이지가 표시되었는데 아직 선택되어 있지 않은 경우 [Chef 12 스택]을 선택합니다.
4. 스택 이름 상자에 이름을 입력합니다(예: **`MyLinuxDemoStack`**). 다른 이름을 입력할 수 있는데 이 연습 전체에서 `MyLinuxDemoStack`을 해당 이름으로 바꿔야 합니다.
5. 리전에서 미국 서부(오레곤)를 선택합니다.
6. VPC의 경우 다음 중 하나를 수행합니다.
   - VPC를 사용할 수 있는 경우 선택합니다. (자세한 내용은 VPC에서 스택 실행 섹션을 참조하세요.)
   - 그렇지 않은 경우 [VPC 없음]을 선택합니다.
7. 기본 운영 체제에서 Linux 및 Ubuntu 18.04 LTS를 선택합니다.
8. [사용자 지정 Chef 쿡북 사용]에 대해 [예]를 선택합니다.
9. [리포지토리 유형]으로 [Http 아카이브]를 선택합니다.
10. 리포지토리 URL로 **`https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`**를 입력합니다.
11. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.
    - [기본 가용 영역]([us-west-2a])
    - [기본 SSH 키]([기본 SSH 키 사용 금지])
    - [사용자 이름](비워 둠)
    - [암호](비워 둠)
    - [스택 색상](짙은 파란색)
12. 고급을 선택합니다.
13. IAM 역할의 경우 다음 작업 중 하나를 수행합니다(자세한 정보는 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 참조).

- [aws-opsworks-service-role]을 사용할 수 있는 경우 선택합니다.
- aws-opsworks-service-role을 사용할 수 없는 경우 새 IAM 역할을 선택합니다.

14. 기본 IAM 인스턴스 프로파일의 경우 다음 중 하나를 수행합니다(자세한 정보는 EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 참조).

    - [aws-opsworks-ec2-role]을 사용할 수 있는 경우 선택합니다.
    - aws-opsworks-ec2-role을 사용할 수 없는 경우 새 IAM 인스턴스 프로파일을 선택합니다.

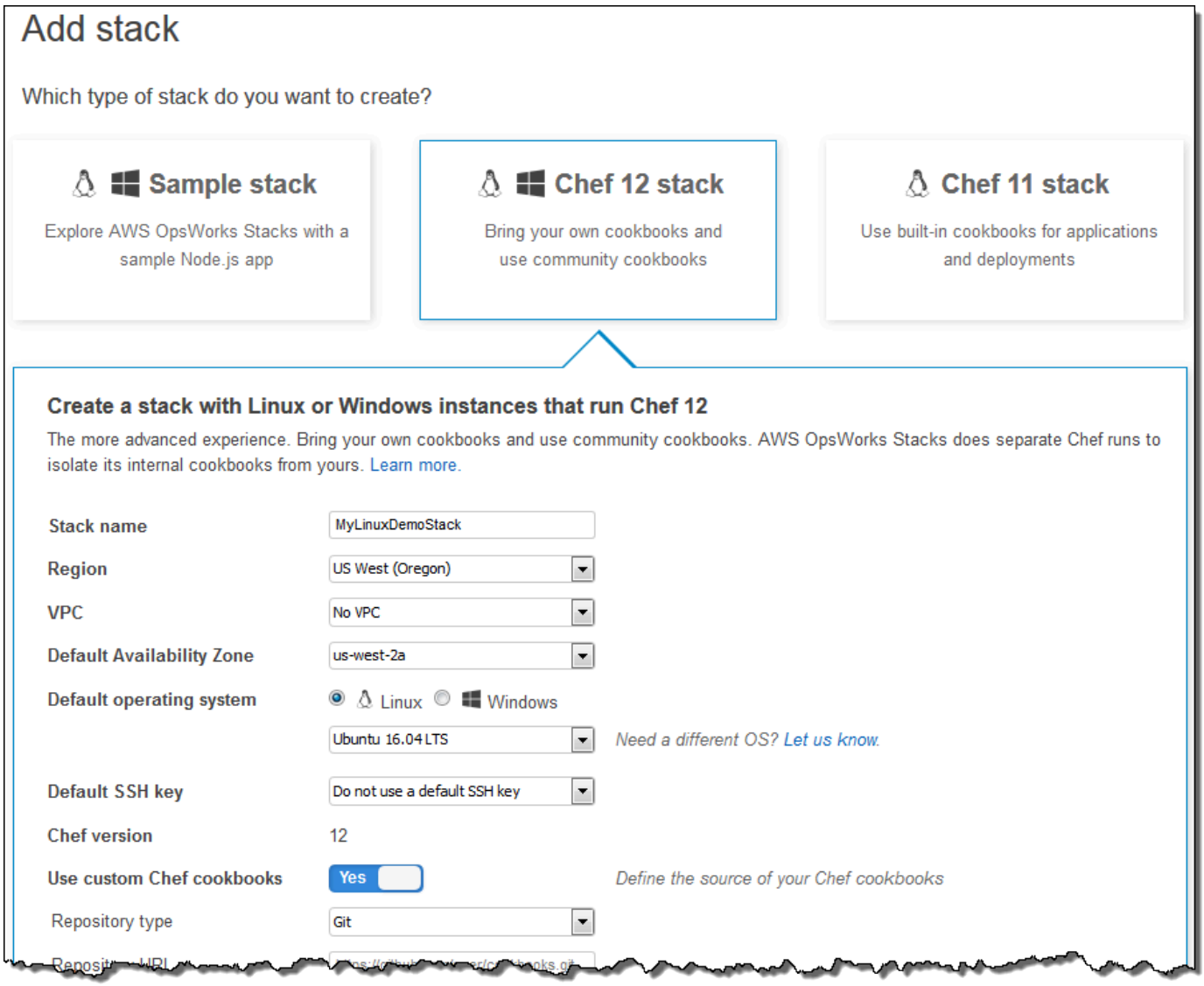
15. [API 엔드포인트 리전]의 경우 스택과 연결하려는 리전 API 엔드포인트를 선택합니다. 스택의 리전을 미국 동부(버지니아 북부) 리전 엔드포인트 내의 미국 서부(오레곤) 리전으로 설정하려면 us-east-1을 선택합니다. 스택의 리전을 미국 서부(오레곤) 리전으로 설정하고 스택을 미국 서부(오레곤) 리전 엔드포인트와 연결하려면 us-west-2를 선택합니다.

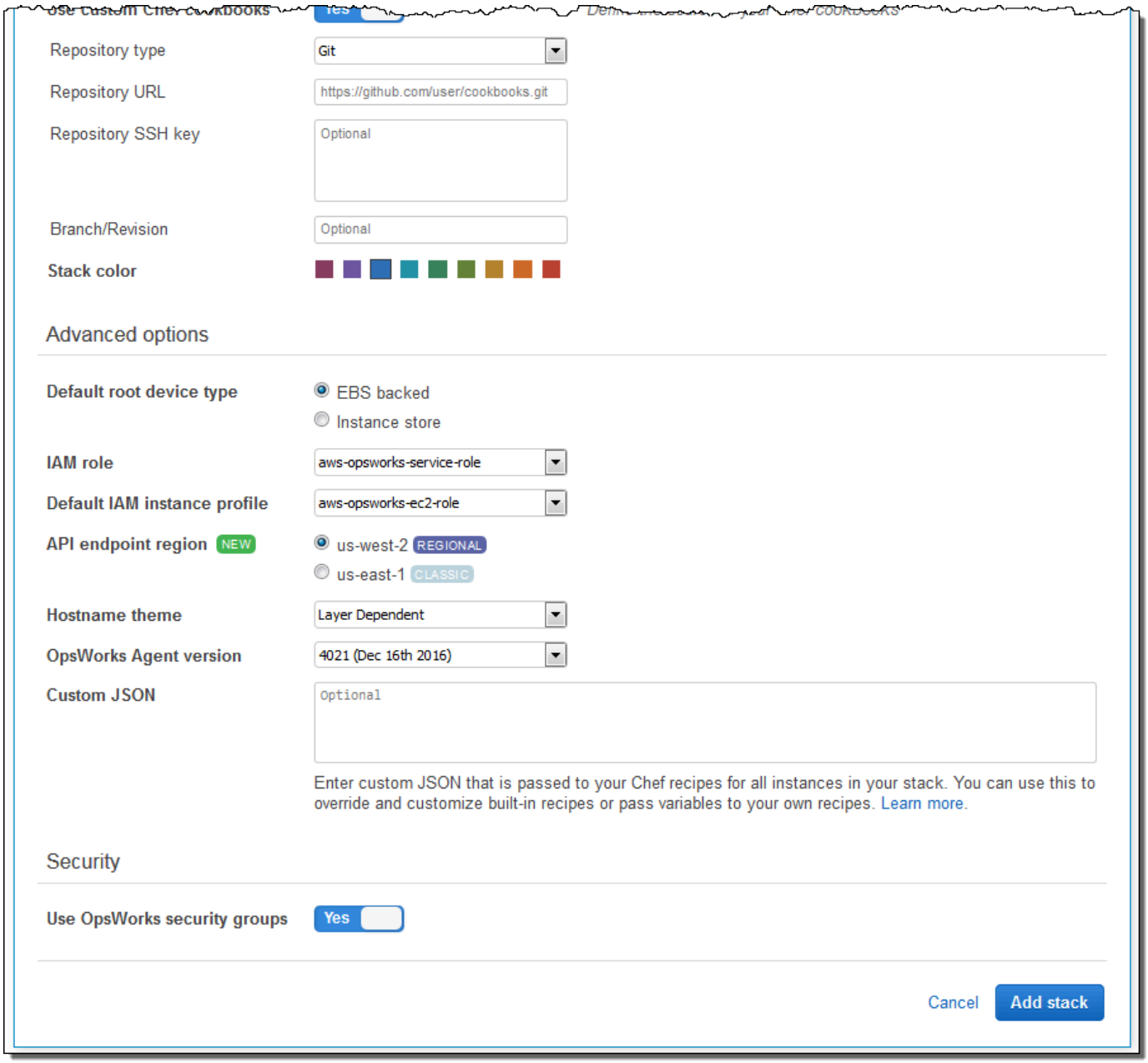
    > **Note**
    >
    > 미국 동부(버지니아 북부) 리전 엔드포인트에는 이전 버전과의 호환성을 AWS 리전 위해 이전 버전이 포함되어 있지만 관리하는 위치와 가장 가까운 리전 엔드포인트를 선택하는 것이 좋습니다 AWS. 자세한 내용은 리전 지원 단원을 참조하십시오.

16. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.

    - [기본 루트 디바이스 유형]([EBS 지원])
    - [호스트 이름 테마]([계층 종속적])
    - [OpsWorks Agent 버전](최신 버전)
    - [사용자 지정 JSON](비워 둠)
    - [OpsWorks 보안 그룹 사용]([예])

17. 결과는 VPC, IAM 역할 및 기본 IAM 인스턴스 프로파일을 제외하고 다음 스크린샷과 일치해야 합니다.


Add stack
Which type of stack do you want to create?
Sample stack
Explore AWS OpsWorks Stacks with a sample Node.js app
Chef 12 stack
Bring your own cookbooks and use community cookbooks
Chef 11 stack
Use built-in cookbooks for applications and deployments
Create a stack with Linux or Windows instances that run Chef 12
The more advanced experience. Bring your own cookbooks and use community cookbooks. AWS OpsWorks Stacks does separate Chef runs to isolate its internal cookbooks from yours. Learn more.
Stack name
MyLinuxDemoStack
Region
US West (Oregon)
VPC
No VPC
Default Availability Zone
us-west-2a
Default operating system
Linux
Windows
Ubuntu 16.04 LTS
Need a different OS? Let us know.
Default SSH key
Do not use a default SSH key
Chef version
12
Use custom Chef cookbooks
Yes
Define the source of your Chef cookbooks
Repository type
Git



18. 스택 추가를 선택합니다. AWS OpsWorks 스택은 스택을 생성하고 MyLinuxDemoStack 페이지를 표시합니다.

이제 이 안내서를 위한 최신 설정을 갖춘 스택이 생겼습니다.

다음 단계에서는 스택에 계층을 추가합니다.

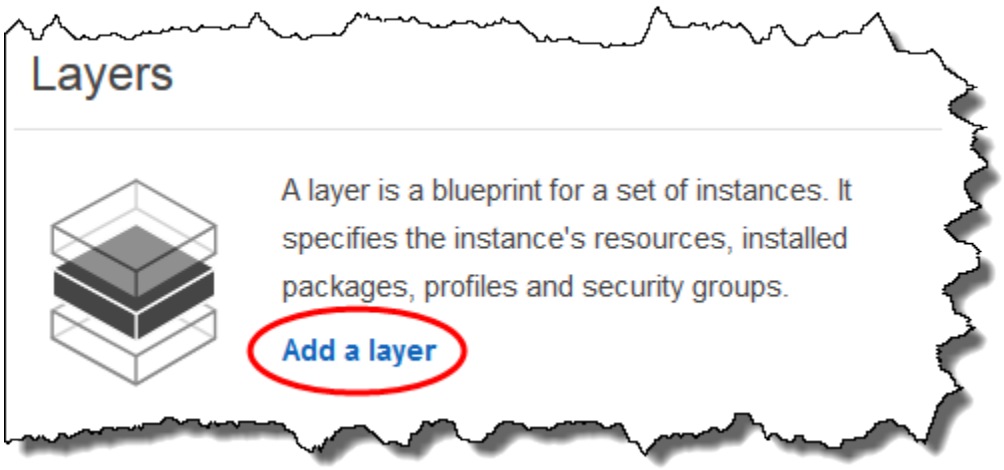
3단계: 스택에 계층 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

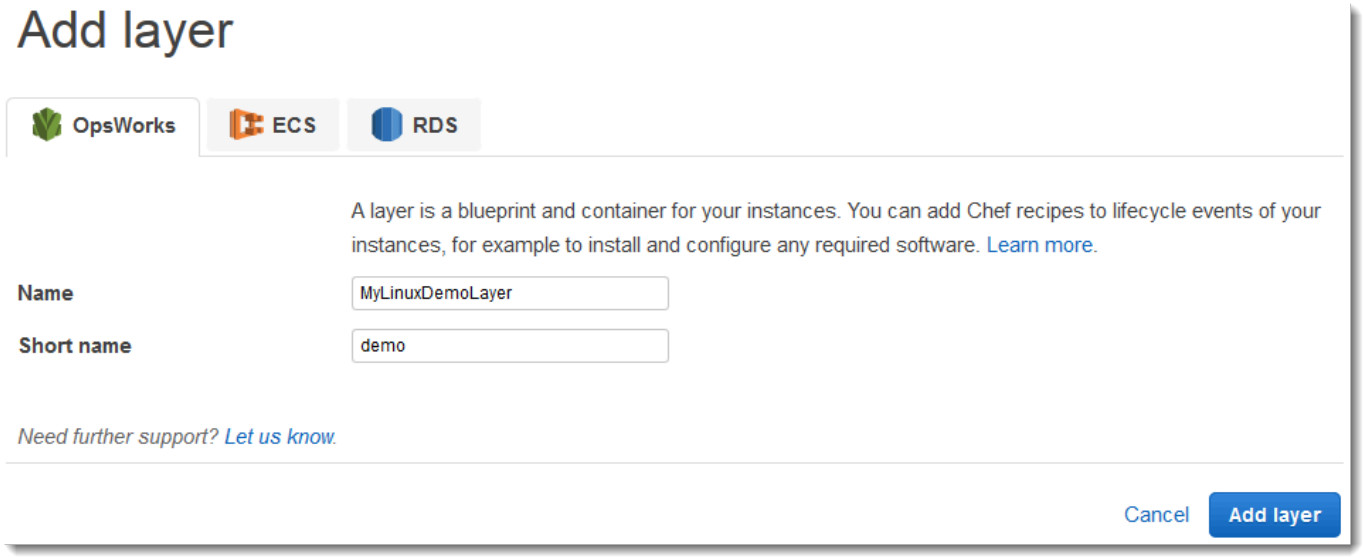
계층은 Amazon EC2 인스턴스와 같은 인스턴스 세트의 청사진입니다. 계층은 인스턴스의 설정, 리소스, 설치된 패키지, 보안 그룹 같은 정보를 지정합니다. 다음으로 계층을 스택에 추가합니다. (계층에 대한 자세한 정보는 계층 단원을 참조하세요.)

스택에 계층을 추가하려면

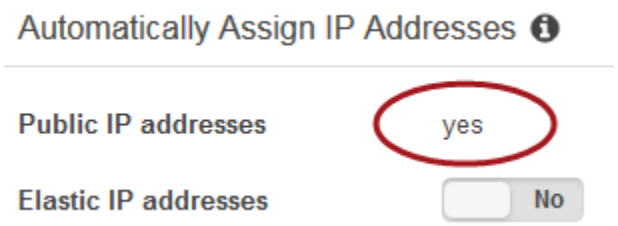
1. 이전 단계의 [MyLinuxDemoStack] 페이지가 표시된 상태에서 [계층]에 대해 [계층 추가]를 선택합니다.



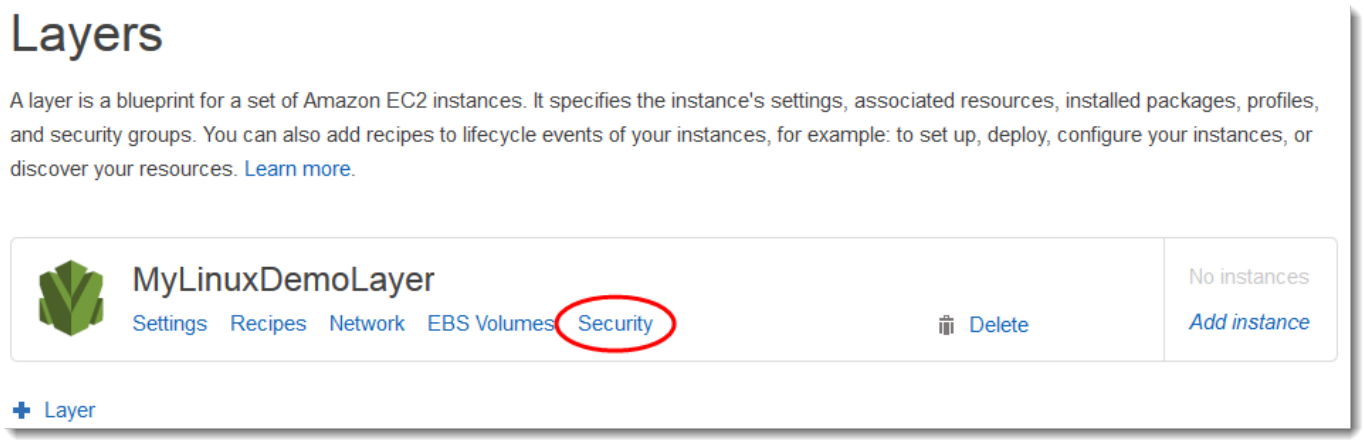
2. [계층 추가] 페이지가 표시됩니다. OpsWorks 탭에서 이름으로 **MyLinuxDemoLayer**를 입력합니다. 다른 이름을 입력할 수 있는데 이 연습 전체에서 MyLinuxDemoLayer을 해당 이름으로 바꿔야 합니다.
3. 짧은 이름으로 **demo**를 입력합니다(다른 이름을 입력할 수 있는데 이 연습 전체에서 demo를 해당 이름으로 바꿔야 함).



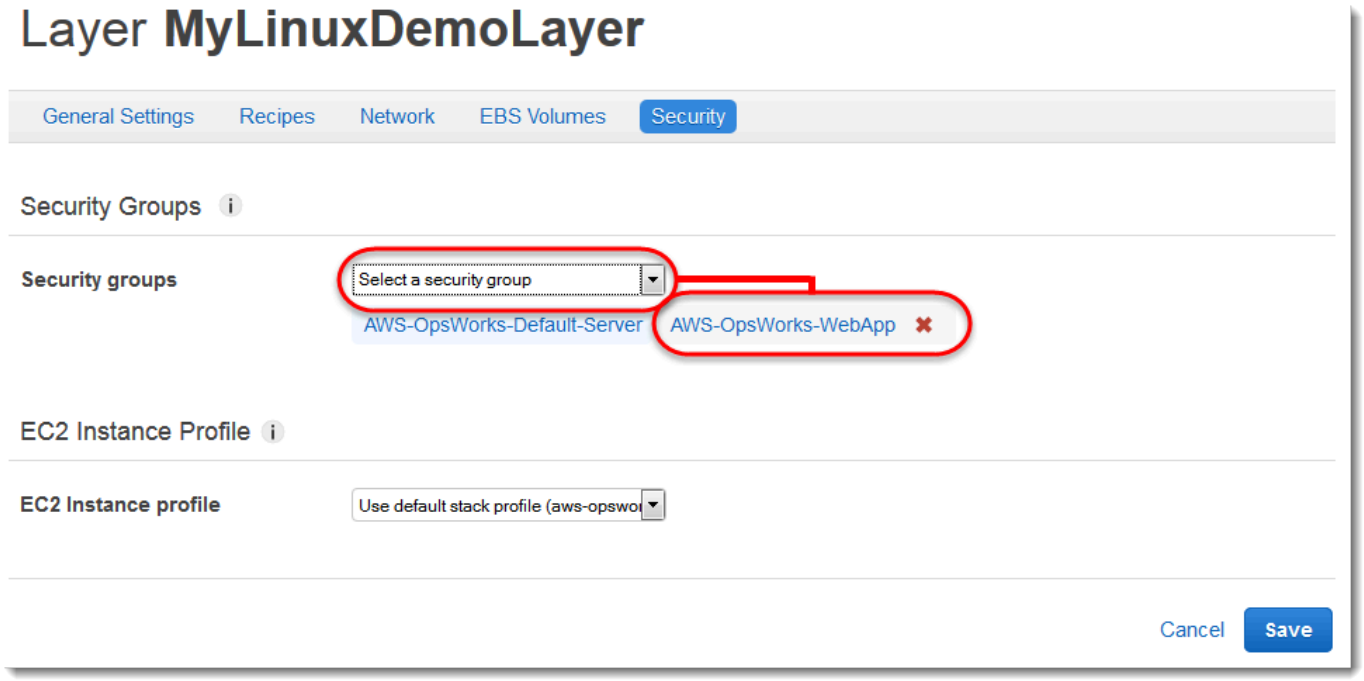
4. 계층 추가를 선택합니다. AWS OpsWorks 스택은 계층을 생성하고 계층 페이지를 표시합니다.
5. [계층] 페이지에서 [MyLinuxDemoLayer]에 대해 [네트워크]를 선택합니다.
6. [네트워크] 탭의 [자동으로 IP 주소 할당]에서 [공용 IP 주소]가 [예]로 설정되어 있는지 확인합니다. 변경한 다음 [저장]을 선택합니다.



7. [계층] 페이지에서 [보안]을 선택합니다.



8. 계층 MyLinuxDemoLayer 페이지가 보안 탭이 열린 상태로 표시됩니다. [보안 그룹]으로 [AWS-OpsWorks-WebApp]을 선택하고 [저장]을 선택합니다.



9. 계층에 `AWS-OpsWorks-WebApp` 보안 그룹이 추가됩니다. (이 보안 그룹을 통해 사용자는 이 안내의 뒷부분에서 인스턴스의 앱에 연결할 수 있습니다. 이 보안 그룹이 없으면 사용자는 웹 브라우저에서 인스턴스에 연결할 수 없다는 메시지를 받게 됩니다.)

이제 이 안내서를 위한 최신 설정을 갖춘 계층이 생겼습니다.

다음 단계에서는 인스턴스에 배포할 앱을 지정합니다.

4단계: 인스턴스에 배포할 앱 지정

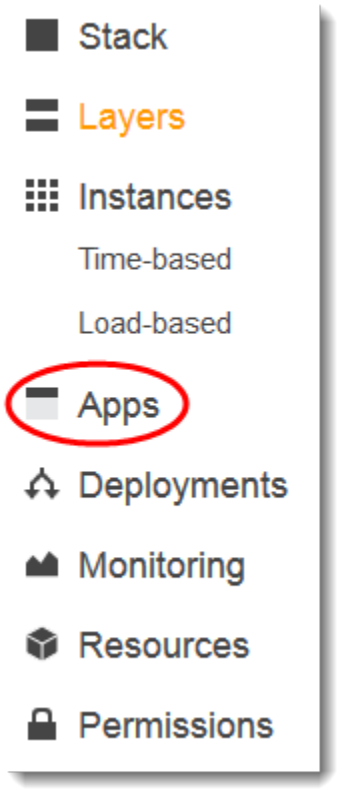
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 연습의 뒷부분에서 인스턴스에 배포할 앱에 대해 AWS OpsWorks Stacks에 알립니다. 이 컨텍스트에서 AWS OpsWorks Stacks는 앱을 인스턴스에서 실행하려는 코드로 정의합니다. (자세한 내용은 앱 섹션을 참조하세요.)

이 섹션의 절차는 Chef 12 이상의 스택에 적용됩니다. Chef 11 스택에서 계층에 앱을 추가하는 방법에 대한 자세한 정보는 2.4단계: 앱 생성 및 배포 - Chef 11 단원을 참조하세요.

배포할 앱을 지정하려면

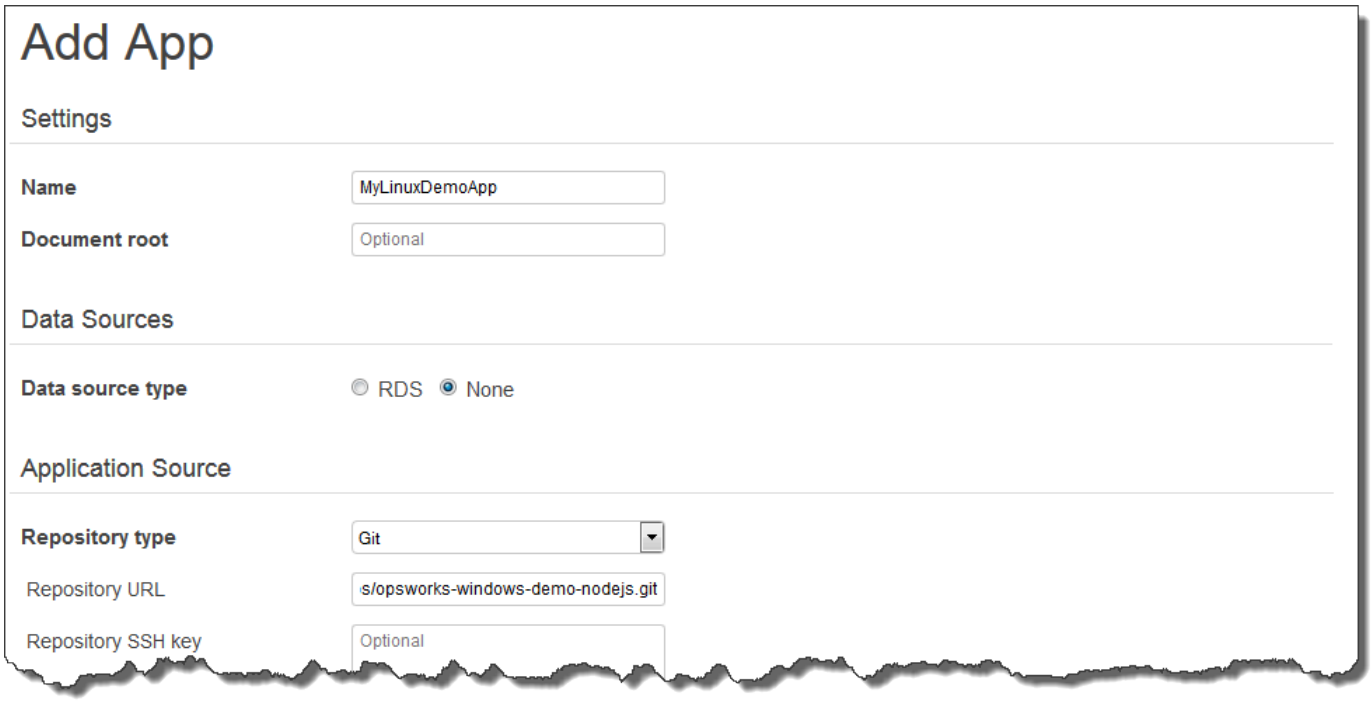
1. 서비스 탐색 창에서 [앱]을 선택합니다.

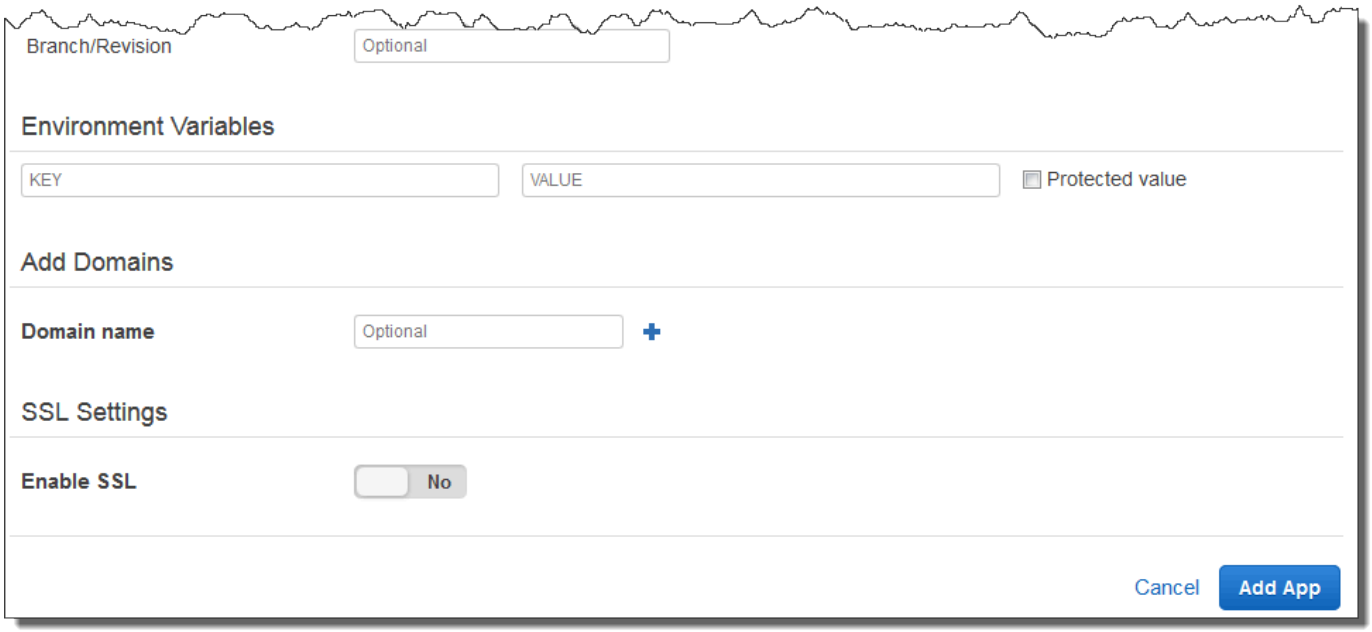


2. [앱] 페이지가 표시됩니다. [앱 추가]를 선택합니다. [앱 추가] 페이지가 표시됩니다.
3. 설정의 경우 이름으로 **MyLinuxDemoApp**을 입력합니다. 다른 이름을 입력할 수 있는데 이 연습 전체에서 MyLinuxDemoApp을 해당 이름으로 바꿔야 합니다.
4. 애플리케이션 소스의 경우 리포지토리 URL로 **https://github.com/awslabs/opsworks-windows-demo-nodejs.git**를 입력합니다.
5. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.

   - [설정], [문서 루트](비워 둠)

- [데이터 소스], [데이터 소스 유형]([없음])
- [리포지토리 유형]([Git])
- [리포지토리 SSH 키](비워 둠)
- [분기/개정](비워 둠)
- [환경 변수]([키] 비워 둠, [값] 비워 둠, [보호된 값] 선택 안 함)
- [도메인 추가], [도메인 이름](비워 둠)
- [SSL 설정], [SSL 활성화]([아니요])





6. 앱 추가를 선택합니다. AWS OpsWorks 스택은 앱을 추가하고 앱 페이지를 표시합니다.

이제 이 안내서를 위한 최신 설정을 갖춘 앱이 생겼습니다.

다음 단계에서는 인스턴스를 시작합니다.

5단계: 인스턴스 시작

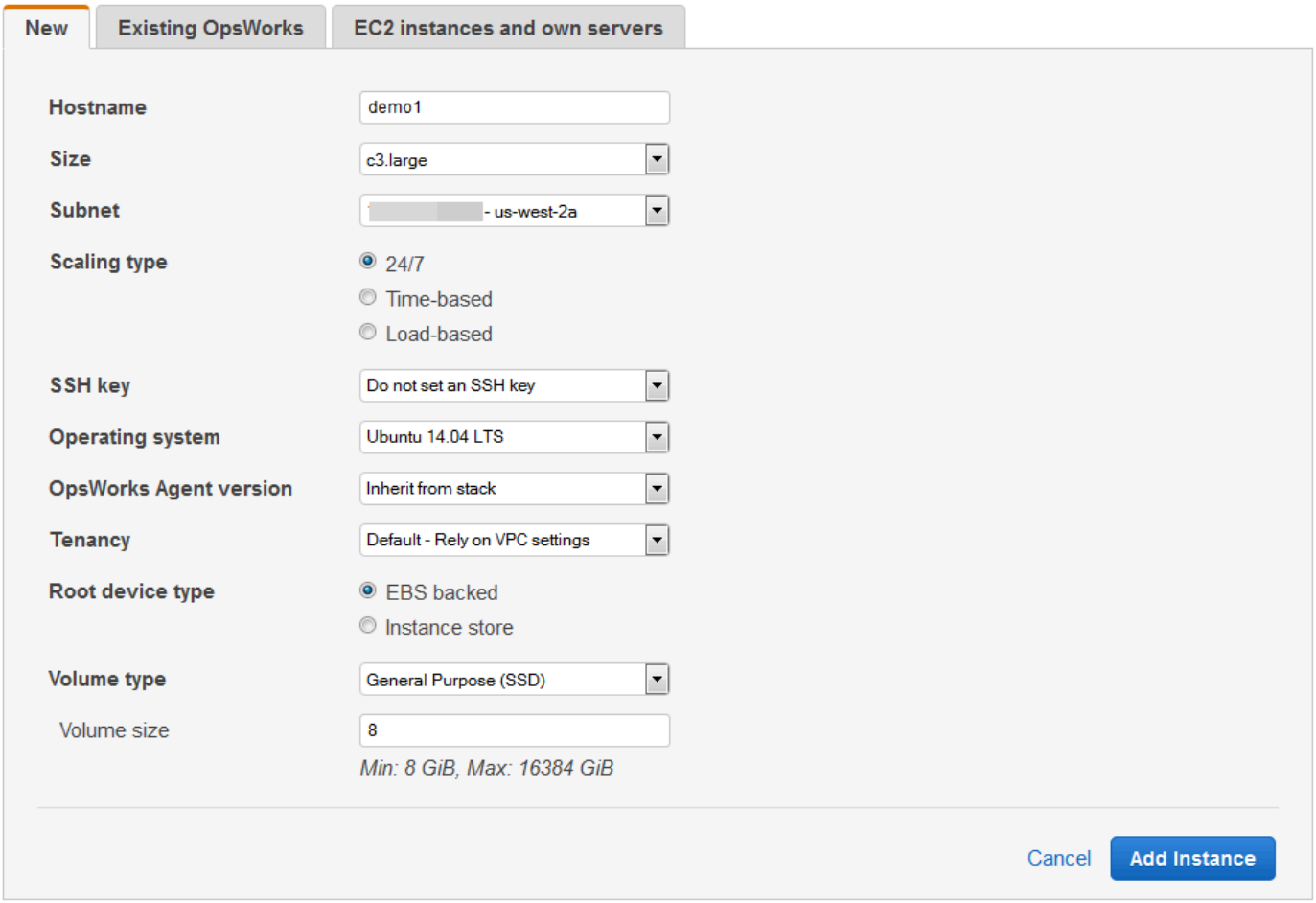
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks를 사용하여 Ubuntu Server Amazon EC2 인스턴스를 시작합니다. 이 인스턴스는 이 안내서 앞부분에서 만든 계층에서 정의한 설정을 사용합니다. (자세한 내용은 인스턴스 섹션을 참조하세요.)

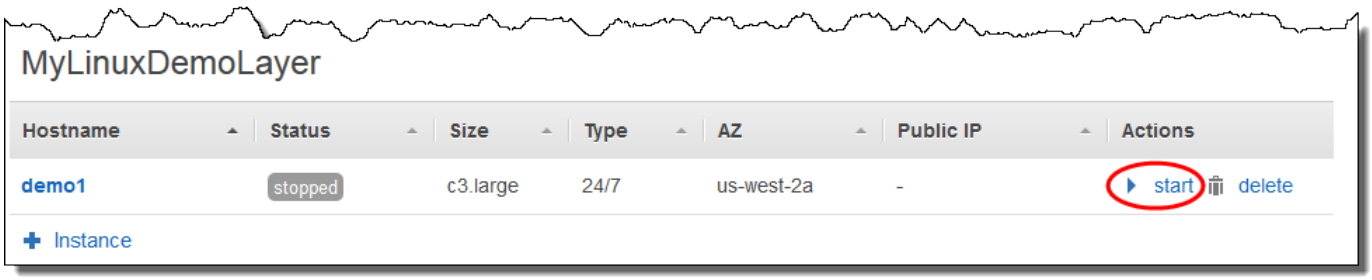
인스턴스를 시작하려면

1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.

2. [MyLinuxDemoLayer]에 대해 [인스턴스 추가]를 선택합니다.
3. [New] 탭에서 다음 옵션을 기본값으로 둡니다.
   - [호스트 이름]([demo1])
   - [크기]([c3.large])
   - [서브넷](*IP ##* us-west-2a)
4. 고급을 선택합니다.
5. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.
   - [조정 유형]([24/7])
   - [SSH 키]([기본 SSH 키 사용 금지])
   - 운영 체제(Ubuntu 18.04 LTS)
   - [OpsWorks Agent 버전]([스택에서 상속])
   - [테넌시]([기본값 - VPC 설정 사용])
   - [루트 디바이스 유형]([EBS 지원])
   - [볼륨 유형]([범용(SSD)])
   - [볼륨 크기]([8])
6. 결과는 다음 스크린샷과 비슷해야 합니다.



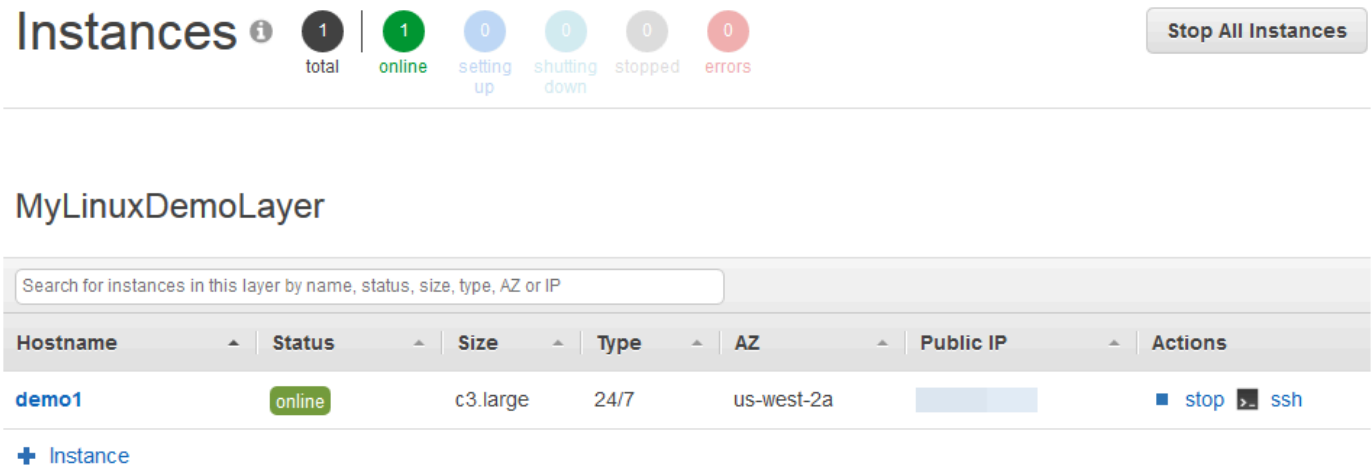
7. 인스턴스 추가를 선택합니다. AWS OpsWorks 스택은 인스턴스를 계층에 추가하고 인스턴스 페이지를 표시합니다.
8. [MyLinuxDemoLayer]의 [demo1]에 해당하는 [작업]에서 [시작]을 선택합니다.



9. 몇 분 동안 다음 작업이 수행됩니다.
   - [설정] 원이 0에서 1로 변경됩니다.

- [상태]가 [중지됨]에서 [요청됨], [보류 중], [부팅 중], [실행 중_설정]으로 차례로 바뀐 뒤 마지막으로 [온라인]으로 변경됩니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.
- [상태]가 [온라인]으로 변경되면 [설정] 원 표시기가 [1]에서 [0]으로 바뀌고, [온라인] 원이 [0]에서 [1]로 변경되고 밝은 녹색으로 변합니다. [온라인] 원이 밝은 녹색으로 바뀌고 [1]이 표시되어 인스턴스가 온라인 상태임을 나타낼 때까지 진행하지 마십시오.

10. 결과가 다음 스크린샷과 일치해야 계속 진행할 수 있습니다(실패 메시지가 표시되면 디버깅 및 문제 해결 안내서 단원 참조).



이제 앱을 배포할 인스턴스가 준비되었습니다.

> **Note**
>
> 인스턴스에 로그인하여 더 자세히 알아보려면 먼저 AWS OpsWorks Stacks에 퍼블릭 SSH 키(ssh-keygen 또는 PuTTYgen을 사용하여 생성할 수 있음)에 관한 정보를 제공한 다음 사용자가 인스턴스에 로그인할 수 있도록 `MyLinuxDemoStack` 스택에서 권한을 설정해야 합니다. 지침은 사용자의 퍼블릭 SSH 키 등록 및 SSH를 사용하여 로그인 단원을 참조하세요. SSH를 사용하여 PuTTY를 통해 인스턴스에 연결하려는 경우 AWS 설명서의 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결을 참조하세요.

다음 단계에서는 앱을 인스턴스에 배포합니다.

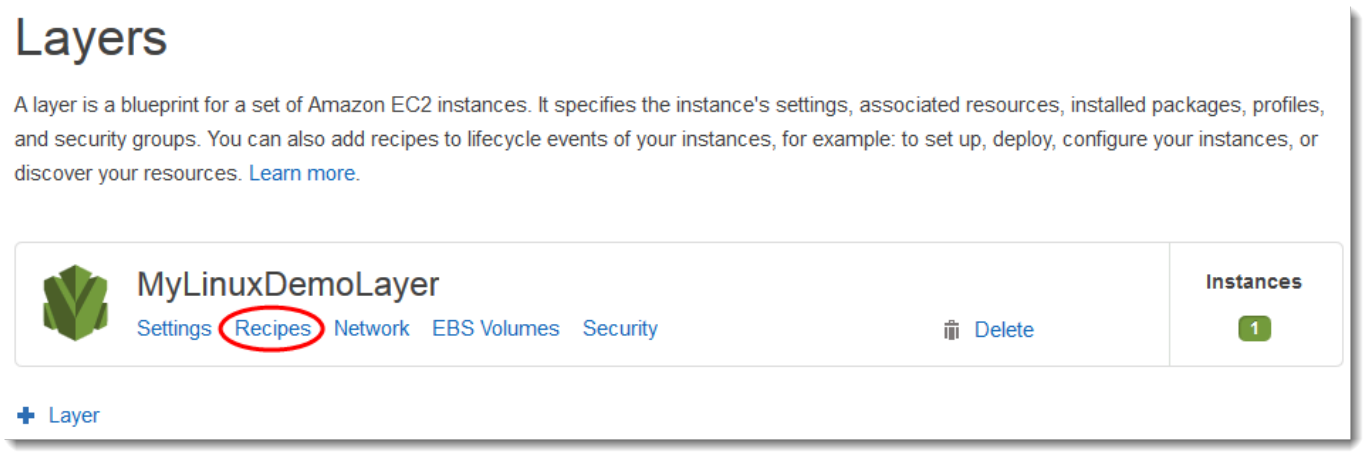
6단계: 인스턴스에 앱 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 단계에서는 실행 중인 인스턴스에 GitHub의 앱을 배포합니다. (자세한 내용은 앱 배포 섹션을 참조하세요.) 앱을 배포하기 전에 배포를 조정하는 데 사용할 레시피를 지정해야 합니다. 레시피는 Chef 개념의 하나입니다. 레시피는 Ruby 언어 구문으로 작성된 지침으로서 사용할 리소스와 이러한 리소스가 적용되는 순서를 지정합니다. (자세한 내용은 Learn Chef 웹 사이트의 레시피 정보를 참조하세요.)

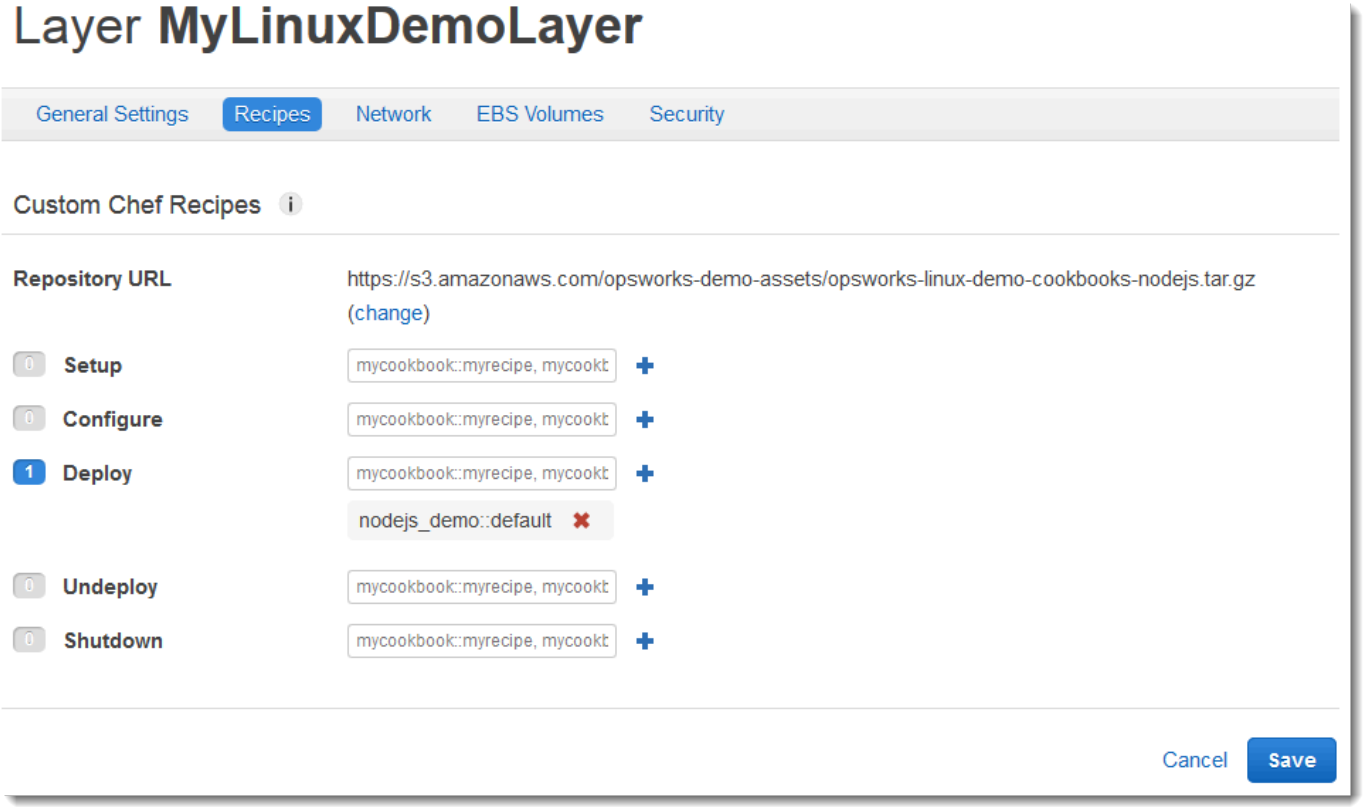
인스턴스에 앱을 배포하는 데 사용할 레시피를 지정하려면

1. 서비스 탐색 창에서 [계층]을 선택합니다. [계층] 페이지가 표시됩니다.
2. [MyLinuxDemoLayer]에 대해 [레시피]를 선택합니다.



   계층 MyLinuxDemoLayer 페이지는 레시피 탭이 열린 상태로 표시됩니다.

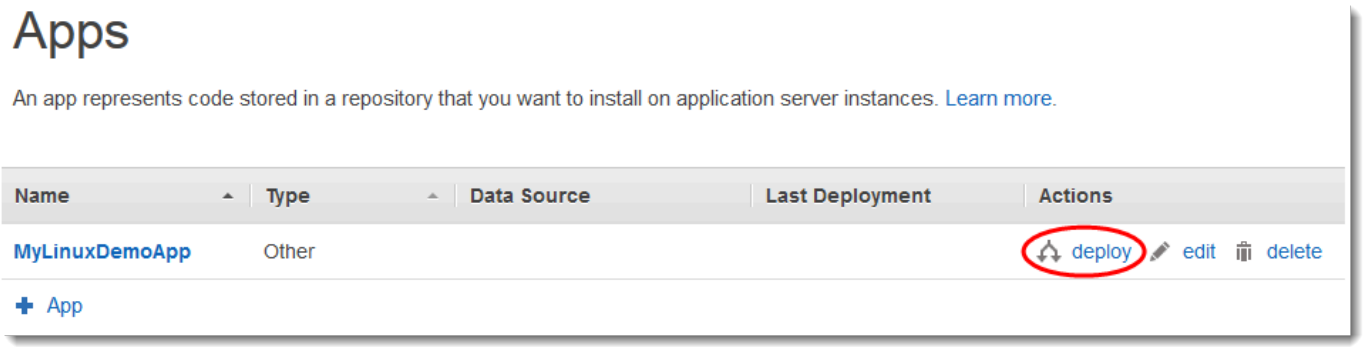
3. 사용자 지정 Chef Recipes(사용자 정의 Chef 레시피)의 경우 배포에 **`nodejs_demo::default`**를 입력한 다음 Enter를 누릅니다. `nodejs_demo`는 쿡북의 이름이고, `default`는 쿡북 내 대상 레시피의 이름입니다. 레시피의 코드를 살펴보려면 자세히 알아보기: 이 안내서에서 사용한 쿡북 살펴보기 단원을 참조하세요. 결과는 다음 스크린샷과 일치해야 합니다.



4. 저장을 선택합니다. AWS OpsWorks 스택은 계층의 배포 수명 주기 이벤트에 레시피를 추가합니다.

인스턴스에 앱을 배포하려면

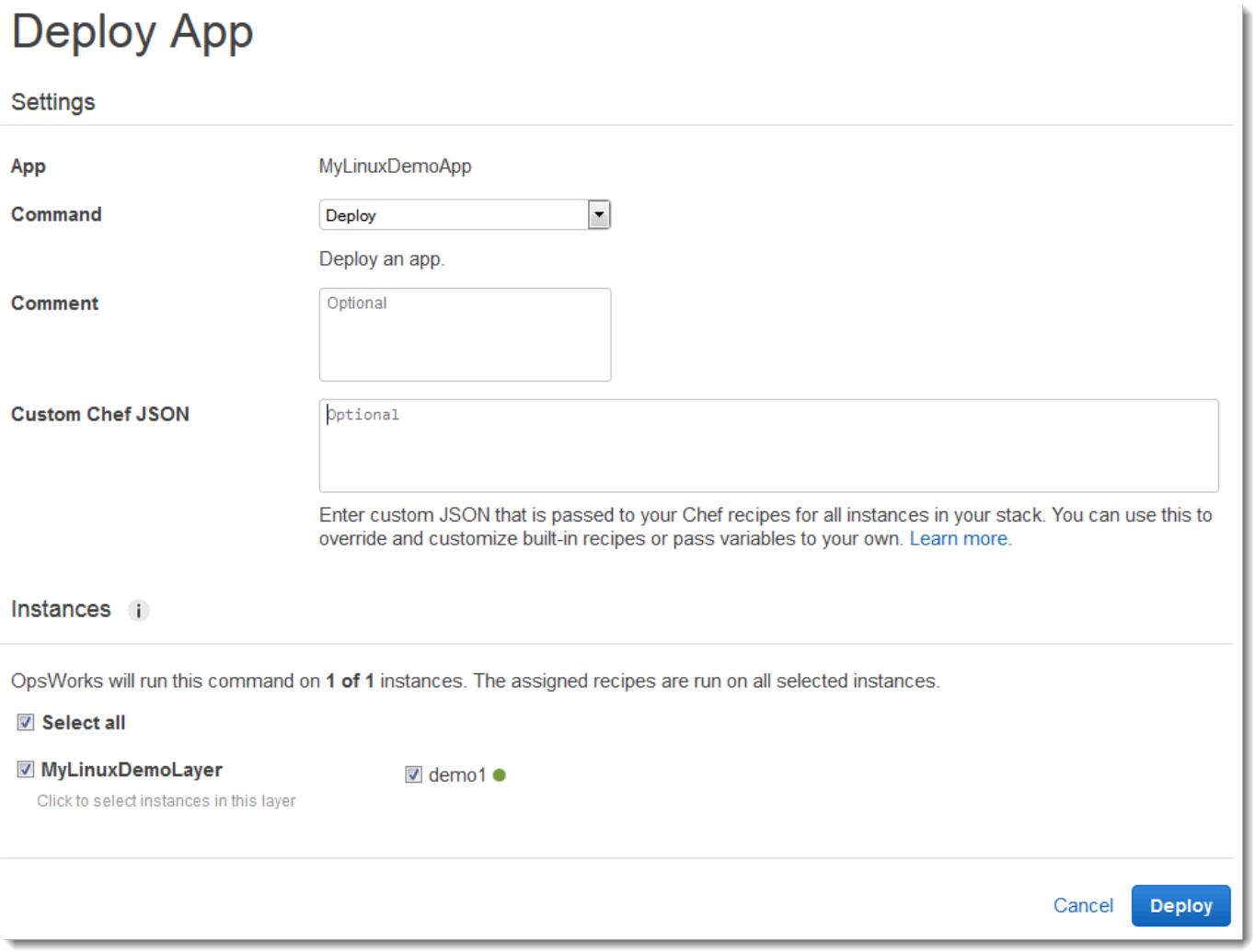
1. 서비스 탐색 창에서 [앱]을 선택합니다. [앱] 페이지가 표시됩니다.
2. 스크린샷에 표시된 것처럼 MyLinuxDemoApp의 [작업]에서 [배포]를 선택합니다.



3. [앱 배포] 페이지에서 다음 옵션을 기본값으로 둡니다.

- [명령]([배포])
- [설명](비워 둠)
- [설정], [고급], [사용자 지정 Chef JSON](비워 둠)
- [인스턴스], [고급]([모두 선택] 선택, [MyLinuxDemoLayer] 선택, [demo1] 선택)

4. 결과는 다음 스크린샷과 일치해야 합니다.



5. 배포(Deploy)를 선택합니다. [배포 MyLinuxDemoApp - 배포] 페이지가 표시됩니다. [상태]가 [실행 중]에서 [성공]으로 변경됩니다. [demo1] 옆에 회전하는 원이 표시된 다음 녹색 확인 표시로 바뀝니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다. [상태]가 [성공]으로 변경되고 녹색 확인 표시 아이콘이 표시될 때까지 진행하지 마십시오.

6. 결과는 [생성 시간], [완료 시간], [지속 시간] 및 [사용자]를 제외하고 다음 스크린샷과 일치해야 합니다. [상태]가 [실패]이면 문제 해결을 위해 [로그]에서 [표시]를 선택하여 오류에 대한 세부 정보를 확인하세요.

이제 인스턴스에 성공적으로 앱을 배포했습니다.

다음 단계에서는 인스턴스에 배포된 앱을 테스트합니다.

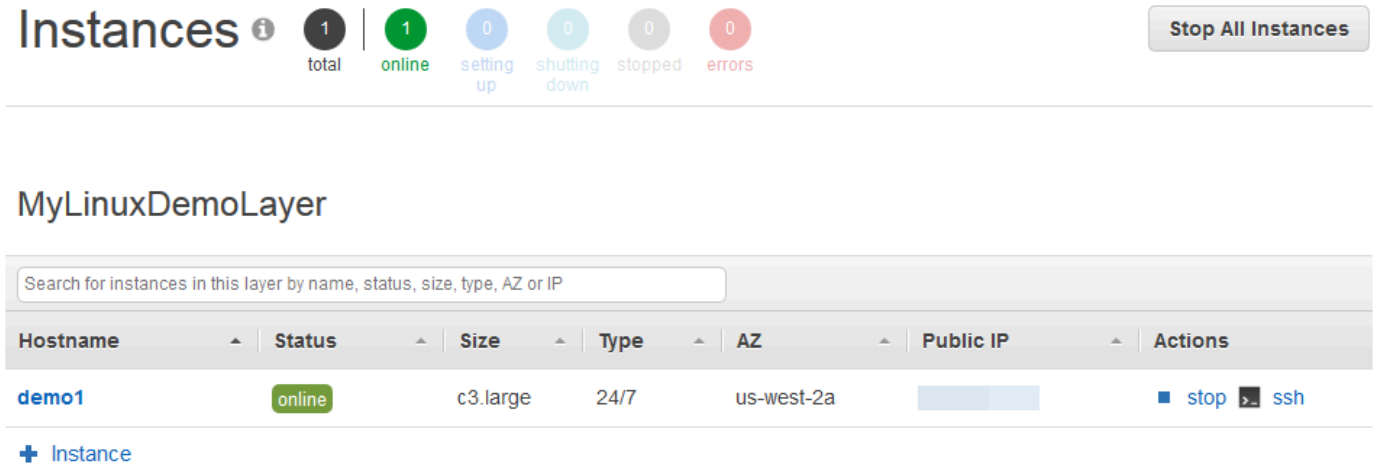
7단계: 인스턴스에 배포된 앱 테스트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제 인스턴스에서의 앱 배포를 테스트합니다.

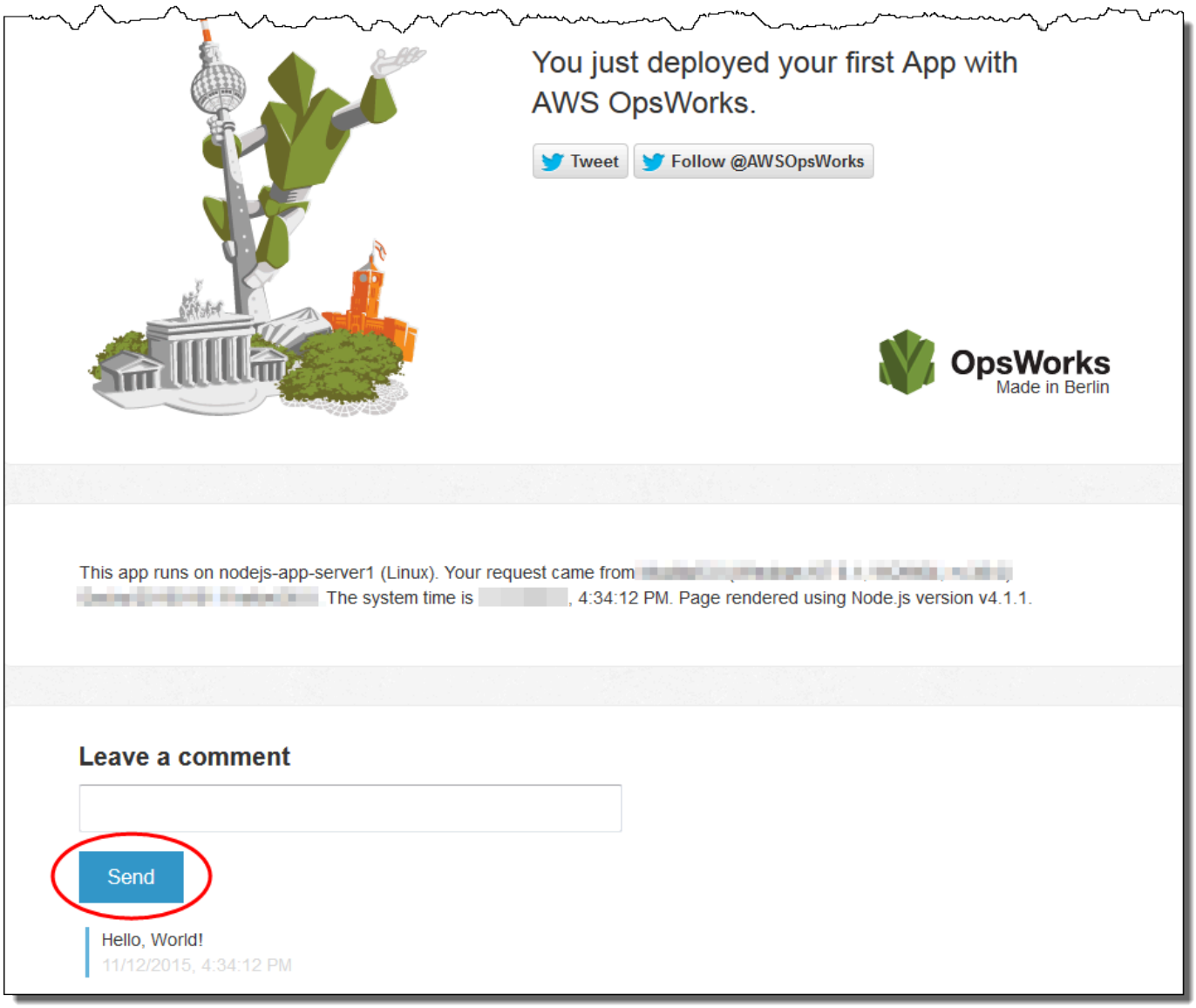
인스턴스에서 배포를 테스트하려면

1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. [MyLinuxDemoLayer]에 있는 [demo1]의 [퍼블릭 IP]에서 IP 주소를 선택합니다.



새 웹 브라우저 탭에 앱이 표시됩니다.

3. 성공을 알리는 웹 페이지의 [댓글 남기기] 텍스트 상자에 댓글을 입력한 다음 [보내기]를 선택하여 앱을 테스트합니다. 앱이 웹 페이지에 사용자의 댓글을 추가합니다. 원하는 수만큼 계속해서 댓글을 남기고 [보내기]를 선택합니다.



4. Twitter 계정이 있으면 [트윗] 또는 [팔로우 @AWSOpsWorks]를 선택하고 화면에 표시되는 지침에 따라 앱을 트윗하거나 @AWSOpsWorks를 팔로우하세요.

이제 인스턴스에 배포된 앱을 성공적으로 테스트했습니다.

다음 단계에서는이 연습에 사용한 AWS 리소스를 정리할 수 있습니다. 이 다음 단계는 선택 사항입니다. AWS OpsWorks Stacks에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용할 수 있습니다. 그러나 이러한 AWS 리소스를 주변에 보관하면 AWS 계정에 요금이 계속 부과될 수 있습니다. 나중에 사용할 수 있도록 이러한 AWS 리소스를 계속 유지하려면 이제이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

8단계(선택 사항): 정리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 계정에 추가 요금이 발생하지 않도록이 연습에 AWS 사용된 리소스를 삭제할 수 있습니다. 이러한 AWS 리소스에는 AWS OpsWorks Stacks 스택과 스택의 구성 요소가 포함됩니다. (자세한 내용은 AWS OpsWorks 요금을 참조하세요.) 그러나 AWS OpsWorks Stacks에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용하고 싶을 수 있습니다. 이러한 AWS 리소스를 계속 사용하려면 이제 이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

이번 연습에서 생성한 리소스에 보관된 콘텐츠는 개인 식별 정보를 포함할 수 있습니다. 이 정보를 AWS에 저장하고 싶지 않다면, 이 주제에서 설명하는 다음 단계를 따르십시오.

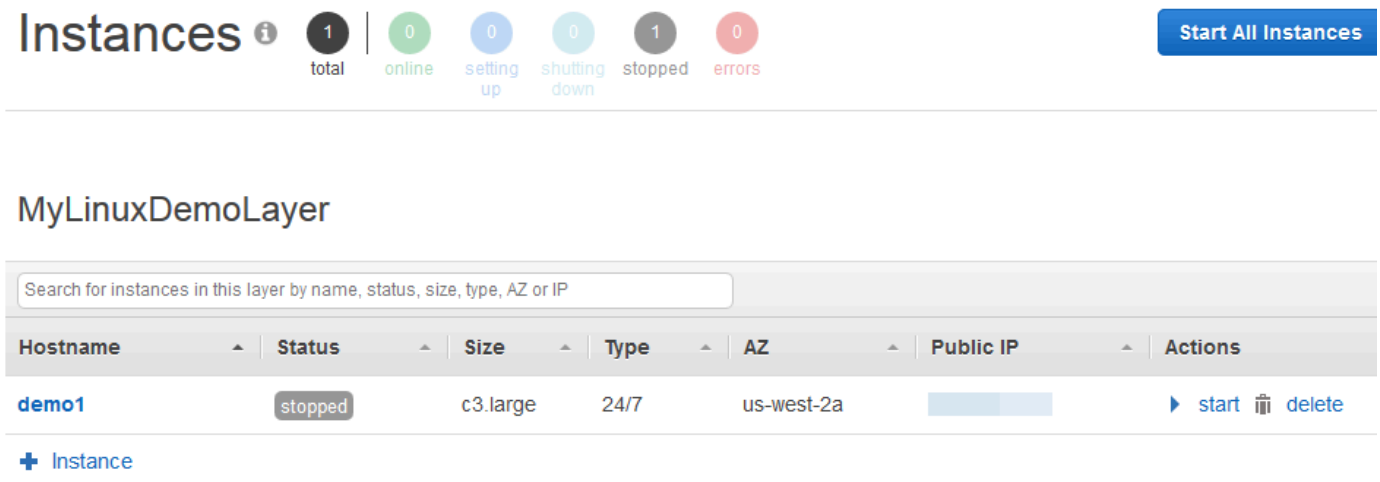
스택에서 앱을 삭제하려면

1. AWS OpsWorks Stacks 콘솔의 서비스 탐색 창에서 앱을 선택합니다. [앱] 페이지가 표시됩니다.
2. [MyLinuxDemoApp]의 [작업]에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 앱을 삭제합니다.

스택의 인스턴스를 삭제하려면

1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. [MyLinuxDemoLayer]의 [demo1]에 해당하는 [작업]에서 [중지]를 선택합니다. 확인 메시지가 표시되면 [중지]를 선택합니다. 다음과 같이 진행됩니다.
   - [상태]가 [온라인]에서 [중지 중]으로 바뀌었다가 마지막으로 [중지됨]으로 바뀝니다.
   - [온라인]이 [1]에서 [0]으로 바뀝니다.
   - [종료 중]이 [0]에서 [1]로 바뀌었다가 마지막에는 다시 [0]으로 바뀝니다.
   - [중지됨]가 마지막으로 [0]에서 [1]로 변경됩니다.

이 프로세스는 몇 분 정도 걸릴 수 있습니다. AWS OpsWorks Stacks가 완료되면 다음 결과가 표시됩니다.



3. 작업에 대해 삭제를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 인스턴스를 삭제하고 인스턴스 없음 메시지를 표시합니다.

스택을 삭제하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [MyLinuxDemoStack] 페이지가 표시됩니다.
2. [스택 삭제]를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 스택을 삭제하고 OpsWorks 대시보드 페이지를 표시합니다.

선택적으로 다른 AWS 서비스 및 EC2 인스턴스에 액세스하는 데 재사용하지 않으려는 경우이 연습에 사용한 사용자 및 Amazon EC2 키 페어를 삭제할 수 있습니다. 지침은 IAM 사용자 및 Amazon EC2 키 페어 및 Linux 인스턴스 삭제를 참조하세요.

이제 이 안내서를 성공적으로 완료했습니다. 자세한 내용은 다음 단계 단원을 참조하십시오.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

> 레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제이 연습을 완료했으므로 AWS OpsWorks Stacks 사용에 대해 자세히 알아볼 수 있습니다.

- 이 안내서에서 사용한 쿡북과 앱을 살펴보십시오. 자세히 알아보기: 이 안내서에서 사용한 쿡북 살펴보기 및 자세히 알아보기: 이 안내서에서 사용한 앱 살펴보기 섹션을 참조하세요.
- Windows 인스턴스에서 AWS OpsWorks Stacks 사용을 연습합니다. 시작하기: Windows을(를) 참조하세요.
- 새 스택 생성 학습을 통해 스택에 대해 자세히 알아봅니다.
- OpsWorks 계층의 구성 편집를 통해 계층에 대해 자세히 알아봅니다.
- 계층에 인스턴스 추가를 통해 인스턴스에 대해 자세히 알아봅니다.
- 앱 배포를 통해 앱에 대해 자세히 알아봅니다.
- 쿡북과 레시피 단원에 대해 자세히 알아보세요.
- 자체 쿡북을 생성합니다. 시작하기 - 쿡북을(를) 참조하세요.
- 보안 및 권한을 사용하여 스택에 대한 액세스를 제어하는 방법을 알아봅니다.

자세히 알아보기: 이 안내서에서 사용한 쿡북 살펴보기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는 Stacks가 연습에 사용한 AWS OpsWorks 쿡북에 대해 설명합니다.

쿡북은 Chef 개념의 하나입니다. 쿡북은 레시피, 속성 값, 파일, 템플릿, 라이브러리, 정의, 사용자 지정 리소스 같은 구성 정보가 포함되어 있는 아카이브 파일입니다. 레시피 역시 Chef 개념입니다. 레시피는 Ruby 언어 구문으로 작성된 지침으로서 사용할 리소스와 이러한 리소스가 적용되는 순서를 지정합니다. 자세한 내용은 Learn Chef 웹 사이트의 쿡북 정보 및 레시피 정보를 참조하세요.

이 안내서에서 사용한 쿡북의 콘텐츠를 보려면 opsworks-linux-demo-cookbooks-nodejs.tar.gz 파일의 콘텐츠를 로컬 워크스테이션의 빈 디렉터리에 압축 해제하세요. 쿡북을 배포한 인스턴스에 로그인해 /var/chef/cookbooks 디렉터리의 내용을 살펴볼 수도 있습니다.

cookbooks/nodejs_demo/recipes 디렉터리의 default.rb 파일은 쿡북이 코드를 실행하는 곳입니다.

```
app = search(:aws_opsworks_app).first
app_path = "/srv/#{app['shortname']}"

package "git" do
  options "--force-yes" if node["platform"] == "ubuntu" && node["platform_version"] ==
 "18.04"
end

application app_path do
  javascript "4"
  environment.update("PORT" => "80")

  git app_path do
    repository app["app_source"]["url"]
    revision app["app_source"]["revision"]
  end

  link "#{app_path}/server.js" do
    to "#{app_path}/index.js"
  end

  npm_install
  npm_start
end
```

이 파일이 하는 일은 다음과 같습니다.

- search(:aws_opsworks_app).first는 Chef 검색을 사용하여 최종적으로 인스턴스에 배포될 앱에 대한 정보를 찾습니다. 이 정보에는 앱의 짧은 이름과 소스 리포지토리 세부 정보 같은 설정이 포함됩니다. 이 안내서에서는 앱 하나만 배포했으므로 Chef 검색은 인스턴스에서 aws_opsworks_app 검색 인덱스 내 정보의 첫 번째 항목에서 이러한 설정을 가져옵니다. 인스턴스가 시작될 때마다 AWS OpsWorks Stacks는이 정보와 기타 관련 정보를 인스턴스 자체에 데이터 백 세트로 저장하고 Chef 검색을 통해 데이터 백 콘텐츠를 가져옵니다. 이 설정들을 이 레시피에 하드 코딩할 수 있지만 데이터 백과 Chef 검색을 사용하는 것이 더 확실한 방법입니다. 데이터 백에 대한

자세한 정보는 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하세요. Learn Chef 웹 사이트의 데이터 백에 대한 정보도 참조하세요. Chef 검색에 대한 자세한 내용은 Learn Chef 웹 사이트의 검색 정보를 참조하세요.

- `package` 리소스는 인스턴스에 Git를 설치합니다.
- `application` 리소스는 웹 애플리케이션을 설명하고 배포합니다.
  - `javascript`는 설치할 JavaScript 런타임의 버전입니다.
  - `environment`는 환경 변수를 설정합니다.
  - `git`는 지정된 리포지토리와 브랜치에서 소스 코드를 가져옵니다.
  - `app_path`는 리포지토리를 복제할 경로입니다. 인스턴스에 경로가 없는 경우 AWS OpsWorks Stacks가 경로를 생성합니다.
  - `link`는 심볼 링크를 생성합니다.
  - `npm_install`는 Node.js의 기본 패키지 관리자인 Node Package Manager를 설치합니다.
  - `npm_start`는 Node.js를 실행합니다.

Stacks가이 연습에 사용되는 AWS OpsWorks 쿡북을 생성했지만 자체 쿡북을 생성할 수 있습니다. 자세한 방법은 시작하기 - 쿡북을 참조하세요. 또한 Learn Chef 웹 사이트의 쿡북 정보, 레시피 정보 및 Chef Basics on Ubuntu 배우기 페이지로 이동하고, Chef 시작하기 웹 사이트의 Chef를 이용한 첫 단계에서 “우리의 첫 Chef 쿡북” 섹션을 참조하세요.

자세히 알아보기: 이 안내서에서 사용한 앱 살펴보기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는이 연습을 위해 AWS OpsWorks Stacks가 인스턴스에 배포하는 앱을 설명합니다.

앱의 소스 코드를 보려면 opsworks-windows-demo-nodejs GitHub 리포지토리의 콘텐츠를 로컬 워크스테이션의 빈 디렉터리에 압축 해제하세요. 쿡북을 배포한 인스턴스에 로그인해 `/srv/mylinuxdemoapp` 디렉터리의 내용을 살펴볼 수도 있습니다.

`index.js` 파일은 앱에 가장 중요한 코드를 포함하고 있습니다.

```
var express = require('express');
var app = express();
var path = require('path');
var os = require('os');
var bodyParser = require('body-parser');
var fs = require('fs');

var add_comment = function(comment) {
  var comments = get_comments();
  comments.push({"date": new Date(), "text": comment});
  fs.writeFileSync('./comments.json', JSON.stringify(comments));
};

var get_comments = function() {
  var comments;
  if (fs.existsSync('./comments.json')) {
    comments = fs.readFileSync('./comments.json');
    comments = JSON.parse(comments);
  } else {
    comments = [];
  }
  return comments;
};

app.use(function log (req, res, next) {
  console.log([req.method, req.url].join(' '));
  next();
});
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: false }))

app.set('view engine', 'jade');
app.get('/', function(req, res) {
  var comments = get_comments();
  res.render("index",
    { agent: req.headers['user-agent'],
      hostname: os.hostname(),
      nodeversion: process.version,
      time: new Date(),
      admin: (process.env.APP_ADMIN_EMAIL || "admin@unconfigured-value.com" ),
      comments: get_comments()
    });
});
```

```
app.post('/', function(req, res) {
  var comment = req.body.comment;
  if (comment) {
    add_comment(comment);
    console.log("Got comment: " + comment);
  }
  res.redirect("/#form-section");
});

var server = app.listen(process.env.PORT || 3000, function() {
  console.log('Listening on %s', process.env.PORT);
});
```

이 파일이 하는 일은 다음과 같습니다.

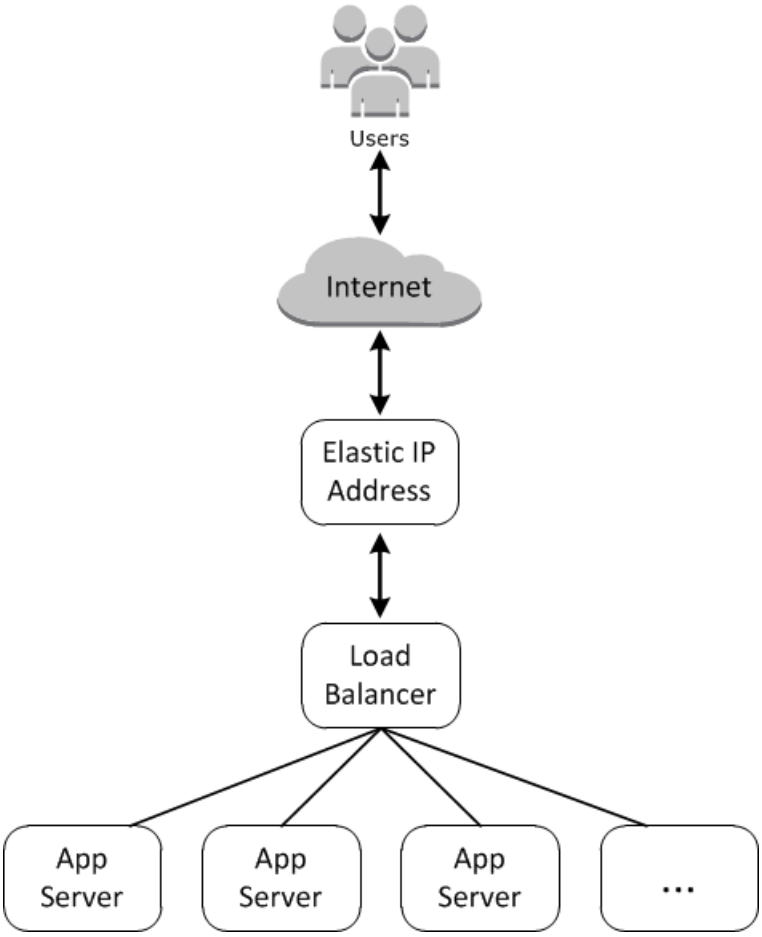
- `require`는 이 웹 앱이 예상대로 실행되는 데 필요한 일부 종속 코드가 포함된 모듈을 로드합니다.
- `add_comment` 및 `get_comments` 함수는 `comments.json` 파일에 정보를 작성하고 이 파일에서 정보를 읽어 옵니다.
- `app.get`, `app.listen`, `app.post`, `app.set`, `app.use`에 대해서는 Express API Reference를 참조하세요.

배포를 위해 앱을 만들고 패키징하는 방법은 애플리케이션 소스 단원을 참조하세요.

Windows 스택 시작하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

클라우드 기반 애플리케이션에는 일반적으로 애플리케이션 서버, 데이터베이스 서버 등과 같은 관련 리소스 그룹이 필요하며 이러한 리소스를 공동으로 생성하고 관리해야 합니다. 이러한 인스턴스의 모음을 스택이라고 합니다. 다음은 간단한 애플리케이션 스택의 예입니다.



기본 아키텍처는 다음과 같이 구성됩니다.

- 사용자 요청을 수신하기 위한 탄력적 IP 주소.
- 요청을 수신하여 애플리케이션 서버에 고르게 분산하는 로드 밸런서.
- 트래픽을 처리하는 데 필요한 만큼의 애플리케이션 서버 인스턴스 집합

또한 일반적으로 애플리케이션을 애플리케이션 서버에 분산하고 사용자 권한을 관리하는 등의 기능이 필요합니다.

AWS OpsWorks Stacks는 스택과 관련 애플리케이션 및 리소스를 생성하고 관리하는 간단하고 간단한 방법을 제공합니다. 이 장에서는 다이어그램에서 애플리케이션 서버 스택을 생성하는 프로세스를 단계별로 안내하여 AWS OpsWorks Stacks의 기본 사항과 더욱 정교한 기능을 소개합니다. AWS OpsWorks Stacks가 따르기 쉽게 만드는 증분 개발 모델을 사용합니다. 기본 스택을 설정하고 올바르게 작동한 후 전체 기능을 구현할 때까지 구성 요소를 추가합니다.

- 1단계: 사전 조건 완료에서는 이 안내서를 시작하기 위해 필요한 사전 준비를 설명합니다.

- 2단계: 기본 애플리케이션 서버 스택 생성에서는 기본 스택을 생성하여 Internet Information Services(IIS)를 지원하고 앱을 서버에 배포하는 방법을 설명합니다.
- 3단계: IISExample 확장에서는 증가하는 부하를 처리하기 위한 더 많은 애플리케이션 서버, 수신 트래픽을 분산할 로드 밸런서, 요청을 수신할 탄력적 IP 주소를 추가하여 스택을 확장하는 방법을 설명합니다.

주제

- 1단계: 사전 조건 완료
- 2단계: 기본 애플리케이션 서버 스택 생성
- 3단계: IISExample 확장
- 다음 단계

1단계: 사전 조건 완료

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

안내서를 시작하기 전에 다음 설정 단계를 완료합니다. 이러한 설정 단계에는 AWS 계정 가입, 관리 사용자 생성, AWS OpsWorks Stacks에 액세스 권한 할당이 포함됩니다.

이미 시작하기: 샘플 또는 시작하기: Linux 안내서를 마쳤다면 이 안내서의 사전 조건을 충족한 것이며, 2단계: 기본 애플리케이션 서버 스택 생성으로 단원으로 건너뛸 수 있습니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 서비스 액세스 권한 할당
- AWS OpsWorks Stacks 사용자가 도메인에 추가되었는지 확인

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

서비스 액세스 권한 할당

역할 또는 사용자에게 `AWSOpsWorks_FullAccess` 및 `AmazonS3FullAccess` 권한을 추가하여 AWS OpsWorks Stacks 서비스(및 AWS OpsWorks Stacks가 의존하는 관련 서비스)에 대한 액세스를 활성화합니다.

권한 추가에 대한 자세한 내용은 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

AWS OpsWorks Stacks 사용자가 도메인에 추가되었는지 확인

Chef 12.2 스택에 포함된 `aws_opsworks_users` 쿡북은 SSH가 있고 Windows 기반 인스턴스에 RDP(Remote Desktop Protocol) 액세스 권한이 있는 사용자를 생성합니다. 스택의 Windows 인스턴스를 Active Directory 도메인에 조인하면 Stacks 사용자가 Active Directory에 없는 경우이 AWS OpsWorks 쿡북 실행이 실패할 수 있습니다. Active Directory에서 사용자를 인식하지 못하는 경우, 도

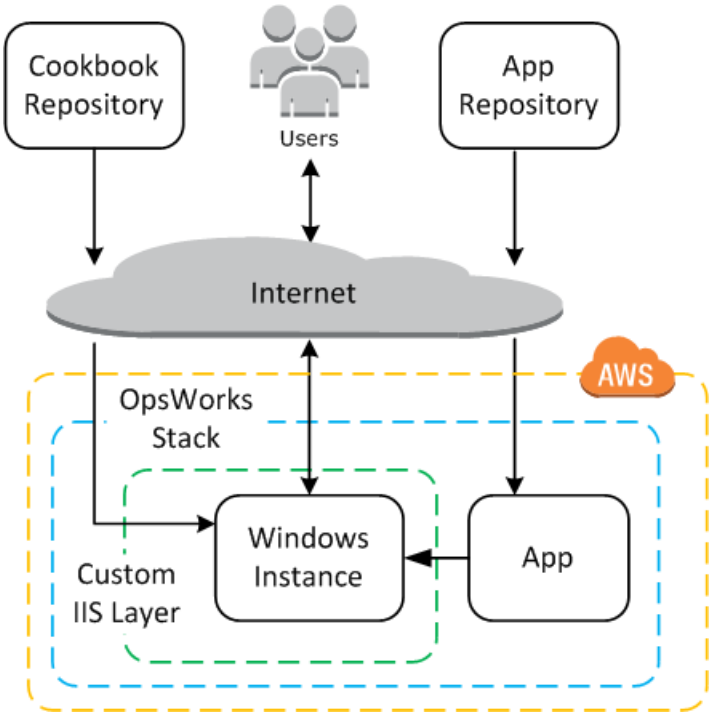
메인 가입 후 인스턴스를 재시작하면 인스턴스가 `setup failed` 상태를 입력할 수 있습니다. 도메인에 가입한 Windows 인스턴스의 경우 AWS OpsWorks 스택 사용자의 권한 페이지에서 SSH/RDP 액세스 권한을 부여하는 것으로는 충분하지 않습니다.

Chef 12.2 스택의 Windows 인스턴스를 Active Directory 도메인에 조인하기 전에 Windows 기반 스택의 모든 AWS OpsWorks Stacks 사용자가 도메인의 멤버인지 확인합니다. 이렇게 하는 가장 좋은 방법은 Windows 기반 스택을 생성하기 전에 IAM으로 페더레이션 ID를 구성한 다음 스택의 인스턴스를 도메인에 조인하기 전에 페더레이션 사용자를 AWS OpsWorks Stacks로 가져오는 것입니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 AWS 보안 블로그의 Windows Active Directory, ADFS 및 SAML 2.0을 사용하여 AWS에 대한 페더레이션 활성화 및 IAM 사용 설명서의 기존 사용자 페더레이션을 참조하세요.

2단계: 기본 애플리케이션 서버 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

기본 애플리케이션 서버 스택은 사용자 요청을 수신하기 위한 퍼블릭 IP 주소를 가진 단일 애플리케이션 서버 인스턴스로 구성됩니다. 애플리케이션 코드 및 관련 파일은 별도의 리포지토리에 저장되다가 서버로 배포됩니다. 다음 다이어그램은 이러한 스택을 보여줍니다.



스택은 다음 구성 요소를 갖습니다.

- 계층 - 인스턴스의 그룹을 나타내며 인스턴스 그룹을 어떻게 구성할지를 지정합니다.

  이 예제에서 계층은 IIS 인스턴스의 그룹을 나타냅니다.
- Amazon EC2 인스턴스를 나타내는 인스턴스입니다.

  이 경우, 계층이 IIS를 실행할 단일 인스턴스를 구성하지만 계층의 인스턴스 수에는 제한이 없습니다.
- 앱 - 인스턴스에 애플리케이션을 설치하는 데 필요한 정보를 포함합니다.
- 쿡북 - 사용자 지정 IIS 계층을 지원하는 사용자 지정 Chef 레시피를 포함합니다. 쿡북과 앱 코드는 Amazon S3 버킷 또는 Git 리포지토리의 아카이브 파일 같은 원격 리포지토리에 저장됩니다.

다음 섹션에서는 AWS OpsWorks Stacks 콘솔을 사용하여 스택을 생성하고 애플리케이션을 배포하는 방법을 설명합니다.

주제

- 2.1단계: 스택 생성
- 2.2단계: RDP 액세스 승인
- 2.3단계: 사용자 지정 쿡북 구현

- 2.4단계: IIS 계층 추가
- 2.5단계: 앱 배포

2.1단계: 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스 및 기타 리소스의 컨테이너 역할을 하는 스택을 생성하여 AWS OpsWorks Stacks 프로젝트를 시작합니다. 스택 구성은 AWS 리전, 기본 운영 체제 등 스택의 모든 인스턴스가 공유하는 일부 기본 설정을 지정합니다.

새 스택을 생성하려면

1. stack 추가

   아직 수행하지 않은 경우 AWS OpsWorks Stacks 콘솔에 로그인합니다.

   - 계정에 기존 스택이 없는 경우에는 [AWS OpsWorks 시작하기] 페이지가 표시되는데 [첫 번째 스택 추가]를 선택합니다.
   - 그렇지 않으면 계정의 AWS OpsWorks 스택을 나열하는 Stacks 대시보드가 표시됩니다. 스택 추가를 선택합니다.

2. 스택 구성

   [스택 추가] 페이지에서 [Chef 12 스택]을 선택하고 다음 설정을 지정합니다.

   스택 이름

   스택의 이름을 입력합니다. 이 이름에는 영숫자(a-z, A-Z 및 0-9) 및 하이픈(-)을 사용할 수 있습니다. 이 연습에서 사용되는 예제 스택의 이름은 **`IISWalkthrough`**입니다.

   리전

   스택 리전으로 미국 서부(오레곤)를 선택합니다.

스택은 모든 리전에서 생성할 수 있지만 자습서의 경우 미국 서부(오레곤)를 선택하는 것이 좋습니다.

기본 운영 체제

Windows를 선택한 다음 기본 설정인 Microsoft Windows Server 2022 Base를 지정합니다.

사용자 지정 Chef 쿡북 사용

이번 연습에서 이 옵션은 [아니요]로 지정합니다.

3. [고급]을 선택하여 IAM 역할 및 기본 IAM 인스턴스 프로파일이 선택되어 있는지 확인합니다.

   IAM 역할

   스택의 IAM(AWS Identity and Access Management) 역할을 지정합니다. AWS OpsWorks 스택은 Amazon EC2 인스턴스 생성 및 관리와 같은 작업을 수행하기 위해 다른 AWS 서비스에 액세스해야 합니다. IAM 역할은 AWS OpsWorks Stacks가 사용자를 대신하여 다른 AWS 서비스와 함께 작업하기 위해 수임하는 역할을 지정합니다. 자세한 내용은 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 단원을 참조하십시오.

   - 계정에 기존 AWS OpsWorks Stacks IAM 역할이 있는 경우 목록에서 해당 역할을 선택할 수 있습니다.

     AWS OpsWorks Stacks에서 역할을 생성한 경우 이름이 로 지정됩니다`aws-opsworks-service-role`.

   - 그렇지 않으면 새 IAM 역할을 선택하여 AWS OpsWorks Stacks에 올바른 권한을 가진 새 역할을 생성하도록 지시합니다.

     참고: AWS OpsWorks Stacks 전체 액세스 권한이 있는 경우 새 역할을 생성하려면 여러 가지 추가 IAM 권한이 필요합니다. 자세한 내용은 정책 예제 단원을 참조하십시오.

4. 다른 설정에 대해서는 기본값을 수락하고 [스택 추가]를 선택합니다. 다양한 스택 설정에 대한 자세한 정보는 새 스택 생성 단원을 참조하세요.

2.2단계: RDP 액세스 승인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제 스택을 생성했으므로 계층을 생성하고 계층에 Windows 인스턴스를 추가합니다. 하지만 그 전에 먼저 RDP를 사용해 사용자 지정 계층의 인스턴스에 연결할 수 있도록 스택을 구성해야 합니다. 그러려면 다음을 수행해야 합니다.

- RDP 액세스를 제어하는 인바운드 규칙을 보안 그룹에 추가합니다.
- RDP 액세스를 허용하도록이 스택에 대한 AWS OpsWorks Stacks 권한을 설정합니다.

리전에서 첫 번째 스택을 생성하면 AWS OpsWorks Stacks가 보안 그룹 세트를 생성합니다. 여기에는 과 같은 이름의 인스턴스가 포함되며`AWS-OpsWorks-RDP-Server`, AWS OpsWorks Stacks는 모든 Windows 인스턴스에 연결하여 RDP 액세스를 허용합니다. 하지만 이 보안 그룹은 기본적으로 아무 규칙도 포함하고 있지 않으므로, 인스턴스에 RDP 액세스를 허용하는 인바운드 규칙을 추가해야 합니다.

RDP 액세스를 허용하려면

1. Amazon EC2 콘솔을 열고, 콘솔을 스택의 리전으로 설정한 다음, 탐색 창에서 보안 그룹을 선택합니다.
2. [AWS-OpsWorks-RDP-Server], [인바운드] 탭, [편집]을 차례대로 선택합니다.
3. [역할 추가]를 선택하고 다음 설정을 지정합니다.

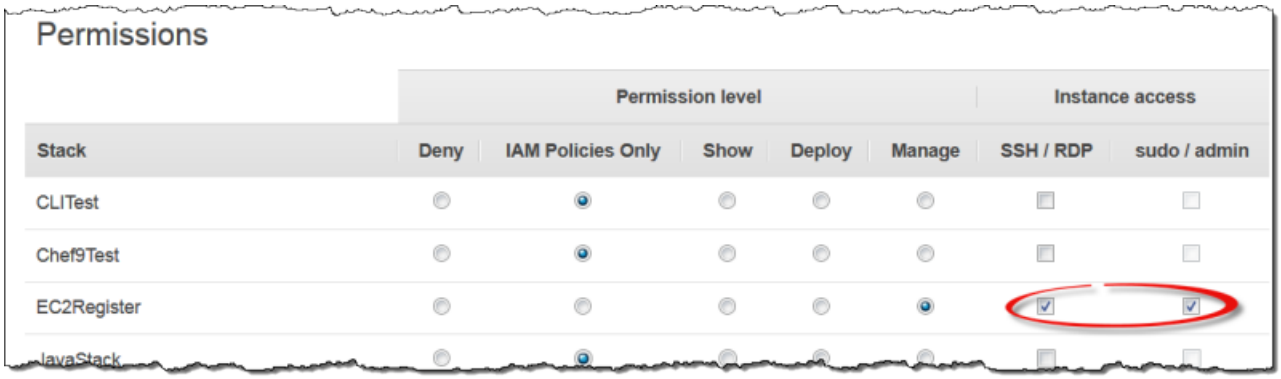
   - 유형 – RDP.
   - 소스 – 허용 가능한 소스 IP 주소.

     일반적으로 IP 주소 또는 지정된 IP 주소 범위(대개 회사 IP 주소)에서 들어오는 인바운드 RDP 요청을 허용합니다. 학습 목적으로는 모든 IP 주소의 RDP 액세스를 허용하는 0.0.0.0/0을 지정하는 것으로 충분합니다.

보안 그룹은 인스턴스가 RDP 연결 요청을 수신하도록 허용하지만 이것은 절반에 불과합니다. 일반 사용자는 AWS OpsWorks Stacks에서 제공한 암호를 사용하여 인스턴스에 로그인합니다. Stacks가 해당 암호를 생성하도록 AWS OpsWorks 하려면 사용자에 대한 RDP 액세스 권한을 명시적으로 부여해야 합니다.

사용자에 대해 RDP를 승인하려면

1. AWS OpsWorks 스택 대시보드에서 IISWalkthrough 스택을 선택합니다.
2. 스택의 탐색 창에서 [권한]을 선택합니다.
3. 권한 페이지에서 편집을 선택합니다.
4. 사용자 목록에서 필요한 권한을 부여하고자 하는 사용자의 SSH/RDP 확인란을 선택합니다. 이 사용자에게 관리자 권한도 부여하려면 [sudo/admin]도 선택합니다.



5. 저장을 선택합니다.

그러면 뒤에서 설명하는 것처럼 사용자는 암호를 얻어 인스턴스 로그인에 사용할 수 있습니다.

> **Note**
>
> 관리자로 로그인할 수도 있습니다. 자세한 내용은 관리자로 로그인 단원을 참조하십시오.

2.3단계: 사용자 지정 쿡북 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택은 기본적으로 인스턴스의 컨테이너이지만 인스턴스를 스택에 직접 추가하는 것은 아닙니다. 관련된 인스턴스 그룹을 각각 나타내는 하나 이상의 계층을 추가한 후 인스턴스를 계층에 추가합니다.

계층은 기본적으로 AWS OpsWorks Stacks가 동일한 구성으로 Amazon EC2 인스턴스 세트를 생성하는 데 사용하는 블루프린트입니다. 인스턴스는 운영 체제 기본 버전으로 시작하며, 인스턴스의 계층은 다음을 포함한 다양한 작업을 인스턴스에서 수행하여 이 청사진을 구현합니다.

- 디렉터리 및 파일 생성
- 사용자 관리
- 소프트웨어 설치 및 구성
- 서버 시작 또는 중지
- 애플리케이션 코드 및 관련 파일 배포.

계층은 Chef 레시피(줄여서 레시피)를 실행하여 인스턴스에 대한 작업을 수행합니다. 레시피란 Chef의 DSL(Domain-Specific Language)을 사용하여 인스턴스의 최종 상태를 설명하는 Ruby 애플리케이션입니다. AWS OpsWorks Stacks에서는 일반적으로 각 레시피가 설정, 구성, 배포, 배포 취소 및 종료와 같은 계층의 수명 주기 이벤트 중 하나에 할당됩니다. 인스턴스에서 수명 주기 이벤트가 발생하면 AWS OpsWorks Stacks는 이벤트의 레시피를 실행하여 적절한 작업을 수행합니다. 예를 들어, Setup 이벤트는 인스턴스 부팅이 완료된 후에 발생합니다. 그런 다음 AWS OpsWorks Stacks는 일반적으로 서버 소프트웨어 설치 및 구성, 관련 서비스 시작과 같은 작업을 수행하는 Setup 레시피를 실행합니다.

AWS OpsWorks Stacks는 각 계층에 표준 작업을 수행하는 내장 레시피 세트를 제공합니다. 추가 작업을 수행하는 사용자 지정 레시피를 구현하여 계층의 수명 주기 이벤트에 할당하면 계층의 기능을 확장할 수 있습니다. Windows 스택이 지원하는 사용자 지정 계층에는 소수의 기본적 작업만 수행하는 최소한의 레시피 세트가 있습니다. Windows 인스턴스에 기능을 추가하기 위해서는 소프트웨어 설치, 애플리케이션 배포 등을 수행하는 사용자 지정 레시피를 구현해야 합니다. 이 주제에서는 간단한 사용자 지정 계층을 생성해 IIS 인스턴스를 지원하는 방법을 설명합니다.

주제

- 쿡북 및 레시피에 대한 간략한 소개
- 레시피를 구현하여 IIS를 설치하고 시작
- 사용자 지정 쿡북 활성화

쿡북 및 레시피에 대한 간략한 소개

레시피는 인스턴스의 예상되는 상태 중 하나 이상의 측면, 즉 어떤 디렉터리가 있어야 하고, 어떤 소프트웨어 패키지를 설치해야 하고, 어떤 앱을 배포해야 하는지 등을 정의합니다. 레시피는 일반적으로 하나 이상의 관련 레시피와 구성 파일을 생성하기 위한 템플릿 등의 관련 파일을 포함하는 쿡북에 패키징되어 있습니다.

이 주제는 레시피에 대한 아주 기초적인 소개로서 쿡북을 구현하여 간단한 사용자 지정 IIS 계층을 지원하는 방법을 보여 주는 정도로 그칩니다. 쿡북에 대한 보다 전반적인 소개는 쿡북과 레시피 단원을 참조하세요. 일부 Windows 전용 주제를 포함한 상세한 쿡북 구현 자습서는 쿡북 101 단원을 참조하세요.

Chef 레시피는 기술적으로 Ruby 애플리케이션이지만 전부는 아니더라도 대부분의 코드는 Chef DSL로 되어 있습니다. DSL은 대체로 인스턴스 상태의 한 측면을 선언적으로 지정하는 데 사용할 수 있는 리소스 세트로 구성됩니다. 예를 들어 `directory` 리소스는 시스템에 추가할 디렉터리를 정의합니다. 다음 예제는 지정된 사용자에게 속하고 상위 디렉터리에서 권한을 상속하지 않는, 완전한 제어 권한이 있는 `C:\data` 디렉터리를 정의합니다.

```
directory 'C:\data' do
  rights :full_control, 'WORKGROUP\username'
  inherits false
  action :create
end
```

Chef는 레시피를 실행할 때 연결된 공급자, 즉 인스턴스 상태 수정의 세부 사항을 처리하는 Ruby 객체에 데이터를 전달함으로써 각각의 리소스를 실행합니다. 이 경우, 공급자는 지정된 구성을 가진 새 디렉터리를 생성합니다.

사용자 지정 IIS 계층을 위한 사용자 지정 쿡북이 수행해야 하는 작업은 다음과 같습니다.

- IIS 기능 설치 및 서비스 시작.

  일반적으로 이 작업은 인스턴스 부팅이 완료된 직후 설정 도중에 수행합니다.
- 인스턴스에 앱 배포(이 예제에서는 간단한 HTML 페이지).

  일반적으로 이 작업은 설정 도중에 수행합니다. 다만 앱은 정기적으로 업데이트해야 하므로 인스턴스가 온라인 상태일 때 업데이트도 배포해야 합니다.

하나의 레시피가 이 모든 작업을 수행하도록 할 수도 있지만 설정 작업과 배포 작업을 별도의 레시피가 수행하도록 하는 것이 좋습니다. 이렇게 하면 설정 코드를 실행하지 않고도 언제든 앱 업데이트를 배포할 수 있습니다. 다음은 사용자 지정 IIS 계층을 지원하도록 쿡북을 설정하는 방법을 설명합니다. 이후의 주제에서는 레시피를 구현하는 방법을 살펴봅니다.

시작하기

1. 워크스테이션의 편리한 위치에 디렉터리 `iis-cookbook`을 만듭니다.

2. `iis-cookbook`에 다음 콘텐츠가 포함된 `metadata.rb` 파일을 추가합니다.

```
name "iis-cookbook"
version "0.1.0"
```

   이 예제에서는 최소 `metadata.rb`를 사용합니다. 이 파일 사용 방법에 대한 자세한 정보는 metadata.rb를 참조하세요.

3. `recipes` 디렉터리를 `iis-cookbook`에 추가합니다.

   이 디렉터리(이름이 `recipes`여야 함)에는 쿡북의 레시피가 포함되어 있습니다.

일반적으로 쿡북에는 그 밖의 다양한 디렉터리가 포함될 수 있습니다. 예를 들어 레시피가 템플릿을 사용하여 구성 파일을 생성하는 경우, 이 템플릿은 일반적으로 `templates\default` 디렉터리로 이동합니다. 이 예제의 쿡북은 레시피만으로 구성되어 있으므로 다른 디렉터리가 필요하지 않습니다. 또한 이 예제는 단일 쿡북을 사용하지만 필요에 따라 얼마든지 많은 쿡북을 사용할 수 있습니다. 복잡한 프로젝트에는 대체로 여러 개의 쿡북이 더 좋습니다. 예를 들어 설정 작업을 위한 쿡북과 배포 작업을 쿡북을 따로 둘 수 있습니다. 더 많은 쿡북 예제는 쿡북과 레시피 단원을 참조하세요.

레시피를 구현하여 IIS를 설치하고 시작

IIS는 Windows 기능으로서 Windows 서버에 설치할 수 있는 선택적 시스템 구성 요소 세트 중 하나입니다. 다음 방법 중 하나로 레시피가 IIS를 설치하도록 할 수 있습니다.

- `powershell_script` 리소스를 사용하여 `Install-WindowsFeature` cmdlet을 실행하는 방법.
- Chef windows 쿡북의 `windows_feature` 리소스를 사용하는 방법.

  `windows` 쿡북에는 공급자가 Deployment Image Servicing and Management(DISM)를 사용하여 기능 설치를 비롯한 다양한 Windows 인스턴스 작업을 수행하는 일련의 리소스가 포함되어 있습니다.

> **Note**
>
> `powershell_script`는 Windows 레시피에 가장 유용한 리소스 중 하나입니다. 이 리소스를 사용하면 PowerShell 스크립트 또는 cmdlet을 실행하여 인스턴스에서 다양한 작업을 수행할 수 있습니다. 특히 Chef 리소스가 지원하지 않는 작업에 유용합니다.

이 예제는 PowerShell 스크립트를 실행하여 Web Server(IIS)를 설치하고 시작합니다. `windows` 쿡북은 나중에 설명합니다. `windows_feature`를 사용하여 IIS를 설치하는 방법의 예제는 Windows 기능 설치: IIS를 참조하세요.

다음 콘텐츠를 포함하는 `install.rb`라는 쿡북을 쿡북의 `recipes` 디렉터리에 추가합니다.

```
powershell_script 'Install IIS' do
  code 'Install-WindowsFeature Web-Server'
  not_if "(Get-WindowsFeature -Name Web-Server).Installed"
end

service 'w3svc' do
  action [:start, :enable]
end
```

레시피에는 다음 2개의 리소스가 포함됩니다.

powershell_script

`powershell_script`는 지정된 PowerShell 스크립트 또는 cmdlet를 실행합니다. 예제에는 다음과 같은 속성 설정이 있습니다.

- `code` – 실행할 PowerShell cmdlet.

  이 예제는 Web Server(IIS)를 설치하는 단일 `Install-WindowsFeature` cmdlet을 실행합니다. 일반적으로 `code` 속성은 줄의 수에 제한이 없기 때문에 필요하다면 얼마든지 많은 cmdlet을 실행할 수 있습니다.

- `not-if` – 아직 설치되지 않은 경우에만 레시피가 IIS를 설치하도록 하는 guard 속성.

  일반적으로 원하는 레시피는 idempotent 방식이므로 같은 작업을 두 번 이상 수행하여 시간을 낭비하지 않습니다.

모든 리소스에는 공급자가 수행할 작업을 지정하는 작업이 있습니다. 이 예제에는 명시적인 작업이 없으므로 공급자는 지정된 PowerShell 스크립트를 실행하는 기본 `:run` 작업을 수행합니다. 자세한 내용은 Windows PowerShell 스크립트 실행 단원을 참조하십시오.

service

`service`는 서비스, 이 경우에는 Web Server IIS 서비스(W3SVC)를 관리합니다. 예제는 기본 속성을 사용하여 IIS를 시작하고 활성화하는 `:start` 및 `:enable`이라는 두 가지 작업을 지정합니다.

> **Note**
>
> MSI 같은 패키지 설치 프로그램을 사용하는 소프트웨어를 설치하려면 `windows_package` 리소스를 사용하면 됩니다. 자세한 내용은 패키지 설치 단원을 참조하십시오.

사용자 지정 쿡북 활성화

AWS OpsWorks Stacks는 각 인스턴스의 로컬 캐시에서 레시피를 실행합니다. 사용자 지정 레시피를 실행하려면 다음과 같은 작업을 수행해야 합니다.

- 쿡북을 원격 리포지토리에 저장합니다.

  AWS OpsWorks Stacks는이 리포지토리에서 각 인스턴스의 로컬 캐시로 쿡북을 다운로드합니다.
- 스택을 편집하여 사용자 지정 쿡북을 활성화합니다.

  사용자 지정 쿡북은 기본적으로 비활성화되어 있으므로 스택용 사용자 지정 쿡북을 활성화하고 리포지토리 URL 및 관련 정보를 제공해야 합니다.

AWS OpsWorks Stacks는 사용자 지정 쿡북에 대해 S3 아카이브 및 Git 리포지토리를 지원합니다.이 예제에서는 S3 아카이브를 사용합니다. 자세한 내용은 쿡북 리포지토리 단원을 참조하십시오.

S3 아카이브를 사용하려면

1. `iis-cookbook` 디렉터리의 `.zip` 아카이브를 생성합니다.

   AWS OpsWorks Stacks는 Windows 스택에 대한 `.tgz` (gzip 압축 tar) 아카이브도 지원합니다.
2. 이 아카이브를 미국 서부(캘리포니아 북부) 리전의 S3 버킷에 업로드하고 이 파일을 퍼블릭으로 설정합니다. 또한 프라이빗 S3 아카이브를 사용할 수도 있지만 이 예제에는 퍼블릭 아카이브면 충분합니다. 퍼블릭 아카이브가 작업하기 더 간단합니다.

   a. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

   b. `us-west-1`에 아직 버킷이 없는 경우 버킷 생성을 선택하여 미국 서부(캘리포니아 북부) 리전에서 버킷을 생성합니다.

   c. 버킷 목록에서 파일을 업로드할 버킷의 이름을 선택한 다음, 업로드를 클릭합니다.

   d. 파일 추가를 선택합니다.

   e. 업로드할 아카이브 파일을 선택한 다음, 열기를 선택합니다.

f. [업로드 - 파일 및 폴더 선택] 대화 상자 아래에서 [세부 정보 설정]을 선택합니다.

g. [세부 정보 설정] 대화 상자 아래에서 [권한 설정]을 선택합니다.

h. [권한 설정] 대화 상자에서 [모든 항목을 퍼블릭으로 설정]을 선택합니다.

i. [권한 설정] 대화 상자 아래에서 [업로드 시작]을 선택합니다. 업로드가 완료되면 `iis-cookbook.zip` 파일이 버킷에 나타납니다.

j. 버킷을 선택한 다음, 해당 버킷의 속성 탭을 선택합니다. 나중에 사용하기 위해 [링크] 옆에 있는 아카이브 파일의 URL을 기록해 둡니다.

Amazon S3 버킷에 파일을 업로드하는 방법에 대한 자세한 내용은 Amazon S3 콘솔 사용 설명서의 S3 버킷에 파일 및 폴더를 업로드하려면 어떻게 해야 합니까?를 참조하세요.

> **Important**
>
> 지금까지 안내서에서 소요된 시간은 얼마 되지 않으며, AWS OpsWorks Stacks 서비스 자체는 무료입니다. 단, Amazon S3 스토리지 등 사용하는 AWS 리소스에 대해서는 요금을 지불해야 합니다. 아카이브를 업로드하는 즉시 요금이 발생하기 시작합니다. 자세한 내용은 AWS 요금을 참조하세요.

스택에 대한 사용자 지정 쿡북을 활성화하려면

1. AWS OpsWorks Stacks 콘솔의 탐색 창에서 스택을 선택한 다음 오른쪽 상단의 스택 설정을 선택합니다.
2. [설정] 페이지의 오른쪽 위에서 [편집]을 선택합니다.
3. [설정] 페이지에서 [사용자 지정 Chef 쿡북 사용]을 [예]로 설정하고 다음 정보를 입력합니다.
   - 리포지토리 유형– S3 아카이브.
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 파일의 S3 URL.
4. 저장을 선택하여 스택 구성을 업데이트합니다.

AWS OpsWorks Stacks는 모든 새 인스턴스에 사용자 지정 쿡북을 설치합니다. AWS OpsWorks Stacks는 온라인 인스턴스에서 자동으로 사용자 지정 쿡북을 설치하거나 업데이트하지 않습니다. 이 작업은 뒤에 설명하는 것처럼 수동으로 수행할 수 있습니다.

2.4단계: IIS 계층 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

쿡북에는 IIS를 설치하고 시작하는 레시피가 하나 있습니다. 계층을 생성하고 IIS 인스턴스 작동을 확인하는 데는 이것으로 충분합니다. 나중에 이 계층에 애플리케이션 배포 기능을 추가합니다.

계층 생성

스택에 계층을 추가하는 것으로 시작하세요. 그런 다음 적절한 수명 주기 이벤트에 사용자 지정 레시피를 할당하여 해당 계층에 기능을 추가합니다.

IIS 계층을 스택에 추가하려면

1. 탐색 창에서 계층을 선택한 다음 계층 추가를 선택합니다.
2. 다음과 같이 계층을 구성합니다.
   - 이름– **`IISExample`**
   - 짧은 이름 – **`iisexample`**

     AWS OpsWorks Stacks는 짧은 이름을 사용하여 내부적으로 계층을 식별합니다. 이 예제에서는 짧은 이름을 사용하지 않지만 사용자가 이 이름을 사용해 레시피에서 계층을 식별할 수 있습니다. 짧은 이름은 어느 것이든 지정할 수 있지만 소문자 영숫자와 몇 개의 구두점으로만 구성할 수 있습니다. 자세한 내용은 사용자 지정 계층 단원을 참조하십시오.
3. [계층 추가]를 선택합니다.

이때 IISWalkthrough에 인스턴스를 추가하고 시작하는 경우, AWS OpsWorks Stacks는 자동으로 쿡북을 설치하지만 `install.rb`를 실행하지는 않습니다. 인스턴스가 온라인 상태가 된 후 레시피 실행 스택 명령을 사용하면 수동으로 레시피를 실행할 수 있습니다. 그러나 더 나은 접근 방식은 계층의 수명 주기 이벤트 중 하나에 레시피를 할당하는 것입니다. 그러면 AWS OpsWorks Stacks는 인스턴스 수명 주기의 적절한 지점에서 레시피를 자동으로 실행합니다.

인스턴스 부팅이 완료되는 즉시 IIS를 설치하고 시작합니다. 이렇게 하려면 `install.rb`를 계층의 `Setup` 이벤트에 할당합니다.

수명 주기 이벤트에 레시피를 할당하려면

1. 탐색 창에서 [계층]를 선택합니다.
2. [IISExample] 계층에 해당하는 상자에서 [레시피]를 선택합니다.
3. 오른쪽 위에서 편집을 선택합니다.
4. 사용자 지정 Chef 레시피의 설정 레시피 상자에 **`iis-cookbook::install`**을 입력합니다.

   > **Note**
   >
   > *`cookbook-name`*`::`*`recipe-name`*을 사용하여 레시피를 식별합니다. 여기서 레시피 이름의 `.rb` 접미사는 생략합니다.

5. [+]를 선택하여 계층에 레시피를 추가합니다. 이후에 쉽게 제거할 수 있도록 레시피 옆에 빨간색 x가 나타납니다.
6. 저장을 선택하여 새 구성을 저장합니다. 이제 사용자 지정 설정 레시피에 `iis-cookbook::install`이 포함되어 있어야 합니다.

인스턴스를 계층에 추가하고 시작

계층에 인스턴스를 추가하고 인스턴스를 시작하여 레시피를 시도할 수 있습니다. AWS OpsWorks Stacks는 인스턴스 부팅이 완료되는 즉시 설정 `install.rb` 중에 쿡북을 자동으로 설치하고 실행합니다.

인스턴스를 계층에 추가하고 시작하려면

1. AWS OpsWorks 스택 탐색 창에서 인스턴스를 선택합니다.
2. [IISExample] 계층에서 [인스턴스 추가]를 선택합니다.
3. 적절한 크기를 선택합니다. t2.micro(또는 가능한 가장 작은 크기)면 이 예제에 충분해야 합니다.
4. [인스턴스 추가]를 선택합니다. 기본적으로 AWS OpsWorks Stacks는 계층의 짧은 이름에 정수를 추가하여 인스턴스 이름을 생성하므로 인스턴스의 이름은 iisexample1이어야 합니다.
5. 인스턴스의 작업 열에서 시작을 선택하여 인스턴스를 시작합니다. 그러면 AWS OpsWorks Stacks가 EC2 인스턴스를 시작하고 Setup 레시피를 실행하여 인스턴스를 구성합니다. 이 시점에 계층에 Deploy 레시피가 있는 경우 Setup 레시피가 완료된 후 AWS OpsWorks Stacks가 이를 실행합니다.

이 프로세스를 완료하는 데에는 몇 분 가량 소요될 수 있으며 그 동안 [상태] 열에는 일련의 상태가 표시됩니다. [온라인] 상태가 되면 설정 프로세스가 완료된 것으로 인스턴스가 사용할 준비가 된 것입니다.

IIS 설치 및 실행 확인

RDP를 사용하여 인스턴스에 연결하고 설정 레시피가 올바로 작동했는지 확인할 수 있습니다.

IIS가 설치되어 실행 중인지 확인하려면

1. 탐색 창에서 인스턴스를 선택하고 iisexample1 인스턴스의 작업 열에서 rdp를 선택합니다. AWS OpsWorks Stacks는 지정된 기간 후에 만료되는 RDP 암호를 자동으로 생성합니다.
2. [세션 유효 시간]을 2시간으로 설정하고 [암호 생성]을 선택합니다.
3. AWS OpsWorks Stacks는 사용자의 편의를 위해 암호와 인스턴스의 퍼블릭 DNS 이름 및 사용자 이름을 표시합니다. 이러한 3가지 정보를 모두 복사한 다음 [확인 및 닫기]를 클릭합니다.
4. RDP 클라이언트를 열고 3단계의 데이터를 사용하여 인스턴스에 연결합니다.
5. 인스턴스에서 Windows 탐색기를 열고 `C:` 드라이브를 검사합니다. 이 드라이브에는 IIS 설치 시 생성된 `C:\inetpub` 디렉터리가 있어야 합니다.
6. 제어판의 [관리 도구] 애플리케이션을 연 다음 [서비스]를 엽니다. 목록의 맨 아래 근처에 IIS 서비스가 있어야 합니다. 이 서비스의 이름은 World Wide Web Publishing Service로, 상태는 [실행 중]이어야 합니다.
7. AWS OpsWorks Stacks 콘솔로 돌아가서 iisexample1 인스턴스의 퍼블릭 IP 주소를 선택합니다. Amazon EC2 콘솔이 아닌 AWS OpsWorks Stacks에서이 작업을 수행해야 합니다. 그러면 HTTP 요청이 이 주소로 자동으로 전송되고 기본 IIS 시작 페이지가 열려야 합니다.

다음 주제에서는 앱(이 예제에서는 간단한 정적 HTML 페이지)을 인스턴스에 배포하는 방법을 살펴봅니다. 하지만 잠시 쉬고 싶다면 iiexample1 인스턴스의 작업 열에서 stop을 선택하여 인스턴스를 중지하고 불필요한 요금이 발생하지 않도록 하세요. 계속할 준비가 되었을 때 인스턴스를 다시 시작할 수 있습니다.

2.5단계: 앱 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

IIS 설치는 애플리케이션의 코드와 관련 파일을 위한 `C:\inetpub\wwwroot` 디렉터리를 생성합니다. 다음 단계는 이 디렉터리에 앱을 설치하는 것입니다. 이 예제에서는 정적 HTML 홈 페이지인 `default.html`을 `C:\inetpub\wwwroot`에 설치합니다. 일반적인 방법을 쉽게 확장하여 ASP.NET 애플리케이션 같은 보다 복잡한 시나리오도 처리할 수 있습니다.

애플리케이션의 파일을 쿡북에 포함시키고, `install.rb`가 이러한 파일을 `C:\inetpub\wwwroot`에 복사하도록 할 수 있습니다. 이렇게 하는 방법의 예제는 예제 6: 파일 생성 섹션을 참조하세요. 다만 이 방법은 아주 유연하거나 효율적이지는 않으며, 일반적으로는 쿡북 개발과 애플리케이션 개발을 분리하는 것이 좋습니다.

기본 해결 방법은 쿡북 리포지토리뿐 아니라 원하는 리포지토리에서 애플리케이션의 코드 및 관련 파일을 가져와 각 IIS 서버 인스턴스에 설치하는 별도의 배포 레시피를 구현하는 것입니다. 이 방법은 쿡북 개발과 애플리케이션 개발을 분리하며, 앱을 업데이트해야 하는 경우에는 쿡북을 업데이트할 필요 없이 배포 레시피를 다시 실행하기만 하면 됩니다.

이 주제에서는 `default.htm`을 IIS 서버에 배포하는 간단한 배포 레시피를 구현하는 방법을 살펴봅니다. 이 예제는 보다 복잡한 애플리케이션으로 쉽게 확장할 수 있습니다.

주제

- 애플리케이션을 생성하여 리포지토리에 저장
- 레시피를 구현하여 애플리케이션 배포
- 인스턴스의 쿡북 업데이트
- 사용자 지정 IIS 계층에 레시피 추가
- 앱 추가
- 앱 배포 및 애플리케이션 열기

애플리케이션을 생성하여 리포지토리에 저장

선호하는 어떤 리포지토리도 애플리케이션에 사용할 수 있습니다. 간단히 하기 위해 이 예제에서는 `default.htm`을 퍼블릭 S3 버킷에 저장합니다.

애플리케이션을 생성하려면

1. 워크스테이션의 편리한 위치에 디렉터리 `iis-application`을 만듭니다.

2. `iis-application`에 다음 콘텐츠가 포함된 `default.htm` 파일을 추가합니다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>IIS Example</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

3. S3 버킷을 생성하고, 버킷에 `default.htm`을 업로드하고, 나중에 사용하기 위해 이 URL을 적어 둡니다. 간단하게 설명하기 위해 이 파일을 퍼블릭으로 설정합니다.

> **Note**
>
> 이는 매우 간단한 애플리케이션이지만 프로덕션 레벨 애플리케이션을 처리하기 위해 기본 원칙을 확장할 수 있습니다.
>
> - 여러 파일이 있는 복잡한 애플리케이션의 경우 일반적으로 `iis-application`의 .zip 아카이브를 생성하여 S3 버킷에 업로드하는 것이 더 간단합니다.
>
> 그런 다음 이 .zip 파일을 다운로드하고 내용을 적절한 디렉터리에 추출할 수 있습니다. 여러 파일을 다운로드하거나 디렉터리 구조를 생성하는 등의 작업을 수행할 필요가 없습니다.
> - 프로덕션 애플리케이션의 경우 파일을 프라이빗으로 유지해야 할 것입니다. 프라이빗 S3 버킷에서 레시피 다운로드 파일을 얻는 방법의 예는 AWS OpsWorks Stacks Windows 인스턴스에서 SDK for Ruby 사용 단원을 참조하세요.
> - 애플리케이션은 적절한 모든 리포지토리에 저장할 수 있습니다.
>
> 애플리케이션은 일반적으로 리포지토리의 퍼블릭 API를 사용하여 다운로드합니다. 이 예에서는 Amazon S3 API를 사용합니다. 예를 들어, GitHub에 애플리케이션을 저장하는 경우 GitHub API를 사용할 수 있습니다.

레시피를 구현하여 애플리케이션 배포

다음 내용이 포함된 `deploy.rb` 레시피를 `iis-cookbook recipes` 디렉터리에 추가합니다.

```
chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    #1
    # Aws.config[:ssl_ca_bundle] = 'C:\ProgramData\Git\bin\curl-ca-bundle.crt'
    Aws.use_bundled_cert!

    #2
    query = Chef::Search::Query.new
    app = query.search(:aws_opsworks_app, "type:other").first
    s3region = app[0][:environment][:S3REGION]
    s3bucket = app[0][:environment][:BUCKET]
    s3filename = app[0][:environment][:FILENAME]

    #3
    s3_client = Aws::S3::Client.new(region: s3region)
    s3_client.get_object(bucket: s3bucket,
                         key: s3filename,
                         response_target: 'C:\inetpub\wwwroot\default.htm')
  end
  action :run
end
```

이 예제는 SDK for Ruby v2를 사용하여 파일을 다운로드합니다. 그러나 AWS OpsWorks Stacks는 Windows 인스턴스에이 SDK를 설치하지 않으므로 레시피는 해당 작업을 처리하는 `chef_gem` 리소스로 시작합니다.

> **Note**
>
> `chef_gem` 리소스는 레시피가 사용하는 버전인 Chef의 전용 Ruby 버전에 gem을 설치합니다. 시스템 전체 Ruby 버전용 gem을 설치하려면 gem_package 리소스를 사용하세요.

레시피의 대부분인 `ruby_block` 리소스는 SDK for Ruby를 사용하여 `default.htm`을 다운로드하는 Ruby 코드의 블록을 실행합니다. `ruby_block`의 코드는 코드 예제에서 번호가 지정된 설명에 해당하는 다음과 같은 섹션으로 나눌 수 있습니다.

1: 인증서 번들 지정

Amazon S3는 SSL을 사용하므로 S3 버킷에서 객체를 다운로드하려면 적절한 인증서가 필요합니다. SDK for Ruby v2에는 인증서 번들이 포함되어 있지 않으므로 인증서 번들을 제공하고 이를 사용하도록 SDK for Ruby를 구성해야 합니다. AWS OpsWorks Stacks는 인증서 번들을 직접 설치하지 않지만 인증서 번들()이 포함된 Git을 설치합니다`curl-ca-bundle.crt`. 편의상 이 예제에서는 SDK for Ruby가 SSL용 Git 인증서 번들을 사용하도록 구성합니다. 자체 SDK를 설치하고 적절히 구성할 수도 있습니다.

2: 리포지토리 데이터 검색

Amazon S3에서 객체를 다운로드하려면 AWS 리전, 버킷 이름, 키 이름이 필요합니다. 뒤에서 설명하겠지만 이 예제는 환경 변수 세트를 앱에 연결하여 이 정보를 제공합니다. 앱을 배포할 때 AWS OpsWorks Stacks는 인스턴스의 노드 객체에 속성 세트를 추가합니다. 이 속성들은 기본적으로 환경 변수를 비롯한 앱 구성이 포함된 해시 테이블입니다. 이 애플리케이션의 앱 속성은 다음과 비슷한 JSON 형식으로 되어 있습니다.

```
{
  "app_id": "8f71a9b5-de7f-451c-8505-3f35086e5bb3",
  "app_source": {
      "password": null,
      "revision": null,
      "ssh_key": null,
      "type": "other",
      "url": null,
      "user": null
  },
  "attributes": {
      "auto_bundle_on_deploy": true,
      "aws_flow_ruby_settings": {},
      "document_root": null,
      "rails_env": null
  },
  "data_sources": [{"type": "None"}],
  "domains": ["iis_example_app"],
  "enable_ssl": false,
  "environment": {
      "S3REGION": "us-west-2",
```

```
        "BUCKET": "windows-example-app",
        "FILENAME": "default.htm"
    },
    "name": "IIS-Example-App",
    "shortname": "iis_example_app",
    "ssl_configuration": {
        "certificate": null,
        "private_key": null,
        "chain": null
    },
    "type": "other",
    "deploy": true
}
```

앱의 환경 변수는 `[:environment]` 속성에 저장됩니다. 이 환경 변수를 검색하려면 Chef 검색 쿼리를 사용하여 앱의 해시 테이블을 검색합니다(`aws_opsworks_app` 노드 아래에 있음). 이 앱은 `other` 유형으로 정의되므로 쿼리는 해당 유형의 앱을 검색합니다. 레시피는 이 인스턴스에 앱이 하나뿐이라는 점을 활용하므로 관심 대상 해시 테이블은 단순히 `app[0]`입니다. 편의상 레시피는 그 다음에 리전, 버킷 및 파일 이름을 변수에 할당합니다.

Chef 검색을 사용하는 방법에 대한 자세한 정보는 Chef 검색을 사용하여 속성 값 가져오기 단원을 참조하세요.

3: 파일 다운로드

레시피의 세 번째 부분은 S3 클라이언트 객체를 생성하고 `get_object` 메서드를 사용하여 `default.htm`을 인스턴스의 `C:\inetpub\wwwroot` 디렉터리에 다운로드합니다.

> **Note**
>
> 레시피는 Ruby 애플리케이션입니다. 따라서 Ruby 코드가 반드시 `ruby_block`에 있을 필요는 없습니다. 하지만 레시피 본문의 코드가 먼저 실행된 다음 리소스들이 순서대로 실행됩니다. 이 예제에서는 다운로드 코드를 레시피 본문에 배치하면 코드가 실패합니다. `chef_gem` 리소스가 아직 SDK for Ruby를 설치하지 않았기 때문입니다. 리소스가 실행될 때 `ruby_block` 리소스 내 코드가 실행되며, 이는 `chef_gem` 리소스가 SDK for Ruby를 설치한 후에 이루어집니다.

인스턴스의 쿡북 업데이트

AWS OpsWorks Stacks는 새 인스턴스에 사용자 지정 쿡북을 자동으로 설치합니다. 하지만 기존 인스턴스에서 작업 중이므로 수동으로 쿡북을 업데이트해야 합니다.

인스턴스의 쿡북을 업데이트하려면

1. `iis-cookbook`의 `.zip` 아카이브를 만들어 S3 버킷에 업로드합니다.

   이렇게 하면 기존 쿡북을 덮어쓰지만 URL은 동일하게 유지되므로 스택 구성을 업데이트할 필요가 없습니다.
2. 인스턴스가 온라인 상태가 아니면 인스턴스를 다시 시작하세요.
3. 인스턴스가 온라인 상태가 되면 탐색 창에서 [스택]을 선택한 다음 [명령 실행]을 선택합니다.
4. [명령]에 대해서는 [사용자 지정 쿡북 업데이트]를 선택합니다. 이 명령은 업데이트된 쿡북을 인스턴스에 설치합니다.
5. [사용자 지정 쿡북 업데이트]를 선택합니다. 명령을 완료하려면 몇 분 정도 걸릴 수 있습니다.

사용자 지정 IIS 계층에 레시피 추가

`install.rb`와 마찬가지로 배포를 처리하는 좋은 방법은 `deploy.rb`를 적절한 수명 주기 이벤트에 할당하는 것입니다. 일반적으로는 배포 레시피들을 Deploy 이벤트에 할당하며, 총칭하여 Deploy 레시피라고 합니다. 레시피를 Deploy 이벤트에 할당해도 이벤트가 트리거되지는 않습니다. 그 대신

- 새 인스턴스의 경우 Setup 레시피가 완료된 후 AWS OpsWorks Stacks가 Deploy 레시피를 자동으로 실행하므로 새 인스턴스에는 현재 애플리케이션 버전이 자동으로 지정됩니다.
- 온라인 인스턴스의 경우, deploy 명령을 사용하여 수동으로 새 애플리케이션 또는 업데이트된 애플리케이션을 설치합니다.

  이 명령은 스택의 인스턴스에서 Deploy 레시피를 실행하는 Deploy 이벤트를 트리거합니다.

계층의 Deploy 이벤트에 deploy.rb를 할당하려면

1. 탐색 창에서 계층을 선택한 다음 계층 IISExample 아래에 있는 레시피를 선택합니다.
2. 사용자 지정 Chef 레시피에서 **`iis-cookbook::deploy`**를 레시피 배포 상자에 추가하고 +를 선택하여 레시피를 계층에 추가합니다.
3. 저장을 선택하여 새 구성을 저장합니다. 이제 사용자 지정 Deploy 레시피에 `iis-cookbook::deploy`가 포함되어 있어야 합니다.

앱 추가

마지막 작업은 스택에 앱을 추가하여 AWS OpsWorks Stacks 환경에서 애플리케이션을 나타내는 것입니다. 앱에는 애플리케이션의 표시 이름과 리포지토리에서 앱을 다운로드하는 데 필요한 데이터 같은 메타데이터가 포함되어 있습니다.

스택에 앱을 추가하려면

1. 탐색 창에서 [앱]을 선택한 다음 [앱 추가]를 선택합니다.
2. 다음 설정으로 앱을 구성합니다:
   - 이름 – I**IIS-Example-App**
   - 리포지토리 유형 –기타
   - 환경 변수 – 다음 환경 변수 3개를 추가합니다.
     - **S3REGION** - 버킷의 리전(이 경우의 `us-west-1`).
     - **BUCKET** - 버킷의 이름(예: `windows-example-app`).
     - **FILENAME** - 파일 이름: **`default.htm`**.
3. 나머지 설정에 대해서는 기본값을 수락하고 앱 추가를 선택하여 스택에 앱을 추가합니다.

> **Note**
>
> 이 예제에서는 환경 변수를 사용하여 다운로드 데이터를 제공합니다. 다른 방법은 S3 Archive 리포지토리 유형을 사용하고 파일의 URL을 제공하는 것입니다. AWS OpsWorks Stacks는 AWS 자격 증명과 같은 선택적 데이터와 함께 정보를 앱의 `app_source` 속성에 추가합니다. 배포 레시피는 앱 속성에서 URL을 가져온 다음 구문 분석하여 리전, 버킷 이름 및 파일 이름을 추출해야 합니다.

앱 배포 및 애플리케이션 열기

AWS OpsWorks Stacks는 앱을 새 인스턴스에 자동으로 배포하지만 온라인 인스턴스에는 배포하지 않습니다. 인스턴스가 이미 실행 중이기 때문에 수동으로 앱을 배포해야 합니다.

앱을 배포하려면

1. 탐색 창에서 앱을 선택한 다음 앱의 작업 열에서 배포를 선택합니다.

2. 명령은 배포로 설정되어야 합니다. 배포 앱 페이지의 오른쪽 하단에서 배포를 선택합니다. 명령을 완료하려면 몇 분 정도 걸릴 수 있습니다.

   배포가 완료된 후에는 앱 페이지로 돌아갑니다. 상태 표시기에 성공이 녹색으로 표시되고 앱 이름 옆에는 녹색 확인 표시가 나타나 배포에 성공했음을 알립니다.

> **Note**
>
> Windows 앱은 항상 기타 앱 유형이므로 앱을 배포하면 다음이 수행됩니다.
>
> - 앞서 설명한 것처럼 앱의 데이터를 스택 구성 및 배포 속성에 추가합니다.
> - 스택의 인스턴스에서 Deploy 레시피를 실행하는 Deploy 이벤트를 트리거합니다.

> **Note**
>
> 실패한 배포 또는 애플리케이션 문제 해결에 대한 자세한 정보는 레시피 디버깅 단원을 참조하세요.

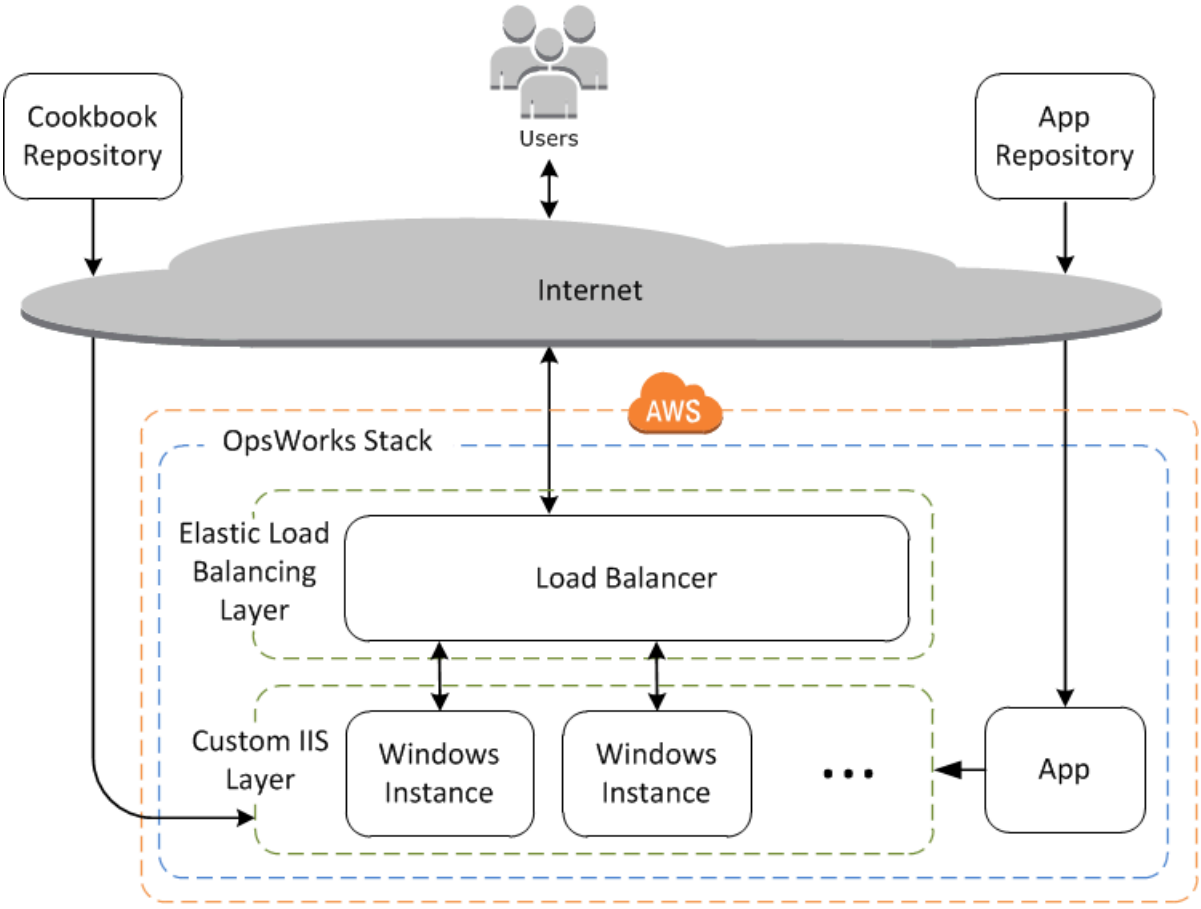
이제 앱이 설치되었습니다. 탐색 창에서 인스턴스를 선택한 다음 인스턴스의 퍼블릭 IP 주소를 선택하여 열 수 있습니다. 이렇게 하면 인스턴스로 HTTP 요청이 전송됩니다. 브라우저에 다음과 같이 표시되어야 합니다.

Hello World!

3단계: IISExample 확장

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

들어오는 사용자 요청이 하나의 t2.micro 인스턴스로 처리할 수 있는 한도에 접근하는 경우, 서버 용량을 늘려야 합니다. 더 큰 인스턴스로 옮길 수도 있지만 그것도 한계가 있습니다. 보다 유연한 방법은 스택에 인스턴스를 추가하고 로드 밸런서 뒤에 배치하는 것입니다. 기본 아키텍처는 다음과 같습니다.



여러 장점이 있지만 이 방법은 큰 단일 인스턴스보다 훨씬 견고합니다.

- 인스턴스 중 하나에 장애가 발생하면 로드 밸런서가 들어오는 요청을 나머지 인스턴스에 분산시키므로 애플리케이션은 계속 작동합니다.
- 인스턴스들을 서로 다른 가용 영역에 배치하면(권장되는 관행) 가용 영역 하나에 문제가 발생하더라도 애플리케이션이 계속 작동합니다.

AWS OpsWorks Stacks를 사용하면 스택을 쉽게 확장할 수 있습니다. 이 섹션에서는 IISExample에 두 번째 24/7 PHP 서버 인스턴스를 추가하고 두 인스턴스 모두 Elastic Load Balancing 뒤에 배치하여 스택을 확장하는 기본적인 방법을 설명합니다. 절차를 쉽게 확장하여 임의의 수의 24/7 인스턴스를 추가하거나 시간 기반 인스턴스를 사용하여 AWS OpsWorks Stacks가 스택을 자동으로 확장하도록 할 수 있습니다. 자세한 내용은 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리 단원을 참조하십시오.

로드 밸런서 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Elastic Load Balancing은 수신 애플리케이션 트래픽을 여러 Amazon EC2 인스턴스에 자동으로 분산하는 AWS 서비스입니다. 로드 밸런서는 두 가지 목적을 충족할 수 있습니다. 명백한 목적은 애플리케이션 서버 간에 로드를 평활화하는 것입니다. 많은 사이트에서는 애플리케이션 서버 및 데이터베이스를 직접 사용자 액세스로부터 격리하기를 선호합니다. 트래픽을 분산하는 것에 더하여 Elastic Load Balancing은 다음 기능을 수행합니다.

- 비정상 Amazon EC2 인스턴스를 탐지합니다.

  또한 비정상 인스턴스가 복원될 때까지 트래픽을 나머지 정상 인스턴스로 다시 라우팅합니다.
- 수신 트래픽에 맞춰 요청 처리 용량을 자동으로 조정합니다.

> **Note**
>
> AWS OpsWorks Stacks는 Application Load Balancer를 지원하지 않습니다. Classic Load Balancer는 AWS OpsWorks Stacks와 함께만 사용할 수 있습니다.

Elastic Load Balancing은 종종 계층으로 불리지만 다른 내장 계층과는 약간 다르게 작동합니다. 계층을 생성하고 여기에 인스턴스를 추가하는 대신 Amazon EC2 콘솔을 사용하여 Elastic Load Balancing 로드 밸런서를 생성한 다음 기존 계층 중 하나, 일반적으로 애플리케이션 서버 계층에 연결합니다. 그런 다음 AWS OpsWorks Stacks는 계층의 기존 인스턴스를 서비스에 등록하고 새 인스턴스를 자동으로 추가합니다. 다음 절차에서는 로드 밸런서를 추가하는 방법에 대해 설명합니다.

사용자 지정 IIS 계층에 로드 밸런서를 연결하려면

1. Amazon EC2 콘솔을 사용하여 IISExample의 새 로드 밸런서를 생성합니다. 자세한 내용은 Elastic Load Balancing 시작하기를 참조하세요. [로드 밸런서 생성] 마법사를 실행하면 다음과 같이 로드 밸런서를 구성합니다.

1: 로드 밸런서 정의

로드 밸런서에 IIS-LB와 같이 쉽게 알아볼 수 있는 이름을 할당하여 AWS OpsWorks Stacks 콘솔에서 쉽게 찾을 수 있도록 합니다. 나머지 설정에 대해서는 기본값을 수락한 다음 [다음: 보안 그룹 할당]을 선택합니다.

2: 보안 그룹 지정

사용자 계정에서 기본 VPC를 지원하는 경우 마법사에 이 페이지가 표시되어 로드 밸런서의 보안 그룹을 확인합니다. EC2 Classic의 경우에는 이 페이지가 표시되지 않습니다.

이 연습에서는 [기본 VPC 보안 그룹]을 지정한 다음 [다음: 보안 설정 구성]를 선택합니다.

3: 보안 설정 구성

이 연습에서는 로드 밸런서가 보안 리스너(즉, 프런트 엔드 연결에 대해 HTTPS 또는 SSL)를 사용해야 하므로 [다음: 상태 검사 구성]을 선택하여 계속 진행합니다.

4: 상태 확인 구성

ping 경로를 **/**로 설정합니다. 나머지 설정에 대해서는 기본값을 수락한 다음 [다음: EC2 인스턴스 추가]를 선택합니다.

5: EC2 인스턴스 추가

AWS OpsWorks Stacks는 로드 밸런서에 인스턴스를 자동으로 등록합니다. 계속하려면 Next 태그 추가를 선택합니다.

6: 태그 추가

이 예제에서는 태그를 사용하지 않습니다. [검토 및 생성]을 선택합니다.

7: 검토

선택 항목을 검토하고 [만들기]를 선택한 다음 [닫기]를 선택합니다. 그러면 로드 밸런서가 시작됩니다.

2. 계정이 기본 VPC를 지원하는 경우 로드 밸런서를 시작한 후 로드 밸런서의 보안 그룹에 적절한 인바운드 규칙이 있는지 확인해야 합니다. 기본 규칙은 어떠한 인바운드 트래픽도 허용하지 않습니다.

   1. Amazon EC2 탐색 창에서 보안 그룹을 선택합니다.
   2. [기본 VPC 보안 그룹]을 선택합니다.
   3. 인바운드 탭에서 편집을 선택합니다.

4. 이 연습에서는 Source를 Anywhere로 설정합니다. 그러면 로드 밸런서에 모든 IP 주소에서의 수신 트래픽을 허용하도록 지시합니다.
5. 저장을 클릭합니다.

3. AWS OpsWorks Stacks 콘솔로 돌아갑니다. [계층] 페이지에서 [네트워크]를 선택합니다.
4. Elastic Load Balancing에서 1단계에서 생성한 IIS-LB 로드 밸런서를 선택한 다음 저장을 클릭합니다.

   로드 밸런서를 계층에 연결한 후 AWS OpsWorks Stacks는 계층의 현재 인스턴스를 자동으로 등록하고 온라인 상태가 되면 새 인스턴스를 추가합니다.
5. [계층] 페이지에서 로드 밸런서의 이름을 클릭하여 로드 밸런서의 세부정보 페이지를 엽니다. 로드 밸런서 페이지의 인스턴스 옆에 있는 녹색 확인 표시는 인스턴스가 상태 확인을 통과했음을 나타냅니다.

이제 로드 밸런서로 요청을 전송하여 IIS-Example-App을 실행할 수 있습니다.

로드 밸런서를 통해 IIS-Example-App을 실행하려면

1. [계층]을 선택합니다. IIS-ELB 로드 밸런서가 계층으로 나열되어야 하며 상태 열에 인스턴스 하나가 녹색으로 표시되어야 합니다. 이 표시는 정상 인스턴스를 나타냅니다.
2. 로드 밸런서의 DNS 이름을 선택하여 IIS-Example-App을 실행합니다. 로드 밸런서의 이름 아래에 DNS 이름이 표시되어야 하며 DNS 이름은 `IIS-LB-1802910859.us-west-2.elb.amazonaws.com`와 유사해야 합니다. 로드 밸런서는 인스턴스에 요청을 전달하고 응답을 반환합니다. 이 응답은 인스턴스의 퍼블릭 IP 주소를 클릭했을 때 얻는 응답과 정확히 동일해야 합니다.

이 시점에서는 인스턴스가 하나뿐이므로 로드 밸런서가 실제로 큰 도움이 되지 않습니다. 하지만 이제 추가 인스턴스를 계층에 추가할 수 있습니다.

계층에 인스턴스를 추가하려면

1. [인스턴스] 및 [+ 인스턴스]를 차례로 선택하여 계층에 다른 인스턴스를 추가합니다.
2. 인스턴스를 시작합니다.

새 인스턴스이므로 AWS OpsWorks Stacks는 현재 사용자 지정 쿡북을 자동으로 설치하고 설정 중에 현재 앱 버전을 배포합니다. 인스턴스가 온라인 상태가 되면 AWS OpsWorks Stacks는 자동으로 이를

로드 밸런서에 추가하므로 인스턴스는 즉시 요청 처리를 시작합니다. 애플리케이션이 여전히 작동 중인지 확인하려면 로드 밸런서의 DNS 이름을 다시 선택하면 됩니다.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 안내서에서는 간단한 Windows 애플리케이션 서버 스택을 설정하는 기본 사항을 살펴봤습니다. 아래는 다음 단계에 대한 몇 가지 제안입니다.

- 자세한 내용을 알아보려면에서 쿡북 구현에 대한 자습서를 시작하기 - 쿡북 제공하고 여러 AWS OpsWorks Stacks 관련 예제를 포함합니다.
- Amazon Relational Database Service(Amazon RDS) 계층을 스택에 추가하여 백엔드 데이터베이스 서버로 사용할 수 있습니다. 애플리케이션을 데이터베이스에 연결하는 방법은 사용자 지정 레시피 사용 단원을 참조하세요.

Stacks에서 AWS OpsWorks 쿡북 시작하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

프로덕션 수준 AWS OpsWorks Stacks 스택에는 일반적으로 사용자 지정이 필요하며, 이는 종종 사용자 지정 Chef 쿡북을 구현하는 것을 의미합니다. 쿡북은 레시피라고 하는 지침 등의 구성 정보가 포함된 패키지 파일입니다. 레시피는 Ruby 언어 구문으로 작성된 하나 이상의 지침 세트로서 사용할 리소스와 이러한 리소스가 적용되는 순서를 지정합니다. Chef에서 사용되는 리소스란 구성 정책 설명입니

다. 이 연습에서는 Stacks용 Chef AWS OpsWorks 쿡북 구현에 대한 기본 소개를 제공합니다. Chef, 레시피, 리소스에 대한 자세한 정보는 다음 단계의 링크를 참조하세요.

이 안내서에서는 주로 본인의 쿡북을 만드는 방법을 설명합니다. Chef Supermarket과 같은 웹 사이트에서 커뮤니티가 제공하는 쿡북을 사용할 수도 있습니다. 커뮤니티 쿡북 시작을 돕기 위해 이 안내서 뒷부분에 Chef Supermarket의 커뮤니티 쿡북 사용 지침이 포함되어 있습니다.

이 안내서를 시작하기 전에 몇 가지 설정 단계를 완료하세요. 시작하기: 샘플 등 이 장의 다른 안내서를 이미 마쳤다면 이 안내서의 사전 조건을 충족했을 것이므로 이 안내서 시작으 단원으로 건너뛸 수 있습니다. 그렇지 않다면 사전 조건을 완료한 다음 이 안내서로 돌아오십시오.

주제

- 1단계: 쿡북 생성
- 2단계: 스택 및 구성 요소 생성
- 3단계: 레시피 실행 및 테스트
- 4단계: 패키지를 설치하도록 쿡북 업데이트
- 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행
- 6단계: 사용자를 추가하도록 쿡북 업데이트
- 7단계: 디렉터리를 생성하도록 쿡북 업데이트
- 8단계: 파일을 생성하고 복사하도록 쿡북 업데이트
- 9단계: 명령을 실행하도록 쿡북 업데이트
- 10단계: 스크립트를 실행하도록 쿡북 업데이트
- 11단계: 서비스를 관리하도록 쿡북 업데이트
- 12단계: 사용자 지정 JSON을 사용하도록 쿡북 업데이트
- 13단계: 데이터 백을 사용하도록 쿡북 업데이트
- 14단계: 반복을 사용하도록 쿡북 업데이트
- 15단계: 조건부 논리를 사용하도록 쿡북 업데이트
- 16단계: 커뮤니티 쿡북을 사용하도록 쿡북 업데이트
- 17단계: (선택 사항) 정리
- 다음 단계

1단계: 쿡북 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

쿡북 만들기로 시작합니다. 이 쿡북은 시작하는 데 큰 도움이 되는 것은 아니지만, 이 연습의 나머지 부분에 대한 기초 역할을 합니다.

> **Note**
>
> 이 단계에서는 수동으로 쿡북을 만드는 방법을 보여 줍니다. Chef 개발 키트(Chef DK)를 사용하면 로컬 워크스테이션에서 chef generate cookbook 명령을 실행하여 더 짧은 시간 안에 쿡북을 만들 수 있습니다. 하지만 이 명령은 이 안내서에서는 필요 없는 몇몇 폴더와 파일을 생성합니다.

쿡북을 생성하려면

1. 로컬 워크스테이션에 디렉터리 `opsworks_cookbook_demo`를 만듭니다. 다른 이름을 사용할 수 있는데 이렇게 하려면 이 연습 전체에서 `opsworks_cookbook_demo`를 해당 이름으로 바꿔야 합니다.
2. `opsworks_cookbook_demo` 디렉터리에서 텍스트 편집기를 사용하여 `metadata.rb`라는 파일을 생성합니다. 다음 코드를 추가하여 쿡북의 이름을 지정합니다. `metadata.rb`에 대한 자세한 내용은 Chef 웹 사이트의 metadata.rb를 참조하세요.

   ```
   name "opsworks_cookbook_demo"
   ```

3. `opsworks_cookbook_demo` 디렉터리에서 하위 디렉터리 `recipes`를 만듭니다. 이 하위 디렉터리에는 이 연습의 쿡북을 위해 생성하는 레시피가 모두 저장됩니다.
4. `recipes` 디렉터리에서 `default.rb`라는 파일 만듭니다. 이 파일에는 파일 이름과 같은 레시피가 있는데 파일 확장명 `default`는 생략되어 있습니다. 다음 코드 한 행을 `default.rb` 파일에 추가합니다. 이 코드는 레시피 실행 시 로그에 간단한 메시지를 표시하는 한 행짜리 레시피입니다.

```
Chef::Log.info("********** Hello, World! **********")
```

5. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 그 내용이 포함된 opsworks_cookbook_demo.tar.gz 파일을 만듭니다. 예시:

   ```
   tar -czvf opsworks_cookbook_demo.tar.gz opsworks_cookbook_demo/
   ```

   다른 파일 이름을 사용할 수 있는데 이렇게 하면 이 연습 전체에서 opsworks_cookbook_demo.tar.gz를 해당 이름으로 바꿔야 합니다.

   > **Note**
   >
   > Windows에서 tar 파일을 만드는 경우 최상위 디렉터리는 쿡북의 상위 디렉터리여야 합니다. 이 연습은 Linux에서는 tar 패키지에서 제공하는 tar 명령을 사용하고 Windows에서는 Git Bash에서 제공하는 tar 명령을 사용하여 테스트되었습니다. 다른 명령 또는 프로그램을 사용하여 압축 TAR(.tar.gz) 파일을 만들면 예상대로 진행되지 않을 수 있습니다.

6. S3 버킷을 생성하거나 기존 버킷을 사용합니다. 자세한 정보는 버킷 만들기를 참조하세요.
7. opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다. 자세한 내용은 버킷에 객체 추가를 참조하세요.

이제 이 안내서에서 사용할 쿡북이 만들어졌습니다.

다음 단계에서는 나중에 AWS OpsWorks 쿡북을 업로드하고 쿡북의 레시피를 실행하는 데 사용할 Stacks 스택을 생성합니다.

2단계: 스택 및 구성 요소 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계층과 인스턴스를 포함하는 AWS OpsWorks Stacks 스택과 해당 구성 요소를 생성합니다. 이후 단계에서는 쿡북을 인스턴스에 업로드한 다음 해당 인스턴스에서 쿡북의 레시피를 실행합니다.

스택을 생성하는 방법

1. https://console.aws.amazon.com/opsworks AWS OpsWorks Stacks 콘솔에 로그인합니다.
2. 해당하는 경우 다음 중 하나를 수행하세요.
   - AWS OpsWorks 스택 시작 페이지가 표시되면 첫 번째 스택 추가 또는 첫 번째 AWS OpsWorks 스택 추가를 선택합니다(두 선택 모두 동일한 작업을 수행함). [스택 추가] 페이지가 표시됩니다.
   - [OpsWorks 대시보드] 페이지가 표시되면 [스택 추가]를 선택합니다. [스택 추가] 페이지가 표시됩니다.
3. [Chef 12 스택]을 선택합니다.
4. 스택 이름 상자에 스택 이름(예: **MyCookbooksDemoStack**)을 입력하세요. 다른 이름을 입력할 수 있는데 이렇게 하면 이 연습 전체에서 `MyCookbooksDemoStack`를 해당 이름으로 바꿔야 합니다.
5. 리전에서 미국 서부(오레곤)를 선택합니다.
6. VPC의 경우 다음 중 하나를 수행합니다.
   - VPC를 사용할 수 있는 경우 선택합니다. 자세한 내용은 VPC에서 스택 실행 단원을 참조하십시오.
   - 그렇지 않은 경우 [VPC 없음]을 선택합니다.
7. [사용자 지정 Chef 쿡북 사용]에 대해 [예]를 선택합니다.
8. [리포지토리 유형]으로 [S3 아카이브]를 선택합니다.

   > **Note**
   >
   > 시작하기: Linux 연습에서는 [Http 아카이브]를 선택합니다. 여기에서는 대신 [S3 아카이브]를 선택합니다.

9. [리포지토리 URL]로 S3의 `opsworks_cookbook_demo.tar.gz` 파일에 대한 경로를 입력합니다. 이 경로를 확인하려면 S3 콘솔에서 [opsworks_cookbook_demo.tar.gz] 파일을 선택합니다. [속성] 창에서 [링크] 필드의 값을 복사합니다. 이 값은 `https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz`와 유사해야 합니다.

10. S3 버킷이 기본값인 프라이빗 버킷인 경우 액세스 키 ID 및 보안 액세스 키에 대해 이 연습에서 사용 중인 IAM 사용자의 액세스 키 ID 및 보안 액세스 키를 입력합니다. 자세한 정보는 객체 권한 편집 및 다른 사용자와 객체 공유를 참조하세요.
11. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.
    - [기본 가용 영역]([us-west-2a])
    - 기본 운영 체제(Linux 및 Amazon Linux 2016.09)
    - [기본 SSH 키]([기본 SSH 키 사용 금지])
    - [스택 색상](짙은 파란색)
12. 고급을 선택합니다.
13. IAM 역할에 대해 다음 중 하나를 수행합니다.
    - [aws-opsworks-service-role]을 사용할 수 있는 경우 선택합니다.
    - aws-opsworks-service-role을 사용할 수 없는 경우 새 IAM 역할을 선택합니다.
14. 기본 IAM 인스턴스 프로파일의 경우 다음 중 하나를 수행합니다.
    - [aws-opsworks-ec2-role]을 사용할 수 있는 경우 선택합니다.
    - aws-opsworks-ec2-role을 사용할 수 없는 경우 새 IAM 인스턴스 프로파일을 선택합니다.
15. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.
    - [기본 루트 디바이스 유형]([EBS 지원])
    - [호스트 이름 테마]([계층 종속적])
    - [OpsWorks Agent 버전](최신 버전)
    - [사용자 지정 Chef JSON](비워 둠)
    - [보안], [OpsWorks 보안 그룹 사용]([예])
16. 스택 추가를 선택합니다. AWS OpsWorks 스택은 스택을 생성하고 MyCookbooksDemoStack 페이지를 표시합니다.

계층을 생성하려면

1. 서비스 탐색 창에서 [계층]을 선택합니다. [계층] 페이지가 표시됩니다.
2. [계층 추가]를 선택합니다.
3. OpsWorks 탭에서 이름으로 **`MyCookbooksDemoLayer`**를 입력합니다. 다른 이름을 입력할 수 있는데 이렇게 하면 이 연습 전체에서 `MyCookbooksDemoLayer`를 해당 이름으로 바꿔야 합니다.

4. 짧은 이름에 **cookbooks-demo**를 입력합니다. 다른 이름을 입력할 수 있는데 이렇게 하면 이 연습 전체에서 cookbooks-demo를 해당 이름으로 바꿔야 합니다.
5. 계층 추가를 선택합니다. AWS OpsWorks 스택은 계층을 추가하고 계층 페이지를 표시합니다.

인스턴스를 생성 및 시작하려면

1. 서비스 탐색 창에서 [인스턴스]를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. 인스턴스 추가를 선택합니다.
3. [새로 만들기] 탭에서 [고급]를 선택합니다.
4. 다음 옵션의 경우 기본값을 그대로 둘 수 있습니다.
   - [호스트 이름]([cookbooks-demo1])
   - [크기]([c3.large])
   - [서브넷](*IP ##* us-west-2a)
   - [조정 유형]([24/7])
   - [SSH 키]([기본 SSH 키 사용 금지])
   - 운영 체제(Amazon Linux 2016.09)
   - [OpsWorks Agent 버전]([스택에서 상속])
   - [테넌시]([기본값 - VPC 설정 사용])
   - [루트 디바이스 유형]([EBS 지원])
   - [볼륨 유형]([범용(SSD)])
   - [볼륨 크기]([8])
5. [인스턴스 추가]를 선택합니다.
6. [MyCookbooksDemoLayer]의 [cookbooks-demo1]에 해당하는 [작업]에서 [시작]을 선택합니다. [상태]가 [온라인]으로 변경될 때까지 진행하지 마십시오. 이 프로세스에는 몇 분이 걸릴 수 있으니 조금만 기다려 주십시오.

이제 스택, 계층 및 S3 버킷에서 쿡북이 자동으로 복사된 인스턴스가 생겼습니다. 다음 단계에서는 이 인스턴스의 쿡북 안에서 기본 레시피를 실행하고 테스트합니다.

3단계: 레시피 실행 및 테스트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Stacks가 인스턴스에 복사한 AWS OpsWorks 쿡북 내에서 `default` 레시피를 실행하고 테스트합니다. 앞서 살펴본 것처럼 이것은 실행될 때 로그에 간단한 메시지를 표시하는 한 줄짜리 레시피입니다.

레시피를 실행하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [MyCookbooksDemoStack] 페이지가 표시됩니다.
2. [명령 실행]을 선택합니다. [명령 실행] 페이지가 표시됩니다.
3. [명령]에 대해 [레시피 실행]을 선택합니다.
4. 실행할 레시피에 **`opsworks_cookbook_demo::default`**를 입력합니다.

   `opsworks_cookbook_demo`는 `metadata.rb` 파일에 정의된 대로 쿡북의 이름입니다. **`default`**는 실행할 레시피의 이름, 즉 쿡북의 `recipes` 하위 디렉터리에 있는 `default.rb` 파일의 이름으로, 파일 확장명은 빠져 있습니다.
5. 다음 기본 설정은 그래도 두십시오.

   - [설명](비워 둠)
   - [고급], [사용자 지정 Chef JSON](비워 둠)
   - [인스턴스]([모두 선택] 선택, [MyCookbooksDemoLayer] 선택, [cookbooks-demo1] 선택)
6. [레시피 실행]을 선택합니다. [execute_recipes 명령 실행 중] 페이지가 표시됩니다. [상태]가 [성공]으로 변경될 때까지 진행하지 마십시오. 이 프로세스에는 몇 분이 걸릴 수 있으니 조금 기다려 주십시오.

레시피의 결과를 확인하려면

1. [execute_recipes 명령 실행 중] 페이지가 표시된 상태에서 [cookbooks-demo1]의 [로그]에 대해 [표시]를 선택합니다. [execute_recipes] 로그 페이지가 표시됩니다.
2. 로그를 아래로 스크롤하여 다음과 유사한 항목을 찾습니다.

```
[2015-11-13T19:14:39+00:00] INFO: ********** Hello, World! **********
```

첫 레시피를 성공적으로 실행했습니다! 다음 단계에서는 인스턴스에 패키지를 설치하는 레시피를 추가하여 쿡북을 업데이트합니다.

4단계: 패키지를 설치하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인기 텍스트 편집기인 GNU Emacs가 포함된 패키지를 인스턴스에 설치하는 레시피를 추가하여 쿡북을 업데이트합니다.

인스턴스에 쉽게 로그인하고 패키지를 한 번 설치할 수 있지만 레시피를 작성하면 AWS OpsWorks Stacks에서 레시피를 한 번 실행하여 스택의 여러 인스턴스에 여러 패키지를 동시에 설치할 수 있습니다.

패키지를 설치하도록 쿡북을 업데이트하려면

1. 로컬 워크스테이션으로 돌아와 다음 코드를 사용하여 `opsworks_cookbook_demo` 디렉터리의 `recipes` 하위 디렉터리에 `install_package.rb` 파일을 생성합니다.

   ```
   package "Install Emacs" do
     package_name "emacs"
   end
   ```

   이 레시피는 인스턴스에 `emacs` 패키지를 설치합니다. 자세한 정보는 패키지를 참조하세요.

   > **Note**
   >
   > 레시피에는 원하는 파일 이름을 지정할 수 있습니다. AWS OpsWorks Stacks가 레시피를 실행하도록 하려면 항상 올바른 레시피 이름을 지정해야 합니다.

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.

이 새 레시피는 인스턴스에서 쿡북을 업데이트한 다음 업데이트된 쿡북에서 새 레시피를 실행할 때 실행됩니다. 다음 단계에서는 그 방법을 설명합니다.

다음 단계를 마친 후에는 인스턴스에 로그인한 다음 명령 프롬프트에서 emacs를 입력하여 GNU Emacs를 시작할 수 있습니다. (자세한 내용은 Linux 인스턴스에 연결을 참조하세요.) GNU Emacs를 끝내려면 Ctrl+X 버튼을 누른 다음 Ctrl+C 버튼을 누릅니다.

> **Important**
>
> 인스턴스에 로그인하려면 먼저 AWS OpsWorks Stacks에 퍼블릭 SSH 키(ssh-keygen 또는 PuTTYgen과 같은 도구를 사용하여 생성할 수 있음)에 대한 정보를 제공한 다음 MyCookbooksDemoStack 스택에 대한 권한을 설정하여 사용자가 인스턴스에 로그인할 수 있도록 해야 합니다. 지침은 사용자의 퍼블릭 SSH 키 등록 및 SSH를 사용하여 로그인 섹션을 참조하세요.

5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에서 쿡북을 업데이트한 다음 인스턴스의 업데이트된 쿡북에서 새 레시피를 실행합니다. 이 안내서의 나머지 부분에서도 새 레시피를 추가하여 쿡북을 업데이트할 때마다 이 단계를 반복합니다.

인스턴스에서 쿡북을 업데이트하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [MyCookbooksDemoStack] 페이지가 표시됩니다.
2. [명령 실행]을 선택합니다. [명령 실행] 페이지가 표시됩니다.

3. [명령]에 대해서는 [사용자 지정 쿡북 업데이트]를 선택합니다.
4. 다음 기본 설정은 그래도 두십시오.
   - [설명](비워 둠)
   - [고급], [사용자 지정 Chef JSON](비워 둠)
   - [고급], [인스턴스]([모두 선택] 선택, [MyCookbooksDemoLayer] 선택, [cookbooks-demo1] 선택)
5. [사용자 지정 쿡북 업데이트]를 선택합니다. [Running command update_custom_cookbooks] 페이지가 표시됩니다. [상태]가 [성공]으로 변경될 때까지 진행하지 마십시오. 이 프로세스에는 몇 분이 걸릴 수 있으니 조금만 기다려 주십시오.

레시피를 실행하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [MyCookbooksDemoStack] 페이지가 표시됩니다.
2. [명령 실행]을 선택합니다. [명령 실행] 페이지가 표시됩니다.
3. [명령]에 대해 [레시피 실행]을 선택합니다.
4. [실행할 레시피]에 실행할 레시피 이름을 입력합니다. 처음으로 입력할 때 레시피 이름을 **`opsworks_cookbook_demo::install_package`**로 지정합니다.

   > **Note**
   >
   > 이후에 이 절차를 반복할 때에는 쿡북의 이름(**`opsworks_cookbook_demo`**)을 입력하고 콜론을 두 번 입력한 다음(**`::`**) 레시피 이름을 입력합니다. 이때, 레시피 파일 이름에서 `.rb` 파일 확장명은 제외합니다.

5. 다음 기본 설정은 그래도 두십시오.
   - [설명](비워 둠)
   - [고급], [사용자 지정 Chef JSON](비워 둠)
   - [인스턴스]([모두 선택] 선택, [MyCookbooksDemoLayer] 선택, [cookbooks-demo1] 선택)
6. [레시피 실행]을 선택합니다. [execute_recipes 명령 실행 중] 페이지가 표시됩니다. [상태]가 [성공]으로 변경될 때까지 진행하지 마십시오. 이 프로세스에는 몇 분이 걸릴 수 있으니 조금 기다려 주십시오.

> **Note**
>
> 수동으로 레시피를 실행할 필요는 없습니다. 설정 및 구성 이벤트와 같은 계층의 수명 주기 이벤트에 레시피를 할당할 수 있으며, AWS OpsWorks Stacks는 이벤트가 발생할 때 해당 레시피를 자동으로 실행합니다. 자세한 내용은 AWS OpsWorks Stacks 수명 주기 이벤트 단원을 참조하십시오.

다음 단계에서는 사용자를 인스턴스에 추가하도록 쿡북을 업데이트합니다.

6단계: 사용자를 추가하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

로컬 사용자를 인스턴스에 추가하고 사용자의 홈 디렉터리와 셸을 설정하는 레시피를 추가하여 쿡북을 업데이트합니다. 이것은 Linux adduser 또는 useradd 명령이나 Windows net user 명령을 실행하는 것과 비슷합니다. 예컨대 인스턴스의 파일 및 디렉터리에 대한 액세스를 제어하려는 경우, 인스턴스에 로컬 사용자를 추가합니다.

쿡북을 사용하지 않고 사용자를 관리할 수도 있습니다. 자세한 내용은 사용자 관리 단원을 참조하십시오.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션으로 돌아와 다음 코드를 사용하여 `opsworks_cookbook_demo` 디렉터리의 `recipes` 하위 디렉터리에 `add_user.rb` 파일을 생성합니다(자세한 정보는 사용자 참조).

```
user "Add a user" do
  home "/home/jdoe"
  shell "/bin/bash"
  username "jdoe"
end
```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::add_user**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 새 사용자가 추가되었는지 확인합니다.

```
grep jdoe /etc/passwd
```

사용자 이름, ID 번호, 그룹 ID 번호, 홈 디렉터리, 셸 등 세부 정보를 비롯하여 사용자에 대한 다음과 유사한 정보가 표시됩니다.

```
jdoe:x:501:502::/home/jdoe:/bin/bash
```

다음 단계에서는 인스턴스에서 디렉터리를 생성하도록 쿡북을 업데이트합니다.

7단계: 디렉터리를 생성하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에 디렉터리를 추가하는 레시피를 추가하여 쿡북을 업데이트합니다. 이것은 Linux mkdir 명령이나 Windows md 또는 mkdir 명령을 실행하는 것과 비슷합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 opsworks_cookbook_demo 디렉터리의 recipes 하위 디렉터리에 create_directory.rb 파일을 생성합니다. 자세한 정보는 디렉터리를 참조하세요.

   ```
   directory "Create a directory" do
     group "root"
     mode "0755"
     owner "ec2-user"
     path "/tmp/create-directory-demo"
   end
   ```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::create_directory**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 새 디렉터리가 추가되었는지 확인합니다.

   ```
   ls -la /tmp/create-directory-demo
   ```

   권한, 소유자 이름 및 그룹 이름 등의 정보를 비롯하여 새로 추가된 디렉터리에 대한 정보가 표시됩니다.

   ```
   drwxr-xr-x 2 ec2-user root 4096 Nov 18 00:35 .
   drwxrwxrwt 6 root     root 4096 Nov 24 18:17 ..
   ```

다음 단계에서는 인스턴스에서 파일을 생성하도록 쿡북을 업데이트합니다.

8단계: 파일을 생성하고 복사하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에 2개의 파일을 추가하는 레시피를 추가하여 쿡북을 업데이트합니다. 레시피의 첫 번째 리소스는 순전히 레시피 코드를 사용하여 파일을 생성합니다. 이것은 Linux cat, echo 또는 touch 명령이나 Windows echo 또는 fsutil 명령을 실행하는 것과 비슷합니다. 이 기법은 소수의 작거나 단순한 파일에 유용합니다. 레시피의 두 번째 리소스는 쿡북의 파일을 인스턴스의 다른 디렉터리에 복사합니다. 이것은 Linux cp 명령이나 Windows copy 명령을 실행하는 것과 비슷합니다. 이 기법은 다수의 크거나 복잡한 파일에 유용합니다.

이 단계를 시작하기 전에 7단계: 디렉터리를 생성하도록 쿡북 업데이트를 완료하여 파일의 상위 디렉터리가 이미 존재하는지 확인하세요.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션의 `opsworks_cookbook_demo` 디렉터리에서 하위 디렉터리 `files`를 만듭니다.
2. `files` 하위 디렉터리에 **`Hello, World!`**라는 텍스트가 포함된 `hello.txt` 파일을 만듭니다.
3. 다음 코드를 사용하여 `recipes` 디렉터리의 `opsworks_cookbook_demo` 하위 디렉터리에 `create_files.rb` 파일을 생성합니다. 자세한 정보는 파일 및 cookbook_file 단원을 참조하세요.

```
file "Create a file" do
  content "<html>This is a placeholder for the home page.</html>"
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/index.html"
end

cookbook_file "Copy a file" do
  group "root"
```

```
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/hello.txt"
  source "hello.txt"
end
```

`file` 리소스는 지정된 경로에 파일을 생성합니다. `cookbook_file` 리소스는 쿡북에서 방금 만든 `files` 디렉터리에서(Chef에서는 파일을 복사해 올 표준 이름 하위 디렉터리 `files`를 찾음) 인스턴스의 다른 디렉터리로 파일을 복사합니다.

4. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 `opsworks_cookbook_demo` 디렉터리와 업데이트된 내용 포함된 `opsworks_cookbook_demo.tar.gz` 파일의 새 버전을 만듭니다.
5. 업데이트된 `opsworks_cookbook_demo.tar.gz` 파일을 S3 버킷에 업로드합니다.
6. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::create_files**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 한 번에 하나씩 새 파일이 추가되었는지 확인합니다.

```
sudo cat /tmp/create-directory-demo/index.html

sudo cat /tmp/create-directory-demo/hello.txt
```

파일의 내용이 다음과 같이 표시됩니다.

```
<html>This is a placeholder for the home page.</html>

Hello, World!
```

다음 단계에서는 인스턴스에서 명령을 실행하도록 쿡북을 업데이트합니다.

9단계: 명령을 실행하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에서 SSH 키를 생성하는 명령을 실행하는 레시피를 추가하여 쿡북을 업데이트합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 `opsworks_cookbook_demo` 디렉터리의 `recipes` 하위 디렉터리에 `run_command.rb` 파일을 생성합니다. 자세한 정보는 실행 단원을 참조하세요.

   ```
   execute "Create an SSH key" do
     command "ssh-keygen -f /tmp/my-key -N fLyC3jbY"
   end
   ```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 `opsworks_cookbook_demo` 디렉터리와 업데이트된 내용 포함된 `opsworks_cookbook_demo.tar.gz` 파일의 새 버전을 만듭니다.
3. 업데이트된 `opsworks_cookbook_demo.tar.gz` 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **`opsworks_cookbook_demo::run_command`**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 한 번에 하나씩 SSH 키가 생성되었는지 확인합니다.

   ```
   sudo cat /tmp/my-key

   sudo cat /tmp/my-key.pub
   ```

SSH 프라이빗 및 퍼블릭 키의 내용이 표시됩니다.

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,DEF7A09C...541583FA
A5p9dCuo...wp0YYH1c
-----END RSA PRIVATE KEY-----

ssh-rsa AAAAB3N...KaNogZkT root@cookbooks-demo1
```

다음 단계에서는 인스턴스에서 스크립트를 실행하도록 쿡북을 업데이트합니다.

10단계: 스크립트를 실행하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에서 스크립트를 실행하는 레시피를 추가하여 쿡북을 업데이트합니다. 이 레시피는 디렉터리를 생성한 다음 해당 디렉터리에서 파일을 생성합니다. 여러 명령이 포함된 스크립트를 실행하도록 레시피를 작성하는 것이 이러한 명령을 한 번에 하나씩 실행하는 것보다 쉽습니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 `opsworks_cookbook_demo` 디렉터리의 `recipes` 하위 디렉터리에 `run_script.rb` 파일을 생성합니다. 자세한 정보는 스크립트를 참조하세요.

```
script "Run a script" do
  interpreter "bash"
  code <<-EOH
    mkdir -m 777 /tmp/run-script-demo
    touch /tmp/run-script-demo/helloworld.txt
    echo "Hello, World!" > /tmp/run-script-demo/helloworld.txt
  EOH
```

```
end
```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::run_script**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 새 파일이 추가되었는지 확인합니다.

```
sudo cat /tmp/run-script-demo/helloworld.txt
```

파일의 내용이 다음과 같이 표시됩니다.

```
Hello, World!
```

다음 단계에서는 인스턴스에서 서비스를 관리하도록 쿡북을 업데이트합니다.

11단계: 서비스를 관리하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에서 서비스를 관리하는 레시피를 추가하여 쿡북을 업데이트합니다. 이것은 Linux service 명령이나 Windows net stop, net start 및 유사한 명령을 실행하는 것과 비슷합니다. 이 레시피는 인스턴스에서 crond 서비스를 중지합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 opsworks_cookbook_demo 디렉터리의 recipes 하위 디렉터리에 manage_service.rb 파일을 생성합니다. 자세한 정보는 서비스를 참조하세요.

   ```
   service "Manage a service" do
     action :stop
     service_name "crond"
   end
   ```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::manage_service**를 입력합니다.

레시피를 테스트하려면

1. 아직 로그인하지 않았다면 인스턴스에 로그인합니다.
2. 명령 프롬프트에서 다음 명령을 실행하여 crond 서비스가 중지되었는지 확인합니다.

   ```
   service crond status
   ```

   다음과 같이 표시됩니다.

   ```
   crond is stopped
   ```

3. crond 서비스를 다시 시작하려면 다음 명령을 실행합니다.

   ```
   sudo service crond start
   ```

   다음과 같이 표시됩니다.

   ```
   Starting crond:  [  OK  ]
   ```

4. crond 서비스가 시작되었는지 확인하려면 다음 명령을 다시 실행합니다.

```
service crond status
```

다음과 유사한 정보가 표시됩니다.

```
crond (pid  3917) is running...
```

다음 단계에서는 인스턴스에 사용자 지정 JSON으로 저장된 정보를 참조하도록 쿡북을 업데이트합니다.

12단계: 사용자 지정 JSON을 사용하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에 저장된 JSON 단원을 참조하는 레시피를 추가하여 쿡북을 업데이트합니다.

스택을 생성, 업데이트 또는 복제할 때마다 혹은 배포 또는 스택 명령을 실행할 때 사용자 지정 JSON 형식으로 정보를 지정할 수 있습니다. 이것은 예컨대 데이터베이스에서 데이터를 가져오는 대신 데이터 중 작고 변하지 않는 부분을 인스턴스에서 레시피가 사용할 수 있도록 만드는 데 유용합니다. 자세한 내용은 사용자 지정 JSON 사용 단원을 참조하십시오.

이 안내서에서는 사용자 지정 JSON을 사용하여 고객 인보이스에 대한 몇 가지 가상 정보를 제공합니다. 사용자 지정 JSON은 이 단계 뒷부분에서 설명합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 레시피 코드를 사용하여 `recipes` 디렉터리의 `opsworks_cookbook_demo` 하위 디렉터리에 `custom_json.rb` 파일을 생성합니다.

```
Chef::Log.info("********** For customer '#{node['customer-id']}' invoice
 '#{node['invoice-number']}' **********")
Chef::Log.info("********** Invoice line number 1 is a '#{node['line-items']
['line-1']}' **********")
```

```
Chef::Log.info("********** Invoice line number 2 is a '#{node['line-items']
['line-2']}' **********")
Chef::Log.info("********** Invoice line number 3 is a '#{node['line-items']
['line-3']}' **********")
```

이 레시피는 로그에 사용자 지정 JSON의 값에 대한 메시지를 표시합니다.

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::custom_json**를 입력합니다. [고급]의 [사용자 지정 Chef JSON]에는 다음 사용자 지정 JSON을 입력합니다.

```
{
  "customer-id": "0123",
  "invoice-number": "9876",
  "line-items": {
    "line-1": "tractor",
    "line-2": "passenger car",
    "line-3": "trailer"
  }
}
```

레시피를 테스트하려면

1. 이전 절차의 [execute_recipes 명령 실행 중] 페이지가 표시된 상태에서 [cookbooks-demo1]의 [로그]에 대해 [표시]를 선택합니다. [execute_recipes] 로그 페이지가 표시됩니다.
2. 로그를 아래로 스크롤하면서 다음과 유사한 항목을 찾습니다.

```
[2015-11-14T14:18:30+00:00] INFO: ********** For customer '0123' invoice '9876'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 1 is a 'tractor'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 2 is a 'passenger
 car' **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 3 is a 'trailer'
 **********
```

이러한 항목에는 [고급]의 [사용자 지정 Chef JSON] 상자에 입력한 사용자 지정 JSON의 정보가 표시됩니다.

다음 단계에서는 Stacks가 각 인스턴스에 저장하는 스택 설정 모음인 데이터 백에서 정보를 가져오도록 AWS OpsWorks 쿡북을 업데이트합니다.

13단계: 데이터 백을 사용하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Stacks가 인스턴스에 데이터 백 세트로 저장하는 스택 설정을 참조하는 레시피를 추가하여 AWS OpsWorks 쿡북을 업데이트합니다. 이 레시피는 인스턴스에 저장된 특정 스택 설정에 대한 메시지를 로그에 표시합니다. 자세한 내용은 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하십시오.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 `recipes` 디렉터리의 `opsworks_cookbook_demo` 하위 디렉터리에 `data_bags.rb` 파일을 생성합니다.

```
instance = search("aws_opsworks_instance").first
layer = search("aws_opsworks_layer").first
stack = search("aws_opsworks_stack").first

Chef::Log.info("********** This instance's instance ID is
 '#{instance['instance_id']}' **********")
Chef::Log.info("********** This instance's public IP address is
 '#{instance['public_ip']}' **********")
Chef::Log.info("********** This instance belongs to the layer '#{layer['name']}'
 **********")
Chef::Log.info("********** This instance belongs to the stack '#{stack['name']}'
 **********")
Chef::Log.info("********** This stack gets its cookbooks from
 '#{stack['custom_cookbooks_source']['url']}' **********")
```

이 레시피는 인스턴스에 저장된 특정 스택 설정에 대한 메시지를 로그에 표시합니다.

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 `opsworks_cookbook_demo` 디렉터리와 업데이트된 내용 포함된 `opsworks_cookbook_demo.tar.gz` 파일의 새 버전을 만듭니다.
3. 업데이트된 `opsworks_cookbook_demo.tar.gz` 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **`opsworks_cookbook_demo::data_bags`**를 입력합니다.

레시피를 테스트하려면

1. 이전 절차에서 표시된 [execute_recipes 명령 실행 중] 페이지에서 [cookbooks-demo1]의 [로그]에 대해 [표시]를 선택합니다. [execute_recipes] 로그 페이지가 표시됩니다.
2. 로그를 아래로 스크롤하면서 다음과 유사한 항목을 찾습니다.

```
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's instance ID is
 'f80fa119-81ab-4c3c-883d-6028e52c89EX' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's public IP address is
 '192.0.2.0' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the layer
 'MyCookbooksDemoLayer' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the stack
 'MyCookbooksDemoStack' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This stack gets its cookbooks from
 'https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz'
 **********
```

이 레시피는 인스턴스에 저장된 특정 스택 설정에 대한 메시지를 표시합니다.

다음 단계에서는 레시피 코드를 여러 번 실행하도록 쿡북을 업데이트합니다.

14단계: 반복을 사용하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피 코드를 여러 번 반복하는 기법인 반복을 사용하는 레시피를 추가하여 쿡북을 업데이트합니다. 이 레시피는 여러 콘텐츠를 포함하는 데이터 백 항목에 대한 메시지를 로그에 표시합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 다음 코드를 사용하여 `recipes` 디렉터리의 `opsworks_cookbook_demo` 하위 디렉터리에 `iteration_demo.rb` 파일을 생성합니다.

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")

stack["custom_cookbooks_source"].each do |content|
  Chef::Log.info("********** '#{content}' **********")
end
```

> **Note**
>
> 이전 레시피 코드를 작성하는 것이 반복을 사용하지 않는 다음 레시피 코드를 작성하는 것보다 더 짧고, 보다 유연하고, 오류 발생 가능성이 더 낮습니다.
>
> ```
> stack = search("aws_opsworks_stack").first
> Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")
>
> Chef::Log::info("********** '[\"type\", \"#{stack['custom_cookbooks_source']
> ['type']}\"]' **********")
> Chef::Log::info("********** '[\"url\", \"#{stack['custom_cookbooks_source']
> ['url']}\"]' **********")
> Chef::Log::info("********** '[\"username\",
>  \"#{stack['custom_cookbooks_source']['username']}\"]' **********")
> Chef::Log::info("********** '[\"password\",
>  \"#{stack['custom_cookbooks_source']['password']}\"]' **********")
> Chef::Log::info("********** '[\"ssh_key\",
>  \"#{stack['custom_cookbooks_source']['ssh_key']}\"]' **********")
> Chef::Log::info("********** '[\"revision\",
>  \"#{stack['custom_cookbooks_source']['revision']}\"]' **********")
> ```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::iteration_demo**를 입력합니다.

레시피를 테스트하려면

1. 이전 절차의 [execute_recipes 명령 실행 중] 페이지가 표시된 상태에서 [cookbooks-demo1]의 [로그]에 대해 [표시]를 선택합니다. [execute_recipes] 로그 페이지가 표시됩니다.
2. 로그를 아래로 스크롤하면서 다음과 유사한 항목을 찾습니다.

```
[2015-11-16T19:56:56+00:00] INFO: ********** Content of 'custom_cookbooks_source'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["type", "s3"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["url", "https://s3.amazonaws.com/
amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["username", "secret-key-value"]'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["password", "secret-access-key-
value"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["ssh_key", nil]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["revision", nil]' **********
```

   이 레시피는 여러 콘텐츠를 포함하는 데이터 백 항목에 대한 메시지를 로그에 표시합니다. 데이터 백 항목은 aws_opsworks_stack 데이터 백에 있습니다. 데이터 백 항목에는custom_cookbooks_source라는 콘텐츠가 있습니다. 이 컨텐츠 내에는 type, url, username, password, ssh_key 및 revision라는 6개의 컨텐츠가 있으며 이러한 컨텐츠의 값도 표시됩니다.

다음 단계에서는 특정 조건이 충족되는 경우에만 레시피 코드를 실행하도록 쿡북을 업데이트합니다.

15단계: 조건부 논리를 사용하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제 특정 조건이 충족되는 경우에만 코드를 실행하는 기법인 조건부 논리를 사용하는 레시피를 추가하여 쿡북을 업데이트합니다. 자세한 정보는 if 문과 case 문 단원을 참조하세요.

이 레시피는 데이터 백 콘텐츠에 기반하여 다음 두 가지를 수행합니다. 인스턴스가 실행되는 운영 체제를 식별하는 메시지를 로그에 표시하고, 운영 체제가 Linux인 경우에만 주어진 Linux 배포에 맞는 패키지 관리자를 사용하여 패키지를 설치합니다. tree로 명명되는 이 패키지는 디렉터리 목록을 시각화하기 위한 간단한 앱입니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. 로컬 워크스테이션에서 `opsworks_cookbook_demo directory`의 `recipes` 하위 디렉터리에 다음 코드가 포함된 `conditional_logic.rb` 파일을 생성합니다.

```
instance = search("aws_opsworks_instance").first
os = instance["os"]

if os == "Red Hat Enterprise Linux 7"
  Chef::Log.info("********** Operating system is Red Hat Enterprise Linux.
 **********")
elsif os == "Ubuntu 14.04 LTS" || os == "Ubuntu 16.04 LTS" || os == "Ubuntu 18.04
 LTS"
  Chef::Log.info("********** Operating system is Ubuntu. **********")
elsif os == "Microsoft Windows Server 2012 R2 Base"
  Chef::Log.info("********** Operating system is Windows. **********")
elsif os == "Amazon Linux 2015.03" || os == "Amazon Linux 2015.09" || os == "Amazon
 Linux 2016.03" || os == "Amazon Linux 2016.09" || os == "Amazon Linux 2017.03"
 || os == "Amazon Linux 2017.09" || os == "Amazon Linux 2018.03" || os == "Amazon
 Linux 2"
  Chef::Log.info("********** Operating system is Amazon Linux. **********")
elsif os == "CentOS Linux 7"
  Chef::Log.info("********** Operating system is CentOS 7. **********")
```

```
else
  Chef::Log.info("********** Cannot determine operating system. **********")
end

case os
when "Ubuntu 14.04 LTS", "Ubuntu 16.04 LTS", "Ubuntu 18.04 LTS"
  apt_package "Install a package with apt-get" do
    package_name "tree"
  end
when "Amazon Linux 2015.03", "Amazon Linux 2015.09", "Amazon Linux 2016.03",
 "Amazon Linux 2016.09", "Amazon Linux 2017.03", "Amazon Linux 2017.09", "Amazon
 Linux 2018.03", "Amazon Linux 2", "Red Hat Enterprise Linux 7", "CentOS Linux 7"
  yum_package "Install a package with yum" do
    package_name "tree"
  end
else
  Chef::Log.info("********** Cannot determine operating system type, or operating
 system is not Linux. Package not installed. **********")
end
```

2. 터미널 또는 명령 프롬프트에서 tar 명령을 사용하여 opsworks_cookbook_demo 디렉터리와 업데이트된 내용 포함된 opsworks_cookbook_demo.tar.gz 파일의 새 버전을 만듭니다.
3. 업데이트된 opsworks_cookbook_demo.tar.gz 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::conditional_logic**를 입력합니다.

레시피를 테스트하려면

1. 이전 절차의 [execute_recipes 명령 실행 중] 페이지가 표시된 상태에서 [cookbooks-demo1]의 [로그]에 대해 [표시]를 선택합니다. [execute_recipes] 로그 페이지가 표시됩니다.
2. 로그를 아래로 스크롤하면서 다음과 유사한 항목을 찾습니다.

```
[2015-11-16T19:59:05+00:00] INFO: ********** Operating system is Amazon Linux.
 **********
```

인스턴스의 운영 체제가 Amazon Linux 2016.09이므로, (레시피 코드의 가능한 5개 항목 중) 이전 항목만 로그에 표시됩니다.

3. 운영 체제가 Linux이므로 레시피가 트리 패키지를 설치합니다. 디렉터리의 내용을 시각화해 살펴보려면 원하는 디렉터리의 명령 프롬프트에서 또는 원하는 디렉터리의 경로를 사용하여 **tree**를 입력합니다(예: `tree /var/chef/runs`).

다음 단계에서는 Chef 커뮤니티가 제공하는 외부 쿡북의 기능을 사용하도록 쿡북을 업데이트합니다.

16단계: 커뮤니티 쿡북을 사용하도록 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

마지막으로 Chef 커뮤니티가 제공하는 외부 쿡북에 있는 기능을 사용하도록 쿡북을 업데이트합니다. 이 안내서에서 사용할 외부 쿡북은 외부 Chef 쿡북에 액세스할 수 있는 인기 있는 위치인 Chef Supermarket을 통해 얻을 수 있습니다. 이 외부 쿡북은 4단계: 패키지를 설치하도록 쿡북 업데이트에서와 비슷하게 애플리케이션을 다운로드하고 설치할 수 있도록 해 주는 사용자 지정 리소스를 제공합니다. 다만 이 리소스는 패키지 외에 웹 애플리케이션과 그 밖의 애플리케이션 유형을 설치할 수 있습니다.

쿡북이 다른 쿡북에 의존하는 경우에는 다른 쿡북에 대한 종속성을 지정해야 합니다. 쿡북 종속성을 선언하고 관리하기 위해서는 Berkshelf라는 도구를 사용하는 것이 좋습니다. 로컬 워크스테이션에서 Berkshelf를 설치하는 방법에 대한 자세한 내용은 Chef 웹 사이트의 About Berkshelf를 참조하세요.

Berkshelf를 설치한 후 다음 절차에 따라 쿡북 종속성을 선언한 다음 외부 쿡북의 리소스를 호출하는 레시피를 만듭니다.

쿡북 종속성을 선언하려면

1. 로컬 워크스테이션의 `opsworks_cookbook_demo` 디렉터리에서 `metadata.rb` 파일의 끝에 다음 행을 추가합니다.

   ```
   depends "application", "5.0.0"
   ```

   이 행은 `application`, 버전 5.0.0 쿡북에 대한 종속성을 선언합니다.

2. opsworks_cookbook_demo 디렉터리의 루트에서 다음 명령을 실행합니다. 명령 끝의 마침표는 의도적인 것입니다.

```
berks init .
```

Berkshelf에서는 이후에 보다 고급 시나리오에 사용할 수 있는 여러 폴더 및 파일을 생성합니다. 이 연습에 필요한 유일한 파일은 Berksfile입니다.

3. Berksfile 파일 끝부분에서 다음 행을 추가합니다.

```
cookbook "application", "5.0.0"
```

이 행은 Berkshelf에 애플리케이션 쿡북 버전 5.0.0을 사용하려고 함을 알립니다. Berkshelf는 Chef Supermarket에서 이 버전을 다운로드합니다.

4. 터미널 또는 명령 프롬프트에서 opsworks_cookbook_demo 디렉터리의 루트에서 다음 명령을 실행합니다.

```
berks install
```

Berkshelf에서는 사용자 쿡북 및 애플리케이션 쿡북 둘 다에 대한 종속성 목록을 생성합니다. Berkshelf는 다음 절차에서 이 종속성 목록을 사용합니다.

인스턴스에서 쿡북을 업데이트하고 새 레시피를 실행하려면

1. recipes 디렉터리의 opsworks_cookbook_demo 하위 디렉터리에 다음 코드가 포함된 dependencies_demo.rb 파일을 생성합니다.

```
application "Install NetHack" do
  package "nethack.x86_64"
end
```

이 레시피는 애플리케이션 쿡북의 애플리케이션 리소스에 따라 인스턴스에 유명한 텍스트 기반 어드벤처 게임인 NetHack을 설치합니다. 물론, 인스턴스에서 패키지를 패키지 관리자에 쉽게 사용할 수 있는 경우 원하는 다른 패키지 이름으로 대체할 수 있습니다.

2. opsworks_cookbook_demo 디렉터리의 루트에서 다음 명령을 실행합니다.

```
berks package
```

Berkshelf에서는 이전 절차에서 생성한 종속성 목록을 사용하여 `cookbooks-timestamp.tar.gz` 파일을 만듭니다. 이 파일에는 `opsworks_cookbook_demo` 디렉터리와 쿡북의 종속 쿡북을 비롯한 업데이트된 콘텐츠가 들어 있습니다. 이 파일의 이름을 `opsworks_cookbook_demo.tar.gz`으로 바꿉니다.

3. 업데이트되어 이름이 바뀐 `opsworks_cookbook_demo.tar.gz` 파일을 S3 버킷에 업로드합니다.
4. 5단계: 인스턴스에서 쿡북을 업데이트하고 레시피 실행 단원의 절차에 따라 인스턴스에서 쿡북을 업데이트하고 레시피를 실행합니다. "레시피 실행" 절차에서 실행할 레시피에 **opsworks_cookbook_demo::dependencies_demo**를 입력합니다.
5. 이 레시피를 실행하면 인스턴스에 로그인할 수 있어야 합니다. 그런 다음 명령 프롬프트에 **nethack**을 입력해 실행을 시작합니다. 이 게임에 대한 자세한 정보는 NetHack 및 NetHack 가이드북 단원을 참조하세요.

다음 단계에서는 이 연습에 사용한 AWS 리소스를 정리할 수 있습니다. 이 다음 단계는 선택 사항입니다. AWS OpsWorks Stacks에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용할 수 있습니다. 그러나 이러한 AWS 리소스를 주변에 보관하면 AWS 계정에 요금이 계속 부과될 수 있습니다. 나중에 사용할 수 있도록 이러한 AWS 리소스를 계속 사용하려는 경우 이제이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

17단계: (선택 사항) 정리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 계정에 추가 요금이 발생하지 않도록이 연습에 AWS 사용된 리소스를 삭제할 수 있습니다. 이러한 AWS 리소스에는 S3 버킷, AWS OpsWorks Stacks 스택 및 스택의 구성 요소가 포함됩니다. (자세한 정보는 AWS OpsWorks 요금 단원을 참조하세요.) 그러나 Stacks에 대해 AWS OpsWorks 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용하고 싶을 수 있습니다. 이러한 AWS 리소스를 계속 사용하려면 이제이 연습을 완료했으며 로 건너뛸 수 있습니다다음 단계.

이번 연습에서 생성한 리소스에 보관된 콘텐츠는 개인 식별 정보를 포함할 수 있습니다. 이 정보를 AWS에 저장하고 싶지 않다면, 이 주제에서 설명하는 다음 단계를 따르십시오.

S3 버킷을 삭제하려면

- Amazon S3 버킷 삭제를 참조하세요.

스택의 인스턴스를 삭제하려면

1. AWS OpsWorks Stacks 콘솔의 서비스 탐색 창에서 인스턴스를 선택합니다. [인스턴스] 페이지가 표시됩니다.
2. [MyCookbooksDemoLayer]의 [cookbooks-demo1]에 해당하는 [작업]에서 [중지]를 선택합니다. 확인 메시지가 표시되면 [중지]를 선택합니다.
3. 몇 분 동안 아래와 같은 변경이 진행됩니다. 다음 변경 작업이 모두 완료될 때까지 계속 진행하지 마십시오.
   - [상태]가 [온라인]에서 [중지 중]으로 바뀌었다가 마지막으로 [중지됨]로 바뀝니다.
   - [온라인]이 [1]에서 [0]으로 바뀝니다.
   - [종료 중]이 [0]에서 [1]로 바뀌었다가 마지막에는 다시 [0]으로 바뀝니다.
   - [중지됨]가 마지막으로 [0]에서 [1]로 변경됩니다.
4. 작업에 대해 삭제를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 인스턴스를 삭제하고 인스턴스 없음을 표시합니다.

스택을 삭제하려면

1. 서비스 탐색 창에서 [스택]을 선택합니다. [MyCookbooksDemoStack] 페이지가 표시됩니다.
2. [스택 삭제]를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 스택을 삭제하고 대시보드 페이지를 표시합니다.

선택적으로 다른 AWS 서비스 및 EC2 인스턴스에 액세스하는 데 재사용하지 않으려는 경우이 연습에 사용한 IAM 사용자 및 Amazon EC2 키 페어를 삭제할 수 있습니다. 지침은 IAM 사용자 및 Amazon EC2 키 페어 및 Linux 인스턴스 삭제를 참조하세요.

이제 이 안내서를 성공적으로 완료했습니다. 자세한 내용은 다음 단계 단원을 참조하십시오.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제이 연습을 완료했으므로 다음 리소스를 검토하여 Chef AWS OpsWorks 쿡북에 대한 Stacks 지원에 대해 자세히 알아볼 수 있습니다.

- 쿡북과 레시피 - AWS OpsWorks Stacks가 현재 지원하는 Chef 및 Ruby의 버전을 설명합니다. 또한 인스턴스에서 사용자 지정 쿡북을 설치 및 업데이트하는 방법과 인스턴스에서 레시피를 실행하는 방법도 보여 줍니다.
- Learn Chef – Chef 자습서, Chef 스킬 라이브러리, 전체 Chef 설명서, Chef 교육 클래스의 링크를 제공합니다.
- All about Chef – 완전한 Chef 설명서를 제공합니다. 특정 관심 주제에는 다음이 포함됩니다.
  - About Cookbooks – 속성, 레시피, 파일, 메타데이터, 템플릿 등 주요 쿡북 구성 요소를 설명합니다.
  - About Recipes – 데이터 백 작업 방법, 다른 레시피를 포함시키는 방법, 레시피에서 Ruby 코드를 사용하는 방법 같은 레시피의 기초적 내용을 설명합니다.
  - 리소스 - `apt_package` `cookbook_file`, `directory`, `execute`, `file` 및 `package` 등 모든 내장 Chef 리소스를 사용하는 방법을 설명합니다.
  - About the Recipe DSL - `if` `case`, `data_bag`, `data_bag_item` 및 `search` 등 Chef 레시피용 코드를 쓰는 방법을 설명합니다.
- About Templates – Embedded Ruby(ERB) 템플릿을 사용하여 구성 파일과 같은 정적 텍스트 파일을 동적으로 생성하는 방법을 설명합니다.
- Learning Tracks – Chef를 사용한 인스턴스 관리 방법, 기본적 웹 앱 관리 방법, 인프라 코드 개발 및 테스트 방법, Chef 분석 방법 등을 설명합니다.
- Learning Chef – Chef를 소개합니다. 발행: O'Reilly Media
- Learning Chef code examples – O'Reilly Media가 발행한 Learning Chef 서적에 포함되는 코드 예제를 제공합니다.

AWS OpsWorks Stacks 모범 사례

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션의 전략, 기법 및 제안은 AWS OpsWorks Stacks의 최대 이점과 최적의 결과를 얻는 데 도움이 될 수 있습니다.

주제

- 모범 사례: 인스턴스용 루트 디바이스 스토리지
- 모범 사례: 애플리케이션 서버 수 최적화
- 모범 사례: 권한 관리
- 모범 사례: 앱과 쿡북의 관리 및 배포
- 로컬로 쿡북 종속성 패키징

모범 사례: 인스턴스용 루트 디바이스 스토리지

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제는 Amazon Elastic Block Store 지원을 받는 Windows 인스턴스에는 해당하지 않습니다.

Amazon Elastic Compute Cloud(Amazon EC2) Linux 인스턴스에는 다음과 같은 루트 디바이스 스토리지 옵션이 있습니다.

- 인스턴스 스토어 지원 인스턴스 — 루트 디바이스는 임시입니다.

  이 인스턴스를 중지하면 루트 디바이스의 데이터가 사라지고 복구할 수 없습니다. 자세한 내용은 Amazon EC2 인스턴스 스토어를 참조하세요.
- Amazon EBS 지원 인스턴스 - 루트 디바이스는 Amazon EBS 볼륨입니다.

  인스턴스를 중지해도 Amazon EBS 볼륨은 유지됩니다. 인스턴스를 다시 시작하면 볼륨이 다시 탑재되고 인스턴스 상태와 저장된 데이터가 복원됩니다. 볼륨을 다른 인스턴스에 탑재할 수도 있습니다. 자세한 내용은 Amazon Elastic Block Store(Amazon EBS)를 참조하세요.

어떤 루트 디바이스 스토리지 옵션을 사용할지 결정할 때는 다음을 고려하세요.

부팅 시간

Amazon EBS 인스턴스는 최초 시작 후에는 일반적으로 더 빨리 재시작됩니다.

두 가지 스토리지 유형의 최초 시작 시간은 대체로 동일합니다. 두 유형 모두 원격 리포지토리에서 패키지 설치 같은 비교적 시간이 많이 걸리는 작업을 포함한 전체 설정을 수행해야 합니다. 하지만 이후에 인스턴스를 다시 시작할 때는 다음과 같은 차이점이 있습니다.

- 인스턴스 스토어 지원 인스턴스는 패키지 설치를 포함하여 최초 시작에서와 동일한 설정 작업을 수행합니다.

  재시작에는 최초 시작과 거의 동일한 시간이 소요됩니다.
- Amazon EBS 지원 인스턴스는 루트 볼륨을 다시 탑재하고 설정 레시피를 실행합니다.

  설정 레시피는 루트 볼륨에 이미 설치된 패키지를 재설치하는 것과 같은 작업을 수행하지 않아도 되기 때문에 일반적으로 재시작이 최초 시작보다 상당히 빠릅니다.

비용

Amazon EBS 지원 인스턴스가 비용이 더 많이 듭니다.

- 인스턴스 스토어 지원 인스턴스의 경우, 인스턴스가 실행 중일 때만 요금을 지불합니다.
- Amazon EBS 지원 인스턴스의 경우, 인스턴스가 실행 중인지 여부에 상관없이 Amazon EBS 볼륨에 대해 요금을 지불합니다.

  자세한 내용은 Amazon EBS 요금을 참조하세요.

로깅

Amazon EBS 지원 인스턴스는 자동으로 로그를 유지합니다.

- 인스턴스 스토어 지원 인스턴스의 경우, 인스턴스가 중지되면 로그가 사라집니다.

  인스턴스를 중지하기 전에 로그를 가져오거나 CloudWatch Logs 같은 서비스를 사용하여 선택한 로그를 원격으로 저장해야 합니다.
- Amazon EBS 지원 인스턴스의 경우, Amazon EBS 볼륨에 로그가 저장됩니다.

  로그는 인스턴스를 다시 시작하거나 볼륨을 다른 인스턴스에 탑재하면 볼 수 있습니다.

종속성

두 가지 스토리지 유형은 종속성이 서로 다릅니다.

- 인스턴스 스토어 지원 인스턴스는 Amazon S3에 의존합니다.

  인스턴스는 시작될 때 Amazon S3에서 AMI를 다운로드해야 합니다.
- Amazon EBS 지원 인스턴스는 Amazon EBS 지원 인스턴스에 의존합니다.

  인스턴스는 시작될 때 Amazon EBS 루트 볼륨을 탑재해야 합니다.

권장 사항: 요건에 가장 적합한 스토리지 유형이 무엇인지 모르겠다면 Amazon EBS 인스턴스 유형으로 시작하는 것이 좋습니다. Amazon EBS 볼륨은 어느 정도 비용이 발생하지만 의도치 않은 데이터 손실의 위험이 적습니다.

모범 사례: 애플리케이션 서버 수 최적화

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

프로덕션 스택은 일반적으로 여러 가용 영역으로 분산된 복수의 애플리케이션 서버를 포함합니다. 하지만 수신되는 요청의 수는 일중 또는 주중 시간에 따라 현저히 변동할 수 있습니다. 단지 최대 예상 로드를 처리하기에 충분한 수의 서버를 실행하기를 원할 수 있지만, 그럴 경우 많은 시간에서는 필요한

서버 용량보다 많은 용량에 대해 비용을 지불하게 됩니다. 사이트를 효율적으로 운영하기 위해 권장되는 방법은 서버 수를 현재 요청 볼륨과 일치시키는 것입니다.

AWS OpsWorks Stacks는 서버 인스턴스 수를 관리하는 세 가지 방법을 제공합니다.

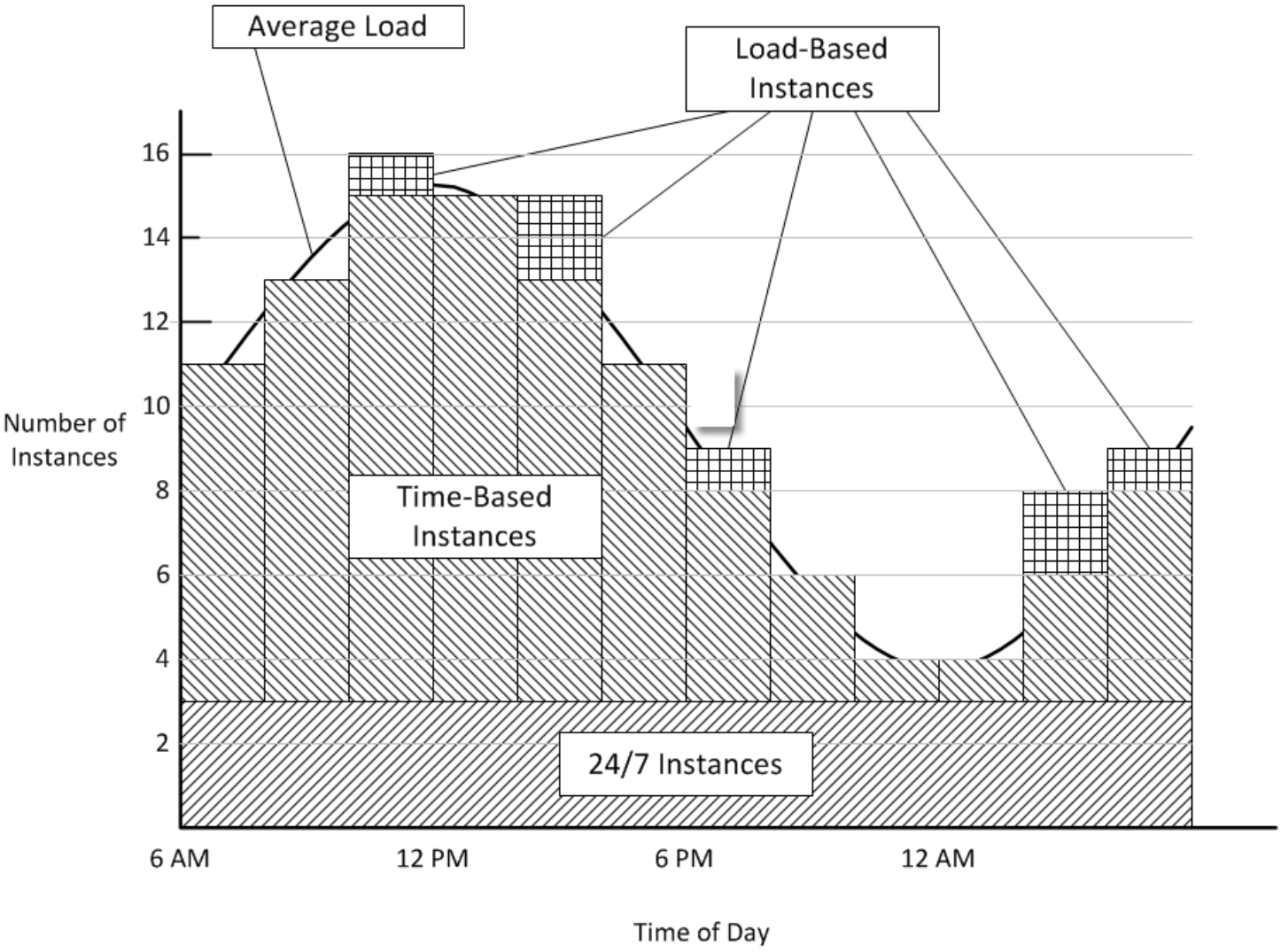
- 24/7 인스턴스는 자동으로 시작되어 수동으로 중지할 때까지 실행됩니다.
- 시간 기반 인스턴스는 사용자 지정 일정에 따라 AWS OpsWorks Stacks에 의해 자동으로 시작 및 중지됩니다.
- 로드 기반 인스턴스는 CPU 또는 메모리 사용률과 같은 사용자 지정 로드 지표에 대한 임계값을 초과하면 AWS OpsWorks Stacks에 의해 자동으로 시작되고 중지됩니다.

> **Note**
>
> 사용자가 스택의 시간 및 로드 기반 인스턴스를 생성하고 구성하면 AWS OpsWorks Stacks가 지정된 구성을 기반으로 자동으로 인스턴스를 시작하고 중지합니다. 사용자는 인스턴스의 구성 또는 수를 변경하는 이외에는 다시 손댈 필요가 없습니다.

권장 사항: 3개 이상의 애플리케이션 서버 인스턴스를 포함하는 스택을 관리하는 경우 위의 세 가지 인스턴스 유형을 모두 혼용하는 것이 좋습니다. 다음은 아래와 같은 특성을 갖는 가변적 일일 요청 볼륨을 처리하기 위해 스택의 서버 용량을 관리하는 방법의 예제입니다.

- 평균 요청 볼륨이 일중 파형적으로 변동
- 최소 평균 요청 볼륨을 처리하기 위해 5개의 애플리케이션 서버 인스턴스가 필요
- 최대 평균 요청 볼륨을 처리하기 위해 16개의 애플리케이션 서버 인스턴스가 필요
- 요청 볼륨 스파이크를 일반적으로 1~2개의 애플리케이션 서버 인스턴스에서 처리 가능

이 모델은 설명상 편의를 위한 것이지만, 어떠한 요청 볼륨 변동에도 간편하게 적응시킬 수 있으며 주 단위 변동을 처리하도록 확장할 수도 있습니다. 다음 다이어그램은 세 인스턴스 유형을 사용하여 이 요청 볼륨을 관리하는 방법을 보여줍니다.



이 예제에는 다음과 같은 특성이 있습니다.

- 스택에 3개의 24/7 인스턴스가 있습니다. 이들 인스턴스는 항상 켜져 있고 베이스 로드를 처리합니다.
- 스택에 12개의 시간 기반 인스턴스가 있습니다. 이들 인스턴스는 평균 일중 변동을 처리하도록 구성되어 있습니다.

  오후 10시부터 오전 2시까지는 1개의 인스턴스가 실행되고, 오후 8~10시 및 오전 2~4시에는 2개의 인스턴스가 추가로 실행되는 식입니다. 간결한 표시를 위해 다이어그램에서는 2시간마다 시간 기반 인스턴스 수를 수정하지만, 보다 세밀한 제어를 원한다면 1시간마다 수정할 수 있습니다.
- 이 스택은 24/7 및 시간 기반 인스턴스에서 처리할 수 있는 볼륨을 초과하는 트래픽 스파이크를 처리하는 데 충분한 로드 기반 인스턴스를 포함합니다.

AWS OpsWorks Stacks는 현재 실행 중인 모든 서버의 로드가 지정된 지표를 초과하는 경우에만 로드 기반 인스턴스를 시작합니다. 실행 중이지 않은 서버의 비용은 최소한(Amazon EBS 지원 인스턴스)이거나 없습니다(인스턴스 스토어 지원 인스턴스). 따라서 권장되는 방법은 최대 예상 요청 볼륨을 무리 없이 처리할 수 있는 충분한 수의 인스턴스를 생성하는 것입니다. 이 예제의 경우, 스택에 적어도 3개의 로드 기반 인스턴스가 있어야 합니다.

> **Note**
>
> 서비스 중단이 발생할 경우 영향을 완화시키려면 세 인스턴스 유형이 모두 여러 가용 영역으로 분산되어야 합니다.

모범 사례: 권한 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계정의 리소스에 액세스하려면 일정 형식의 AWS 자격 증명이 있어야 합니다. 다음은 직원들에게 액세스 권한을 제공하기 위한 몇 가지 일반적 지침입니다.

- 무엇보다도 AWS 리소스에 액세스할 때 계정의 루트 자격 증명을 사용하지 않는 것이 좋습니다.

  대신 직원을 위한 IAM ID를 생성하고 적절한 액세스를 제공하는 권한을 추가하세요. 그러면 직원 각자가 자신의 자격 증명을 사용하여 리소스에 액세스할 수 있습니다.
- 직원들은 자신의 업무를 수행하는 데 필요한 리소스에만 액세스할 수 있는 권한을 가져야 합니다.

  예를 들어 애플리케이션 개발자는 개발자의 애플리케이션을 실행하는 스택에만 액세스해야 합니다.
- 직원들은 자신의 업무를 수행하는 데 필요한 작업만을 사용할 수 있는 권한을 가져야 합니다.

  애플리케이션 개발자는 개발 스택에 대한 완전한 권한과 앱을 해당 프로덕션 스택에 배포할 수 있는 권한이 필요할 수 있습니다. 프로덕션 스택에서 인스턴스를 시작하거나 중지하고 계층을 생성 또는 삭제하는 등의 권한은 필요하지 않을 수 있습니다.

권한 관리에 대한 일반적인 내용은 AWS 보안 자격 증명을 참조하세요.

AWS OpsWorks Stacks 또는 IAM을 사용하여 사용자 권한을 관리할 수 있습니다. 두 방법은 상호 배타적이지 않습니다. 경우에 따라서는 두 방법을 모두 사용하는 것이 바람직합니다.

AWS OpsWorks 스택 권한 관리

각 스택에는 사용자에게 스택에 액세스할 권한을 부여하고 사용자가 수행할 수 있는 작업을 지정하는 데 사용할 수 있는 [권한] 페이지가 있습니다. 다음 권한 수준 중 하나를 설정하여 사용자의 권한을 지정합니다. 각각의 수준은 표준 작업 집합에 대한 권한을 부여하는 IAM 정책을 나타냅니다.

- [거부]는 어떤 방식으로든 스택과 상호 작용할 수 있는 권한을 거부합니다.
- [표시]는 스택 구성을 볼 수는 있지만 스택 상태는 수정하지 못하는 권한을 부여합니다.
- [배포]는 [표시] 권한을 포함하며 앱을 배포할 수 있는 사용자 권한도 부여합니다.
- [관리]는 [배포] 권한을 포함하며 사용자가 인스턴스 및 계층 생성 또는 삭제 등 다양한 스택 관리 작업을 수행하는 것도 허용합니다.

> **Note**
>
> 권한 관리 수준은 AWS OpsWorks 스택 생성 또는 복제를 포함하여 소수의 상위 수준 스택 작업에 대한 권한을 부여하지 않습니다. 이러한 권한을 부여하기 위해서는 IAM 정책을 사용해야 합니다.

권한 수준을 설정하는 것 외에도 스택의 [권한] 페이지를 사용하여 사용자가 스택의 인스턴스에서 SSH/RDP 및 sudo/admin 권한을 가질지 여부를 지정할 수 있습니다. AWS OpsWorks Stacks 권한 관리에 대한 자세한 정보는 스택별 권한 부여 단원을 참조하세요. SSH 액세스 관리에 대한 자세한 정보는 SSH 액세스 관리를 참조하세요.

IAM 권한 관리

IAM 권한 관리를 통해 IAM 콘솔, API 또는 CLI를 사용하여 사용자의 권한을 명시적으로 지정하는 JSON 형식 정책을 사용자에게 연결할 수 있습니다. IAM 권한 관리에 대한 자세한 내용은 IAM이란?을 참조하세요.

권장 사항: AWS OpsWorks Stacks 권한 관리로 시작합니다. 사용자의 권한을 세부 조정하거나 [Manage] 권한 수준에 포함되지 않은 사용자 권한을 부여해야 하는 경우, 두 가지 방법을 조합할 수 있습니다. AWS OpsWorks 그런 다음 Stacks는 두 정책을 모두 평가하여 사용자의 권한을 결정합니다.

> **Important**
>
> 사용자에게 권한이 충돌하는 여러 정책이 있는 경우, 거부가 항상 우선합니다. 예를 들어, 특정 스택에 대한 액세스를 허용하는 IAM 정책을 사용자에게 연결하면서 스택의 권한 페이지를 사용하여 사용자에게 거부 권한 수준을 할당한다고 가정해 보겠습니다. 이 경우, [거부] 권한 수준이 우선하며, 해당 사용자는 스택에 액세스할 수 없습니다. 자세한 내용은 IAM 정책 평가 논리를 참조하세요.

예를 들어 계층 추가 또는 삭제를 제외한 대부분의 작업을 사용자가 스택에서 수행할 수 있도록 하려는 경우를 가정해 보십시오.

- 사용자가 계층 생성 및 삭제를 포함한 대부분의 스택 관리 작업을 수행하는 것을 허용하는 [관리] 권한 수준을 지정합니다.
- 다음과 같은 고객 관리형 정책을 사용자에게 연결합니다. 그러면 해당 스택에서 CreateLayer 및 DeleteLayer 작업을 사용할 권한이 거부됩니다. 스택의 설정 페이지에서 찾을 수 있는 Amazon 리소스 이름(ARN)으로 스택을 식별합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

정책 예제를 포함한 자세한 내용은 IAM 정책을 연결하여 AWS OpsWorks 스택 권한 관리을(를) 참조하세요.

> **Note**
>
> IAM 정책을 사용하는 또 다른 방법은 지정된 IP 주소나 주소 범위를 가진 직원들로만 스택 액세스를 제한하는 조건을 설정하는 것입니다. 예를 들어 직원들이 기업 방화벽 내부에서만 스택에 액세스하도록 하려면 기업 IP 주소 범위로 액세스를 제한하는 조건을 설정합니다. 자세한 내용은 조건을 참조하세요.

모범 사례: 앱과 쿡북의 관리 및 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 원격 리포지토리에서 각 새 인스턴스에 앱과 쿡북을 배포합니다. 인스턴스의 수명 주기 동안 종종 스택의 온라인 인스턴스에서 앱 또는 쿡북을 업데이트하여 기능을 추가하고, 버그를 수정하는 등의 작업을 수행해야 합니다. 다양한 방법으로 스택의 앱과 쿡북을 관리할 수 있지만, 다음의 일반 요구 사항을 충족하는 접근 방식을 사용해야 합니다.

- A/B 테스트와 같은 일부 목적을 제외하고 모든 프로덕션 스택 인스턴스는 동일한 애플리케이션 및 사용자 지정 쿡북 코드를 실행해야 합니다.
- 업데이트 배포 시 뭔가가 잘못되더라도 사이트의 운영이 중단되지 않아야 합니다.

이 섹션에서는 앱과 사용자 지정 쿡북을 관리 및 배포하기 위해 권장되는 방법을 설명합니다.

주제

- 일관성 유지
- 온라인 인스턴스에 코드 배포

일관성 유지

일반적으로 프로덕션 스택에서 실행되는 앱 또는 쿡북 코드는 엄밀히 통제해야 합니다. 대개 모든 인스턴스가 현재 승인된 버전의 코드를 실행해야 합니다. 나중에 설명하는 것처럼 앱 또는 쿡북을 업데이트할 때 그리고 A/B 테스트와 같은 특별한 경우는 예외입니다.

앱 및 쿡북 코드는 다음 두 가지 방식으로 지정된 소스 리포지토리에서 스택의 인스턴스에 배포됩니다.

- 인스턴스를 시작하면 AWS OpsWorks Stacks는 현재 앱과 쿡북 코드를 인스턴스에 자동으로 배포합니다.
- 온라인 인스턴스에 대해서는 배포 명령(앱의 경우) 또는 사용자 지정 쿡북 업데이트 명령(쿡북의 경우)을 실행하여 현재 앱 또는 쿡북 코드를 수동으로 배포해야 합니다.

배포 메커니즘이 두 가지이므로 의도치 않게 인스턴스마다 다른 코드를 실행하지 않도록 주의하여 소스 코드를 관리하는 것이 매우 중요합니다. 예를 들어, Git 마스터 브랜치에서 앱 또는 쿡북을 배포하는 경우 AWS OpsWorks Stacks는 해당 브랜치에 있는 항목을 당시에 배포합니다. 사용자가 마스터 브랜치에서 코드를 업데이트한 후 새 인스턴스를 시작한다면 이 인스턴스에서는 이전 인스턴스보다 최신 버전의 코드가 실행됩니다. 더 최신 버전이 프로덕션용으로 승인되지 않았을 수도 있습니다.

권장 사항: Amazon S3 아카이브

모든 인스턴스에서 승인된 코드 버전이 실행되도록 하려면 Amazon Simple Storage Service(S3) 아카이브에서 앱 및 쿡북을 배포하는 것이 좋습니다. 이렇게 하면 코드가 명시적으로 업데이트되어야 하는 정적 아티팩트(.zip 또는 기타 아카이브 파일)임을 확인할 수 있습니다. 또한 Amazon S3는 안정성이 뛰어나므로 아카이브에 액세스하지 못하는 경우는 거의 없습니다. 추가적으로 일관성을 보장하려면, 명명 규칙을 사용하거나 감사 트레일과 간편하게 이전 버전 복구를 제공하는 Amazon S3 버전 관리를 사용하여 각 아카이브 파일을 명시적으로 버전 관리합니다.

예를 들어 Jenkins 같은 도구를 사용하여 배포 파이프라인을 생성할 수 있습니다. 배포하려는 코드를 커밋하고 테스트한 후 아카이브 파일을 생성하여 Amazon S3로 업로드합니다. 모든 앱 배포 또는 쿡북 업데이트가 이 아카이브 파일의 코드를 설치하고 모든 인스턴스가 동일한 코드를 실행합니다.

권장 사항: Git 또는 하위 버전 리포지토리

Git 또는 하위 버전 리포지토리 사용을 선호하는 경우 마스터 브랜치에서 배포하지 마십시오. 대신, 승인된 버전을 태그 지정하고 해당 버전을 앱 또는 쿡북 소스로 지정합니다.

온라인 인스턴스에 코드 배포

AWS OpsWorks Stacks는 업데이트된 코드를 온라인 인스턴스에 자동으로 배포하지 않습니다. 사용자가 수동으로 이 작업을 수행해야 하는데, 이때 다음과 같은 도전 과제를 해결해야 합니다.

- 배포 프로세스 도중 사이트가 고객 요청을 처리하는 능력에 악영향을 주지 않고 효율적으로 업데이트를 배포
- 배포된 앱 또는 쿡북의 문제 또는 배포 프로세스 자체의 문제로 인해 실패한 배포를 처리

가장 간단한 접근 방식은 모든 인스턴스에 동시에 업데이트를 배포하는 기본 배포 명령(앱의 경우) 또는 사용자 지정 쿡북 업데이트 명령(쿡북의 경우)을 실행하는 것입니다. 이 접근 방식은 간단하고 빠르지만 오류의 여지가 없습니다. 배포가 실패하거나 업데이트된 코드에 문제가 있을 경우 프로덕션 스택의 모든 인스턴스가 영향을 받을 수 있으므로 문제를 해결하거나 이전 버전으로 롤백할 때까지는 사이트가 중단 또는 비활성화될 수 있습니다.

권장 사항: 사용자가 배포 성공을 확인하여 확신을 가지고 모든 수신 트래픽을 새 버전으로 이전할 수 있을 때까지는 이전 버전의 코드를 실행하여 요청을 계속 처리하도록 인스턴스를 허용하는 강력한 배포 전략을 사용합니다.

다음 섹션에서는 강력한 배포 전략의 두 가지 예를 제시하고 배포 시 백엔드 데이터베이스를 관리하는 방법을 설명합니다. 간결한 설명을 위해 앱 업데이트에 대한 설명만 나와 있지만 쿡북에도 유사하게 적용할 수 있습니다.

주제

- 롤링 배포 사용
- 별도 스택 사용
- 백엔드 데이터베이스 관리

롤링 배포 사용

롤링 배포는 스택의 온라인 애플리케이션 서버 인스턴스에서 애플리케이션을 여러 단계에 걸쳐 업데이트합니다. 각 단계마다 온라인 인스턴스의 하위 집합을 업데이트하고 업데이트가 성공했는지 확인한 후 다음 단계를 시작합니다. 문제가 발생할 경우 이전 앱 버전을 실행 중인 인스턴스가 문제 해결 시까지 계속 수신 트래픽을 처리할 수 있습니다.

다음 예제는 사용자가 권장 방법을 사용하여 여러 가용 영역에 걸쳐 스택의 애플리케이션 서버 인스턴스를 배포하는 것으로 가정합니다.

롤링 배포를 수행하려면

1. 앱 배포 페이지에서 [고급]을 선택하고 단일 애플리케이션 서버 인스턴스를 선택한 다음 앱을 해당 인스턴스에 배포합니다.

   좀 더 주의를 기울이고 싶다면 앱을 배포하기 전에 로드 밸런서에서 인스턴스를 제거할 수 있습니다. 그러면 사용자가 애플리케이션이 제대로 작동하는지 확인할 때까지 애플리케이션이 업데이트되지 않습니다. Elastic Load Balancing을 사용하는 경우 Elastic Load Balancing 콘솔, CLI 또는 SDK를 사용하여 로드 밸런서에서 인스턴스를 제거하세요.
2. 업데이트된 앱이 제대로 작동하고 인스턴스의 성능 지표가 허용 가능한지 확인합니다.

   Elastic Load Balancing 로드 밸런서에서 인스턴스를 제거한 경우 Elastic Load Balancing 콘솔, CLI 또는 SDK를 사용하여 복원합니다. 이제는 업데이트된 앱 버전이 사용자 요청을 처리합니다.
3. 가용 영역에서 인스턴스의 나머지 부분에 대해 업데이트를 배포하고 인스턴스가 올바르게 작동하고 지표가 허용 가능한지 확인합니다.
4. 스택의 다른 가용 영역에 대해 3단계를 반복하되, 한 번에 한 영역씩 처리합니다. 특히 주의를 기울이려는 경우 1 - 3단계를 반복합니다.

> **Note**
>
> Elastic Load Balancing 로드 밸런서를 사용하는 경우 상태 확인을 사용하여 배포 성공 여부를 확인할 수 있습니다. 단, ping 경로를 단지 애플리케이션 서버가 실행 중인지 확인하는 정적 파일이 아니라 종속성을 검사하고 모든 것이 올바로 작동하는지 확인하는 애플리케이션으로 설정해야 합니다.

별도 스택 사용

애플리케이션을 관리하기 위한 또 하나의 접근 방식은 애플리케이션 수명 주기의 각 단계마다 별도의 스택을 사용하는 것입니다. 때로는 이러한 스택을 환경이라고 부르기도 합니다. 이 배열은 공개적으로 액세스할 수 없는 스택에서 개발 및 테스트를 수행할 수 있게 해줍니다. 업데이트를 배포할 준비가 완료되면 사용자 트래픽을 현재 애플리케이션 버전을 호스트하는 스택에서 업데이트된 버전을 호스트하는 스택으로 전환합니다.

주제

- 개발, 스테이징 및 프로덕션 스택 사용
- 블루-그린 배포 전략 사용

개발, 스테이징 및 프로덕션 스택 사용

가장 일반적인 접근 방식에서는 다음과 같은 스택을 사용합니다.

개발 스택

새로운 기능 구현 또는 버그 수정과 같은 작업에 개발 스택을 사용합니다. 개발 스택은 기본적으로 프로덕션 스택과 동일한 계층, 앱, 리소스 등을 포함하는 프로토타입 프로덕션 스택입니다. 대개의 경우 개발 스택이 프로덕션 스택과 동일한 로드를 처리할 필요는 없으므로 더 적은 수의 인스턴스를 사용하는 것이 일반적입니다.

개발 스택은 퍼블릭 스택이 아니며, 다음과 같이 액세스를 제어합니다.

- 애플리케이션 서버 또는 로드 밸런서의 보안 그룹 인바운드 규칙을 지정된 IP 주소 또는 주소 범위로부터 수신된 요청만 허용하도록 구성하여 네트워크 액세스를 제한합니다.

  예를 들어 HTTP, HTTPS 및 SSH 액세스를 회사 주소 범위 내 주소로 제한합니다.
- AWS OpsWorks 스택의 권한 페이지를 사용하여 스택 스택 관리 기능에 대한 액세스를 제어합니다.

  예를 들어 개발 팀에 관리 권한 수준을, 다른 모든 직원에게는 표시 권한을 부여합니다.

스테이징 스택

스테이징 스택을 사용하여 업데이트된 프로덕션 스택의 후보를 테스트하고 완료합니다. 개발을 완료했으면 개발 스택을 복제하여 스테이징 스택을 생성합니다. 그런 다음 스테이징 스택에서 테스트 도구 모음을 실행하고 업데이트를 이 스택에 배포하여 발생하는 문제를 해결합니다.

스테이징 스택도 퍼블릭 스택이 아닙니다. 개발 스택에서와 마찬가지 방식으로 스택 및 네트워크 액세스를 제어합니다. 개발 스택을 복제하여 스테이징 스택을 생성할 때 AWS OpsWorks Stacks 권한 관리에서 부여한 권한을 복제할 수 있습니다. 하지만 복제가 사용자의 IAM 정책에 의해 부여되는 권한에는 영향을 미치지 않습니다. 이러한 권한을 수정하려면 IAM 콘솔, CLI 또는 SDK를 사용해야 합니다. 자세한 내용은 사용자 권한 관리 단원을 참조하십시오.

프로덕션 스택

프로덕션 스택은 현재 애플리케이션을 지원하는 퍼블릭 스택입니다. 스테이징 스택이 테스트를 통과하면 이 스택을 프로덕션으로 승격하고 이전 프로덕션 스택은 사용 중지합니다. 이렇게 하는 방법의 예는 블루-그린 배포 전략 사용 섹션을 참조하세요.

> **Note**
>
> AWS OpsWorks Stacks 콘솔을 사용하여 스택을 수동으로 생성하는 대신 각 스택에 대한 AWS CloudFormation 템플릿을 생성합니다. 이 접근 방식에는 다음과 같은 장점이 있습니다.
>
> - 속도 및 편의성 - 템플릿을 시작하면, AWS CloudFormation 이(가) 모든 필요한 인스턴스를 포함하여 스택을 자동으로 생성합니다.
> - 일관성 - 소스 리포지토리에 각 스택의 템플릿 저장하여 개발자들이 동일한 목적에 동일한 스택을 사용하게 합니다.

블루-그린 배포 전략 사용

블루-그린 배포 전략은 별도의 스택을 효율적으로 사용하여 애플리케이션 업데이트를 프로덕션에 배포하는 일반적인 전략입니다.

- 블루 환경은 현재 애플리케이션을 호스트하는 프로덕션 스택입니다.
- 그린 환경은 업데이트된 애플리케이션을 호스트하는 스테이징 스택입니다.

업데이트된 앱을 프로덕션에 배포할 준비가 완료되면 사용자 트래픽을 블루 스택에서 그린 스택으로 전환합니다. 그러면 그린 스택이 새 프로덕션 스택이 됩니다. 그런 다음 기존의 블루 스택을 사용 중지합니다.

다음 예제에서는 AWS OpsWorks Stacks 스택을 Route 53 및 Elastic Load Balancing 로드 밸런서 풀과 함께 사용하여 블루-그린 배포를 수행하는 방법을 설명합니다. 스택을 전환하기 전에 다음 사항을 확인해야 합니다.

- 그린 스택의 애플리케이션 업데이트가 테스트를 통과하고 프로덕션 준비가 완료되어야 합니다.
- 그린 스택이 업데이트된 앱을 포함하는 점 이외에 블루 스택과 동일하고 퍼블릭 스택이 아니어야 합니다.

  두 스택 모두 권한이 동일하고, 각 계층 내 인스턴스 수 및 유형이 동일하고, 시간 기반 및 로드 기반 구성이 동일해야 합니다.
- 모든 그린 스택의 24/7 인스턴스 및 예약된 시간 기반 인스턴스가 온라인 상태여야 합니다.
- Elastic Load Balancing 로드 밸런서 풀은 어느 스택의 계층에 동적으로 연결될 수 있고 예상 트래픽 볼륨을 처리하도록 미리 워밍할 수 있습니다.

- Route 53 가중치 기반 라우팅 기능을 사용하여 로드 밸런서 풀을 포함하는 호스팅 영역에서 레코드 세트를 생성해야 합니다.
- 블루 스택의 애플리케이션 서버 계층에 연결된 로드 밸런서에 0이 아닌 가중치를 할당하고 사용되지 않는 로드 밸런서에는 0의 가중치를 할당했어야 합니다. 이는 블루 스택의 로드 밸런서가 모든 수신 트래픽을 처리하도록 보장합니다.

사용자를 그린 스택으로 전환하려면

1. 그린 스택의 애플리케이션 서버 계층에 풀의 사용되지 않는 로드 밸런서 중 하나를 연결합니다. 플래시 트래픽이 예상되거나 트래픽을 점차적으로 증가시키도록 부하 테스트를 구성할 수 없는 경우와 같은 일부 시나리오에서는 로드 밸런서를 사전 워밍하여 예상 트래픽을 처리합니다.
2. 그린 스택의 모든 인스턴스가 Elastic Load Balancing 상태 확인을 통과하면 Route 53 레코드 세트에서 가중치를 변경합니다. 그러면 그린 스택의 로드 밸런서 가중치가 0이 아닌 상태가 되고 블루 스택의 로드 밸런서가 이에 따라 가중치를 줄입니다. 그린 스택에서 요청의 극히 일부, 대략 5% 정도를 처리하도록 하고 나머지는 블루 스택에서 처리하도록 하는 것으로 시작하면 좋습니다. 이제, 수신되는 요청의 일부를 처리하는 그린 스택과 나머지 요청을 처리하는 블루 스택, 이렇게 두 가지 프로덕션 스택이 있습니다.
3. 그린 스택의 성능 지표를 모니터링합니다. 이러한 성능 지표가 허용 가능한 경우 수신 트래픽의 10%를 처리하도록 그린 스택의 가중치를 늘립니다.
4. 그린 스택이 수신 트래픽의 약 절반을 처리할 때까지 3단계를 반복합니다. 이 시점까지 모든 문제가 표면화되어야 합니다. 따라서 그린 스택이 허용 가능한 상태로 수행 중인 경우 블루 스택의 가중치를 0으로 줄여 프로세스를 완료할 수 있습니다. 이제 그린 스택이 새로운 블루 스택이 되어 수신 트래픽을 모두 처리 중입니다.
5. 이전 블루 스택의 애플리케이션 서버 계층에서 로드 밸런서를 분리하고 풀로 반환합니다.
6. 이전 블루 스택이 사용자 요청을 더 이상 처리하지 않더라도 새 블루 스택에 문제가 발생하는 경우를 대비해 얼마 동안은 이전 블루 스택을 유지하는 것이 좋습니다. 문제가 발생하면 위의 절차를 반대로 수행해 수신 트래픽을 다시 이전 블루 스택으로 연결하여 업데이트를 롤백할 수 있습니다. 새 블루 스택이 허용 가능한 상태로 작동한다는 확신이 들면 이전 블루 스택을 종료하세요.

백엔드 데이터베이스 관리

애플리케이션이 백엔드 데이터베이스에 의존하는 경우 이전 애플리케이션에서 새 애플리케이션으로 전환해야 합니다. AWS OpsWorks Stacks는 다음 데이터베이스 옵션을 지원합니다.

Amazon RDS 계층

Amazon Relational Database Service(RDS) 계층에서는 RDS DB 인스턴스를 별도로 생성한 후 스택에 등록합니다. RDS DB 인스턴스는 한 번에 한 스택에만 등록할 수 있지만 RDS DB 인스턴스를 한 스택에서 다른 스택으로 전환할 수 있습니다.

AWS OpsWorks Stacks는 애플리케이션에서 쉽게 사용할 수 있는 형식으로 애플리케이션 서버에 연결 데이터가 포함된 파일을 설치합니다. AWS OpsWorks Stacks는 또한 데이터베이스 연결 정보를 스택 구성 및 배포 속성에 추가하며, 레시피에서 액세스할 수 있습니다. JSON을 사용하여 애플리케이션에 연결 데이터를 제공할 수도 있습니다. 자세한 내용은 데이터베이스에 연결 단원을 참조하십시오.

데이터베이스에 의존하는 애플리케이션을 업데이트할 때 다음 두 가지의 기본적인 도전 과제를 해결해야 합니다.

- 전환 시 모든 트랜잭션을 적절히 기록하면서도 새 애플리케이션 버전과 기존 버전 간 경합 조건을 방지
- 사이트의 성능에 대한 영향을 제한하고 가동 중지를 최소화 내지는 배제하는 방식으로 전환을 수행

이 항목에서 설명하는 배포 전략을 사용하는 경우 단지 데이터베이스를 기존 애플리케이션에서 분리하고 새 애플리케이션에 다시 연결할 수는 없습니다. 전환 시 애플리케이션의 두 버전이 동시에 실행되며 동일한 데이터에 액세스할 수 있어야 합니다. 다음은 전환을 관리하는 두 가지 접근 방식에 대한 설명입니다. 각 접근 방식 모두 장점과 도전 과제가 있습니다.

접근 방식 1: 두 애플리케이션 모두 동일한 데이터베이스에 연결

장점

- 전환 시 가동 중지가 없습니다.

  한 애플리케이션이 점진적으로 데이터베이스 액세스를 중지하고 다른 애플리케이션이 점진적으로 대신합니다.
- 두 데이터베이스 간에 데이터를 동기화할 필요가 없습니다.

도전 과제

- 두 애플리케이션 모두 동일한 데이터베이스에 액세스하므로 데이터 손실 또는 손상이 방지되도록 액세스를 관리해야 합니다.
- 새 데이터베이스 스키마를 마이그레이션해야 할 경우 기존 애플리케이션 버전이 새 스키마를 사용할 수 있어야 합니다.

별도의 스택을 사용할 경우 아마도 이 접근 방식이 Amazon RDS에 가장 적합할 것입니다. 인스턴스가 특정 스택에 고정적으로 결합되지 않고 다른 스택에서 실행되는 애플리케이션에 의해 액세스될 수 있기 때문입니다. 하지만 RDS DB 인스턴스를 여러 스택에 동시에 등록할 수는 없습니다. 그러므로 예를 들어 JSON을 사용하여 두 애플리케이션에 모두 연결 데이터를 제공해야 합니다. 자세한 내용은 사용자 지정 레시피 사용 단원을 참조하십시오.

롤링 업그레이드를 사용하는 경우 기존 및 새 애플리케이션 버전이 동일한 스택에서 호스트됩니다. 따라서 Amazon RDS 또는 MySQL 계층을 사용할 수 있습니다.

접근 방식 2: 각 애플리케이션 버전에 자체 데이터베이스를 제공

장점

- 각 버전마다 자체 데이터베이스가 있으므로 스키마 호환성이 필요하지 않습니다.

도전 과제

- 전환 시 데이터 손실 또는 손상 없이 두 데이터베이스 간에 데이터를 동기화해야 합니다.
- 동기화 절차로 인해 상당한 가동 중지가 발생하거나 사이트의 성능을 현저히 저하되지 않아야 합니다.

별도의 스택을 사용하는 경우 각 스택에 자체 데이터베이스가 있습니다. 롤링 배포를 사용하는 경우 각 애플리케이션마다 하나씩 두 데이터베이스를 스택에 연결할 수 있습니다. 기존 애플리케이션과 업데이트된 애플리케이션에 호환되는 데이터베이스 스키마가 없을 경우 이 접근 방식이 더 유용합니다.

권장 사항: 일반적으로 Amazon RDS 계층을 애플리케이션의 백엔드 데이터베이스로 사용하는 것이 좋습니다. 이 접근 방식이 더 유연하며 어떤 전환 시나리오에도 사용 가능하기 때문입니다. 전환을 처리하는 방법에 대한 자세한 내용은 Amazon RDS 사용 설명서를 참조하세요.

로컬로 쿡북 종속성 패키징

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Berkshelf를 사용하면 쿡북 종속성을 로컬로 패키징하고, Amazon S3에 패키지를 업로드하며, Amazon S3에서 패키지를 쿡북 소스로 사용하도록 스택을 수정할 수 있습니다. Amazon S3 버킷에 전

달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

다음 연습에서는 쿡북과 해당 종속성을 .zip 파일로 사전 패키징한 다음 .zip 파일을 AWS OpsWorks Stacks에서 Linux 인스턴스의 쿡북 소스로 사용하는 방법을 설명합니다. 첫 번째 안내서에서는 1개의 쿡북을 패키징하는 방법을 설명합니다. 두 번째 안내서에서는 여러 쿡북을 패키징하는 방법을 설명합니다.

시작하기 전에 Chef 커뮤니티가 만든 도구 모음인 Chef Development Kit(Chef DK라고도 함)를 설치하세요. `chef` 명령줄 도구를 사용하려면 이 키트가 필요합니다.

Chef 12에서 로컬로 종속성 패키징

Chef 12 Linux에서는 더 이상 Berkshelf가 스택 인스턴스에 기본적으로 설치되지 않습니다. 로컬 개발 시스템에서 Berkshelf를 설치한 후 사용하여 로컬에서 쿡북 종속성을 패키징하는 것이 좋습니다. 종속성이 포함된 패키지를 Amazon S3로 업로드합니다. 마지막으로 쿡북 소스로 업로드된 패키지를 사용하도록 Chef 12 Linux 스택을 수정합니다. Chef 12에서 쿡북을 패키징하는 경우 다음과 같은 차이점이 있습니다.

1. 로컬 컴퓨터에서 `chef` 명령줄 도구를 실행하여 쿡북을 생성합니다.

   ```
   chef generate cookbook "server-app"
   ```

   이 명령은 쿡북, Berksfile, `metadata.rb` 파일 및 레시피 디렉터리를 생성해 쿡북과 이름이 같은 폴더에 저장합니다. 다음 예제는 생성된 항목의 구조를 보여줍니다.

   ```
   server-app <-- the cookbook you've just created
       ### Berksfile
       ### metadata.rb
       ### recipes
   ```

2. 텍스트 편집기에서 `server-app` 쿡북이 종속될 쿡북을 가리키도록 Berksfile을 편집합니다. 이 예제에서는 `server-app`이 Chef Supermarket의 `java` 쿡북에 종속되도록 합니다. 버전 1.50.0 이상 마이너 버전을 지정하지만 작은 따옴표 안에 게시된 버전은 어떤 것이든 입력할 수 있습니다. 변경 내용을 저장하고 파일을 닫습니다.

   ```
   source 'https://supermarket.chef.io'
   cookbook 'java', '~> 1.50.0'
   ```

3. `metadata.rb` 파일을 편집하여 종속성을 추가합니다. 변경 내용을 저장하고 파일을 닫습니다.

```
depends 'java' , '~> 1.50.0'
```

4. Chef에서 자동으로 생성된 server-app 쿡북 디렉터리로 변경한 다음 package 명령을 실행하여 쿡북의 tar 파일을 만듭니다. 여러 쿡북을 패키징하는 경우 모든 쿡북이 저장된 루트 디렉터리에서 이 명령을 실행합니다. 단일 쿡북을 패키징하려면 쿡북 디렉터리 레벨에서 이 명령을 실행합니다. 이 예에서는 server-app 디렉터리에서 이 명령을 실행합니다.

```
berks package cookbooks.tar.gz
```

다음과 유사하게 출력됩니다. tar.gz file 파일이 로컬 디렉터리에 생성됩니다.

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

5. 에서 방금 생성한 패키지를 Amazon S3에 AWS CLI업로드합니다. 스택 설정에 필요하므로 S3에 업로드한 후 쿡북 패키지의 새 URL을 적어 둡니다.

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

다음과 유사하게 출력됩니다.

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

6. 스택에서 업로드한 패키지를 AWS OpsWorks 쿡북 소스로 사용하도록 스택을 수정합니다.

   a. [사용자 지정 Chef 쿡북 사용] 설정을 [예]로 설정합니다.

   b. [리포지토리 유형]을 [S3 아카이브]로 설정합니다.

   c. [리포지토리 URL]에 5단계에서 업로드한 쿡북 패키지의 URL을 붙여 넣습니다.

   스택 변경 내용을 저장합니다.

로컬로 종속성 패키징(1개 쿡북)

1. 로컬 컴퓨터에서 chef 명령줄 도구를 사용하여 쿡북을 생성합니다.

```
chef generate cookbook "server-app"
```

이 명령은 쿡북 및 Berksfile을 생성해 쿡북과 이름이 같은 폴더에 저장합니다.

2. Chef에서 자동으로 생성된 쿡북 디렉터리로 변경한 후 다음 명령을 실행하여 모든 항목을 패키징합니다.

```
berks package cookbooks.tar.gz
```

출력은 다음과 같습니다.

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

3. 에서 방금 생성한 패키지를 Amazon S3에 AWS CLI업로드합니다.

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

출력은 다음과 같습니다.

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

4. 스택에서 업로드한 패키지를 AWS OpsWorks 쿡북 소스로 사용하도록 스택을 수정합니다.

로컬로 종속성 패키징(여러 쿡북)

이 예제는 2개의 쿡북을 생성하고 종속성을 패키징합니다.

1. 로컬 컴퓨터에서 다음 chef 명령을 실행하여 쿡북을 2개 생성합니다.

```
chef generate cookbook "server-app"
chef generate cookbook "server-utils"
```

이 예제에서는 server-app 쿡북이 Java 구성을 수행하므로 Java에서 종속성을 추가해야 합니다.

2. community Java 쿡북에 대한 종속성을 추가하도록 server-app/metadata.rb를 편집합니다.

```
maintainer "The Authors"
maintainer_email "you@example.com"
license "all_rights"
description "Installs/Configures server-app"
long_description "Installs/Configures server-app"
version "0.1.0"
```

```
depends "java"
```

3. 쿡북 루트 디렉터리에서 Berksfile 파일을 다음과 같이 편집하여 Berkshelf에 패키징할 항목을 알립니다.

```
source "https://supermarket.chef.io"
cookbook "server-app", path: "./server-app"
cookbook "server-utils", path: "./server-utils"
```

이제 파일 구조가 아래와 같이 표시될 것입니다.

```
..
   ### Berksfile
   ### server-app
   ### server-utils
```

4. 마지막으로 zip 패키지를 생성하여 Amazon S3에 업로드하고 새 AWS OpsWorks 쿡북 소스를 사용하도록 Stacks 스택을 수정합니다. 이렇게 하려면 로컬로 종속성 패키징(1개 쿡북) 단원의 2~4 단계를 따릅니다.

추가 리소스

쿡북 종속성 패키징에 대한 자세한 정보는 다음 문서를 참조하세요.

- AWS DevOps 블로그의 How to Package Cookbook Dependencies Locally with Berkshelf
- AWS OpsWorks 포럼에서 Berkshelf를 사용하는 Linux Chef 12
- AWS OpsWorks 포럼에서 Chef 12의 Berkshelf
- 이 설명서의 사용자 지정 쿡북 설치 단원
- 이 설명서의 쿡북 리포지토리

스택

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택은 최상위 Stacks AWS OpsWorks 개체입니다. 일반적으로 PHP 애플리케이션 서비스와 같은 공통의 목적이 있기 때문에 집합적으로 관리하려는 인스턴스 집합을 나타냅니다. 스택은 컨테이너 역할을 수행하는 이외에 애플리케이션 및 쿡북 관리와 같이 인스턴스 그룹에 전체적으로 적용되는 작업을 처리합니다.

예를 들어 웹 애플리케이션 서비스가 목적인 스택은 다음과 같을 것입니다.

- 수신 트래픽의 일부를 처리하는 애플리케이션 서버 인스턴스 집합
- 트래픽을 수신하여 애플리케이션 서버 간에 분산하는 로드 밸런서 인스턴스 1개
- 애플리케이션 서버용 백엔드 데이터 스토어의 역할을 하는 데이터베이스 인스턴스 1개

일반적인 방법은 서로 다른 환경을 나타내는 여러 스택을 사용하는 것입니다. 일반적으로 스택 집합은 다음과 같이 구성됩니다.

- 개발자가 기능을 추가하고, 버그를 수정하고, 다른 개발 및 유지 관리 작업을 수행하는 데 사용할 개발 스택
- 업데이트 또는 버그 수정을 공개 전에 검증하기 위한 스테이징 스택
- 사용자로부터 수신되는 요청을 처리하는 퍼블릭 버전의 프로덕션 스택

이 섹션에서는 기본적인 스택 사용 방법을 설명합니다.

주제

- Amazon EC2-Classic에서 VPC로 스택 마이그레이션
- 새 스택 생성
- VPC에서 스택 실행
- 스택 업데이트
- 스택 복제
- AWS OpsWorks Stacks Stack 명령 실행
- 사용자 지정 JSON 사용
- 스택 삭제

Amazon EC2-Classic에서 VPC로 스택 마이그레이션

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는 Amazon EC2 Classic 네트워크 플랫폼에서 Amazon Virtual Private Cloud(VPC) 네트워크로 AWS OpsWorks Stacks 스택을 마이그레이션하는 방법을 설명합니다.

2013-12-04 이전에 AWS 계정을 생성한 경우 일부 AWS 리전에서 EC2-Classic을 지원할 수 있습니다. 향상된 네트워킹 및 새로운 인스턴스 유형과 같은 일부 Amazon EC2 리소스 및 기능에는 Virtual Private Cloud(VPC)가 필요합니다. 일부 리소스는 EC2-Classic 및 VPC 간에 공유될 수 있으며 일부는 공유될 수 없습니다. 서비스 중단을 방지하려면 AWS OpsWorks Stacks 스택을 VPC로 마이그레이션하는 것이 좋습니다.

주제

- 사전 조건
- AWS OpsWorks Stacks 스택을 VPC로 마이그레이션
- 다음 사항도 참조하세요.

사전 조건

시작하기 전에 AWS OpsWorks Stacks 구성 요구 사항을 충족하는 VPC가 있어야 합니다. 용 VPC에서 프라이빗 서브넷을 구성하려면이 가이드VPC에서 스택 실행의 섹션을 AWS OpsWorks Stacks참조하세요. Amazon VPC 관리 콘솔을 사용하여 사용자 지정 VPC를 만들 수 있습니다. 자세한 내용은 Amazon VPC 콘솔 마법사 구성과 Amazon Virtual Private Cloud 사용 설명서의 VPC 및 서브넷을 참조하세요.

마이그레이션을 계속하려면 사용할 VPC ID와 서브넷 ID가 필요합니다.

AWS OpsWorks Stacks 스택을 VPC로 마이그레이션

먼저 AWS OpsWorks Stacks 콘솔 또는 API를 사용하여 기존 EC2-Classic 스택을 복제합니다. 그런 다음 기존 스택의 리소스를 새 스택으로 이동합니다. 복제된 스택에서 새 인스턴스를 시작하고 앱을 배포

합니다. 새로운 인스턴스가 작동 중인지 확인합니다. 마지막으로 EC2-Classic 스택에서 EC2-Classic 리소스를 삭제한 다음 이전 스택을 삭제합니다.

1. 기존 EC2-Classic 스택을 VPC에 복제합니다. 스택을 복제하면 스택 설정, 계층, 앱, 사용자 및 사용자 권한이 새 스택에 복사됩니다. 스택을 복제하는 방법에 대한 자세한 내용은 이 가이드에서 스택 복제의 내용을 참조하세요.

   AWS OpsWorks Stacks API를 사용하여 스택을 복제할 수도 있습니다. AWS CLI 또는 AWS SDKs를 사용하여 스택을 복제하는 경우 `VpcId` 파라미터 값을에서 생성한 VPC의 ID로 설정합니다사전 조건. 자세한 내용을 알아보려면 AWS OpsWorks Stacks API 참조의 CloneStack 섹션을 참조하세요.
2. 복제된 스택의 계층에 새 인스턴스를 생성합니다. 사전 조건에서 생성한 서브넷의 ID를 지정해야 합니다. 스택에서 인스턴스를 생성하는 방법에 대한 자세한 내용은 이 가이드에서 계층에 인스턴스 추가의 내용을 참조하세요.
3. EC2 보안 그룹, Elastic Load Balancing 로드 밸런서, 탄력적 IP 주소와 같은 클래식 리소스를 VPC로 마이그레이션한 다음 복제된 스택과 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서에서 VPC로 리소스 마이그레이션을 참조하세요.
4. Amazon EBS 볼륨과 Amazon RDS 인스턴스를 복제된 스택에 등록합니다. 스택에 리소스를 등록하는 방법에 대한 자세한 내용은 이 안내서의 스택에 리소스 등록 섹션을 참조하세요.

   Amazon EBS 볼륨은 VPC와 연결되어 있지 않으므로 EC2-Classic 스택과 VPC의 스택 모두에 있는 인스턴스에서 사용할 수 있습니다. EC2-Classic의 Amazon RDS 인스턴스를 EC2-Classic 스택과 VPC의 스택을 모두 사용하여 등록할 수 있습니다.
5. 복제된 스택에서 인스턴스를 시작한 다음 워크로드의 일부를 복제된 스택으로 이동합니다. 예를 들어 소량의 트래픽을 복제된 스택의 Elastic Load Balancing 로드 밸런서로 이동합니다. Amazon Route 53를 사용하는 경우 Amazon Route 53 개발자 안내서의 ELB 로드 밸런서로 트래픽 라우팅을 참조하세요.

   새 스택이 제대로 작동하고 애플리케이션을 지원하는지 확인할 때까지 소량의 트래픽만 라우팅하세요. 시험 사용 기간(예: 일주일) 동안 적은 비율의 트래픽만 새 스택을 처리하도록 하세요. 새 스택이 작동하는지 확인한 후 나머지 트래픽을 스택으로 라우팅합니다.
6. 복제된 스택이 제대로 작동하는지 확인한 후 나머지 프로덕션 트랙 또는 워크로드를 복제된 스택으로 이동합니다. 이제 EC2-Classic 스택에서 인스턴스를 중지할 수 있습니다. 마이그레이션 후 몇 주 내에 새 스택에 문제가 발생할 경우 이전 스택으로 워크로드를 다시 이동할 수 있도록 이전 스택을 몇 주 동안 이용 가능하게 유지하는 것이 좋습니다.
7. 새 스택이 몇 주 동안 작동하면 EC2-Classic 스택에서 인스턴스를 삭제하세요. 스택 삭제 방법에 대한 자세한 내용은 이 설명서의 AWS OpsWorks 스택 인스턴스 삭제 섹션을 참조하세요.

> **Important**
>
> Amazon EC2 콘솔 또는 API를 사용하여 AWS OpsWorks 인스턴스를 중지하거나 삭제하지 마세요.

8. EC2-Classic 스택에서 앱을 삭제합니다. 앱을 삭제하는 방법에 대한 자세한 내용은 이 설명서의 스택에서 앱을 삭제하려면 섹션을 참조하세요.
9. EC2-Classic 스택을 삭제합니다. 스택을 삭제하는 방법에 대한 자세한 정보는 이 설명서의 스택 삭제 섹션을 참조하세요.

다음 사항도 참조하세요.

- EC2-Classic에서 VPC로 마이그레이션
- 디버깅 및 문제 해결 안내서
- VPC에서 스택 실행

새 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

새 스택을 생성하려면 AWS OpsWorks Stacks 대시보드에서 스택 추가를 클릭합니다. 그런 다음 [스택 추가] 페이지를 사용하여 스택을 구성할 수 있습니다. 다 마치면 [스택 추가]를 클릭합니다.

주제

- 생성할 스택 유형 선택
- 기본 옵션
- 고급 옵션

생성할 스택 유형 선택

스택을 생성하기 전에 생성하려는 스택 유형을 결정해야 합니다. 도움을 얻으려면 다음 표를 참조하세요.

...를 생성하려면	...하려면 이 유형의 스택 생성	방법을 알려면 다음 지침에 따르십시오.
샘플 스택	Linux 기반 Chef 12 스택과 샘플 Node.js 앱을 사용하여 AWS OpsWorks의 기초를 살펴봅니다.	시작하기: 샘플
Linux 기반 Chef 12 스택	AWS OpsWorks가 지원하는 최신 버전 Chef를 사용하는 Linux 기반 스택을 생성합니다. 커뮤니티 쿡북을 다수 선택하여 이득을 얻고자 하거나 본인만의 사용자 지정 쿡북을 작성하려는 고급 Chef 사용자인 경우, 이 옵션을 선택합니다. 자세한 내용은 Chef 12 Linux 단원을 참조하십시오.	시작하기: Linux
Windows 기반 Chef 12.2 스택	Windows 기반 스택을 생성합니다.	시작하기: Windows
Linux 기반 Chef 11.10 스택	조직에서 이전 버전과의 호환성을 위해 Linux와 함께 Chef 11.10을 사용해야 하는 경우, 이 스택을 생성하세요.	Chef 11 Linux 스택 시작하기

기본 옵션

[스택 추가] 페이지에는 다음과 같은 기본 옵션이 있습니다.

스택 이름

(필수) AWS OpsWorks Stacks 콘솔에서 스택을 식별하는 데 사용되는 이름입니다. 이름은 고유할 필요가 없습니다. AWS OpsWorks Stacks는 스택을 고유하게 식별하는 GUID인 스택 ID도 생성합니다. 예를 들어 업데이트 스택과 같은 AWS CLI 명령의 경우 스택 ID를 사용하여 특정 스택을 식별합니다. 스택을 생성한 후에 탐색 창에서 [스택]을 선택한 다음 [스택 설정]를 선택하면 스택의 ID를 찾을 수 있습니다. ID는 [OpsWorks ID]로 레이블 지정됩니다.

리전

(필수) 인스턴스가 시작될 AWS 리전.

VPC

(선택 사항) 스택이 시작될 VPC의 ID. 모든 인스턴스는 이 VPC로 시작되며, 나중에 ID를 변경할 수 없습니다.

- 계정이 EC2 Classic을 지원하는 경우, VPC를 사용하지 않으려면 [VPC 없음](기본값)을 지정할 수 있습니다.

  EC2 Classic에 대한 자세한 내용은 지원되는 플랫폼을 참조하세요.
- 계정이 EC2 Classic을 지원하지 않는 경우, VPC를 지정해야 합니다.

  기본 설정인 [기본 VPC]는 EC2 Classic의 쉬운 사용법과 VPC 네트워킹 기능의 장점을 결합합니다. 일반 VPC에서 스택을 실행하려면 VPC 콘솔, API 또는 CLI를 사용하여 VPC를 생성해야 합니다. AWS OpsWorks Stacks 스택용 VPC를 만드는 방법에 대한 자세한 정보는 VPC에서 스택 실행 단원을 참조하세요. 일반적인 내용은 Amazon Virtual Private Cloud를 참조하세요.

기본 가용 영역/기본 서브넷

(선택 사항) 이 설정은 스택을 VPC에서 생성하는지 여부에 따라 달라집니다.

- 계정이 EC2 Classic을 지원하고 [VPC]를 [VPC 없음]으로 설정하는 경우, 이 설정은 인스턴스가 시작될 기본 AWS 가용 영역을 지정하는 [기본 가용 영역]으로 레이블 지정됩니다.
- 계정이 EC2 Classic을 지원하지 않거나 VPC를 지정하는 경우, 이 필드는 인스턴스가 시작될 기본 서브넷을 지정하는 [기본 서브넷]으로 레이블 지정됩니다. 인스턴스를 생성할 때 이 값을 재정의하여 다른 서브넷에서 인스턴스를 시작할 수 있습니다. 각 서브넷은 하나의 가용 영역에 연결됩니다.

인스턴스를 생성할 때이 설정을 재정의하여 AWS OpsWorks Stacks가 다른 가용 영역 또는 서브넷에서 인스턴스를 시작하도록 할 수 있습니다.

VPC에서 스택을 실행하는 방법에 대한 자세한 정보는 VPC에서 스택 실행 단원을 참조하세요.

기본 운영 체제

(선택 사항) 각 인스턴스에 기본적으로 설치되는 운영 체제. 다음과 같은 옵션이 있습니다:

- 내장 Linux 운영 체제 중 하나.
- Microsoft Windows Server 2012 R2.
- 지원되는 운영 체제 중 하나에 기반하는 사용자 지정 AMI.

  [사용자 지정 AMI 사용]을 선택하는 경우, 인스턴스를 생성할 때 지정하는 사용자 지정 AMI에 의해 운영 체제가 결정됩니다. 자세한 내용은 사용자 지정 AMI 사용 단원을 참조하십시오.

사용 가능한 운영 체제에 대한 자세한 정보는 AWS OpsWorks 스택 운영 체제를 참조하세요.

> **Note**
>
> 인스턴스를 생성할 때 기본 운영 체제를 재정의할 수 있습니다. 하지만 Linux 운영 체제를 재정의해 Windows를 지정하거나 Windows를 재정의해 Linux 운영 체제를 지정할 수는 없습니다.

기본 SSH 키

(선택 사항) 스택의 리전의 Amazon EC2 키 페어. 기본값은 없습니다. 키 페어를 지정하면 AWS OpsWorks Stacks가 인스턴스에 퍼블릭 키를 설치합니다.

- Linux 인스턴스의 경우, SSH 클라이언트와 함께 프라이빗 키를 사용하여 스택의 인스턴스에 로그인할 수 있습니다.

  자세한 내용은 SSH를 사용하여 로그인 단원을 참조하십시오.
- Windows 인스턴스의 경우, Amazon EC2 콘솔 또는 CLI와 함께 프라이빗 키를 사용하여 인스턴스의 관리자 암호를 가져올 수 있습니다.

  그런 다음 이 암호를 RDP 클라이언트와 함께 사용하여 인스턴스에 Administrator로 로그인할 수 있습니다. 자세한 내용은 RDP를 사용하여 로그인 단원을 참조하십시오.

SSH 키를 관리하는 방법에 대한 자세한 정보는 SSH 액세스 관리를 참조하세요.

> **Note**
>
> 인스턴스를 생성할 때 다른 키 페어를 지정하거나 키 페어를 지정하지 않으면 이 설정을 재정의할 수 있습니다.

Chef 버전

선택한 Chef 버전을 보여 줍니다.

Chef 버전에 대한 자세한 정보는 Chef 버전 단원을 참조하세요.

사용자 지정 Chef 쿡북 사용

스택의 인스턴스에 사용자 지정 Chef 쿡북을 설치할지 여부.

Chef 12의 경우, 기본 설정은 [예]입니다. Chef 11의 경우 기본 설정은 아니요입니다. 예 옵션에는 AWS OpsWorks 리포지토리 URL과 같이 리포지토리에서 스택의 인스턴스로 사용자 지정 쿡북을 배포하는 데 필요한 정보를 Stacks에 제공하는 몇 가지 추가 설정이 표시됩니다. 세부적인 사항은 쿡북에 사용하는 리포지토리에 따라 달라집니다. 자세한 내용은 사용자 지정 쿡북 설치 단원을 참조하십시오.

스택 색상

(선택 사항) AWS OpsWorks Stacks 콘솔에서 스택을 나타내는 데 사용되는 색조입니다. 예컨대 개발 스택, 스테이징 스택, 프로덕션 스택을 구분할 수 있도록 스택마다 다른 색상을 사용할 수 있습니다.

스택 태그

스택 및 계층 수준에서 태그를 적용할 수 있습니다. 태그를 생성하면 태그가 지정된 구조 내의 모든 리소스에 태그가 적용됩니다. 예를 들어 스택에 태그를 적용하는 경우, 해당 태그가 모든 계층에 적용되며, 각 계층 내에서 계층에 있는 모든 인스턴스, Amazon EBS 볼륨 또는 Elastic Load Balancing 로드 밸런서에 태그가 적용됩니다. 태그를 활성화하고 이를 사용하여 AWS OpsWorks Stacks 리소스의 비용을 추적하고 관리하는 방법에 대한 자세한 내용은 Billing and Cost Management 사용 설명서의 비용 할당 태그 사용 및 사용자 정의 비용 할당 태그 활성화를 참조하세요. AWS OpsWorks Stacks의 태그 지정에 대한 자세한 내용은 섹션을 참조하세요Tags.

고급 옵션

고급 설정의 경우, [고급 >>]를 클릭하여 [고급 옵션] 및 [보안] 섹션을 표시합니다.

[고급 옵션] 섹션에는 다음 옵션이 있습니다.

기본 루트 디바이스 유형

인스턴스의 루트 볼륨에 사용할 스토리지 유형을 결정합니다. 자세한 내용은 스토리지를 참조하세요.

- Linux 스택은 기본적으로 Amazon EBS 지원 루트 볼륨을 사용하지만 인스턴스 스토어 지원 루트 볼륨을 지정할 수도 있습니다.
- Windows 스택은 Amazon EBS 지원 루트 볼륨을 사용해야 합니다.

IAM 역할

(선택 사항) Stacks가 사용자를 대신하여 AWS와 상호 작용하는 데 사용하는 스택의 AWS Identity and Access Management(IAM) 역할입니다. AWS OpsWorks

기본 IAM 인스턴스 프로파일

(선택 사항) 스택의 Amazon EC2 인스턴스에 연결할 기본 IAM 역할. 이 역할은 스택의 인스턴스에서 실행되는 애플리케이션에게 S3 버킷 같은 AWS 리소스에 액세스할 수 있는 권한을 부여합니다.

- 애플리케이션에 구체적인 권한을 부여하려면 적절한 정책이 있는 기존 인스턴스 프로파일(역할)을 선택합니다.
- 처음에는 프로파일의 역할이 아무런 권한도 부여하지 않지만 IAM 콘솔, API 또는 CLI를 사용하여 적절한 정책을 연결할 수 있습니다. 자세한 내용은 EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 단원을 참조하십시오.

API 엔드포인트 리전

이 설정은 스택의 기본 설정에서 선택하는 리전에서 값을 가져옵니다. 다음 리전 엔드포인트 중에서 선택할 수 있습니다.

- US East (N. Virginia) Region
- 미국 동부(오하이오) 리전
- US West (Oregon) Region
- US West (N. California) Region
- 캐나다(중부) 리전(API만 해당,에서 생성된 스택에는 사용할 수 없음 AWS Management Console
- Asia Pacific (Mumbai) Region
- Asia Pacific (Singapore) Region
- 아시아 태평양(시드니) 리전
- 아시아 태평양(도쿄) 리전
- Asia Pacific (Seoul) Region
- Europe (Frankfurt) Region
- Europe (Ireland) Region
- Europe (London) Region

- Europe (Paris) Region
- South America (São Paulo) Region

API 엔드포인트에서 생성되는 스택은 다른 API 엔드포인트에서 사용할 수 없습니다. AWS OpsWorks Stacks 사용자도 리전별로 다르므로 이러한 엔드포인트 리전 중 하나의 AWS OpsWorks Stacks 사용자가 다른 엔드포인트 리전의 스택을 관리하도록 하려면 사용자를 스택이 연결된 엔드포인트로 가져와야 합니다. 사용자 가져오기에 대한 자세한 내용은 AWS OpsWorks Stacks로 사용자 가져오기를 참조하세요.

호스트 이름 테마

(선택 사항) 각 인스턴스의 기본 호스트 이름을 생성하는 데 사용되는 문자열. 기본값인 [계층 종속적]은 인스턴스 계층의 짧은 이름을 사용하며 각 인스턴스에 고유한 번호를 추가합니다. 예를 들어 역할 종속적 로드 밸런서 테마 루트는 "lb"입니다. 계층에 추가하는 첫 번째 인스턴스는 "lb1"으로, 두 번째 인스턴스는 "lb2"로 명명되며, 나머지도 같은 방식입니다.

OpsWorks 에이전트 버전

(선택 사항) 새 버전을 사용할 수 있을 때 AWS OpsWorks Stacks 에이전트를 자동으로 업데이트할지 아니면 지정된 에이전트 버전을 사용하여 수동으로 업데이트할지 여부입니다. 이 기능은 Chef 11.10 및 Chef 12 스택에서 사용할 수 있습니다. 기본 설정은 [수동 업데이트]이며 최신 에이전트 버전으로 설정되어 있습니다.

AWS OpsWorks Stacks는 서비스와 통신하고 수명 주기 이벤트에 대한 응답으로 Chef 실행을 시작하는 등의 작업을 처리하는 에이전트를 각 인스턴스에 설치합니다. 이 에이전트는 정기적으로 업데이트됩니다. 스택의 에이전트 버전을 지정하는 옵션은 다음 두 가지입니다.

- 자동 업데이트 - AWS OpsWorks 스택은 업데이트를 사용할 수 있게 되는 즉시 스택의 인스턴스에 각 새 에이전트 버전을 자동으로 설치합니다.
- 수동 업데이트 - AWS OpsWorks 스택은 스택의 인스턴스에 지정된 에이전트 버전을 설치합니다.

  AWS OpsWorks Stacks는 새 에이전트 버전을 사용할 수 있을 때 스택 페이지에 메시지를 게시하지만 스택의 인스턴스는 업데이트하지 않습니다. 에이전트를 업데이트하려면 스택 설정을 수동으로 업데이트하여 새 에이전트 버전을 지정해야 합니다. 그러면 AWS OpsWorks Stacks가 스택의 인스턴스를 업데이트합니다.

특정 인스턴스의 기본 [OpsWorks Agent 버전] 설정은 인스턴스의 구성을 업데이트하여 재정의할 수 있습니다. 이 경우, 인스턴스의 설정이 우선 적용됩니다. 예를 들어 기본 설정이 [자동 업데이트]이지만 특정 인스턴스에 대해 [수동 업데이트]를 지정한다고 가정해 보십시오. AWS OpsWorks

Stacks가 새 에이전트 버전을 릴리스하면 수동 업데이트로 설정된 인스턴스를 제외한 스택의 모든 인스턴스가 자동으로 업데이트됩니다. 해당 인스턴스에 새 에이전트 버전을 설치하려면 수동으로 인스턴스의 구성을 업데이트하고 새 버전을 지정해야 합니다.

> **Note**
>
> 콘솔에 축약된 에이전트 버전 번호가 표시됩니다. 완전한 버전 번호를 보려면 AWS CLI describe-agent-versions 명령이나 이에 상당하는 API 또는 SDK 메서드를 호출하세요. 그러면 사용할 수 있는 에이전트 버전의 전체 버전 번호가 반환됩니다.

사용자 지정 JSON

(선택 사항) JSON 구조로 형식이 지정된 하나 이상의 사용자 지정 속성. 이러한 속성은 모든 인스턴스에 설치되고 레시피가 사용할 수 있는 스택 구성 및 배포 속성에 병합됩니다. 예를 들어 사용자 지정 JSON을 사용하여 기본 설정을 지정하는 내장 속성을 재정의하면 구성 설정을 사용자 지정할 수 있습니다. 자세한 내용은 사용자 지정 JSON 사용 단원을 참조하십시오.

보안에는 OpsWorks 보안 그룹 사용 옵션이 있습니다.이 옵션을 사용하면 AWS OpsWorks Stacks 기본 제공 보안 그룹을 스택 계층과 연결할지 여부를 지정할 수 있습니다.

AWS OpsWorks Stacks는 기본적으로 계층과 연결된 각 계층에 대해 하나씩 표준 내장 보안 그룹 세트를 제공합니다. [OpsWorks 보안 그룹 사용]을 사용하면 그 대신 자체 사용자 지정 보안 그룹을 제공할 수 있습니다. 자세한 내용은 보안 그룹 사용 단원을 참조하십시오.

[OpsWorks 보안 그룹 사용]에는 다음의 설정이 있습니다.

- 예 - AWS OpsWorks Stacks는 적절한 기본 제공 보안 그룹을 각 계층에 자동으로 연결합니다(기본 설정).

  추가 보안 그룹을 생성한 후 계층에 연결할 수 있지만 내장 보안 그룹은 삭제할 수 없습니다.
- No - AWS OpsWorks Stacks는 내장 보안 그룹을 계층과 연결하지 않습니다.

  적절한 EC2 보안 그룹을 생성하고 생성하는 각 계층에 보안 그룹을 연결해야 합니다. 그러나 생성 시 한 계층에 기본 보안 그룹을 수동으로 연결할 수도 있습니다. 커스텀 보안 그룹은 커스텀 설정이 필요한 계층에서만 필요합니다.

다음 사항에 유의하세요.

- [OpsWorks 보안 그룹 사용]가 [예]로 설정된 경우, 계층에 보다 제한적인 보안 그룹을 추가하더라도 기본 보안 그룹의 포트 액세스 설정을 제한할 수 없습니다. 여러 보안 그룹이 있는 경우, Amazon EC2는 가장 허용적인 설정을 사용합니다. 또한 내장 보안 그룹 구성을 수정하여 더 제한적인 설정을 생성할 수 없습니다. 스택을 생성할 때 AWS OpsWorks Stacks는 기본 제공 보안 그룹의 구성을 표준 설정으로 덮어쓰므로 다음에 스택을 생성할 때 변경한 내용이 모두 손실됩니다. 계층에 내장 보안 그룹보다 더 제한적인 보안 그룹 설정이 필요한 경우, [OpsWorks 보안 그룹 사용]을 [아니요]로 설정하고 기본 설정으로 사용자 지정 보안 그룹을 생성한 다음 생성 시 계층에 할당합니다.
- 실수로 AWS OpsWorks Stacks 보안 그룹을 삭제하고 다시 생성하려는 경우 그룹 이름의 대문자를 포함하여 원본과 정확히 중복되어야 합니다. 수동으로 그룹을 다시 생성하는 것보다는 AWS OpsWorks Stacks가 대신 이 작업을 수행하도록 하는 것이 좋습니다. 동일한 AWS 리전에 새 스택을 생성하고 VPC가 있는 경우 AWS OpsWorks Stacks는 삭제한 보안 그룹을 포함하여 모든 기본 제공 보안 그룹을 자동으로 다시 생성합니다. 그런 다음 더 이상 사용할 일이 없으면 스택을 삭제할 수 있습니다. 보안 그룹은 그대로 남습니다.

VPC에서 스택 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Virtual Private Cloud(VPC)에서 스택의 인스턴스를 생성하여 이에 대한 사용자 액세스를 제어할 수 있습니다. 예를 들어 스택의 앱 서버나 데이터베이스에 사용자들이 직접 액세스하는 대신 모든 퍼블릭 트래픽을 탄력적 로드 밸런서를 통해 채널링되도록 해야 할 경우가 있습니다.

VPC에서 스택을 실행하는 기본적 절차는 다음과 같습니다.

1. Amazon VPC 콘솔이나 API 또는 AWS CloudFormation 템플릿을 사용하여 적절히 구성된 VPC를 생성합니다.
2. 스택을 생성할 때 VPC ID를 지정합니다.
3. 적절한 서브넷에서 스택의 인스턴스를 시작합니다.

다음은 AWS OpsWorks Stacks에서 VPC가 작동하는 방식에 대한 간략한 설명입니다.

Important

VPC 엔드포인트 기능을 사용하는 경우, 스택의 각 인스턴스는 Amazon Simple Storage Service(S3)에서 다음 작업을 완료할 수 있어야 합니다.

- 인스턴스 에이전트 설치.
- Ruby 등의 자산 설치.
- Chef 실행 로그 업로드.
- 스택 명령 검색.

이러한 작업이 가능하려면 스택의 인스턴스는 스택의 리전과 일치하는 다음 버킷에 액세스할 수 있어야 합니다. 그렇지 않으면 이전 작업이 실패합니다.
Chef 12 Linux 및 Chef 12.2 Windows의 경우, 버킷은 다음과 같습니다.

Agent Buckets	Asset Buckets	Log Buckets	DNA Buckets
• opsworks-instance-agent-sa-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-northeast-1 • opsworks-instance-agent-ap-northeast-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-southeast-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1 • opsworks-instance-agent-eu-west-2	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-northeast-1 • opsworks-instance-assets-ap-northeast-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-southeast-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1	• opsworks-us-east-2-log • opsworks-us-east-1-log • opsworks-ap-south-1-log • opsworks-ap-northeast-1-log • opsworks-ap-northeast-2-log • opsworks-ap-southeast-1-log • opsworks-ap-southeast-2-log • opsworks-ca-central-1-log • opsworks-eu-central-1-log • opsworks-eu-west-1-log • opsworks-eu-west-2-log • opsworks-eu-west-3-log • opsworks-sa-east-1-log • opsworks-us-west-1-log • opsworks-us-west-2-log	• opsworks-us-east-2-dna • opsworks-us-east-1-dna • opsworks-ap-south-1-dna • opsworks-ap-northeast-1-dna • opsworks-ap-northeast-2-dna • opsworks-ap-southeast-1-dna • opsworks-ap-southeast-2-dna • opsworks-ca-central-1-dna • opsworks-eu-central-1-dna • opsworks-eu-west-1-dna • opsworks-eu-west-2-dna • opsworks-eu-west-3-dna • opsworks-sa-east-1-dna • opsworks-us-west-1-dna • opsworks-us-west-2-dna

Agent Buckets	Asset Buckets	Log Buckets	DNA Buckets
• opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-west-1 • opsworks-instance-agent-us-west-2	• opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1 • opsworks-instance-assets-us-west-2		

Linux용 Chef 11.10 및 이전 버전의 경우, 버킷은 다음과 같습니다. Chef 11.4 스택은 미국 동부(버지니아 북부) 밖의 리전 엔드포인트에서는 지원되지 않습니다.

Agent Buckets	Asset Buckets	Log Buckets	DNA Buckets
• opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-northeast-1 • opsworks-instance-agent-ap-northeast-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-southeast-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-northeast-1 • opsworks-instance-assets-ap-northeast-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-southeast-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1	• prod_stage-log	• prod_stage-dna

Agent Buckets	Asset Buckets	Log Buckets	DNA Buckets
• opsworks-instance-agent-eu-west-2 • opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-west-1 • opsworks-instance-agent-us-west-2	• opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1 • opsworks-instance-assets-us-west-2		

자세한 내용은 VPC 엔드포인트를 참조하세요.

Note

AWS OpsWorks Stacks 에이전트가 여전히 퍼블릭 엔드포인트에 액세스해야 하므로 AWS OpsWorks Stacks가 활성화한 VPC 엔드포인트에 연결하려면 NAT 또는 퍼블릭 IP에 대한 라우팅도 구성해야 합니다.

주제

- VPC 기초
- AWS OpsWorks Stacks 스택용 VPC 생성

VPC 기초

VPC에 대한 자세한 논의는 Amazon Virtual Private Cloud를 참조하세요. 간략히 말해 VPC는 각각 하나 이상의 인스턴스를 포함하고 있는 하나 이상의 서브넷으로 구성됩니다. 각각의 서브넷에는 대상 IP 주소를 기반으로 아웃바운드 트래픽을 전달하는 연결된 라우팅 테이블이 있습니다.

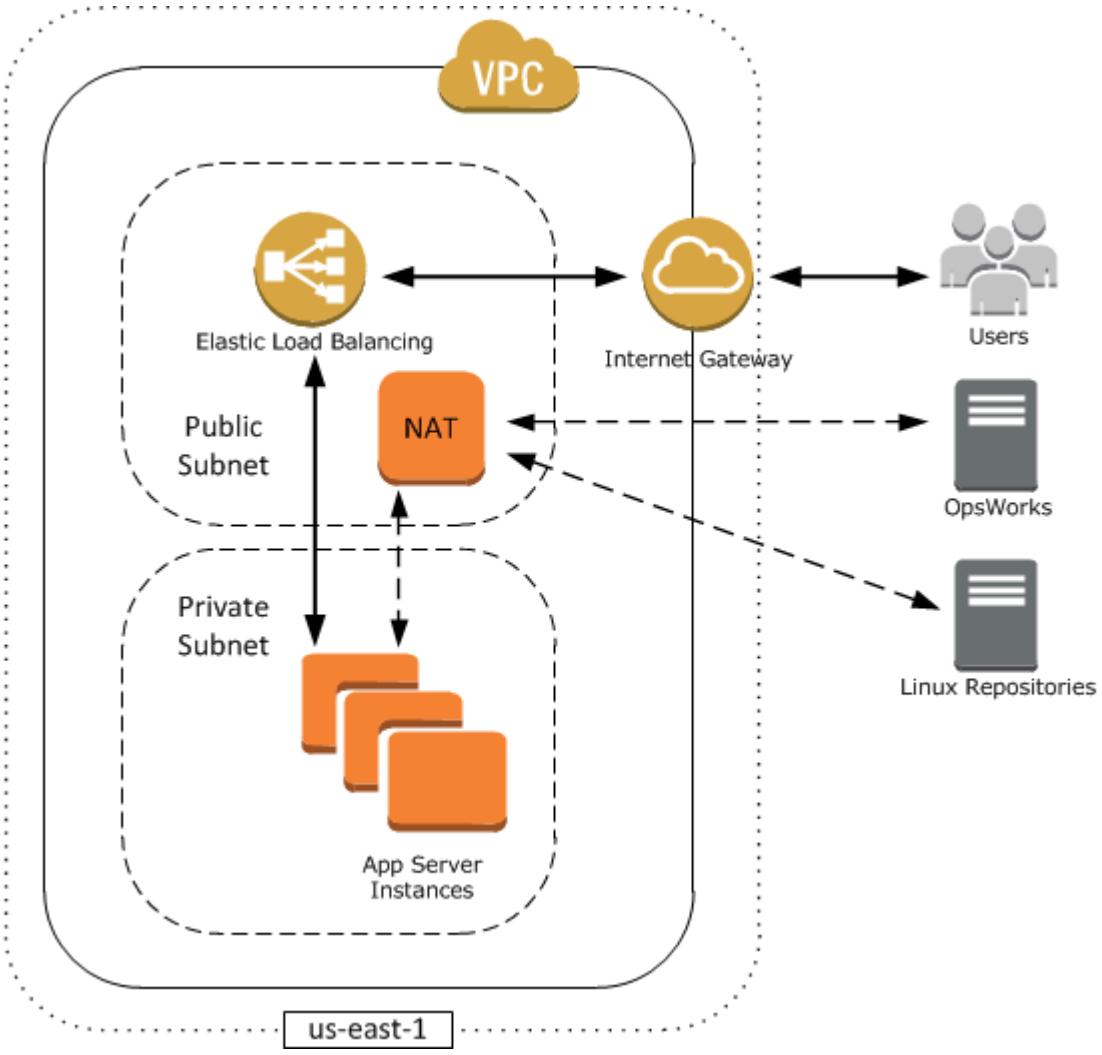
- VPC 내의 인스턴스는 기본적으로 서브넷에 상관없이 서로 통신할 수 있습니다. 하지만 네트워크 ACL(액세스 제어 목록) 또는 보안 그룹 정책을 변경하거나 고정 IP 주소를 사용할 경우 이 통신이 불가능할 수 있습니다.
- 인스턴스가 인터넷과 통신할 수 있는 서브넷을 퍼블릭 서브넷이라고 합니다.
- 인스턴스가 VPC 내의 다른 인스턴스와만 통신할 수 있고 인터넷과 직접 통신할 수 없는 서브넷은 프라이빗 서브넷이라고 합니다.

AWS OpsWorks 프라이빗 서브넷의 인스턴스를 포함하여 스택의 모든 인스턴스가 다음 엔드포인트에 액세스할 수 있도록 Stacks를 구성해야 합니다.

- 의 AWS OpsWorks "리전 지원" 섹션에 나열된 Stacks 서비스 엔드포인트 중 하나입니다AWS OpsWorks Stacks 시작하기.
- AWS OpsWorks Stacks 에이전트에서 사용하는 다음 인스턴스 서비스 엔드포인트 중 하나입니다. 이 에이전트는 관리형 고객 인스턴스에서 실행되면서 서비스와 데이터를 교환합니다.
  - opsworks-instance-service.us-east-2.amazonaws.com
  - opsworks-instance-service.us-east-1.amazonaws.com
  - opsworks-instance-service.us-west-1.amazonaws.com
  - opsworks-instance-service.us-west-2.amazonaws.com
  - opsworks-instance-service.ap-south-1.amazonaws.com
  - opsworks-instance-service.ap-northeast-1.amazonaws.com
  - opsworks-instance-service.ap-northeast-2.amazonaws.com
  - opsworks-instance-service.ap-southeast-1.amazonaws.com
  - opsworks-instance-service.ap-southeast-2.amazonaws.com
  - opsworks-instance-service.ca-central-1.amazonaws.com
  - opsworks-instance-service.eu-central-1.amazonaws.com
  - opsworks-instance-service.eu-west-1.amazonaws.com
  - opsworks-instance-service.eu-west-2.amazonaws.com
  - opsworks-instance-service.eu-west-3.amazonaws.com

- Amazon S3
- Amazon Linux 또는 Ubuntu Linux 리포지토리와 같이 운영 체제가 기반하는 패키지 리포지토리.
- 앱 및 사용자 지정 쿡북 리포지토리.

이 연결을 제공하도록 VPC를 구성하는 방법은 다양합니다. 다음은 AWS OpsWorks Stacks 앱 서버 스택에 대한 VPC를 구성하는 방법의 간단한 예입니다.



이 VPC에는 몇 가지 구성 요소가 있습니다.

서브넷

VPC에는 2개의 서브넷(퍼블릭 서브넷 1개, 프라이빗 서브넷 1개)이 있습니다.

- 퍼블릭 서브넷에는 로드 밸런서와 외부 주소 및 프라이빗 서브넷의 인스턴스와 통신할 수 있는 네트워크 주소 변환(NAT) 디바이스가 포함됩니다.

- 프라이빗 서브넷에는 퍼블릭 서브넷의 NAT 및 로드 밸런서와 통신할 수 있지만 직접 외부 주소와는 통신할 수 없는 애플리케이션 서버가 포함됩니다.

인터넷 게이트웨이

인터넷 게이트웨이는 로드 밸런서처럼 퍼블릭 IP 주소를 가진 인스턴스가 VPC 외부의 주소와 통신할 수 있도록 합니다.

로드 밸런서

Elastic Load Balancing 로드 밸런서는 사용자에게서 오는 트래픽을 받아 프라이빗 서브넷의 앱 서버에 분산시키고 응답을 사용자에게 반환합니다.

NAT

(NAT) 디바이스가 앱 서버에 제공하는 제한적인 인터넷 액세스는 일반적으로 외부 리포지토리에서 소프트웨어 업데이트를 다운로드하는 등의 용도에 사용됩니다. 모든 AWS OpsWorks Stacks 인스턴스는 AWS OpsWorks Stacks 및 적절한 Linux 리포지토리와 통신할 수 있어야 합니다. 이 문제를 처리하는 한 가지 방법은 연결된 탄력적 IP 주소가 있는 NAT 디바이스를 퍼블릭 서브넷에 두는 것입니다. 그러면 NAT을 통해 프라이빗 서브넷의 인스턴스에서 아웃바운드 트래픽을 라우팅할 수 있습니다.

> **Note**
>
> 단일 NAT 인스턴스는 프라이빗 서브넷의 아웃바운드 트래픽에서 단일 장애 지점을 생성합니다. 한 인스턴스에 장애가 발생하면 다른 인스턴스가 대체하는 NAT 인스턴스 쌍으로 VPC를 구성하면 안정성을 높일 수 있습니다. 자세한 내용은 Amazon VPC NAT 인스턴스의 고가용성을 참조하세요. NAT 게이트웨이도 사용할 수 있습니다. 자세한 정보는 Amazon VPC 사용 설명서의 NAT 섹션을 참조하세요.

최적의 VPC 구성은 AWS OpsWorks Stacks 스택에 따라 달라집니다. 다음은 일부 VPC 구성을 사용할 수 있는 몇 가지 예입니다. 다른 VPC 시나리오의 예는 Amazon VPC 시나리오를 참조하세요.

Working with one instance in a public subnet(퍼블릭 서브넷에서 단일 인스턴스로 작업)

공개적으로 액세스할 수 없는 Amazon RDS 인스턴스와 같이 프라이빗 리소스가 연결되어 있지 않은 단일 인스턴스 스택이 있는 경우, 퍼블릭 서브넷 하나로 VPC를 만들고 해당 서브넷에 인스턴스를 배치할 수 있습니다. 기본 VPC를 사용하지 않는 경우, 인스턴스의 계층이 인스턴스에 탄력적 IP 주소를 할당하도록 해야 합니다. 자세한 내용은 OpsWorks 계층 기초 단원을 참조하십시오.

[프라이빗 리소스 사용하기]

공용 액세스가 가능하면 안 되는 리소스가 있는 경우, 퍼블릭 서브넷 1개와 프라이빗 서브넷 1개가 있는 VPC를 생성할 수 있습니다. 예를 들어 로드 밸런싱된 자동 조정 환경에서는 모든 Amazon EC2 인스턴스를 프라이빗 서브넷에 두고 로드 밸런서를 퍼블릭 서브넷에 둘 수 있습니다. 이렇게 하면 Amazon EC2 인스턴스는 인터넷에서 직접 액세스할 수 없으며, 모든 수신 트래픽은 로드 밸런서를 통해 라우팅되어야 합니다.

프라이빗 서브넷은 Amazon EC2 직접 사용자 액세스로부터 인스턴스를 격리시키지만, 여전히 AWS 및 적절한 Linux 패키지 리포지토리로 아웃바운드 요청을 전송해야 합니다. 이러한 요청을 허용하려면 예컨대 네트워크 주소 변환(NAT) 디바이스를 탄력적 IP 주소와 함께 사용한 다음 NAT을 통해 인스턴스의 아웃바운드 트래픽을 라우팅할 수 있습니다. 앞의 예에 나온 것처럼 로드 밸런서와 같은 퍼블릭 서브넷에 NAT을 둘 수 있습니다.

- Amazon RDS 인스턴스와 같은 백엔드 데이터베이스를 사용하는 경우, 이러한 인스턴스를 프라이빗 서브넷에 둘 수 있습니다. Amazon RDS 인스턴스의 경우, 서로 다른 가용 영역에 2개 이상의 서로 다른 서브넷을 지정해야 합니다.
- 프라이빗 서브넷의 인스턴스에 직접 액세스해야 하는 경우(예: SSH를 사용하여 인스턴스에 로그인하려는 경우) 인터넷의 요청을 프록시하는 퍼블릭 서브넷에 Bastion Host를 배치할 수 있습니다.

[AWS로 자체 네트워크 확장하기]

네트워크를 클라우드로 확장하고 VPC에서 직접 인터넷에 액세스하려는 경우, VPN 게이트웨이를 생성할 수 있습니다. 자세한 내용은 시나리오 3: 퍼블릭 및 프라이빗 서브넷이 있고 하드웨어 VPN 액세스를 제공하는 VPC를 참조하세요.

AWS OpsWorks Stacks 스택용 VPC 생성

이 섹션에서는 예제 AWS CloudFormation 템플릿을 사용하여 AWS OpsWorks Stacks 스택에 대한 VPC를 생성하는 방법을 보여줍니다. 템플릿은 OpsWorksVPCtemplates.zip 파일로 다운로드할 수 있습니다. 이 주제에서 설명한 것과 같은 VPC를 수동으로 생성하는 방법에 대한 자세한 내용은 시나리오 2: 퍼블릭 및 프라이빗 서브넷이 있는 VPC를 참조하세요. 라우팅 테이블, 보안 그룹 등등의 구성 방법에 대한 자세한 정보는 예제 템플릿 단원을 참조하세요.

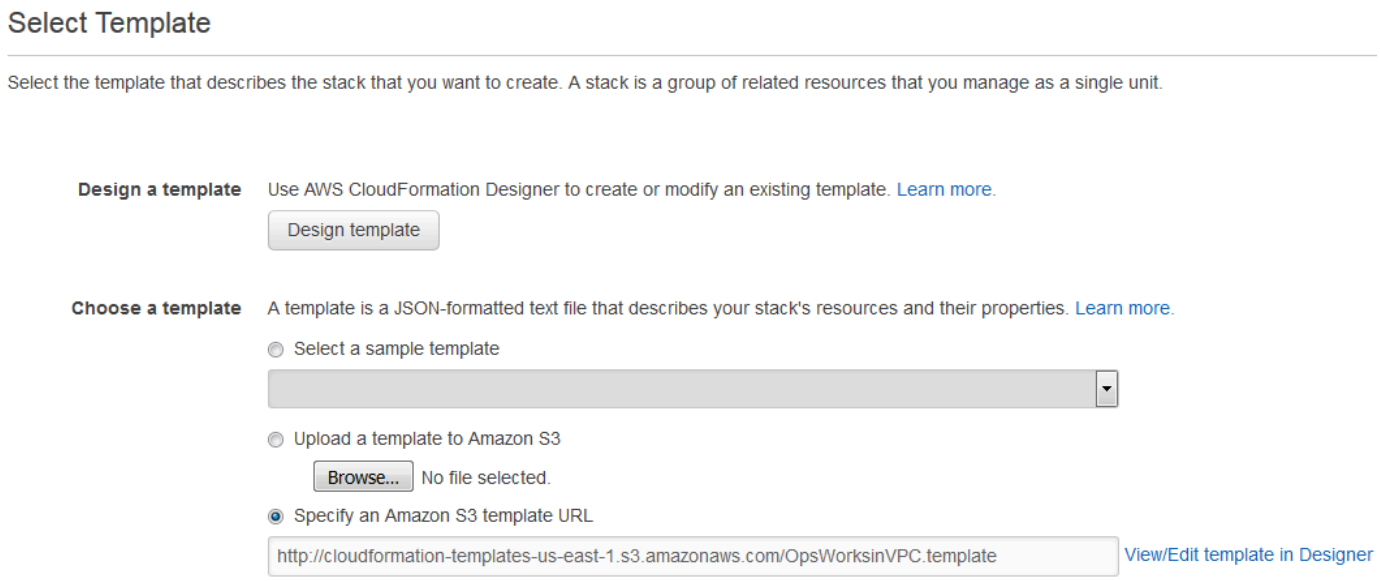
> **Note**
>
> 기본적으로 AWS OpsWorks Stacks는와 같은 CIDR 범위와 가용 영역을 연결하여 서브넷 이름을 표시합니다`10.0.0.1/24 - us-east-1b`. 이름을 더 읽기 쉽게 만들려면 키가 로,

Name 값이 서브넷 이름으로 설정된 각 서브넷에 대한 태그를 생성합니다. 그런 다음 AWS OpsWorks 스택은 서브넷 이름을 기본 이름에 추가합니다. 예를 들어 다음 예제의 프라이빗 서브넷에는 이름이 **Private**으로 설정된 태그가 있으며, OpsWorks는 이를 `10.0.0.1/24 us-east - 1b - Private`으로 표시합니다.

몇 단계만으로 AWS CloudFormation 콘솔을 사용하여 VPC 템플릿을 시작할 수 있습니다. 다음 절차에서는 예제 템플릿을 사용하여 미국 동부(버지니아 북부) 리전에 VPC를 생성합니다. 템플릿을 사용해 다른 리전에서 VPC를 생성하는 방법에 대한 지침은 절차 뒤에 나오는 참고를 참조하세요.

VPC를 생성하려면

1. AWS CloudFormation 콘솔을 열고, 미국 동부(버지니아 북부) 리전을 선택한 다음 스택 생성을 선택합니다.
2. 템플릿 선택 페이지에서 템플릿 업로드를 선택합니다. OpsWorksVPCtemplates.zip 파일에서 다운로드한 `OpsWorksinVPC.template` 파일을 찾아봅니다. 계속을 선택합니다.

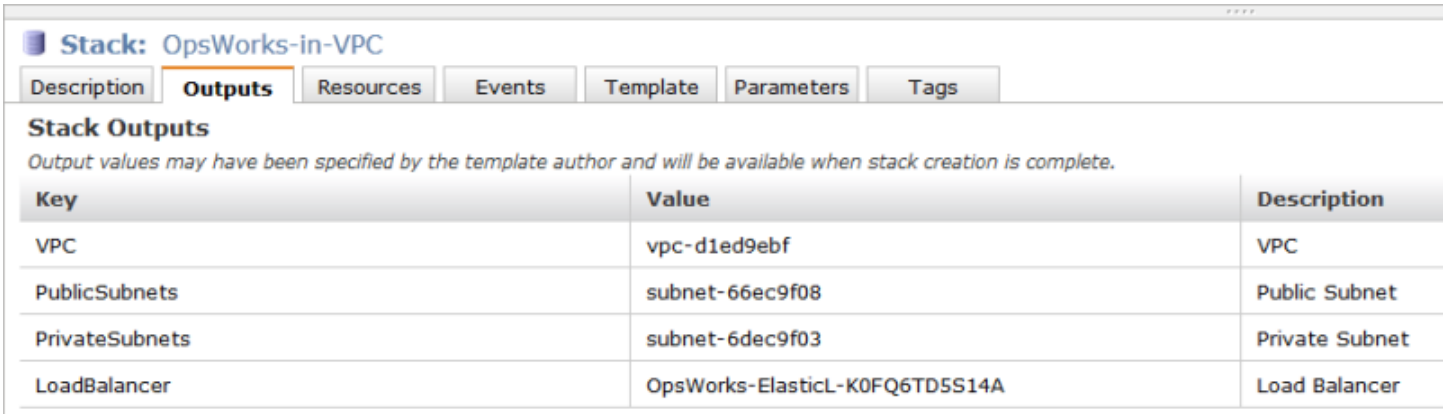


   AWS CloudFormation 샘플 템플릿을 열고 AWS OpsWorks Stacks VPC 템플릿을 찾은 다음 스택 시작을 선택하여이 스택을 시작할 수도 있습니다.
3. 파라미터 지정 페이지에서 기본값을 수락하고 계속을 선택합니다.
4. 태그 추가 페이지에서 키는 **Name**으로, 값은 VPC 이름으로 설정하여 태그를 생성합니다. 이 태그를 사용하면 AWS OpsWorks Stacks 스택을 생성할 때 VPC를 더 쉽게 식별할 수 있습니다.
5. 계속 및 닫기를 차례로 선택하여 스택을 시작합니다.

참고: 다음 방법 중 하나를 사용하여 다른 리전에서 VPC를 생성할 수 있습니다.

- 다른 리전에서 템플릿 사용으로 이동하여 적절한 리전을 선택하고 AWS OpsWorks Stacks VPC 템플릿을 찾은 다음 스택 시작을 선택합니다.
- 템플릿을 시스템에 복사하고 AWS CloudFormation 콘솔에서 적절한 리전을 선택한 다음 스택 생성 마법사의 Amazon S3에 템플릿 업로드 옵션을 사용해 시스템에서 템플릿을 업로드합니다.

예제 템플릿에는 Stacks 스택을 생성하는 데 필요한 VPC, 서브넷 및 로드 AWS OpsWorks 밸런서 IDs를 제공하는 출력이 포함되어 있습니다. AWS CloudFormation 콘솔 창 하단의 출력 탭을 선택하여 볼 수 있습니다.



스택 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택을 생성한 후 언제라도 구성을 업데이트할 수 있습니다. [스택] 페이지에서 [스택 설정], [편집]을 차례로 클릭합니다. 그러면 [설정] 페이지가 표시됩니다. 원하는 설정을 변경한 후 [저장]을 클릭합니다.

설정은 새 스택 생성 섹션에서 설명한 설정과 동일합니다. 세부 정보는 이 항목 단원을 참조하세요. 다만 다음을 참고하세요.

- 리전 또는 VPC ID는 수정할 수 없습니다.

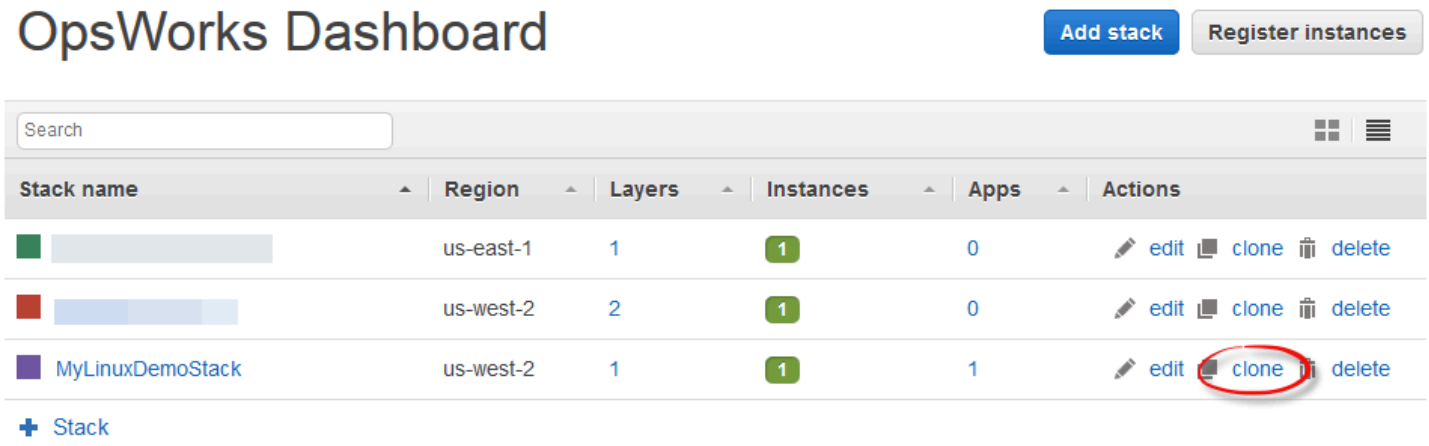
- 스택이 VPC에서 실행 중인 경우 설정에는 VPC의 서브넷을 나열하는 [기본 서브넷] 설정이 포함됩니다. 스택이 VPC에서 실행 중이지 않을 경우 이 설정은 [기본 가용 영역]으로 라벨링되고 리전의 가용 영역을 나열합니다.
- 기본 운영 체제를 변경할 수 있지만, Windows 스택에 Linux 운영 체제를 지정하거나 Linux 스택에 Windows를 지정할 수는 없습니다.
- [호스트 이름 테마] 또는 [기본 SSH 키] 같은 기본 인스턴스 설정을 변경할 경우 새 값은 기존 인스턴스에는 적용되지 않고 새 인스턴스에만 적용됩니다.
- 이름을 변경해도 콘솔에 표시되는 이름은 변경되며, AWS OpsWorks Stacks가 스택을 식별하는 데 사용하는 기본 짧은 이름은 변경되지 않습니다.
- [OpsWorks 보안 그룹 사용]를 [예]에서 [아니요]로 변경하려면 각 계층에 계층의 내장 보안 그룹 이외에 하나 이상의 보안 그룹이 있어야 합니다. 자세한 내용은 OpsWorks 계층의 구성 편집 단원을 참조하십시오.

  AWS OpsWorks 그런 다음 Stacks는 모든 계층에서 기본 제공 보안 그룹을 삭제합니다.
- OpsWorks 보안 그룹 사용을 아니요에서 예로 변경하면 AWS OpsWorks Stacks는 각 계층에 적절한 기본 제공 보안 그룹을 추가하지만 기존 보안 그룹은 삭제하지 않습니다.

스택 복제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 사본을 여러 개 만들면 유용할 경우가 있습니다. 예를 들어 재해 복구나 예방 조치로 중복성을 추가하거나 기존 스택을 새 스택의 시작점으로 사용할 수 있습니다. 가장 간단한 방법은 원본 스택을 복제하는 것입니다. AWS OpsWorks 스택 대시보드의 복제하려는 스택 행의 작업 열에서 복제를 선택하면 스택 복제 페이지가 열립니다.



처음에는 스택 이름에 "copy"라는 단어가 추가된다는 점만 제외하면 복제된 스택의 설정이 원본 스택의 설정과 동일합니다. 이 설정에 대한 내용은 새 스택 생성을 참조하세요. 두 가지 추가 옵션 설정도 있습니다.

권한

[모든 권한]이 선택된 경우(기본값), 원본 스택 권한이 복제된 스택에 적용됩니다.

앱

원본 스택에 배포된 앱을 나열합니다. 나열된 각각의 앱은 해당 확인란이 선택된 경우(기본값), 복제된 스택에 배포됩니다.

> **Note**
>
> 한 리전 엔드포인트에서 다른 엔드포인트로 스택을 복제할 수 없습니다. 예를 들어 미국 서부(오레곤) 리전(us-west-2)에서 아시아 태평양(뭄바이) 리전(ap-south-1)으로 스택을 복제할 수 없습니다.

설정을 완료했으면 스택 복제를 선택합니다. AWS OpsWorks 스택은 소스 스택의 계층과 선택적으로 앱 및 권한으로 구성된 새 스택을 생성합니다. 계층의 구성은 원본과 동일하며, 모든 수정이 적용됩니다. 다만 복제는 인스턴스를 생성하지 않습니다. 복제된 스택의 각 계층에 적절한 인스턴스 세트를 추가한 다음 인스턴스를 시작해야 합니다. 모든 스택과 마찬가지로 복제된 스택에서도 계층 추가, 삭제 또는 수정이나 앱 추가 및 배포 같은 일반적인 관리 작업을 수행할 수 있습니다.

복제된 스택을 작동하려면 인스턴스를 시작합니다. AWS OpsWorks 스택은 계층 멤버십에 따라 각 인스턴스를 설정하고 구성합니다. 또 새 스택에서와 마찬가지로 애플리케이션도 배포합니다.

AWS OpsWorks Stacks Stack 명령 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 스택 인스턴스에서 다양한 작업을 수행하는 데 사용할 수 있는 스택 명령 세트를 제공합니다. 스택 명령을 실행하려면 스택 페이지에서 명령 실행을 클릭합니다. 그런 다음 적절한 명령을 선택하고, 옵션을 지정하고, 오른쪽 하단의 버튼을 누릅니다. 그러면 명령의 이름이 버튼의 레이블로 지정됩니다.

> **Note**
>
> AWS OpsWorks Stacks는 앱 배포를 관리하는 데 사용하는 배포 명령 세트도 지원합니다. 자세한 내용은 앱 배포 단원을 참조하십시오.

다음 스택 명령은 모든 스택에서 실행할 수 있습니다.

사용자 지정 쿡북 업데이트

인스턴스의 사용자 지정 쿡북을 리포지토리의 현재 버전으로 업데이트합니다. 이 명령은 레시피는 실행하지 않습니다. 업데이트된 레시피를 실행하려면 `Execute Recipes`, `Setup`, 또는 `Configure` 스택 명령을 사용하거나 애플리케이션을 재배포하여 Deploy 레시피를 실행할 수 있습니다. 사용자 지정 쿡북에 대한 자세한 정보는 쿡북과 레시피 단원을 참조하세요.

레시피 실행

인스턴스에서 지정된 레시피 세트를 실행합니다. 자세한 내용은 수동으로 레시피 실행 단원을 참조하십시오.

설정

인스턴스의 설정 레시피를 실행합니다.

구성

인스턴스의 Configure 레시피를 실행합니다.

> **Note**
>
> [설정] 또는 [구성]을 사용하여 인스턴스에서 레시피를 실행하려면 해당 레시피가 인스턴스 계층의 해당 수명 주기 이벤트에 할당되어야 합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.

다음 스택 명령은 Linux 기반 스택에서만 실행할 수 있습니다.

Install Dependencies

인스턴스의 패키지를 설치합니다. Chef 12부터는 이 명령을 사용할 수 없습니다.

Update Dependencies

(리눅스만 해당. Chef 12부터는 이 명령을 사용할 수 없습니다.) 정기적인 운영 체제 업데이트 및 패키지 업데이트를 설치합니다. 세부 내용은 인스턴스의 운영 체제에 따라 다릅니다. 자세한 내용은 보안 업데이트 관리 단원을 참조하십시오.

인스턴스를 새 Amazon Linux 버전으로 업그레이드하려면 운영 체제 업그레이드 명령을 사용합니다.

Upgrade Operating System

(Linux 전용) 인스턴스의 Amazon Linux 운영 체제를 최신 버전으로 업그레이드합니다. 자세한 내용은 AWS OpsWorks 스택 운영 체제 단원을 참조하십시오.

> **Important**
>
> 운영 체제 업그레이드를 실행한 후 설정도 실행하는 것이 좋습니다. 그러면 서비스가 올바로 재시작됩니다.

스택 명령에는 다음과 같은 옵션이 있으며, 일부는 특정 명령에서만 표시됩니다.

설명

(선택 사항) 추가하려는 사용자 지정 설명을 입력합니다.

실행할 레시피

(필수 사항) 이 설정은 [레시피 실행] 명령을 선택하는 경우에만 표시됩니다. 표준 *cookbook_name*::*recipe_name* 형식을 사용하여 실행할 레시피를 쉼표로 구분하여 입력합니다. 여러 레시피를 지정하는 경우 AWS OpsWorks Stacks는 나열된 순서로 레시피를 실행합니다.

Allow reboot

(선택 사항) 이 설정은 [운영 체제 업그레이드] 명령을 선택하는 경우에만 표시됩니다. 기본값은 예이며, 업그레이드를 설치한 후 인스턴스를 재부팅하도록 AWS OpsWorks Stacks에 지시합니다.

사용자 지정 Chef JSON

(선택 사항) [고급]을 선택하여 이 옵션을 표시합니다. 그러면 스택 구성 및 배포 속성에 통합할 사용자 지정 JSON 속성을 지정할 수 있습니다.

인스턴스

(선택 사항) 명령이 실행될 인스턴스를 지정합니다. 모든 온라인 인스턴스가 기본적으로 선택됩니다. 인스턴스의 하위 집합에서 명령을 실행하려면 해당 계층 또는 인스턴스를 선택합니다.

> **Note**
>
> 실행하지 않은 execute_recipes 실행이 [배포] 및 [명령] 페이지에 나열되어 있을 수 있습니다. 일반적으로 이는 사용자에 대한 SSH 권한 부여 또는 제거와 같은 권한 변경의 결과입니다. 이러한 변경을 수행하면 AWS OpsWorks Stacks는 execute_recipes를 사용하여 인스턴스에 대한 권한을 업데이트합니다.

사용자 지정 JSON 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

여러 AWS OpsWorks Stacks 작업을 사용하면 AWS OpsWorks Stacks가 인스턴스에 설치하고 레시피에서 사용할 수 있는 사용자 지정 JSON을 지정할 수 있습니다.

다음과 같은 상황에서 사용자 지정 JSON을 지정할 수 있습니다.

- 스택을 생성, 업데이트 또는 복제하는 경우

  AWS OpsWorks Stacks이 이후의 모든 수명 주기 이벤트에서 모든 인스턴스에 사용자 지정 JSON을 설치합니다.
- 배포 또는 스택 명령을 실행하는 경우

  AWS OpsWorks Stacks가 해당 이벤트에서만 사용자 지정 JSON을 인스턴스로 전달합니다.

사용자 지정 JSON은 유효한 JSON 객체로 표현되고 형식 지정되어야 합니다. 예시:

```
{
  "att1": "value1",
  "att2": "value2"
  ...
}
```

AWS OpsWorks Stacks는 다음 위치에 사용자 지정 JSON을 저장합니다.

Linux 인스턴스:

- `/var/chef/runs/`*`run-ID`*`/attribs.json`
- `/var/chef/runs/`*`run-ID`*`/nodes/`*`hostname`*`.json`

Windows 인스턴스:

- *`drive`*`:\chef\runs\`*`run-ID`*`\attribs.json`
- *`drive`*`:\chef\runs\`*`run-ID`*`\nodes\`*`hostname`*`.json`

> **Note**
>
> Linux용 Chef 11.10 및 이전 버전에서는 사용자 지정 JSON이 Linux 인스턴스의 다음 경로에 위치합니다. Windows 인스턴스는 사용할 수 없으며 `attribs.json` 파일이 없습니다. 로그는 JSON과 동일한 폴더 또는 디렉터리에 저장됩니다. Linux용 Chef 11.10 및 이전 버전의 사용자 지정 JSON에 대한 자세한 정보는 사용자 지정 JSON을 사용한 속성 재정의 및 Chef 로그 단원을 참조하세요.

```
/var/lib/aws/opsworks/chef/hostname.json
```

위 경로에서 *run-ID*는 AWS OpsWorks Stacks가 인스턴스의 각 Chef 실행에 할당하는 고유한 ID이고, *hostname*은 인스턴스의 호스트 이름입니다.

Chef 레시피에서 사용자 지정 JSON에 액세스하려면 표준 Chef node 구문을 사용합니다.

예를 들어 배포하려는 앱에서 앱이 초기에 표시 가능한지 여부와 앱의 초기 전경색 및 배경색과 같은 간단한 설정을 정의하려고 합니다. 다음과 같이 JSON 객체를 사용하여 이러한 앱 설정을 정의하는 것으로 가정합니다.

```
{
  "state": "visible",
  "colors": {
    "foreground": "light-blue",
    "background": "dark-gray"
  }
}
```

스택에 대해 사용자 지정 JSON을 선언하는 방법:

1. 스택 페이지에서 [스택 설정]을 클릭한 다음 [편집]을 클릭합니다.
2. [사용자 지정 Chef JSON]의 경우 JSON 객체를 입력한 다음 [저장]을 선택합니다.

> **Note**
>
> 배포, 계층 및 스택 수준에서 사용자 지정 JSON을 선언할 수 있습니다. 일부 사용자 지정 JSON이 개별 배포 또는 계층에만 보이도록 하려면 이렇게 하는 것이 좋습니다. 또는 예를 들어 계층 수준에서 선언된 사용자 지정 JSON으로 스택 수준에서 선언된 사용자 지정 JSON을 일시적으로 재정의할 수도 있습니다. 여러 수준에서 사용자 지정 JSON을 선언할 경우 배포 수준에서 선언된 사용자 지정 JSON이 계층 및 스택 수준에서 선언된 사용자 지정 JSON을 모두 재정의합니다. 계층 수준에서 선언된 사용자 지정 JSON은 스택 수준에서 선언된 사용자 지정 JSON만 재정의합니다.
>
> AWS OpsWorks Stacks 콘솔을 사용하여 배포에 대한 사용자 지정 JSON을 지정하려면 앱 배포 페이지에서 고급을 선택합니다. [사용자 지정 Chef JSON] 상자에 사용자 지정 JSON을 입력한 다음 [저장]을 선택합니다.

AWS OpsWorks Stacks 콘솔을 사용하여 계층에 대한 사용자 지정 JSON을 지정하려면 계층 페이지에서 원하는 계층의 설정을 선택합니다. [사용자 지정 JSON] 상자에 사용자 지정 JSON을 입력한 다음 [저장]을 선택합니다.
자세한 내용은 OpsWorks 계층의 구성 편집 및 앱 배포 섹션을 참조하세요.

배포 또는 스택 명령을 실행하는 경우 레시피가 표준 Chef node 구문을 사용하여 이러한 사용자 지정 값을 검색할 수 있습니다. 이는 사용자 지정 JSON 객체 내 계층 구조와 직접 매핑됩니다. 예를 들어 다음 레시피 코드는 Chef 로그에 이전의 사용자 지정 JSON 값에 대한 메시지를 기록합니다.

```
Chef::Log.info("********** The app's initial state is '#{node['state']}' **********")
Chef::Log.info("********** The app's initial foreground color is '#{node['colors']
['foreground']}' **********")
Chef::Log.info("********** The app's initial background color is '#{node['colors']
['background']}' **********")
```

이 접근 방식은 레시피에 데이터를 전달하는 데 유용할 수 있습니다. AWS OpsWorks Stacks는 해당 데이터를 인스턴스에 추가하고 레시피는 표준 Chef node 구문을 사용하여 데이터를 검색할 수 있습니다.

Note

사용자 지정 JSON은 120KB로 제한됩니다. 용량이 더 필요할 경우 일부 데이터를 Amazon Simple Storage Service(S3)에 저장하는 것이 좋습니다. 그러면 사용자 지정 레시피가 AWS CLI 또는 AWS SDK for Ruby을(를) 사용하여 데이터를 Amazon S3 버킷에서 인스턴스로 다운로드할 수 있습니다.

스택 삭제

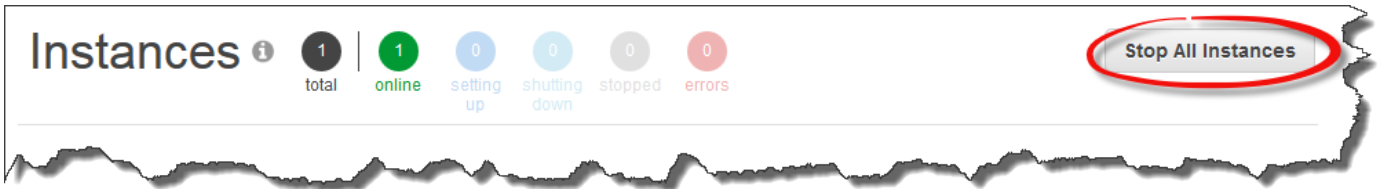
Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

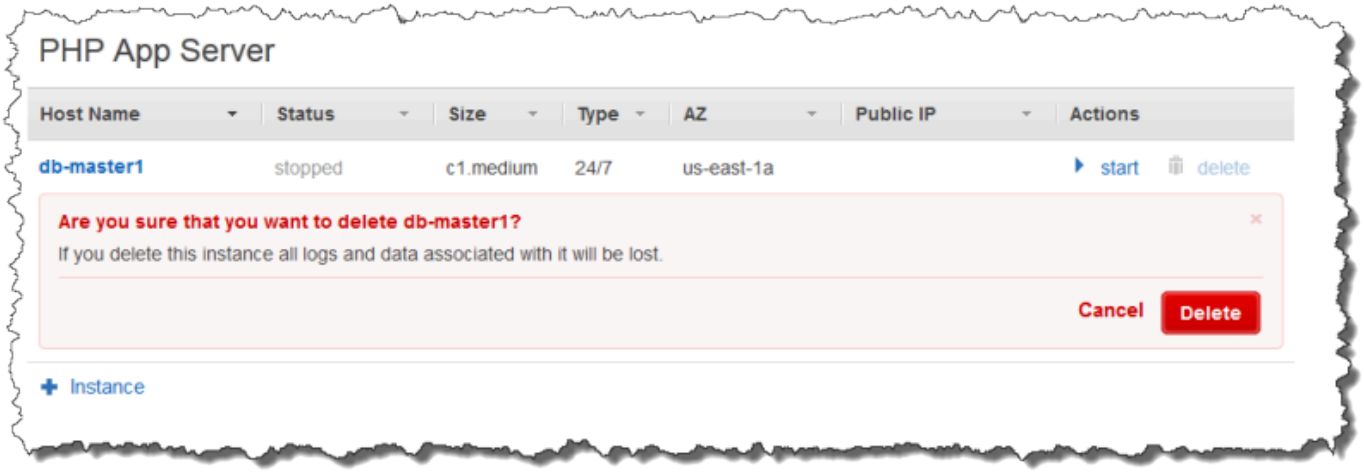
스택이 더 이상 필요하지 않으면 삭제할 수 있습니다. 빈 스택만 삭제할 수 있습니다. 먼저 스택에 있는 인스턴스, 앱과 계층을 모두 삭제하세요.

스택을 삭제하려면

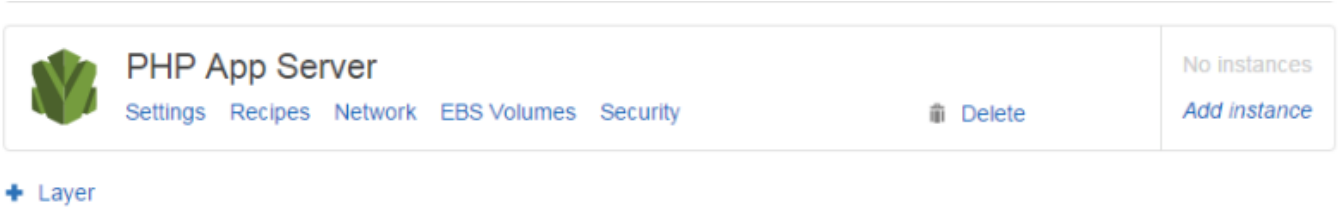
1. AWS OpsWorks 스택 대시보드에서 종료하고 삭제할 스택을 선택합니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. 인스턴스 페이지에서 모든 인스턴스 중지를 선택합니다.



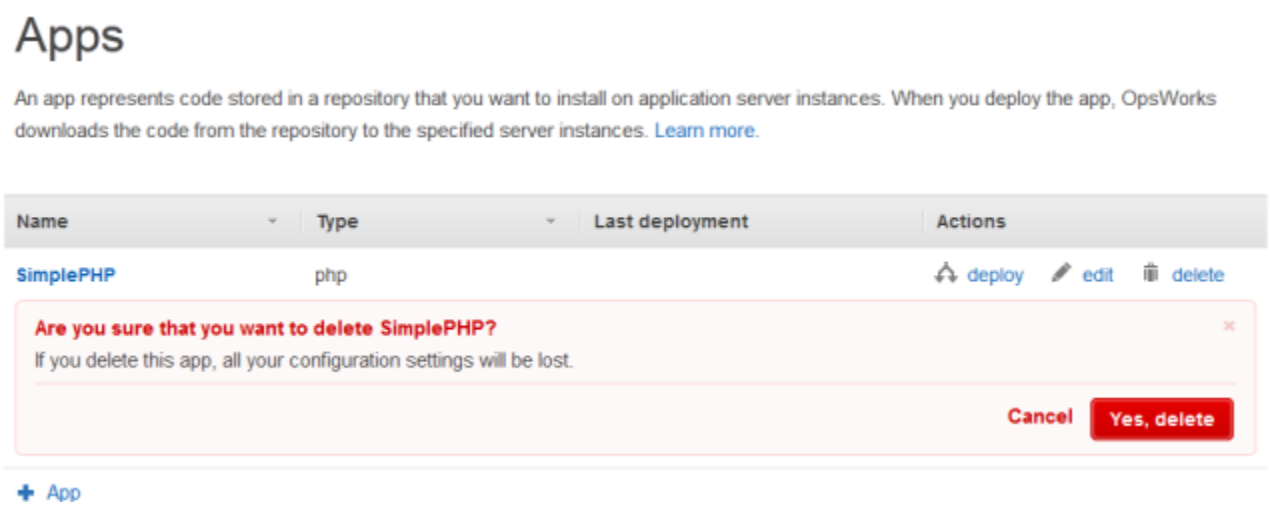
4. 인스턴스가 중지되면 계층의 각 인스턴스에 대해 작업 열에서 삭제를 선택합니다. 확인하라는 메시지가 나타나면 예, 삭제합니다.를 선택합니다.



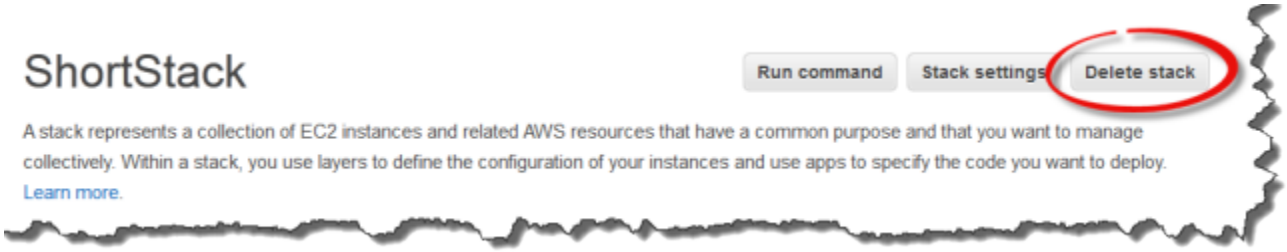
5. 모든 인스턴스가 삭제되면 탐색 창에서 계층을 선택합니다.
6. 계층 페이지에서 스택의 각 계층에 대해 삭제를 선택합니다. 확인 프롬프트에서 예, 삭제합니다.를 선택합니다.



7. 모든 계층이 삭제되면 탐색 창에서 앱을 선택합니다.
8. 앱 페이지의 스택에 있는 각 앱에 대해 작업 열에서 삭제를 선택합니다. 확인 프롬프트에서 예, 삭제합니다.를 선택합니다.



9. 모든 앱이 삭제되면 탐색 창에서 스택을 선택합니다.
10. 스택 페이지에서 스택 삭제를 선택합니다. 확인 프롬프트에서 예, 삭제합니다.를 선택합니다.



스택에서 사용하는 다른 AWS 리소스 삭제

AWS OpsWorks Stacks에서 다른 AWS 리소스를 사용하여 스택을 생성하고 관리합니다. 스택을 삭제할 때 다른 스택이 스택을 사용하지 않고 AWS OpsWorks Stacks 외부의 리소스가 스택을 사용하지 않는 경우 스택으로 작업한 리소스도 삭제하는 것이 좋습니다. 다음은 스택과 함께 사용한 외부 AWS 리소스를 정리하는 권장 이유입니다.

- 외부 AWS 리소스는 계정에 계속 요금이 발생할 수 있습니다 AWS .
- Amazon S3 버킷 같은 리소스가 개인 식별 정보, 민감한 정보나 기밀 정보를 보유할 수 있습니다.

> **Important**
>
> 다른 스택이 이 리소스를 사용한다면 리소스를 삭제하면 안 됩니다. IAM 역할과 보안 그룹은 전역이며, 따라서 다른 리전에 있는 스택이 같은 리소스를 사용할 수도 있습니다.

다음은 스택이 사용하는 다른 AWS 리소스와 이를 삭제하는 방법에 대한 정보 링크입니다.

서비스 역할 및 인스턴스 프로파일

스택을 생성할 때 Stacks가 사용자를 대신하여 허용된 리소스를 생성하는 데 사용하는 IAM 역할과 인스턴스 프로파일을 지정합니다. AWS OpsWorks 기존 역할을 선택하지 않으면 AWS OpsWorks 가 역할 및 인스턴스 프로파일을 생성합니다. 에서 AWS OpsWorks 자동으로 생성하는 역할 및 인스턴스 프로파일의 이름은 `aws-opsworks-ec2-role`각각 `aws-opsworks-service-role` 및 입니다. 계정 내 다른 스택이 IAM 역할과 인스턴스 프로파일을 사용하지 않는다면, 이 리소스를 삭제해도 됩니다. IAM 역할 및 인스턴스 프로파일 삭제 방법에 대한 내용은 IAM 사용 설명서의 역할 또는 인스턴스 프로파일 삭제를 참조하세요.

보안 그룹

AWS OpsWorks Stacks에서는 계층 수준에서 사용자 정의 보안 그룹을 지정할 수 있습니다. Amazon EC2 콘솔이나 API를 사용해 보안 그룹을 만듭니다. 다른 리전에 있는 스택과 계층도 같은 보안 그룹을 사용할 수 있는데, 보안 그룹이 전역이기 때문입니다. 다른 AWS 리소스에서 사용하지 않는 보안 그룹은 삭제할 수 있습니다. 보안 그룹을 삭제하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 보안 그룹 삭제를 참조하세요.

Amazon EBS 볼륨

AWS OpsWorks Stacks에서는 계층 수준에서 EBS 볼륨을 추가하면 해당 볼륨이 계층의 인스턴스에 연결됩니다. Amazon EC2 서비스 콘솔 또는 API를 사용하여 EBS 볼륨을 생성한 다음 계층 수준에서 AWS OpsWorks Stacks 인스턴스에 연결합니다. EBS 볼륨은 가용 영역 전용입니다. 특정 리전과 가용 영역의 어떤 스택에서도 EBS 볼륨을 사용하지 않는다면, 해당 볼륨을 삭제할 수 있습니다. Amazon EBS 볼륨을 삭제하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EBS 볼륨 삭제를 참조하세요.

Amazon Simple Storage Service(Amazon S3) 버킷

AWS OpsWorks Stacks에서는 다음에 Amazon S3 버킷을 사용할 수 있습니다. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

- 앱 코드 저장
- 쿡북 및 레시피 저장
- AWS OpsWorks Stacks에서 CloudTrail 로깅을 활성화한 경우 CloudTrail 로그
- Amazon CloudWatch Logs 스트림을 AWS OpsWorks Stacks에서 활성화한 경우

탄력적 IP 주소

AWS OpsWorks Stacks에 탄력적 IP 주소를 등록했는데 탄력적 IP 주소가 더 이상 필요하지 않은 경우 탄력적 IP 주소를 해제할 수 있습니다.

Elastic Load Balancing 로드 밸런서

스택의 계층과 함께 사용한 Elastic Load Balancing Classic Load Balancer가 더 이상 필요하지 않다면 삭제해도 됩니다. 자세한 정보는 Classic Load Balancer 사용 설명서의 로드 밸런서 삭제를 참조하세요.

Amazon Relational Database Service(RDS) 인스턴스

Amazon RDS 데이터베이스(DB) 인스턴스를 AWS OpsWorks Stacks에 등록했는데 더 이상 필요하지 않은 경우 DB 인스턴스를 삭제할 수 있습니다. DB 인스턴스를 삭제하는 방법에 대한 자세한 내용은 Amazon RDS 사용 설명서의 DB 인스턴스 삭제를 참조하세요.

Amazon Elastic Container Service(Amazon ECS) 클러스터

스택에 ECS 클러스터 계층이 포함되어 있고 계층에 등록된 ECS 클러스터를 더 이상 사용하지 않는 경우, ECS 클러스터를 삭제할 수 있습니다. ECS 클러스터를 삭제하는 방법에 대한 자세한 내용은 Amazon ECS 개발자 안내서의 클러스터 삭제를 참조하세요.

계층

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

모든 스택에는 각각 로드 밸런서 또는 애플리케이션 서버 세트와 같은 스택 구성 요소를 나타내는 하나 이상의 계층이 포함됩니다.

AWS OpsWorks Stacks 계층을 사용할 때는 다음 사항에 유의하세요.

- 스택의 각 계층에는 하나 이상의 인스턴스가 있어야 하며, 필요하다면 여러 개의 인스턴스가 있을 수 있습니다.

- 등록된 인스턴스를 제외하고 스택의 각 인스턴스는 하나 이상의 계층에 속해야 합니다.

  SSH 키와 호스트 이름 같은 일부 기본적 설정을 제외하면 인스턴스를 직접 구성할 수 없습니다. 적절한 계층을 생성하고 구성한 다음 인스턴스를 계층에 추가해야 합니다.

Amazon EC2 인스턴스는 필요할 경우 여러 계층에 속할 수 있습니다. 이 경우 AWS OpsWorks Stacks는 레시피를 실행하여 인스턴스의 각 계층에 대해 패키지를 설치 및 구성하고 애플리케이션을 배포하는 등의 작업을 수행합니다.

인스턴스를 여러 계층에 할당하면 예컨대 다음이 가능합니다.

- 데이터베이스 서버와 로드 밸런서를 단일 인스턴스에서 호스팅함으로써 비용을 절감합니다.
- 애플리케이션 서버 중 하나를 관리에 사용합니다.

  사용자 지정 관리 계층을 생성하고 애플리케이션 서버 인스턴스 중 하나를 이 계층에 추가합니다. 관리 계층의 레시피는 해당 애플리케이션 서버 인스턴스가 관리 작업을 수행하도록 구성하고 필요한 추가 소프트웨어를 설치합니다. 다른 애플리케이션 서버 인스턴스는 애플리케이션 서버일 뿐입니다.

이 섹션에서는 계층 작업 방법에 대해 설명합니다.

주제

- OpsWorks 계층 기초
- Elastic Load Balancing 계층
- Amazon RDS 서비스 계층
- ECS 클러스터 계층
- 사용자 지정 AWS OpsWorks 스택 계층
- 계층별 운영 체제 패키지 설치

OpsWorks 계층 기초

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 모든 AWS OpsWorks Stacks 계층에 공통적인 작업을 수행하는 방법을 설명합니다.

주제

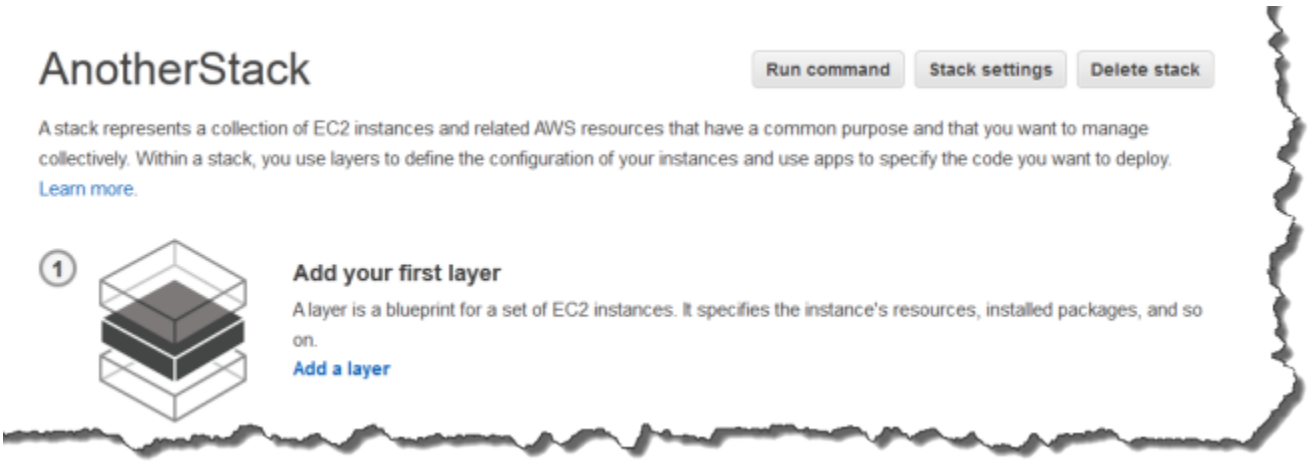
- OpsWorks 계층 생성
- OpsWorks 계층의 구성 편집
- 자동 복구를 사용하여 실패한 인스턴스 대체
- OpsWorks 계층 삭제

OpsWorks 계층 생성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

새 스택을 만들 때는 다음 페이지를 참조하세요.



첫 번째 OpsWorks 계층을 추가하려면

1. [계층 추가]를 클릭합니다.

2. [계층 추가] 페이지에서 해당 계층을 선택합니다. 그러면 계층의 구성 옵션이 표시됩니다.
3. 계층을 적절하게 구성하고 [계층 추가]를 클릭하여 스택에 추가합니다. 다음 단원에서는 다양한 계층을 구성하는 방법에 대해 설명합니다.

> **Note**
>
> [계층 추가] 페이지에는 각 계층에 대해 일반적으로 사용되는 구성 설정만 표시됩니다. 계층을 편집하여 추가 설정을 지정할 수 있습니다.

4. 인스턴스를 계층에 추가하고 시작합니다.

> **Note**
>
> 인스턴스가 여러 계층에 속한 경우 인스턴스를 시작하기 전에 모든 계층에 인스턴스를 추가해야 합니다. 계층에는 온라인 인스턴스를 추가할 수 없습니다.

더 많은 계층을 추가하려면 [계층] 페이지를 열고 [+ 계층]을 클릭하여 [계층 추가] 페이지를 엽니다.

인스턴스를 시작하면 AWS OpsWorks Stacks는 인스턴스의 각 계층에 대한 설정 및 배포 레시피를 자동으로 실행하여 적절한 패키지를 설치 및 구성하고 적절한 애플리케이션을 배포합니다. 적절한 수명 주기 이벤트에 사용자 지정 레시피를 할당하는 등 다양한 방법으로 계층의 설정 및 구성 프로세스를 사용자 지정할 수 있습니다. AWS OpsWorks Stacks는 각 이벤트에 대한 표준 레시피 이후에 사용자 지정 레시피를 실행합니다. 자세한 내용은 쿡북과 레시피 단원을 참조하십시오.

다음 계층별 섹션에서는가 다양한 AWS OpsWorks Stacks 계층에 대해 2단계와 3단계를 처리하는 방법을 설명합니다. 인스턴스를 추가하는 방법에 대한 자세한 정보는 계층에 인스턴스 추가를 참조하세요.

OpsWorks 계층의 구성 편집

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계층을 생성한 후에 일부 속성(AWS 리전 등)은 변경이 불가능하지만 대부분의 계층 구성은 언제든 변경할 수 있습니다. 계층을 편집하면 [계층 추가] 페이지에서 사용할 수 없는 구성 설정에도 액세스할 수 있습니다. 설정은 새 구성을 저장하는 즉시 적용됩니다.

OpsWorks 계층을 편집하려면

1. 탐색 창에서 [계층]을 클릭합니다.
2. [계층] 페이지에서 계층 이름을 선택하여 세부 정보 페이지를 열면 현재 구성이 표시됩니다.

   > **Note**
   >
   > 계층 이름 아래에서 이름 중 하나를 선택하면 세부 정보 페이지의 연결된 탭으로 이동합니다.

3. [편집]을 클릭한 다음 [일반 설정], [레시피], [네트워크], [EBS 볼륨] 또는 [보안] 중에서 적절한 탭을 선택합니다.

다음 섹션에서는 모든 계층에 사용할 수 있는 다양한 탭에서의 설정에 대해 설명합니다. 일부 계층에는 페이지 상단에 표시되는 추가적인 계층별 설정이 있습니다. 또한 일부 설정은 언급했듯이 Linux 기반 스택에서만 사용할 수 있습니다.

주제

- 일반 설정
- 레시피
- Network
- EBS 볼륨
- 보안
- CloudWatch Logs
- Tags

일반 설정

모든 계층에는 다음과 같은 설정이 있습니다.

자동 복구 활성화

계층의 인스턴스에 대해 자동 복구가 활성화되어 있는지 여부. 기본 설정은 [예]입니다.

사용자 지정 JSON

이 계층의 모든 인스턴스에 대해 Chef 레시피에 전달되는 JSON 형식의 데이터. 예를 들어 이를 사용하여 데이터를 본인의 레시피에 전달할 수 있습니다. 자세한 내용은 사용자 지정 JSON 사용 단원을 참조하십시오.

> **Note**
>
> 배포, 계층 및 스택 수준에서 사용자 지정 JSON을 선언할 수 있습니다. 일부 사용자 지정 JSON이 스택에서 또는 개별 배포에만 보이도록 하려면 이렇게 하는 것이 좋습니다. 또는 예를 들어 배포 수준에서 선언된 사용자 지정 JSON으로 계층 수준에서 선언된 사용자 지정 JSON을 일시적으로 재정의할 수도 있습니다. 여러 수준에서 사용자 지정 JSON을 선언할 경우 배포 수준에서 선언된 사용자 지정 JSON이 계층 및 스택 수준에서 선언된 사용자 지정 JSON을 모두 재정의합니다. 계층 수준에서 선언된 사용자 지정 JSON은 스택 수준에서 선언된 사용자 지정 JSON만 재정의합니다.
>
> AWS OpsWorks Stacks 콘솔을 사용하여 배포에 대한 사용자 지정 JSON을 지정하려면 앱 배포 페이지에서 고급을 선택합니다. [사용자 지정 Chef JSON] 상자에 사용자 지정 JSON을 입력한 다음 [저장]을 선택합니다.
>
> AWS OpsWorks Stacks 콘솔을 사용하여 스택에 대한 사용자 지정 JSON을 지정하려면 스택 설정 페이지에서 사용자 지정 JSON 상자에 사용자 지정 JSON을 입력한 다음 저장을 선택합니다.
>
> 자세한 내용은 사용자 지정 JSON 사용 및 앱 배포 섹션을 참조하세요.

인스턴스 종료 제한 시간

종료 수명 주기 이벤트를 트리거한 후 Amazon EC2 인스턴스를 중지하거나 종료하기 전에 AWS OpsWorks 스택이 대기하는 시간(초)을 지정합니다. 기본 설정은 120초입니다. 이 설정의 목적은 인스턴스의 Shutdown 레시피에게 인스턴스 종료 전에 작업을 완료할 수 있도록 충분한 시간을 주는 것입니다. 사용자 지정 Shutdown 레시피에 시간이 더 필요할 경우, 그에 따라 설정을 수정합니다. 인스턴스 종료에 대한 자세한 정보는 인스턴스 중지 단원을 참조하세요.

이 탭의 나머지 설정은 계층 유형에 따라 다르며, 계층의 [계층 추가] 페이지의 설정과 동일합니다.

레시피

[레시피] 탭에는 다음 설정이 포함되어 있습니다.

[사용자 지정 Chef 레시피]

사용자 지정 Chef 레시피를 계층의 수명 주기 이벤트에 할당할 수 있습니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.

Network

[네트워크] 탭에는 다음 설정이 포함되어 있습니다.

Elastic Load Balancing

Elastic Load Balancing 로드 밸런서를 어떤 계층에도 연결할 수 있습니다. AWS OpsWorks 그런 다음 Stacks는 계층의 온라인 인스턴스를 로드 밸런서에 자동으로 등록하고 오프라인 상태가 될 때 등록을 취소합니다. 로드 밸런서의 연결 드레이닝 기능을 활성화한 경우 AWS OpsWorks Stacks가 이를 지원하는지 여부를 지정할 수 있습니다. 자세한 내용은 Elastic Load Balancing 계층 단원을 참조하십시오.

[자동으로 IP 주소 할당]

AWS OpsWorks Stacks가 계층의 인스턴스에 퍼블릭 또는 탄력적 IP 주소를 자동으로 할당할지 여부를 제어할 수 있습니다. 이 옵션을 활성화하면 다음과 같은 일이 발생합니다.

- 인스턴스 스토어 지원 인스턴스의 경우 AWS OpsWorks Stacks는 인스턴스가 시작될 때마다 주소를 자동으로 할당합니다.
- Amazon EBS 지원 인스턴스의 경우 AWS OpsWorks Stacks는 인스턴스가 처음 시작될 때 주소를 자동으로 할당합니다.
- 인스턴스가 둘 이상의 계층에 속하는 경우 하나 이상의 계층에 대해 자동 할당을 활성화한 경우 AWS OpsWorks Stacks는 주소를 자동으로 할당합니다.

> **Note**
>
> 퍼블릭 IP 주소 자동 할당을 활성화하면 새 인스턴스에만 적용됩니다. AWS OpsWorks Stacks는 기존 인스턴스의 퍼블릭 IP 주소를 업데이트할 수 없습니다.

스택이 VPC에서 실행 중인 경우, 퍼블릭 IP 주소와 탄력적 IP 주소의 설정이 따로 있습니다. 다음 표는 이들이 상호 작용하는 방식을 설명합니다.

Elastic IP addresses \ Public IP addresses	Yes	No
Yes	Instances receive an Elastic IP address when they are started for the first time, or a public IP address if an Elastic IP cannot be assigned.	Instances receive an Elastic IP address when they are started for the first time.
No	Instances receive a public IP address each time they are started.	Instances receive only a private IP address, which is not accessible from outside the VPC.

> **Note**
>
> 인스턴스에는 Stacks 서비스, Linux 패키지 리포지토리 및 AWS OpsWorks 쿡북 리포지토리와 통신할 수 있는 방법이 있어야 합니다. 퍼블릭 또는 탄력적 IP 주소를 지정하지 않는 경우, 계층의 인스턴스가 외부 사이트와 통신할 수 있도록 해 주는 NAT 등의 구성 요소가 VPC에 포함되어 있어야 합니다. 자세한 내용은 VPC에서 스택 실행 단원을 참조하십시오.

스택이 VPC에서 실행되고 있지 않은 경우, [탄력적 IP 주소]가 유일한 설정입니다.

- [예]: 인스턴스는 처음으로 시작될 때 탄력적 IP 주소를 받거나 탄력적 IP 주소를 할당할 수 없는 경우에는 퍼블릭 IP 주소를 받습니다.
- [아니요]: 인스턴스는 시작될 때마다 퍼블릭 IP 주소를 받습니다.

EBS 볼륨

[EBS 볼륨] 탭에는 다음 설정이 포함되어 있습니다.

EBS 최적 인스턴스

계층의 인스턴스를 Amazon Elastic Block Store(Amazon EBS)에 맞게 최적화해야 하는지 여부. 자세한 내용은 Amazon EBS 최적화 인스턴스를 참조하세요.

[추가 EBS 볼륨]

(Linux만 해당)계층의 인스턴스에 Amazon EBS 볼륨을 추가하거나 제거할 수 있습니다. 인스턴스를 시작하면 AWS OpsWorks Stacks가 자동으로 볼륨을 생성하여 인스턴스에 연결합니다. [리소스] 페이지를 사용하여 스택의 EBS 볼륨을 관리할 수 있습니다. 자세한 내용은 리소스 관리 단원을 참조하십시오.

- 탑재 지점 – (필수)EBS 볼륨이 탑재될 탑재 지점 또는 디렉터리를 지정합니다.

- 디스크 수 – (선택 사항)RAID 어레이를 지정한 경우, 어레이에 있는 디스크 수.

  각각의 RAID 레벨에는 기본 디스크 수가 있지만 목록에서 더 큰 수를 선택할 수 있습니다.
- 총 크기(GiB) - (필수)볼륨의 크기(GiB).

  RAID 어레이의 경우, 이 설정은 각 디스크의 크기가 아니라 총 어레이 크기를 지정합니다.

  다음 표에는 각 볼륨 유형에 허용되는 최소 및 최대 볼륨 크기가 나와 있습니다.

볼륨 유형	최소 크기(GiB)	최대 크기(GiB)
마그네틱	1	1024
프로비저닝된 IOPS(SSD)	4	16384
범용(SSD)	1	16384
처리량에 최적화된 HDD	500	16384
콜드 HDD	500	16384

- 볼륨 유형 – (선택 사항) 마그네틱, 범용 SSD, 처리량에 최적화된 HDD, 콜드 HDD 또는 PIOPS 볼륨 중에서 생성할 볼륨을 지정합니다.

  기본값은 [마그네틱]입니다.
- 암호화 - (선택 사항) EBS 볼륨의 콘텐츠를 암호화할지 여부를 지정합니다.
- 디스크당 IOPS – (프로비저닝된 IOPS SSD 및 범용 SSD 볼륨의 경우, 필수) 프로비저닝된 IOPS SSD 또는 범용 SSD 볼륨을 지정하는 경우, 디스크당 IOPS도 지정해야 합니다.

  프로비저닝된 IOPS 볼륨의 경우, 볼륨 생성 시 IOPS 속도를 지정할 수 있습니다. 프로비저닝된 IOPS와 요청한 볼륨 크기의 비율은 최대 30입니다. 다시 말해서, IOPS가 3000인 볼륨은 최소 100GB가 되어야 합니다. 범용(SSD) 볼륨 유형의 기준 IOPS는 볼륨 크기의 3배로 최대 10000 IOPS이며, 30분 동안 3000 IOPS까지 버스트할 수 있습니다.

계층에 볼륨을 추가하거나 계층에서 볼륨을 제거할 때는 다음에 유의하세요.

- 볼륨을 추가하는 경우, 새로운 모든 인스턴스에 새 볼륨이 생기지만 AWS OpsWorks Stacks는 기존 인스턴스를 업데이트하지 않습니다.
- 볼륨을 제거하는 경우에는 새 인스턴스에만 적용되며, 기존 인스턴스는 볼륨을 유지합니다.

탑재 지점 지정

원하는 어떤 탑재 지점도 지정할 수 있습니다. 그러나 일부 탑재 지점은 AWS OpsWorks Stacks 또는 Amazon EC2에서 사용하도록 예약되어 있으므로 Amazon EBS 볼륨에 사용해서는 안 됩니다. `/home` 또는 `/etc` 같은 일반적인 Linux 시스템 폴더를 사용하지 마십시오.

다음 탑재 지점은 AWS OpsWorks Stacks에서 사용하도록 예약되어 있습니다.

- `/srv/www`
- `/var/log/apache2`(Ubuntu)
- `/var/log/httpd`(Amazon Linux)
- `/var/log/mysql`
- `/var/www`

인스턴스가 부팅되거나 재부팅될 때 autofs(자동 탑재 데몬)는 바인드 탑재에 `/media/ephemeral0` 같은 휘발성 디바이스 탑재 지점을 사용합니다. 이 작업은 Amazon EBS 볼륨이 탑재되기 전에 이루어집니다. Amazon EBS 볼륨의 탑재 지점이 autofs와 충돌하지 않도록 하려면 휘발성 디바이스 탑재 지점을 지정하지 마세요. 가능한 임시 디바이스 탑재 지점은 특정 인스턴스 유형과 인스턴스 스토어 지원 또는 Amazon EBS 지원 여부에 따라 다릅니다. autofs와의 충돌을 방지하려면 다음과 같이 합니다.

- 특정 인스턴스 유형의 휘발성 디바이스 탑재 지점과 사용하려는 보조 저장소를 확인합니다.
- Amazon EBS 지원 인스턴스로 전환할 경우, 인스턴스 스토어 지원 인스턴스에서 작동하는 탑재 지점이 autofs와 충돌할 수 있으며 그 반대의 경우도 마찬가지라는 점에 유의하세요.

> **Note**
>
> 인스턴스 스토어 블록 디바이스 매핑을 변경하려는 경우, 사용자 지정 AMI를 생성할 수 있습니다. 자세한 내용은 Amazon EC2 인스턴스 스토어를 참조하세요. AWS OpsWorks Stacks용 사용자 지정 AMI를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 지정 AMI 사용.

다음은 사용자 지정 레시피를 사용하여 볼륨의 탑재 지점이 autofs와 충돌하지 않도록 하는 방법의 예제입니다. 특정 사용 사례의 필요에 따라 이 방법을 조정할 수 있습니다.

탑재 지점 충돌을 피하려면

1. 원하는 계층에 Amazon EBS 볼륨을 할당하되, autofs와 충돌하지 않는 탑재 지점(예: `/mnt/workspace`)을 사용합니다.
2. Amazon EB 볼륨에 애플리케이션 디렉터리를 만들어 `/srv/www/`에서 이 디렉터리에 연결하는 다음 사용자 지정 레시피를 구현합니다. 사용자 지정 레시피를 구현하는 방법에 대한 자세한 정보는 쿡북과 레시피 및 AWS OpsWorks 스택 사용자 지정 단원을 참조하세요.

```
mount_point = node['ebs']['raids']['/dev/md0']['mount_point'] rescue nil

if mount_point
  node[:deploy].each do |application, deploy|
    directory "#{mount_point}/#{application}" do
      owner deploy[:user]
      group deploy[:group]
      mode 0770
      recursive true
    end

    link "/srv/www/#{application}" do
      to "#{mount_point}/#{application}"
    end
  end
end
```

3. 사용자 지정 쿡북의 `metadata.rb` 파일에 `depends 'deploy'` 라인을 추가합니다.
4. 이 레시피를 계층의 설정 이벤트에 할당합니다.

보안

[보안] 탭에는 다음 설정이 포함되어 있습니다.

보안 그룹

계층에는 하나 이상의 연결된 보안 그룹이 있어야 합니다. 스택을 생성하거나 업데이트할 때 보안 그룹을 연결하는 방법을 지정합니다. AWS OpsWorks Stacks는 표준 보안 그룹 세트를 제공합니다.

- 기본 옵션은 AWS OpsWorks Stacks가 적절한 기본 제공 보안 그룹을 각 계층에 자동으로 연결하도록 하는 것입니다.

- 자동으로 내장 보안 그룹을 연결하지 않고 계층을 생성할 때 각 계층에 사용자 지정 보안 그룹을 연결할 수도 있습니다.

보안 그룹에 대한 자세한 정보는 보안 그룹 사용 단원을 참조하세요.

계층이 생성된 후 [보안 그룹]을 사용하여 [사용자 지정 보안 그룹] 목록에서 보안 그룹을 선택하면 더 많은 보안 그룹을 계층에 추가할 수 있습니다. 계층에 보안 그룹을 추가한 후 AWS OpsWorks Stacks는 모든 새 인스턴스에 보안 그룹을 추가합니다. (다시 시작된 인스턴스 스토어 인스턴스는 새 인스턴스로 표시되므로 새 보안 그룹도 갖게 됩니다.) AWS OpsWorks Stacks는 온라인 인스턴스에 보안 그룹을 추가하지 않습니다.

다음과 같이 [x]를 클릭하여 기존 보안 그룹을 삭제할 수 있습니다.

- AWS OpsWorks Stacks가 내장 보안 그룹을 자동으로 연결하도록 선택한 경우 x를 클릭하여 이전에 추가한 사용자 지정 보안 그룹을 삭제할 수 있지만 내장 그룹은 삭제할 수 없습니다.
- 자동으로 내장 보안 그룹을 연결하지 않으려면 계층에 최소 하나의 그룹이 유지되는 한 원래 보안 그룹을 포함하여 기존 보안 그룹을 삭제할 수 있습니다.

계층에서 보안 그룹을 제거한 후 AWS OpsWorks Stacks는 새 인스턴스 또는 다시 시작된 인스턴스에 보안 그룹을 추가하지 않습니다. AWS OpsWorks Stacks는 온라인 인스턴스에서 보안 그룹을 제거하지 않습니다.

> **Note**
>
> 스택이 VPC에서 실행 중인 경우 Amazon EC2 콘솔, API 또는 CLI를 사용하여 온라인 인스턴스의 보안 그룹을 추가하거나 제거할 수 있습니다. 그러나이 보안 그룹은 AWS OpsWorks Stacks 콘솔에 표시되지 않습니다. 보안 그룹을 제거하려면 Amazon EC2도 사용해야 합니다. 자세한 내용은 보안 그룹을 참조하세요.

다음 사항에 유의하세요.

- 더 많은 제한적 보안 그룹을 추가하더라도 내장 보안 그룹의 포트 액세스 설정을 제한할 수 없습니다. 여러 보안 그룹이 있는 경우, Amazon EC2는 가장 허용적인 설정을 사용합니다.
- 내장 보안 그룹의 구성을 수정해서는 안 됩니다. 스택을 생성할 때 AWS OpsWorks Stacks는 기본 제공 보안 그룹의 구성을 덮어쓰므로 다음에 스택을 생성할 때 변경한 내용이 모두 손실됩니다.

하나 이상의 계층에 대해 더 제한적인 보안 그룹 설정이 필요하다는 것을 알게 되는 경우, 다음 단계를 수행합니다.

1. 적절한 설정으로 사용자 지정 보안 그룹을 생성하여 적절한 계층에 추가합니다.

   사용자 지정 설정이 필요한 계층이 하나뿐이더라도 스택의 모든 계층에는 내장 그룹 외에 적어도 하나의 보안 그룹이 있어야 합니다.
2. 스택 구성을 편집하고 [OpsWorks 보안 그룹 사용] 설정을 [아니요]로 바꿉니다.

   AWS OpsWorks Stacks는 모든 계층에서 내장 보안 그룹을 자동으로 제거합니다.

보안 그룹에 대한 자세한 내용은 Amazon EC2 보안 그룹을 참조하세요.

EC2 인스턴스 프로파일

계층의 인스턴스에 대한 EC2 프로파일을 변경할 수 있습니다. 자세한 내용은 EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 단원을 참조하십시오.

CloudWatch Logs

CloudWatch Logs 탭에서는 Amazon CloudWatch Logs를 활성화하거나 비활성화할 수 있습니다. CloudWatch Logs 통합은 Chef 11.10 및 Chef 12 Linux 기반 스택에서 작동합니다. CloudWatch Logs 통합 활성화와 CloudWatch Logs 콘솔에서 관리하려는 로그 지정에 대한 자세한 정보는 AWS OpsWorks Stacks에서 Amazon CloudWatch Logs 사용 섹션을 참조하세요.

Tags

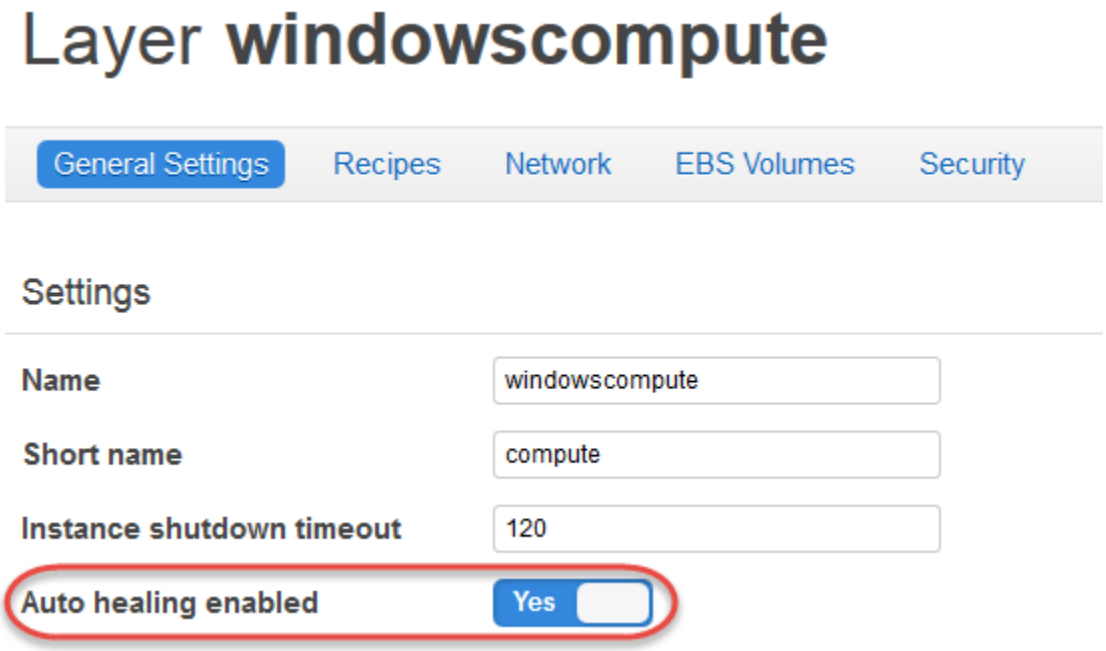
[태그] 탭을 통해 계층에 비용 할당 태그를 적용할 수 있습니다. 태그를 추가한 후 AWS 결제 및 비용 관리 콘솔에서 태그를 활성화할 수 있습니다. 태그를 생성하면 태그가 지정된 구조 내의 모든 리소스에 태그가 적용됩니다. 예를 들어 계층에 태그를 적용하면 계층에 있는 모든 인스턴스, Amazon EBS 볼륨 또는 Elastic Load Balancing 로드 밸런서에 태그가 적용됩니다. 태그를 활성화하고 이를 사용하여 AWS OpsWorks Stacks 리소스의 비용을 추적하고 관리하는 방법에 대한 자세한 내용은 Billing and Cost Management 사용 설명서의 비용 할당 태그 사용 및 사용자 정의 비용 할당 태그 활성화를 참조하세요. AWS OpsWorks Stacks의 태깅에 대한 자세한 내용은 Tags 단원을 참조하세요.

자동 복구를 사용하여 실패한 인스턴스 대체

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

모든 인스턴스에는 서비스와 정기적으로 통신하는 AWS OpsWorks Stacks 에이전트가 있습니다. AWS OpsWorks Stacks는 해당 통신을 사용하여 인스턴스 상태를 모니터링합니다. 에이전트가 약 5분 이상 서비스와 통신하지 않는 경우 AWS OpsWorks Stacks는 인스턴스가 실패한 것으로 간주합니다.

자동 복구는 계층 수준에서 설정됩니다. 다음 스크린샷에 나온 것처럼 계층 설정을 편집하여 자동 복구 설정을 변경할 수 있습니다.



> **Note**
>
> 인스턴스는 여러 계층에 속할 수 있습니다. 이러한 계층 중 하나라도 자동 복구가 비활성화된 경우 AWS OpsWorks Stacks는 실패하더라도 인스턴스를 복구하지 않습니다.

계층에 기본 설정인 자동 복구가 활성화된 경우AWS OpsWorks 스택은 다음과 같이 계층의 실패한 인스턴스를 자동으로 대체합니다.

인스턴스 스토어 지원 인스턴스

1. Amazon EC2 인스턴스를 중지하고 종료되었는지 확인합니다.
2. 루트 볼륨의 데이터를 삭제합니다.
3. 호스트 이름, 구성, 계층 멤버십이 동일한 새 Amazon EC2 인스턴스를 생성합니다.
4. 예전 인스턴스가 원래 시작될 때 연결됐던 볼륨을 포함하여 Amazon EBS 볼륨을 다시 연결합니다.
5. 새 퍼블릭 및 프라이빗 IP 주소를 할당합니다.

6. 예전 인스턴스가 탄력적 IP 주소에 연결된 경우, 새 인스턴스를 동일한 IP 주소에 연결합니다.

Amazon EBS 지원 인스턴스

1. Amazon EC2 인스턴스를 중지하고 중지되었는지 확인합니다.
2. EC2 인스턴스를 시작합니다.

자동 복구된 인스턴스가 다시 온라인 상태가 되면 AWS OpsWorks 스택은 스택의 모든 인스턴스에서 수명 주기 구성 이벤트를 트리거합니다. 연결된 스택 구성 및 배포 속성에는 인스턴스의 퍼블릭 및 프라이빗 IP 주소가 포함됩니다. Custom Configure 레시피는 노드 객체에서 새 IP 주소를 가져올 수 있습니다.

계층의 인스턴스에 Amazon EBS 볼륨을 지정하는 경우 AWS OpsWorks Stacks는 새 볼륨을 생성하고 인스턴스가 시작될 때 각 인스턴스에 연결합니다. 나중에 인스턴스에서 볼륨을 분리하려면 리소스 페이지를 사용하세요.

AWS OpsWorks Stacks는 계층의 인스턴스 중 하나를 자동으로 복구할 때 다음과 같은 방식으로 볼륨을 처리합니다.

- 인스턴스가 실패할 때 볼륨이 인스턴스에 연결된 경우 볼륨과 해당 데이터가 저장되고 AWS OpsWorks Stacks는 이를 새 인스턴스에 연결합니다.
- 인스턴스가 실패할 때 볼륨이 인스턴스에 연결되지 않은 경우, AWS OpsWorks Stacks는 계층에 의해 지정된 구성으로 새로운 빈 볼륨을 생성하여 새 인스턴스에 연결합니다.

자동 복구는 모든 계층에서 기본적으로 활성화되어 있지만 계층의 일반 설정을 편집하여 비활성화할 수 있습니다.

> **Important**
>
> 자동 복구를 활성화하는 경우, 반드시 다음을 수행해야 합니다.
>
> - 인스턴스를 중지하려면 AWS OpsWorks Stacks 콘솔, CLI 또는 API만 사용합니다.
>
> Amazon EC2 콘솔을 사용하는 등 그 밖의 방법으로 인스턴스를 중지하는 경우, AWS OpsWorks Stacks는 인스턴스를 실패한 것으로 취급하고 자동으로 복구합니다.
> - 인스턴스가 자동 복구되는 경우에 잃고 싶지 않은 데이터는 Amazon EBS 볼륨을 사용하여 저장하세요.

자동 복구는 이전 Amazon EC2 인스턴스를 중지하여 Amazon EBS 볼륨에 저장되지 않은 모든 데이터를 삭제합니다. Amazon EBS 볼륨은 새 인스턴스에 다시 연결되어 저장된 모든 데이터를 보존합니다.

OpsWorks 계층 삭제

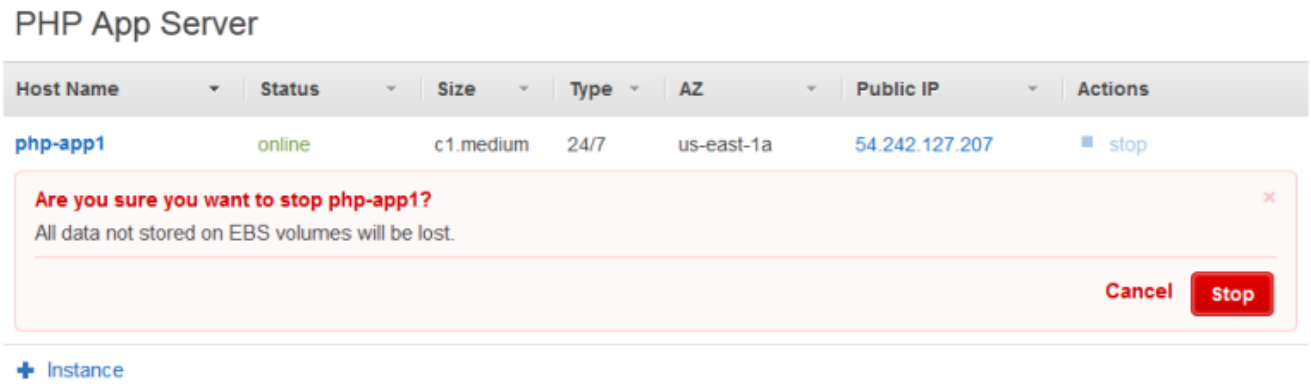
Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

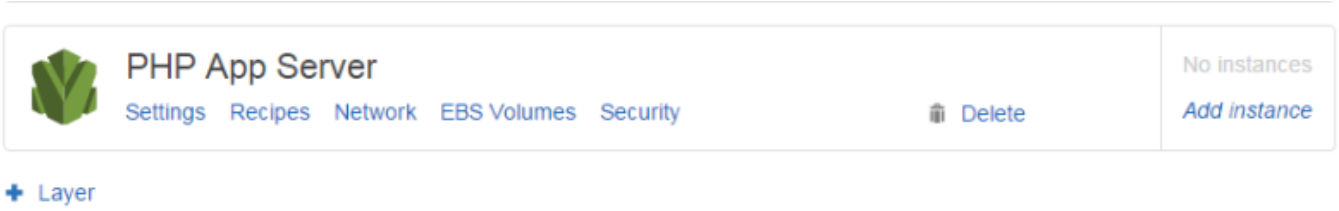
AWS OpsWorks Stacks 계층이 더 이상 필요하지 않은 경우 스택에서 삭제할 수 있습니다.

OpsWorks 계층을 삭제하려면

1. 탐색 창에서 인스턴스를 클릭합니다.
2. 인스턴스 페이지에서 삭제하려는 계층 이름 아래에 있는 각 인스턴스의 작업 열에서 중지를 클릭합니다.



3. 각 인스턴스가 중지되면 [삭제]를 클릭하여 계층에서 인스턴스를 제거합니다.
4. 탐색 창에서 [계층]을 클릭합니다.
5. [계층] 페이지에서 [삭제]를 선택합니다.



Elastic Load Balancing 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Elastic Load Balancing은 AWS OpsWorks Stacks 계층과 약간 다르게 작동합니다. 계층을 생성하고 이 계층에 인스턴스를 추가하는 대신, Elastic Load Balancing 콘솔 또는 API를 사용하여 로드 밸런서를 생성한 후 기존 계층에 연결합니다. 계층의 인스턴스로 트래픽을 분배하는 것 이외에, Elastic Load Balancing이 수행하는 역할은 다음과 같습니다.

- 비정상 Amazon EC2 인스턴스를 검색하고 비정상 인스턴스가 복원될 때까지 트래픽을 나머지 정상 인스턴스로 다시 라우팅합니다.
- 수신 트래픽에 맞춰 요청 처리 용량을 자동으로 조정합니다.
- 연결 드레이닝 기능을 활성화할 경우 로드 밸런서가 비정상이거나 연결 상태를 유지하지만 곧 등록 취소될 인스턴스에 대해서는 지정된 값의 제한 시간 동안 새로운 요청 전송을 중지하여 인스턴스가 인플라이트 요청을 모두 완료하도록 합니다.

로드 밸런서를 계층에 연결한 후 AWS OpsWorks Stacks는 다음을 수행합니다.

- 현재 등록된 인스턴스를 등록 취소합니다.
- 온라인 상태가 되면 계층의 인스턴스(로드 기반 및 시간 기반 인스턴스 포함)를 자동으로 등록하고 오프라인 상태가 되면 등록 취소합니다.
- 해당 가용 영역에서 등록된 인스턴스에 대한 라우팅 요청을 자동으로 시작합니다.

로드 밸런서의 연결 드레이닝 기능을 활성화한 경우 AWS OpsWorks Stacks가 이를 지원하는지 여부를 지정할 수 있습니다. 연결 드레이닝 지원(기본 설정)을 활성화하면 인스턴스가 종료된 후 AWS OpsWorks Stacks는 다음을 수행합니다.

- 로드 밸런서에서 인스턴스를 등록 해제합니다.

  로드 밸런서는 새 요청 전송을 중지하고 연결 드레이닝을 시작합니다.
- 로드 밸런서가 연결 드레이닝을 완료할 때까지 Shutdown 수명 주기 이벤트 트리거를 늦춥니다.

연결 드레이닝 지원을 활성화하지 않으면 인스턴스가 로드 밸런서에 여전히 연결되어 있더라도 AWS OpsWorks Stacks는 인스턴스가 종료되는 즉시 종료 이벤트를 트리거합니다.

스택에서 Elastic Load Balancing을 사용하려면 먼저 Elastic Load Balancing 콘솔, CLI 또는 API를 사용하여 동일한 리전에서 하나 이상의 로드 밸런서를 생성해야 합니다. 다음 사항을 숙지해야 합니다.

- 각 계층에 로드 밸런서를 하나만 연결할 수 있습니다.
- 각 로드 밸런서는 한 계층만 처리할 수 있습니다.
- AWS OpsWorks Stacks는 Application Load Balancer를 지원하지 않습니다. Classic Load Balancer는 AWS OpsWorks Stacks와 함께만 사용할 수 있습니다.

따라서 밸런싱하려는 각 스택의 각 계층마다 별도의 Elastic Load Balancing 로드 밸런서를 생성하고 각각 해당 목적으로만 사용해야 합니다. 권장되는 방법은 AWS OpsWorks Stacks에서 사용하려는 각 Elastic Load Balancing 로드 밸런서에 MyStack1-RailsLayer-ELB와 같은 구별 가능한 이름을 할당하여 로드 밸런서를 여러 목적으로 사용하는 것을 방지하는 것입니다.

> **Important**
>
> Stacks 계층에 대해 새 Elastic Load Balancing 로드 AWS OpsWorks 밸런서를 생성하는 것이 좋습니다. 기존 Elastic Load Balancing 로드 밸런서를 사용하려는 경우 먼저 해당 로드 밸런서가 다른 목적으로 사용되고 있지 않고 연결된 인스턴스가 없는지 확인해야 합니다. 로드 밸런서가 계층에 연결되면 OpsWorks가 기존 인스턴스를 모두 제거하고 로드 밸런서를 해당 계층의 인스턴스만 처리하도록 구성합니다. 로드 밸런서를 계층에 연결한 후 Elastic Load Balancing 콘솔 또는 API를 사용하여 로드 밸런서의 구성을 수정하는 것은 기술적으로는 가능하지만 변경 사항이 영구적이지 않으므로 이렇게 하면 안 됩니다.

Elastic Load Balancing 로드 밸런서를 계층에 연결하려면

1. 아직 연결하지 않은 경우 Elastic Load Balancing 콘솔, API 또는 CLI를 사용하여 스택의 리전에서 로드 밸런서를 생성합니다. 로드 밸런서를 생성할 때 다음 작업을 수행합니다.

   - 애플리케이션에 적합한 상태 확인 ping 경로를 지정해야 합니다.

     기본 ping 경로는 `/index.html`이고, 애플리케이션 루트에 `index.html`이 포함되지 않은 경우 적절한 ping 경로를 지정해야 상태 확인에 실패하지 않습니다.
   - 연결 드레이닝을 사용하려는 경우 이 기능이 활성화되어 있고 이 기능에 적절한 제한 시간 값이 설정되어 있어야 합니다.

   자세한 내용은 Elastic Load Balancing을 참조하세요.
2. 균형을 맞추려는 계층을 생성하거나 기존 계층의 네트워크 설정을 편집합니다.

   > **Note**
   >
   > 사용자 지정 계층을 생성하는 경우 로드 밸런서를 연결할 수 없습니다. 계층의 설정을 편집해야 합니다.
3. Elastic Load Balancing에서 계층에 연결할 로드 밸런서를 선택하고 AWS OpsWorks Stacks가 연결 드레이닝을 지원할지 여부를 지정합니다.

로드 밸런서를 계층에 연결하면 AWS OpsWorks 스택은 스택의 인스턴스에서 수명 주기 구성 이벤트를 트리거하여 변경 사항을 알립니다. AWS OpsWorks 또한 스택은 로드 밸런서를 분리할 때 구성 이벤트를 트리거합니다.

> **Note**
>
> 인스턴스가 부팅되면 AWS OpsWorks Stacks는 패키지를 설치하고 애플리케이션을 배포하는 설정 및 배포 레시피를 실행합니다. 이러한 레시피가 완료되면 인스턴스는 온라인 상태가 되고 AWS OpsWorks Stacks는 인스턴스를 Elastic Load Balancing에 등록합니다. AWS OpsWorks 또한 스택은 인스턴스가 온라인 상태가 된 후 구성 이벤트를 트리거합니다. 이는 Elastic Load Balancing 등록과 Configure 레시피가 동시에 실행될 수 있으며, Configure 레시피가 완료되기 전에 인스턴스가 등록될 수 있다는 의미입니다. 인스턴스가 Elastic Load Balancing에 등록되기 전에 레시피가 완료되도록 레시피를 계층의 설정 또는 Deploy 수명 주기 이벤트에 추가해야 합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.

인스턴스를 로드 밸런서에서 제거하는 것이 유용할 때가 가끔 있습니다. 예를 들어 앱을 업데이트할 때 앱을 단일 인스턴스에 배포하고 앱이 올바로 작동하는지 확인한 후에 모든 인스턴스로 배포하는 것이 좋습니다. 일반적으로 업데이트가 검증될 때까지 사용자 요청을 수신하지 않도록 해당 인스턴스를 로드 밸런서에서 제거합니다.

온라인 인스턴스를 일시적으로 로드 밸런서에서 제거하려면 Elastic Load Balancing 콘솔 또는 API를 사용해야 합니다. 다음 섹션에서는 콘솔 사용 방법을 설명합니다.

인스턴스를 로드 밸런서에서 일시적으로 제거하려면

1. Amazon EC2 콘솔을 열고 로드 밸런서를 선택합니다.
2. 해당 로드 밸런서를 선택하고 [인스턴스] 탭을 엽니다.
3. 인스턴스의 작업 열에서 로드 밸런서에서 제거를 선택합니다.
4. 마치면 [인스턴스 편집]을 선택하여 인스턴스를 로드 밸런서로 반환합니다.

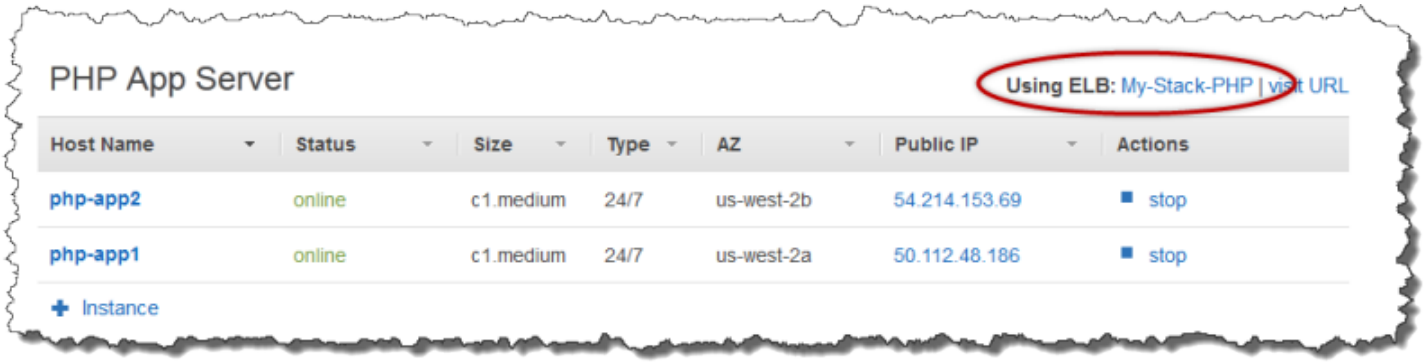
> **Important**
>
> Elastic Load Balancing 콘솔 또는 API를 사용하여 로드 밸런서에서 인스턴스를 제거하는 경우 Elastic Load Balancing을 사용하여 다시 배치해야 합니다. AWS OpsWorks Stacks는 다른 서비스 콘솔 또는 APIs로 수행하는 작업을 인식하지 못하며 인스턴스를 로드 밸런서에 반환하지 않습니다. 경우에 따라 AWS OpsWorks Stacks는 인스턴스를 ELB에 다시 추가할 수 있지만, 이는 보장되는 동작이 아니며 모든 경우에 발생하지 않습니다.

다음과 같이 특정 인스턴스 집합에 여러 로드 밸런서를 연결할 수 있습니다.

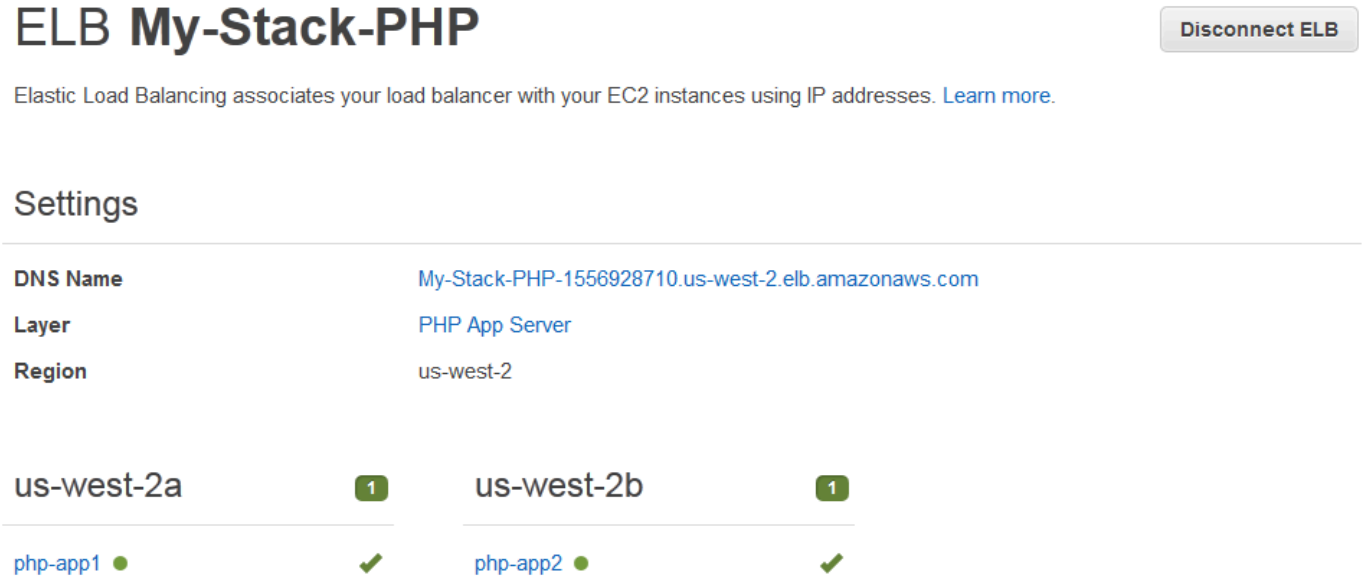
여러 로드 밸런서를 연결하려면

1. Elastic Load Balancing 콘솔, API 또는 CLI를 사용하여 로드 밸런서 세트를 생성합니다.
2. 각 로드 밸런서에 대해 사용자 지정 계층을 생성하고 해당 계층에 로드 밸런서 중 하나를 연결합니다. 이러한 계층에 대해 사용자 지정 레시피를 구현할 필요가 없습니다. 기본 사용자 지정 계층이면 충분합니다.
3. 각 사용자 지정 계층에 인스턴스 세트를 추가합니다.

[인스턴스] 페이지로 이동하여 해당 로드 밸런서 이름을 클릭하면 로드 밸런서의 속성을 확인할 수 있습니다.



[ELB] 페이지에는 연결된 인스턴스의 DNS 이름 및 상태를 비롯해 로드 밸런서의 기본 속성이 표시됩니다. 스택이 VPC에서 실행되는 경우 이 페이지에는 가용 영역이 아니라 서브넷이 표시됩니다. 녹색 확인 표시는 정상 인스턴스를 나타냅니다. 이름을 클릭하면 로드 밸런서를 통해 서버에 연결할 수 있습니다.



Amazon RDS 서비스 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon RDS 서비스 계층은 Amazon RDS 인스턴스를 나타냅니다. 이 계층은 기존 Amazon RDS 인스턴스만 나타낼 수 있습니다. 이 인스턴스는 Amazon RDS 콘솔 또는 API를 사용하여 별도로 생성해야 합니다.

Amazon RDS 서비스 계층을 스택으로 통합하는 기본 절차는 다음과 같습니다.

1. Amazon RDS 콘솔, API 또는 CLI를 사용하여 인스턴스를 생성합니다.

   인스턴스의 ID, 마스터 사용자 이름, 마스터 암호 및 데이터베이스 이름을 기록해야 합니다.
2. Amazon RDS 계층을 스택에 추가하려면 Amazon RDS 인스턴스를 스택에 등록합니다.
3. 계층을 앱에 연결합니다. 그러면 Amazon RDS 인스턴스의 연결 정보가 앱의 `deploy` 속성에 추가됩니다.
4. 언어별 파일 또는 `deploy` 속성 내 정보를 사용하여 애플리케이션을 Amazon RDS 인스턴스에 연결합니다.

   애플리케이션을 데이터베이스 서버에 연결하는 방법에 대한 자세한 정보는 the section called “데이터베이스에 연결” 단원을 참조하세요.

> **Warning**
>
> 인스턴스의 마스터 암호 및 사용자 이름의 문자가 애플리케이션 서버와 호환되는지 확인하세요. 예를 들어 Java 앱 서버 계층에서는 이러한 문자열에 &를 포함할 경우 XML 구문 분석 오류가 발생하여 Tomcat 서버가 시작하지 못합니다.

주제

- 보안 그룹 지정
- 스택에 Amazon RDS 인스턴스 등록
- Amazon RDS 서비스 계층을 앱에 연결
- 스택에서 Amazon RDS 서비스 계층 제거

보안 그룹 지정

AWS OpsWorks Stacks에서 Amazon RDS 인스턴스를 사용하려면 데이터베이스 또는 VPC 보안 그룹이 적절한 IP 주소에서 액세스를 허용해야 합니다. 프로덕션 용도에서 보안 그룹은 일반적으로 액세스 권한을 데이터베이스에 액세스하는 데 필요한 IP 주소로만 제한합니다. 일반적으로 데이터베이스를 관

리하는 데 사용하는 시스템의 주소와 데이터베이스에 액세스해야 하는 AWS OpsWorks Stacks 인스턴스가 포함됩니다. AWS OpsWorks Stacks는 리전에서 첫 번째 스택을 생성할 때 각 계층 유형에 대해 Amazon EC2 보안 그룹을 자동으로 생성합니다. AWS OpsWorks Stacks 인스턴스에 대한 액세스를 제공하는 간단한 방법은 Amazon RDS 인스턴스 또는 VPC에 적절한 AWS OpsWorks Stacks 보안 그룹을 할당하는 것입니다.

기존 Amazon RDS 인스턴스에 대한 보안 그룹을 지정하려면

1. https://console.aws.amazon.com/rds/에서 Amazon RDS 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 클릭하고 해당 Amazon RDS 인스턴스를 선택합니다. [인스턴스 작업], [수정]을 클릭합니다.
3. [보안 그룹] 목록에서 다음 보안 그룹을 선택한 다음 [계속] 및 [DB 인스턴스 수정]을 클릭합니다.
   - [AWS-OpsWorks-DB-Master-Server (***security_group_id***)] 보안 그룹
   - 인스턴스가 데이터베이스에 연결되는 앱 서버 계층의 보안 그룹. 그룹 이름에는 계층 이름이 포함됩니다. 예를 들어, PHP 앱 서버 인스턴스에 대한 데이터베이스 액세스 권한을 제공하려면 AWS-OpsWorks-PHP-App-Server 그룹을 지정합니다.

새 Amazon RDS 인스턴스를 생성하는 경우 DB 인스턴스 시작 마법사의 고급 설정 구성 페이지에서 적절한 AWS OpsWorks Stacks 보안 그룹을 지정할 수 있습니다. 이 마법사를 사용하는 방법에 대한 자세한 내용은 MySQL DB 인스턴스를 만들고 MySQL DB 인스턴스의 데이터베이스에 연결을 참조하세요.

VPC 보안 그룹을 지정하는 방법에 대한 자세한 내용은 VPC의 보안 그룹을 참조하세요.

스택에 Amazon RDS 인스턴스 등록

스택에서 Amazon RDS 서비스 계층을 추가하려면 인스턴스를 스택에 등록해야 합니다.

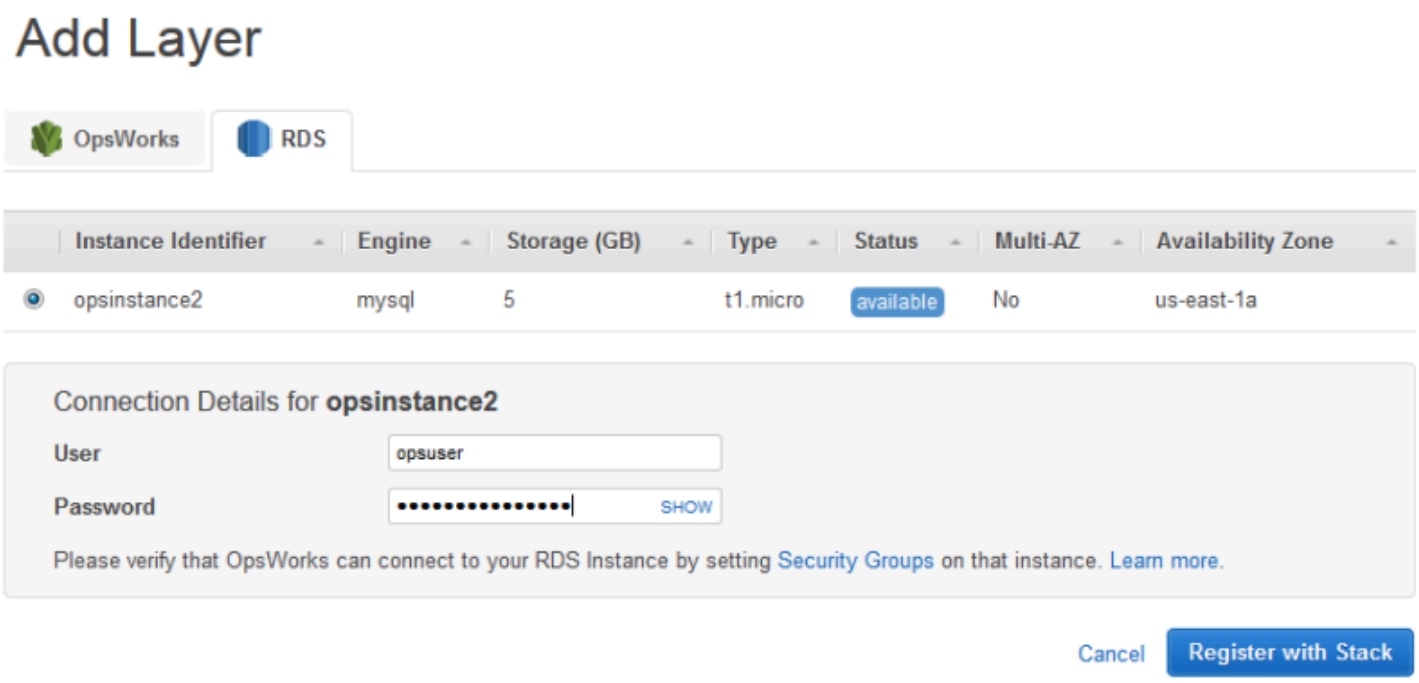
Amazon RDS 인스턴스를 스택에 등록하려면

1. AWS OpsWorks Stacks 콘솔의 탐색 창에서 계층을 클릭하고 + 계층 또는 계층 추가를 클릭하여 계층 추가 페이지를 연 다음 RDS 탭을 클릭합니다.
2. 필요한 경우 스택의 서비스 역할 사용 단원에서 설명한 대로 스택의 서비스 역할을 업데이트합니다.
3. RDS 탭을 클릭하여 사용 가능한 Amazon RDS 인스턴스를 나열합니다.

> **Note**
>
> 계정에 Amazon RDS 인스턴스가 없으면 RDS 탭에서 RDS 인스턴스 추가를 클릭하여 인스턴스를 하나 만듭니다. 이 옵션을 클릭하면 Amazon RDS 콘솔로 연결되어 DB 인스턴스 시작 마법사가 시작됩니다. 또는 Amazon RDS 콘솔로 직접 이동하여 DB 인스턴스 시작을 클릭하거나 Amazon RDS API 또는 CLI를 사용합니다. Amazon RDS 인스턴스 생성에 대한 자세한 내용은 Amazon RDS 시작하기를 참조하세요.

4. 적절한 인스턴스를 선택하고, [사용자] 및 [암호]를 적절한 사용자 및 암호 값으로 설정한 다음 [스택에 등록]을 클릭합니다.

> **Important**
>
> Amazon RDS 인스턴스 등록에 사용할 사용자 및 암호가 유효한 것인지 확인해야 합니다. 그렇지 않으면 애플리케이션에서 인스턴스에 연결할 수 없게 됩니다. 그러나 계층을 편집하여 유효한 사용자 및 암호 값을 입력한 다음 앱을 다시 배포할 수는 있습니다.



스택에 Amazon RDS 서비스 계층을 추가하면 AWS OpsWorks Stacks는 스택에 ID를 할당하고 연결된 Amazon RDS 구성을 스택 구성 및 배포 속성의 `[:opsworks][:stack]` 속성에 추가합니다.

> **Note**
>
> 등록된 Amazon RDS 인스턴스의 암호를 변경하는 경우 AWS OpsWorks Stacks에서 암호를 수동으로 업데이트한 다음 앱을 재배포하여 스택 인스턴스의 스택 구성 및 배포 속성을 업데이트해야 합니다.

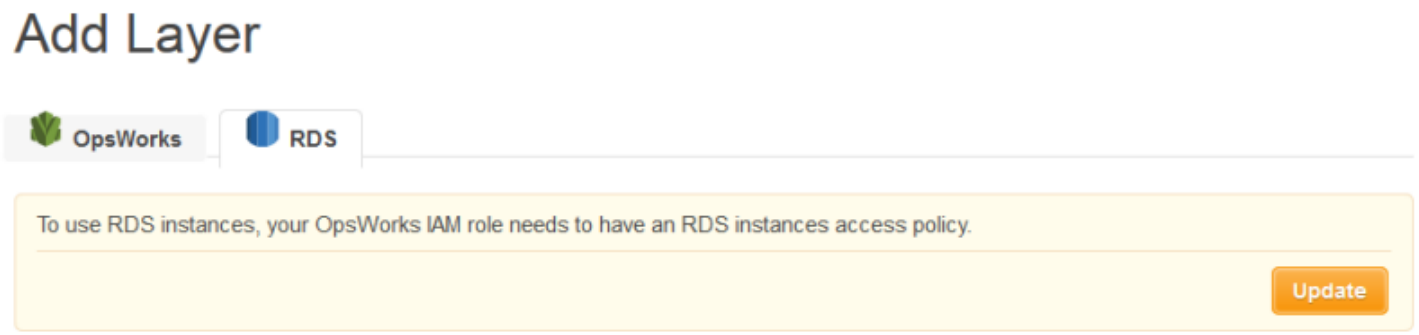
주제

- 스택의 서비스 역할 사용

스택의 서비스 역할 사용

모든 스택에는 AWS OpsWorks Stacks가 다른 AWS 서비스와 함께 사용자를 대신하여 수행할 수 있는 작업을 지정하는 IAM 서비스 역할이 있습니다. 스택에 Amazon RDS 인스턴스를 등록하려면 해당 서비스 역할이 AWS OpsWorks Stacks에 Amazon RDS에 액세스할 수 있는 권한을 부여해야 합니다.

처음으로 Amazon RDS 서비스 계층을 스택 중 하나에 추가할 때 서비스 역할에 필요한 권한이 없을 수 있습니다. 그럴 경우 [계층 추가] 페이지에서 RDS 탭을 클릭하면 다음이 표시됩니다.



업데이트를 클릭하여 AWS OpsWorks Stacks가 서비스 역할의 정책을 다음과 같이 업데이트하도록 합니다.

```
{"Statement": [{"Action": ["ec2:*", "iam:PassRole",
                      "cloudwatch:GetMetricStatistics",
                      "elasticloadbalancing:*",
                      "rds:*"],
            "Effect": "Allow",
            "Resource": ["*"] }]
}
```

> **Note**
>
> 업데이트는 한 번만 수행하면 됩니다. 업데이트된 역할이 자동으로 모든 스택에서 사용됩니다.

Amazon RDS 서비스 계층을 앱에 연결

Amazon RDS 서비스 계층을 추가한 후 계층을 앱에 연결할 수 있습니다.

- 앱을 생성할 때 또는 나중에 앱 구성을 편집하여 Amazon RDS 계층을 앱에 연결할 수 있습니다.
- 앱에서 Amazon RDS 계층을 연결 해제하려면 앱의 구성을 편집하여 다른 데이터베이스 서버를 지정하거나 아무 서버도 지정하지 마세요.

  Amazon RDS 계층은 스택의 일부로 유지되며 다른 앱에 연결될 수 있습니다.

Amazon RDS 인스턴스를 앱과 연결하면 AWS OpsWorks Stacks는 앱의 서버에 데이터베이스 연결 정보를 저장합니다. 각 서버 인스턴스의 애플리케이션이 이 정보를 사용하여 데이터베이스에 연결할 수 있습니다. Amazon RDS 인스턴스에 연결하는 방법에 대한 자세한 정보는 the section called “데이터베이스에 연결” 섹션을 참조하세요.

스택에서 Amazon RDS 서비스 계층 제거

스택에서 Amazon RDS 서비스 계층을 제거하려면 등록을 해제합니다.

Amazon RDS 서비스 계층 등록을 취소하려면

1. 탐색 창에서 계층을 클릭하고 Amazon RDS 서비스 계층의 이름을 클릭합니다.
2. [등록 취소]를 클릭하고 해당 계층의 등록 취소를 확인합니다.

이 절차는 스택에서 계층을 제거하지만, 기본 Amazon RDS 인스턴스는 삭제하지 않습니다. 인스턴스와 모든 데이터베이스는 계정에 유지되며 다른 스택에 등록될 수 있습니다. 인스턴스를 삭제하려면 Amazon RDS 콘솔, API 또는 CLI를 사용해야 합니다. 자세한 내용은 DB 인스턴스 삭제 단원을 참조하세요.

ECS 클러스터 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon Elastic Container Service(Amazon ECS)는 컨테이너 인스턴스라고 하는 Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스 클러스터에서 도커 컨테이너를 관리합니다. ECS 클러스터 계층은 Amazon ECS 클러스터를 나타내며, 다음과 같은 기능을 제공하여 클러스터 관리를 단순화합니다.

- 간소화된 컨테이너 인스턴스 프로비저닝 및 관리
- 컨테이너 인스턴스 운영 체제 및 패키지 업데이트
- 사용자 권한 관리
- 컨테이너 인스턴스 성능 모니터링
- Amazon Elastic Block Store(Amazon EBS) 볼륨 관리
- 퍼블릭 및 탄력적 IP 주소 관리
- 보안 그룹 관리

ECS 클러스터 계층의 제한 및 요구 사항은 다음과 같습니다.

- 이 계층은 기본 VPC를 비롯한 VPC에서 실행되는 Chef 11.10 또는 Chef 12 Linux 스택에서만 사용할 수 있습니다.
- 이 계층의 인스턴스에서 다음 운영 체제 중 하나가 실행 중이어야 합니다.
  - Amazon Linux 2
  - Amazon Linux 2018.03
  - Amazon Linux 2017.09
  - Amazon Linux 2017.03
  - Amazon Linux 2016.09
  - Amazon Linux 2016.03

- Amazon Linux 2015.09
- Amazon Linux 2015.03
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS
- Ubuntu 14.04 LTS
- 사용자 지정(Custom)

- 이 계층의 인스턴스에 설치된 AWS OpsWorks Stacks 에이전트 버전은 `3425-20150727112318` 이상이어야 합니다.

주제

- 스택에 ECS 클러스터 계층 추가
- ECS 클러스터 관리
- 스택에서 ECS 클러스터 계층 삭제

스택에 ECS 클러스터 계층 추가

AWS OpsWorks Stacks는 기존 Amazon ECS 클러스터의 컨테이너 인스턴스를 시작하고 유지 관리하는 프로세스를 간소화합니다. 클러스터와 작업 같은 다른 Amazon ECS 엔터티를 생성하거나 시작하려면 Amazon ECS 콘솔, 명령줄 인터페이스(CLI) 또는 API를 사용하세요. (자세한 내용은 Amazon Elastic Container Service 개발자 안내서를 참조하세요.) 그런 다음 Stacks에서 클러스터를 관리하는 데 사용할 수 있는 ECS 클러스터 계층을 생성하여 클러스터를 AWS OpsWorks 스택과 연결할 수 있습니다.

클러스터를 스택에 연결하는 방법은 다음과 같습니다.

- 각 스택은 단일 클러스터를 나타내는 ECS 클러스터 계층 하나를 가질 수 있습니다.
- 하나의 클러스터는 하나의 스택에만 연결할 수 있습니다.

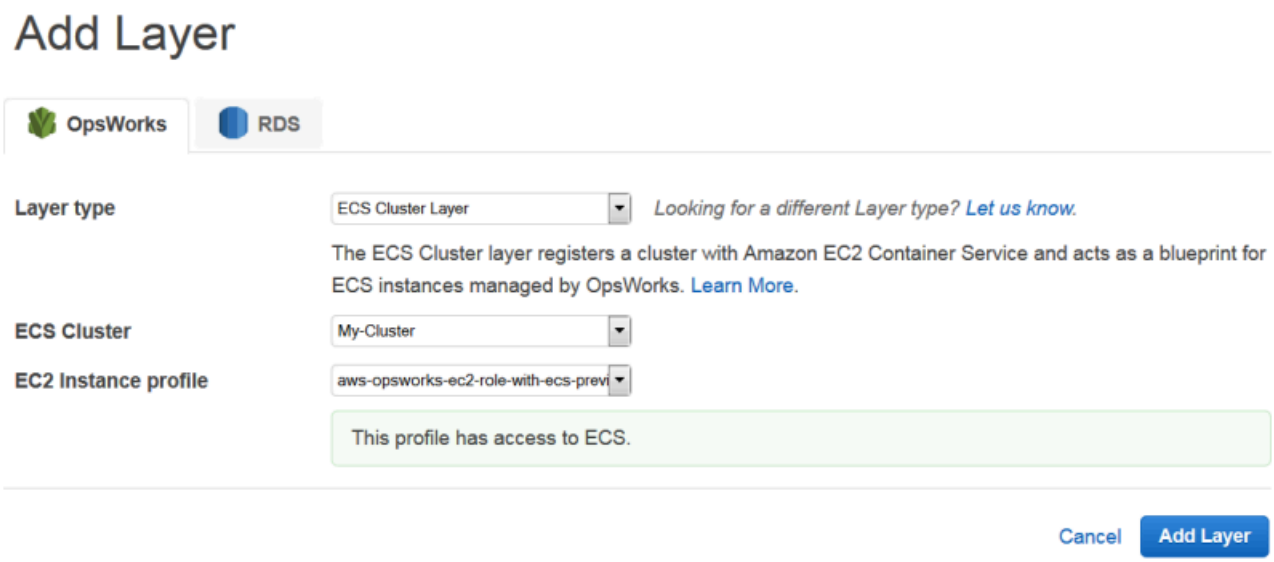
스택에 ECS 클러스터 계층을 추가하려면 먼저 일반적으로 AWS OpsWorks 이름이 인 Stacks AWS Identity and Access Management (IAM) 서비스 역할을 업데이트`aws-opsworks-service-role`하여 AWS OpsWorks Stacks가 사용자를 대신하여 Amazon ECS와 상호 작용할 수 있도록 해야 합니다. 서비스 역할에 대한 자세한 정보는 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 단원을 참조하세요.

ECS 클러스터 계층을 처음 생성할 때 콘솔은 AWS OpsWorks Stacks에 역할을 업데이트하도록 지시할 수 있는 업데이트 버튼을 제공합니다. 그러면 AWS OpsWorks 스택에 계층을 추가할 수 있도록 스택에 계층 추가 페이지가 표시됩니다. 서비스 역할은 한 번만 업데이트해야 합니다. 그 다음 업데이트된 역할을 사용하여 원하는 스택에 ECS 클러스터 계층을 추가할 수 있습니다.

> **Note**
>
> 원한다면 다음과 같이 `ecs:*` 권한을 기존 정책에 추가하여 서비스 역할의 정책을 수동으로 업데이트할 수 있습니다.

```
{
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "elasticloadbalancing:*",
        "rds:*",
        "ecs:*"
      ],
      "Effect": "Allow",
      "Resource": ["*"]
    }
  ]
}
```

클러스터를 스택에 연결하려면 클러스터를 스택에 등록한 다음 연결된 계층을 생성하는 두 가지 작업이 필요합니다. AWS OpsWorks Stacks 콘솔은 이러한 단계를 결합합니다. 계층 생성은 지정된 클러스터를 자동으로 등록합니다. AWS OpsWorks Stacks API, CLI 또는 SDK를 사용하는 경우 별도의 작업을 사용하여 클러스터를 등록하고 연결된 계층을 생성해야 합니다. 콘솔을 사용하여 ECS 클러스터 계층을 스택에 추가하려면 계층을 선택하고 +계층 또는 계층 추가를 선택한 다음 ECS 클러스터 계층 유형을 선택합니다.



[계층 추가] 페이지에는 다음 구성 옵션이 포함됩니다.

ECS 클러스터

스택에 등록하려는 Amazon ECS 클러스터.

EC2 인스턴스 프로파일

클러스터의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 프로파일. 이 프로파일은 클러스터의 컨테이너 인스턴스에서 실행되는 애플리케이션에게 Amazon ECS 등 다른 AWS 서비스에 액세스할 수 있는 권한을 부여합니다. 첫 번째 ECS 클러스터 계층을 생성할 때 ECS 액세스 권한이 있는 새 프로파일을 선택하여 AWS OpsWorks Stacks에 이라는 필수 프로파일을 생성하도록 지시합니다`aws-opsworks-ec2-role-with-ecs`. 그러면 이후의 모든 ECS 클러스터 계층에 이 프로파일을 사용할 수 있습니다. 인스턴스 프로파일에 대한 자세한 정보는 EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 단원을 참조하세요.

다음을 비롯한 계층의 구성을 편집하여 다른 설정을 지정할 수 있습니다.

- Elastic Load Balancing 로드 밸런서를 계층에 연결.

  이 방법은 일부 사용 사례에 적합할 수 있지만 Amazon ECS는 보다 복잡한 옵션을 제공합니다. 자세한 내용은 서비스 로드 밸런싱을 참조하세요.
- 컨테이너 인스턴스에 자동으로 퍼블릭 IP 주소 또는 탄력적 IP 주소를 할당할지 여부를 지정.

  두 가지 주소 유형 모두 자동 할당을 비활성화하면 서브넷에 적절히 구성된 NAT이 없는 한 인스턴스는 온라인 상태가 되지 않습니다. 자세한 내용은 VPC에서 스택 실행 단원을 참조하십시오.

ECS 클러스터 관리

ECS 클러스터 계층을 생성한 후 AWS OpsWorks Stacks를 사용하여 다음과 같이 클러스터를 관리할 수 있습니다.

컨테이너 인스턴스 프로비저닝 및 관리

원본 클러스터에는 포함되어 있더라도 처음에는 ECS 클러스터 계층에 컨테이너 인스턴스가 포함되어 있지 않습니다. 한 가지 방법은 적절한 다음 조합을 사용하여 계층의 인스턴스를 관리하는 것입니다.

- 수동으로 계층에 24/7 인스턴스를 추가 하고 더 이상 필요하지 않을 때 인스턴스를 삭제합니다.
- 계층에 시간 기반 인스턴스를 추가하여 일정에 따라 인스턴스를 추가하거나 삭제합니다.
- 계층에 로드 기반 인스턴스를 추가하여 AWS OpsWorks Stacks 호스트 지표 또는 CloudWatch 경보를 기반으로 인스턴스를 추가하거나 삭제합니다. 자동 로드 기반 조정 사용

> **Note**
>
> Amazon ECS가 스택의 기본 운영 체제에 지원되지 않는 경우, 컨테이너 인스턴스를 생성할 때 다음과 같이 지원되는 운영 체제를 명시적으로 지정해야 합니다. Amazon Linux 2, Amazon Linux 2018.03, Amazon Linux 2017.09, Amazon Linux 2017.03, Amazon Linux 2016.09, Amazon Linux 2016.03, Amazon Linux 2015.09, Amazon Linux 2015.03, Ubuntu 18.04 LTS, Ubuntu 16.04 LTS, Ubuntu 14.04 LTS 또는 사용자 지정. 이 AMI에는 이미 ECS 에이전트가 포함되어 있으므로 ECS 최적화 AMI를 사용하여 ECS 계층에서 인스턴스를 생성하지 마십시오. AWS OpsWorks Stacks는 인스턴스 설정 프로세스 중에 ECS 에이전트를 설치하려고 시도하며 충돌로 인해 설정이 실패할 수 있습니다.

자세한 내용은를 참조하세요서버 수 최적화. AWS OpsWorks Stacks는 각 인스턴스에 AWS-OpsWorks-ECS-Cluster 보안 그룹을 할당합니다. 각각의 새 인스턴스 부팅이 완료되면 AWS OpsWorks Stacks는 Docker와 Amazon ECS 에이전트를 설치한 다음 클러스터에 인스턴스를 등록하여 이를 컨테이너 인스턴스로 변환합니다.

기존 컨테이너 인스턴스를 사용하려면 스택에 인스턴스를 등록하고 ECS 클러스터 계층에 할당합니다. 인스턴스는 지원되는 운영 체제(Amazon Linux 2015.03 이상 또는 Ubuntu 14.04 LTS 이상)에서 실행되어야 합니다.

> **Note**
>
> 컨테이너 인스턴스는 ECS 클러스터 계층과 다른 내장 계층에 모두 속할 수는 없습니다. 하지만 컨테이너 인스턴스는 ECS 클러스터 계층 및 하나 이상의 사용자 지정 계층에 속할 수 있습니다.

운영 체제 및 패키지 업데이트 실행

새 인스턴스 부팅이 완료되면 AWS OpsWorks Stacks가 최신 업데이트를 설치합니다. 그런 다음 AWS OpsWorks Stacks를 사용하여 컨테이너 인스턴스를 최신 상태로 유지할 수 있습니다. 자세한 내용은 보안 업데이트 관리 단원을 참조하십시오.

사용자 권한 관리

AWS OpsWorks Stacks는 사용자의 SSH 키 관리를 포함하여 컨테이너 인스턴스에 대한 권한을 관리하는 간단한 방법을 제공합니다. 자세한 내용은 사용자 권한 관리 및 SSH 액세스 관리 섹션을 참조하세요.

성능 측정치 모니터링

AWS OpsWorks Stacks는 스택, 계층 또는 개별 인스턴스의 성능 지표를 모니터링하는 다양한 방법을 제공합니다. 자세한 내용은 모니터링 단원을 참조하십시오.

작업 또는 서비스 생성 같은 다른 관리 작업은 Amazon ECS를 통해 처리합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서를 참조하세요.

> **Note**
>
> Amazon ECS 콘솔의 클러스터 페이지로 직접 이동하려면 인스턴스를 선택한 다음 ECS 클러스터 계층 섹션의 오른쪽 상단 모서리에 있는 ECS 클러스터를 선택합니다.

스택에서 ECS 클러스터 계층 삭제

클러스터가 더 이상 필요하지 않으면 ECS 클러스터 계층을 삭제하고 연결된 클러스터를 등록 해제합니다. 스택에서 클러스터를 제거하려면 클러스터를 등록 해제한 다음 연결된 계층을 삭제하는 두 가지 작업이 필요합니다. AWS OpsWorks Stacks 콘솔은 이러한 단계를 결합합니다. 계층 삭제는 지정된 클러스터의 등록을 자동으로 취소합니다. AWS OpsWorks Stacks API, CLI 또는 SDK를 사용하는 경우 별도의 작업을 사용하여 클러스터 등록을 취소하고 연결된 계층을 삭제해야 합니다.

콘솔을 사용하여 ECS 클러스터 계층을 삭제하려면

1. 작업 종료 방법을 조정하려면 Amazon ECS 콘솔, API 또는 CLI에서 클러스터의 서비스를 축소한 후 삭제합니다. 자세한 내용은 Amazon ECS 리소스 정리를 참조하세요.
2. 계층의 인스턴스를 중지한 다음 삭제합니다. 컨테이너 인스턴스를 중지하면 AWS OpsWorks Stacks는 실행 중인 작업을 자동으로 중지하고 클러스터에서 인스턴스를 등록 취소하고 인스턴스를 종료합니다.

   > **Note**
   >
   > 스택에 기존 컨테이너 인스턴스를 등록한 경우 계층에서 인스턴스의 할당을 해제한 다음 인스턴스를 등록 취소할 수 있습니다. 그러면 인스턴스가 다시 ECS의 제어를 받습니다.

3. 계층을 삭제합니다. AWS OpsWorks Stacks는 연결된 클러스터를 등록 취소하지만 삭제하지는 않습니다. 클러스터는 Amazon ECS에 그대로 남아 있습니다.

사용자 지정 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 계층에는 최소한의 레시피 집합만 있습니다. 그런 다음 사용자 지정 레시피를 구현해 계층의 수명 주기에 할당하여 계층에 적절한 기능을 추가합니다.

사용자 지정 계층에는 다음과 같은 구성 설정이 있습니다.

> **Note**
>
> AWS OpsWorks Stacks는 계층의 인스턴스에 Ruby를 자동으로 설치합니다. 인스턴스에서 Ruby 코드를 실행하되 기본 Ruby 버전을 사용하지 않으려면 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 선호하는 버전을 지정할 수 있습니다. 자세한 내용은 Ruby 버전 단원을 참조하십시오.

사용자 지정 계층을 생성하는 기본적 절차는 다음 단계로 구성됩니다.

1. 패키지를 설치 및 구성하고, 구성 변경 사항을 처리하고, 앱을 배포하는 등의 작업을 수행하는 데 필요한 레시피와 관련 파일이 들어 있는 쿡북을 구현합니다.

   요구 사항에 따라 배포 취소 및 종료 작업을 처리하기 위한 레시피가 필요할 수도 있습니다. 자세한 내용은 쿡북과 레시피 단원을 참조하십시오.
2. 사용자 지정 계층을 생성합니다.
3. 적절한 수명 주기 이벤트에 레시피를 할당합니다.

그런 다음 계층에 인스턴스를 추가해 인스턴스를 시작하고 그 인스턴스에 앱을 배포합니다.

> **Important**
>
> 사용자 지정 계층의 인스턴스에 앱을 배포하려면 작업을 배포하고 배포한 작업을 계층의 Deploy 이벤트에 할당하는 레시피를 구현해야 합니다.

계층별 운영 체제 패키지 설치

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 12부터는 서로 다른 운영 체제에서 실행되는 계층에 패키지를 설치하려면 사용자 지정 레시피를 사용해야 합니다. 이 방법은 가장 유연하게 패키지를 설치하고 제어할 수 있는 방법입니다.

가령 RedHat, Ubuntu 및 Linux 운영 체제 Amazon 버전에서 실행되는 계층에 Apache를 설치하려 한다고 가정해 보십시오. RedHat 및 Amazon Linux용 Apache 패키지는 `httpd`라고 하지만 Ubuntu에서는 `apache2`라고 합니다.

패키지 이름의 차이를 해결하기 위해 다음 예제 레시피와 비슷한 구문을 사용할 수 있습니다. 이 레시피는 각 운영 체제에 적절한 Apache 패키지를 설치합니다. 이 예제는 Chef 설명서에 기반합니다.

```
package "Install Apache" do
   case node[:platform]
      when "redhat", "amazon"
         package_name "httpd"
      when "ubuntu"
         package_name "apache2"
   end
end
```

package 리소스를 사용하여 패키지를 관리하는 방법에 대한 자세한 내용은 Chef 설명서의 패키지 페이지를 참조하세요.

또는 Chef 레시피 DSL(Domain-Specific Language)의 value_for_platform 도우미 메서드를 사용하면 더 간단하게 같은 결과를 얻을 수 있습니다.

```
package "Install Apache" do
   package_name value_for_platform(
      ["redhat", "amazon"] => { "default" => "httpd" },
      ["ubuntu"] => { "default" => "apache2" }
   )
end
```

value_for_platform 도우미 메서드 사용에 대한 내용은 레시피 DSL에 대하여를 참조하세요.

인스턴스

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스는 Amazon EC2 인스턴스처럼 애플리케이션 서비스, 트래픽 밸런싱 등의 작업을 처리하는 컴퓨팅 리소스를 나타냅니다. 인스턴스의 운영 체제에는 몇몇 Linux 배포 중 하나 또는 Windows Server 2012 R2가 있을 수 있습니다.

다음 방법 중 하나로 스택에 인스턴스를 추가할 수 있습니다.

- AWS OpsWorks 스택을 사용하여 스택에 인스턴스를 추가합니다. 추가하는 인스턴스는 Amazon EC2 인스턴스를 나타냅니다.
- Linux 기반 스택의 경우, Amazon EC2로 생성한 인스턴스와 자체 하드웨어에서 실행되는 온프레미스 인스턴스를 포함하여 다른 곳에서 생성된 인스턴스를 등록할 수 있습니다.

  그런 다음 AWS OpsWorks Stacks를 사용하여 Stacks로 생성된 인스턴스와 거의 동일한 방식으로 이러한 인스턴스를 관리할 수 AWS OpsWorks 있습니다.

이 섹션에서는 AWS OpsWorks Stacks를 사용하여 인스턴스를 생성하고 관리하는 방법을 설명합니다.

주제

- AWS OpsWorks Stacks 인스턴스 사용
- AWS OpsWorks Stacks 외부에서 생성된 컴퓨팅 리소스 사용
- 인스턴스 구성 편집
- AWS OpsWorks 스택 인스턴스 삭제
- SSH를 사용하여 Linux 인스턴스에 로그인
- RDP를 사용하여 Windows 인스턴스에 로그인

AWS OpsWorks Stacks 인스턴스 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks를 사용하여 인스턴스를 생성하고 스택에 추가할 수 있습니다.

주제

- AWS OpsWorks 스택 운영 체제
- 계층에 인스턴스 추가
- 사용자 지정 AMI 사용

- 수동으로 24/7 인스턴스 시작, 중지 및 재부팅
- 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리

AWS OpsWorks 스택 운영 체제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Amazon 및 Ubuntu Linux 배포판과 Microsoft Windows Server를 포함한 여러 내장 운영 체제의 64비트 버전을 지원합니다. 다음은 몇 가지 일반 참고 사항입니다.

- 스택의 인스턴스는 Linux 또는 Windows에서 실행될 수 있습니다.

  단일 스택에서 인스턴스마다 Linux 버전 또는 배포가 다를 수 있지만 Linux 및 Windows 인스턴스를 혼용할 수는 없습니다.
- 사용자 지정 AMIs(Amazon Machine Image)를 사용할 수 있지만이 섹션의 주제에 설명된 AWS OpsWorks Stacks 지원 AMIs 중 하나를 기반으로 해야 합니다. 사용자 지정 또는 커뮤니티 생성 AMI에서 생성된 다른 운영 체제(예: CentOS 6.x)에서 인스턴스를 생성하거나 등록하는 것도 가능하지만 이런 운영 체제는 공식 지원되지 않습니다.
  - Linux 운영 체제
  - Microsoft Windows Server
- 수동으로 인스턴스를 시작 및 중지할 수도 있고, AWS OpsWorks Stacks가 인스턴스 수를 자동 조정하게 할 수도 있습니다.

  모든 스택에서 시간 기반 자동 조정을 사용할 수 있습니다. Linux 스택에서는 로드 기반 조정을 사용할 수도 있습니다.
- AWS OpsWorks Stacks를 사용하여 Amazon EC2 인스턴스를 생성하는 것 외에도 AWS OpsWorks Stacks 외부에서 생성된 Linux 스택에 인스턴스를 등록할 수도 있습니다.

  Amazon EC2 인스턴스와 자체 하드웨어에서 실행되는 인스턴스가 여기에 포함됩니다. 하지만 이러한 인스턴스는 지원되는 Linux 배포 중 하나를 실행해야 합니다. Amazon EC2 또는 온프레미스 Windows 인스턴스는 등록할 수 없습니다.

AWS OpsWorks Stacks DescribeOperatingSystems API를 실행하여 지원되는 운영 체제 목록과 지원되는 Chef 버전을 반환할 수 있습니다. 다음은 AWS CLI를 사용한 명령의 예입니다.

```
aws opsworks describe-operating-systems
```

다음은 응답의 예입니다.

```
{
    "OperatingSystems": [
        {
            "Name": "Amazon Linux",
            "Id": "Amazon Linux",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2",
            "Id": "Amazon Linux 2",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2"
```

```
        },
        {
            "Name": "Amazon Linux 2014.09",
            "Id": "Amazon Linux 2014.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.09",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2015.03",
            "Id": "Amazon Linux 2015.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
```

```
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2015.09",
            "Id": "Amazon Linux 2015.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.09",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2016.03",
            "Id": "Amazon Linux 2016.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
```

```
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.03"
        },
        {
            "Name": "Amazon Linux 2016.09",
            "Id": "Amazon Linux 2016.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.09"
        },
        {
            "Name": "Amazon Linux 2017.03",
            "Id": "Amazon Linux 2017.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
```

```
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.03"
        },
        {
            "Name": "Amazon Linux 2017.09",
            "Id": "Amazon Linux 2017.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.09"
        },
        {
```

```
            "Name": "Amazon Linux 2018.03",
            "Id": "Amazon Linux 2018.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2018.03"
        },
        {
            "Name": "CentOS Linux 7",
            "Id": "CentOS Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "CentOS Linux",
            "ReportedVersion": "7"
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 Base",
            "Id": "Microsoft Windows Server 2012 R2 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
```

```
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Express",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2019 Base",
            "Id": "Microsoft Windows Server 2019 Base",
```

```
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Express",
            "Id": "Microsoft Windows Server 2019 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Web",
            "Id": "Microsoft Windows Server 2019 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
```

```
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 Base",
            "Id": "Microsoft Windows Server 2022 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Express",
            "Id": "Microsoft Windows Server 2022 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
```

```
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Web",
            "Id": "Microsoft Windows Server 2022 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Red Hat Enterprise Linux 7",
            "Id": "Red Hat Enterprise Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "Red Hat Enterprise Linux",
            "ReportedVersion": "7"
        },
        {
            "Name": "Ubuntu 12.04 LTS",
            "Id": "Ubuntu 12.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
```

```
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "12.04",
            "Supported": false
        },
        {
            "Name": "Ubuntu 14.04 LTS",
            "Id": "Ubuntu 14.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "14.04"
        },
        {
            "Name": "Ubuntu 16.04 LTS",
            "Id": "Ubuntu 16.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "16.04"
        },
        {
            "Name": "Ubuntu 18.04 LTS",
```

```
            "Id": "Ubuntu 18.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "18.04"
        },
        {
            "Name": "Ubuntu 20.04 LTS",
            "Id": "Ubuntu 20.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "20.04"
        },
        {
            "Name": "Custom",
            "Id": "Custom",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
```

```
                }
            ]
        },
        {
            "Name": "CustomWindows",
            "Id": "CustomWindows",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ]
        }
    ]
}
```

주제

- Linux 운영 체제
- Microsoft Windows Server

Linux 운영 체제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 다음 Linux 운영 체제의 64비트 버전을 지원합니다.

- Amazon Linux 및 Amazon Linux 2(현재 지원되는 버전은 AWS OpsWorks Stacks 콘솔 참조)
- Ubuntu 20.04 LTS
- CentOS 7
- Red Hat Enterprise Linux 7

이러한 운영 체제를 기반으로 한 사용자 지정 AMI도 사용할 수 있습니다.

다음은 Linux 인스턴스에 대한 일반 참고 사항입니다.

지원되는 패키지 버전

패키지(예: Ruby)에 대해 지원되는 버전 및 패치 수준은 다음 단원에서 설명하는 대로 운영 체제 및 버전에 따라 다릅니다.

업데이트

기본적으로 AWS OpsWorks Stacks는 인스턴스 부팅 `apt-get update` 후 `yum update` 또는를 자동으로 호출하여 Linux 인스턴스에 최신 보안 패치가 적용되도록 합니다. 자동 업데이트를 비활성화하려면 CreateInstance, UpdateInstance, CreateLayer 또는 UpdateLayer 작업(또는 이에 상응하는 AWS SDK 메서드 또는 AWS CLI 명령)을 사용하여 `InstallUpdatesOnBoot` 파라미터를 `false`로 설정합니다.

서비스 중단을 방지하기 위해 AWS OpsWorks Stacks는 인스턴스가 온라인 상태가 된 후 업데이트를 자동으로 설치하지 않습니다. 언제라도 Upgrade Operating System 스택 명령을 실행하여 온라인 인스턴스의 운영 체제를 수동으로 업데이트할 수 있습니다. 보안 업데이트를 관리하는 방법에 대한 자세한 정보는 보안 업데이트 관리 단원을 참조하세요.

AWS OpsWorks Stacks가 인스턴스를 업데이트하는 방법을 더 잘 제어하려면 지원되는 운영 체제 중 하나를 기반으로 사용자 지정 AMI를 생성합니다. 예를 들어 사용자 지정 AMI를 사용하여 인스턴스에 설치되는 패키지 버전을 지정할 수 있습니다. 각 Linux 배포는 지원 일정 및 패키지-병합 정책이 서로 다르므로 요구 사항에 가장 적합한 접근 방식이 무엇인지 고려해야 합니다. 자세한 내용은 사용자 지정 AMI 사용 단원을 참조하십시오.

호스트 파일

각 온라인 인스턴스에는 IP 주소를 호스트 이름에 매핑하는 `/etc/hosts` 파일이 있습니다. AWS OpsWorks 스택에는 각 인스턴스`hosts`의 파일에 있는 스택의 모든 온라인 인스턴스에 대한 퍼블릭 및 프라이빗 주소가 포함됩니다. 예를 들어 2개의 Node.js 앱 서버 인스턴스, nodejs-app1 및 nodejs-app2와 하나의 MySQL 인스턴스, db-master1을 포함하는 스택을 가정해 봅시다. nodejs-app1 인스턴스의 `hosts` 파일은 다음 예제와 비슷하고, 다른 인스턴스의 `hosts` 파일도 비슷할 것입니다.

```
...
# OpsWorks Layer State
192.0.2.0 nodejs-app1.localdomain nodejs-app1
10.145.160.232 db-master1
```

```
198.51.100.0 db-master1-ext
10.243.77.78 nodejs-app2
203.0.113.0 nodejs-app2-ext
10.84.66.6 nodejs-app1
192.0.2.0 nodejs-app1-ext
```

AWS OpsWorks Stacks 에이전트 프록시 지원

Chef AWS OpsWorks 11.10 이상 스택용 Stacks 에이전트에는 일반적으로 격리된 VPCs. 프록시 서버 지원을 활성화하려면 인스턴스에 적절한 HTTP 및 HTTPS 트래픽 설정을 담은 `/etc/environment` 파일이 있어야 합니다. 이 파일은 다음 예제와 비슷하며, 여기서 강조 표시된 텍스트는 프록시 서버의 URL 및 포트로 대체됩니다.

```
http_proxy="http://myproxy.example.com:8080/"
https_proxy="http://myproxy.example.com:8080/"
no_proxy="169.254.169.254"
```

프록시 지원을 활성화하려면 적절한 `/etc/environment` 파일이 포함된 사용자 지정 AMI를 생성하고 이 AMI를 사용하여 인스턴스를 생성하는 것이 좋습니다.

> **Note**
>
> 사용자 지정 레시피를 사용하여 인스턴스에 `/etc/environment` 파일을 생성하는 것은 권장하지 않습니다. AWS OpsWorks 스택에는 설정 프로세스 초기에 사용자 지정 레시피가 실행되기 전에 프록시 서버 데이터가 필요합니다.

주제

- Amazon Linux
- Ubuntu LTS
- CentOS
- Red Hat Enterprise Linux

Amazon Linux

AWS OpsWorks Stacks는 Amazon Linux 및 Amazon Linux 2의 64비트 버전을 지원합니다. Amazon Linux는 정기적 업데이트 및 패치 이외에 약 6개월마다 상당한 변경이 포함된 새 버전을 릴리스합니다.

스택 또는 새 인스턴스를 생성할 때 사용할 Amazon Linux 버전을 지정해야 합니다. AWS에서 새 버전을 릴리스할 경우 인스턴스는 사용자가 명시적으로 변경하기 전에는 지정된 버전을 계속 실행합니다. 새 Amazon Linux 버전 릴리스 이후 4주일의 마이그레이션 기간이 있습니다. 이 기간 동안 AWS는 기존 버전에 대한 정기 업데이트를 계속 제공합니다. 마이그레이션 기간이 끝난 후에도 사용자의 인스턴스는 기존 버전을 계속 실행할 수 있지만, AWS에서 추가 업데이트를 제공하지는 않습니다. 자세한 내용은 Amazon Linux AMI FAQ를 참조하세요.

새 Amazon Linux 버전이 릴리스될 경우 마이그레이션 기간 이내에 새 버전으로 업데이트할 것을 권장합니다. 그러면 인스턴스가 보안 업데이트를 계속 받을 수 있습니다. 프로덕션 스택의 인스턴스를 업데이트하기 전에 새 인스턴스를 시작하고 앱이 새 버전에서 올바로 실행되는지 확인하는 것이 좋습니다. 그런 다음 프로덕션 스택 인스턴스를 업데이트할 수 있습니다.

> **Note**
>
> 기본적으로 Amazon Linux를 기반으로 한 사용자 지정 AMI는 새 버전이 릴리스될 경우 자동으로 업데이트됩니다. 권장되는 방법은 사용자 지정 AMI를 특정 Amazon Linux 버전에 고정시키는 것입니다. 그러면 새 버전 테스트를 마칠 때까지 업데이트를 연기할 수 있습니다. 자세한 내용은 내 AMI를 특정 버전으로 고정시키려면 어떻게 해야 합니까?를 참조하세요.
> AWS CloudFormation 템플릿을 사용하여 Amazon Linux를 실행하는 인스턴스가 있는 스택을 생성하는 경우 템플릿은 Amazon Linux 버전을 명시적으로 지정해야 합니다. 특히, 템플릿이 `Amazon Linux`를 지정할 경우, 인스턴스가 버전 2016.09를 계속 실행합니다. 자세한 내용은 AWS::OpsWorks::Stack 및 AWS::OpsWorks::Instance를 참조하세요.

인스턴스의 Amazon Linux 버전을 업데이트하려면 다음 중 한 가지를 수행합니다.

- 온라인 인스턴스의 경우, Upgrade Operating System 스택 명령을 실행합니다.

  새 Amazon Linux 버전이 사용 가능하면 [인스턴스] 및 [스택] 페이지에 [명령 실행] 페이지로 연결되는 링크와 함께 알림이 표시됩니다. 그러면 [운영 체제 업그레이드]를 실행하여 인스턴스를 업그레이드할 수 있습니다.
- 오프라인 Amazon Elastic Block Store 지원(EBS 지원) 인스턴스의 경우, 인스턴스를 시작하고 위에서 설명한 대로 운영 체제 업그레이드을 실행합니다.
- 오프라인 인스턴스 스토어 지원 인스턴스의 경우, 인스턴스의 [운영 체제] 설정을 편집하여 새 버전을 지정합니다.

  AWS OpsWorks Stacks는 인스턴스를 다시 시작할 때 새 버전으로 자동으로 업데이트합니다.

Amazon Linux: 지원되는 Node.js 버전

Amazon Linux 버전	Node.js 버전
2	(Not applicable to operating systems that are available for Chef 12 and higher stacks only)
2018.03	0.12.18
2017.09	0.12.18
2017.03	0.12.18
2016.09	0.12.18 0.12.17 0.12.16 0.12.15
2016.03	0.12.18 0.12.17 0.12.16 0.12.15 0.12.14 0.12.13 0.12.12 0.12.10

Amazon Linux: 지원되는 Chef 버전

Chef 버전	지원되는 Amazon Linux 버전
12	Amazon Linux 2 Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09

Chef 버전	지원되는 Amazon Linux 버전
	Amazon Linux 2016.03
11.10	Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.4 (deprecated)	Amazon Linux 2016.09 Amazon Linux 2016.03

Important

t1.micro 인스턴스를 업데이트하기 전에 인스턴스에 임시 스왑 파일 /var/swapfile이 있는지 확인하세요. Chef 0.9 스택의 t1.micro 인스턴스에는 스왑 파일이 없습니다. Chef 11.4 및 Chef 11.10 스택의 경우 최근 버전의 인스턴스 에이전트가 t1.micro 인스턴스용 스왑 파일을 자동으로 생성합니다. 하지만 이 변경은 몇 주간에 걸쳐 도입되었기 때문에 대략 2014년 3월 24일 이전에 생성된 인스턴스에서는 /var/swapfile이 있는지 확인해야 합니다.
t1.micro 인스턴스에 스왑 파일이 없을 경우 다음과 같이 스왑 파일을 생성할 수 있습니다.

- Chef 11.10 이상 스택의 경우 새 t1.micro 인스턴스를 생성합니다. 그러면 자동으로 스왑 파일이 생성됩니다.
- Chef 0.9 스택의 경우, 각 인스턴스에서 루트 사용자로 다음 명령을 실행합니다.

```
dd if=/dev/zero of=/var/swapfile bs=1M count=256
 mkswap /var/swapfile
 chown root:root /var/swapfile
 chmod 0600 /var/swapfile
 swapon /var/swapfile
```

새 인스턴스를 생성하지 않으려는 경우 Chef 11.10 이상 스택에서 이러한 명령을 사용할 수도 있습니다.

Ubuntu LTS

Ubuntu는 약 2년마다 새 Ubuntu LTS 버전을 릴리스하며, 각 릴리스를 약 5년간 지원합니다. Ubuntu는 운영 체제 지원 기간 동안 보안 패치 및 업데이트를 제공합니다. 자세한 정보는 LTS - Ubuntu Wiki를 참조하세요.

- 기존 Ubuntu 인스턴스를 최신 Ubuntu 릴리스로 업데이트할 수 없습니다.

  새 Ubuntu 인스턴스를 생성한 다음 기존 인스턴스를 삭제해야 합니다.
- Ubuntu 20.04 LTS는 Chef 12 이상 스택에서만 지원됩니다.

CentOS

AWS OpsWorks Stacks는 CentOS 7의 64비트 버전을 지원합니다. 버전 CentOS 7부터 지원되며 CentOS는 약 2년마다 새 버전을 릴리스합니다.

CentOS 스택에서 새 인스턴스를 시작하면 AWS OpsWorks Stacks는 최신 CentOS 버전을 자동으로 설치합니다. AWS OpsWorks Stacks는 새 CentOS 마이너 버전이 릴리스될 때 기존 인스턴스에서 운영 체제를 자동으로 업데이트하지 않으므로 새로 생성된 인스턴스는 스택의 기존 인스턴스보다 최신 버전을 수신할 수 있습니다. 스택에서 버전 일관성을 유지하려면 다음과 같이 기존 인스턴스를 현재 CentOS 버전으로 업데이트할 수 있습니다.

- 온라인 인스턴스의 경우, Upgrade Operating System 스택 명령을 실행합니다. 그러면 지정된 인스턴스에서 `yum update`가 실행되어 인스턴스를 최신 버전으로 업데이트합니다.

  새 CentOS 7 마이너 버전이 사용 가능하면 [인스턴스] 및 [스택] 페이지에 [명령 실행] 페이지로 연결되는 링크와 함께 알림이 표시됩니다. 그러면 [운영 체제 업그레이드]을 실행하여 인스턴스를 업그레이드할 수 있습니다.
- 오프라인 Amazon EBS 지원 인스턴스의 경우, 인스턴스를 시작하고 위에서 설명한 대로 운영 체제 업그레이드을 실행합니다.
- 오프라인 인스턴스 스토어 지원 인스턴스의 경우 인스턴스가 다시 시작될 때 AWS OpsWorks Stacks가 새 버전을 자동으로 설치합니다.

CentOS: 지원되는 Chef 버전

Chef 버전	지원되는 CentOS 버전
`12`	`CentOS 7`

Chef 버전	지원되는 CentOS 버전
`11.10`	`(None supported)`
`11.4 (deprecated)`	`(None supported)`

> **Note**
>
> AWS OpsWorks Stacks는 CentOS 인스턴스용 Apache 2.4를 지원합니다.

Red Hat Enterprise Linux

AWS OpsWorks Stacks는 Red Hat Enterprise Linux 7(RHEL 7)의 64비트 버전을 지원합니다. 버전 RHEL 7.1부터 지원되며 Red Hat은 약 9개월마다 새 마이너 버전을 릴리스합니다. 마이너 버전이 RHEL 7.0과 호환되어야 합니다. 자세한 정보는 수명 주기 및 업데이트 정책을 참조하세요.

새 인스턴스를 시작하면 AWS OpsWorks Stacks가 현재 RHEL 7 버전을 자동으로 설치합니다. 새 RHEL AWS OpsWorks 7 마이너 버전이 릴리스될 때 Stacks가 기존 인스턴스의 운영 체제를 자동으로 업데이트하지 않으므로 새로 생성된 인스턴스는 스택의 기존 인스턴스보다 최신 버전을 수신할 수 있습니다. 스택에서 버전 일관성을 유지하려면 다음과 같이 기존 인스턴스를 현재 RHEL 7 버전으로 업데이트할 수 있습니다.

- 온라인 인스턴스의 경우, Upgrade Operating System 스택 명령을 실행합니다. 그러면 지정된 인스턴스에서 `yum update`가 실행되어 인스턴스를 최신 버전으로 업데이트합니다.

  새 RHEL 7 버전이 사용 가능하면 [인스턴스] 및 [스택] 페이지에 [명령 실행] 페이지로 연결되는 링크와 함께 알림이 표시됩니다. 그러면 [운영 체제 업그레이드]을 실행하여 인스턴스를 업그레이드할 수 있습니다.
- 오프라인 Amazon EBS 지원 인스턴스의 경우, 인스턴스를 시작하고 위에서 설명한 대로 운영 체제 업그레이드을 실행합니다.
- 오프라인 인스턴스 스토어 지원 인스턴스의 경우 인스턴스가 다시 시작될 때 AWS OpsWorks Stacks가 새 버전을 자동으로 설치합니다.

Red Hat Enterprise Linux: 지원되는 Node.js 버전

RHEL 버전	Node.js 버전
7	(Node.js versions only apply to Chef 11.10 stacks) 0.8.19 0.8.26 0.10.11 0.10.21 0.10.24 0.10.25 0.10.27 0.10.29 0.10.40 0.12.10 0.12.12 0.12.13 0.12.15

Red Hat Enterprise Linux: 지원되는 Chef 버전

Chef 버전	지원되는 RHEL 버전
12	Red Hat Enterprise Linux 7
11.10	Red Hat Enterprise Linux 7
11.4 (deprecated)	(None supported)

0.10.40 이전의 모든 Node.js 버전은 더 이상 사용되지 않으며 0.12.7 및 0.12.9도 더 이상 사용되지 않습니다.

> **Note**
>
> AWS OpsWorks Stacks는 RHEL 7 인스턴스에 대해 Apache 2.4를 지원합니다.

Microsoft Windows Server

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

다음 참고에서는 Windows 인스턴스에 대한 AWS OpsWorks Stacks 지원을 설명합니다. Windows 인스턴스는 Chef 12.2 스택에만 사용할 수 있습니다. Windows 스택에서 Chef의 정확한 버전은 12.22입니다.

현재 AWS OpsWorks Stacks 에이전트는 영어 - 미국(en-US) 이외의 시스템 UI 언어를 사용하는 Windows 기반 인스턴스에 설치할 수 없으며 AWS OpsWorks Stacks는 관리할 수 없습니다.

버전

AWS OpsWorks Stacks는 다음 Windows 64비트 버전을 지원합니다.

- Microsoft Windows Server 2022 Base
- Microsoft Windows Server 2022 with SQL Server Express
- Microsoft Windows Server 2022 with SQL Server Standard
- Microsoft Windows Server 2022 with SQL Server Web
- Microsoft Windows Server 2019 Base
- Microsoft Windows Server 2019 with SQL Server Express
- Microsoft Windows Server 2019 with SQL Server Standard
- Microsoft Windows Server 2019 with SQL Server Web

인스턴스 생성

AWS OpsWorks Stacks 콘솔, API 또는 CLI를 사용하여 Windows 인스턴스를 생성합니다. Windows 인스턴스는 Amazon EBS 지원이지만 추가 Amazon EBS 볼륨을 탑재할 수 없습니다.

Windows 스택은 사용자가 수동으로 시작하고 중지할 수 있는 24/7 인스턴스를 사용할 수 있습니다. 또한 사용자가 지정한 일정에 따라 자동으로 인스턴스를 시작하고 중지하는 시간 기반 자동 조정도 사용할 수 있습니다. Windows 기반 스택은 로드 기반 자동 조정을 사용할 수 없습니다.

AWS OpsWorks 스택 외부에서 생성된 Windows 인스턴스는 스택에 등록할 수 없습니다.

업데이트

AWS는 각 패치 세트에 대해 Windows AMI를 업데이트합니다. 따라서 사용자가 인스턴스를 생성할 때 최신 업데이트가 적용됩니다. 그러나 AWS OpsWorks Stacks는 온라인 Windows 인스턴스에 업데이트를 적용하는 방법을 제공하지 않습니다. Windows가 최신 상태를 유지하도록 하는 가장 간편한 방법은 인스턴스가 항상 최신 AMI를 실행하도록 정기적으로 인스턴스를 교체하는 것입니다.

계층

소프트웨어 설치 및 구성 또는 앱 배포와 같은 작업을 처리하기 위해 사용자 지정 레시피를 포함하는 사용자 지정 계층을 하나 이상 생성해야 합니다.

Chef

Windows 인스턴스는 Chef 12.22를 사용하며 Chef-client를 로컬 모드에서 실행합니다. 그러면 chef-zero라는 로컬 인메모리 Chef 서버가 시작됩니다. 이 서버가 존재하면 사용자 지정 레시피가 Chef 검색 및 데이터 백을 사용할 수 있습니다.

원격 로그인

AWS OpsWorks Stacks는 권한 있는 IAM 사용자에게 Windows 인스턴스에 로그인하는 데 사용할 수 있는 암호를 제공합니다. 이 암호는 지정된 시간 이후 만료됩니다. 관리자는 SSH 키 페어를 사용하여 인스턴스의 관리자 암호를 검색할 수 있습니다. 이 암호는 제한이 없는 RDP 액세스를 제공합니다. 자세한 내용은 RDP를 사용하여 로그인 단원을 참조하십시오.

AWS SDK

AWS OpsWorks Stacks는 각 인스턴스AWS SDK for .NET에를 자동으로 설치합니다. 이 패키지에는 AWS .NET 라이브러리와 AWS Tools for PowerShell 비롯한 Windows용 AWS 도구가 포함되어 있습니다. Ruby SDK를 사용하려면 사용자 지정 레시피가 적절한 젬을 설치하도록 할 수 있습니다.

모니터링 및 지표

Windows 인스턴스는 표준 Amazon CloudWatch(CloudWatch) 지표를 지원합니다. 이 지표는 CloudWatch 콘솔에서 볼 수 있습니다.

Ruby

Stacks가 Windows 인스턴스에 설치하는 Chef AWS OpsWorks 12.22 클라이언트에는 Ruby 2.3.6이 함께 제공됩니다. 그러나 AWS OpsWorks Stacks는 실행 파일의 디렉터리를 PATH 환경 변수에 추가하지 않습니다. 애플리케이션에서 이 Ruby 버전을 사용하도록 하려면 일반적으로 `C:\opscode\chef\embedded\bin\`에서 이 버전을 찾을 수 있습니다.

AWS OpsWorks Stacks 에이전트 CLI

Windows 인스턴스의 AWS OpsWorks Stacks 에이전트는 명령줄 인터페이스를 노출하지 않습니다.

프록시 지원

Windows 인스턴스에 대한 프록시 지원을 설정하려면 다음을 수행합니다.

1. 를 수정`machine.config`하여 다음을 추가하면 Windows PowerShell(초기 부트스트랩) 및 .NET(AWS OpsWorks 스택 에이전트) 애플리케이션에 프록시 지원이 추가됩니다.

```
<system.net>
  <defaultProxy>
    <proxy autoDetect="false" bypassonlocal="true"
 proxyaddress="http://10.100.1.91:3128"  usesystemdefault="false" />
    <bypasslist>
      <add address="localhost" />
      <add address="169.254.169.254" />
    </bypasslist>
  </defaultProxy>
</system.net>
```

2. 다음 명령을 실행하여 나중에 Chef 및 Git에서 사용할 환경 변수를 설정합니다.

```
setx /m no_proxy "localhost,169.254.169.254"
setx /m http_proxy "http://10.100.1.91:3128"
setx /m https_proxy "http://10.100.1.91:3128"
```

> **Note**
>
> Stacks가 인스턴스를 업데이트하는 방법을 더 잘 제어하려면 Microsoft Windows Server AWS OpsWorks 2022 Base를 기반으로 사용자 지정 AMI를 생성합니다. 예를 들어 사용자 지정 AMI를 사용하여 인스턴스에 설치될 소프트웨어를 지정할 수 있습니다(예: 웹 서버(IIS)). 자세한 내용은 사용자 지정 AMI 사용 단원을 참조하십시오.

계층에 인스턴스 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계층을 생성한 후 일반적으로 하나 이상의 인스턴스를 추가합니다. 현재의 인스턴스 세트가 부하를 처리하지 못하는 경우, 나중에 더 많은 인스턴스를 추가할 수 있습니다. 로드 기반 또는 시간 기반 인스턴스를 사용하여 인스턴스 수를 자동으로 조정할 수도 있습니다.

새 인스턴스나 기존 인스턴스를 계층에 추가할 수 있습니다.

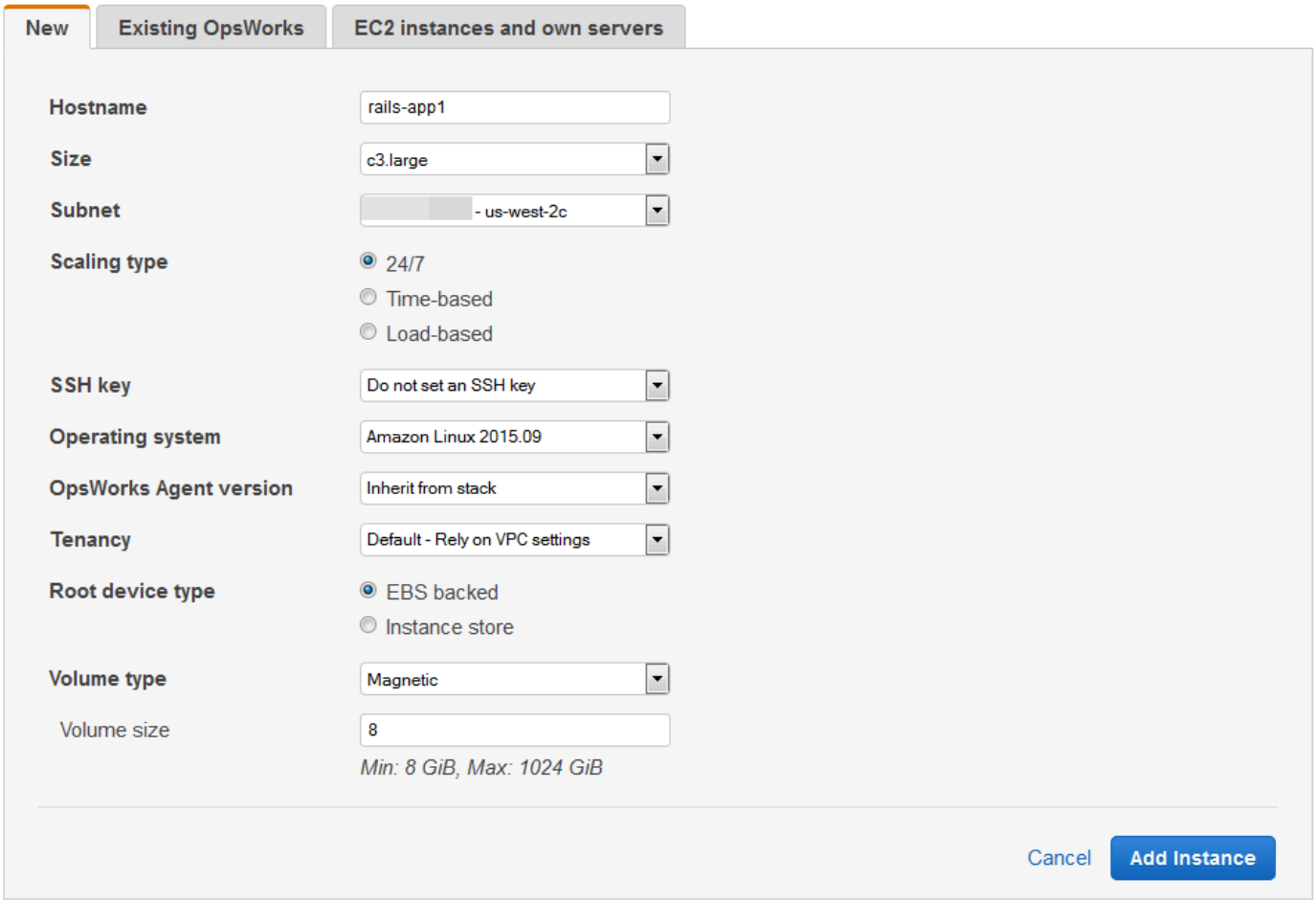
- 새로 만들기-OpsWorks가 사양에 맞게 구성된 새 인스턴스를 생성하여 계층의 구성원으로 만듭니다.
- 기존-호환되는 계층의 기존 인스턴스를 추가할 수 있지만 인스턴스가 오프라인(중지) 상태여야 합니다.

인스턴스가 여러 계층에 속하는 경우, AWS OpsWorks Stacks는 수명 주기 이벤트가 발생하거나 스택 또는 배포 명령을 실행할 때 인스턴스의 각 계층에 대해 레시피를 실행합니다.

인스턴스의 구성을 편집하여 인스턴스를 여러 계층의 구성원으로 만들 수도 있습니다. 자세한 내용은 인스턴스 구성 편집 단원을 참조하십시오.

새 인스턴스를 계층에 추가하려면

1. [인스턴스] 페이지에서 해당 계층에 대해 [+인스턴스]를 선택하고 (필요한 경우) [새로 만들기] 탭을 선택합니다. [호스트 이름], [크기] 및 [서브넷] 또는 [가용 영역] 이외의 다른 옵션을 구성하려면 [고급 >>]를 선택하여 옵션을 추가로 표시합니다. 다음 그림에는 전체 옵션 세트가 나와 있습니다.



2. 원한다면 기본 구성을 재정의할 수 있으며, 기본 구성은 대부분 스택을 생성할 때 지정한 것입니다. 자세한 내용은 새 스택 생성 단원을 참조하십시오.

   Hostname

   네트워크에서 인스턴스를 식별합니다. 기본적으로 AWS OpsWorks Stacks는 스택을 생성할 때 지정한 호스트 이름 테마를 사용하여 각 인스턴스의 호스트 이름을 생성합니다. 이 값을 재정의하여 선호하는 호스트 이름을 지정할 수 있습니다.

   크기

   메모리 양 또는 가상 코어 수와 같은 인스턴스의 리소스를 지정하는 Amazon EC2 인스턴스 유형입니다. AWS OpsWorks 스택은 각 인스턴스의 기본 크기를 지정하며, 원하는 인스턴스 유형으로 재정의할 수 있습니다.

   AWS OpsWorks Stacks에서 지원하는 인스턴스 유형은 스택이 VPC에 있는지 여부에 따라 달라집니다. 또한 계정에서 AWS 프리 티어를 사용하는 경우 인스턴스 유형은 제한됩니다. 크기 드롭다운 목록은 스택이 지원하는 Chef 버전에 대해 지원되는 인스턴스 유형을 표시합니다.

t1.micro와 같은 마이크로 인스턴스에는 일부 계층을 지원하기에 충분한 리소스가 없을 수 있습니다. 자세한 내용은 인스턴스 유형을 참조하세요.

> **Note**
>
> 로드밸런싱된 인스턴스를 사용 중인 경우, 수명 주기 이벤트 구성을 수행하면 상당한 CPU 부하가 급격히 증가해 1분 이상 이어질 수 있습니다. 인스턴스 크기가 작을 때는 이러한 부하 급증으로 인해 확장이 트리거될 수 있고, Configure 이벤트가 빈번하게 발생하는 로드 밸런싱된 대형 스택의 경우 특히 더 그렇습니다. 다음은 불필요한 확장을 일으키는 Configure 이벤트의 발생 가능성을 줄이기 위한 몇 가지 방법입니다.
>
> - 더 큰 인스턴스 유형을 사용하여 Configure 이벤트의 추가 부하로 인해 확장이 트리거되지 않게 합니다.
> - CPU 리소스를 공유하는 T2와 같은 인스턴스 유형을 사용하지 마십시오.
>
> 이렇게 하면 Configure 이벤트 발생 시 인스턴스의 모든 CPU 리소스를 즉시 사용할 수 있습니다.
>
> - `exceeded threshold` 시간을 Configure 이벤트를 처리하는 데 필요한 시간(대략 5분)보다 훨씬 길게 설정합니다.
>
> 자세한 내용은 자동 로드 기반 조정 사용 단원을 참조하십시오.

가용 영역/서브넷

스택이 VPC에 없는 경우 이 설정은 [가용 영역]으로 라벨링되고 리전의 가용 영역을 나열합니다. 이 설정을 사용하여 스택을 생성할 때 지정한 기본 가용 영역을 재정의할 수 있습니다.

스택이 VPC에서 실행 중인 경우 이 설정은 [서브넷]으로 라벨링되고 VPC의 서브넷을 나열합니다. 이 설정을 사용하여 스택을 생성할 때 지정한 기본 서브넷을 재정의할 수 있습니다.

> **Note**
>
> 기본적으로 AWS OpsWorks Stacks는 서브넷의 CIDR 범위를 나열합니다. 목록을 더 읽기 쉽게 만들려면 VPC 콘솔 또는 API를 사용하여 키가 로, **`Name`** 값이 서브넷 이름으로 설정된 각 서브넷에 태그를 추가합니다. AWS OpsWorks Stacks는 해당 이름을 CIDR 범위에 추가합니다. 앞선 예제에서는 서브넷의 이름 태그를 **`Private`**으로 설정했습니다.

조정 유형

인스턴스가 시작 및 중지되는 방법을 결정합니다.

- 기본값은 사용자가 수동으로 시작하고 중지할 수 있는 24/7 인스턴스입니다.
- AWS OpsWorks Stacks는 지정된 일정에 따라 시간 기반 인스턴스를 시작하고 중지합니다.
- (Linux만 해당) AWS OpsWorks Stacks는 지정된 로드 지표를 기반으로 로드 기반 인스턴스를 시작하고 중지합니다.

> **Note**
>
> 로드 기반 또는 시간 기반 인스턴스는 사용자가 직접 시작하거나 중지할 수 없습니다. 대신 인스턴스를 구성하면 AWS OpsWorks Stacks가 구성에 따라 인스턴스를 시작하고 중지합니다. 자세한 내용은 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리 단원을 참조하십시오.

SSH 키

Amazon EC2 키 페어. AWS OpsWorks Stacks는 인스턴스에 퍼블릭 키를 설치합니다.

- Linux 인스턴스의 경우, SSH 클라이언트와 함께 해당 프라이빗 키를 사용하여 인스턴스에 로그인할 수 있습니다.
- Windows 인스턴스의 경우 해당하는 프라이빗 키를 사용하여 인스턴스의 관리자 암호를 검색할 수 있습니다. 그런 다음 이 암호를 RDP와 함께 사용하여 인스턴스에 Administrator로 로그인할 수 있습니다.

처음에 이 설정은 스택을 생성할 때 지정한 기본 SSH 키 값입니다.

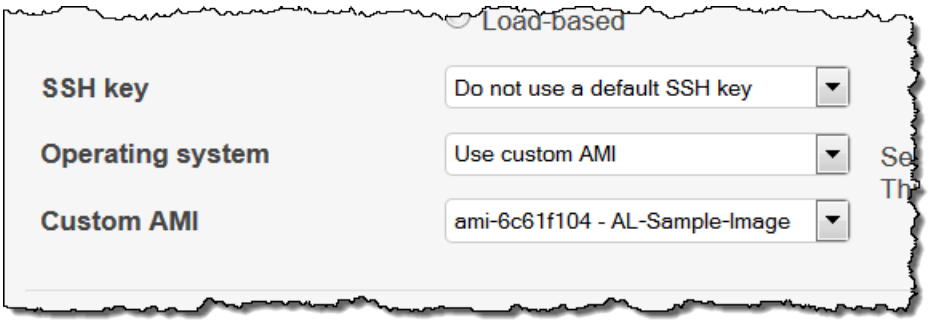
- 기본값이 기본 SSH 키 사용 안 함으로 설정되어 있으면 계정의 Amazon EC2 키 하나를 지정할 수 있습니다.
- 기본값이 Amazon EC2 키로 설정되어 있으면 다른 키를 지정하거나 아무 키도 지정하지 않을 수 있습니다.

운영 체제

운영 체제는 인스턴스가 실행 중인 운영 체제를 지정합니다. AWS OpsWorks Stacks는 64비트 운영 체제만 지원합니다.

처음에 이 설정은 스택을 생성할 때 지정한 기본 운영 체제 값입니다. 기본값을 재정의해 다른 Linux 운영 체제 또는 사용자 지정 Amazon 머신 이미지(AMI)를 지정할 수 있습니다. 그러나 Linux에서 Windows로, 또는 Windows에서 Linux로 전환할 수는 없습니다.

[사용자 지정 AMI 사용]을 선택하면 페이지에 [아키텍처] 및 [루트 디바이스 유형] 대신 사용자 지정 AMI 목록이 표시됩니다.



자세한 내용은 사용자 지정 AMI 사용 단원을 참조하십시오.

OpsWorks 에이전트 버전

OpsWorks 에이전트 버전은 인스턴스에서 실행할 AWS OpsWorks Stacks 에이전트의 버전을 지정합니다. AWS OpsWorks Stacks가 에이전트를 자동으로 업데이트하도록 하려면 스택에서 상속을 선택합니다. 에이전트의 특정 버전을 설치하고 인스턴스에서 해당 에이전트를 수동으로 업데이트하려면 드롭다운 목록에서 버전을 선택합니다.

> **Note**
>
> 일부 운영 체제 버전에서 작동하지 않는 에이전트 버전도 있습니다. 인스턴스가 에이전트를 실행 중이거나 인스턴스 운영 체제에서 완전히 지원되지 않는 인스턴스에 에이전트를 설치하는 경우 AWS OpsWorks Stacks 콘솔에 호환되는 에이전트를 설치하라는 오류 메시지가 표시됩니다.

Tenancy

인스턴스에 대한 테넌시 옵션을 선택합니다. 전용 물리적 서버에서 사용할 인스턴스를 실행하도록 선택할 수 있습니다.

- [기본값 - VPC 설정 사용]. 테넌시가 없거나 VPC에서 테넌시 설정을 상속합니다.

- [전용 - 전용 인스턴스 실행]. 단일 테넌트 하드웨어에서 실행되는 인스턴스에 대한 비용을 시간 단위로 지불합니다. 자세한 정보는 Amazon VPC 사용 설명서의 전용 인스턴스와 Amazon EC2 전용 인스턴스를 참조하세요.
- [전용 호스트 -전용 호스트에서 이 인스턴스 실행]. 인스턴스 실행을 전담하는 실제 호스트 비용을 지불하며, 기존의 소켓, 코어 또는 VM 소프트웨어별 라이선스를 가져와 비용을 절감합니다. 자세한 내용은 Amazon EC2 설명서의 전용 호스트 개요 및 Amazon EC2 전용 호스트를 참조하세요.

루트 디바이스 유형

인스턴스의 루트 디바이스 스토리지를 지정합니다.

- Linux 인스턴스는 Amazon EBS 지원 인스턴스 또는 인스턴스 스토어 지원 인스턴스일 수 있습니다.
- Windows 인스턴스는 Amazon EBS 지원 인스턴스여야 합니다.

자세한 내용은 스토리지를 참조하세요.

> **Note**
>
> AWS OpsWorks Stacks가 인스턴스의 소프트웨어를 처음부터 다시 설치할 필요가 없으므로 초기 부팅 후 Amazon EBS 지원 인스턴스가 인스턴스 스토어 지원 인스턴스보다 빠르게 부팅됩니다. 자세한 내용은 루트 디바이스 스토리지 단원을 참조하십시오.

볼륨 유형

[마그네틱], [프로비저닝된 IOPS(SSD)] 또는 [범용(SSD)] 중에서 루트 디바이스 볼륨 유형을 지정합니다. 자세한 내용은 Amazon EBS 볼륨 유형을 참조하세요.

볼륨 크기

지정한 볼륨 유형에 대한 루트 디바이스 볼륨 크기를 지정합니다. 자세한 내용은 Amazon EBS 볼륨 유형을 참조하세요.

- 범용(SSD). 허용되는 최소 크기는 8GiB이고 최대 크기는 16384GiB입니다.
- 프로비저닝된 IOPS(SSD). 허용되는 최소 크기는 8GiB이고 최대 크기는 16384GiB입니다. 최소 100개의 IOPS(초당 입/출력 작업)와 최대 240개의 IOPS를 설정할 수 있습니다.
- 마그네틱. 허용되는 최소 크기는 8GiB이고 최대 크기는 1024GiB입니다.

3. [인스턴스 추가]를 선택하여 새 인스턴스를 생성합니다.

> **Note**
>
> 인스턴스를 생성할 때 스택의 기본 에이전트 버전 설정을 재정의할 수 없습니다. 사용자 지정 에이전트 버전 설정을 지정하려면 인스턴스를 생성한 다음 인스턴스의 구성을 편집해야 합니다.

계층에 기존 인스턴스를 추가하려면

1. [인스턴스] 페이지에서 해당 계층에 대해 [+인스턴스]를 선택하고 [기존] 탭을 엽니다.

   > **Note**
   >
   > 마음을 바꿔 기존 인스턴스를 사용하지 않으려면 [New]를 선택하여 앞의 절차에서 설명한 대로 새 인스턴스를 생성합니다.

2. [기존] 탭의 목록에서 인스턴스를 선택합니다.
3. [인스턴스 추가]를 선택하여 새 인스턴스를 생성합니다.

인스턴스는 Amazon EC2 인스턴스를 나타내지만 기본적으로 AWS OpsWorks Stacks 데이터 구조일 뿐입니다. 앞의 섹션에서 설명한 것처럼 실행 중인 Amazon EC2 인스턴스를 생성하려면 인스턴스를 시작해야 합니다.

> **Important**
>
> 인스턴스를 기본 VPC로 시작하는 경우, VPC 구성을 수정하는 데 주의해야 합니다. 인스턴스는 항상 AWS OpsWorks Stacks 서비스, Amazon S3 및 패키지 리포지토리와 통신할 수 있어야 합니다. 예를 들어 기본 게이트웨이를 제거하면 인스턴스가 AWS OpsWorks Stacks 서비스에 대한 연결이 끊어지고, 그러면 인스턴스가 실패로 처리되어 자동으로 복구됩니다. 하지만 AWS OpsWorks Stacks는 복구된 인스턴스에 인스턴스 에이전트를 설치할 수 없습니다. 에이전트가 없으면 인스턴스는 서비스와 통신할 수 없으며, 시작 프로세스는 `booting` 상태를 넘어 진행될 수 없습니다. 기본 VPC에 대한 자세한 내용은 지원되는 플랫폼을 참조하세요.

Linux 컴퓨팅 리소스를 Stacks 외부에서 생성된 AWS OpsWorks 스택에 통합할 수도 있습니다.

- Amazon EC2 콘솔, CLI 또는 API를 사용하여 직접 생성한 Amazon EC2 인스턴스.
- 가상 머신에서 실행되는 인스턴스를 비롯하여 자체 하드웨어에서 실행되는 온프레미스 인스턴스.

자세한 내용은 AWS OpsWorks Stacks 외부에서 생성된 컴퓨팅 리소스 사용 단원을 참조하십시오.

사용자 지정 AMI 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 사용자 지정 Amazon Machine Image(AMIs)와 Chef 레시피라는 두 가지 방법으로 인스턴스를 사용자 지정할 수 있습니다. 두 방법 모두 설치되는 패키지와 패키지 버전 및 구성 방식 등을 제어할 수 있습니다. 다만 각 방법은 장점이 서로 다르므로 요구 사항에 따라 가장 좋은 방법이 달라집니다.

사용자 지정 AMI 사용을 고려해야 하는 주된 이유는 다음과 같습니다.

- 특정 패키지를 인스턴스 부팅 후 설치하는 대신 사전 번들하려고 하는 경우.
- 계층의 일관된 기본 이미지를 제공하기 위해 패키지 업데이트 타이밍을 제어하려고 하는 경우.
- 인스턴스(특히 로드 기반 인스턴스)가 최대한 빨리 부팅되기를 원하는 경우.

Chef 레시피 사용을 고려해야 하는 주된 이유는 다음과 같습니다.

- Chef 레시피는 사용자 지정 AMI보다 유연합니다.
- 업데이트가 보다 쉽습니다.
- 실행 중인 인스턴스에서 업데이트를 수행할 수 있습니다.

실제로는 두 방법의 조합이 최적의 해법일 수 있습니다. 레시피에 대한 자세한 내용은 쿡북과 레시피 단원을 참조하세요.

주제

- 사용자 지정 AMIs AWS OpsWorks Stacks에서 작동하는 방식
- AWS OpsWorks Stacks용 사용자 지정 AMI 생성

사용자 지정 AMIs AWS OpsWorks Stacks에서 작동하는 방식

인스턴스에 대한 사용자 지정 AMI를 지정하려면 새 인스턴스를 생성할 때 사용자 지정 AMI를 인스턴스의 운영 체제로 사용을 선택합니다. 그러면 AWS OpsWorks 스택이 스택의 리전에 사용자 지정 AMIs 목록을 표시하고 목록에서 적절한 AMI를 선택합니다. 자세한 내용은 계층에 인스턴스 추가 단원을 참조하십시오.

> **Note**
>
> 특정 사용자 지정 AMI를 스택의 기본 운영 체제로 지정할 수는 없습니다. `Use custom AMI`를 스택의 기본 운영 체제로 지정할 수 있지만 새 인스턴스를 계층에 추가할 때만 특정 AMI를 지정할 수 있습니다. 자세한 내용은 계층에 인스턴스 추가 및 새 스택 생성 단원을 참조하세요. 사용자 지정 또는 커뮤니티 생성 AMI에서 생성된 다른 운영 체제(예: CentOS 6.x)에서 인스턴스를 생성하는 것도 가능하지만 이런 운영 체제는 공식 지원되지 않습니다.

이 주제에서는 사용자 지정 AMI를 생성하거나 사용하기 전에 고려해야 하는 몇 가지 일반적인 문제를 살펴봅니다.

주제

- 시작 동작
- 계층 선택
- 애플리케이션 처리

시작 동작

인스턴스를 시작하면 AWS OpsWorks Stacks는 지정된 사용자 지정 AMI를 사용하여 새 Amazon EC2 인스턴스를 시작합니다. 그런 다음 AWS OpsWorks Stacks는 cloud-init를 사용하여 인스턴스에 AWS OpsWorks Stacks 에이전트를 설치하고 에이전트는 인스턴스의 설정 레시피와 배포 레시피를 실행합니다. 인스턴스가 온라인 상태가 된 후 에이전트는 새로 추가된 인스턴스를 포함하여 스택의 모든 인스턴스에 대해 Configure 레시피를 실행합니다.

계층 선택

AWS OpsWorks Stacks 에이전트는 일반적으로 설치된 패키지와 충돌하지 않습니다. 그러나 인스턴스는 하나 이상의 계층의 멤버여야 합니다. AWS OpsWorks 스택은 항상 해당 계층의 레시피를 실행하므로 문제가 발생할 수 있습니다. 사용자 지정 AMI가 있는 인스턴스를 계층에 추가하기 전에 항상 해당 계층의 레시피가 하는 일이 무엇인지 정확히 이해해야 합니다.

특정 계층 유형이 인스턴스에서 어떤 레시피를 실행하는지 보려면 해당 계층이 포함된 스택을 엽니다. 그런 다음 탐색 창에서 [계층]를 클릭하고 관심 있는 계층의 [레시피]를 클릭합니다. 실제 코드를 보려면 레시피 이름을 클릭합니다.

> **Note**
>
> Linux AMIs의 경우 충돌 가능성을 줄이는 한 가지 방법은 AWS OpsWorks Stacks를 사용하여 사용자 지정 AMI의 기반인 인스턴스를 프로비저닝하고 구성하는 것입니다. 자세한 내용은 AWS OpsWorks Stacks 인스턴스에서 사용자 지정 Linux AMI 생성 단원을 참조하십시오.

애플리케이션 처리

AMI에 패키지 외에 애플리케이션을 포함시킬 수도 있습니다. 크고 복잡한 애플리케이션이 있는 경우, AMI에 포함시키면 인스턴스의 시작 시간이 단축될 수 있습니다. AMI에 작은 애플리케이션을 포함할 수 있지만 AWS OpsWorks Stacks가 애플리케이션을 배포하는 것과 관련된 시간 이점은 거의 없거나 전혀 없습니다.

한 가지 방법은 AMI에 애플리케이션을 포함시키고 이 애플리케이션을 리포지토리에서 인스턴스로 배포하는 앱도 생성하는 것입니다. 이 방법은 부팅 시간을 단축시킬 뿐 아니라 인스턴스 실행 후 애플리케이션을 간편하게 업데이트할 수 있습니다. Chef 레시피는 idempotent 방식이므로 리포지토리에 있는 버전이 인스턴스에 있는 버전과 동일하다면 배포 레시피가 애플리케이션을 수정하지 않는다는 점에 유의하세요.

AWS OpsWorks Stacks용 사용자 지정 AMI 생성

AWS OpsWorks Stacks에서 사용자 지정 AMI를 사용하려면 먼저 사용자 지정 인스턴스에서 AMI를 생성해야 합니다. 다음 두 가지 옵션 중에서 선택할 수 있습니다.

- Amazon EC2 콘솔 또는 API를 사용하여 AWS OpsWorks Stacks 지원 AMI 중 하나의 64비트 버전을 기반으로 인스턴스를 생성하고 사용자 지정합니다.
- Linux AMI의 경우, OpsWorks를 사용하여 연결된 계층의 구성을 기반으로 Amazon EC2 인스턴스를 생성합니다.

사용자 지정 Linux AMI를 생성하기 전에 `/tmp` 파티션`noexec`에서를 비활성화하여 AWS OpsWorks Stacks가 사용자 지정 Linux 인스턴스에 에이전트를 설치할 수 있도록 합니다.

> **Note**
>
> AMI와 호환되지 않는 인스턴스 유형이 있을 수 있으므로 사용하려는 인스턴스 유형과 시작 AMI가 호환되는지 확인해야 합니다. 특히 R3 인스턴스 유형은 하드웨어 보조 가상화(HVM) AMI가 필요합니다.

그런 다음 Amazon EC2 콘솔 또는 API를 사용하여 사용자 지정 인스턴스에서 사용자 지정 AMI를 생성합니다. 같은 리전에 있는 스택이라면 인스턴스를 계층에 추가하고 사용자 지정 AMI를 지정하여 사용자 지정 AMI를 사용할 수 있습니다. 사용자 지정 AMI를 사용하는 인스턴스를 생성하는 방법에 대한 자세한 정보는 계층에 인스턴스 추가를 참조하세요.

> **Note**
>
> 기본적으로 AWS OpsWorks Stacks는 부팅 시 모든 Amazon Linux 업데이트를 설치하여 최신 릴리스를 제공합니다. 또한 Amazon Linux는 약 6개월마다 상당한 변경이 포함된 새 버전을 릴리스합니다. 기본적으로 Amazon Linux를 기반으로 한 사용자 지정 AMI는 새 버전이 릴리스될 경우 자동으로 업데이트됩니다. 권장되는 방법은 사용자 지정 AMI를 특정 Amazon Linux 버전에 고정시키는 것입니다. 그러면 새 버전 테스트를 마칠 때까지 업데이트를 연기할 수 있습니다. 자세한 내용은 내 AMI를 특정 버전으로 고정시키려면 어떻게 해야 합니까?를 참조하세요.

주제

- Amazon EC2를 사용하여 사용자 지정 AMI를 생성합니다.
- AWS OpsWorks Stacks 인스턴스에서 사용자 지정 Linux AMI 생성
- 사용자 지정 Windows AMI 생성

Amazon EC2를 사용하여 사용자 지정 AMI를 생성합니다.

사용자 지정 AMI를 생성하는 가장 간단한 방법이자 Windows AMI의 유일한 옵션은 Amazon EC2 콘솔 또는 API를 사용하여 전체 작업을 수행하는 것입니다. 다음 단계에 대한 세부 정보는 자체 AMI 생성을 참조하세요.

Amazon EC2 콘솔 또는 API를 사용해 사용자 지정 AMI를 생성하려면

1. AWS OpsWorks Stacks 지원 AMI 64비트 버전을 사용하여 인스턴스를 생성합니다.

2. 1단계에서 생성한 인스턴스를 구성하고 패키지를 설치하는 등 사용자 지정합니다. 설치하는 모든 항목이 AMI를 기반으로 모든 인스턴스에서 재현되므로, 특정 인스턴스에만 설치해야 하는 항목을 지금 설치해서는 안 됩니다.
3. 인스턴스를 중지하고 사용자 지정 AMI를 생성합니다.

AWS OpsWorks Stacks 인스턴스에서 사용자 지정 Linux AMI 생성

사용자 지정 AWS OpsWorks Stacks Linux 인스턴스를 사용하여 AMI를 생성하려면 OpsWorks에서 생성한 모든 Amazon EC2 인스턴스에 고유한 자격 증명이 포함되어 있다는 점에 유의하세요. 이러한 인스턴스에서 생성되는 사용자 지정 AMI에는 해당 자격 증명이 포함되며, 이 AMI에 기반하는 모든 인스턴스는 동일한 자격 증명을 갖습니다. 사용자 지정 AMI에 기반하는 인스턴스가 고유 자격 증명을 가지도록 하려면 AMI 생성 전에 사용자 지정 인스턴스에서 자격 증명을 제거해야 합니다.

AWS OpsWorks Stacks 인스턴스에서 사용자 지정 AMI를 생성하려면

1. Linux 스택을 생성하고 계층을 하나 이상 추가하여 사용자 지정된 인스턴스의 구성을 정의합니다. 적절히 사용자 지정된 내장 계층은 물론 완전히 사용자 지정된 계층을 사용할 수 있습니다. 자세한 내용은 AWS OpsWorks 스택 사용자 지정 단원을 참조하십시오.
2. 계층을 편집하고 자동 복구를 비활성화합니다.
3. 계층에 원하는 Linux 배포판이 설치된 인스턴스를 추가하고 시작합니다. Amazon EBS 지원 인스턴스를 사용하는 것이 좋습니다. 인스턴스의 세부 정보 페이지를 열고 나중을 위해 인스턴스의 Amazon EC2 ID를 기록해 둡니다.
4. 인스턴스가 온라인 상태이면 SSH를 사용하여 로그인하고 인스턴스 운영 체제에 따라 다음 네 단계 중 하나를 수행합니다.
5. Chef 11 또는 Chef 12 스택의 Amazon Linux 인스턴스나 Chef 11 스택의 Red Hat 엔터프라이즈 Linux 7 인스턴스를 보려면 다음과 같이 하세요.

    a. `sudo /etc/init.d/monit stop`

    b. `sudo /etc/init.d/opsworks-agent stop`

    c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef`

    > **Note**
    >
    > Chef 12 스택의 인스턴스에서는 이 명령에 다음 두 폴더를 추가합니다.

- /var/chef
- /opt/chef

d. `sudo rpm -e opsworks-agent-ruby`

e. `sudo rpm -e chef`

6. Chef 12 스택의 Ubuntu 16.04 LTS 또는 18.04 LTS 인스턴스를 보려면 다음과 같이 하세요.

a. `sudo systemctl stop opsworks-agent`

b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

c. `sudo apt-get -y remove chef`

d. `sudo dpkg -r opsworks-agent-ruby`

e. `systemctl stop apt-daily.timer`

f. `systemctl stop apt-daily-upgrade.timer`

g. `rm /var/lib/systemd/timers/stamp-apt-daily.timer`

h. `rm /var/lib/systemd/timers/stamp-apt-daily-upgrade.timer`

7. Chef 12 스택에서 지원되는 기타 Ubuntu 버전을 보려면 다음과 같이 하세요.

a. `sudo /etc/init.d/monit stop`

b. `sudo /etc/init.d/opsworks-agent stop`

c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

d. `sudo apt-get -y remove chef`

e. `sudo dpkg -r opsworks-agent-ruby`

8. Chef 12 스택의 Red Hat 엔터프라이즈 Linux 7 인스턴스를 보려면 다음과 같이 하세요.

a. `sudo systemctl stop opsworks-agent`

b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-`

```
agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/
cloud/ /etc/chef /var/chef
```

c. `sudo rpm -e opsworks-agent-ruby`

d. `sudo rpm -e chef`

9. 이 단계는 인스턴스 유형에 따라 다릅니다.

   - Amazon EBS 지원 인스턴스의 경우 Amazon EBS 지원 Linux AMI 생성에 설명된 대로 AWS OpsWorks Stacks 콘솔을 사용하여 인스턴스를 중지하고 AMI를 생성합니다. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html
   - 인스턴스 스토어 지원 인스턴스의 경우 인스턴스 스토어 지원 Linux AMI 생성에 설명된 대로 AMI를 생성한 다음 AWS OpsWorks Stacks 콘솔을 사용하여 인스턴스를 중지합니다.

     AMI를 생성할 때 인증서 파일을 포함시켜야 합니다. 예를 들어, `-i` 인수를 `-i $(find /etc /usr /opt -name '*.pem' -o -name '*.crt' -o -name '*.gpg' | tr '\n' ',')`(으)로 설정한 상태에서 `ec2-bundle-vol` 명령을 호출할 수 있습니다. 번들 중 apt 퍼블릭 키를 제거하지 마십시오. 이 작업은 기본 `ec2-bundle-vol` 명령이 처리합니다.

10. 스택 AWS OpsWorks 콘솔로 돌아가 스택에서 인스턴스를 삭제하여 스택을 정리합니다.

사용자 지정 Windows AMI 생성

다음 절차는 Windows Server 2022 Base 사용자 지정 AMI를 생성합니다. Amazon EC2 관리 콘솔에서 다른 Windows 서버 운영 체제를 선택할 수 있습니다.

> **Important**
>
> 현재 AWS OpsWorks Stacks 에이전트는 영어 - 미국(en-US) 이외의 시스템 UI 언어를 사용하는 Windows 기반 인스턴스에 설치할 수 없으며 AWS OpsWorks Stacks는 관리할 수 없습니다.

주제

- Sysprep을 사용하여 사용자 지정 Windows AMI 생성
- Sysprep 없이 사용자 지정 Windows AMI 생성
- 사용자 지정 Windows AMI를 사용하여 새 인스턴스 추가

Sysprep을 사용하여 사용자 지정 Windows AMI 생성

Sysprep을 사용하여 사용자 지정 Windows AMI를 생성하면 일반적으로 인스턴스 시작은 느리지만 프로세스는 더 깔끔합니다. 로 생성된 이미지에서 생성된 인스턴스를 처음 시작하는 Sysprep 데는 Sysprep 활동, 재시작, AWS OpsWorks 스택 프로비저닝 및 설정 및 구성을 포함한 첫 번째 AWS OpsWorks 스택 실행으로 인해 시간이 더 오래 걸립니다. Amazon EC2 콘솔에서 사용자 지정 Windows AMI 생성 단계를 완료하세요.

Sysprep으로 사용자 지정 Windows AMI를 생성하려면

1. Amazon EC2 콘솔에서 인스턴스 시작을 선택합니다.
2. Microsoft Windows Server 2022 Base를 찾은 다음 선택을 선택합니다.
3. 원하는 인스턴스 유형을 선택하고 [인스턴스 세부 정보 구성]을 선택합니다. 머신 이름, 스토리지 및 보안 그룹 설정 등을 비롯하여 AMI에 대한 구성을 변경합니다. 시작을 선택합니다.
4. 인스턴스 부팅 프로세스가 끝나면 암호를 얻은 다음 Windows 원격 데스크탑 연결 창에서 인스턴스에 연결합니다.
5. Windows 시작 화면에서 시작을 선택한 다음 결과에 EC2ConfigServiceSettings 콘솔이 **ec2configservice** 표시될 때까지 입력을 시작합니다. 콘솔을 엽니다.
6. 일반 탭에서 UserData 실행 활성화 확인란이 채워져 있는지 확인합니다(에이 옵션이 필요하지 않지만 Sysprep AWS OpsWorks Stacks가 에이전트를 설치해야 함). 인스턴스의 컴퓨터 이름 설정... 확인란 선택을 취소합니다. 이 옵션을 선택하면 AWS OpsWorks Stacks와 함께 재시작 루프가 발생할 수 있기 때문입니다.
7. 이미지 탭에서 관리자 암호를 무작위으로 설정하여 사용자가 SSH 키를 사용하여 가져올 수 있는 암호를 Amazon EC2에서 자동으로 생성하도록 하거나 지정로 설정하여 고유한 암호를 지정하도록 합니다. Sysprep은(는) 이 설정을 저장합니다. 고유한 암호를 지정한 경우 편리한 위치에 암호를 저장해 둡니다. [기존 항목 유지]는 선택하지 않는 것이 좋습니다.
8. [적용]을 선택한 다음 [Sysprep과 함께 종료]를 선택합니다. 취소를 확인하라는 메시지가 나타나면 [예]를 선택합니다.
9. 인스턴스가 중지되면 Amazon EC2 콘솔의 인스턴스 목록에서 인스턴스를 마우스 오른쪽 버튼으로 클릭하여 이미지를 선택한 다음 이미지 생성을 선택합니다.
10. [이미지 생성] 페이지에서 이미지의 이름과 설명을 입력하고 볼륨 구성을 지정합니다. 모두 마쳤으면 [이미지 생성]을 선택합니다.
11. [이미지] 페이지를 열고 이미지의 상태가 [보류 중] 단계에서 [사용 가능]으로 변경될 때까지 기다립니다. 이제 새 AMI를 사용할 준비가 되었습니다.

Sysprep 없이 사용자 지정 Windows AMI 생성

Amazon EC2 콘솔에서 사용자 지정 Windows AMI 생성 단계를 완료하세요.

Sysprep을 사용하지 않고 사용자 지정 Windows AMI를 생성하려면

1. Amazon EC2 콘솔에서 인스턴스 시작을 선택합니다.
2. Microsoft Windows Server 2022 Base를 찾은 다음 선택을 선택합니다.
3. 원하는 인스턴스 유형을 선택하고 [인스턴스 세부 정보 구성]을 선택합니다. 머신 이름, 스토리지 및 보안 그룹 설정 등을 비롯하여 AMI에 대한 구성을 변경합니다. 시작을 선택합니다.
4. 인스턴스 부팅 프로세스가 끝나면 암호를 얻은 다음 Windows 원격 데스크탑 연결 창에서 인스턴스에 연결합니다.
5. 인스턴스에서 `C:\Program Files\Amazon\Ec2ConfigService\Settings\config.xml`을 열고 다음 두 가지 설정을 변경한 다음 이 파일을 저장한 후 닫습니다.
   - `Ec2SetPassword~Enabled`
   - `Ec2HandleUserData~Enabled`
6. 원격 데스크톱 세션의 연결을 끊은 다음 Amazon EC2 콘솔로 돌아갑니다.
7. 인스턴스 목록에서 인스턴스를 선택합니다.
8. 인스턴스가 중지되면 Amazon EC2 콘솔의 인스턴스 목록에서 인스턴스를 마우스 오른쪽 버튼으로 클릭하여 이미지를 선택한 다음 이미지 생성을 선택합니다.
9. [이미지 생성] 페이지에서 이미지의 이름과 설명을 입력하고 볼륨 구성을 지정합니다. 모두 마쳤으면 [이미지 생성]을 선택합니다.
10. [이미지] 페이지를 열고 이미지의 상태가 [보류 중] 단계에서 [사용 가능]으로 변경될 때까지 기다립니다. 이제 새 AMI를 사용할 준비가 되었습니다.

사용자 지정 Windows AMI를 사용하여 새 인스턴스 추가

이미지가 [available] 상태로 변경된 후 사용자 지정 Windows AMI에 기반하는 새 인스턴스를 생성할 수 있습니다. 운영 체제 목록에서 사용자 지정 Windows AMI 사용을 선택하면 AWS OpsWorks Stacks에 사용자 지정 AMI 목록이 표시됩니다.

사용자 지정 Windows AMI를 기반으로 새 인스턴스를 추가하려면

1. 새 AMI를 사용할 수 있게 AWS OpsWorks 되면 스택 콘솔로 이동하여 Windows 스택의 인스턴스 페이지를 열고 페이지 하단 근처의 + 인스턴스를 선택하여 새 인스턴스를 추가합니다.

2. [새로 만들기] 탭에서 [고급]를 선택합니다.
3. [운영 체제] 드롭다운 목록에서 [사용자 지정 Windows AMI 사용]을 선택합니다.
4. 앞서 만든 AMI를 [사용자 지정 AMI] 드롭다운 목록에서 선택한 다음 [인스턴스 추가]를 선택합니다.

이제 인스턴스를 시작하고 실행할 수 있습니다.

수동으로 24/7 인스턴스 시작, 중지 및 재부팅

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> Linux 및 Windows 스택 모두에서 24/7 인스턴스를 사용할 수 있습니다.

24/7 인스턴스를 계층에 추가할 경우 해당하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 수동으로 시작하고 수동으로 중지하여 Amazon EC2 인스턴스를 종료해야 합니다. 제대로 작동하지 않는 인스턴스를 수동으로 재부팅할 수도 있습니다. AWS OpsWorks 스택은 시간 기반 및 로드 기반 인스턴스를 자동으로 시작하고 중지합니다. 자세한 내용은 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리 단원을 참조하십시오.

> **Important**
>
> AWS OpsWorks Stacks 인스턴스는 콘솔에서만 AWS OpsWorks 시작, 중지 및 재시작해야 합니다. AWS OpsWorks 는 Amazon EC2 콘솔에서 수행되는 시작, 중지 또는 재시작 작업을 인식하지 못합니다.

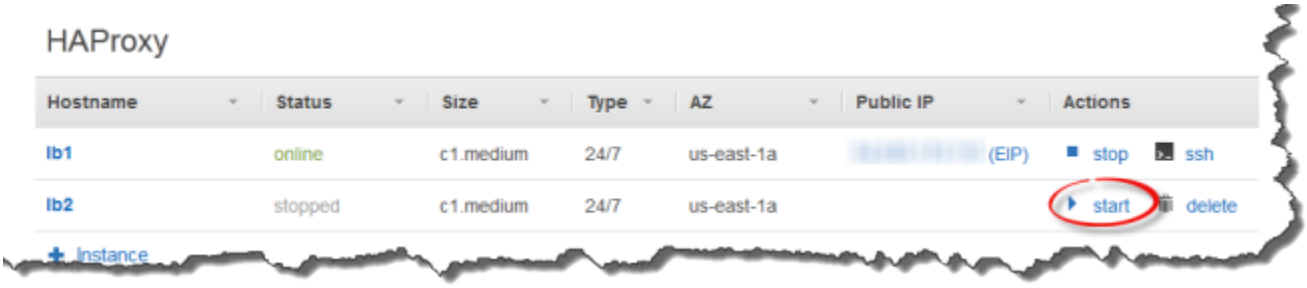
주제

- 인스턴스 시작 또는 재시작

- 인스턴스 중지
- 인스턴스 재부팅

인스턴스 시작 또는 재시작

새 인스턴스를 시작하려면 인스턴스 페이지의 인스턴스 작업 열에서 시작을 클릭합니다.



[모든 인스턴스 시작]을 클릭하면 여러 인스턴스를 생성하여 동시에 시작할 수도 있습니다.

인스턴스를 시작한 후 AWS OpsWorks Stacks는 Amazon EC2 인스턴스를 시작하고 운영 체제를 부팅합니다. 시작 프로세스는 몇 분 정도 소요되며, 일반적으로 Windows 인스턴스가 Linux 인스턴스보다 약간 느립니다. 시작 프로세스가 진행되는 동안 인스턴스의 [상태] 필드에 다음 값이 차례로 표시됩니다.

1. 요청됨 - AWS OpsWorks Stacks가 Amazon EC2 서비스를 호출하여 Amazon EC2 인스턴스를 생성했습니다.
2. 보류 중 - AWS OpsWorks Stacks가 Amazon EC2 인스턴스가 시작될 때까지 대기 중입니다.
3. booting - Amazon EC2 인스턴스가 부팅 중입니다.
4. running_setup - AWS OpsWorks Stacks가 설정 이벤트를 트리거하고 계층의 Setup 레시피를 실행한 다음 Deploy 레시피를 실행합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오. 스택에 사용자 지정 쿡북을 추가한 경우 AWS OpsWorks 스택은 Setup 및 Deploy 레시피를 실행하기 전에 리포지토리에서 현재 버전을 설치합니다.
5. online - 인스턴스를 사용할 준비가 되었습니다.

[상태]가 [온라인]으로 바뀌면 인스턴스가 완전히 작동하는 것입니다.

- 계층에 연결된 로드 밸런서가 있는 경우 AWS OpsWorks Stacks는 인스턴스를 추가합니다.
- AWS OpsWorks Stacks는 각 인스턴스의 Configure 레시피를 실행하는 Configure 이벤트를 트리거합니다.

필요에 따라 이들 레시피가 새 인스턴스를 수용하기 위해 인스턴스를 업데이트합니다.

- AWS OpsWorks Stacks는 인스턴스의 시작 작업을 중지로 대체하며, 이를 사용하여 인스턴스를 중지할 수 있습니다.

인스턴스가 성공적으로 시작하지 못했거나 설정 레시피가 실패한 경우 상태가 각각 [start_failed] 또는 [setup_failed]로 설정됩니다. 로그를 확인하여 원인을 판단할 수 있습니다. 자세한 내용은 디버깅 및 문제 해결 안내서 단원을 참조하십시오.

중지된 인스턴스는 스택의 일부로 유지되며 모든 리소스를 보존합니다. 예를 들어 Amazon EBS 볼륨 및 탄력적 IP 주소가 중지된 인스턴스에 계속 연결됩니다. 인스턴스의 작업 열에서 시작을 선택하여 중지된 인스턴스를 다시 시작할 수 있습니다. 중지된 인스턴스를 재시작하면 다음 작업이 수행됩니다.

- 인스턴스 스토어 지원 인스턴스 - AWS OpsWorks Stacks는 동일한 구성으로 새 Amazon EC2 인스턴스를 시작합니다.
- Amazon EBS 지원 인스턴스 - AWS OpsWorks Stacks는 루트 볼륨을 다시 연결하는 Amazon EC2 인스턴스를 다시 시작합니다.

인스턴스 부팅이 완료되면 AWS OpsWorks Stacks는 초기 시작과 마찬가지로 운영 체제 업데이트를 설치하고 Setup 및 Deploy 레시피를 실행합니다. AWS OpsWorks Stacks는 다시 시작된 인스턴스에 대해 적절하게 다음을 수행합니다.

- 탄력적 IP 주소를 재연결합니다.
- Amazon Elastic Block Store(Amazon EBS) 볼륨을 다시 연결합니다.
- 인스턴스 스토어 지원 인스턴스의 경우, 최신 쿡북 버전을 설치합니다.

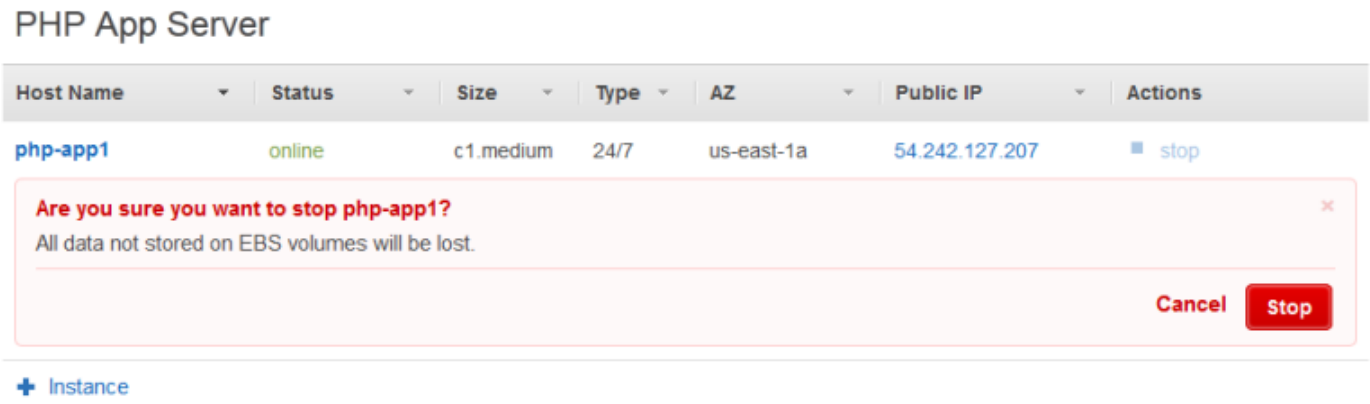
  Amazon EBS 지원 인스턴스는 루트 볼륨에 저장된 사용자 지정 쿡북을 계속 사용합니다. 인스턴스가 중지한 이후 사용자 지정 쿡북이 변경된 경우 인스턴스가 온라인으로 전환된 후 수동으로 사용자 지정 쿡북을 업데이트해야 합니다. 자세한 내용은 사용자 지정 쿡북 업데이트 단원을 참조하십시오.

> **Note**
>
> 탄력적 IP 주소가 재시작된 인스턴스에 다시 연결되려면 시간이 약간 걸릴 수 있습니다. 인스턴스의 [탄력적 IP] 설정은 메타데이터를 표시하며, 단지 해당 주소가 인스턴스와 연결되어야 함을 나타냅니다. [퍼블릭 IP] 설정은 인스턴스의 상태를 반영하며 처음에는 비어 있을 수 있습

니다. 탄력적 IP 주소가 인스턴스에 연결되면 해당 주소가 [퍼블릭 IP] 설정에 할당됩니다(끝에 (EIP)가 추가됨).

인스턴스 중지

인스턴스 페이지에서 인스턴스의 작업 열에서 중지를 클릭합니다. 그러면 AWS OpsWorks Stacks에 종료 레시피를 실행하고 EC2 인스턴스를 종료하도록 알립니다.



[모든 인스턴스 중지]를 클릭하면 스택 내 인스턴스를 모두 종료할 수도 있습니다.

인스턴스를 중지한 후 AWS OpsWorks Stacks는 여러 작업을 수행합니다.

1. 인스턴스의 계층에 연결된 Elastic Load Balancing 로드 밸런서가 있는 경우 AWS OpsWorks Stacks는 인스턴스의 등록을 취소합니다.

   계층이 로드 밸런서의 연결 드레이닝(Connection Draining) 기능을 지원하는 경우 AWS OpsWorks Stacks는 연결 드레이닝(Connection Draining)이 완료될 때까지 기다렸다가 Shutdown 이벤트를 트리거합니다. 자세한 내용은 Elastic Load Balancing 계층 단원을 참조하십시오.
2. AWS OpsWorks Stacks는 인스턴스의 Shutdown 레시피를 실행하는 Shutdown 이벤트를 트리거합니다.
3. Shutdown 이벤트를 트리거한 후 AWS OpsWorks Stacks는 레시피가 완료될 때까지 지정된 Shutdown 시간 동안 기다린 다음 다음을 수행합니다.
   - 인스턴스 스토어 지원 인스턴스를 종료합니다. 그러면 모든 데이터가 삭제됩니다.
   - Amazon EBS 지원 인스턴스를 중지합니다. 루트 볼륨의 데이터는 보존됩니다.

   인스턴스 스토리지에 대한 자세한 내용은 스토리지를 참조하세요.

> **Note**
>
> 기본 종료 제한 시간 설정은 120초입니다. Shutdown 레시피가 더 오랜 시간을 필요로 하는 경우 계층 구성을 편집하여 설정을 변경할 수 있습니다.

인스턴스의 [상태] 열을 관찰하여 종료 프로세스를 모니터링할 수 있습니다. 종료 프로세스가 진행되는 동안 이 필드에 다음 값이 차례로 표시됩니다.

1. 종료 - AWS OpsWorks Stacks가 Amazon EC2 인스턴스를 종료하고 있습니다.
2. shutting_down - AWS OpsWorks Stacks가 계층의 Shutdown 레시피를 실행하고 있습니다.
3. terminated - Amazon EC2 인스턴스가 종료되었습니다.
4. stopped - 인스턴스가 중지했습니다.

인스턴스 재부팅

[인스턴스] 페이지에서 작동하지 않는 인스턴스의 이름을 클릭하여 세부 정보 페이지를 연 다음 [재부팅]을 클릭합니다.

이 명령은 연결된 Amazon EC2 인스턴스의 소프트 재부팅을 수행합니다. 이 프로세스는 인스턴스 스토어 지원 인스턴스에서도 인스턴스 데이터를 삭제하지 않으며 어떤 수명 주기 이벤트도 트리거하지 않습니다.

> **Note**
>
> Stacks가 실패한 인스턴스를 자동으로 교체하도록 AWS OpsWorks 하려면 자동 복구를 활성화합니다. 자세한 내용은 자동 복구 사용 단원을 참조하십시오.

시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

수신 트래픽이 변동함에 따라 스택의 인스턴스 수가 로드를 무리 없이 처리하기에는 너무 적거나 필요 이상으로 많을 수 있습니다. 시간 기반 또는 로드 기반 인스턴스를 사용하여 계층의 인스턴스 수를 자동으로 늘리거나 줄이면 불필요한 용량에 대한 비용을 지불할 필요 없이 수신 트래픽을 적절히 처리하는 데 충분한 인스턴스를 항상 유지하면서도 시간과 비용을 모두 절감할 수 있습니다. 서버 로드를 모니터링하거나 인스턴스를 수동으로 시작 또는 중지할 필요가 없습니다. 또한 시간 기반 및 로드 기반 인스턴스는 동일 리전 내의 여러 가용 영역에서 자동으로 애플리케이션을 분산하고 확장하며 밸런싱하므로 지리적 중복성 및 확장성을 제공합니다.

자동 조정은 다양한 기준에 따라 계층의 온라인 인스턴스를 조정하는 두 인스턴스 유형을 기반으로 합니다.

- 시간 기반 인스턴스

  이 인스턴스 유형은 특정 시간 또는 특정 요일에만 실행되는 인스턴스를 포함시켜 예측 가능한 패턴을 따르는 부하를 처리할 수 있게 해줍니다. 예를 들어 야간 백업 작업을 수행하기 위해 오후 6시부터 일부 인스턴스를 시작하거나 트래픽이 적은 주말에 일부 인스턴스를 중지할 수 있습니다.
- 로드 기반 인스턴스

  이 인스턴스 유형은 스택이 다양한 로드 측정치를 기반으로 트래픽이 많을 때는 추가 인스턴스를 시작하고 트래픽이 적을 때는 인스턴스를 중지하여 가변적인 로드를 처리할 수 있게 해줍니다. 예를 들어 Stacks가 평균 CPU 사용률이 AWS OpsWorks 80%를 초과할 때 인스턴스를 시작하고 평균 CPU 부하가 60% 미만으로 떨어질 때 인스턴스를 중지하도록 할 수 있습니다.

Linux 스택에서는 시간 기반 인스턴스와 로드 기반 인스턴스가 모두 지원되지만, Windows 스택에서는 시간 기반 인스턴스만 지원됩니다.

수동으로 시작 및 중지해야 하는 24/7 인스턴스와 달리, 시간 기반 또는 로드 기반 인스턴스는 사용자가 직접 시작하거나 중지하지 않습니다. 대신 인스턴스를 구성하면 AWS OpsWorks Stacks가 구성에

따라 인스턴스를 시작하거나 중지합니다. 예를 들어 지정된 일정에 따라 시작 및 중지하도록 시간 기반 인스턴스를 구성합니다. 그러면 AWS OpsWorks 스택이 해당 구성에 따라 인스턴스를 시작하고 중지합니다.

일반적인 방법은 다음과 같이 세 가지 인스턴스 유형을 모두 함께 사용하는 것입니다.

- 베이스 로드를 처리하기 위한 24/7 인스턴스 집합. 일반적으로 이러한 인스턴스를 시작한 후 계속 작동시킵니다.
- AWS OpsWorks Stacks가 예측 가능한 트래픽 변동을 처리하기 위해 시작하고 중지하는 시간 기반 인스턴스 세트입니다. 예를 들어 근무 시간 중 트래픽이 가장 높을 경우 시간 기반 인스턴스를 아침에 시작하고 저녁에 중지하도록 구성합니다.
- AWS OpsWorks Stacks가 예측할 수 없는 트래픽 변형을 처리하기 위해 시작하고 중지하는 로드 기반 인스턴스 세트입니다. AWS OpsWorks 스택은 로드가 스택의 24/7 및 시간 기반 인스턴스 용량에 도달하면 시작하고 트래픽이 정상으로 돌아올 때 중지합니다.

이러한 조정 시간을 사용하는 방법에 대한 자세한 정보는 서버 수 최적화 단원을 참조하세요.

> **Note**
>
> 인스턴스 계층용 앱을 생성했거나 사용자 지정 쿡북을 생성한 경우 AWS OpsWorks Stacks는 처음 시작될 때 최신 버전을 시간 기반 및 로드 기반 인스턴스에 자동으로 배포합니다. 그러나 AWS OpsWorks Stacks는 다시 시작된 오프라인 인스턴스에 최신 쿡북을 반드시 배포하지는 않습니다. 자세한 내용은 앱 편집 및 사용자 지정 쿡북 업데이트 단원을 참조하세요.

주제

- 자동 시간 기반 조정 사용
- 자동 로드 기반 조정 사용
- 로드 기반 조정과 자동 복구의 차이점

자동 시간 기반 조정 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

시간 기반 조정을 사용하면 지정된 일정에 따라 인스턴스를 시작하거나 중지하여 특정 시간대 또는 요일에 계층이 온라인 상태여야 하는 인스턴스 수를 제어할 수 있습니다. AWS OpsWorks 스택은 2분마다를 확인하고 필요에 따라 인스턴스를 시작하거나 중지합니다. 다음과 같이 각 인스턴스마다 따로 일정을 지정합니다.

- 시간. 예를 들어 야간보다 주간에 더 많은 인스턴스를 실행할 수 있습니다.
- 요일. 예를 들어 주말보다 주중에 더 많은 인스턴스를 실행할 수 있습니다.

> **Note**
>
> 특정 날짜를 지정할 수는 없습니다.

주제

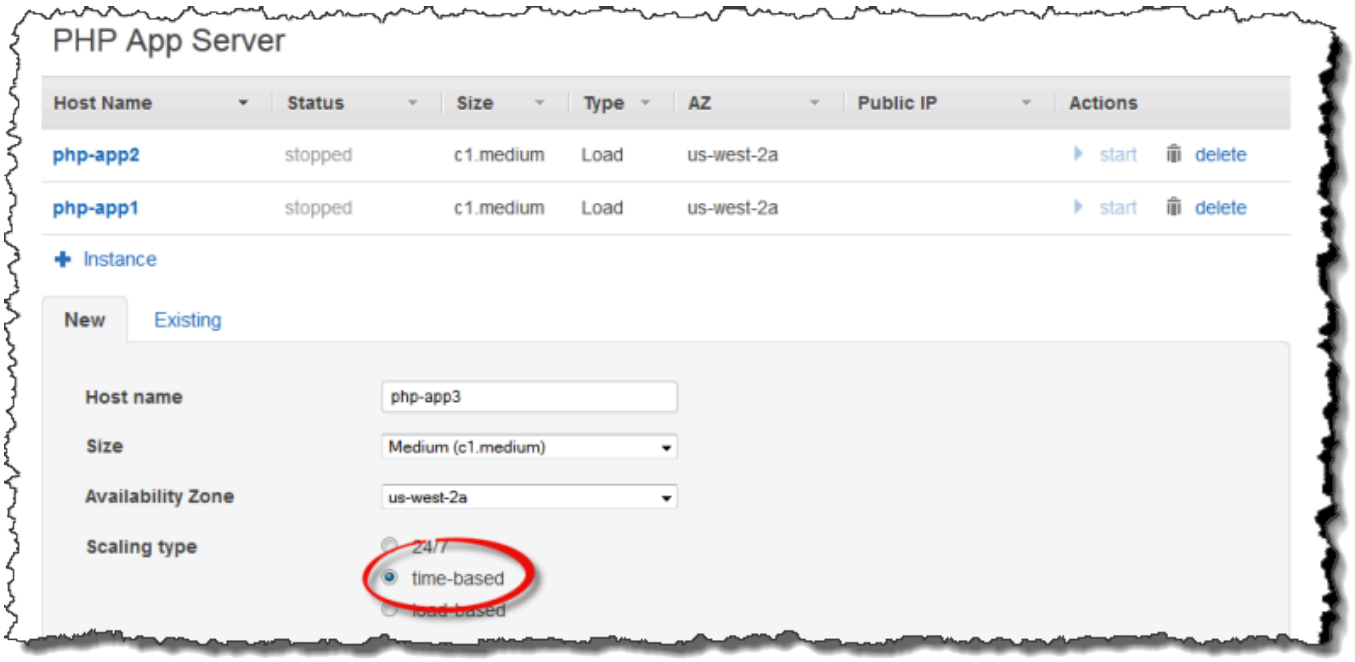
- 계층에 시간 기반 인스턴스 추가
- 시간 기반 인스턴스 구성

계층에 시간 기반 인스턴스 추가

새로운 시간 기반 인스턴스를 계층에 추가할 수도 있고 기존 인스턴스를 사용할 수도 있습니다.

새 시간 기반 인스턴스를 추가하려면

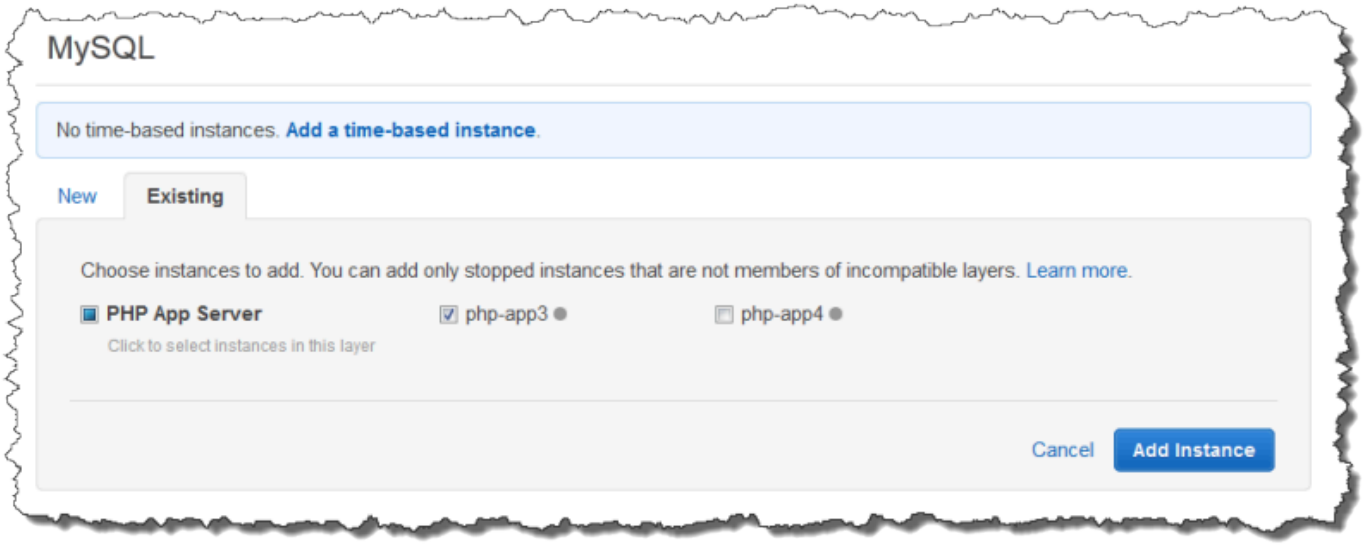
1. 인스턴스 페이지에서 + 인스턴스-를 클릭하여 인스턴스를 추가합니다. 새 탭에서 고급을 선택한 다음 시간 기반을 선택합니다.



2. 인스턴스를 구성합니다. 그런 다음 인스턴스 추가를 클릭하여 계층에 인스턴스를 추가합니다.

계층에 기존의 시간 기반 인스턴스를 추가하려면

1. 계층에 시간 기반 인스턴스가 이미 있는 경우 시간 기반 인스턴스 페이지에서 + 인스턴스를 클릭합니다. 그렇지 않으면 시간 기반 인스턴스 추가를 클릭합니다. 그런 다음 기존 탭을 선택합니다.



2. 기존 탭의 목록에서 인스턴스를 선택합니다. 이 목록에는 시간 기반 인스턴스만 표시됩니다.

> **Note**
>
> 기존 인스턴스를 사용하지 않으려면 이전 절차에서 설명한 것처럼 새 탭을 클릭하여 새 인스턴스를 생성합니다.

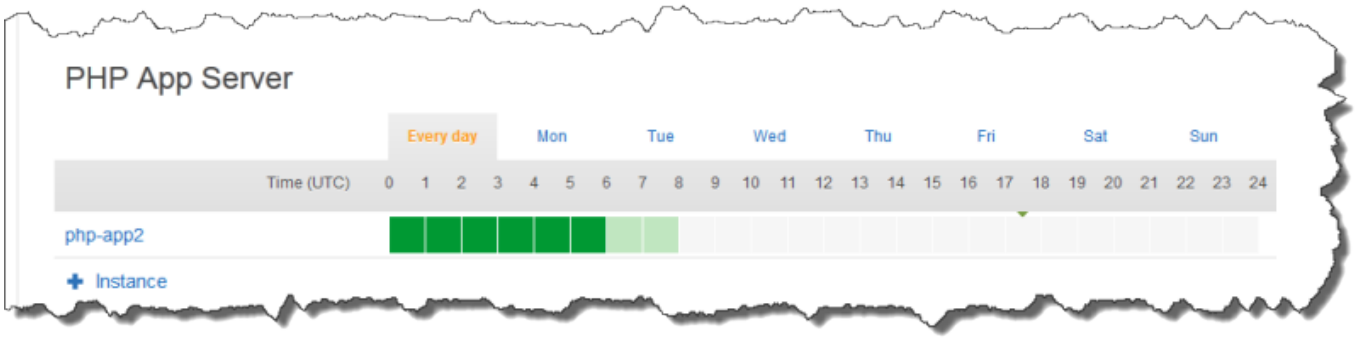
3. 인스턴스 추가를 클릭하여 계층에 인스턴스를 추가합니다.

시간 기반 인스턴스 구성

시간 기반 인스턴스를 계층에 추가한 후 다음과 같이 인스턴스 일정을 구성합니다.

시간 기반 인스턴스를 구성하려면

1. 탐색 창의 인스턴스에서 시간 기반을 선택합니다.
2. 원하는 시간 아래의 해당 상자를 채워 시간 기반 인스턴스별로 온라인 기간을 지정합니다.
   - 매일 동일한 일정을 사용하려면 매일 탭을 클릭한 다음 온라인 기간을 지정합니다.
   - 매일 다른 일정을 사용하려면 각 날짜를 클릭하고 적절한 기간을 선택합니다.



> **Note**
>
> 인스턴스를 시작하는 데 걸리는 시간을 허용하고 AWS OpsWorks Stacks는 몇 분마다 인스턴스를 시작하거나 중지해야 하는지 확인합니다. 예를 들어 인스턴스가 1:00 UTC부터 실행해야 할 경우 0:00 UTC에 시작하세요. 그렇지 않으면 AWS OpsWorks Stacks는 1:00 UTC 이후 몇 분 후에 인스턴스를 시작할 수 있으며 인스턴스가 온라인 상태가 되는 데 몇 분 더 걸립니다.

이전 단계를 수행하여 언제든지 인스턴스의 온라인 기간을 변경할 수 있습니다. 다음에 AWS OpsWorks Stacks가 확인할 때 새 일정을 사용하여 인스턴스를 시작할지 또는 중지할지 결정합니다.

> **Note**
>
> 시간 기반 페이지를 열고, 시간 기반 인스턴스 추가(계층에 아직 시간 기반 인스턴스가 추가되지 않은 경우) 또는 + 인스턴스(계층에 이미 하나 이상의 시간 기반 인스턴스가 있는 경우)를 클릭하여 새 시간 기반 인스턴스를 추가할 수도 있습니다. 그런 다음 이전 절차에 설명된 대로 인스턴스를 구성합니다.

자동 로드 기반 조정 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

로드 기반 인스턴스를 사용하면 수신 트래픽의 변화에 대응하여 인스턴스를 빠르게 시작하거나 중지할 수 있습니다. AWS OpsWorks Stacks는 Amazon CloudWatch 데이터를 사용하여 각 계층에 대해 다음 지표를 계산합니다. 이는 계층의 모든 인스턴스에서 평균 값을 나타냅니다.

- CPU: 평균 CPU 사용률, 예: 80%
- 메모리: 평균 메모리 사용률, 예: 60%
- 로드: 시스템이 1분에 수행할 수 있는 평균 컴퓨팅 작업

사용자는 이러한 측정치 중 일부 또는 모두에 대해 확장 및 축소 임계값을 정의합니다. 사용자 지정 CloudWatch 경보를 임계값으로 사용할 수도 있습니다.

임계값을 넘어서면 조정 이벤트가 트리거됩니다. 사용자는 다음을 지정하여 AWS OpsWorks Stacks가 조정 이벤트에 응답하는 방법을 결정합니다.

- 시작 또는 중지할 인스턴스 수.

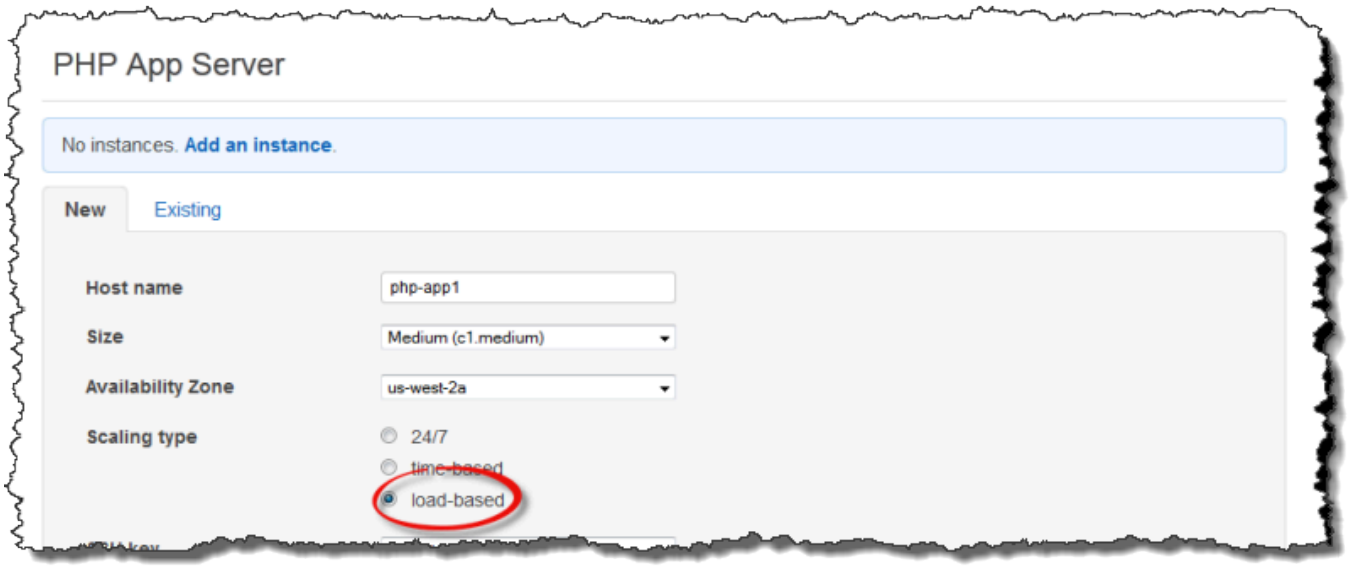
- 인스턴스를 시작하거나 삭제하기 전에 AWS OpsWorks Stacks가 임계값을 초과한 후 대기해야 하는 시간입니다. 예를 들어 CPU 사용률이 최소 15분 이상 임계값을 초과해야 합니다. 이 값은 일시적인 트래픽 변동을 무시할 수 있게 해줍니다.
- 지표를 다시 모니터링하기 전에 인스턴스를 시작하거나 중지한 후 AWS OpsWorks Stacks가 기다려야 하는 시간입니다. 일반적으로 시작된 인스턴스가 온라인 상태로 전환하거나 중지된 인스턴스가 종료될 때까지 충분한 시간을 허용한 이후에 계층이 여전히 임계값을 초과하는지 평가해야 합니다.

조정 이벤트가 발생하면 AWS OpsWorks Stacks는 로드 기반 인스턴스만 시작하거나 중지합니다. 24/7 인스턴스 또는 시간 기반 인스턴스는 시작하거나 중지하지 않습니다.

> **Note**
>
> 자동 로드 기반 조정은 새 인스턴스를 생성하지 않습니다. 이미 생성된 인스턴스만 시작하고 중지합니다. 따라서 최대 예상 로드를 처리할 수 있도록 미리 충분한 로드 기반 인스턴스를 프로비저닝해야 합니다.

로드 기반 인스턴스를 생성하려면

1. 인스턴스 페이지에서 + 인스턴스를 클릭하여 인스턴스를 추가합니다. 고급을 선택한 다음 로드 기반을 선택합니다.



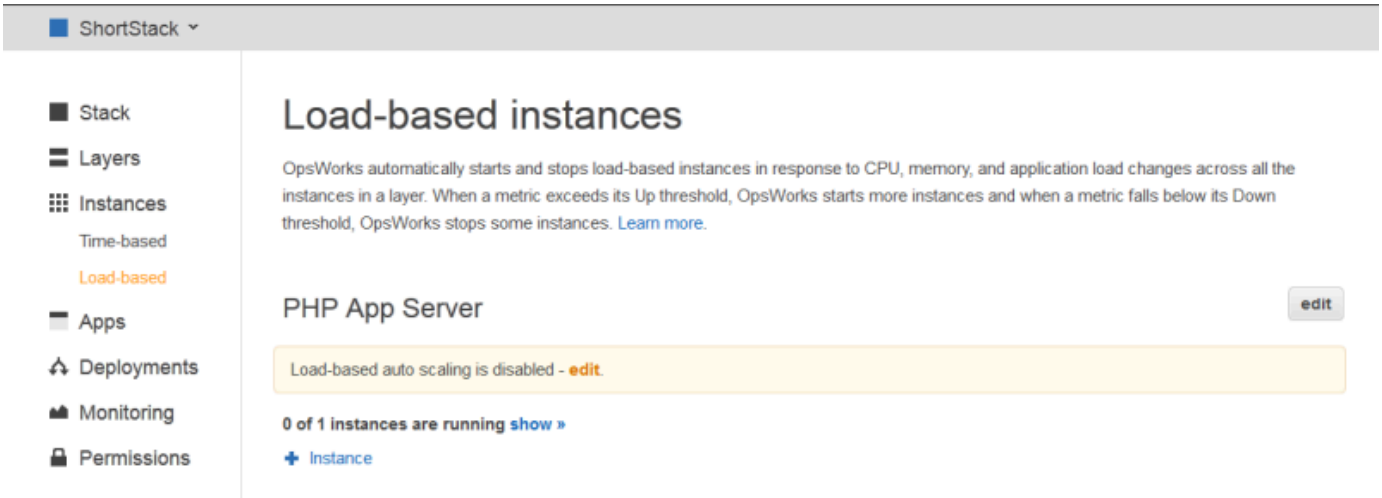
2. 인스턴스를 구성한 다음 인스턴스 추가를 선택하여 계층에 인스턴스를 추가합니다.

충분한 수의 인스턴스가 생성될 때까지 이 절차를 반복합니다. 필요할 경우 나중에 인스턴스를 추가하거나 제거할 수 있습니다.

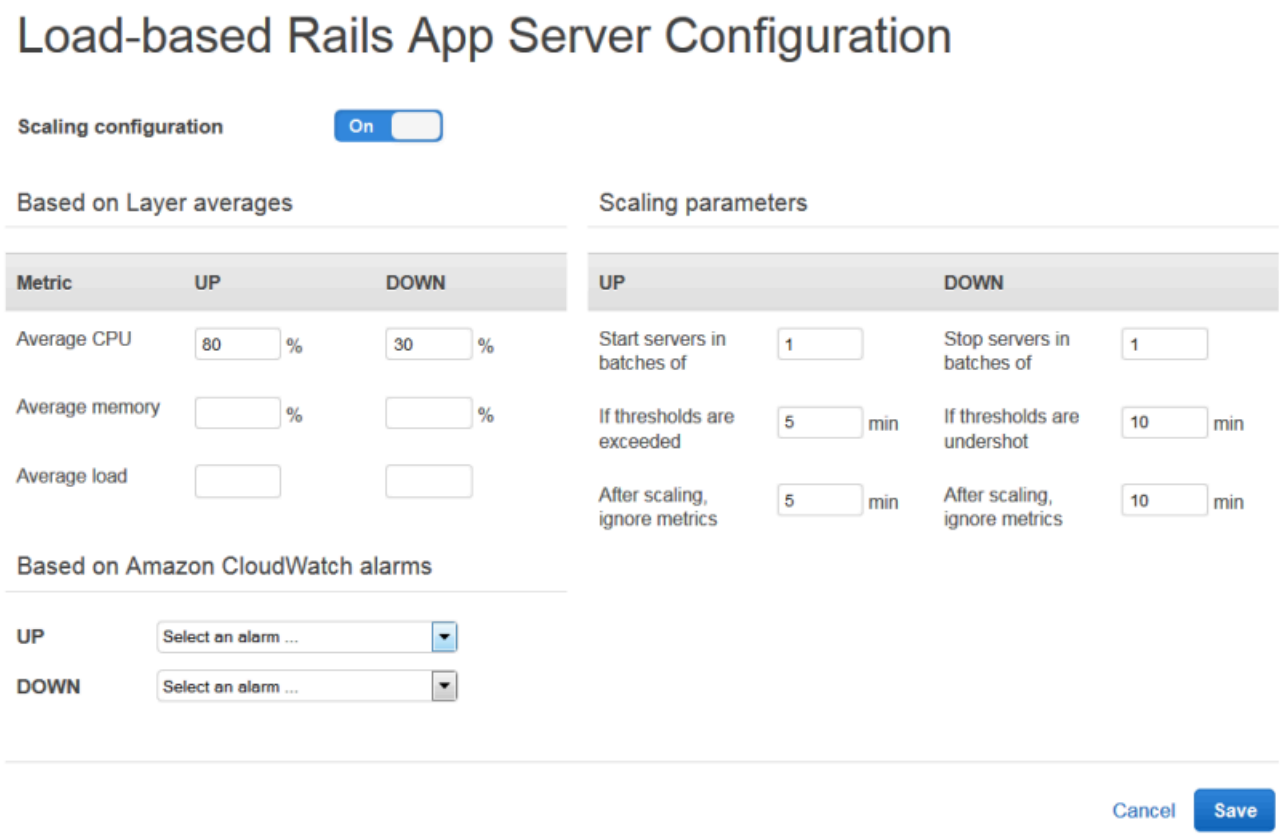
계층에 로드 기반 인스턴스를 추가한 후에는 로드 기반 조정을 활성화하고 구성을 지정해야 합니다. 로드 기반 조정 구성은 인스턴스 속성이 아니라 계층 속성으로, 계층이 로드 기반 인스턴스를 시작 또는 중지할지 여부를 지정합니다. 이 구성은 로드 기반 인스턴스를 사용하는 각 계층에 개별적으로 지정해야 합니다

자동 로드 기반 조정을 활성화하고 구성하려면

1. 탐색 창의 인스턴스에서 로드 기반을 선택한 다음 적절한 계층에 대한 편집을 선택합니다.



2. 로드 기반 Auto Scaling 활성화를 켜기로 설정합니다. 그런 다음 임계값 및 조정 파라미터를 설정하고 인스턴스를 추가 또는 삭제하는 방법과 시점을 정의합니다.



계층 평균 임계값

계층의 모든 인스턴스에 대해 산출한 아래의 평균 값을 기준으로 조정 임계값을 설정할 수 있습니다.

- 평균 CPU – 계층의 평균 CPU 사용률로, 전체 사용률에 대한 백분율로 표시합니다.
- 평균 메모리 – 계층의 평균 메모리 사용률로, 전체 사용률에 대한 백분율로 표시합니다.
- 평균 로드 – 계층의 평균 로드입니다.

  로드를 계산하는 방법에 대한 자세한 내용은 위키피디아의 로드(계산)를 참조하세요.

임계값을 초과하면 조정 이벤트가 발생하고, 인스턴스가 더 필요한 경우 확장되고, 인스턴스가 더 적은 경우 축소됩니다. AWS OpsWorks 그런 다음 Stacks는 조정 파라미터를 기반으로 인스턴스를 추가하거나 삭제합니다.

사용자 지정 CloudWatch 경보

최대 다섯 개의 사용자 지정 CloudWatch 경보를 확장 또는 축소 임계값으로 사용할 수 있습니다. 경보는 스택과 같은 리전에 있어야 합니다. 사용자 지정 경보를 생성하는 방법에 대한 자세한 내용은 Amazon CloudWatch 경보 생성을 참조하세요.

> **Note**
>
> 사용자 지정 경보를 사용하려면 `cloudwatch:DescribeAlarms`를 허용하도록 서비스 역할을 업데이트해야 합니다. 이 기능을 처음 사용할 때 AWS OpsWorks Stacks가 역할을 업데이트하도록 하거나 역할을 수동으로 편집할 수 있습니다. 자세한 내용은 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 단원을 참조하십시오. 로드 기반 구성을 위해 구성된 경보가 여러 개 있는 경우 경보가 `INSUFFICIENT_DATA` 메트릭 경보 상태에 있으면 다른 경보가 `ALARM` 상태에 있더라도 로드 기반 인스턴스 확장이 발생할 수 없습니다. 모든 경보가 `OK` 또는 `ALARM` 상태인 경우에만 Auto Scaling을 진행할 수 있습니다. Amazon CloudWatch 경보에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용을 참조하세요.

조정 파라미터

다음 파라미터는 AWS OpsWorks Stacks가 조정 이벤트를 관리하는 방법을 제어합니다.

- 다음의 배치에서 서버 시작 – 조정 이벤트가 발생할 때 추가하거나 제거할 인스턴스의 수입니다.
- 임계값을 초과하는 경우 - AWS OpsWorks Stacks가 조정 이벤트를 트리거하기 전에 로드가 업스케일링 임계값을 초과하거나 다운스케일링 임계값 미만으로 유지되어야 하는 시간(분)입니다.
- 조정 후 지표 무시 - AWS OpsWorks Stacks가 지표를 무시하고 추가 조정 이벤트를 억제해야 하는 조정 이벤트 발생 후 시간(분)입니다.

  예를 들어 AWS OpsWorks Stacks는 확장 이벤트 후 새 인스턴스를 추가하지만 인스턴스는 부팅 및 구성될 때까지 부하를 줄이기 시작하지 않습니다. 새 인스턴스가 온라인 상태가 되어 요청을 처리하게 될 때까지의 몇 분 동안은 조정 이벤트가 추가로 발생해도 소용이 없습니다. 이 설정을 통해 AWS OpsWorks Stacks에서 새 인스턴스가 온라인 상태가 될 때까지 조정 이벤트를 충분히 억제하도록 할 수 있습니다.

  또한 이 설정을 늘리면 평균 CPU, 평균 메모리 또는 평균 로드 등 계층 평균에서 일시적으로 불일치가 발생하는 경우 급격한 조정 변동을 방지할 수 있습니다.

  예를 들어 CPU 사용량이 한도를 초과했는데 메모리 사용량이 축소 상태에 가까워지면 인스턴스 확대 이벤트 직후에 메모리 축소 이벤트가 발생할 수 있습니다. 이를 방지하기 위해 조

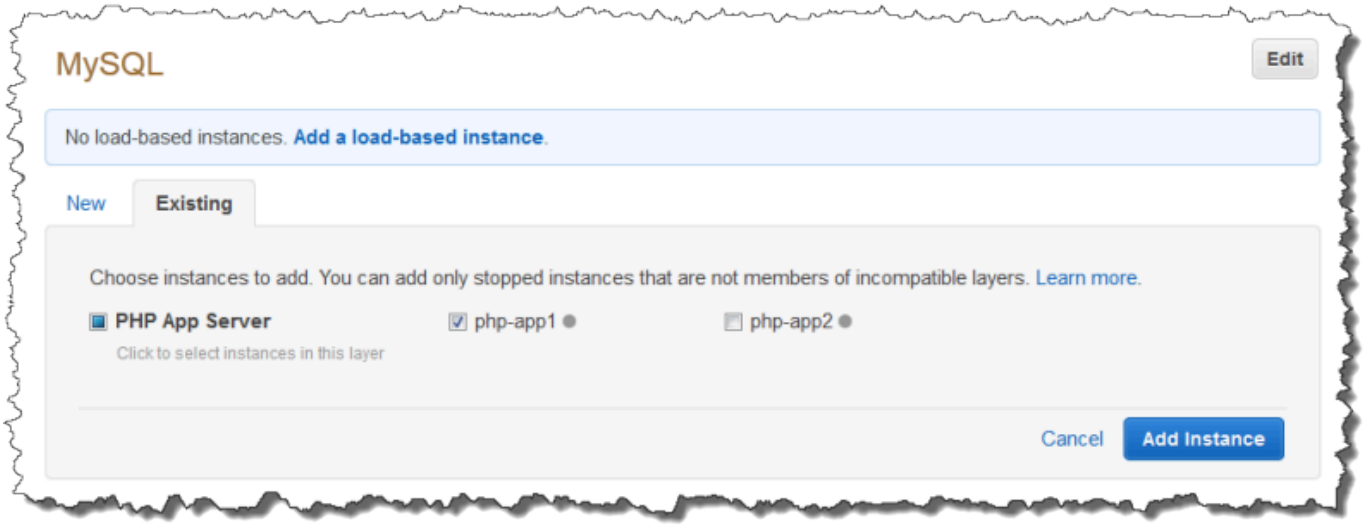
정 후 지표 무시 설정에서 시간(분)을 늘릴 수 있습니다. 그러면 언급한 예에서 CPU 조정은 발생하지만 메모리 축소 이벤트는 발생하지 않습니다.

3. 로드 기반 인스턴스를 더 추가하려면 + 인스턴스를 클릭하고 설정을 구성한 다음 인스턴스 추가를 클릭합니다. 예상되는 최대 로드를 처리하기에 충분한 로드 기반 인스턴스가 확보될 때까지 반복합니다. 그런 다음 저장을 선택합니다.

> **Note**
>
> 로드 기반 페이지를 열어서 로드 기반 인스턴스 추가(계층에 아직 로드 기반 인스턴스가 추가되지 않은 경우) 또는 + 인스턴스(계층에 이미 하나 이상의 로드 기반 인스턴스가 있는 경우)를 클릭하여 새 로드 기반 인스턴스를 추가할 수도 있습니다. 그런 다음 이 섹션의 이전 절차에 설명된 대로 인스턴스를 구성합니다.

계층에 기존의 로드 기반 인스턴스를 추가하려면

1. 탐색 창의 인스턴스에서 로드 기반을 선택합니다.
2. 계층에 대해 로드 기반의 자동 조정이 이미 활성화된 경우 + 인스턴스를 선택합니다. 그렇지 않으면 로드 기반 인스턴스 추가를 선택합니다. 기존 탭을 선택합니다.



3. 기존 탭에서 인스턴스를 선택합니다. 목록에는 로드 기반 인스턴스만 표시됩니다.

> **Note**
>
> 기존 인스턴스를 사용하지 않으려면 이전 절차에서 설명한 것처럼 새 탭을 클릭하여 새 인스턴스를 생성합니다.

4. 인스턴스 추가를 클릭하여 계층에 인스턴스를 추가합니다.

언제라도 자동 로드 기반 조정 구성을 수정하거나 비활성화할 수 있습니다.

자동 로드 기반 조정을 비활성화하려면

1. 탐색 창의 인스턴스에서 로드 기반을 선택한 다음 적절한 계층에 대한 편집을 선택합니다.
2. 로드 기반 Auto Scaling 활성화를 아니오로 전환합니다.

로드 기반 조정과 자동 복구의 차이점

자동 로드 기반 조정은 모든 실행 중 인스턴스에서 평균된 로드 측정치를 사용합니다. 지표가 지정된 임계값 사이에 남아 있는 경우 AWS OpsWorks Stacks는 인스턴스를 시작하거나 중지하지 않습니다. 반면 자동 복구를 사용하면 인스턴스가 응답을 중지하면 AWS OpsWorks Stacks는 동일한 구성으로 새 인스턴스를 자동으로 시작합니다. 인스턴스는 네트워크 문제 또는 인스턴스 문제 때문에 응답하지 못할 수 있습니다.

예를 들어 CPU 확장 임계값이 80%이고 한 인스턴스가 응답을 멈췄다고 가정해 봅시다.

- 자동 복구가 비활성화되어 있고 실행 중인 나머지 인스턴스가 평균 CPU 사용률을 80% 미만으로 유지할 수 있는 경우 AWS OpsWorks Stacks는 새 인스턴스를 시작하지 않습니다. 나머지 인스턴스에서 평균 CPU 사용률이 80%를 초과할 경우에만 대체 인스턴스를 시작합니다.
- 자동 복구가 활성화된 경우 AWS OpsWorks Stacks는 로드 임계값에 관계없이 대체 인스턴스를 시작합니다.

AWS OpsWorks Stacks 외부에서 생성된 컴퓨팅 리소스 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

인스턴스은(는) AWS OpsWorks Stacks를 사용하여 (Amazon Elastic Compute Cloud(Amazon EC2)) 인스턴스 그룹을 생성하고 관리하는 방법을 설명합니다. Linux 컴퓨팅 리소스를 Stacks 외부에서 생성된 AWS OpsWorks 스택에 통합할 수도 있습니다.

- Amazon EC2 콘솔, CLI 또는 API를 사용하여 직접 생성한 Amazon EC2 인스턴스.
- 가상 머신에서 실행되는 인스턴스를 비롯하여 자체 하드웨어에서 실행되는 온프레미스 인스턴스.

이러한 컴퓨팅 리소스는 AWS OpsWorks Stacks 관리형 인스턴스가 되며 일반 AWS OpsWorks Stacks 인스턴스와 마찬가지로 관리할 수 있습니다.

- 사용자 권한 관리 - AWS OpsWorks Stacks 사용자 관리를 사용하여 어떤 사용자에게 스택 액세스를 허용할지, 스택 인스턴스에서 어떤 작업을 수행하도록 허용할지, 이들에게 SSH 액세스 및 sudo 권한을 부여할지 여부를 지정할 수 있습니다.
- 작업 자동화 - AWS OpsWorks Stacks가 사용자 지정 Chef 레시피를 실행하여 단일 명령으로 스택 인스턴스의 일부 또는 전부에서 스크립트 실행과 같은 작업을 수행하도록 할 수 있습니다.

  인스턴스를 계층에 할당하면 AWS OpsWorks Stacks는 사용자 지정 레시피를 포함하여 수명 주기의 주요 지점에서 인스턴스에 대해 지정된 Chef 레시피 세트를 자동으로 실행합니다. 등록된 Amazon EC2 인스턴스만 사용자 지정 계층에 할당할 수 있습니다.
- 리소스 관리 - 스택을 사용하면에서 리소스를 그룹화하고 관리할 수 있으며 AWS 리전, OpsWorks 대시보드에는 모든 리전에서 스택의 상태가 표시됩니다.
- 패키지 설치 – Chef 레시피를 사용하여 스택의 어느 인스턴스에나 패키지를 설치할 수 있습니다.
- 운영 체제 업데이트 - AWS OpsWorks 스택은 스택의 인스턴스에 운영 체제 보안 패치 및 업데이트를 설치하는 간단한 방법을 제공합니다.
- 애플리케이션 배포 - AWS OpsWorks Stacks는 모든 스택의 애플리케이션 서버 인스턴스에 애플리케이션을 일관되게 배포합니다.

- 모니터링 - AWS OpsWorks 스택은 사용자 지정 CloudWatch 지표를 생성하여 스택의 모든 인스턴스를 모니터링합니다.

요금에 대한 자세한 내용은 AWS OpsWorks 요금을 참조하세요.

다음은 등록된 인스턴스 작업의 기본 절차입니다.

1. 스택에 인스턴스를 등록합니다.

   인스턴스는 이제 스택의 일부이며 AWS OpsWorks Stacks에서 관리합니다.
2. 필요한 경우, 인스턴스를 계층에 할당합니다.

   이 단계를 통해 AWS OpsWorks Stacks 관리 기능을 최대한 활용할 수 있습니다. 등록된 온프레미스 인스턴스는 어떤 계층에나 할당할 수 있지만 등록된 Amazon EC2 인스턴스는 사용자 지정 계층에만 할당할 수 있습니다.
3. AWS OpsWorks Stacks를 사용하여 인스턴스를 관리합니다.
4. 스택에 인스턴스가 더 이상 필요하지 않으면 인스턴스 등록을 취소하면 AWS OpsWorks Stacks에서 인스턴스가 제거됩니다.

다음 섹션에서는 이 프로세스를 상세히 설명합니다.

주제

- Stacks AWS OpsWorks Stack에 인스턴스 등록
- 등록된 인스턴스 관리
- 등록된 인스턴스를 계층에 할당
- 등록된 인스턴스 할당 해제
- 등록된 인스턴스 등록 해제
- 등록된 인스턴스 수명 주기

Stacks AWS OpsWorks Stack에 인스턴스 등록

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

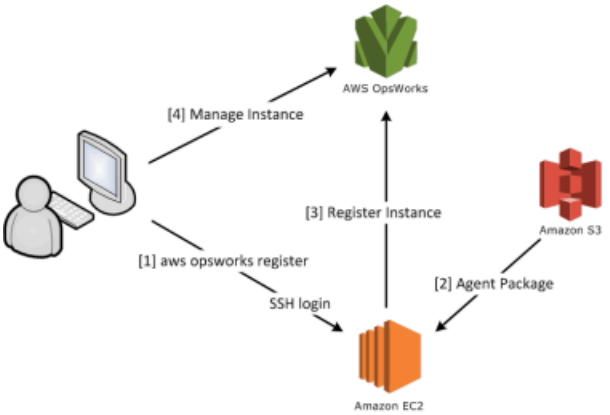
AWS OpsWorks Stacks 외부에 AWS CLI aws opsworks register 있는 인스턴스를 등록하려면 명령을 실행합니다. 등록할 인스턴스에서 또는 다른 컴퓨터에서 이 명령을 실행할 수 있습니다. `AWSOpsWorksRegisterCLI_EC2` 또는 `AWSOpsWorksRegisterCLI_OnPremises` 정책을 사용자 또는 그룹에 적용하여가 EC2 또는 온프레미스 인스턴스를 등록 AWS CLI 하는 데 각각 필요한 권한을 부여합니다. 이러한 정책에는 버전 1.16.180 AWS CLI 이상의 이상이 필요합니다.

> **Note**
>
> 사용자나 역할이 인스턴스를 등록하지 못하게 하려면 register 명령에 대한 액세스를 거부하도록 인스턴스 프로파일을 업데이트하세요.

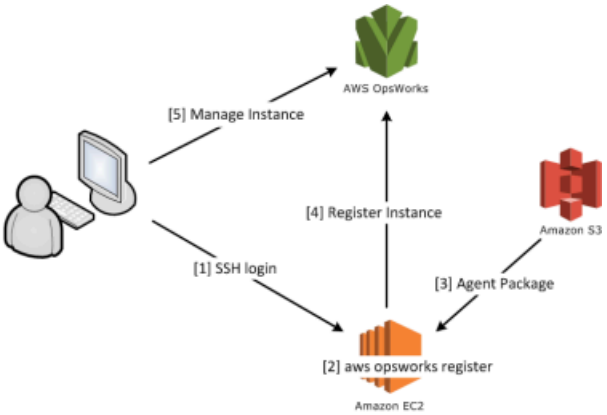
등록 프로세스는 AWS OpsWorks 스택을 사용하여 관리하려는 인스턴스에 에이전트를 설치하고 지정한 AWS OpsWorks 스택에 인스턴스를 등록합니다. 인스턴스를 등록한 후 인스턴스는 스택의 일부이며 AWS OpsWorks Stacks에서 관리됩니다. 자세한 내용은 등록된 인스턴스 관리 단원을 참조하십시오.

> **Note**
>
> PowerShell용 AWS 도구에는 `register` API 작업을 호출하는 `Register-OpsInstance` cmdlet이 포함되어 있지만 대신 AWS CLI 를 사용하여 `register` 명령을 실행하는 것이 좋습니다.

다음 다이어그램은 Amazon EC2 인스턴스를 등록하는 두 가지 방법을 모두 보여 줍니다. 온프레미스 인스턴스 등록에도 같은 방법을 사용할 수 있습니다.



Registering an instance from a workstation



Registering an instance from the instance itself

> **Note**
>
> AWS OpsWorks Stacks 콘솔을 사용하여 등록된 인스턴스를 관리할 수 있지만, 인스턴스를 등록하려면 AWS CLI `register` 명령을 사용해야 합니다. 이렇게 해야 하는 이유는 등록 프로세스를 인스턴스에서 실행해야 하며 콘솔을 통해 수행할 수 없기 때문입니다.

다음 섹션에서는 이 절차를 상세히 설명합니다.

주제

- 안내: 워크스테이션에서 인스턴스 등록
- Amazon EC2 및 온프레미스 인스턴스 등록

안내: 워크스테이션에서 인스턴스 등록

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

등록 프로세스는 여러 시나리오를 지원합니다. 이 섹션에서는 한 시나리오의 전체적인 예제를 통해 워크스테이션을 사용하여 Amazon EC2 인스턴스를 등록하는 방법을 안내합니다. 다른 등록 시나리오도 비슷한 절차를 사용합니다. 자세한 내용은 Amazon EC2 및 온프레미스 인스턴스 등록 단원을 참조하십시오.

> **Note**
>
> 일반적으로는 기존 Amazon EC2 인스턴스를 등록하게 됩니다. 하지만 이 안내에서는 새 인스턴스와 새 스택을 생성한 다음 마친 후 삭제할 수도 있습니다.

주제

- 1단계: 스택 및 인스턴스 생성
- 2단계: AWS CLI설치 및 구성
- 3단계: EC2Register 스택에 인스턴스 등록

1단계: 스택 및 인스턴스 생성

시작하려면 스택과 해당 스택에 등록할 Amazon EC2 인스턴스가 필요합니다.

스택 및 인스턴스를 생성하려면

1. AWS OpsWorks Stacks 콘솔을 사용하여 **EC2Register**(이)라는 이름의 새로운 스택을 생성합니다. 다른 스택 설정에 대해서는 기본값을 수락할 수 있습니다.
2. Amazon EC2 콘솔에서 새 인스턴스를 시작합니다. 다음을 참조하세요.
   - 인스턴스는 스택과 같은 리전 및 VPC에 있어야 합니다.

     VPC를 사용하는 경우 이 연습에서는 퍼블릭 서브넷을 선택합니다.
   - SSH 키를 생성해야 하는 경우 워크스테이션에 프라이빗 키 파일을 저장하고 파일의 이름 및 위치를 적어 둡니다.

기존 키를 사용하는 경우 이름 및 프라이빗 키 파일 위치를 적어 둡니다. 이러한 값은 나중에 필요합니다.

- 인스턴스는 지원되는 Linux 운영 체제 중 하나를 기반으로 해야 합니다. 예를 들어, 스택이 미국 서부(오레곤)에 있는 경우 `ami-35501205`을(를) 사용하여 해당 리전에 있는 Ubuntu 14.04 LTS 인스턴스를 시작할 수 있습니다.

그렇지 않으면 기본값을 수락합니다.

인스턴스가 부팅 중일 때 다음 섹션으로 넘어갈 수 있습니다.

2단계: AWS CLI설치 및 구성

등록은 명령을 사용하여 AWS CLI aws opsworks register 수행됩니다. 첫 번째 인스턴스를 등록하기 전에 AWS CLI 이상 버전 1.16.180을 실행해야 합니다. 설치 세부 정보는 워크스테이션의 운영 체제에 따라 다릅니다. 설치에 대한 자세한 내용은 AWS 명령줄 인터페이스 설치를 AWS CLI참조하세요. https://docs.aws.amazon.com/cli/latest/userguide/installing.html 실행 중인 AWS CLI 의 버전을 확인하려면 셸 세션에 `aws --version`을 입력합니다.

> **Note**
>
> 사용자나 역할이 인스턴스를 등록하지 못하게 하려면 register 명령에 대한 액세스를 거부하도록 인스턴스 프로파일을 업데이트하세요.

워크스테이션 AWS CLI 에서를 이미 실행하고 있더라도이 단계를 건너뛰지 않는 것이 좋습니다. 보안 모범 사례로 AWS CLI 의 가장 최신 릴리스를 사용하는 것이 좋습니다.

적절한 권한이 있는 AWS 자격 증명 세트와 함께 `register`를 제공해야 합니다. 보안 인증을 인스턴스에 직접 설치하지 않도록 하기 위해 권장되는 방법은 인스턴스 프로파일로 시작된 인스턴스를 등록한 다음 `register` 명령에 `--use-instance-profile` 스위치를 추가하는 것입니다. 인스턴스 프로파일에서 자격 증명을 얻는 경우 이 주제의 3단계: EC2Register 스택에 인스턴스 등록 단원으로 건너뛰십시오. 그러나 인스턴스 프로파일로 인스턴스가 실행되지 않은 경우 IAM 사용자를 생성할 수 있습니다. 다음 절차에서는 적절한 권한이 있는 새 사용자를 생성하고 이 사용자의 자격 증명을 워크스테이션에 설치한 다음 이러한 자격 증명을 `register`에 전달합니다.

> **Warning**
>
> IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

사용자를 생성하려면

1. IAM 콘솔의 탐색 창에서 사용자를 선택한 다음, 사용자 추가를 선택합니다.
2. **EC2Register**라는 사용자를 추가합니다.
3. Next(다음)를 선택합니다.
4. 권한 설정 페이지에서 직접 기존 정책 연결을 선택합니다.
5. **OpsWorks** 권한 정책 필터 상자에를 입력하여 AWS OpsWorks 정책을 표시하고 다음 정책 중 하나를 선택한 다음 다음: 검토를 선택합니다. 이 정책은 `register`를 실행하는 데 필요한 권한을 사용자에게 부여합니다.
   - 사용자 권한이 인스턴스 프로파일을 사용하는 EC2 인스턴스를 등록하도록 허용하려면 `AWSOpsWorksRegisterCLI_EC2`를 선택합니다.
   - 사용자 권한이 온프레미스 인스턴스를 등록하도록 허용하려면 `AWSOpsWorksRegisterCLI_OnPremises`를 선택합니다.
6. 다음을 선택합니다.
7. 검토 페이지에서 사용자 생성을 선택합니다.
8. 사용자에 대해 액세스 키를 생성합니다. 탐색 창에서 사용자를 선택한 후 액세스 키를 생성하려는 사용자를 선택합니다.
9. 보안 자격 증명 탭을 선택한 후 액세스 키 생성을 선택합니다.
10. 작업에 가장 적합한 액세스 키 모범 사례 및 대안을 선택합니다.
11. Next(다음)를 선택합니다.
12. (선택 사항) 액세스 키를 식별할 태그를 입력합니다.
13. Next(다음)를 선택합니다.
14. .csv 파일 다운로드를 선택하고 시스템의 원하는 위치에 자격 증명 파일을 저장한 다음 완료를 선택합니다.

IAM 사용자의 자격 증명을 `register`에 제공해야 합니다. 이 안내서에서는 EC2Register 자격 증명을 워크스테이션의 `credentials` 파일에 설치하여 이 작업을 처리합니다. 의 자격 증명을 관리하는 다른 방법에 대한 자세한 내용은 구성 및 자격 증명 파일을 AWS CLI참조하세요.

사용자의 자격 증명을 설치하려면

1. 워크스테이션의 `credentials` 파일을 생성하거나 엽니다. 이 파일은 `~/.aws/credentials`(Linux, Unix 및 OS X) 또는 `C:\Users\`*`User_Name`*`\.aws\credentials`(Windows 시스템)에 있습니다.
2. `credentials` 파일에 EC2Register 사용자에 대한 프로파일을 다음과 같은 형식으로 추가합니다.

   ```
   [ec2register]
   aws_access_key_id = access_key_id
   aws_secret_access_key = secret_access_key
   ```

   앞서 다운로드한 EC2Register 키로 *`access_key_id`* 및 *`secret_access_key`*를 바꿉니다.

3단계: EC2Register 스택에 인스턴스 등록

이제 인스턴스를 등록할 준비가 되었습니다.

인스턴스를 등록하려면

1. AWS OpsWorks 스택에서 EC2Register 스택으로 돌아가 탐색 창에서 인스턴스를 선택한 다음 인스턴스 등록을 선택합니다.
2. EC2 인스턴스를 선택하고, 다음: 인스턴스 선택을 클릭한 다음 목록에서 인스턴스를 선택합니다.
3. 다음: AWS CLI 설치 및 다음: 인스턴스 등록을 선택합니다. AWS OpsWorks 스택은 스택 ID 및 인스턴스 ID와 같은 사용 가능한 정보를 자동으로 사용하여 인스턴스 등록 페이지에 표시되는 `register` 명령 템플릿을 생성합니다. 이 예제에서는 `register`가 SSH 키를 사용하여 인스턴스에 로그인하도록 하고 키 파일을 명시적으로 지정할 것이기 때문에 SSH 키를 사용하여 인스턴스에 연결을 예로 설정합니다. 이 명령 템플릿은 다음과 유사합니다.

   ```
   aws opsworks register --infrastructure-class ec2 --region region endpoint ID
     --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username]
     --ssh-private-key [key-file] i-f1245d10
   ```

> **Note**
>
> 스택이 리전 엔드포인트와 연결된 클래식 리전 내에 있는 경우 리전을 스택의 리전이 아닌 AWS OpsWorks Stacks 서비스의 `us-east-1` 엔드포인트 리전으로 설정해야 합니다. AWS OpsWorks 스택은 스택 ID에서 스택의 리전을 결정합니다.

4. 명령 템플릿에는 사용자 지정 인수 값이 여러 개 포함되어 있는데, 이러한 인수 값은 대괄호로 표시되며 적절한 값으로 바꿔야 합니다. 명령 템플릿을 텍스트 편집기에 복사하고 다음과 같이 편집합니다.

> **Important**
>
> 등록 프로세스 중에 생성되는 IAM 사용자는 등록된 인스턴스의 수명이 끝날 때까지 필요합니다. 사용자를 삭제하면 AWS OpsWorks Stacks 에이전트가 서비스와 통신할 수 없게 됩니다. 사용자가 우연히 삭제될 경우 발생하는 등록된 인스턴스 관리 문제를 예방하려면 `register` 명령에 `--use-instance-profile` 파라미터를 추가하여 인스턴스의 내장 인스턴스 프로파일을 대신 사용합니다. 또한 `--use-instance-profile` 파라미터를 추가하면 에이전트가 사용할 수 있는 액세스 키와 필요한 IAM 사용자 간의 불일치를 AWS OpsWorks 방지하기 때문에 90일마다 AWS 계정 액세스 키를 교체할 때 오류가 발생하지 않습니다(권장하는 모범 사례).

- *`# ##`*은 인스턴스를 생성할 때 저장한 Amazon EC2 키 페어의 프라이빗 키 파일에 대한 정규화된 경로로 바꿉니다.

  원한다면 상대 경로를 사용해도 됩니다.
- *`username`*을 인스턴스의 사용자 이름으로 바꿉니다.

  이 예제에서 사용자 이름은 Ubuntu 인스턴스의 경우 `ubuntu`이고, Red Hat Enterprise Linux(RHEL) 또는 Amazon Linux 인스턴스의 경우 `ec2-user`입니다.
- 키 교체 중 또는 주요 IAM 사용자를 실수로 삭제한 경우 오류를 방지하기 위해 인스턴스 프로파일로 `register`을(를) 실행하는 `--use-instance-profile`을(를) 추가합니다.

명령은 다음과 유사합니다.

```
aws opsworks register --use-instance-profile --infrastructure-class ec2 \
```

```
  --region us-west-2  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-
username ubuntu \
  --ssh-private-key "./keys/mykeys.pem" i-f1245d10
```

5. 워크스테이션에서 터미널 창을 열고 편집기에서 `register` 명령을 붙여 넣은 다음 명령을 실행합니다.

   등록에는 일반적으로 약 5분 가량 걸립니다. 완료되면 AWS OpsWorks Stacks 콘솔로 돌아가서 완료를 선택합니다. 탐색 창에서 인스턴스를 선택합니다. 인스턴스가 [미할당 인스턴스]에 표시되어야 합니다. 그런 다음 계획한 인스턴스 관리 방법에 따라 해당 인스턴스를 계층에 할당하거나 그대로 둘 수 있습니다.

6. 완료되면 인스턴스를 중지한 다음 AWS OpsWorks Stacks 콘솔 또는 명령을 사용하여 인스턴스를 삭제합니다. 그러면 Amazon EC2 인스턴스가 종료되어 더 이상 요금이 발생하지 않습니다.

Amazon EC2 및 온프레미스 인스턴스 등록

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

이 섹션에서는 Amazon EC2 또는 온프레미스 인스턴스를 AWS OpsWorks Stacks 스택에 등록하는 방법을 설명합니다.

주제

- 인스턴스 준비
- AWS CLI설치 및 구성
- 인스턴스 등록
- register 명령 사용

- 예제 register 명령
- 인스턴스 등록 정책

인스턴스 준비

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

인스턴스를 등록하기 전에 AWS OpsWorks Stacks와 호환되는지 확인해야 합니다. 온프레미스를 등록하는지 Amazon EC2 인스턴스를 등록하는지에 따라 세부 사항이 달라집니다.

온프레미스 인스턴스

온프레미스 인스턴스는 다음 기준을 충족해야 합니다.

- 인스턴스는 지원되는 Linux 운영 체제 중 하나에서 실행되어야 합니다. 사용자 지정 또는 커뮤니티 생성 AMI에서 생성된 다른 운영 체제(예: CentOS 6.x)에서 인스턴스를 생성하거나 등록하는 것도 가능하지만 이런 운영 체제는 공식 지원되지 않습니다.

  인스턴스에 `libyaml` 패키지를 설치해야 합니다. Ubuntu 인스턴스의 경우, 패키지는 `libyaml-0-2`로 명명됩니다. CentOS 및 Red Hat Enterprise Linux 인스턴스의 경우, 패키지는 `libyaml`로 명명됩니다.
- 인스턴스는 지원되는 인스턴스 유형(경우에 따라 인스턴스 크기라고 함)이어야 합니다. 지원되는 인스턴스 유형은 운영 체제에 따라 다를 수 있으며, 스택이 VPC에 있는지에 따라 달라집니다. 지원되는 인스턴스 유형 목록을 보려면 대상 스택에서 새 인스턴스를 생성하려고 할 때 AWS OpsWorks Stacks 콘솔에 표시되는 크기 드롭다운 목록 값을 확인합니다. 인스턴스 유형이 회색으로 표시되어 있고 대상 스택에서 생성할 수 없다면 해당 유형의 인스턴스도 등록할 수 없습니다.

- 인스턴스에는 AWS OpsWorks Stacks 서비스 엔드포인트인와 통신할 수 있는 인터넷 액세스 권한이 있어야 합니다opsworks.us-east-1.amazonaws.com (HTTPS). 인스턴스는 Amazon S3 등 AWS 리소스로의 아웃바운드 연결도 지원해야 합니다.
- 별도의 워크스테이션에서 인스턴스를 등록하려는 경우, 등록된 인스턴스는 워크스테이션에서의 SSH 로그인을 지원해야 합니다.

  인스턴스에서 등록 명령을 실행하는 경우에는 SSH 로그인이 필요하지 않습니다.
- AWS 액세스 키는 에이전트에서 AWS OpsWorks AWS OpsWorks Stacks 서비스로 인증하는 데 사용됩니다. 90일마다 권장 사항에 따라 액세스 키를 교체하는 경우 AWS OpsWorks 에이전트를 수동으로 업데이트하여 새 키를 사용합니다. 온프레미스 컴퓨터 또는 인스턴스에서 새 액세스 키 및 암호 키로 /etc/aws/opsworks/instance-agent.yml 파일을 편집합니다. 다음 명령은 이 파일의 액세스 키와 암호 키를 표시합니다. 이전 키를 사용하는 에이전트는 오류를 일으킬 수 있습니다.

```
cat /etc/aws/opsworks/instance-agent.yml | egrep "access_key|secret_key"
:access_key_id: AKIAIOSFODNN7EXAMPLE
:secret_access_key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Amazon EC2 인스턴스

Amazon EC2 인스턴스는 다음 기준을 충족해야 합니다.

- AMI는 지원되는 Linux 운영 체제 중 하나에 기반해야 합니다. 현재 목록은 AWS OpsWorks 스택 운영 체제를 참조하세요.

  자세한 내용은 사용자 지정 AMI 사용 단원을 참조하십시오.

  인스턴스가 지원되는 표준 AMI에서 파생된 사용자 지정 AMI에 기반하거나 인스턴스에 극히 최소한의 설정이 포함된 경우, 인스턴스에 libyaml 패키지를 설치해야 합니다. Ubuntu 인스턴스의 경우, 패키지는 libyaml-0-2로 명명됩니다. Amazon Linux 및 Red Hat Enterprise Linux 인스턴스의 경우, 패키지는 libyaml(으)로 명명됩니다.
- 인스턴스는 지원되는 인스턴스 유형(경우에 따라 인스턴스 크기라고 함)이어야 합니다. 지원되는 인스턴스 유형은 운영 체제에 따라 다를 수 있으며, 스택이 VPC에 있는지에 따라 달라집니다. 지원되는 인스턴스 유형 목록을 보려면 대상 스택에서 새 인스턴스를 생성하려고 할 때 AWS OpsWorks Stacks 콘솔에 표시되는 크기 드롭다운 목록 값을 확인합니다. 인스턴스 유형이 회색으로 표시되어 있고 대상 스택에서 생성할 수 없다면 해당 유형의 인스턴스도 등록할 수 없습니다.
- 인스턴스는 running 상태여야 합니다.
- 인스턴스는 Auto Scaling 그룹의 일부가 아니어야 합니다.

자세한 내용은 Auto Scaling 그룹에서 EC2 인스턴스 분리를 참조하세요.

- 인스턴스는 VPC의 일부일 수 있지만 스택과 동일한 VPC에 있어야 하며 VPC가 AWS OpsWorks Stacks에서 제대로 작동하도록 구성되어야 합니다.
- 스팟 인스턴스는 자동 복구와 호환되지 않기 때문에 지원되지 않습니다.

Amazon EC2 인스턴스를 등록할 때 AWS OpsWorks Stacks는 인스턴스의 보안 그룹 또는 규칙을 수정하지 않습니다. 인스턴스의 보안 그룹 규칙이 다음 AWS OpsWorks Stacks 요구 사항과 일치하는지 확인합니다.

수신 규칙

수신 규칙은 다음을 허용해야 합니다.

- SSH 로그인.
- 적절한 계층으로부터의 트래픽.

  예를 들어 데이터베이스 서버는 일반적으로 스택의 애플리케이션 서버 계층으로부터의 인바운드 트래픽을 허용합니다.
- 적절한 포트로의 트래픽.

  예를 들어 애플리케이션 서버 인스턴스는 일반적으로 80번 포트(HTTP)와 443번 포트(HTTPS)로의 인바운드 트래픽을 허용합니다.

송신 규칙

송신 규칙은 다음을 허용해야 합니다.

- 인스턴스에서 실행되는 애플리케이션에서 AWS OpsWorks Stacks 서비스로의 트래픽입니다.
- AWS API를 사용하여 애플리케이션에서 Amazon S3와 같은 AWS 리소스에 액세스하는 트래픽.

한 가지 일반적인 방법은 송신 규칙을 지정하지 않음으로써 아웃바운드 트래픽에 제한을 두지 않는 것입니다.

AWS CLI설치 및 구성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

첫 번째 인스턴스를 등록하기 전에를 실행하는 컴퓨터에서 버전 1.16.180 AWS CLI 이상의를 실행해야 합니다`register`. 설치 세부 정보는 워크스테이션의 운영 체제에 따라 다릅니다. 설치에 대한 자세한 내용은 AWS 명령줄 인터페이스 설치 및 AWS 명령줄 인터페이스 구성을 AWS CLI참조하세요. https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html 실행 중인 AWS CLI 의 버전을 확인하려면 셸 세션에 `aws --version`을 입력합니다.

> **Note**
>
> PowerShell용 AWS 도구에는 `register` API 작업을 호출하는 `Register-OpsInstance` cmdlet이 포함되어 있지만 대신 AWS CLI 를 사용하여 `register` 명령을 실행하는 것이 좋습니다.

적절한 권한으로 `register`를 실행해야 합니다. IAM 역할을 사용하여 권한을 얻을 수 있습니다. 또는 최적의 방법은 아니지만, 적절한 권한이 있는 사용자 자격 증명을 워크스테이션이나 등록할 인스턴스에 설치하여 권한을 얻을 수도 있습니다. 그런 다음 나중에 설명하는 것처럼 이러한 자격 증명을 사용하여 `register`를 실행할 수 있습니다. IAM 정책을 사용자 또는 역할에 연결하여 권한을 지정합니다. `register`의 경우, `AWSOpsWorksRegisterCLI_EC2` 또는 `AWSOpsWorksRegisterCLI_OnPremises` 정책을 사용합니다. 이러한 정책은 각각 Amazon EC2 또는 온프레미스 인스턴스를 등록할 수 있는 권한을 부여합니다.

> **Note**
>
> Amazon EC2 인스턴스에서 `register`을(를) 실행하는 경우, 원칙적으로는 IAM 역할을 사용하여 자격 증명을 제공해야 합니다. 기존 인스턴스에 IAM 역할을 연결하는 방법에 대한 내용은 인스턴스에 IAM 역할 연결 또는 Amazon EC2 사용 설명서의 IAM 역할 교체를 참조하세요.

`AWSOpsWorksRegisterCLI_EC2` 및 `AWSOpsWorksRegisterCLI_OnPremises` 정책의 예제 코드 조각은 인스턴스 등록 정책 단원을 참조하세요. AWS 자격 증명 생성 및 관리에 대한 자세한 내용은 AWS 보안 자격 증명을 참조하세요.

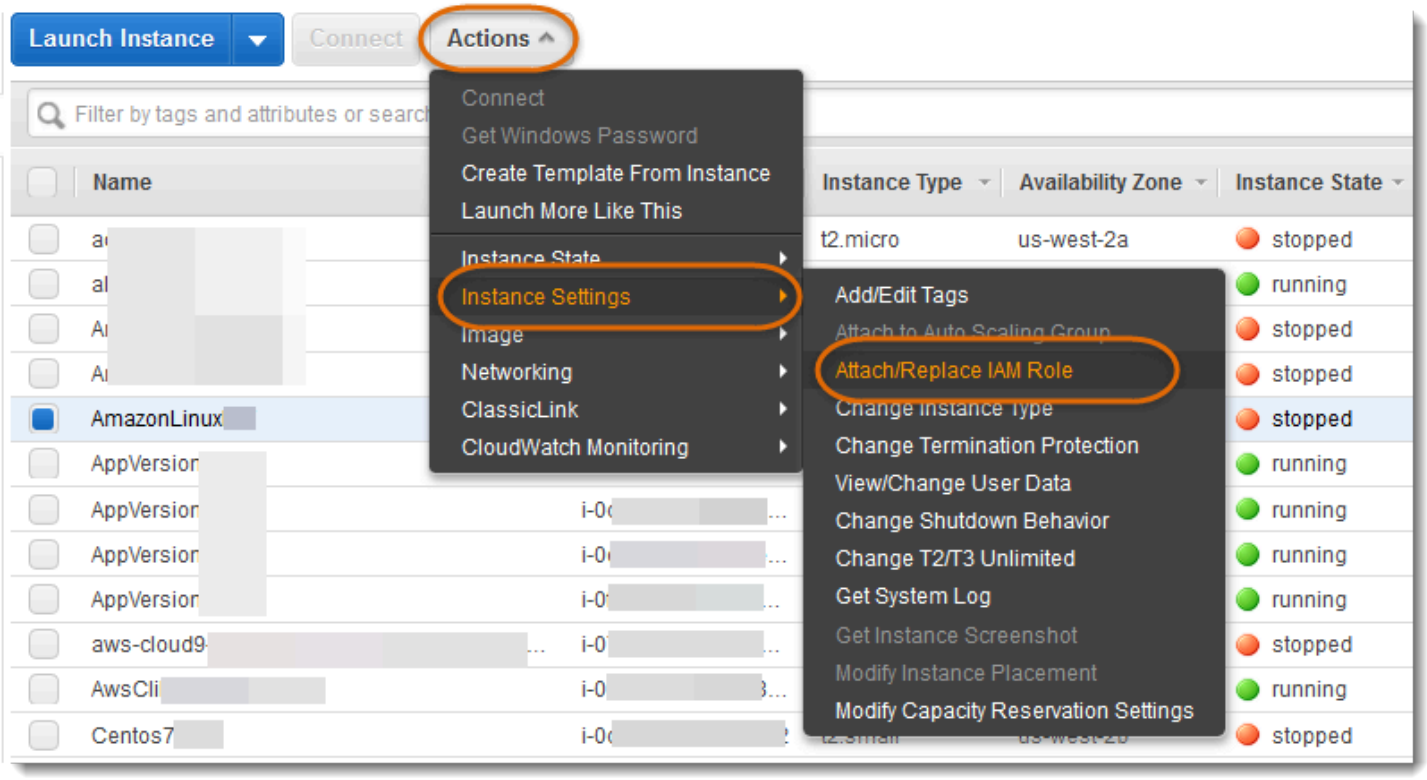
주제

- IAM 역할 사용

- 설치된 자격 증명 사용

IAM 역할 사용

등록하려는 Amazon EC2 인스턴스에서 명령을 실행하는 경우, `register`에 자격 증명을 제공하기 위한 기본 전략은 `AWSOpsWorksRegisterCLI_EC2` 정책 또는 동등한 권한이 연결된 IAM 역할을 사용하는 것입니다. 이 방법을 사용하면 인스턴스에 자격 증명을 설치하지 않아도 됩니다. 이렇게 하는 한 가지 방법은 다음 그림과 같이 EC2 콘솔에서 IAM 역할 연결/교체 명령을 사용하는 것입니다.



기존 인스턴스에 IAM 역할을 연결하는 방법에 대한 내용은 인스턴스에 IAM 역할 연결 또는 Amazon EC2 사용 설명서의 IAM 역할 교체를 참조하세요. 인스턴스 프로파일(권장)로 실행한 인스턴스의 경우 `register` 명령에 `--use-instance-profile` 스위치를 추가하여 자격 증명을 추가합니다. 단, `--profile` 파라미터는 사용하지 마십시오.

인스턴스가 실행 중이고 역할을 가지고 있는 경우, `AWSOpsWorksRegisterCLI_EC2` 정책을 역할에 연결하여 필요한 권한을 부여할 수 있습니다. 이 역할은 인스턴스에 기본 자격 증명 세트를 제공합니다. 인스턴스에 자격 증명을 설치하지 않았다면 `register`가 역할을 자동으로 수행하고 권한을 실행합니다.

> **Important**
>
> 인스턴스에 자격 증명을 설치하지 않는 것이 좋습니다. 보안 위험을 생성하는 것 외에도 인스턴스의 역할은가 기본 자격 증명을 찾는 데 AWS CLI 사용하는 기본 공급자 체인의 끝에 있습니다. 설치된 자격 증명이 역할보다 우선할 수 있으며 따라서 `register`가 필요한 권한을 갖지 못할 수 있습니다. 자세한 내용은 AWS CLI시작하기를 참조하세요.

실행 중인 인스턴스에 역할이 없는 경우, 설치된 자격 증명 사용에 설명된 것처럼 필요한 권한이 있는 자격 증명을 인스턴스에 설치해야 합니다. 인스턴스 프로파일을 사용하여 실행되는 인스턴스를 사용하는 것이 권장되는 방법으로, 더 쉽고, 오류가 적게 발생합니다.

설치된 자격 증명 사용

시스템에 사용자 자격 증명을 설치하고 AWS CLI 명령에 제공하는 방법에는 여러 가지가 있습니다. 아래에는 더 이상 권장되지 않는 접근 방식에 대한 설명이 나와 있지만, 인스턴스 프로파일을 사용하지 않고 실행한 EC2 인스턴스를 등록하는 경우에는 이러한 방식을 사용할 수 있습니다. 연결된 정책이 필요한 권한을 부여한다면 기존 사용자의 자격 증명을 사용할 수도 있습니다. 자격 증명을 설치하는 기타 방법에 대한 설명을 포함한 자세한 내용은 구성 및 자격 증명 파일을 참조하세요.

설치된 자격 증명을 사용하려면

1. IAM 사용자를 생성하고 안전한 위치에 액세스 키 ID 및 비밀 액세스 키를 저장합니다.

   > **Warning**
   >
   > IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

2. 이 사용자에게 AWSOpsWorksRegisterCLI_OnPremises 정책을 연결합니다. `AWSOpsWorksRegisterCLI_OnPremises` 권한을 포함하고 있는 한, 원한다면 더 광범위한 권한을 부여하는 정책을 연결할 수 있습니다.
3. 시스템의 `credentials` 파일에서 사용자에 대한 프로파일을 만듭니다. 이 파일은 `~/.aws/credentials`(Linux, Unix 및 OS X) 또는 `C:\Users\`*`User_Name`*`\.aws\credentials`(Windows 시스템)에 있습니다. 이 파일에는 다음 형식의 프로파일이 하나 이상 포함되어 있으며 각 프로파일에는 사용자의 액세스 키 ID 및 비밀 액세스 키가 포함되어 있습니다.

```
[profile_name]
aws_access_key_id = access_key_id
aws_secret_access_key = secret_access_key
```

IAM ## ### access_key_id # secret_access_key 값으로 대체하세요. 프로파일 이름에 원하는 이름을 지정할 수 있지만, 이름은 고유해야 하고 기본 프로파일의 이름은 `default`여야 한다는 두 가지 제한 사항이 있습니다. 또한 필요한 권한이 포함되어 있는 한 기존 프로파일을 사용할 수도 있습니다.

4. `register` 명령의 `--profile` 파라미터를 사용하여 프로파일 이름을 지정합니다. 그러면 `register` 명령이 연결된 자격 증명에 부여된 권한으로 실행됩니다.

   뿐만 아니라 `--profile`은 생략할 수도 있습니다. 이러한 경우 `register`는 기본 자격 증명으로 실행됩니다. 하지만 이러한 자격 증명이 기본 프로파일의 자격 증명이 아닐 수도 있기 때문에 기본 자격 증명에 필수 권한이 있는지 확인해야 합니다. 가 기본 자격 증명을 AWS CLI 결정하는 방법에 대한 자세한 내용은 AWS 명령줄 인터페이스 구성을 참조하세요.

인스턴스 등록

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 기능은 Linux 스택에서만 지원됩니다.

워크스테이션 또는 인스턴스에서 명령을 실행 AWS CLI `register`하여 인스턴스를 등록합니다. 이 작업을 처리하는 가장 간단한 방법은 명령 문자열 생성 프로세스를 단순화하는 AWS OpsWorks Stacks 콘솔의 등록 마법사를 사용하는 것입니다. 등록 절차에 익숙해진 후에는 원한다면 마법사를 건너뛰고 `register` 명령을 실행할 수 있습니다.

다음은 등록 마법사를 사용하여 인스턴스를 기존 스택에 등록하는 방법을 설명합니다.

> **Note**
>
> 인스턴스를 새 스택에 등록하려면 AWS OpsWorks Stacks 대시보드에서 인스턴스 등록을 선택하면 됩니다. 이렇게 하면 새 스택을 구성하는 추가 페이지만 제외하고 기존 스택의 마법사와 동일한 마법사가 시작됩니다.

등록 마법사를 사용하여 인스턴스를 등록하려면

1. AWS OpsWorks Stacks 콘솔에서 스택을 생성하거나 기존 스택을 엽니다.
2. 탐색 창에서 [인스턴스]를 선택한 다음 [인스턴스 등록]을 선택합니다.
3. 인스턴스 유형 추가 페이지에서 Amazon EC2 또는 온프레미스 인스턴스를 등록할지 여부를 지정합니다.
   - Amazon EC2 인스턴스를 등록 중인 경우 다음: 인스턴스 선택을 선택합니다.
   - 온프레미스 인스턴스를 등록 중인 경우 다음: AWS CLI 설치를 선택한 다음 5단계로 진행합니다.
4. Amazon EC2 인스턴스를 등록하는 경우 인스턴스 선택 페이지를 열어 등록할 인스턴스를 선택합니다. AWS OpsWorks Stacks는 명령을 빌드하는 데 필요한 정보를 수집합니다. 완료되면 다음: AWS CLI 설치를 선택합니다.
5. 실행하려는 인스턴스는 버전 1.16.180 AWS CLI 이상의를 실행해야 `register` 합니다. AWS CLI를 설치하거나 업데이트할 수 있도록 등록 마법사 페이지에서는 설치 및 구성 지침에 대한 링크를 제공합니다. AWS CLI 설치를 확인한 후, 등록할 인스턴스 또는 개별 워크스테이션에서 명령을 실행 중인지 여부를 선택한 다음, 다음: 인스턴스 등록을 선택합니다.
6. [인스턴스 등록] 페이지에는 선택한 옵션을 포함한 `register` 명령 문자열에 대한 템플릿이 표시됩니다. 예를 들어, 별도의 워크스테이션에서 Amazon EC2 인스턴스를 등록하면 기본 템플릿은 다음과 유사합니다.

```
aws opsworks register --infrastructure-class ec2 --region us-west-2
  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username] i-
f1245d10
```

> **Important**
>
> 등록 프로세스 중에 생성되는 IAM 사용자는 등록된 인스턴스의 수명이 끝날 때까지 필요합니다. 사용자를 삭제하면 AWS OpsWorks Stacks 에이전트가 서비스와 통신할 수 없게

됩니다. 사용자가 우연히 삭제될 경우 발생하는 등록된 인스턴스 관리 문제를 예방하려면 `register` 명령에 `--use-instance-profile` 파라미터를 추가하여 인스턴스의 내장 인스턴스 프로파일을 대신 사용합니다. 또한 `--use-instance-profile` 파라미터를 추가하면 에이전트가 사용할 수 있는 액세스 키와 필요한 IAM 사용자 간의 불일치를 AWS OpsWorks 방지하기 때문에 90일마다 AWS 계정 액세스 키를 교체할 때 오류가 발생하지 않습니다(권장하는 모범 사례).

SSH 키 사용을 예로 설정하면 AWS OpsWorks Stacks는 프라이빗 SSH 키 파일을 지정하는 데 사용할 수 있는 `--ssh-private-key` 인수를 문자열에 추가합니다.

> **Note**
>
> 비밀번호로 `register`이(가) 로그온하도록 하려면 SSH 키 사용을 아니요로 설정하세요. `register`을(를) 실행하면 암호를 입력하라는 메시지가 표시됩니다.

이 문자열을 텍스트 편집기에 복사하고 필요에 따라 편집합니다. 다음을 참조하세요.

- 대괄호로 묶인 텍스트는 입력해야 하는 정보(예: SSH 키 파일의 위치)를 나타냅니다.
- 이 템플릿에서는 기본 AWS 자격 증명을 사용하여 `register`를 실행 중이라고 가정합니다. 그렇지 않은 경우 명령 문자열에 `--profile` 인수를 추가하고 사용하려는 자격 증명 프로파일 이름을 지정합니다.

다른 시나리오의 경우 명령을 좀 더 변경해야 할 수 있습니다. 사용 가능한 `register` 인수와 명령 문자열을 구성하는 대체 방법에 대한 설명은 `register` 명령 사용 단원을 참조하세요. 또한 명령 줄에서 `aws opsworks help register`를 실행하여 명령의 문서를 표시할 수도 있습니다. 몇 가지 예제 명령 문자열은 예제 register 명령 단원을 참조하세요.

7. 명령 문자열 편집을 마치면 워크스테이션에서 터미널 창을 열거나 SSH를 사용하여 인스턴스에 로그온하고 명령을 실행합니다. 전체 작업에는 일반적으로 약 5분 가량이 소요되며 그 동안 인스턴스의 상태는 [등록 중]입니다.
8. 작업이 완료되면 [완료]를 선택합니다. 이제 인스턴스는 [등록됨] 상태이고 스택의 [인스턴스] 페이지에 할당되지 않은 인스턴스로 나열됩니다.

`register` 명령은 다음 작업을 수행합니다.

1. `register`가 워크스테이션에서 실행 중인 경우, 이 명령은 먼저 SSH를 사용하여 등록할 인스턴스에 로그인합니다.

   나머지 프로세스는 인스턴스에서 이루어지며, 명령을 어디서 실행하는가에 상관없이 동일합니다.
2. Amazon S3에서 AWS OpsWorks Stacks 에이전트 패키지를 다운로드합니다.
3. 에이전트와 AWS SDK for Ruby 같은 에이전트의 종속성을 압축 해제하고 설치합니다.
4. 다음을 생성합니다.
   - 에이전트를 AWS OpsWorks Stacks 서비스로 부트스트랩하여 보안 통신을 제공하는 IAM 사용자입니다.

     이 사용자의 권한은 `opsworks:RegisterInstance` 작업만을 허용하며, 15분 후 만료됩니다.
   - 등록된 인스턴스의 사용자가 포함된 스택의 IAM 그룹.
5. RSA 키 페어를 생성하고 퍼블릭 키를 AWS OpsWorks Stacks로 전송합니다.

   이 키 페어는 에이전트와 AWS OpsWorks Stacks 간 통신을 암호화하는 데 사용됩니다.
6. 인스턴스를 AWS OpsWorks Stacks에 등록합니다. 그러면 스택이 초기 설정 레시피 세트를 실행하여 다음과 같이 인스턴스를 구성합니다.
   - 인스턴스의 호스트 파일 덮어쓰기.

     인스턴스를 등록하면 SSH 로그인 권한을 제어하기 위한 자체 호스트 파일이 있어야 하는 AWS OpsWorks Stacks에 사용자 관리를 인계했습니다.
   - Amazon EC2 인스턴스의 경우, 연결된 Amazon EBS 볼륨 또는 탄력적 IP 주소를 스택에 등록하는 것 역시 초기 설정에 포함됩니다.

     Amazon EBS 볼륨이 `/var/www`을(를) 비롯한 예약된 탑재 지점과 인스턴스의 계층에 의해 예약된 탑재 지점에 탑재되지 않아야 합니다. 스택 리소스 관리에 대한 자세한 정보는 리소스 관리 단원을 참조하세요. 계층 탑재 지점에 대한 자세한 정보는 AWS OpsWorks Stacks 계층 참조 단원을 참조하세요.

   초기 설정 구성 변경에 대한 전체 설명은 초기 설정 구성 변경 단원을 참조하세요.

> **Note**
>
> 초기 설정은 등록된 인스턴스의 운영 체제를 업데이트하지 않으므로 이 작업은 직접 처리해야 합니다. 자세한 내용은 보안 업데이트 관리 단원을 참조하십시오.

`register` 명령 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

인스턴스를 등록하려면 최소한 버전 1.16.180 이상의 AWS CLI를 실행하고 있어야 합니다. 다음은 `register` 명령의 일반적 구문을 보여 줍니다.

```
aws opsworks register \
  [--profile profile_name] \
  [--region region_name] \
  --infrastructure-class instance_type \
  --stack-id stack ID \
  [--local] | [--ssh-private-key key_file --ssh-username username] | [--override-
ssh command_string] \
  [--override-hostname hostname] \
  [--debug] \
  [--override-public-ip public IP] \
  [--override-private-ip private IP] \
..[--use-instance-profile] \
  [ [IP address] | [hostname] | [instance ID]
```

다음 인수는 모든 AWS CLI 명령에 공통입니다.

`--profile`

(선택 사항) 자격 증명의 프로파일 이름. 이 인수를 생략하면 명령이 기본 자격 증명으로 실행됩니다. 가 기본 자격 증명을 AWS CLI 결정하는 방법에 대한 자세한 내용은 AWS 명령줄 인터페이스 구성을 참조하세요.

`--region`

(선택 사항) AWS OpsWorks Stacks 서비스 엔드포인트의 리전입니다. 스택의 리전`--region`으로 설정하지 마세요. AWS OpsWorks 스택은 스택 ID에서 스택의 리전을 자동으로 결정합니다.

> **Note**
>
> 기본 리전이 이미 설정되어 있다면 이 인수를 생략해도 됩니다. 기본 리전을 지정하는 방법에 대한 자세한 내용은 AWS 명령줄 인터페이스 구성을 참조하세요.

Amazon EC2 인스턴스와 온프레미스 인스턴스에 다음 인수를 사용하세요.

`--infrastructure-class`

(필수) 이 파라미터는 Amazon EC2 인스턴스를 등록하는지 또는 온프레미스 인스턴스를 등록하는지 표시하기 위해 각각 `ec2` 또는 `on-premises`(으)로 설정해야 합니다.

`--stack-id`

(필수) 인스턴스를 등록할 스택의 ID.

> **Note**
>
> 스택 ID를 찾으려면 스택 페이지에서 설정을 클릭합니다. 스택 ID는 [OpsWorks ID]로 레이블이 지정되어 있으며, `ad21bce6-7623-47f1-bf9d-af2affad8907`와 비슷한 GUID입니다.

SSH 로그인 인수

다음 인수를 사용하여 `register`가 인스턴스에 어떻게 로그인할지 지정합니다.

`--local`

(선택 사항) 명령을 실행하는 인스턴스를 등록하는 데 이 인수를 사용합니다.

이 경우, `register`가 인스턴스에 로그인할 필요가 없습니다.

`--ssh-private-key` 및 **`--ssh-username`**

(선택 사항) 별도의 워크스테이션에서 인스턴스를 등록하고 사용자 이름 또는 프라이빗 키 파일을 명시적으로 지정하려는 경우, 이 인수를 사용합니다.

- `--ssh-username` – SSH 사용자 이름을 지정하려면 이 인수를 사용합니다.

  `--ssh-username`를 생략하는 경우, `ssh`는 기본 사용자 이름을 사용합니다.
- `--ssh-private-key` – 프라이빗 키 파일을 명시적으로 지정하려면 이 인수를 사용합니다.

  `--ssh-private-key`를 생략하는 경우, `ssh`는 기본 프라이빗 키 사용 등 암호가 필요 없는 인증 기법을 사용하여 로그인을 시도합니다. 이러한 기법이 모두 지원되지 않는 경우, `ssh`가 암호를 쿼리합니다. `ssh`가 인증을 처리하는 방법에 대한 자세한 정보는 Secure Shell(SSH) 인증 프로토콜 단원을 참조하세요.

`--override-ssh`

(선택 사항) 별도의 워크스테이션에서 인스턴스를 등록하고 `ssh` 명령 문자열을 지정하려는 경우, 이 인수를 사용합니다. `register` 명령은 이 명령 문자열을 사용하여 등록된 인스턴스에 로그인합니다.

`ssh`에 대한 자세한 정보는 SSH를 참조하세요.

`--override-hostname`

(선택 사항) AWS OpsWorks Stacks에서만 사용되는 인스턴스의 호스트 이름을 지정합니다. 기본값은 인스턴스의 호스트 이름입니다.

`--debug`

(선택 사항) 등록 프로세스가 실패하는 경우, 디버깅 정보를 제공합니다. 문제 해결 정보는 인스턴스 등록 문제 해결를 참조하세요.

`--use-instance-profile`

선택 사항이지만 Amazon EC2 인스턴스에 대해 매우 권장되는 옵션입니다. `register` 명령은 IAM 사용자를 생성하는 대신 연결된 인스턴스 프로파일을 사용합니다. 이 파라미터를 추가하면 IAM 사용자가 우연히 삭제되었을 때 등록된 인스턴스를 관리하려 시도하는 경우에 발생하는 오류를 예방하는 데 도움이 됩니다.

> **Important**
>
> 등록 프로세스 중에 생성되는 IAM 사용자는 등록된 인스턴스의 수명이 끝날 때까지 필요합니다. 사용자를 삭제하면 AWS OpsWorks Stacks 에이전트가 서비스와 통신할 수 없게 됩니다. 사용자가 우연히 삭제될 경우 발생하는 등록된 인스턴스 관리 문제를 예방하려면 `register` 명령에 `--use-instance-profile` 파라미터를 추가하여 인스턴스의 내장 인스턴스 프로파일을 대신 사용합니다. 또한 `--use-instance-profile` 파라미터를 추가

하면 AWS OpsWorks 90일마다 AWS 계정 액세스 키를 교체할 때 오류가 발생하지 않습니다(권장 모범 사례). 에이전트가 사용할 수 있는 액세스 키와 필요한 사용자 간의 불일치를 방지하기 때문입니다.

대상

(조건부) 이 명령을 워크스테이션에서 실행하는 경우, 명령 문자열의 최종 값은 다음 방법 중 하나로 등록 대상을 지정합니다.

- 인스턴스의 퍼블릭 IP 주소.
- 인스턴스의 호스트 이름.
- Amazon EC2 인스턴스의 경우, 인스턴스 ID입니다.

  AWS OpsWorks Stacks는 인스턴스 ID를 사용하여 인스턴스의 퍼블릭 IP 주소를 포함한 인스턴스 구성을 가져옵니다. 기본적으로 AWS OpsWorks Stacks는이 주소를 사용하여 인스턴스에 로그인하는 데 사용하는 `ssh` 명령 문자열을 구성합니다. 프라이빗 IP 주소에 연결해야 하는 경우, `--override-ssh`를 사용하여 사용자 지정 명령 문자열을 제공해야 합니다. 예시는 워크스테이션에서 온프레미스 인스턴스 등록에서 확인하십시오.

> **Note**
>
> 호스트 이름을 지정하면 `ssh`는 DNS 서버에 의존하여 이름을 특정 인스턴스로 확인합니다. 호스트 이름이 고유한지 확실하지 않으면 `ssh`를 사용하여 호스트 이름이 올바른 인스턴스로 확인되는지 확인합니다.

등록할 인스턴스에서 이 명령을 실행하는 경우, 인스턴스 ID를 생략하고 대신 `--local` 인수를 사용하세요.

다음 인수는 온프레미스 인스턴스에만 사용합니다.

`--override-public-ip`

(선택 사항) AWS OpsWorks Stacks는 지정된 주소를 인스턴스의 퍼블릭 IP 주소로 표시합니다. 인스턴스의 퍼블릭 IP 주소는 변경하지 않습니다. 그러나 사용자가 인스턴스 페이지에서 주소를 선택하는 등 콘솔을 사용하여 인스턴스에 연결하는 경우 AWS OpsWorks Stacks는 지정된 주소를 사용합니다. AWS OpsWorks Stacks는 인수의 기본값을 자동으로 결정합니다.

--override-private-ip

(선택 사항) AWS OpsWorks Stacks는 지정된 주소를 인스턴스의 프라이빗 IP 주소로 표시합니다. 인스턴스의 프라이빗 IP 주소는 변경되지 않습니다. AWS OpsWorks 스택은 인수의 기본값을 자동으로 결정합니다.

예제 register 명령

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

이 섹션에는 `register` 명령 문자열의 몇 가지 예가 포함되어 있습니다.

워크스테이션에서 Amazon EC2 인스턴스 등록

다음 예제는 워크스테이션에서 Amazon EC2 인스턴스를 등록합니다. 명령 문자열은 기본 자격 증명을 사용하며, Amazon EC2 인스턴스 ID로 인스턴스를 식별합니다. `ec2`을(를) `on-premises`(으)로 변경하면 예제를 온프레미스 인스턴스에 사용할 수 있습니다.

```
aws opsworks register \
  --region us-west-2 \
  --use-instance-profile \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --ssh-user-name my-sshusername \
  --ssh-private-key "./keys/mykeys.pem" \
  i-2422b9c5
```

워크스테이션에서 온프레미스 인스턴스 등록

다음 예제는 별도의 워크스테이션에서 온프레미스 인스턴스를 등록합니다. 명령 문자열은 기본 자격 증명을 사용하며, 지정된 ssh 명령 문자열로 인스턴스에 로그인합니다. 인스턴스에 암호가 필요한 경우, register가 메시지를 표시합니다. on-premises을(를) ec2(으)로 변경하면 예제를 Amazon EC2 인스턴스에 사용할 수 있습니다.

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --override-ssh "ssh your-user@192.0.2.0"
```

> **Note**
>
> --override-ssh를 사용하여 사용자 지정 SSH 명령 문자열을 지정할 수 있습니다. 그런 다음 AWS OpsWorks Stacks는 명령 문자열을 구성하는 대신 지정된 문자열을 사용하여 인스턴스에 로그인합니다. 다른 예제는 사용자 지정 SSH 명령 문자열을 사용하여 인스턴스 등록를 참조하세요.

사용자 지정 SSH 명령 문자열을 사용하여 인스턴스 등록

다음 예제는 워크스테이션에서 온프레미스 인스턴스를 등록하며, --override-ssh 인수를 사용해 register이(가) 인스턴스 로그인에 사용할 사용자 지정 SSH 명령을 지정합니다. 이 예제는 sshpass를 사용하여 사용자 이름과 암호로 로그인하지만 유효한 어떤 ssh 명령 문자열도 지정할 수 있습니다.

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
  --override-ssh "sshpass -p 'mypassword' ssh your-user@192.0.2.0"
```

인스턴스에서 **register** 명령을 실행하여 인스턴스 등록

다음 예제는 인스턴스 자체에서 register을(를) 실행하여 Amazon EC2 인스턴스를 등록하는 방법을 보여 줍니다. 명령 문자열은 권한을 기본 자격 증명에 의존합니다. 예제를 온프레미스 인스턴스에 사용하려면 --infrastructure-class을(를) on-premises(으)로 변경하세요.

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --local
```

프라이빗 IP 주소가 있는 인스턴스 등록

기본적으로 `register`는 인스턴스의 퍼블릭 IP 주소를 사용하여 인스턴스에 로그인합니다. VPC의 프라이빗 서브넷에 있는 인스턴스처럼 프라이빗 IP 주소가 있는 인스턴스를 등록하려면 `--override-ssh`를 사용하여 사용자 지정 ssh 명령 문자열을 지정해야 합니다.

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class ec2 \
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
  --override-ssh "ssh -i mykey.pem ec2-user@10.183.201.93" \
  i-2422b9c5
```

인스턴스 등록 정책

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`AWSOpsWorksRegisterCLI_EC2` 및 `AWSOpsWorksRegisterCLI_OnPremises` 정책은 각각 EC2 및 온프레미스 인스턴스를 등록하기 위한 올바른 권한을 제공합니다. EC2 인스턴스를 등록하려면 `AWSOpsWorksRegisterCLI_EC2`을(를) IAM 사용자에게 추가하지만, 온프레미스 인스턴스를 등록하려면 `AWSOpsWorksRegisterCLI_OnPremises`을(를) 사용자에게 추가합니다. 이러한 정책을 사용하려면 버전 1.16.180 AWS CLI 이상의 이상을 실행해야 합니다.

AWSOpsWorksRegisterCLI_EC2 정책

EC2 인스턴스를 등록하려면 AWSOpsWorksRegisterCLI_EC2을(를) 사용자에게 추가합니다. EC2 인스턴스만 등록할 계획인 경우 이 프로파일을 사용해야 합니다. 이 정책을 사용하면 EC2 인스턴스의 인스턴스 프로파일에서 권한이 제공됩니다.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "opsworks:AssignInstance",
            "opsworks:CreateLayer",
            "opsworks:DeregisterInstance",
            "opsworks:DescribeInstances",
            "opsworks:DescribeStackProvisioningParameters",
            "opsworks:DescribeStacks",
            "opsworks:UnassignInstance"
          ],
          "Resource": [
            "*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "ec2:DescribeInstances"
          ],
          "Resource": [
            "*"
          ]
        }
      ]
    }
```

AWSOpsWorksRegisterCLI_OnPremises 정책

온프레미스 인스턴스를 등록하려면 AWSOpsWorksRegisterCLI_OnPremises을(를) 사용자에게 추가합니다. 이 정책에는 AttachUserPolicy와(과) 같은 IAM 권한이 포함되지만, 이러한 권한이 적용되는 리소스가 제한됩니다.

```
    {
```

```
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "opsworks:AssignInstance",
          "opsworks:CreateLayer",
          "opsworks:DeregisterInstance",
          "opsworks:DescribeInstances",
          "opsworks:DescribeStackProvisioningParameters",
          "opsworks:DescribeStacks",
          "opsworks:UnassignInstance"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeInstances"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "iam:CreateGroup",
          "iam:AddUserToGroup"
        ],
        "Resource": [
          "arn:aws:iam::*:group/AWS/OpsWorks/OpsWorks-*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "iam:CreateUser",
          "iam:CreateAccessKey"
        ],
        "Resource": [
          "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
```

```
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:AttachUserPolicy"
          ],
          "Resource": [
            "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
          ],
          "Condition": {
            "ArnEquals":
              {
                "iam:PolicyARN": "arn:aws:iam::aws:policy/
AWSOpsWorksInstanceRegistration"
              }
            }
        }
      ]
    }
```

(사용되지 않음) **AWSOpsWorksRegisterCLI** 정책

> **Important**
>
> AWSOpsWorksRegisterCLI 정책은 더 이상 사용되지 않으며 새 인스턴스를 등록하는 데 사용할 수 없습니다. 이 정책은 이미 등록된 인스턴스에서 이전 버전과의 호환성을 위해서만 사용할 수 있습니다. AWSOpsWorksRegisterCLI 정책에는 CreateUser, PutUserPolicy, AddUserToGroup 등의 수많은 IAM 권한이 포함됩니다. 이러한 권한은 관리자 수준 권한이기 때문에 신뢰할 수 있는 관리 사용자에게만 AWSOpsWorksRegisterCLI 정책을 할당해야 합니다.

등록된 인스턴스 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

인스턴스를 등록하면 인스턴스가 AWS OpsWorks Stacks 인스턴스가 되고 AWS OpsWorks Stacks로 생성된 인스턴스와 거의 동일한 방식으로 인스턴스를 관리할 수 있습니다. 두 가지 주요 차이점이 있습니다.

- 등록된 인스턴스는 계층에 할당할 필요가 없습니다.
- 등록된 인스턴스는 등록 해제하여 직접 제어로 되돌릴 수 있습니다.

인스턴스를 등록한 후 인스턴스는 등록된 상태입니다. AWS OpsWorks Stacks는 등록된 모든 인스턴스에 다음과 같은 관리 기능을 제공합니다.

- 상태 확인 - AWS OpsWorks 스택은 에이전트를 모니터링하여 인스턴스가 계속 작동하는지 평가합니다.

  인스턴스가 상태 확인에 실패하면 AWS OpsWorks Stacks는 등록된 Amazon EC2 인스턴스를 자동으로 복구하고 등록된 온프레미스 인스턴스의 상태를 로 변경합니다`connection lost`. Amazon EC2
- CloudWatch 모니터링 – 등록된 인스턴스에 대해 모니터링이 활성화됩니다.

  CPU 사용률과 가용 메모리 같은 측정치를 모니터링하고 필요한 경우, 측정치가 지정된 임계값을 초과하면 알림을 수신할 수 있습니다.
- 사용자 관리 - AWS OpsWorks Stacks는 인스턴스에 액세스할 수 있는 사용자와 수행할 수 있는 작업을 지정하는 간단한 방법을 제공합니다. 자세한 내용은 사용자 권한 관리 단원을 참조하십시오.
- 레시피 실행 – 레시피 실행 스택 명령을 사용하여 인스턴스에서 Chef 레시피를 실행할 수 있습니다.
- 운영 체제 업데이트 – 종속 업데이트 스택 명령을 사용하여 인스턴스의 운영 체제를 업데이트할 수 있습니다.

AWS OpsWorks Stacks 관리 기능을 최대한 활용하려면 인스턴스를 계층에 할당하면 됩니다. 자세한 내용은 등록된 인스턴스를 계층에 할당 단원을 참조하십시오.

AWS OpsWorks Stacks가 Amazon EC2 인스턴스를 관리하는 방식에는 차이가 있습니다.

Amazon EC2 인스턴스

- 등록된 Amazon EC2 인스턴스를 중지하면 AWS OpsWorks Stacks는 인스턴스 스토어 지원 인스턴스를 종료하고 Amazon EBS 지원 인스턴스를 중지합니다.

  인스턴스는 여전히 스택에 등록되어 있고 계층에 할당되어 있으므로 필요하다면 다시 시작할 수 있습니다. 등록된 인스턴스를 스택에서 제거하려면 명시적으로 등록을 해제하거나 인스턴스를 삭제하여 자동으로 등록 해제해야 합니다.
- 등록된 Amazon EC2 인스턴스를 다시 시작하거나 인스턴스가 실패하여 자동 복구되는 경우, Amazon EC2를 사용하여 인스턴스를 중단하고 다시 시작하는 것과 결과는 동일합니다. 다음과 같은 차이점이 있습니다.
  - 인스턴스 스토어 지원 인스턴스 - AWS OpsWorks Stacks는 동일한 AMI로 새 인스턴스를 시작합니다.

    AWS OpsWorks Stacks는 소프트웨어 패키지 설치와 같이 인스턴스가 등록되기 전에 수행한 작업에 대해 알지 못합니다. Stacks가 시작 시 패키지를 설치하거나 다른 구성 작업을 수행하도록 AWS OpsWorks 하려면 필요한 작업을 수행하는 사용자 지정 Chef 레시피를 제공하고 적절한 계층의 설정 이벤트에 할당해야 합니다.
  - Amazon EBS 지원 인스턴스 - AWS OpsWorks Stacks는 동일한 AMI로 새 인스턴스를 시작하고 루트 볼륨을 다시 연결하여 인스턴스를 이전 구성으로 복원합니다.
- 등록된 Amazon EC2 인스턴스를 등록 해제하면 일반 Amazon EC2 인스턴스로 되돌아갑니다.

온프레미스 인스턴스

- AWS OpsWorks Stacks는 등록된 온프레미스 인스턴스를 중지하거나 시작할 수 없습니다.

  등록된 온프레미스 인스턴스 할당 해제는 Shutdown 이벤트를 트리거합니다. 다만 이 이벤트는 할당된 계층의 Shutdown 레시피를 실행할 뿐입니다. 이 레시피는 서비스 종료 같은 작업을 수행하지만 인스턴스를 중지하지는 않습니다.
- AWS OpsWorks Stacks는 등록된 온프레미스 인스턴스가 실패하면 자동으로 복구할 수 없지만 인스턴스는 연결이 끊긴 것으로 표시됩니다.
- 온프레미스 인스턴스는 Elastic Load Balancing, Amazon EBS 또는 탄력적 IP 주소 서비스를 사용할 수 없습니다.

등록된 인스턴스를 계층에 할당

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 스택에서만 지원됩니다.

인스턴스를 등록한 후 하나 이상의 계층에 인스턴스를 할당할 수 있습니다. 인스턴스를 할당되지 않은 상태로 두는 대신 계층에 인스턴스를 할당하면 계층의 수명 주기 이벤트에 사용자 지정 레시피를 할당할 수 있다는 이점이 있습니다. 그러면 AWS OpsWorks 스택은 해당 이벤트에 대한 계층의 레시피 이후에 적절한 시간에 인스턴스를 자동으로 실행합니다.

- 등록된 모든 인스턴스는 사용자 지정 계층에 할당할 수 있습니다. 사용자 지정 계층에는 패키지를 전혀 설치하지 않는 최소한의 레시피 세트만 있으므로 인스턴스의 기존 구성과 충돌하지 않습니다.
- 온프레미스 인스턴스를 AWS OpsWorks Stacks 내장 계층에 할당할 수 있습니다.

  모든 내장 계층에는 자동으로 하나 이상의 패키지를 설치하는 레시피가 포함되어 있습니다. 예를 들어 Java 앱 서버 설정 레시피는 Apache와 Tomcat을 설치합니다. 계층의 레시피는 서비스 재시작 및 애플리케이션 배포 같은 다른 작업도 수행할 수 있습니다. 내장 계층에 온프레미스 인스턴스를 할당하기 전에 현재 인스턴스에 있는 버전과 다른 애플리케이션 서버 버전을 설치하려 시도하는 등 계층의 레시피가 충돌을 일으키지 않는지 확인해야 합니다. 자세한 내용은 계층 및 AWS OpsWorks Stacks 계층 참조 단원을 참조하세요.

계층에 등록된 인스턴스 할당

1. 사용할 계층을 아직 스택에 추가하지 않은 경우 지금 추가합니다.
2. 탐색 창에서 인스턴스를 선택한 다음 인스턴스의 작업 열에서 할당을 선택합니다.
3. 해당 계층을 선택하고 저장을 선택합니다.

계층에 인스턴스를 할당할 때 AWS OpsWorks Stacks는 다음을 수행합니다.

- 계층의 설정 레시피를 실행합니다.
- 연결된 탄력적 IP 주소 또는 Amazon EBS 볼륨을 스택의 리소스에 추가합니다.

  그런 다음 AWS OpsWorks Stacks를 사용하여 이러한 리소스를 관리할 수 있습니다. 자세한 내용은 리소스 관리 단원을 참조하십시오.

인스턴스가 완료되면 인스턴스는 온라인 상태가 되고 스택에 완전히 통합됩니다. 그러면 AWS OpsWorks 스택은 수명 주기 이벤트가 발생할 때마다 계층에 할당된 레시피를 실행합니다.

등록된 인스턴스 할당 해제

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 기능은 Linux 스택에서만 지원됩니다.

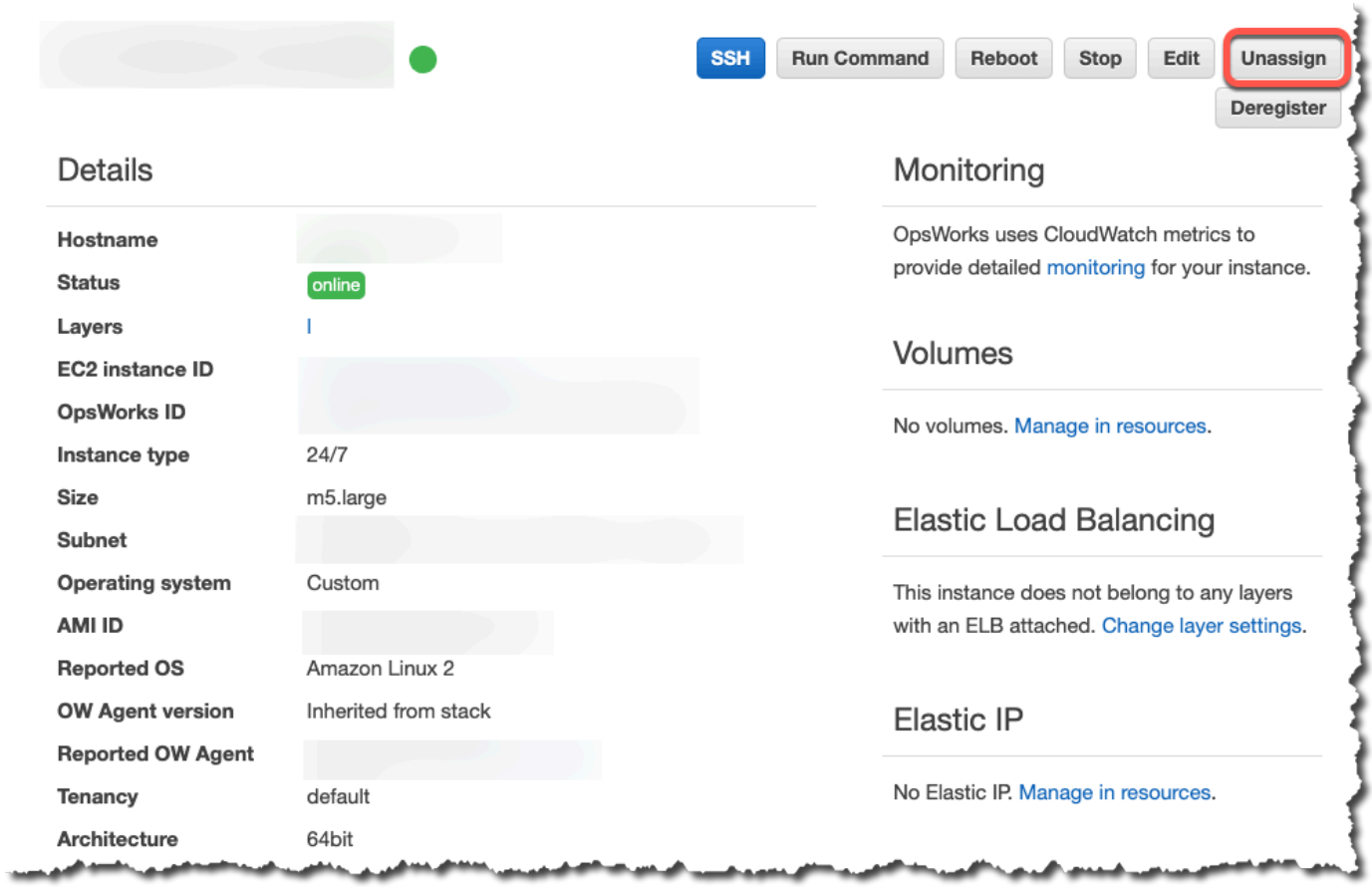
AWS OpsWorks 콘솔 AWS CLI또는 SDK 작업을 사용하여 계층에서 등록된 인스턴스를 할당 취소할 수 있습니다.

인스턴스 할당을 해제하면 AWS OpsWorks Stacks는 인스턴스에서 계층의 종료 레시피를 실행합니다. 이들 레시피는 서비스 종료 같은 작업을 수행하지만 인스턴스를 중지하지는 않습니다. 인스턴스가 여러 계층에 할당된 경우, 할당 해제는 모든 계층에 적용됩니다. 즉, 인스턴스를 계층의 하위 집합에서 할당 해제할 수는 없습니다. 다만 인스턴스는 여전히 스택에 등록되어 있으며, 원한다면 다른 계층에 할당할 수 있습니다.

콘솔을 사용하여 등록된 인스턴스의 할당을 취소하려면

1. 탐색 창에서 인스턴스를 선택합니다.
2. 할당 취소하려는 인스턴스를 선택합니다.

3. 인스턴스의 세부 정보 페이지에서 할당 해제를 선택합니다.



를 사용하여 등록된 인스턴스를 할당 해제하려면 AWS CLI

aws opsworks unassign-instance명령을 실행하여 해당 인스턴스를 사용하는 모든 계층에서 등록된 인스턴스 할당을 취소합니다.

```
aws opsworks unassign-instance --region region --instance-id instance-id
```

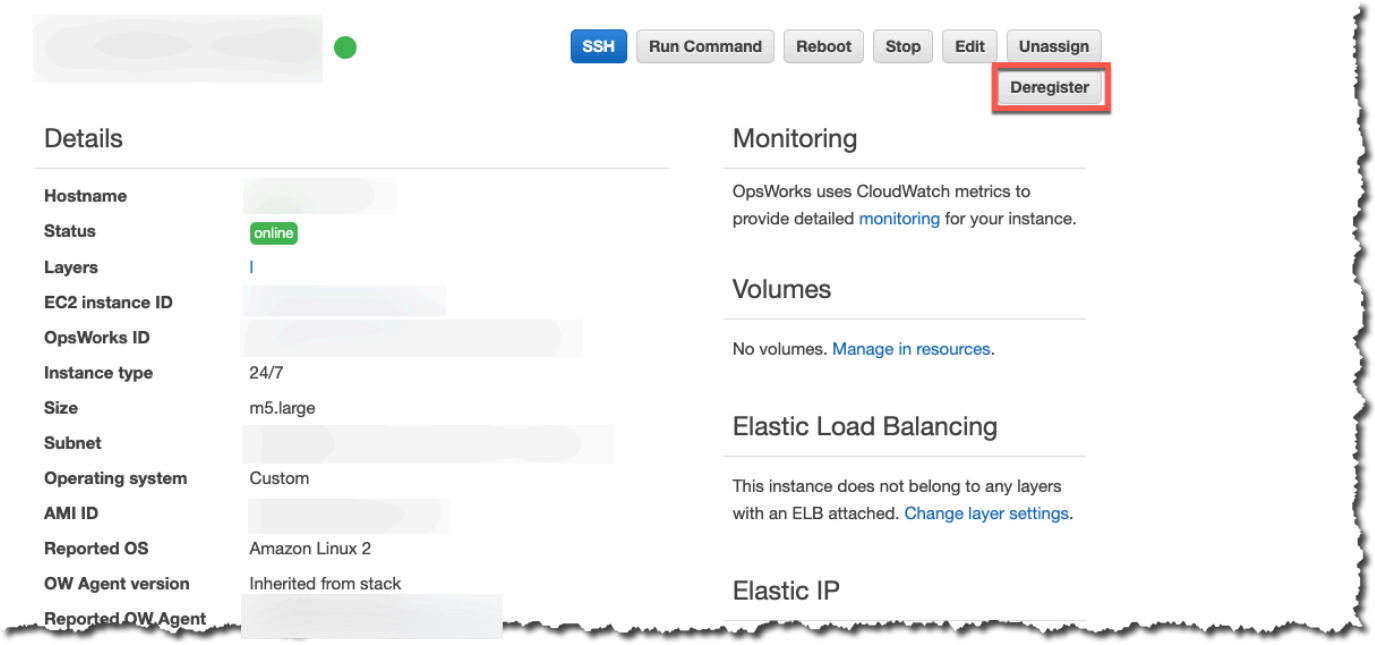
등록된 인스턴스 등록 해제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks 콘솔 AWS CLI또는 SDK 작업을 사용하여 인스턴스 등록을 취소할 수 있습니다.

콘솔을 사용하여 인스턴스 등록을 취소하려면

1. 탐색 창에서 Instances(인스턴스)를 선택합니다.
2. 등록 취소하려는 인스턴스를 선택합니다.
3. 인스턴스의 세부 정보 페이지에서 등록 취소를 선택합니다.



를 사용하여 인스턴스 등록을 취소하려면 AWS CLI

aws opsworks deregister-instance 명령을 실행하여 스택에서 인스턴스 등록을 취소합니다.

```
aws opsworks deregister-instance --region region --instance-id instance-id
```

인스턴스 등록을 취소할 때 AWS OpsWorks Stacks는 다음을 수행합니다.

- 스택에서 인스턴스를 제거합니다.
- 할당된 모든 계층에서 인스턴스 할당을 해제합니다.
- 에이전트를 종료하고 제거합니다.
- 연결된 리소스(탄력적 IP 주소 및 Amazon EBS 볼륨)를 등록 해제합니다.

이 절차에는 등록 전에 인스턴스에 연결된 리소스와 AWS OpsWorks Stacks를 사용하여 스택의 일부인 인스턴스에 연결한 리소스가 포함됩니다. 등록 해제 후에 리소스는 더 이상 스택의 리소스의 일부가 아니지만 인스턴스에 계속 연결되어 있습니다.

- 온프레미스 인스턴스의 경우, 요금 청구가 중단됩니다.
- OpsWorks가 인스턴스에 추가한 모든 태그를 제거합니다.

인스턴스는 실행 중 상태로 유지되지만 직접 제어할 수 있으며 더 이상 AWS OpsWorks Stacks에서 관리하지 않습니다.

> **Note**
>
> 컴퓨터 또는 인스턴스의 등록 및 등록 해제는 Linux 스택 내에서만 완전히 지원됩니다. Windows 스택의 경우 인스턴스 등록을 취소할 수 있지만 인스턴스에서 OpsWorks 에이전트를 제거하지는 않습니다. 등록 해제는 변경된 모든 파일을 제거하지 않으며, 일부 파일의 백업 사본으로 완전히 되돌리지 않습니다. 이 목록은 Chef 11.10 및 Chef 12 스택에 적용됩니다. 두 버전의 차이점은 아래 나와 있습니다.
>
> - `/etc/hosts`는 `/var/lib/aws/opsworks/local-mode-cache/backup/etc/`으로 백업되며 복원되지 않습니다.
> - passwd, group, shadow 파일 등의 `aws` 및 `opsworks` 항목은 그대로 남습니다.
> - `/etc/sudoers` 에는 AWS OpsWorks Stacks 디렉터리에 대한 참조가 포함되어 있습니다.
> - 다음 파일들은 남겨두어도 안전하지만 장기적으로는 `/var/lib/aws/opsworks`는 삭제하는 것이 좋습니다.
>   - `/var/log/aws/opsworks`는 Chef 11.10 스택의 인스턴스에 남아 있습니다.
>   - `/var/lib/aws/opsworks`는 Chef 11.10 및 Chef 12 스택 둘 다에 남아 있습니다.
>   - `/var/chef`는 Chef 12 스택의 인스턴스에 남아 있습니다.
> - 남는 그 밖의 파일은 다음과 같습니다.
>   - `/etc/logrotate.d/opsworks-agent`
>   - `/etc/cron.d/opsworks-agent-updater`
>   - `/etc/ld.so.conf.d/opsworks-user-space.conf`
>   - `/etc/motd.opsworks-static`
>   - `/etc/aws/opsworks`
>   - `/etc/sudoers.d/opsworks`

- `/etc/sudoers.d/opsworks-agent`

등록된 인스턴스 수명 주기

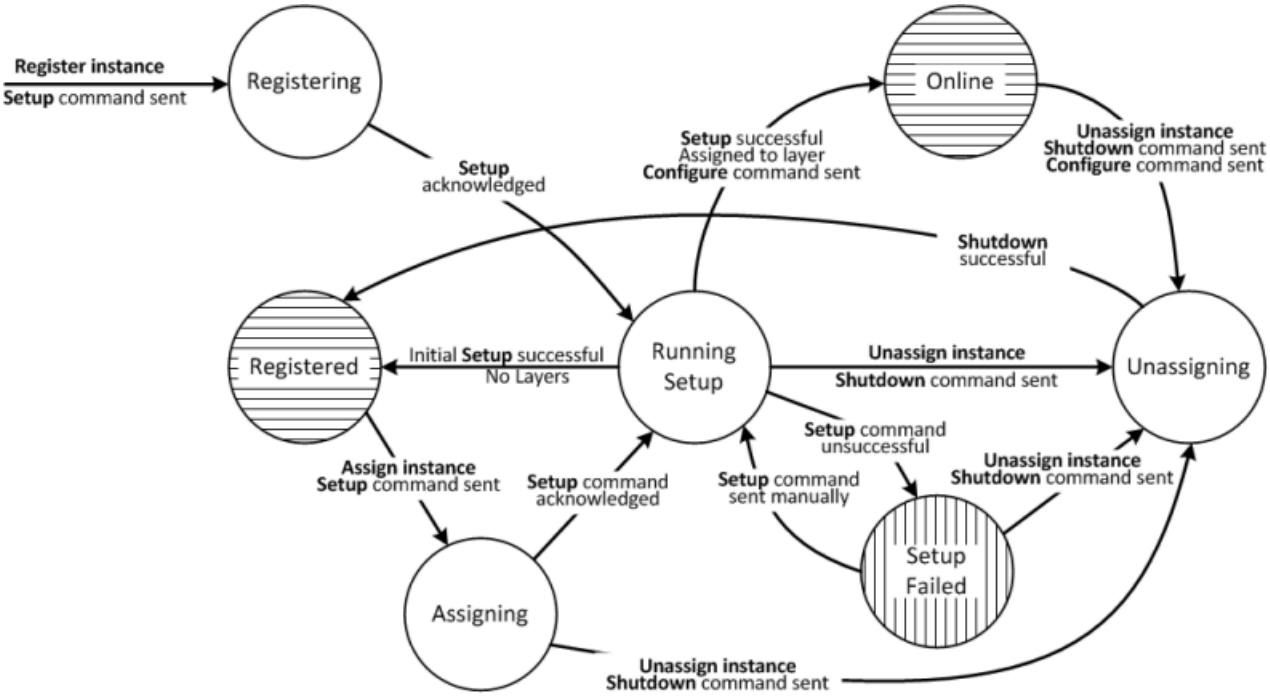
Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 기능은 Linux 스택에서만 지원됩니다.

등록된 인스턴스 수명 주기는 에이전트가 설치되고 실행된 후 시작됩니다. 이때 AWS OpsWorks Stacks에 인스턴스를 스택에 등록하도록 지시합니다. 다음 상태 다이어그램은 주요 수명 주기 요소를 요약합니다.



각 상태는 인스턴스 상태에 대응합니다. 엣지는 다음 AWS OpsWorks Stacks 명령 중 하나를 나타냅니다. 세부 정보는 다음 섹션에 나와 있습니다.

- 설정 – 이 명령은 설정 수명 주기 이벤트에 해당하며 인스턴스의 설정 레시피를 실행합니다.
- 구성 – 이 명령은 Configure 수명 주기 이벤트에 해당합니다.

  AWS OpsWorks Stacks는 인스턴스가 온라인 상태가 되거나 온라인 상태에서 나갈 때 스택의 모든 인스턴스에서이 이벤트를 트리거합니다. 인스턴스들은 Configure 레시피를 실행하여 새 인스턴스 수용에 필요한 변경을 수행합니다.
- 종료 – 이 명령은 인스턴스의 Shutdown 레시피를 실행하는 Shutdown 수명 주기 이벤트에 해당합니다.

  이들 레시피는 서비스 종료 같은 작업을 수행하지만 인스턴스를 중지하지는 않습니다.
- 등록 취소 – 이 명령은 인스턴스를 등록 해제하며 수명 주기 이벤트에 해당하지 않습니다.

> **Note**
>
> 간단히 하기 위해 다이어그램에는 Deregistering 상태와 Deleted 상태가 표시되어 있지 않습니다. 다이어그램의 모든 상태에서 인스턴스 등록을 해제할 수 있으며, 그러면 Deregister 명령이 인스턴스에 전송되어 인스턴스가 Deregistering 상태로 이동합니다.
>
> - 온라인 인스턴스 등록을 취소하면 AWS OpsWorks Stacks는 스택의 나머지 인스턴스에 Configure 명령을 전송하여 인스턴스가 오프라인 상태가 되고 있음을 알립니다.
> - Deregister 명령이 승인된 후 인스턴스는 여전히 실행되지만 Deleted 상태가 되고 더 이상 스택의 일부가 아닙니다. 인스턴스를 스택에 다시 통합하려면 다시 등록해야 합니다.

주제

- 등록
- Running 설정
- 등록
- 할당
- 온라인
- 설정 실패
- 할당 해제

- 초기 설정 구성 변경

등록

에이전트가 등록 요청을 전송한 후 AWS OpsWorks Stacks는 인스턴스에 설정 명령을 전송하여 인스턴스 수명 주기를 시작하고 등록 중 상태로 전환합니다. 인스턴스는 설정 명령을 승인한 후 Running 설정 상태로 이동합니다.

Running 설정

Running 설정 상태는 인스턴스의 설정 레시피를 실행합니다. 설정은 이전 상태에 따라 다르게 작동합니다.

> **Note**
>
> 인스턴스가 설정 실행 상태인 동안 할당을 해제하면 AWS OpsWorks Stacks는 인스턴스의 종료 레시피를 실행하지만 인스턴스를 중지하지 않는 종료 명령을 보냅니다. 인스턴스는 할당 해제 상태로 이동합니다.

주제

- 등록
- 할당
- 설정 실패

등록

등록 프로세스 중에 설정은 스택에 등록된 인스턴스를 나타내는 AWS OpsWorks Stacks 인스턴스를 생성하고 인스턴스에서 코어 설정 레시피 세트를 실행합니다.

초기 설정이 수행하는 중요한 변경 한 가지는 인스턴스의 호스트 파일을 덮어쓰는 것입니다. 인스턴스를 등록하면 사용자 관리를 AWS OpsWorks Stacks에게 인계한 것입니다. SSH 로그인 권한을 제어하기 위해 자체 호스트 파일이 있어야 합니다. 또한 초기 설정은 여러 파일을 생성 또는 수정하며, Ubuntu 시스템에서는 패키지 소스를 수정하고 패키지 세트를 설치합니다. 세부 정보는 초기 설정 구성 변경을 참조하세요.

등록 중에 프로세스는 사전 조건으로 생성하는 IAM 사용자에게 연결된 권한의 일부인 IAM `AttachUserPolicy`을(를) 호출합니다. `AttachUserPolicy`가 존재하지 않으면(대체로 오래된 릴리스의 AWS CLI를 실행하고 있기 때문) 프로세스는 `PutUserPolicy` 호출로 돌아갑니다.

> **Note**
>
> 일관성을 위해 AWS OpsWorks Stacks는 모든 핵심 설정 레시피를 실행합니다. 하지만 일부는 인스턴스가 적어도 하나의 계층에 할당된 경우에만 작업의 전부 또는 일부를 수행하므로 반드시 초기 설정에 영향을 미치지는 않습니다.

- 설정이 성공하면 인스턴스는 등록 상태로 이동합니다.
- 설정이 성공하지 못하면 인스턴스는 설정 실패 상태로 이동합니다.

할당

인스턴스에 계층이 하나 이상 할당되어 있습니다. AWS OpsWorks Stacks는 계층의 설정 이벤트에 할당한 사용자 지정 레시피를 포함하여 각 계층의 설정 레시피를 실행합니다.

- 설정이 성공하면 인스턴스는 Online 상태로 이동하고, AWS OpsWorks Stacks는 스택의 모든 인스턴스에서 Configure 수명 주기 이벤트를 트리거하여 새 인스턴스를 알립니다.
- 설정이 성공하지 못하면 인스턴스는 설정 Failed 상태로 이동합니다.

> **Note**
>
> 이 설정 프로세스는 핵심 레시피를 재차 실행합니다. 다만 Chef 레시피는 idempotent 방식이므로 이미 수행된 작업은 반복하지 않습니다.

설정 실패

할당 상태인 인스턴스의 설정 프로세스가 실패하면 설정 스택 명령을 사용하여 수동으로 인스턴스의 설정 레시피를 재실행하여 다시 시도할 수 있습니다.

- 설정이 성공하면 할당된 인스턴스는 온라인 상태로 이동하고, AWS OpsWorks Stacks는 스택의 모든 인스턴스에서 Configure 수명 주기 이벤트를 트리거하여 새 인스턴스를 알립니다.
- 설정 시도가 성공하지 못하면 인스턴스는 다시 설정 Failed 상태로 이동합니다.

등록

등록된 상태의 인스턴스는 스택의 일부이며 AWS OpsWorks Stacks에서 관리하지만 계층에는 할당되지 않습니다. 이러한 인스턴스는 이 상태로 무기한 남아 있을 수 있습니다.

인스턴스를 하나 이상의 계층에 할당하면 AWS OpsWorks Stacks는 인스턴스에 설정 명령을 전송하고 할당 상태로 이동합니다.

할당

인스턴스는 설정 명령을 승인한 후 Running 설정 상태로 이동합니다.

인스턴스가 할당 중 상태일 때 인스턴스를 할당 해제하면 AWS OpsWorks Stacks는 설정 프로세스를 종료하고 종료 명령을 보냅니다. 인스턴스는 할당 해제 상태로 이동합니다.

온라인

이제 인스턴스는 하나 이상의 계층에 속하며, 일반적인 AWS OpsWorks Stacks 인스턴스처럼 취급됩니다. 이 인스턴스는 이 상태로 무기한 남아 있을 수 있습니다.

인스턴스가 온라인 상태인 동안 인스턴스를 할당 해제하면 AWS OpsWorks Stacks는 인스턴스에 종료 명령을 보내고 나머지 스택 인스턴스에 구성 명령을 보냅니다. 인스턴스는 할당 해제 상태로 이동합니다.

설정 실패

설정 명령이 실패했습니다.

- 설정 스택 명령을 실행하여 다시 시도할 수 있습니다.

  인스턴스는 Running 설정 상태로 되돌아갑니다.
- 인스턴스를 할당 해제하면 AWS OpsWorks Stacks는 인스턴스에 종료 명령을 보냅니다.

  인스턴스는 할당 해제 상태로 이동합니다.

할당 해제

Shutdown 명령이 완료된 후에 인스턴스는 더 이상 어떤 계층에도 할당되지 않으며 등록 상태로 돌아갑니다.

> **Note**
>
> 인스턴스가 여러 계층에 할당된 경우, 할당 해제는 모든 계층에 적용됩니다. 즉, 할당된 계층의 하위 집합을 할당 해제할 수 없습니다. 다른 할당된 계층의 집합을 원하는 경우, 인스턴스를 할당 해제한 다음 원하는 계층을 다시 할당하세요.

초기 설정 구성 변경

초기 구성은 등록된 모든 인스턴스에서 다음 파일 및 디렉터리를 생성하거나 수정합니다.

생성된 파일

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/aws/
/etc/init.d/opsworks-agent
/etc/motd
/etc/motd.opsworks-static
/etc/sudoers.d/opsworks
/etc/sudoers.d/opsworks-agent
/etc/sysctl.d/70-opsworks-defaults.conf
/opt/aws/opsworks/
/usr/sbin/opsworks-agent-cli
/var/lib/aws/
/var/log/aws/
/vol/
```

수정된 파일

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/crontab
/etc/default/monit
/etc/group
/etc/gshadow
/etc/monit/monitrc
/etc/passwd
/etc/security/limits.conf (removing limits only for EC2 micro instances)
/etc/shadow
/etc/sudoers
```

초기 설정은 Amazon EC2 마이크로 인스턴스에서 스왑 파일도 생성합니다.

초기 설정은 Ubuntu 시스템을 다음과 같이 변경합니다.

패키지 소스

초기 설정은 패키지 소스를 다음으로 변경합니다.

- deb http://archive.ubuntu.com/ubuntu/ ${code_name} main universe

  끝: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe

  끝: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security main universe

  끝: deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse

  끝: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse

  끝: deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse
- deb http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse

  끝: deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse
- deb http://security.ubuntu.com/ubuntu ${code_name}-security multiverse

  끝: deb-src http://security.ubuntu.com/ubuntu ${code_name}-security multiverse

패키지

초기 설정은 landscape를 제거하고 다음 패키지를 설치합니다.

autofs	libicu-dev	libopenssl-ruby

libssl-dev	libxml2-dev	libxslt-dev
libyaml-dev	monit	ntpd
procps	ruby	ruby-dev
rubygems	screen	sqlite
vim	xfs	

인스턴스 구성 편집

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

등록된 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 포함하여 인스턴스 구성을 편집할 수 있습니다. 단, 다음의 제약이 있습니다.

- 인스턴스가 중지 상태여야 합니다.

  온라인 인스턴스의 속성을 수정할 수는 없지만 인스턴스의 계층을 편집하여 일부 구성을 변경할 수는 있습니다. 자세한 내용은 OpsWorks 계층의 구성 편집 단원을 참조하십시오.
- [가용 영역] 및 [조정 유형]과 같은 일부 설정은 인스턴스를 생성할 때 결정되며 나중에 수정할 수 없습니다.
- Amazon Elastic Block Store 지원 인스턴스에서는 수정할 수 없고 인스턴스 스토어 지원 인스턴스에서만 수정할 수 있는 설정도 일부 있습니다.

  예를 들어 인스턴스 스토어 지원 인스턴스의 운영 체제를 변경할 수 있습니다. Amazon EBS 기반 인스턴스는 인스턴스를 생성할 때 지정한 운영 체제를 사용해야 합니다. 인스턴스 스토리지에 대한 자세한 내용은 스토리지를 참조하세요.
- 기본적으로 인스턴스는 스택의 에이전트 버전 설정을 상속합니다.

[OpsWorks Agent 버전]을 사용하여 스택의 에이전트 버전 설정을 재정의하고 인스턴스에 특정 에이전트 버전을 지정할 수 있습니다. 인스턴스의 에이전트 버전을 지정하는 경우 AWS OpsWorks 스택의 에이전트 버전 설정이 자동 업데이트인 경우에도 스택은 새 버전을 사용할 수 있을 때 에이전트를 자동으로 업데이트하지 않습니다. 인스턴스 구성을 편집하여 인스턴스의 에이전트 버전을 수동으로 업데이트해야 합니다. 그러면 AWS OpsWorks Stacks가 인스턴스에 지정된 에이전트 버전을 설치합니다.

> **Note**
>
> 등록된 온프레미스 인스턴스의 구성은 편집할 수 없습니다.

인스턴스의 구성을 편집하려면

1. 인스턴스가 아직 중지되지 않은 경우 해당 인스턴스를 중지합니다.
2. [인스턴스] 페이지에서 인스턴스 이름을 클릭하여 세부 정보 페이지를 표시합니다.
3. [편집]을 클릭하여 편집 페이지를 표시합니다.
4. 인스턴스의 구성을 적절하게 편집합니다.

[호스트 이름], [크기], [SSH 키] 및 [운영 체제] 설정에 대한 설명은 계층에 인스턴스 추가 단원을 참조하세요. [계층] 설정을 사용하여 계층을 추가 또는 제거할 수 있습니다. 인스턴스의 현재 계층은 계층 목록 다음에 표시됩니다.

- 다른 계층을 추가하려면 목록에서 계층을 선택합니다.
- 계층 중 하나에서 인스턴스를 제거하려면 해당 계층의 옆에서 [x]를 클릭합니다.

  인스턴스는 적어도 하나 이상의 계층의 멤버여야 하며, 따라서 마지막 계층은 제거할 수 없습니다.

인스턴스를 다시 시작하면 AWS OpsWorks Stacks는 업데이트된 구성으로 새 Amazon EC2 인스턴스를 시작합니다.

AWS OpsWorks 스택 인스턴스 삭제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks를 사용하여 등록된 Amazon EC2 인스턴스를 포함한 인스턴스를 중지할 수 있습니다. 그러면 EC2 인스턴스가 중지되지만 인스턴스가 스택에 유지됩니다. 인스턴스의 작업 열에서 시작을 클릭하여 다시 시작할 수 있습니다. 더 이상 인스턴스가 필요하지 않아 스택에서 제거하려면 인스턴스를 삭제할 수 있습니다. 그러면 인스턴스가 스택에서 제거되고 연결된 Amazon EC2 인스턴스가 종료됩니다. 인스턴스를 삭제하면 연결된 로그 또는 데이터, 그리고 인스턴스의 Amazon Elastic Block Store(EBS) 볼륨도 모두 삭제됩니다.

> **Important**
>
> 이 주제는 AWS OpsWorks Stacks에서 관리하는 Amazon EC2 인스턴스에만 적용됩니다. Amazon EC2 콘솔 또는 API에서 관리되는 인스턴스를 삭제하는 방법에 대한 자세한 내용은 인스턴스 종료를 참조하세요.

> **Note**
>
> AWS OpsWorks Stacks를 사용하여 등록된 온프레미스 인스턴스를 삭제할 수 없습니다.

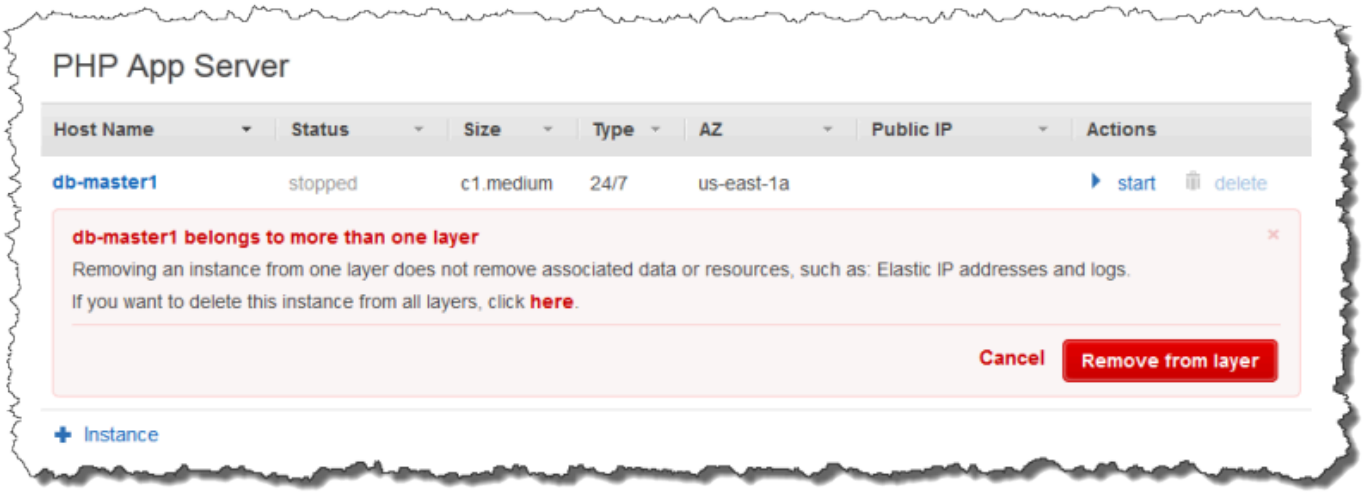
인스턴스가 여러 계층에 속하는 경우 스택에서 인스턴스를 삭제하거나 특정 계층을 제거할 수 있습니다. 또한 인스턴스 구성 편집 섹션에 설명된 대로 인스턴스 구성을 편집하여 인스턴스에서 계층을 제거할 수도 있습니다.

> **Important**
>
> AWS OpsWorks Stacks 콘솔 또는 API를 사용해야만 AWS OpsWorks Stacks 인스턴스를 삭제해야 합니다. 특히 Amazon EC2 작업은 AWS OpsWorks Stacks와 자동으로 동기화되지 않

으므로 Amazon EC2 콘솔 또는 API를 사용하여 AWS OpsWorks Stacks 인스턴스를 삭제해서는 안 됩니다. 예를 들어 자동 복구가 활성화된 상태에서 Amazon EC2 콘솔을 사용하여 인스턴스를 종료하는 경우 AWS OpsWorks Stacks는 종료된 인스턴스를 실패한 인스턴스와 동일하게 취급하고 또 하나의 Amazon EC2 인스턴스를 시작하여 해당 인스턴스를 대체합니다. 자세한 내용은 자동 복구 사용 단원을 참조하십시오.

인스턴스 삭제

1. [인스턴스] 페이지의 해당 계층에서 인스턴스를 찾습니다. 인스턴스가 실행 중인 경우 작업 열에서 중지를 클릭합니다.
2. 상태가 [중지됨]로 변경되면 [삭제]를 클릭합니다. 인스턴스가 둘 이상의 계층의 멤버인 경우 계층 AWS OpsWorks 스택에 다음 섹션이 표시됩니다.



   - 선택한 계층에서만 인스턴스를 제거하려면 [계층에서 제거]를 클릭합니다.

     해당 인스턴스는 다른 계층에는 계속 속해 있으며 다시 시작할 수 있습니다.
   - 모든 계층에서 인스턴스를 삭제하여 스택에서 제거하려면 [여기]를 클릭합니다.
3. 스택에서 인스턴스를 완전히 제거하도록 선택하거나 인스턴스가 단 한 계층의 멤버인 경우 AWS OpsWorks Stacks는 삭제를 확인하라는 메시지를 표시합니다.

   [삭제]를 선택하여 확인합니다. 이 작업은 스택에서 인스턴스를 삭제하는 것 외에도 인스턴스에 연결된 모든 로그 또는 데이터 및 루트 볼륨을 삭제합니다. 모든 인스턴스 볼륨을 제거하려면 삭제를 선택하기 전에 인스턴스의 EBS 볼륨 삭제(스냅샷은 삭제되지 않음)를 선택합니다.

SSH를 사용하여 Linux 인스턴스에 로그인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

내장 MindTerm 클라이언트 또는 타사 클라이언트(예: PuTTY)를 사용하여 SSH로 Linux 인스턴스에 로그인할 수 있습니다. 일반적으로 SSH는 인증을 위해 RSA 키 페어를 사용합니다. 인스턴스에 퍼블릭 키를 설치하고 SSH 클라이언트에 해당 프라이빗 키를 제공합니다. AWS OpsWorks Stacks는 다음과 같이 스택의 인스턴스에 퍼블릭 키 설치를 처리합니다.

- Amazon Elastic Compute Cloud(Amazon EC2) 키 페어 - 스택의 리전에 하나 이상의 Amazon EC2 키 페어가 있을 경우 스택의 기본 SSH 키 페어를 지정할 수 있습니다.

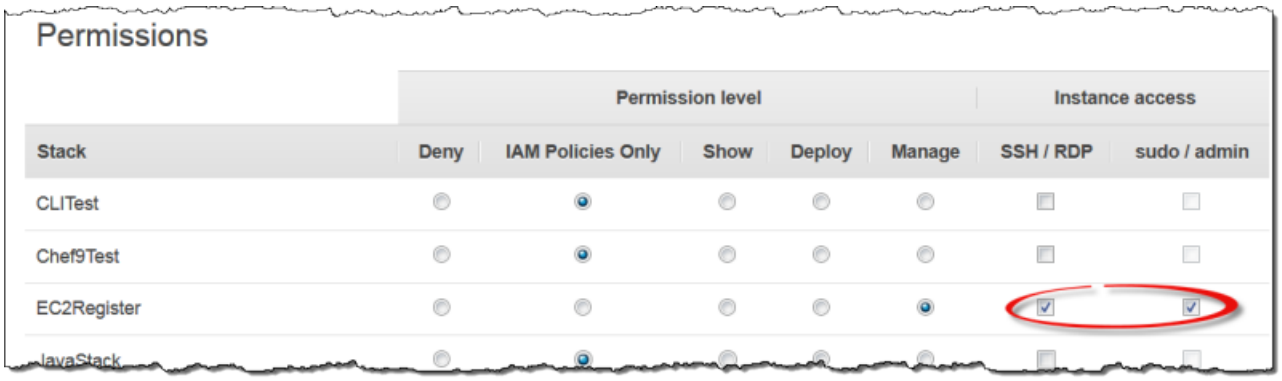
  선택적으로 인스턴스를 생성할 때 기본 키 페어를 오버라이드하고 다른 페어를 지정할 수도 있습니다. 어느 경우든 AWS OpsWorks Stacks는 지정된 키 페어의 퍼블릭 키를 인스턴스에 설치합니다. Amazon EC2 키 페어를 생성하는 방법에 대한 자세한 내용은 Amazon EC2 키 페어를 참조하세요.
- 개인 키 페어 — 각 사용자는 AWS OpsWorks Stacks에 개인 키 페어를 등록할 수 있습니다.

  사용자 또는 관리자는 퍼블릭 키를 AWS OpsWorks Stacks에 등록하고 사용자는 프라이빗 키를 로컬에 저장합니다. 스택에 대한 권한을 설정할 때 관리자가 스택의 인스턴스에 대한 SSH 액세스 권한을 가질 사용자를 지정합니다. AWS OpsWorks Stacks는 권한 있는 각 사용자에 대해 스택의 인스턴스에 시스템 사용자를 자동으로 생성하고 사용자의 개인 퍼블릭 키를 설치합니다.

사용자가 MindTerm SSH 클라이언트를 사용하거나 개인 키 페어를 사용하여 스택의 인스턴스에 로그인하려면 SSH 권한이 부여되어야 합니다.

사용자에 대해 SSH를 승인하려면

1. AWS OpsWorks 스택 탐색 창에서 권한을 클릭합니다.
2. 원하는 IAM 사용자에 대한 SSH/RDP를 선택하여 필요한 권한을 부여합니다. 사용자가 권한을 높이는 데 **sudo**을(를) 사용할 수 있도록 허용하려면(예: 에이전트 CLI 명령 실행) sudo/admin도 선택하세요.



AWS OpsWorks Stacks를 사용하여 SSH 액세스를 관리하는 방법에 대한 자세한 내용은 섹션을 참조하세요SSH 액세스 관리.

주제

- 내장 MindTerm SSH 클라이언트를 사용
- 타사 SSH 클라이언트 사용

내장 MindTerm SSH 클라이언트를 사용

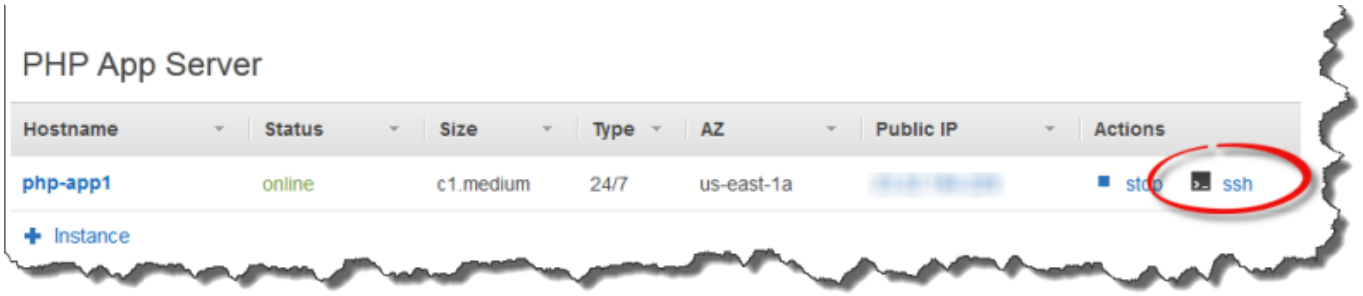
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Linux 인스턴스에 로그인하기 위한 가장 간단한 방법은 내장 MindTerm SSH 클라이언트를 사용하는 것입니다. 각 온라인 인스턴스에는 사용자가 MindTerm 클라이언트를 시작하는 데 사용할 수 있는 ssh 작업이 포함되어 있습니다.

> **Note**
>
> MindTerm 클라이언트를 사용하려면 브라우저에서 Java를 사용하도록 설정해야 합니다.

MindTerm 클라이언트를 사용하여 로그인하려면

1. 아직 권한을 부여하지 않은 경우, 이전 단원에서 설명한 대로 인스턴스에 연결할 IAM 사용자에 대해 SSH 액세스 권한을 부여합니다.
2. 사용자로 로그인합니다.
3. 인스턴스 페이지의 작업 열에서 해당 인스턴스의 ssh를 선택합니다.



4. 프라이빗 키의 경우 인스턴스에 설치한 퍼블릭 키에 따라 사용자의 개인 프라이빗 키 또는 Amazon EC2 프라이빗 키에 대한 경로를 입력합니다.
5. [Mindterm 시작]을 선택하고 터미널 창에서 인스턴스에 대한 명령을 실행합니다.

타사 SSH 클라이언트 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

PuTTY와 같은 타사 SSH 클라이언트를 사용하여 Linux 인스턴스에 연결할 수도 있습니다.

타사 SSH 클라이언트를 사용하려면

1. 앞서 설명한 대로 AWS OpsWorks Stacks가 인스턴스에 Amazon EC2 퍼블릭 키 또는 IAM 사용자의 개인 퍼블릭 키를 설치했는지 확인합니다.
2. 세부 정보 페이지에서 인스턴스의 퍼블릭 DNS 이름 또는 퍼블릭 IP 주소를 확인합니다.
3. 클라이언트에 인스턴스의 호스트 이름을 제공합니다. 이 이름은 다음과 같이 운영 체제에 따라 다릅니다.

- Amazon Linux 및 Red Hat Enterprise Linux(RHEL)- `ec2-user@`*`DNSName/Address`*`.`
- Ubuntu – `ubuntu@`*`DNSName/Address`*.

*`DNSName/Address`*를 이전 단계에서 확인한 퍼블릭 DNS 이름 또는 IP 주소로 바꿉니다.

4. 클라이언트에 설치된 퍼블릭 키에 해당하는 프라이빗 키를 제공합니다. 인스턴스에 설치한 퍼블릭 키에 따라 Amazon EC2 프라이빗 키 또는 IAM 사용자의 개인 프라이빗 키를 사용합니다.

RDP를 사용하여 Windows 인스턴스에 로그인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

다음과 같이 Windows 원격 데스크톱 프로토콜(RDP)을 사용하여 온라인 Windows 인스턴스에 로그인할 수 있습니다.

- 인스턴스에는 RDP 액세스를 허용하는 인바운드 규칙이 포함된 보안 그룹이 있어야 합니다.

  보안 그룹 작업에 대한 자세한 정보는 보안 그룹 사용 단원을 참조하세요.
- 일반 사용자 - AWS OpsWorks Stacks는 권한이 있는 일반 사용자에게 제한된 기간 동안 유효한 RDP 암호를 제공하며,이 암호는 30분에서 12시간 사이일 수 있습니다.

  권한이 부여되는 것 외에도 사용자에게는 최소한 권한 수준 표시 또는 연결된 AWS Identity and Access Management (IAM) 정책이 `opsworks:GrantAccess` 작업을 허용해야 합니다.
- 관리자 — 관리자 암호를 사용하여 로그인할 수 있는 시간은 무제한입니다.

  나중에 설명하듯이 인스턴스에 Amazon Elastic Compute Cloud(Amazon EC2) 키 페어를 지정한 경우 이를 사용하여 관리자 암호를 검색할 수 있습니다.

> **Note**
>
> 이 주제에서는 Windows 워크스테이션에서 Windows 원격 데스크톱 연결 클라이언트를 사용하여 로그인하는 방법을 설명합니다. 사용 가능한 Linux 또는 OS X용 RDP 클라이언트 중 하나를 사용할 수도 있지만, 절차는 약간 다를 수 있습니다. Microsoft Windows Server 2012 R2와 호환되는 RDP 클라이언트에 대한 자세한 정보는 Microsoft Remote Desktop Clients를 참조하세요.

주제

- RDP 액세스를 허용하는 보안 그룹 제공
- 일반 사용자로 로그인
- 관리자로 로그인

RDP 액세스를 허용하는 보안 그룹 제공

RDP를 사용하여 Windows 인스턴스에 로그인할 수 있으려면 인스턴스의 보안 그룹 인바운드 규칙이 RDP 연결을 허용해야 합니다. 리전에서 첫 번째 스택을 생성할 때 AWS OpsWorks Stacks가 보안 그룹 세트를 생성합니다. 여기에는과 같은 이름의 인스턴스가 포함되며`AWS-OpsWorks-RDP-Server`, AWS OpsWorks Stacks는 모든 Windows 인스턴스에 연결하여 RDP 액세스를 허용합니다. 하지만 이 보안 그룹은 기본적으로 아무 규칙도 포함하고 있지 않으므로, 인스턴스에 RDP 액세스를 허용하는 인바운드 규칙을 추가해야 합니다.

RDP 액세스를 허용하려면

1. Amazon EC2 콘솔을 열고, 콘솔을 스택의 리전으로 설정한 다음, 탐색 창에서 보안 그룹을 선택합니다.
2. [AWS-OpsWorks-RDP-Server], [인바운드] 탭, [편집]을 차례대로 선택합니다.
3. [역할 추가]를 선택하고 다음 설정을 지정합니다.

   - 유형 – RDP
   - 소스 – 허용 가능한 소스 IP 주소.

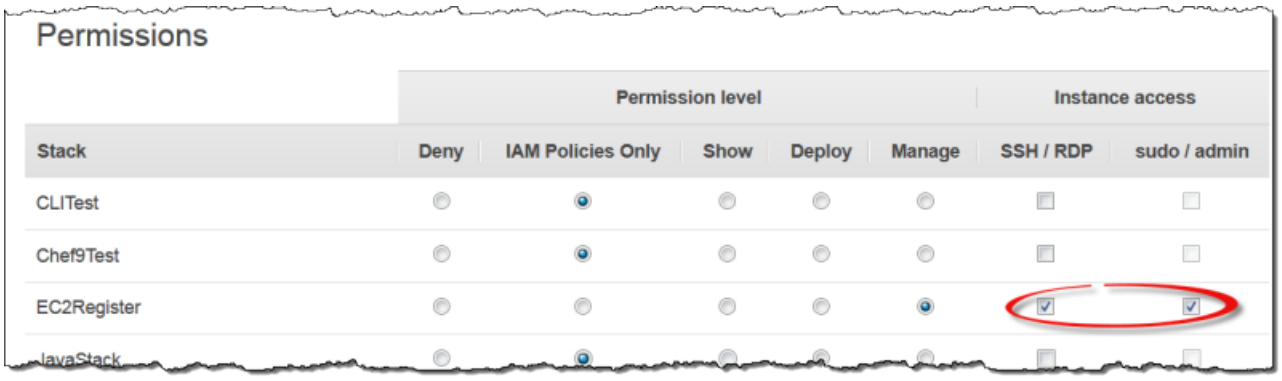
     일반적으로 IP 주소 또는 지정된 IP 주소 범위(대개 회사 IP 주소)에서 들어오는 인바운드 RDP 요청을 허용합니다.

일반 사용자로 로그인

권한 있는 사용자는 AWS OpsWorks Stacks가 제공하는 임시 암호를 사용하여 인스턴스에 로그인할 수 있습니다.

사용자에 대해 RDP를 승인하려면,

1. AWS OpsWorks 스택 탐색 창에서 권한을 클릭합니다.
2. 원하는 사용자에게 필요한 권한을 부여하려면 SSH/RDP 확인란을 선택합니다. 이 사용자에게 관리자 권한을 부여하려면 sudo/admin도 선택해야 합니다.



권한 있는 사용자는 다음과 같이 스택의 모든 온라인 인스턴스에 로그인할 수 있습니다.

일반 IAM 사용자로 로그인하려면,

1. IAM 사용자로 로그인합니다.
2. 인스턴스 페이지의 작업 열에서 해당 인스턴스의 rdp를 선택합니다.
3. 세션 길이(30분~12시간)를 지정하고 [암호 생성]을 선택합니다. 이 암호는 지정된 세션 기간 동안만 유효합니다.
4. [퍼블릭 DNS 이름], [사용자 이름] 및 [암호] 값을 적어 두고 [확인 및 닫기]를 선택합니다.
5. Windows 원격 데스크톱 연결 클라이언트를 열고, [옵션 표시]를 선택한 후 4단계에서 적어 둔 정보를 바탕으로 아래 값을 입력합니다.

   - 컴퓨터 – 인스턴스의 퍼블릭 DNS 이름.

     원하는 경우 퍼블릭 IP 주소를 사용할 수도 있습니다. [인스턴스]를 선택하고 인스턴스의 [퍼블릭 IP] 열에서 주소를 복사합니다.
   - 사용자 이름 – 사용자 이름.

6. 클라이언트에서 자격 증명을 입력하라는 메시지가 나타나면 4단계에서 저장한 암호를 사용합니다.

> **Note**
>
> AWS OpsWorks Stacks는 온라인 인스턴스에 대해서만 사용자 암호를 생성합니다. 인스턴스를 시작한 후 예를 들어 사용자 지정 설정 레시피 중 하나가 실패할 경우 인스턴스는 `setup_failed` 상태가 됩니다. 인스턴스가 AWS OpsWorks Stacks에 관한 한 온라인 상태가 아니더라도 EC2 인스턴스가 실행 중이며, 문제를 해결하기 위해 로그인하는 것이 유용한 경우가 많습니다.이 경우 AWS OpsWorks Stacks는 암호를 생성하지 않지만 인스턴스에 SSH 키 페어를 할당한 경우 EC2 콘솔 또는 CLI를 사용하여 인스턴스의 관리자 암호를 검색하고 관리자로 로그인할 수 있습니다. 자세한 내용은 다음 섹션을 참조하세요.

관리자로 로그인

적절한 암호를 사용하여 관리자로 인스턴스에 로그인할 수 있습니다. 인스턴스에 EC2 키 페어를 할당한 경우 Amazon EC2는 인스턴스가 시작할 때 이 키 페어를 사용하여 자동으로 관리자 암호를 생성 및 암호화합니다. 그러면 EC2 콘솔, API 또는 CLI에서 키 페어의 프라이빗 키를 사용하여 암호를 검색하고 복호화할 수 있습니다.

> **Note**
>
> 개인 SSH 키 페어를 사용하여 관리자 암호를 검색할 수는 없습니다. 반드시 EC2 키 페어를 사용해야 합니다.

다음 절차는 EC2 콘솔을 사용하여 관리자 암호를 검색하고 인스턴스에 로그인하는 방법을 설명합니다. 명령줄 도구를 선호할 경우 AWS CLI `get-password-data` 명령을 사용하여 암호를 검색할 수도 있습니다.

관리자로 로그인하려면

1. 인스턴스에 대해 EC2 키 페어를 지정했는지 확인합니다. 스택을 생성할 때 스택의 모든 인스턴스에 대해 기본 키 페어를 지정하거나 인스턴스를 생성할 때 특정 인스턴스에 대한 키 페어를 지정할 수 있습니다.
2. EC2 콘솔을 열고 콘솔을 스택의 리전으로 설정한 다음 탐색 창에서 인스턴스를 선택합니다.

3. 인스턴스를 선택하고, [연결]을 선택한 다음 [암호 가져오기]를 선택합니다.
4. 워크스테이션에 있는 EC2 키 페어의 프라이빗 키에 대한 경로를 입력하고 [암호 해독]을 선택합니다. 나중에 사용하기 위해 해독된 암호를 복사해 둡니다.
5. Windows 원격 데스크톱 연결 클라이언트를 열고, [옵션 표시]를 선택한 후 다음 정보를 입력합니다.
   - 컴퓨터 – 인스턴스의 퍼블릭 DNS 이름 또는 퍼블릭 IP 주소로, 인스턴스의 세부 정보 페이지에서 확인할 수 있습니다.
   - 사용자 이름 – `Administrator`.
6. 클라이언트에서 자격 증명을 입력하라는 메시지가 나타나면 4단계에서 해독한 암호를 입력합니다.

앱

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 앱은 애플리케이션 서버에서 실행하려는 코드를 나타냅니다. 코드 자체는 Amazon S3 아카이브와 같은 리포지토리에 상주하며, 앱은 이 코드를 적절한 애플리케이션 서버 인스턴스에 배포하는 데 필요한 정보를 포함합니다.

애플리케이션을 배포할 때 AWS OpsWorks Stacks는 각 계층의 Deploy 레시피를 실행하는 Deploy 이벤트를 트리거합니다. AWS OpsWorks Stacks는 앱의 리포지토리 및 데이터베이스 연결 데이터와 같이 앱을 배포하는 데 필요한 모든 정보가 포함된 스택 구성 및 배포 속성도 설치합니다.

스택 구성 및 배포 속성에서 앱의 배포 데이터를 검색하고 배포 작업을 처리하는 사용자 지정 레시피를 구현해야 합니다.

주제

- 앱 추가
- 앱 배포

- 앱 편집
- 데이터베이스 서버에 애플리케이션 연결
- 환경 변수 사용
- 애플리케이션으로 데이터 전달
- Git 리포지토리 SSH 키 사용
- 사용자 지정 도메인 사용
- SSL 사용

앱 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

애플리케이션을 애플리케이션 서버에 배포하는 첫 단계는 스택에 앱을 추가하는 것입니다. 앱은 애플리케이션을 나타내며, 애플리케이션의 이름과 유형, 애플리케이션을 서버 인스턴스에 배포하는 데 필요한 정보(예: 리포지토리 URL) 등 다양한 메타데이터를 포함합니다. 앱을 스택에 추가하려면 Manage 권한이 있어야 합니다. 자세한 내용은 사용자 권한 관리 단원을 참조하십시오.

> **Note**
>
> 이 섹션의 절차는 Chef 12 이상의 스택에 적용됩니다. Chef 11 스택에서 계층에 앱을 추가하는 방법에 대한 자세한 정보는 2.4단계: 앱 생성 및 배포 - Chef 11 단원을 참조하세요.

스택에 앱을 추가하려면

1. 선호하는 리포지토리(Amazon S3 아카이브, Git 리포지토리, 하위 버전 리포지토리 또는 HTTP 아카이브)에 코드를 저장합니다. 자세한 내용은 애플리케이션 소스 단원을 참조하십시오.
2. 탐색 창에서 [앱]을 클릭합니다. [앱] 페이지에서 첫 번째 앱에 대해 [앱 추가]를 클릭합니다. 이후의 앱에는 [+앱]을 클릭합니다.
3. [앱 새로 만들기] 페이지에서 다음 단원의 설명대로 앱을 구성합니다.

앱 구성

[앱 추가] 페이지는 [설정], [애플리케이션 소스], [데이터 소스], [환경 변수], [도메인 추가], [SSL 설정] 섹션으로 구성되어 있습니다.

주제

- 설정
- 애플리케이션 소스
- 데이터 원본
- 환경 변수
- 도메인 및 SSL 설정

설정

명칭

UI에서 앱을 나타내는 데 사용되는 앱 이름입니다. AWS OpsWorks Stacks는이 이름을 사용하여 내부적으로 사용되는 앱의 짧은 이름을 생성하고 스택 구성 및 배포 속성에서 앱을 식별합니다. 앱을 스택에 추가한 후 탐색 창에서 [앱]을 클릭하여 짧은 이름을 볼 수 있으며, 그런 다음 앱의 이름을 클릭하면 세부 정보 페이지를 열 수 있습니다.

[문서 루트]

AWS OpsWorks Stacks는 앱 속성의 `[:document_root]` 속성에 문서 루트 설정을 할당합니다deploy. 기본값은 null입니다. 배포 레시피는 표준 Chef 노드 구문을 사용하여 deploy 속성에서 이 값을 가져올 수 있으며, 지정된 코드를 서버의 적절한 위치에 배포할 수 있습니다. 앱을 배포하는 방법에 대한 자세한 정보는 Deploy 레시피 단원을 참조하세요.

애플리케이션 소스

앱을 배포할 수 있는 리포지토리 유형은 Git, Amazon S3 번들, HTTP 번들 등입니다. 모든 리포지토리 유형은 리포지토리 유형과 리포지토리 URL을 지정해야 합니다. 개별 리포지토리 유형에는 아래 설명처럼 고유의 요구 사항이 있습니다.

> **Note**
>
> AWS OpsWorks Stacks는 표준 리포지토리에서 기본 제공 서버 계층으로 애플리케이션을 자동으로 배포합니다. Windows 스택의 유일한 옵션인 기타 리포지토리 유형을 사용하는 경우

AWS OpsWorks Stacks는 리포지토리 정보를 앱의 `deploy` 속성에 입력하지만 배포 작업을 처리하기 위해 사용자 지정 레시피를 구현해야 합니다.

주제

- HTTP 아카이브
- Amazon S3 아카이브
- Git 리포지토리
- 기타 리포지토리

HTTP 아카이브

공용 액세스가 가능한 HTTP 서버를 리포지토리로 사용하려면

1. 앱 코드 및 관련 파일이 포함된 폴더의 압축 아카이브(zip, gzip, bzip2, Java WAR 또는 tarball)를 생성합니다.

   > **Note**
   >
   > AWS OpsWorks Stacks는 압축되지 않은 tarball을 지원하지 않습니다.

2. 서버에 아카이브 파일을 업로드합니다.
3. 콘솔에서 리포지토리를 지정하려면 리포지토리 유형으로 HTTP 아카이브를 선택하고 해당 URL을 입력합니다.

   아카이브가 암호로 보호된 경우 애플리케이션 소스에서 로그인 자격 증명을 지정합니다.

Amazon S3 아카이브

Amazon Simple Storage Service 버킷을 리포지토리로 사용하려면:

1. 퍼블릭 또는 프라이빗 Amazon S3 버킷을 생성합니다. 자세한 내용은 Amazon S3 설명서를 참조하세요.
2. AWS OpsWorks Stacks가 프라이빗 버킷에 액세스하려면 Amazon S3 버킷에 대한 읽기 전용 권한이 있는 사용자여야 하며 액세스 키 ID와 보안 액세스 키가 필요합니다. 자세한 내용은 AWS Identity and Access Management 설명서를 참조하세요.

3. 코드 및 연결된 모든 파일을 폴더에 저장한 다음 해당 폴더를 압축된 아카이브(zip, gzip, bzip2, Java WAR 또는 tarball)로 저장합니다.

> **Note**
>
> AWS OpsWorks Stacks는 압축되지 않은 tarball을 지원하지 않습니다.

4. 이 아카이브 파일을 Amazon S3 버킷에 업로드하고 URL을 기록해 둡니다.
5. Stacks 콘솔에서 AWS OpsWorks 리포지토리를 지정하려면 리포지토리 유형을 S3 Archive로 설정하고 아카이브의 URL을 입력합니다. 프라이빗 아카이브의 경우, 정책에 따라 버킷 액세스 권한을 배포하는 AWS 액세스 키 ID 및 보안 액세스 키를 제공해야 합니다. 퍼블릭 아카이브의 경우 이러한 설정을 비워 둡니다.

Git 리포지토리

Git 리포지토리는 소스 제어 및 버전 관리를 제공합니다. AWS OpsWorks Stacks는 GitHub 또는 Bitbucket과 같은 공개적으로 호스팅되는 리포지토리 사이트와 프라이빗 호스팅되는 Git 서버를 지원합니다. 앱 및 Git 하위 모듈의 경우, [애플리케이션 소스]에서 리포지토리의 URL을 지정하는 데 사용하는 형식은 리포지토리가 퍼블릭인지 프라이빗인지에 따라 다릅니다.

퍼블릭 리포지토리 - HTTPS 또는 Git 읽기 전용 프로토콜을 사용합니다. 예를 들어 Chef 11 Linux 스택 시작하기는 다음 URL 형식 중 하나를 사용하여 액세스할 수 있는 퍼블릭 GitHub 리포지토리를 사용합니다.

- Git 읽기 전용: **`git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**
- HTTPS: **`https://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**

프라이빗 리포지토리 - 이 예제에 나온 SSH 읽기/쓰기 형식을 사용합니다.

- Github 리포지토리: **`git@github.com:`*`project`*`/`*`repository`***.
- Git 서버의 리포지토리: ***`user`*`@`*`server`*`:`*`project`*`/`*`repository`***

소스 제어에서 Git을 선택하면 다음과 같은 두 가지 추가 옵션 설정이 표시됩니다.

[리포지토리 SSH 키]

프라이빗 Git 리포지토리에 액세스하려면 배포 SSH 키를 지정해야 합니다. 이 필드는 프라이빗 키가 필요합니다. 퍼블릭 키는 Git 리포지토리에 할당됩니다. Git 하위 모듈의 경우, 지정된 키는 이러한 하위 모듈에 액세스할 권한이 있어야 합니다. 자세한 내용은 Git 리포지토리 SSH 키 사용 단원을 참조하십시오.

> **Important**
>
> SSH 키 배포에는 암호가 필요하지 않습니다. AWS OpsWorks Stacks는 암호를 전달할 방법이 없습니다.

브랜치/개정

리포지토리에 여러 브랜치가 있는 경우 AWS OpsWorks Stacks는 기본적으로 마스터 브랜치를 다운로드합니다. 특정 브랜치를 지정하려면 브랜치 이름, SHA1 해시 또는 태그 이름을 입력하세요. 특정 커밋을 지정하려면 완전한 40-hexdigit 커밋 식별자를 입력합니다.

기타 리포지토리

표준 리포지토리가 요구 사항을 충족하지 못하는 경우, Bazaar 같은 기타 리포지토리를 사용할 수 있습니다. 그러나 AWS OpsWorks Stacks는 이러한 리포지토리에서 앱을 자동으로 배포하지 않습니다. 배포 프로세스를 처리하고 적절한 계층의 Deploy 이벤트에 할당하려면 사용자 지정 레시피를 구현해야 합니다. Deploy 레시피를 구현하는 방법의 예제는 Deploy 레시피 단원을 참조하세요.

데이터 원본

이 섹션은 데이터베이스를 앱에 연결합니다. 다음과 같은 옵션이 있습니다:

- RDS – 스택의 Amazon RDS 서비스 계층 중 하나를 연결하세요.
- None – 데이터베이스 서버를 연결하지 마세요.

RDS를 선택한 경우 다음을 지정해야 합니다.

데이터베이스 인스턴스

목록에는 모든 Amazon RDS 서비스 계층이 포함됩니다. 다음 중 하나도 선택할 수 있습니다.

(필수) 어떤 데이터베이스 서버를 앱에 연결할지 지정하세요. 목록의 콘텐츠는 데이터 원본에 따라 달라집니다.

- RDS – 스택의 Amazon RDS 서비스 계층의 목록.

데이터베이스 이름

(선택 사항) 데이터베이스 이름을 지정합니다.

- Amazon RDS 계층 - Amazon RDS 인스턴스에 대해 지정한 데이터베이스 이름을 입력합니다.

  Amazon RDS 콘솔에서 데이터베이스 이름을 가져올 수 있습니다.

연결된 데이터베이스와 함께 앱을 배포하면 AWS OpsWorks Stacks는 데이터베이스 인스턴스의 연결을 앱의 `deploy` 속성에 추가합니다.

사용자 지정 레시피를 작성하여 `deploy` 속성에서 정보를 검색하고 애플리케이션이 액세스할 수 있는 파일에 넣을 수 있습니다. 이것은 기타 애플리케이션 유형에 데이터베이스 연결 정보를 제공할 수 있는 유일한 옵션입니다.

데이터베이스 연결을 처리하는 방법에 대한 자세한 정보는 데이터베이스에 연결를 참조하세요.

앱에서 데이터베이스 서버를 분리하려면 앱의 구성을 편집하여 다른 데이터베이스 서버를 지정하거나 아무 서버도 지정하지 마십시오.

환경 변수

각 앱마다 앱에 고유한 환경 변수 집합을 지정할 수 있습니다. 예를 들어 2개의 앱이 있는 경우, 첫 번째 앱에 대해 정의하는 환경 변수는 두 번째 앱이 사용할 수 없으며, 그 반대의 경우도 마찬가지입니다. 또 여러 앱에 대해 같은 환경 변수를 지정하여 환경 변수에 앱마다 다른 값을 할당할 수 있습니다.

> **Note**
>
> 환경 변수의 수에는 특별한 제한이 없습니다. 다만 변수의 이름, 값, 보호되는 플래그 값을 포함한 연결된 데이터 구조의 크기는 20KB를 초과할 수 없습니다. 이 한도는 전부는 아니더라도 대부분의 사용 사례를 수용해야 합니다. 이 한도를 초과하면 "Environment: is too large (maximum is 20KB)"라는 메시지와 함께 서비스 오류(콘솔) 또는 예외(API)를 초래합니다.

AWS OpsWorks Stacks는 변수를 앱의 속성에 `deploy` 속성으로 저장합니다. 표준 Chef 노드 구문을 사용하여 사용자 지정 레시피가 이러한 값을 검색하도록 할 수 있습니다. 앱의 환경 변수에 액세스하는 방법의 예제는 환경 변수 사용를 참조하세요.

키

변수 이름. 변수 이름은 최대 64자의 대문자, 소문자, 숫자, 밑줄(_)을 포함할 수 있지만 문자나 밑줄로 시작해야 합니다.

값

변수의 값. 변수의 값은 최대 256자를 포함할 수 있으며, 모두 인쇄 가능해야 합니다.

보호되는 값

값이 보호되는지 여부. 이 설정을 통해 암호 같은 중요한 정보를 감출 수 있습니다. 변수에 대해 [보호된 값]을 설정하는 경우, 앱 생성 후

- 앱의 세부 정보 페이지에 변수의 이름만 표시되며 변수의 값은 표시되지 않습니다.
- 앱을 편집할 권한이 있는 경우, [값 업데이트]를 클릭하여 새 값을 지정할 수 있지만 예전 값을 보거나 편집할 수는 없습니다.

> **Note**
>
> Chef 배포 로그는 종종 환경 변수를 포함하기도 합니다. 다시 말해 콘솔에서 보호된 변수를 볼 수 있습니다. 콘솔에서 보호된 변수를 보지 못하게 하려면 콘솔에서 보지 않기를 원하는 보호된 변수의 스토리지로서 Amazon S3 버킷을 사용하시기 바랍니다. 이런 목적으로 S3 버킷을 사용하는 방법의 예는 이 설명서의 Amazon S3 버킷 사용하기 단원을 참조하세요.

도메인 및 SSL 설정

기타 앱 유형의 경우 AWS OpsWorks Stacks는 앱의 `deploy` 속성에 설정을 추가합니다. 레시피는 이러한 속성에서 데이터를 가져와 필요에 맞게 서버를 구성할 수 있습니다.

도메인 설정

이 섹션에는 도메인 지정을 위한 옵션인 [도메인 추가] 필드가 있습니다. 자세한 내용은 사용자 지정 도메인 사용 단원을 참조하십시오.

SSL 설정

이 섹션에는 SSL 활성화 또는 비활성화에 사용할 수 있는 [SSL 지원] 토글이 있습니다. [예]를 클릭하는 경우, SSL 인증서 정보를 제공해야 합니다. 자세한 내용은 SSL 사용 단원을 참조하십시오.

앱 배포

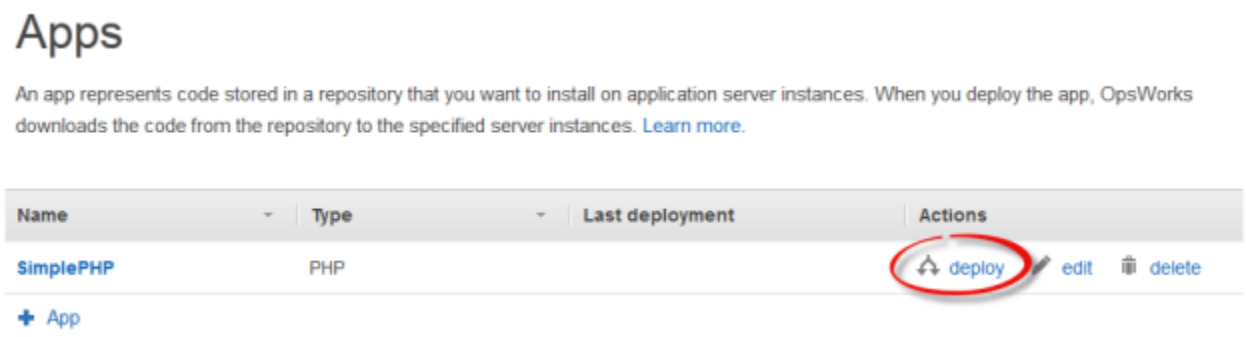
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

배포의 주된 목적은 애플리케이션 코드와 관련 파일을 애플리케이션 서버 인스턴스에 배포하는 것입니다. 배포 작업은 인스턴스의 계층이 결정하는 각 인스턴스의 Deploy 레시피에 의해 처리됩니다.

인스턴스를 시작하면 설정 레시피가 완료되면 AWS OpsWorks Stacks가 인스턴스의 배포 레시피를 자동으로 실행합니다. 하지만 앱을 추가하거나 수정할 때는 온라인 인스턴스에 수동으로 앱을 배포해야 합니다. 앱을 배포하려면 Manage 또는 Deploy 권한이 있어야 합니다. 자세한 내용은 사용자 권한 관리 단원을 참조하십시오.

앱을 배포하려면

1. [앱] 페이지에서 앱의 [배포] 작업을 클릭합니다.



> **Note**
>
> 탐색 창에서 [배포]를 클릭하여 앱을 배포할 수도 있습니다. [배포 및 명령] 페이지에서 [앱 배포]를 클릭합니다. 이때 배포할 앱을 선택할 수도 있습니다.

2. 다음을 지정합니다.

   - (필수) 아직 선택하지 않은 경우 [명령:]을 [배포]로 설정합니다.
   - (선택 사항) 설명을 입력합니다.

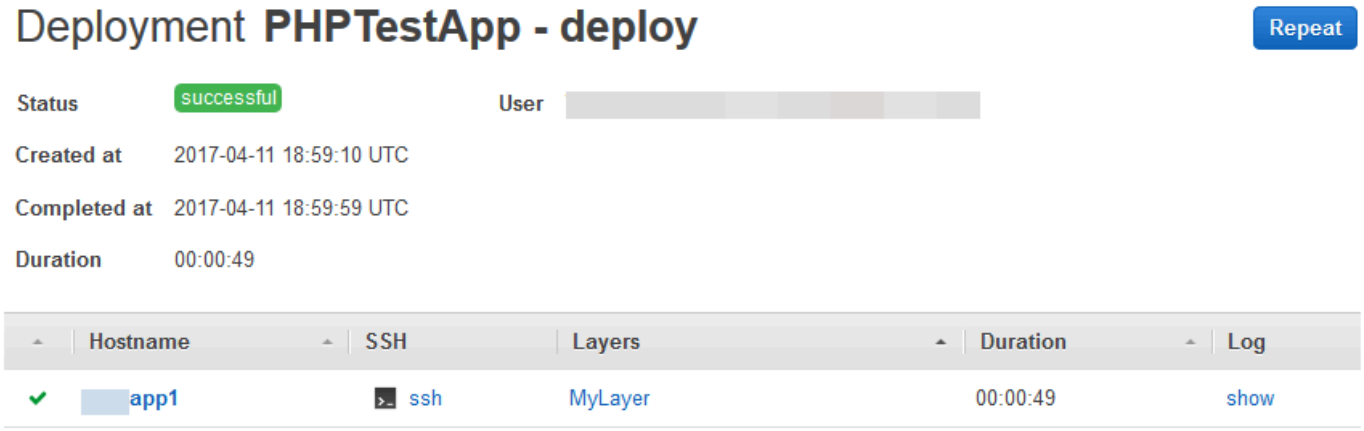
3. 고급 >>를 클릭하여 사용자 지정 JSON을 지정합니다. AWS OpsWorks 스택은 노드 객체에 스택 구성 및 배포 속성 세트를 추가합니다. deploy 속성은 배포 세부 정보를 포함하고 있고 Deploy 레시피에서 설치 및 구성을 처리하는 데 사용할 수 있습니다. Linux 스택에서 사용자 지정 JSON 필드를 사용하여 기본 AWS OpsWorks 스택 설정을 재정의하거나 사용자 지정 설정을 사용자 지정 레시피에 전달할 수 있습니다. 사용자 지정 JSON을 사용하는 방법에 대한 자세한 정보는 사용자 지정 JSON 사용 단원을 참조하세요.

   > **Note**
   >
   > 여기서 사용자 지정 JSON을 지정하면 이 배포 전용 스택 구성 및 배포 속성에 해당 JSON이 추가됩니다. 사용자 지정 JSON을 영구적으로 추가하려면 해당 JSON을 스택에 추가해야 합니다. 사용자 지정 JSON은 120KB로 제한됩니다. 용량이 더 필요할 경우 일부 데이터를 Amazon S3에 저장하는 것이 좋습니다. 그러면 사용자 지정 레시피가 AWS CLI 또는 AWS SDK for Ruby 사용하여 데이터를 버킷에서 인스턴스로 다운로드할 수 있습니다. 예시는 SDK for Ruby 사용에서 확인하십시오.

4. [인스턴스]에서 [고급 >>]을 클릭하고 배포 명령을 실행할 인스턴스를 지정합니다.

   배포 명령은 선택한 인스턴스에서 Deploy 레시피를 실행하는 Deploy 이벤트를 트리거합니다. 연결된 애플리케이션 서버에 대한 Deploy 레시피는 리포지토리에서 코드 및 관련 파일을 다운로드하여 인스턴스에 설치합니다. 따라서 일반적으로 연결된 애플리케이션 서버 인스턴스를 모두 선택합니다. 그러나 다른 인스턴스 유형의 경우 새로운 앱을 수용하려면 구성을 일부 변경해야 할 수 있으므로 대개 그러한 인스턴스에 대해서도 Deploy 레시피를 실행하는 것이 좋습니다. 이러한 레시피는 필요에 따라 구성을 업데이트하지만 앱의 파일은 설치하지 않습니다. 레시피에 대한 자세한 내용은 쿡북과 레시피 단원을 참조하세요.

5. [배포]를 클릭하여 지정한 인스턴스에 대해 배포 레시피를 실행합니다. 그러면 [배포] 페이지가 표시됩니다. 프로세스가 완료되면 AWS OpsWorks Stacks는 앱에 그린 체크 표시를 하여 배포 성공 여부를 나타냅니다. 배포에 실패하면 AWS OpsWorks Stacks는 앱을 빨간색 X로 표시합니다.이 경우 배포 페이지로 이동하여 배포 로그에서 자세한 내용을 확인할 수 있습니다.



> **Note**
>
> JSP 앱에 업데이트를 배포하면 Tomcat이 업데이트를 인식하지 못하고 기존 앱 버전을 계속 실행할 수 있습니다. 예를 들어 JSP 페이지만 포함된 .zip 파일로 앱을 배포하는 경우, 이런 현상이 생길 수 있습니다. Tomcat이 가장 최근에 배포된 버전을 실행하도록 하려면 `web.xml` 파일이 포함된 WEB-INF 디렉터리가 프로젝트의 루트 디렉터리에 포함되어야 합니다. `web.xml` 파일에는 다양한 콘텐츠가 포함될 수 있지만 Tomcat이 업데이트를 인식하고 현재 배포된 앱 버전을 실행하도록 하는 데는 다음으로 충분합니다. 각 업데이트의 버전을 변경할 필요는 없습니다. Tomcat은 버전이 변경되지 않은 경우에도 업데이트를 인식합니다.
>
> ```
> <context-param>
>   <param-name>appVersion</param-name>
>   <param-value>0.1</param-value>
> </context-param>
> ```

기타 배포 명령

[앱 배포] 페이지에는 앱 및 연결된 서버를 관리하기 위한 몇 가지 다른 명령이 포함됩니다. Chef 12 스택의 앱에는 다음 명령 중 [배포 취소]만 사용할 수 있습니다.

Undeploy

undeploy 레시피를 실행해 지정된 인스턴스에서 앱의 모든 버전을 제거하는 Undeploy 수명 주기 이벤트를 트리거합니다.

롤백

이전에 배포된 앱 버전을 복원합니다. 예를 들어 앱을 세 번 배포한 다음 [롤백]을 실행하면 서버는 두 번째 배포의 앱에 서비스합니다. [롤백]을 다시 실행하면 서버는 첫 번째 배포의 앱에 서비스합니다. 기본적으로 AWS OpsWorks Stacks는 5개의 최신 배포를 저장하므로 최대 4개의 버전을 롤백할 수 있습니다. 저장된 버전 수를 초과하는 경우, 명령은 실패하고 가장 오래된 버전을 남깁니다. 이 명령은 Chef 12 스택에서는 사용할 수 없습니다.

웹 서버 시작

지정된 인스턴스에서 애플리케이션 서버를 시작하는 레시피를 실행합니다. 이 명령은 Chef 12 스택에서는 사용할 수 없습니다.

웹 서버 중지

지정된 인스턴스에서 애플리케이션 서버를 중지하는 레시피를 실행합니다. 이 명령은 Chef 12 스택에서는 사용할 수 없습니다.

웹 서버 다시 시작

지정된 인스턴스에서 애플리케이션 서버를 다시 시작하는 레시피를 실행합니다. 이 명령은 Chef 12 스택에서는 사용할 수 없습니다.

Important

[웹 서버 시작], [웹 서버 중지], [웹 서버 다시 시작], [롤백]은 기본적으로 레시피 실행 스택 명령의 사용자 지정 버전입니다. 이들은 지정된 인스턴스에서 작업을 수행하는 레시피 집합을 실행합니다.

- 이러한 명령은 수명 주기 이벤트를 트리거하지 않으므로 사용자 지정 코드를 실행하도록 후크할 수 없습니다.
- 이 명령들은 내장 애플리케이션 서버 계층에만 작동합니다.

  특히 이 명령들은 애플리케이션 서버를 지원하는 경우에도 사용자 지정 계층에는 아무 영향이 없습니다. 사용자 지정 계층에서 서버를 시작, 중지 또는 다시 시작하려면 사용자 지정 레시피를 구현하여 이러한 작업을 수행하고 레시피 실행 스택 명령을 사용하여 실행해야 합니다. 사용자 지정 레시피를 구현하고 설치하는 방법에 대한 자세한 정보는 쿡북과 레시피 단원을 참조하세요.

앱 편집

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

앱을 편집하여 앱 구성을 수정할 수 있습니다. 예를 들어 새 버전을 배포할 준비가 되면 앱의 AWS OpsWorks Stacks 설정을 편집하여 새 리포지토리 브랜치를 사용할 수 있습니다. 앱 구성을 편집하려면 Manage 또는 Deploy 권한이 있어야 합니다. 자세한 내용은 사용자 권한 관리 단원을 참조하십시오.

앱을 편집하려면

1. [앱] 페이지에서 앱 이름을 클릭하여 세부 정보 페이지를 엽니다.
2. [편집]을 클릭하여 앱의 구성을 변경합니다.
   - 앱 이름을 수정하면 AWS OpsWorks Stacks는 새 이름을 사용하여 콘솔에서 앱을 식별합니다.

     이름을 변경하더라도 연결된 짧은 이름은 변경되지 않습니다. 짧은 이름은 스택에 앱을 추가할 때 설정되며 이후에 수정할 수 없습니다.
   - 보호된 환경 변수를 지정하면 값을 확인하거나 편집할 수 없습니다. 그러나 [값 업데이트]를 클릭하여 새 값을 지정할 수 있습니다.
3. [저장]을 클릭하여 새 구성을 저장한 다음 [앱 배포]를 클릭하여 앱을 배포합니다.

앱을 편집하면 AWS OpsWorks Stacks의 설정이 변경되지만 스택의 인스턴스에는 영향을 주지 않습니다. 처음 앱을 배포하면 Deploy 레시피가 코드 및 관련 파일을 앱 서버 인스턴스로 다운로드하여 로컬 복사본을 실행합니다. 리포지토리에서 앱을 수정하거나 다른 설정을 변경하는 경우 다음과 같이 앱을 배포하여 앱 서버 인스턴스에 업데이트를 설치해야 합니다. AWS OpsWorks Stacks는 현재 앱 버전이 시작되면 새 인스턴스에 자동으로 배포합니다. 하지만 기존 인스턴스에서는 상황이 다릅니다.

- AWS OpsWorks Stacks는 현재 앱 버전이 시작되면 새 인스턴스에 자동으로 배포합니다.
- AWS OpsWorks Stacks는 로드 기반 인스턴스와 시간 기반 인스턴스를 포함한 오프라인 인스턴스가 다시 시작될 때 해당 인스턴스에 최신 앱 버전을 자동으로 배포합니다.
- 온라인 인스턴스에는 업데이트된 앱을 수동으로 배포해야 합니다.

앱을 배포하는 방법에 대한 자세한 정보는 앱 배포 단원을 참조하세요.

데이터베이스 서버에 애플리케이션 연결

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

앱을 생성할 때 또는 나중에 앱을 편집하여 Amazon RDS 데이터베이스 서버를 앱에 연결합니다. 그러면 애플리케이션에서 데이터베이스 서버에 연결하기 위해 데이터베이스 연결 정보(사용자 이름, 암호 등)를 사용할 수 있습니다. 앱을 배포할 때 AWS OpsWorks Stacks는 다음 두 가지 방법으로 애플리케이션에이 정보를 제공합니다.

- Linux 스택의 경우, AWS OpsWorks Stacks가 각 내장 애플리케이션 서버 인스턴스에 애플리케이션이 데이터베이스 서버와 연결하는 데 사용할 수 있는 연결 데이터를 포함하는 파일을 생성합니다.
- AWS OpsWorks Stacks에는 각 인스턴스에 설치된 스택 구성 및 배포 속성의 연결 정보가 포함됩니다.

  이러한 속성에서 연결 정보를 추출하여 원하는 형식의 파일에 저장하는 사용자 지정 레시피를 구현할 수 있습니다. 자세한 내용은 애플리케이션으로 데이터 전달 단원을 참조하십시오.

> **Important**
>
> Linux 스택의 경우, Amazon RDS 서버 계층을 앱과 연결하려면 다음과 같이 적절한 드라이버 패키지를 연결된 앱 서버 계층에 추가해야 합니다.
>
> 1. 탐색 창에서 [계층]을 클릭하고 앱 서버의 [레시피] 탭을 엽니다.
> 2. [편집]을 클릭하고 적절한 드라이버 패키지를 [OS 패키지]에 추가합니다. 예를 들어 계층에 Amazon Linux 인스턴스가 포함된 경우 `mysql`을 지정하고 계층에 Ubuntu 인스턴스가 포함된 경우 `mysql-client`를 지정해야 합니다.
> 3. 변경 내용을 저장하고 앱을 재배포합니다.

사용자 지정 레시피 사용

앱의 deploy 속성에서 연결 데이터를 추출하여 애플리케이션이 읽을 수 있는 형식(예: YAML 파일)으로 저장하는 사용자 지정 레시피를 구현할 수 있습니다.

앱을 생성할 때 또는 나중에 앱을 편집하여 데이터베이스 서버를 앱에 연결합니다. 앱을 배포할 때 AWS OpsWorks Stacks는 데이터베이스 연결 정보가 포함된 각 인스턴스에 스택 구성 및 배포 속성을 설치합니다. 그러면 앱이 해당 속성을 가져올 수 있습니다. 세부 사항은 Linux 또는 Windows 스택에 따라 다릅니다.

Linux 스택용 데이터베이스 서버에 연결

Linux 스택의 경우, 스택 구성 및 배포 속성의 deploy 네임스페이스가 배포된 각 앱의 속성을 앱의 짧은 이름으로 명명하여 포함하고 있습니다. 데이터베이스 서버를 앱에 연결하면 AWS OpsWorks Stacks는 앱의 [:database] 속성을 연결 정보로 채우고 각 후속 배포를 위해 스택의 인스턴스에 설치합니다. 속성 값은 사용자가 제공하거나 AWS OpsWorks Stacks가 생성합니다.

> **Note**
>
> AWS OpsWorks Stacks를 사용하면 데이터베이스 서버를 여러 앱에 연결할 수 있지만 각 앱에는 연결된 데이터베이스 서버가 하나만 있을 수 있습니다. 애플리케이션을 여러 데이터베이스 서버에 연결하려는 경우 서버 중 하나를 앱에 연결하고 앱의 deploy 속성에 포함된 정보를 사용하여 해당 서버에 연결합니다. 사용자 지정 JSON을 사용하여 다른 데이터베이스 서버의 연결 정보를 애플리케이션에 전달합니다. 자세한 내용은 애플리케이션으로 데이터 전달 단원을 참조하십시오.

애플리케이션이 인스턴스의 deploy 속성에 포함된 연결 정보를 사용하여 데이터베이스 서버에 연결할 수 있습니다. 하지만 애플리케이션은 해당 정보에 직접 액세스할 수 없으며 레시피만 deploy 속성에 액세스할 수 있습니다. 이 문제는 deploy 속성에서 연결 정보를 추출하여 애플리케이션이 읽을 수 있는 파일에 저장하는 사용자 지정 레시피를 구현하여 해결할 수 있습니다.

환경 변수 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제의 권장 사항은 Chef 11.10 및 이전 버전에 적용됩니다. Chef 12 이상 릴리스에서 환경 변수를 가져오려면 앱 데이터 백을 사용해야 합니다. 자세한 정보는 AWS OpsWorks 데이터 백 참조 및 앱 데이터 백(aws_opsworks_app) 단원을 참조하세요.

앱의 환경 변수를 지정하면 AWS OpsWorks Stacks는 앱의 deploy 속성에 변수 정의를 추가합니다.

사용자 지정 계층은 레시피를 사용하여 표준 노드 구문으로 변수의 값을 검색하고 계층의 앱이 액세스할 수 있는 형식으로 저장할 수 있습니다.

인스턴스의 deploy 속성에서 환경 변수 값을 가져오는 사용자 지정 레시피를 구현해야 합니다. 그러면 레시피가 데이터를 애플리케이션이 액세스할 수 있는 형식(예: YAML 파일)으로 인스턴스에 저장할 수 있습니다. 앱의 환경 변수 정의는 앱의 deploy에서 environment_variables 속성에 저장됩니다. 다음 예제는 JSON을 사용하여 속성 구조를 표시함으로써 simplephpapp 앱의 이러한 속성의 위치를 보여줍니다.

```
{
  ...
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
    ...
  }
}
```

레시피는 표준 노드 구문을 사용하여 변수 값을 가져올 수 있습니다. 다음 예제는 이전 JSON에서 USER_ID 값을 가져와 Chef 로그에 기록하는 방법을 보여줍니다.

```
Chef::Log.info("USER_ID: #{node[:deploy]['simplephpapp'][:environment_variables]
[:USER_ID]}")
```

스택 구성 및 배포 JSON에서 정보를 검색하여 인스턴스에 저장하는 방법에 대한 상세한 설명은 애플리케이션으로 데이터 전달 단원을 참조하세요.

애플리케이션으로 데이터 전달

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

키-값 페어와 같은 데이터를 서버 상의 애플리케이션으로 전달하는 것이 유용한 경우가 있습니다. 이렇게 하려면 사용자 지정 JSON을 사용하여 데이터를 스택에 추가합니다. AWS OpsWorks Stacks는 각 수명 주기 이벤트에 대해 각 인스턴스의 노드 객체에 데이터를 추가합니다.

그러나 레시피는 Chef 속성을 사용하여 노드 객체로부터 사용자 지정 JSON 데이터를 가져올 수 있지만 애플리케이션을 그럴 수 없습니다. 사용자 지정 JSON 데이터를 하나 이상의 애플리케이션으로 가져오기 위한 접근 방법 한 가지는 node 객체에서 데이터를 추출하여 애플리케이션이 읽을 수 있는 파일에 기록하는 사용자 지정 레시피를 구현하는 것입니다. 이 항목의 예제에서는 데이터를 YAML 파일로 기록하는 방법을 보여주지만, JSON 또는 XML과 같은 다른 형식에도 동일한 기본 접근 방법을 사용할 수 있습니다.

키-값 데이터를 스택의 인스턴스에 전달하려면 다음과 같은 사용자 지정 JSON을 스택에 추가합니다. 사용자 지정 JSON을 스택에 추가하는 방법에 대한 자세한 정보는 사용자 지정 JSON 사용 단원을 참조하세요.

```
{
  "my_app_data": {
    "app1": {
```

```
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    },
    "app2": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    }
  }
}
```

예제에서는 짧은 이름이 app1 및 app2이고 각각 세 개의 데이터 값을 갖는 앱이 두 개 있는 것으로 가정합니다. 함께 제공되는 레시피는 사용자가 앱의 짧은 이름을 사용하여 연결된 데이터를 식별하는 것으로 가정합니다. 다른 이름은 임의의 이름입니다. 앱 짧은 이름에 대한 자세한 정보는 설정 단원을 참조하세요.

다음 예제의 레시피는 deploy 속성에서 각 앱에 대한 데이터를 추출하여 .yml 파일에 저장하는 방법을 보여줍니다. 이 레시피는 사용자 지정 JSON이 각 앱에 대한 데이터를 포함하는 것으로 가정합니다.

```
node[:deploy].each do |app, deploy|
  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
end
```

deploy 속성에는 각 앱마다 앱의 짧은 이름으로 명명된 속성이 하나씩 포함됩니다. 각 앱 속성에는 앱에 대한 다양한 정보를 표현하는 속성 세트가 포함됩니다. 이 예제는 [:deploy][:*app_short_name*][:deploy_to] 속성으로 표현되는 앱의 배포 디렉터리를 사용합니다. [:deploy]에 대한 자세한 내용은 deploy 속성 단원을 참조하세요.

deploy의 각 앱에 대해 레시피가 다음을 수행합니다.

1. 애플리케이션의 app_data.yml 디렉터리 내 shared/config 하위 디렉터리에 [:deploy_to]라는 파일을 생성합니다.

   AWS OpsWorks Stacks가 앱을 설치하는 방법에 대한 자세한 내용은 섹션을 참조하세요Deploy 레시피.
2. 앱의 사용자 지정 JSON 값을 YAML으로 변환하여 app_data.yml에 기록합니다.

앱에 데이터를 전달하려면

1. 스택에 앱을 추가하고 해당 앱의 짧은 이름을 기록해 둡니다. 자세한 내용은 앱 추가 단원을 참조하십시오.
2. 앞서 설명한 대로 앱의 데이터가 포함된 사용자 지정 JSON을 deploy 속성에 추가합니다. 사용자 지정 JSON을 스택에 추가하는 방법에 대한 자세한 정보는 사용자 지정 JSON 사용 단원을 참조하세요.
3. 쿡북을 생성한 다음, 사용자 지정 JSON에 사용한 속성 이름에 맞게 수정한 이전 예제의 코드를 사용하여 쿡북에 레시피를 추가합니다. 쿡북 및 레시피를 생성하는 방법에 대한 자세한 정보는 쿡북과 레시피 단원을 참조하세요. 이 스택에 대한 사용자 지정 쿡북이 이미 있는 경우 기존 쿡북에 레시피를 추가하거나 기존 Deploy 레시피에 코드를 추가할 수 있습니다.
4. 스택에 쿡북을 설치합니다. 자세한 내용은 사용자 지정 쿡북 설치 단원을 참조하십시오.
5. 앱 서버 계층의 수명 주기 배포 이벤트에 레시피를 할당합니다. 그러면 AWS OpsWorks 스택이 부팅된 후 새 인스턴스마다 레시피를 실행합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.
6. 앱을 배포합니다. 이제 사용자 데이터가 포함된 스택 구성 및 배포 속성이 설치됩니다.

> **Note**
>
> 데이터 파일이 앱 배포 전에 배치되어야 할 경우 레시피를 계층의 설정 수명 주기 이벤트에도 할당할 수 있습니다. 이 이벤트는 인스턴스 부팅이 완료된 직후 한 번 발생합니다. 그러나 AWS OpsWorks Stacks는 아직 배포 디렉터리를 생성하지 않았으므로 레시피는 데이터 파일을 생성하기 전에 필요한 디렉터리를 명시적으로 생성해야 합니다. 다음 예제에서는 명시적으로 앱의 /shared/config 디렉터리를 생성한 후 여기에 데이터 파일을 생성합니다.

```
node[:deploy].each do |app, deploy|

 directory "#{deploy[:deploy_to]}/shared/config" do
      owner "deploy"
      group "www-data"
      mode 0774
      recursive true
      action :create
    end

  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
```

```
end
```

데이터를 로드하려면 다음의 Sinatra 코드와 같은 코드를 사용할 수 있습니다.

```
#!/usr/bin/env ruby
# encoding: UTF-8
require 'sinatra'
require 'yaml'

get '/' do
  YAML.load(File.read(File.join('..', '..', 'shared', 'config', 'app_data.yml')))
End
```

다음과 같이 사용자 지정 JSON을 업데이트하여 언제라도 앱의 데이터 값을 업데이트할 수 있습니다.

앱 데이터를 업데이트하려면

1. 사용자 지정 JSON을 편집하여 데이터 값을 업데이트합니다.
2. 앱을 다시 배포하면 AWS OpsWorks 스택이 스택의 인스턴스에서 배포 레시피를 실행하도록 지시합니다. 이러한 레시피는 업데이트된 스택 구성의 속성 및 배포 속성을 사용하므로 사용자 지정 레시피가 현재 값으로 데이터 파일을 업데이트합니다.

Git 리포지토리 SSH 키 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

배포 SSH 키라고도 하는 Git 리포지토리 SSH 키는 프라이빗 Git 리포지토리에 대한 액세스 권한을 제공하는 암호 없는 SSH 키입니다. 이론적으로 이 키는 어느 특정 개발자의 소유가 아닙니다. 그 목적은 AWS OpsWorks Stacks가 사용자의 추가 입력 없이 Git 리포지토리에서 앱 또는 쿡북을 비동기적으로 배포할 수 있도록 허용하는 것입니다.

다음은 리포지토리 SSH 키를 생성하는 기본적 절차에 대한 설명입니다. 자세한 정보는 리포지토리 설명서를 참조하세요. 예컨대 배포 키 관리에서는 GitHub 리포지토리용 리포지토리 SSH 키 생성 방법을 설명하며, Bitbucket의 배포 키는 Bitbucket 리포지토리용 리포지토리 SSH 키 생성 방법을 설명합니다. 일부 설명서에서는 서버에서의 키 생성을 설명합니다. Stacks의 경우 지침에서 "서버"를 "워크스테이션"으로 바꾸기만 AWS OpsWorks 하면 됩니다.

리포지토리 SSH 키를 생성하려면

1. ssh-keygen과 같은 프로그램을 사용하여 워크스테이션에서 Git 리포지토리에 대한 배포 SSH 키 페어를 생성합니다.

   > **Important**
   >
   > AWS OpsWorks Stacks는 SSH 키 암호문을 지원하지 않습니다.

2. 리포지토리에 퍼블릭 키를 할당하고 워크스테이션에 프라이빗 키를 저장합니다.
3. 앱을 추가하거나 쿡북 리포지토리를 지정하는 경우 [리포지토리 SSH 키] 상자에 프라이빗 키를 입력합니다. 자세한 내용은 앱 추가 단원을 참조하십시오.

AWS OpsWorks Stacks는 리포지토리 SSH 키를 각 인스턴스에 전달한 다음 기본 제공 레시피에서 키를 사용하여 리포지토리에 연결하고 코드를 다운로드합니다. 키는 deploy 속성에 node[:deploy]['*appshortname*'][:scm][:ssh_key]로 저장되며, 루트 사용자만 액세스할 수 있습니다.

사용자 지정 도메인 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

도메인 이름을 타사와 함께 호스팅하는 경우, 해당 도메인 이름을 앱에 매핑할 수 있습니다. 기본적 절차는 다음과 같습니다.

1. DNS 등록 기관을 사용해 하위 도메인을 생성하여 로드 밸런서의 탄력적 IP 주소 또는 앱 서버의 퍼블릭 IP 주소에 매핑합니다.

2. 하위 도메인을 가리키도록 앱의 구성을 업데이트하고 앱을 다시 배포합니다.

> **Note**
>
> 정규화되지 않은 도메인 이름(예: myapp1.example.com)을 정규화된 도메인 이름(예: www.myapp1.example.com)으로 포워딩해야 둘 다 앱에 매핑됩니다.

앱을 위해 구성하는 도메인은 서버의 구성 파일에서 서버 별칭으로 나열됩니다. 로드 밸런서를 사용 중인 경우, 로드 밸런서는 요청이 들어올 때 URL에서 도메인 이름을 확인하여 도메인에 따라 트래픽을 리디렉션합니다.

하위 도메인을 IP 주소에 매핑하려면

1. 로드 밸런서를 사용하는 경우, [인스턴스] 페이지에서 로드 밸런서 인스턴스를 클릭하여 세부 정보 페이지를 연 다음 인스턴스의 [탄력적 IP] 주소를 확인합니다. 아니면 애플리케이션 서버 인스턴스의 세부 정보 페이지에서 퍼블릭 IP 주소를 확인합니다.
2. DNS 등록 기관에서 제공하는 설명에 따라 하위 도메인을 생성하여 1단계의 IP 주소에 매핑합니다.

> **Note**
>
> 어느 시점에 로드 밸런서 인스턴스가 종료되는 경우, 새로운 탄력적 IP 주소가 할당됩니다. 새 탄력적 IP 주소에 매핑하려면 DNS 등록 기관 설정을 업데이트해야 합니다.

AWS OpsWorks Stacks는 앱의 `deploy` 속성에 도메인 설정을 추가하기만 하면 됩니다. 노드 객체에서 정보를 검색하고 서버를 적절히 구성하려면 사용자 지정 레시피를 구현해야 합니다. 자세한 내용은 쿡북과 레시피 단원을 참조하십시오.

같은 애플리케이션 서버에서 여러 애플리케이션 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제의 내용은 Node.js 앱에는 적용되지 않습니다.

같은 유형의 애플리케이션이 여러 개인 경우, 같은 애플리케이션 서버 인스턴스에서 실행하면 경우에 따라 비용 효율이 높아집니다.

같은 서버에서 여러 애플리케이션을 실행하려면

1. 각 애플리케이션에 해당하는 스택에 앱을 추가합니다.
2. 각 앱의 개별 하위 도메인을 확인하고 이 하위 도메인을 애플리케이션 서버 또는 로드 밸런서의 IP 주소에 매핑합니다.
3. 각 앱의 구성을 편집하여 해당 하위 도메인을 지정합니다.

이러한 작업을 수행하는 방법에 대한 자세한 정보는 사용자 지정 도메인 사용 단원을 참조하세요.

> **Note**
>
> 애플리케이션 서버가 여러 HTTP 애플리케이션을 실행하는 경우, load-balancing에 Elastic Load Balancing를 사용합니다. HTTPS 애플리케이션이 여럿인 경우, 로드 밸런서에서 SSL 연결을 종료하거나 애플리케이션마다 별도의 스택을 생성해야 합니다. HTTPS 요청은 암호화되므로 서버에서 SSL 연결을 종료하면 로드 밸런서는 도메인 이름을 확인하여 어떤 애플리케이션이 요청을 처리해야 하는지 결정할 수 없습니다.

SSL 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

애플리케이션에 SSL을 사용하려면 먼저 인증 기관(CA)에서 디지털 서버 인증서를 가져와야 합니다. 간단히 하기 위해 이 안내서에서는 인증서를 생성한 다음 자체 서명합니다. 학습 및 테스트 용도에는 자체 서명된 인증서가 유용하지만 프로덕션 스택에는 항상 CA가 서명한 인증서를 사용해야 합니다.

이 안내서에서는 다음을 수행합니다.

1. OpenSSL을 설치 및 구성합니다.
2. 프라이빗 키를 만듭니다.
3. 인증서 서명 요청을 생성합니다.
4. 자체 서명된 인증서를 생성합니다.
5. 인증서 정보를 사용하여 애플리케이션을 편집합니다.

Important

애플리케이션이 SSL을 사용하는 경우, 가능하다면 애플리케이션 서버 계층에서 SSLv3를 비활성화하여 CVE-2014-3566에 설명된 취약성을 해결하는 것이 좋습니다. 스택에 Ganglia 계층이 포함된 경우, 해당 계층에서도 SSL v3를 비활성화해야 합니다. 세부 사항은 구체적인 계층에 따라 다르므로 자세한 정보는 다음을 참조하세요.

- Java 앱 서버 AWS OpsWorks 스택 계층
- Node.js 앱 서버 AWS OpsWorks 스택 계층
- PHP 앱 서버 AWS OpsWorks 스택 계층
- Rails 앱 서버 AWS OpsWorks 스택 계층
- 정적 웹 서버 AWS OpsWorks 스택 계층
- Ganglia 계층

주제

- 1단계: OpenSSL 설치 및 구성
- 2단계: 프라이빗 키 생성
- 3단계: 인증서 서명 요청 생성

- 4단계: 인증 기관에 CSR 제출
- 5단계: 앱 편집

1단계: OpenSSL 설치 및 구성

서버 인증서를 생성하고 업로드하려면 SSL 및 TLS 프로토콜을 지원하는 도구가 필요합니다. OpenSSL은 RSA 토큰을 만들고 사용자의 프라이빗 키를 사용하여 서명하는 데 필요한 기본 암호화 기능을 제공하는 오픈 소스 도구입니다.

다음 절차에서는 컴퓨터에 아직 OpenSSL이 설치되어 있지 않다고 가정합니다.

Linux 및 Unix에서 OpenSSL을 설치하려면

1. OpenSSL: Source, Tarballs로 이동합니다.
2. 최신 소스를 다운로드합니다.
3. 패키지를 빌드합니다.

Windows에서 OpenSSL을 설치하려면

1. Microsoft Visual C++ 2008 재배포 가능 패키지가 시스템에 아직 설치되지 않은 경우 패키지를 다운로드하세요.
2. 설치 관리자를 실행하고 Microsoft Visual C++ 2008 재배포 가능 패키지 설치 마법사의 지침에 따라 설치합니다.
3. OpenSSL: Binary Distributions로 이동하여 환경에 맞는 OpenSSL 바이너리 버전을 클릭하고 설치 관리자를 로컬에 저장합니다.
4. 설치 관리자를 실행하고 OpenSSL 설치 마법사의 지침에 따라 바이너리를 설치합니다.

터미널 또는 명령 창을 열고 다음 명령줄을 사용하여 OpenSSL 설치 지점을 가리키는 환경 변수를 만듭니다.

- Linux 및 Unix에서

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

- Windows

```
set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

터미널 또는 명령 창을 열고 다음 명령줄을 사용하여 OpenSSL 이진수 경로를 컴퓨터의 경로 변수에 추가합니다.

- Linux 및 Unix에서

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

- Windows

```
set Path=OpenSSL_HOME\bin;%Path%
```

> **Note**
>
> 이 명령줄을 사용한 환경 변수의 변경은 현재 명령줄 세션에만 유효합니다.

2단계: 프라이빗 키 생성

인증서 서명 요청(CSR)을 생성하려면 고유한 프라이빗 키가 필요합니다. 다음 명령줄을 사용하여 키를 생성합니다.

```
openssl genrsa 2048 > privatekey.pem
```

3단계: 인증서 서명 요청 생성

인증서 서명 요청(CSR)은 디지털 서버 인증서를 신청하기 위해 인증 기관(CA)으로 전송되는 파일입니다. 다음 명령줄을 사용하여 CSR을 생성합니다.

```
openssl req -new -key privatekey.pem -out csr.pem
```

명령의 출력은 다음과 비슷합니다.

```
You are about to be asked to enter information that will be incorporated
 into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields but you can leave some blank
 For some fields there will be a default value,
 If you enter '.', the field will be left blank.
```

다음 표는 인증서 요청을 생성할 때 참조할 수 있습니다.

인증서 요청 데이터

명칭	설명	예제
국가 이름	해당 국가의 두 자리 ISO 약자.	US = 미국
주 또는 지방	해당 조직이 위치한 주 또는 지방의 이름. 이 이름은 약어로 사용할 수 없음.	워싱턴
시 이름	해당 조직이 위치한 주 또는 시의 이름.	시애틀
조직 이름	해당 조직의 정식 이름. 조직 이름의 약칭을 사용하지 마세요.	CorporationX
조직 단위	(선택 사항) 조직에 대한 추가 정보.	마케팅
일반 이름	CNAME의 정규화된 도메인 이름. 이 이름이 정확하게 일치하지 않으면 인증서 이름 검사 경고를 받게 됩니다.	www.example.com
이메일 주소	서버 관리자의 이메일 주소	someone@example.com

> **Note**
>
> 일반 이름 필드는 종종 잘못 이해하여 잘못된 이름을 입력하는 경우가 많습니다. 일반 이름이란 호스트에 도메인 이름을 붙인 것입니다. 예를 들면 "www.example.com" 또는 "example.com"와 같은 형태입니다. 정확한 일반 이름을 사용하여 CSR을 생성해야 합니다.

4단계: 인증 기관에 CSR 제출

프로덕션용으로 CSR을 인증 기관(CA)에 제출하여 서버 인증서를 얻으려면 다른 자격 증명이나 ID 증명이 필요할 수 있습니다. 신청이 성공하면 CA에서 디지털 서명된 자격 증명 인증서와 가능한 경우 체

인 인증서 파일을 보내 줍니다. AWS는 특정 CA를 추천하지 않습니다. 이용 가능한 일부 CA 목록을 보려면 Wikipedia에서 인증 기관 - 공급자를 참조하세요.

테스트 용도로만 사용할 수 있는 자체 서명된 인증서를 생성할 수도 있습니다. 이 예제에서는 다음 명령줄을 사용하여 자체 서명된 인증서를 생성합니다.

```
openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out server.crt
```

다음과 비슷한 출력이 표시될 것입니다.

```
Loading 'screen' into random state - done
Signature ok
subject=/C=us/ST=washington/L=seattle/O=corporationx/OU=marketing/CN=example.com/
emailAddress=someone@example.com
Getting Private key
```

5단계: 앱 편집

인증서를 생성하고 서명했으면 SSL을 사용할 수 있도록 앱을 업데이트하고 인증서 정보를 제공하세요. 앱 페이지에서 세부 정보 페이지를 열 앱을 선택한 다음 앱 편집을 클릭합니다. SSL 지원을 활성화하려면 SSL 활성화를 예로 설정합니다. 그러면 다음 구성 옵션이 표시됩니다.

SSL 인증서

퍼블릭 키 인증서(.crt) 파일의 콘텐츠를 상자에 붙여 넣습니다. 인증서는 다음과 비슷해야 합니다.

```
-----BEGIN CERTIFICATE-----
MIICuTCCAiICCQCtqFKItVQJpzANBgkqhkiG9w0BAQUFADCBoDELMAkGA1UEBhMC
dXMxEzARBgNVBAgMCndhc2hpbmd0b24xEDAOBgNVBAcMB3NlYXR0bGUxDzANBgNV
BAoMBmFtYXpvbjEWMBQGA1UECwwNRGV2IGFuZCBUb29sczEdMBsGA1UEAwwUc3Rl
cGhhbmllYXBpZXJjZS5jb20xIjAgBgkqhkiG9w0BCQEWE3NhcGllcmNlQGFtYXpv
...
-----END CERTIFICATE-----
```

> **Note**
>
> Nginx를 사용하고 인증서 체인 파일이 있는 경우, 콘텐츠를 퍼블릭 키 인증서 파일에 추가해야 합니다.

기존 인증서를 업데이트하는 경우, 다음을 수행합니다.

- SSL 인증서 업데이트를 선택해 인증서를 업데이트합니다.
- 새 인증서가 기존 프라이빗 키와 맞지 않는 경우, SSL 인증서 키 업데이트를 선택합니다.
- 새 인증서가 기존 인증서 체인과 맞지 않는 경우, SSL 인증서 업데이트를 선택합니다.

SSL 인증서 키

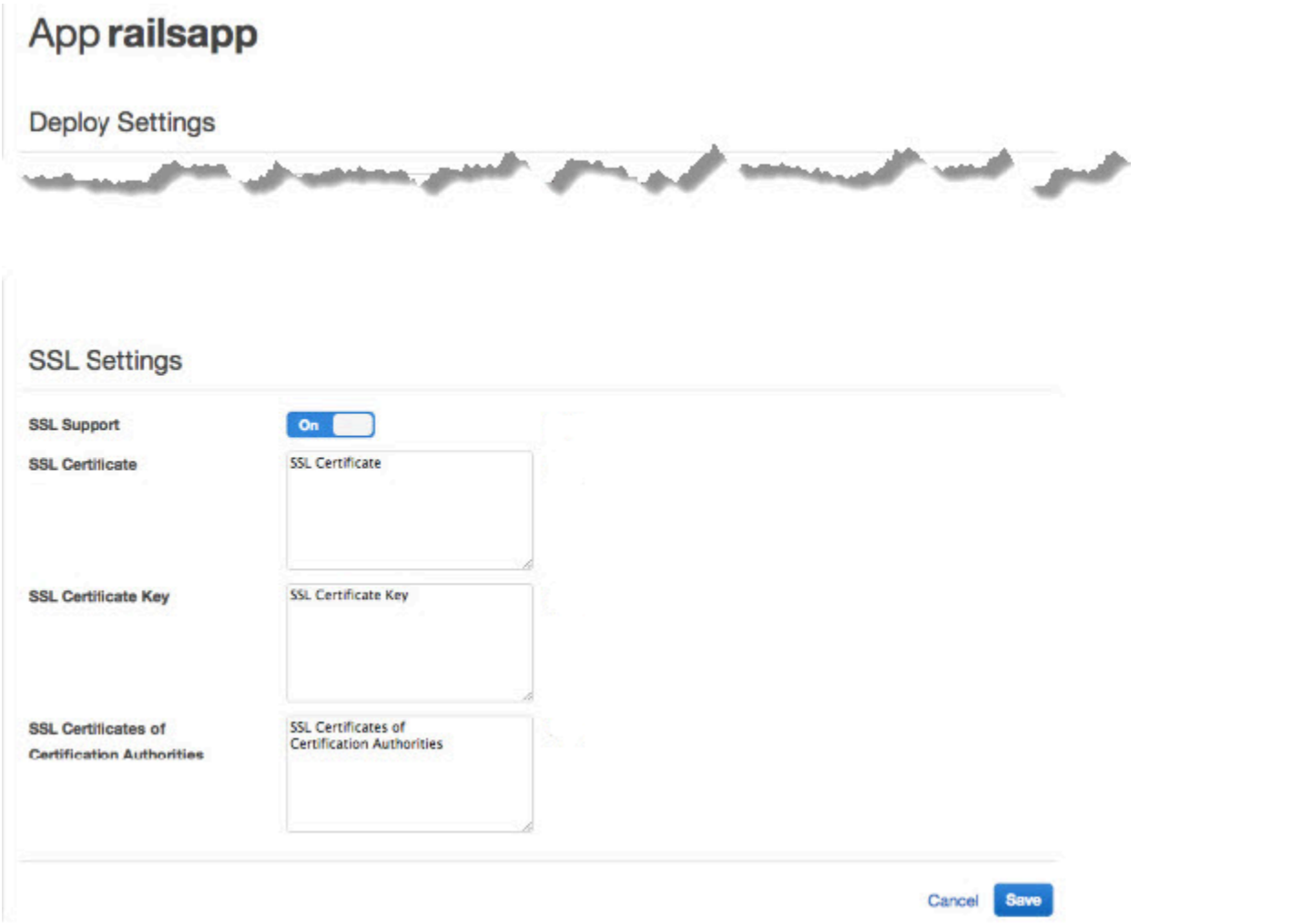
프라이빗 키 파일(.pem 파일)의 콘텐츠를 상자에 붙여 넣습니다. 다음과 같이 보여야 합니다.

```
----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC0CYklJY5r4vV2NHQYEpwtsLuMMBhylMrgBShKq+HHVLYQQCL6
+wGIiRq5qXqZlRXje3GM5Jvcm6q0R71MfRIl1FuzKyqDtneZaAIEYniZibHiUnmO
/UNqpFDosw/6hY3ONk0fSBlU4ivD0Gjpf6J80jL3DJ4R23Ed0sdL4pRT3QIDAQAB
AoGBAKmMfWrNRqYVtGKgnWB6Tji9QrKQLMXjmHeGg95mppdJELiXHhpMvrHtpIyK
...
-----END RSA PRIVATE KEY-----
```

[인증 기관의 SSL 인증서]

인증서 체인 파일이 있는 경우, 콘텐츠를 상자에 붙여 넣습니다.

> **Note**
>
> Nginx를 사용하는 경우, 이 상자를 비워 두어야 합니다. 인증서 체인 파일이 있는 경우, SSL 인증서에서 퍼블릭 키 인증서 파일에 추가해야 합니다.



[저장]을 클릭한 후 애플리케이션을 다시 배포하여 온라인 인스턴스를 업데이트합니다.

기본 제공 애플리케이션 서버 계층의 경우 AWS OpsWorks Stacks는 서버 구성을 자동으로 업데이트합니다. 배포가 완료되면 다음과 같이 OpenSSL 설치가 제대로 되었는지 확인할 수 있습니다.

OpenSSL 설치를 확인하려면

1. [인스턴스] 페이지로 이동합니다.
2. 애플리케이션 서버의 IP 주소를 클릭하거나 로드 밸런서를 사용하는 경우에는 로드 밸런서의 IP 주소를 클릭하여 앱을 실행합니다.
3. IP 주소 접두사를 **`http://`**에서 **`https://`**로 변경하고 브라우저를 새로 고쳐 SSL을 사용하여 페이지가 제대로 로드되는지 확인합니다.

사용자가 Mozilla Firefox에서 실행하도록 앱을 구성한 경우 다음과 같은 인증서 오류가 발생할 수 있습니다. `SEC_ERROR_UNKNOWN_ISSUER` 이 오류는 조직의 바이러스 백신 및 맬웨어 방지 프로그램의 인

증서 교체 기능, 일부 유형의 네트워크 트래픽 모니터링 및 필터링 소프트웨어 또는 맬웨어로 인해 발생할 수 있습니다. 이 오류를 해결하는 방법에 대한 자세한 내용은 Mozilla Firefox 지원 웹 사이트의 보안 웹 사이트에서 보안 오류 코드의 문제를 해결하는 방법을 참조하세요.

사용자 지정 계층 등 다른 모든 계층의 경우, AWS OpsWorks Stacks는 앱의 `deploy` 속성에 SSL 설정을 단순히 추가합니다. 노드 객체에서 정보를 검색하고 서버를 적절히 구성하려면 사용자 지정 레시피를 구현해야 합니다.

쿡북과 레시피

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Chef 쿡북을 사용하여 패키지 설치 및 구성, 앱 배포와 같은 작업을 처리합니다. 이 섹션에서는 Stacks에서 AWS OpsWorks 쿡북을 사용하는 방법을 설명합니다. 자세한 내용은 Chef를 참조하세요.

> **Note**
>
> AWS OpsWorks Stacks는 현재 Chef 버전 12, 11.10.4, 11.4.4 및 0.9.15.5를 지원합니다. 단, Chef 0.9.15.5는 더 이상 사용되지 않으므로 새 스택에는 사용하지 않는 것이 좋습니다. 편의를 위해, Chef 버전은 보통 메이저 및 마이너 버전 번호로만 지칭합니다. Chef 0.9 또는 11.4를 실행하는 스택은 Chef Solo를 사용하며, Chef 12 또는 11.10을 사용하는 스택은 로컬 모드 Chef Client를 사용합니다. Linux 스택의 경우 구성 관리자를 사용하여 스택을 생성할 때 사용할 Chef 버전을 지정할 수 있습니다. Windows 스택은 Chef 12.2를 사용해야 합니다. 스택을 보다 최신 Chef 버전으로 마이그레이션하기 위한 지침을 포함하여 자세한 정보는 Chef 버전 단원을 참조하세요.

주제

- 쿡북 리포지토리
- Chef 버전

- Ruby 버전
- 사용자 지정 쿡북 설치
- 사용자 지정 쿡북 업데이트
- 레시피 실행

쿡북 리포지토리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 쿡북은 온라인 리포지토리(.zip 파일과 같은 아카이브 또는 Git과 같은 소스 제어 관리자)에 저장해야 합니다. 각 스택은 사용자 지정 쿡북 리포지토리가 하나만 있을 수 있지만, 리포지토리에는 쿡북을 제한 없이 저장할 수 있습니다. 쿡북을 설치하거나 업데이트하면 AWS OpsWorks Stacks는 스택의 각 인스턴스에 있는 로컬 캐시에 전체 리포지토리를 설치합니다. 예를 들어 인스턴스가 하나 이상의 레시피를 실행해야 할 경우 로컬 캐시에 저장된 코드를 사용합니다.

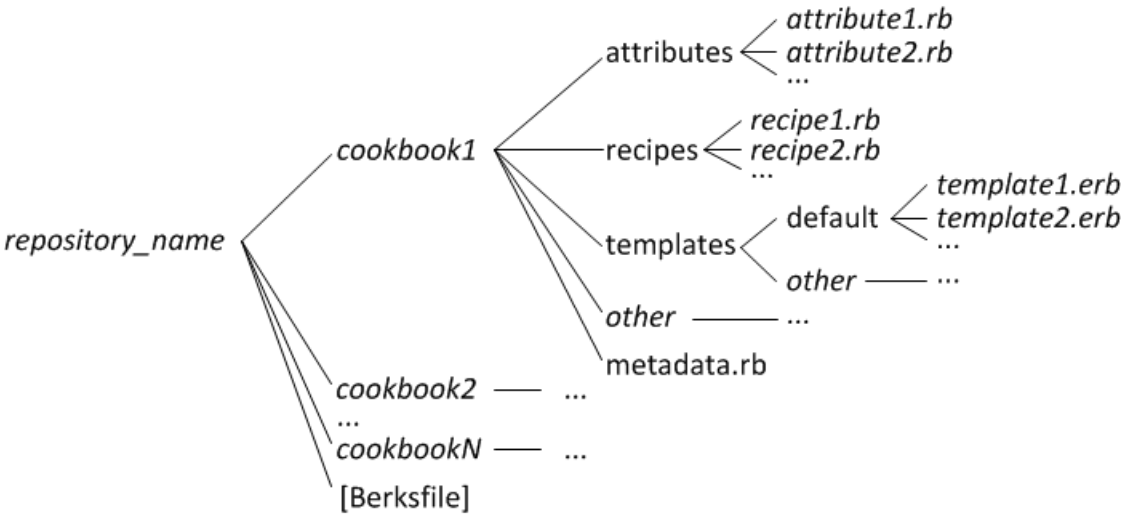
다음은 쿡북 리포지토리를 유형에 따라 구성하는 방법을 설명합니다. 그림에서 기울임꼴 텍스트는 리포지토리 또는 아카이브 이름을 포함하여 사용자 정의 디렉터리 및 파일 이름을 나타냅니다.

소스 제어 관리자

AWS OpsWorks Stacks는 다음과 같은 소스 제어 관리자를 지원합니다.

- Linux 스택 - Git 및 하위 버전
- Windows 스택 - Git

다음은 필수 디렉터리 및 파일 구조입니다.



- 쿡북 디렉터리는 모두 최상위 수준이어야 합니다.

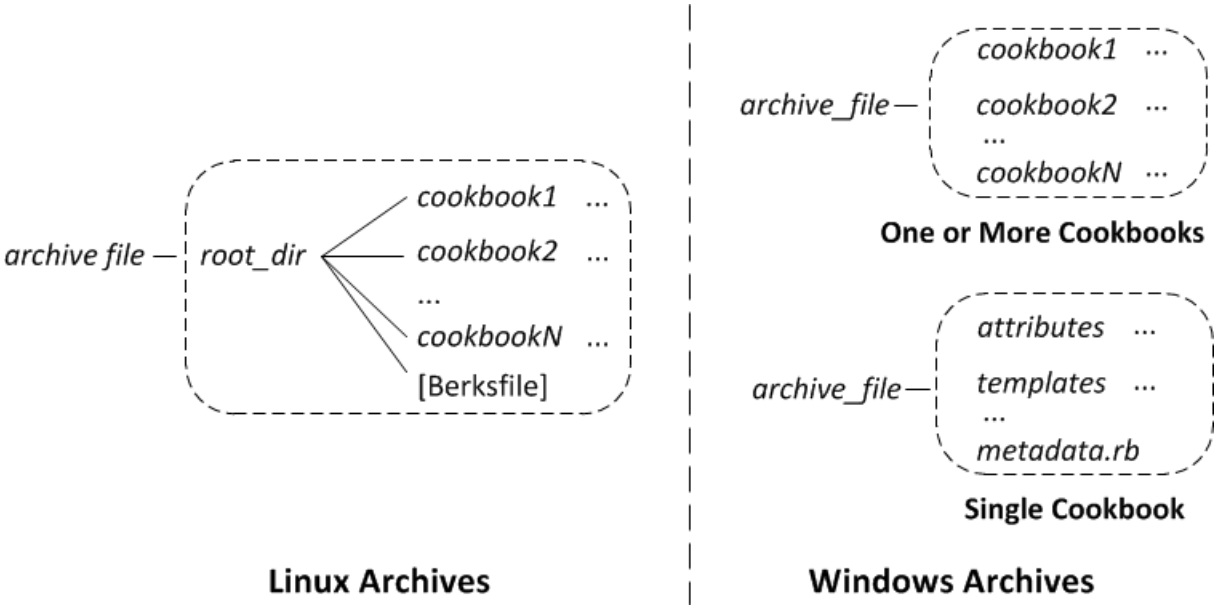
아카이브

AWS OpsWorks Stacks는 다음 아카이브를 지원합니다.

- Linux 스택 - Amazon S3 또는 웹 사이트(HTTP 아카이브)에 저장된 zip, gzip, bzip2 또는 tarball 파일.

  AWS OpsWorks Stacks는 압축되지 않은 tarball을 지원하지 않습니다.
- Windows 스택 - Amazon S3에 저장된 zip 및 tgz(gzip compressed tar) 파일.

다음은 Linux 스택 또는 Windows 스택에 따른 필수 디렉터리 및 파일 구조입니다. 쿡북 구조는 SCM 리포지토리와 동일하므로 줄임표(...)로 표시되어 있습니다.



- Linux 스택 - 쿡북 디렉터리가 루트 디렉터리에 포함되어야 합니다.
- Windows 스택 - 쿡북이 아카이브의 최상위 수준이어야 합니다.

  쿡북이 하나뿐인 경우 선택적으로 쿡북 디렉터리를 생략하고 쿡북 파일을 최상위 수준에 배치할 수 있습니다. 이 경우 AWS OpsWorks Stacks가 metadata.rb에서 쿡북 이름을 가져옵니다.

각 쿡북 디렉터리는 다음의 표준 디렉터리 및 파일을 하나 이상 포함하며(일반적으로 모두 포함) 각 디렉터리 및 파일은 표준 이름을 사용해야 합니다.

- `attributes` – 쿡북의 속성 파일.
- `recipes` – 쿡북의 레시피 파일.
- `templates` – 쿡북의 템플릿 파일.
- `##` – 정의 또는 사양과 같은 기타 파일 형식이 저장되는 선택적 사용자 정의 디렉터리.
- `metadata.rb` – 쿡북의 메타데이터.

  Chef 11.10 이상의 경우 레시피가 다른 쿡북에 기반하는 경우 쿡북의 `depends` 파일에 해당 `metadata.rb` 문을 포함시켜야 합니다. 예를 들어 쿡북에 `include_recipe anothercookbook::somerecipe` 같은 문을 사용하는 레시피가 있는 경우 쿡북의 `metadata.rb` 파일에 `depends "anothercookbook"` 행이 있어야 합니다. 자세한 정보는 About Cookbook Metadata를 참조하세요.

템플릿은 `templates` 디렉터리의 하위 디렉터리에 위치해야 하며, 이 디렉터리는 하나 이상의 하위 디렉터리를 포함합니다(선택적으로 여러 개의 하위 디렉터리를 포함할 수 있음). 각 하위 디렉터리도 하위 디렉터리를 포함할 수 있습니다.

- 템플릿은 일반적으로 `default` 하위 디렉터리를 포함하며, 이 하위 디렉터리에는 Chef가 기본적으로 사용하는 템플릿 파일이 저장됩니다.
- other는 운영 체제별 템플릿에 사용할 수 있는 선택적 하위 디렉터리를 나타냅니다.
- Chef는 File Specificity에 기술된 명명 규칙에 따라 적절한 하위 디렉터리의 템플릿을 자동으로 사용합니다. 예를 들어 Linux 및 운영 체제의 경우 각각 `amazon`amazon 또는 `ubuntu`ubuntu 하위 디렉터리에 운영 체제별 템플릿을 배치할 수 있습니다.

사용자 지정 쿡북을 처리하는 세부 절차는 사용하는 리포지토리 유형에 따라 달라집니다.

아카이브를 사용하려면

1. 이전 섹션에서 표시된 폴더 구조를 사용하여 쿡북을 구현합니다.
2. 압축된 아카이브를 생성하여 Amazon S3 버킷 또는 웹 사이트에 업로드합니다.

   쿡북을 업데이트하는 경우 새 아카이브 파일을 생성해 업로드해야 합니다. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

SCM을 사용하려면

1. 앞서 표시된 구조를 사용하여 Git 또는 하위 버전 리포지토리를 설정합니다.
2. 경우에 따라 리포지토리의 버전 관리 기능을 사용하여 여러 브랜치 또는 버전을 구현합니다.

   쿡북을 업데이트하는 경우 새 브랜치에서 업데이트할 수 있는데 OpsWorks에 새 버전을 사용하도록 지시하기만 하면 됩니다. 또한 태그가 지정된 특정 버전을 지정할 수도 있습니다. 자세한 내용은 사용자 지정 쿡북 리포지토리 지정을 참조하세요.

사용자 지정 쿡북 설치 에서는 Stacks가 스택의 인스턴스에 AWS OpsWorks 쿡북 리포지토리를 설치하도록 하는 방법을 설명합니다.

> **Important**
>
> 리포지토리의 기존 쿡북을 업데이트한 후에는 `update_cookbooks` 스택 명령을 실행하여 AWS OpsWorks Stacks에 각 온라인 인스턴스의 로컬 캐시를 업데이트하도록 지시해야 합니다. 자세한 내용은 스택 명령 실행 단원을 참조하십시오.

Chef 버전

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 여러 버전의 Chef를 지원합니다. 스택을 생성할 때 버전을 선택합니다. 그러면 AWS OpsWorks 스택은 해당 버전과 호환되는 내장 레시피 세트와 함께 스택의 모든 인스턴스에 해당 버전의 Chef를 설치합니다. 사용자 지정 레시피를 설치하는 경우 이러한 레시피가 스택의 Chef 버전과 호환되어야 합니다.

AWS OpsWorks Stacks는 현재 Linux 스택용 Chef 버전 12, 11.10, 11.4 및 0.9와 Windows 스택용 Chef 12.2(현재 Chef 12.22)를 지원합니다. 편의를 위해, Chef 버전은 보통 메이저 및 마이너 버전 번호로만 지칭합니다. Linux 스택의 경우 구성 관리자를 사용하여 스택을 생성할 때 사용할 Chef 버전을 지정할 수 있습니다. Windows 스택은 Chef 12.2를 사용해야 합니다. 스택을 보다 최신 Chef 버전으로 마이그레이션하기 위한 지침을 포함하여 자세한 정보는 Chef 버전 단원을 참조하세요. 전체 버전 정보는 AWS OpsWorks 스택 운영 체제 단원을 참조하세요.

Chef 12.2

Chef 12.2 지원은 2015년 5월에 도입되었으며, Windows 스택에서만 사용됩니다. Windows 스택의 현재 Chef 버전은 Chef 12.22입니다. Ruby 2.3.6과 함께 실행되며 로컬 모드에서 chef-client를 사용하여 chef-zero라는 로컬 인메모리 Chef 서버를 시작합니다. 이 서버가 존재하면 레시피가 Chef 검색 및 데이터 백을 사용할 수 있습니다. 이 지원은 레시피 구현: Chef 12.2 섹션에서 설명한 대로 약간의 제약이 있지만, 많은 커뮤니티 쿡북을 수정 없이 실행할 수 있습니다.

Chef 12

Chef 12 지원은 2015년 12월에 도입되었으며, Linux 스택에서만 사용됩니다. 이 버전은 Ruby 2.1.6 또는 2.2.3과 함께 실행되며, 레시피가 Chef 검색 및 데이터 백을 사용할 수 있게 해주는 로컬 모드 chef-client를 사용합니다. 자세한 내용은 AWS OpsWorks 스택 운영 체제 단원을 참조하십시오.

Chef 11.10

Chef 11.10 지원은 2014년 3월에 도입되었으며, Linux 스택에서만 사용됩니다. 이 버전은 Ruby 2.0.0과 함께 실행되며, 레시피가 Chef 검색 및 데이터 백을 사용할 수 있게 해주는 로컬 모드 chef-client를 사용합니다. 이 지원은 레시피 구현: Chef 11.10 섹션에서 설명한 대로 약간의 제약이 있지만, 많은 커뮤니티 쿡북을 수정 없이 실행할 수 있습니다. Berkshelf를 사용하여 쿡북 종속성을 관리할 수도 있습니다. 지원되는 Berkshelf 버전은 운영 체제에 따라 다릅니다. 자세한 내용은 AWS OpsWorks 스택 운영 체제 단원을 참조하십시오. Chef 11.10을 사용하는 CentOS 스택은 생성할 수 없습니다.

Chef 11.4

Chef 11.4 지원은 2013년 7월에 도입되었으며, Linux 스택에서만 사용됩니다. 이 버전은 Ruby 1.8.7과 함께 실행되며, Chef 검색 또는 데이터 백을 지원하는 않는 chef-solo를 사용합니다. Stacks에서 이러한 기능에 의존하는 커뮤니티 AWS OpsWorks 쿡북을 자주 사용할 수 있지만에 설명된

대로 수정해야 합니다새 Chef 버전으로 마이그레이션. Chef 11.4를 사용하는 CentOS 스택은 생성할 수 없습니다. Chef 11.4 스택은 미국 동부(버지니아 북부) 밖의 리전 엔드포인트에서는 지원되지 않습니다.

Chef 0.9

Chef 0.9는 Linux 스택에서만 사용되며 더 이상 지원되지 않습니다. 다음 사항에 주의하세요.

- 콘솔을 사용하여 새 Chef 0.9 스택을 생성할 수 없습니다.

  CLI 또는 API를 사용해야 합니다. 또는 다른 Chef 버전을 사용하여 스택을 생성한 후 스택 구성을 편집해야 합니다.
- Chef AWS OpsWorks 0.9 스택에는 새로운 Stacks 기능을 사용할 수 없습니다.
- 새로운 운영 체제 버전은 Chef 0.9 스택을 제한적으로만 지원합니다.

  특히 Amazon Linux 2014.09 이상은 Ruby 1.8.7에 의존하는 Rails 앱 서버 계층을 포함하는 Chef 0.9 스택을 지원하지 않습니다.
- 유럽(프랑크푸르트) 등 새로운 AWS 리전은 Chef 0.9 스택을 지원하지 않습니다.

> **Note**
>
> 새 스택에는 Chef 0.9를 사용하지 않는 것이 좋습니다. 가급적 빨리 기존 스택을 최신 Chef 버전으로 마이그레이션해야 합니다.

Stacks에서 커뮤니티 AWS OpsWorks 쿡북을 사용하려면 새 Linux 스택에 Chef 12를 지정하고 기존 Linux 스택을 Chef 12로 마이그레이션하는 것이 좋습니다. AWS OpsWorks Stacks 콘솔, API 또는 CLI를 사용하여 기존 스택을 최신 Chef 버전으로 마이그레이션할 수 있습니다. 자세한 내용은 새 Chef 버전으로 마이그레이션 단원을 참조하십시오.

주제

- Chef 12.2 Stacks용 레시피 구현
- Chef 12 스택용 레시피 구현
- Chef 11.10 스택용 레시피 구현
- Chef 11.4 스택용 레시피 구현
- 기존 Linux 스택을 새 Chef 버전으로 마이그레이션

Chef 12.2 Stacks용 레시피 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 12.2(현재 Chef 12.22)는 이 Chef 버전을 실행해야 하는 Windows 스택에서만 사용할 수 있습니다.

- 레시피는 일부 용도에서 Windows 고유 속성 및 리소스를 사용해야 합니다.

  자세한 정보는 Chef for Microsoft Windows를 참조하세요.
- Chef 실행은 Ruby 2.3.6을 사용합니다. 따라서 레시피가 새로운 Ruby 구문을 사용할 수 있습니다.
- 레시피는 Chef 검색 및 데이터 백을 사용할 수 있습니다.

  Chef 12.2 스택은 많은 커뮤니티 쿡북을 수정 없이 사용할 수 있습니다. 자세한 내용은 Chef 검색 사용 및 데이터 백 사용 섹션을 참조하세요.
- AWS OpsWorks Stacks 데이터 백 참조 및 내장 쿡북 속성 섹션에서 설명하는 스택 구성 및 배포 속성 대부분을 Windows 레시피에 사용할 수 있습니다.

  Chef 검색을 사용하여 이러한 속성 값을 가져올 수 있습니다. 예제는 Chef 검색을 사용하여 속성 값 가져오기 섹션을 참조하세요. 속성 목록은 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하세요.

Chef 12 스택용 레시피 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 12 스택은 Chef 11.10 스택에 비해 다음과 같은 장점이 있습니다.

- Chef 실행은 Ruby 2.1.6을 사용합니다. 따라서 레시피가 새로운 Ruby 구문을 사용할 수 있습니다.
- Chef 12 스택은 훨씬 더 많은 커뮤니티 쿡북을 수정 없이 사용할 수 있습니다. 방해하는 내장 쿡북이 없으므로 더 이상 내장 쿡북과 사용자 지정 쿡북 간의 이름 충돌이 없습니다.
- 더 이상 AWS OpsWorks Stacks가 사전 빌드된 패키지를 제공한 Berkshelf 버전으로 제한되지 않습니다. Berkshelf는 Chef 12의 AWS OpsWorks Stacks 인스턴스에 더 이상 설치되지 않습니다. 그 대신, 로컬 워크스테이션에서 아무런 Berkshelf 버전이든 사용할 수 있습니다.
- 이제 AWS OpsWorks Stacks가 Chef 12(Elastic Load Balancing, Amazon RDS 및 Amazon ECS)와 함께 제공하는 내장 쿡북과 사용자 지정 쿡북이 명확하게 분리됩니다. 그러므로 실패한 Chef 실행을 보다 용이하게 문제 해결할 수 있습니다.

Chef 11.10 스택용 레시피 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 11.10 스택은 Chef 11.4 스택에 비해 다음과 같은 장점이 있습니다.

- Chef 실행은 Ruby 2.0.0을 사용합니다. 따라서 레시피가 새로운 Ruby 구문을 사용할 수 있습니다.
- 레시피는 Chef 검색 및 데이터 백을 사용할 수 있습니다.

  Chef 11.10 스택은 많은 커뮤니티 쿡북을 수정 없이 사용할 수 있습니다.
- Berkshelf를 사용하여 쿡북을 관리할 수 있습니다.

  Berkshelf는 스택에서 사용자 지정 쿡북을 관리하고 커뮤니티 쿡북을 사용하는 훨씬 유연한 방법을 제공합니다.
- 쿡북은 `metadata.rb`에서 종속성을 선언해야 합니다.

  쿡북이 다른 쿡북에 의존하는 경우 해당 쿡북의 `metadata.rb` 파일에서 종속성을 포함시켜야 합니다. 예를 들어 쿡북에 `include_recipe anothercookbook::somerecipe` 같은 문을 사용하는

레시피가 있는 경우 쿡북의 `metadata.rb` 파일에 `depends "anothercookbook"` 행이 있어야 합니다.

- AWS OpsWorks Stacks는 스택에 MySQL 계층이 포함된 경우에만 스택의 인스턴스에 MySQL 클라이언트를 설치합니다.
- AWS OpsWorks Stacks는 스택에 Ganglia 계층이 포함된 경우에만 스택의 인스턴스에 Ganglia 클라이언트를 설치합니다.
- 배포가 `bundle install`을 실행하고 설치가 실패할 경우 배포도 실패합니다.

> **Important**
>
> 내장 쿡북 이름을 사용자 지정 또는 커뮤니티 쿡북에 재사용하지 마십시오. 내장 쿡북과 이름이 동일한 사용자 지정 쿡북은 실패할 수 있습니다. Chef 11.10, 11.4 및 0.9 스택에서 사용할 수 있는 내장 쿡북의 전체 목록은 opsworks-cookbooks repository on GitHub를 참조하세요. Chef 0.9 및 11.4 스택에서 성공적으로 실행되는 비 ASCII 문자를 포함하는 쿡북이 Chef 11.10 스택에서는 실패할 수 있습니다. Chef 11.10 스택은 Chef 실행을 위해 Ruby 1.8.7보다 인코딩이 훨씬 엄격한 Ruby 2.0.0을 사용하기 때문입니다. 그러한 쿡북이 Chef 11.10 스택에서 성공적으로 실행되도록 하려면 비 ASCII 문자를 사용하는 각 파일의 맨 위에 인코딩 관련 힌트를 제공하는 주석이 포함되어야 합니다. 예를 들어 UTF-8 인코딩의 경우 주석은 `# encoding: UTF-8`이 될 수 있습니다. Ruby 2.0.0 인코딩에 대한 자세한 정보는 Encoding 단원을 참조하세요.

주제

- 쿡북 설치 및 우선 순위
- Chef 검색 사용
- 데이터 백 사용
- Berkshelf 사용

쿡북 설치 및 우선 순위

Stacks AWS OpsWorks 쿡북 설치 절차는 Chef 11.10 스택과 이전 Chef 버전에서 약간 다르게 작동합니다. Chef 11.10 스택의 경우 Stacks가 내장, 사용자 지정 및 Berkshelf AWS OpsWorks 쿡북을 설치한 후 다음 순서로 공통 디렉터리에 병합합니다.

1. 내장 쿡북

2. Berkshelf 쿡북(있는 경우)
3. 사용자 지정 쿡북(있는 경우)

AWS OpsWorks Stacks가이 병합을 수행하면 레시피를 포함하여 디렉터리의 전체 콘텐츠를 복사합니다. 중복 항목이 있을 경우 다음 규칙이 적용됩니다.

- Berkshelf 쿡북의 콘텐츠가 내장 쿡북보다 우선합니다.
- 사용자 지정 쿡북의 콘텐츠가 Berkshelf 쿡북보다 우선합니다.

이 프로세스가 작동하는 방법을 예시하기 위해, 세 개의 쿡북 디렉터리 모두 `mycookbook`이라는 쿡북을 포함하고 있는 시나리오를 생각해 봅시다.

- 내장 쿡북 - `mycookbook`은 `someattributes.rb`라는 속성 파일, `sometemplate.erb`이라는 템플릿 파일 및 `somerecipe.rb`라는 이름의 레시피를 포함합니다.
- Berkshelf 쿡북 - `mycookbook`은 `sometemplate.erb` 및 `somerecipe.rb`을(를) 포함합니다.
- 사용자 지정 쿡북 - `mycookbook`은 `somerecipe.rb`을(를) 포함합니다.

병합된 쿡북은 다음 항목을 포함합니다.

- 내장 쿡북의 `someattributes.rb`
- Berkshelf 쿡북의 `sometemplate.erb`
- 사용자 지정 쿡북의 `somerecipe.rb`

> **Important**
>
> 전체 내장 쿡북을 리포지토리로 복사한 후 쿡북의 일부를 수정하여 Chef 11.10 스택을 사용자 지정하면 안 됩니다. 그러면 레시피를 포함하여 전체 내장 쿡북을 재정의합니다. Stacks가 해당 AWS OpsWorks 쿡북을 업데이트하는 경우 프라이빗 복사본을 수동으로 업데이트하지 않는 한 스택은 이러한 업데이트의 이점을 얻지 못합니다. 스택을 사용자 지정하는 방법에 대한 자세한 정보는 AWS OpsWorks 스택 사용자 지정 단원을 참조하세요.

Chef 검색 사용

레시피에서 Chef `search` 메서드를 사용하여 스택 데이터를 쿼리할 수 있습니다. Chef 서버와 동일한 구문을 사용하지만 AWS OpsWorks Stacks는 Chef 서버를 쿼리하는 대신 로컬 노드 객체에서 데이터를 가져옵니다. 이 데이터에는 다음이 포함됩니다.

- 인스턴스의 스택 구성 및 배포 속성
- 인스턴스의 내장 및 사용자 지정 쿡북 속성 파일의 속성
- Ohai가 수집한 시스템 데이터

스택 구성 및 배포 속성에는 스택의 각 온라인 인스턴스에 대한 호스트 이름 및 IP 주소와 같은 데이터를 포함하여 레시피가 일반적으로 검색을 통해 얻는 대부분의 정보가 포함됩니다. AWS OpsWorks Stacks는 각 수명 주기 이벤트에 대해 이러한 속성을 업데이트하여 현재 스택 상태를 정확하게 반영하도록 합니다. 이는 스택에서 검색 의존형 커뮤니티 레시피를 수정 없이 자주 사용할 수 있음을 의미합니다. 검색 메서드는 여전히 해당 데이터를 반환합니다. 데이터는 서버가 아니라 스택 구성 및 배포 속성으로부터 가져옵니다.

AWS OpsWorks Stacks 검색의 주요 제한 사항은 로컬 노드 객체의 데이터, 특히 스택 구성 및 배포 속성만 처리하는 것입니다. 이러한 이유 때문에 다음 형식의 데이터는 검색을 통해 가져오지 못할 수 있습니다.

- 다른 인스턴스의 로컬에서 정의된 속성

  레시피가 속성을 로컬에서 정의하는 경우 해당 정보는 AWS OpsWorks Stacks 서비스에 다시 보고되지 않으므로 검색을 사용하여 다른 인스턴스의 해당 데이터에 액세스할 수 없습니다.
- 사용자 지정 `deploy` 속성

  앱을 배포할 때 사용자 지정 JSON을 지정할 수 있습니다. 그러면 해당 배포에서 스택의 인스턴스에 해당 속성이 설치됩니다. 하지만 사용자가 선택한 인스턴스에만 배포하는 경우 속성이 이러한 인스턴스에만 설치됩니다. 사용자 지정 JSON 속성에 대한 쿼리는 다른 모든 인스턴스에서 실패합니다. 또한 사용자 지정 속성은 특정 배포에 대해서만 스택 구성 및 배포 JSON에 포함됩니다. 이들 속성은 다음 번 수명 주기 이벤트가 새로운 스택 구성 및 배포 속성 세트를 설치할 때까지만 액세스할 수 있습니다. 스택에 대해 사용자 지정 JSON을 지정할 경우 속성은 모든 수명 주기 이벤트에서 모든 인스턴스에 설치되고 언제나 검색을 통해 액세스할 수 있습니다.
- 다른 인스턴스의 Ohai 데이터

  Chef의 Ohai 도구는 인스턴스의 다양한 시스템 데이터를 가져와 노드 객체에 추가합니다. 이 데이터는 로컬에 저장되고 AWS OpsWorks Stacks 서비스로 보고되지 않습니다. 따라서 검색이 다른 인스

턴스의 Ohai 데이터에 액세스할 수 없습니다. 하지만 이 데이터 중 일부가 스택 구성 및 배포 속성에 포함될 수도 있습니다.

- 오프라인 인스턴스

  스택 구성 및 배포 속성은 온라인 인스턴스에 대한 데이터만 포함합니다.

다음 레시피 코드는 검색을 사용하여 PHP 계층 인스턴스의 프라이빗 IP 주소를 가져오는 방법을 보여줍니다.

```
appserver = search(:node, "role:php-app").first
Chef::Log.info("The private IP is '#{appserver[:private_ip]}'")
```

> **Note**
>
> AWS OpsWorks Stacks가 스택 구성 및 배포 속성을 노드 객체에 추가하면 실제로 각각 동일한 데이터를 가진 두 세트의 계층 속성이 생성됩니다. 한 세트는 `layers` AWS OpsWorks Stacks가 데이터를 저장하는 방식인 네임스페이스에 있습니다. 다른 한 세트는 `role` 네임스페이스에 위치합니다. 이것은 Chef 서버가 데이터를 저장하는 방법입니다. `role` 네임스페이스의 목적은 Chef 서버에 구현된 검색 코드가 AWS OpsWorks Stacks 인스턴스에서 실행되도록 허용하는 것입니다. Stacks 전용 코드를 작성하는 경우 이전 예제`role:php-app`에서 `layers:php-app` 또는 AWS OpsWorks 를 사용할 수 있으며 동일한 결과를 반환`search`합니다.

데이터 백 사용

레시피에서 Chef `data_bag_item` 메서드를 사용하여 데이터 백의 정보를 쿼리할 수 있습니다. Chef 서버에서와 동일한 구문을 사용하지만, AWS OpsWorks Stacks는 인스턴스의 스택 구성 및 배포 속성에서 데이터를 가져옵니다. 그러나 AWS OpsWorks Stacks는 현재 Chef 환경을 지원하지 않으므로 `node.chef_environment` 항상를 반환합니다`_default`.

사용자 지정 JSON으로 하나 이상의 속성을 `[:opsworks][:data_bags]` 속성에 추가하여 데이터 백을 생성합니다. 다음 예제는 사용자 지정 JSON에서 데이터 백을 생성하는 일반 형식을 보여줍니다.

> **Note**
>
> 데이터 백을 쿡북 리포지토리에 추가하여 생성할 수는 없습니다. 반드시 사용자 지정 JSON을 사용해야 합니다.

```
{
  "opsworks": {
    "data_bags": {
      "bag_name1": {
        "item_name1: {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      "bag_name2": {
        "item_name1": {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      ...
    }
  }
}
```

일반적으로 스택에 대해 사용자 지정 JSON을 지정합니다. 그러면 이후의 각 수명 주기 이벤트에서 사용자 지정 속성이 모든 인스턴스에 설치됩니다. 앱을 배포할 때 사용자 지정 JSON을 지정할 수도 있습니다. 하지만 이들 속성은 해당 배포에만 설치되고 선택된 인스턴스 집합에만 설치될 수 있습니다. 자세한 내용은 앱 배포 단원을 참조하십시오.

다음 사용자 지정 JSON 예제는 `myapp`이라는 데이터 백을 생성합니다. 이 데이터 백은 하나의 항목 `mysql`을 포함하며 키-값 페어가 두 개입니다.

```
{ "opsworks": {
    "data_bags": {
```

```
      "myapp": {
        "mysql": {
          "username": "default-user",
          "password": "default-pass"
        }
      }
    }
  }
}
```

레시피에서 데이터를 사용하려면 다음 코드에 나온 대로 `data_bag_item`을 호출하고 여기에 데이터 백 및 값 이름을 전달할 수 있습니다.

```
mything = data_bag_item("myapp", "mysql")
Chef::Log.info("The username is '#{mything['username']}' ")
```

데이터 백의 데이터를 수정하려면 사용자 지정 JSON을 수정하면 됩니다. 그러면 JSON이 다음 번 수명 주기 이벤트에서 스택의 인스턴스에 설치됩니다.

Berkshelf 사용

Chef 0.9 및 Chef 11.4 스택에서는 사용자 지정 쿡북 리포지토리를 하나만 설치할 수 있습니다. Chef 11.10 스택의 경우 Berkshelf를 사용하여 쿡북과 해당 종속성을 관리할 수 있습니다. 그러면 여러 리포지토리로부터 쿡북을 설치할 수 있습니다. (자세한 내용은 로컬로 쿡북 종속성 패키징 섹션을 참조하세요.) 특히 Berkshelf를 사용하면 Stacks 호환 커뮤니티 AWS OpsWorks 쿡북을 사용자 지정 쿡북 리포지토리에 복사할 필요 없이 리포지토리에서 직접 설치할 수 있습니다. 지원되는 Berkshelf 버전은 운영 체제에 따라 다릅니다. 자세한 내용은 AWS OpsWorks 스택 운영 체제 단원을 참조하십시오.

Berkshelf를 사용하려면 사용자 지정 쿡북 설치 섹션에서 설명하는 대로 명시적으로 Berkshelf를 활성화해야 합니다. 그런 다음 `Berksfile` 파일을 쿡북 리포지토리의 루프 디렉터리(설치할 쿡북을 지정)에 포함시킵니다.

Berksfile에서 외부 쿡북 소스를 지정하려면 기본 리포지토리 URL을 지정하는 파일의 맨 위에 소스 속성을 포함시킵니다. Berkshelf는 리포지토리가 명시적으로 지정되지 않은 한 소스 URL에서 쿡북을 찾습니다. 그런 다음 설치하려는 각 쿡북을 다음 형식으로 한 줄에 포함시킵니다.

```
cookbook 'cookbook_name', ['>= cookbook_version'], [cookbook_options]
```

cookbook 다음의 필드는 특정 쿡북을 지정합니다.

- *cookbook_name* – (필수) 쿡북의 이름을 지정합니다.

  다른 어떤 필드도 포함시키지 않을 경우 Berkshelf는 지정된 소스 URL로부터 쿡북을 설치합니다.

- *cookbook_version* – (선택) 쿡북 버전을 지정합니다.

  = 또는 >= 같은 접두사를 사용하여 특정 버전 또는 허용 가능한 버전 범위를 지정할 수 있습니다. 버전을 지정하지 않을 경우 Berkshelf는 최신 버전을 설치합니다.

- *cookbook_options* – (선택) 마지막 필드는 리포지토리 위치 같은 옵션을 지정하는 하나 이상의 키-값 페어를 포함하는 해시입니다.

  예를 들어 `git` 키를 포함시켜 특정 Git 리포지토리를 지정하고 `tag` 키를 포함시켜 특정 리포지토리 브랜치를 지정할 수 있습니다. 리포지토리 브랜치 지정은 일반적으로 선호하는 쿡북이 설치되도록 보장하는 최선의 방법입니다.

> **Important**
>
> Berksfile에 `metadata` 행을 포함시키고 `metadata.rb`에서 쿡북 종속성을 선언하여 쿡북을 선언하지 마십시오. 이 방법이 제대로 작동하려면 두 파일이 모두 동일한 디렉터리에 있어야 합니다. Stacks를 사용하면 Berksfile이 AWS OpsWorks 리포지토리의 루트 디렉터리에 있어야 하지만 `metadata.rb` 파일은 해당 쿡북 디렉터리에 있어야 합니다. 대신에 Berksfile에서 외부 쿡북을 명시적으로 선언해야 합니다.

다음은 쿡북을 지정하는 다양한 방법을 보여주는 Berksfile의 예제입니다. Berksfile을 생성하는 방법에 대한 자세한 정보는 Berkshelf를 참조하세요.

```
source "https://supermarket.chef.io"

cookbook 'apt'
cookbook 'bluepill', '>= 2.3.1'
cookbook 'ark', git: 'git://github.com/opscode-cookbooks/ark.git'
cookbook 'build-essential', '>= 1.4.2', git: 'git://github.com/opscode-cookbooks/build-
essential.git', tag: 'v1.4.2'
```

이 파일은 다음 쿡북을 설치합니다.

- 커뮤니티 쿡북 중 최신 버전의 `apt`

- 커뮤니티 쿡북 중 최신 버전의 `bluepill`(버전 2.3.1 이상)
- 지정된 리포지토리로부터 최신 버전의 `ark`

  이 예제의 URL은 GitHub에 있는 퍼블릭 커뮤니티 쿡북 리포지토리이지만, 프라이빗 리포지토리를 포함하여 다른 리포지토리로부터 쿡북을 설치할 수 있습니다. 자세한 정보는 Berkshelf를 참조하세요.
- 지정된 리포지토리의 v1.4.2 브랜치로부터 `build-essential` 쿡북

사용자 지정 쿡북 리포지토리에는 Berksfile 이외에 사용자 지정 쿡북이 포함될 수 있습니다. 이 경우 AWS OpsWorks Stacks는 두 쿡북 세트를 모두 설치합니다. 즉, 인스턴스에 최대 3개의 쿡북 리포지토리가 있을 수 있습니다.

- 내장 쿡북은 `/opt/aws/opsworks/current/cookbooks`에 설치됩니다.
- 사용자 지정 쿡북 리포지토리에 쿡북이 포함되어 있는 경우 이러한 쿡북은 `/opt/aws/opsworks/current/site-cookbooks`에 설치됩니다.
- Berkshelf가 활성화되어 있고 사용자 지정 쿡북 리포지토리에 Berksfile이 포함되어 있는 경우 지정된 쿡북은 `/opt/aws/opsworks/current/berkshelf-cookbooks`에 설치됩니다.

기본 제공 쿡북과 사용자 지정 쿡북은 설정 중에 각 인스턴스에 설치되며 나중에 사용자 지정 쿡북 업데이트 스택 명령을 수동으로 실행하지 않는 한 업데이트되지 않습니다. AWS OpsWorks 스택은 모든 Chef 실행`berks install`에 대해 실행되므로 다음 규칙에 따라 각 수명 주기 이벤트에 대해 Berkshelf 쿡북이 업데이트됩니다.

- 리포지토리에 새 쿡북 버전이 있을 경우 이 작업은 리포지토리로부터 쿡북을 업데이트합니다.
- 그렇지 않을 경우 이 작업은 로컬 캐시로부터 Berkshelf 쿡북을 업데이트합니다.

> **Note**
>
> 이 작업은 Berkshelf 쿡북을 덮어쓰므로 쿡북의 로컬 복사본을 수정한 경우 변경 사항을 덮어쓰게 됩니다. 자세한 정보는 Berkshelf를 참조하세요.

또한 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 Berkshelf 쿡북을 업데이트할 수도 있습니다. 이 경우 Berkshelf 쿡북 및 사용자 지정 쿡북 모두 업데이트됩니다.

Chef 11.4 스택용 레시피 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Important**
>
> 내장 쿡북 이름을 사용자 지정 또는 커뮤니티 쿡북에 재사용하지 마십시오. 내장 쿡북과 이름이 동일한 사용자 지정 쿡북은 실패할 수 있습니다. Chef 11.10, 11.4 및 0.9 스택에서 사용할 수 있는 내장 쿡북의 전체 목록은 opsworks-cookbooks repository on GitHub를 참조하세요.

Chef 11.4 스택의 기본적인 제약은 Chef 검색 또는 데이터 백을 사용할 수 없다는 점입니다. 그러나 AWS OpsWorks Stacks는 다음을 포함하여 검색으로 얻을 수 있는 많은 정보가 포함된 스택 구성 및 배포 속성을 각 인스턴스에 설치합니다.

- 콘솔로부터의 사용자 정의 데이터, 예: 호스트 또는 앱 이름
- 스택의 계층, 앱 및 인스턴스와 같은 AWS OpsWorks Stacks 서비스에서 생성된 스택 구성 데이터 및 IP 주소와 같은 각 인스턴스에 대한 세부 정보입니다.
- 사용자가 제공한 데이터를 포함하고 데이터 백과 거의 같은 용도로 사용할 수 있는 사용자 지정 JSON 속성

AWS OpsWorks Stacks는 이벤트의 Chef 실행을 시작하기 전에 각 수명 주기 이벤트의 각 인스턴스에 현재 버전의 스택 구성 및 배포 속성을 설치합니다. 데이터는 표준 `node[:attribute][:child_attribute][...]` 구문을 통해 레시피에 제공됩니다. 예를 들어 스택 구성 및 배포 속성에는 스택 이름 `node[:opsworks][:stack][:name]`이 포함됩니다.

내장 레시피 중 하나에서 발췌된 다음 코드는 스택 이름을 가져오고 이를 사용하여 구성 파일을 생성합니다.

```
template '/etc/ganglia/gmetad.conf' do
  source 'gmetad.conf.erb'
```

```
  mode '0644'
  variables :stack_name => node[:opsworks][:stack][:name]
  notifies :restart, "service[gmetad]"
end
```

스택 구성 및 배포 속성 값이 여러 속성을 포함하는 경우가 많습니다. 필요한 정보를 얻으려면 이들 속성을 반복적으로 처리해야 합니다. 아래 예제는 스택 구성 및 배포 속성에서 발췌된 코드로, 편의상 JSON 객체로 표현되어 있습니다. 여기에는 최상위 속성 deploy이 포함되어 있습니다. 이 속성은 스택의 각 앱의 속성과 앱의 짧은 이름을 제공합니다.

```
{
  ...
  "deploy": {
    "app1_shortname": {
      "document_root": "app1_root",
      "deploy_to": "deploy_directory",
      "application_type": "php",
      ...
    },
    "app2_shortname": {
      "document_root": "app2_root",
      ...
    }
  },
  ...
}
```

각 앱 속성은 앱의 특징을 나타내는 속성 세트를 포함합니다. 예를 들어 deploy_to 속성은 앱의 배포 디렉터리를 나타냅니다. 다음 코드는 각 앱의 배포 디렉터리에 대해 사용자, 그룹 및 경로를 설정합니다.

```
node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end
  ...
end
```

스택 구성 및 배포 속성에 대한 자세한 정보는 AWS OpsWorks 스택 사용자 지정 단원을 참조하세요. 배포 디렉터리에 대한 자세한 정보는 Deploy 레시피 단원을 참조하세요.

Chef 11.4 스택은 데이터 백을 지원하지 않습니다. 하지만 사용자가 사용자 지정 JSON을 지정하여 스택 구성 및 배포 속성에 임의의 데이터를 추가할 수 있습니다. 그러면 레시피가 표준 Chef 노드 구문을 사용하여 데이터에 액세스할 수 있습니다. 자세한 내용은 사용자 지정 JSON 사용 단원을 참조하십시오.

암호화된 데이터 백의 기능이 필요할 경우 한 가지 옵션은 민감한 속성을 프라이빗 Amazon S3 버킷 같은 안전한 위치에 저장하는 것입니다. 그러면 레시피가 모든 AWS OpsWorks Stacks 인스턴스에 설치된 AWS Ruby SDK를 사용하여 버킷에서 데이터를 다운로드할 수 있습니다.

> **Note**
>
> 각 AWS OpsWorks Stacks 인스턴스에는 인스턴스 프로파일이 있습니다. 연결된 IAM 역할은 인스턴스에서 실행되는 애플리케이션이 어떤 AWS 리소스에 액세스할 수 있는지를 지정합니다. 레시피가 Amazon S3 버킷에 액세스하려면 역할의 정책이 다음과 같은 문장을 포함해야 합니다. 이 문장은 지정된 버킷에서 파일을 검색할 수 있는 권한을 부여합니다.
>
> ```
> "Action": ["s3:GetObject"],
> "Effect": "Allow",
> "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
> ```
>
> 인스턴스 프로파일에 대한 자세한 정보는 EC2 인스턴스에서 실행되는 앱에 대한 권한 지정 단원을 참조하세요.

기존 Linux 스택을 새 Chef 버전으로 마이그레이션

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 콘솔, API 또는 CLI를 사용하여 Linux 스택을 최신 Chef 버전으로 마이그레이션할 수 있습니다. 하지만 새로운 버전과 호환되도록 레시피를 수정해야 할 수 있습니다. 스택 마이그레이션을 준비할 때 다음 사항을 고려해야 합니다.

- 스택을 편집하거나 복제하여 AWS OpsWorks Stacks 스택 버전을 Chef 11에서 Chef 12로 변경할 수 없습니다. 이 섹션에서 설명하는 절차를 사용하여 Chef 메이저 버전 업그레이드를 수행할 수는 없습니다. Chef 11.10에서 Chef 12로 전환에 대한 자세한 정보는 레시피 구현: Chef 12 단원을 참조하세요.
- 한 Chef 버전에서 다른 버전으로 전환하기 위해서는 많은 변경이 필요하며, 이 중에는 호환성에 영향을 미치는 변경도 있습니다.

  Chef 0.9에서 Chef 11.4로 전환에 대한 자세한 정보는 새 Chef 버전으로 마이그레이션 단원을 참조하세요. Chef 11.4에서 Chef 11.10으로 전환에 대한 자세한 정보는 레시피 구현: Chef 11.10 단원을 참조하세요. Chef 11.10에서 Chef 12로 전환에 대한 자세한 정보는 레시피 구현: Chef 12 단원을 참조하세요.
- Chef 실행은 Chef 0.9 및 Chef 11.4 스택(Ruby 1.8.7), Chef 11.10 스택(Ruby 2.0.0) 및 Chef 12 스택(Ruby 2.1.6)에서 다른 Ruby 버전을 사용합니다.

  자세한 내용은 Ruby 버전 단원을 참조하십시오.
- Chef 11.10 스택은 Chef 0.9 또는 Chef 11.4 스택과 다른 방식으로 쿡북 설치를 처리합니다.

  이러한 차이 때문에 사용자 지정 쿡북을 사용하는 스택을 Chef 11.10으로 마이그레이션할 때 문제가 발생할 수도 있습니다. 자세한 내용은 쿡북 설치 및 우선 순위 단원을 참조하십시오.

다음은 Chef 스택을 더 새로운 Chef 버전으로 마이그레이션하기 위한 권장 지침입니다.

최신 Chef 버전으로 스택을 마이그레이션하려면

1. 프로덕션 스택을 복제합니다. [스택 복제] 페이지에서 [고급>>]을 클릭하여 [구성 관리] 섹션을 표시하고 [Chef 버전]을 다음으로 높은 버전으로 변경합니다.

   > **Note**
   >
   > Chef 0.9 스택에서 시작한 경우에는 Chef 11.10으로 직접 업그레이드할 수 없으므로 먼저 Chef 11.4로 업그레이드해야 합니다. 레시피를 테스트하기 전에 스택을 Chef 11.10으로 마이그레이션하려는 경우 업데이트가 실행될 때까지 20분 동안 기다린 다음 스택을 11.4에서 11.10으로 업그레이드하세요.

2. 계층에 인스턴스를 추가하고 테스팅 또는 스테이징 시스템에서 복제된 스택의 애플리케이션 및 쿡북을 테스트합니다. 자세한 정보는 All about Chef ...를 참조하세요.
3. 테스트 결과에 만족하면 다음 중 하나를 수행하세요.
   - 원하는 Chef 버전인 경우 복제된 스택을 프로덕션 스택으로 사용하거나 프로덕션 스택에서 Chef 버전을 재설정할 수 있습니다.
   - 2단계에 걸쳐 Chef 0.9 스택을 Chef 11.10으로 마이그레이션하는 경우 해당 프로세스를 반복해 Chef 11.4에서 Chef 11.10으로 스택을 마이그레이션합니다.

> **Note**
>
> 레시피를 테스트할 때 SSH를 사용하여 인스턴스에 연결한 다음, 인스턴스 에이전트 CLI run_command 명령을 사용하여 다양한 수명 주기 이벤트와 연결된 레시피를 실행할 수 있습니다. 에이전트 CLI는 설정이 실패하고 인스턴스가 온라인 상태에 도달하지 않더라도 사용할 수 있기 때문에 설정 레시피를 테스트할 때 특히 유용합니다. 또한 설정 스택 명령을 사용하여 설정 레시피를 재실행할 수도 있습니다. 하지만 이 명령은 설정이 성공하고 인스턴스가 온라인 상태가 된 경우에만 사용할 수 있습니다.

실행 중 스택을 새 Chef 버전으로 업데이트할 수 있습니다.

실행 중인 스택을 새 Chef 버전으로 업데이트하려면

1. 스택을 편집하여 Chef 버전 스택 설정을 변경합니다.
2. 새 설정을 저장하고 AWS OpsWorks Stacks가 인스턴스를 업데이트할 때까지 기다립니다. 일반적으로 15~20분이 걸립니다.

> **Important**
>
> AWS OpsWorks Stacks는 Chef 버전 업데이트를 수명 주기 이벤트와 동기화하지 않습니다. 프로덕션 스택에서 Chef 버전을 업데이트하려는 경우 다음 번 수명 주기 이벤트 이전에 업데이트가 완료되도록 주의해야 합니다. 이벤트(일반적으로 Deploy 또는 Configure 이벤트)가 발생하면 인스턴스 에이전트는 버전 업데이트 완료 여부에 관계없이 사용자 지정 쿡북을 업데이트하고 이벤트에 할당된 레시피를 실행합니다. 언제 버전 업데이트가 완료되는지 확인할 수 있는 직접적 방법은 없지만, 배포 로그에 Chef 버전이 기록됩니다.

Ruby 버전

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Linux 스택의 모든 인스턴스에는 Ruby가 설치되어 있습니다. AWS OpsWorks Stacks는 각 인스턴스에 Ruby 패키지를 설치합니다.이 패키지는 Chef 레시피와 인스턴스 에이전트를 실행하는 데 사용됩니다. AWS OpsWorks Stacks는 스택이 실행 중인 Chef 버전을 기반으로 Ruby 버전을 결정합니다. 이 버전을 수정하려 하지 마십시오. 그러면 인스턴스 에이전트가 비활성화될 수 있습니다.

AWS OpsWorks Stacks는 Windows 스택에 애플리케이션 Ruby 실행 파일을 설치하지 않습니다. Chef 12.2 클라이언트에는 Ruby 2.0.0 p451이 제공되지만, Ruby 실행 파일이 인스턴스의 PATH 환경 변수에 추가되지 않습니다. 이 실행 파일로 Ruby 코드를 실행하려는 경우, Windows 드라이브의 `\opscode\chef\embedded\bin\ruby.exe`에 이 파일이 있습니다.

다음 표에는 AWS OpsWorks Stacks Ruby 버전이 요약되어 있습니다. 사용 가능한 애플리케이션 Ruby 버전은 인스턴스의 운영 체제에 따라서도 달라집니다. 사용 가능한 패치 버전을 비롯한 자세한 정보는 AWS OpsWorks 스택 운영 체제 단원을 참조하세요.

Chef 버전	Chef Ruby 버전	사용 가능한 애플리케이션 Ruby 버전
0.9(c)	1.8.7	1.8.7(a), 1.9.3(e), 2.0.0
11.4(c)	1.8.7	1.8.7(a), 1.9.3(e), 2.0.0, 2.1, 2.2.0, 2.3
11.10	2.0.0-p481	1.9.3(c, e), 2.0.0, 2.1, 2.2.0, 2.3, 2.6.1
12(b)	2.1.6, 2.2.3	없음
12.22(d)	2.3.6	없음

(a) Amazon Linux 2014.09 이상, Red Hat Enterprise Linux(RHEL) 또는 Ubuntu 14.04 LTS에서는 사용할 수 없음

(b) Linux 스택에서만 사용 가능

(c) RHEL에서는 사용할 수 없음

(d) Windows 스택에서만 사용 가능 메이저 버전은 12.2입니다. 현재 마이너 버전은 12.22입니다.

(e) 사용이 완전히 중단되어 지원이 종료되었습니다.

설치 위치는 Chef 버전에 따라 다릅니다.

- 애플리케이션은 모든 Chef 버전에서 `/usr/local/bin/ruby` 실행 파일을 사용합니다.
- Chef 0.9 및 11.4에서 인스턴스 에이전트 및 Chef 레시피는 `/usr/bin/ruby` 실행 파일을 사용합니다.
- Chef 11.10에서 인스턴스 에이전트 및 Chef 레시피는 `/opt/aws/opsworks/local/bin/ruby` 실행 파일을 사용합니다.

사용자 지정 쿡북 설치

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

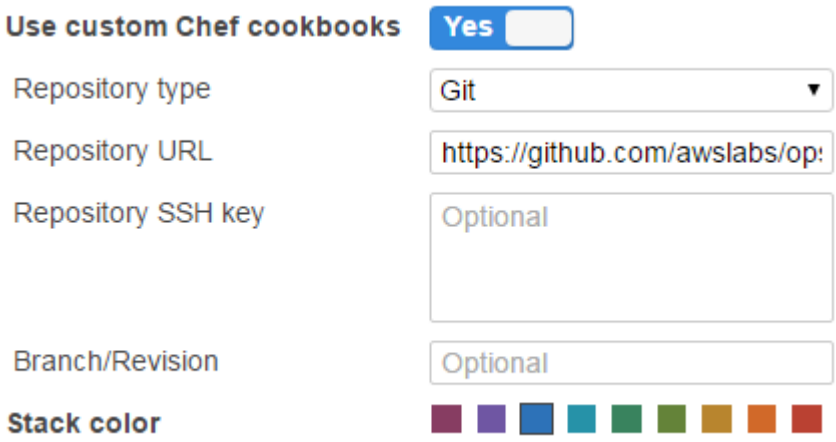
스택이 사용자 지정 쿡북을 설치하고 사용하도록 하려면 사용자 지정 쿡북을 활성화하도록 스택을 구성해야 합니다(아직 구성하지 않은 경우). 그런 다음 리포지토리 URL 및 암호 등 모든 관련 정보를 제공해야 합니다.

Important

사용자 지정 쿡북을 지원하도록 스택을 구성한 후 AWS OpsWorks Stacks는 시작 시 모든 새 인스턴스에 쿡북을 자동으로 설치합니다. 그러나 사용자 지정 AWS OpsWorks 쿡북 스택 업데이트 명령을 실행하여 기존 인스턴스에 새 쿡북 또는 업데이트된 쿡북을 설치하도록 Stacks에 명시적으로 지시해야 합니다. 자세한 내용은 사용자 지정 쿡북 업데이트 단원을 참조하십시오. 스택에서 사용자 지정 Chef 쿡북 사용을 활성화하기 전에, 실행 중인 사용자 지정 및 커뮤니티 쿡북이 스택에서 사용되는 Chef 버전을 지원하는지 확인해야 합니다.

사용자 지정 쿡북에 맞게 스택을 구성하려면

1. 스택 페이지에서 [스택 설정]를 클릭하여 스택의 [설정] 페이지를 표시하고 [편집]을 클릭하여 설정을 편집합니다.
2. [사용자 지정 Chef 쿡북 사용]을 [예]로 전환합니다.



3. 사용자 지정 쿡북을 구성합니다.

다 마치면 [저장]을 클릭하여 업데이트된 스택을 저장합니다.

사용자 지정 쿡북 리포지토리 지정

Linux 스택은 다음 리포지토리 유형의 사용자 지정 쿡북을 설치할 수 있습니다.

- HTTP 또는 Amazon S3 아카이브.

  아카이브는 퍼블릭이거나 프라이빗일 수 있지만 프라이빗 아카이브로는 일반적으로 Amazon S3가 선호됩니다.
- Git 및 하위 버전 리포지토리는 소스 제어 및 다중 버전 기능을 제공합니다.

Windows 스택은 Amazon S3 아카이브와 Git 리포지토리의 사용자 지정 쿡북을 설치할 수 있습니다.

모든 리포지토리 유형에는 다음과 같은 필수 필드가 있습니다.

- 리포지토리 유형 - 리포지토리 유형
- 리포지토리 URL - 리포지토리 URL

AWS OpsWorks Stacks는 GitHub 또는 Bitbucket과 같이 공개적으로 호스팅되는 Git 리포지토리 사이트와 비공개로 호스팅되는 Git 서버를 지원합니다. https://bitbucket.org Git 리포지토리의 경우, 리포지토리가 퍼블릭인지 프라이빗인지에 따라 다음 URL 형식 중 하나를 사용해야 합니다. Git 하위 모듈의 경우에도 같은 URL 지침을 따르십시오.

퍼블릭 Git 리포지토리의 경우, HTTPS 또는 Git 읽기 전용 프로토콜을 사용하세요.

- Git 읽기 전용 – `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`.
- HTTPS – `https://github.com/amazonwebservices/opsworks-example-cookbooks.git`.

프라이빗 Git 리포지토리의 경우, 다음 예제에 나온 것처럼 SSH 읽기/쓰기 형식을 사용해야 합니다.

- Github 리포지토리 – `git@github.com:`*`project`*`/`*`repository`*.
- Git 서버의 리포지토리 – *`user`*`@`*`server`*`:`*`project`*`/`*`repository`*

나머지 설정은 리포지토리 유형에 따라 다르며, 다음 섹션에 설명되어 있습니다.

HTTP 아카이브

리포지토리 유형으로 HTTP 아카이브를 선택하면 두 가지 추가 설정이 표시되며, 아카이브가 비밀번호로 보호되는 경우 이 설정을 완료해야 합니다.

- 사용자 이름—사용자 이름입니다
- 암호–사용자 암호

Amazon S3 아카이브

리포지토리 유형으로 S3 아카이브를 선택하면 다음과 같은 추가 선택적 설정이 표시됩니다. AWS OpsWorks Stacks는 Stacks API를 사용하든 콘솔을 사용하든 Amazon EC2 역할(호스트 운영 체제 관리자 인증)을 사용하여 AWS OpsWorks 리포지토리에 액세스할 수 있습니다.

- 액세스 키 ID -와 같은 AWS 액세스 키 ID입니다AKIAIOSFODNN7EXAMPLE.
- 보안 액세스 키 -와 같은 해당 AWS 보안 액세스 키입니다wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY.

Git 리포지토리

소스 제어에서 Git을 선택하면 다음과 같은 추가 옵션 설정이 표시됩니다.

[리포지토리 SSH 키]

프라이빗 Git 리포지토리에 액세스하려면 배포 SSH 키를 지정해야 합니다. Git 하위 모듈의 경우, 지정된 키는 이러한 하위 모듈에 액세스할 권한이 있어야 합니다. 자세한 내용은 Git 리포지토리 SSH 키 사용 단원을 참조하십시오.

> **Important**
>
> SSH 키 배포에는 암호가 필요하지 않습니다. AWS OpsWorks Stacks는 암호를 전달할 방법이 없습니다.

브랜치/개정

리포지토리에 여러 브랜치가 있는 경우 AWS OpsWorks Stacks는 기본적으로 마스터 브랜치를 다운로드합니다. 특정 브랜치를 지정하려면 브랜치 이름, SHA1 해시 또는 태그 이름을 입력하세요. 특정 커밋을 지정하려면 완전한 40-hexdigit 커밋 ID를 입력합니다.

하위 버전 리포지토리

소스 제어에서 하위 버전을 선택하면 다음과 같은 추가 설정이 표시됩니다.

- 사용자 이름 - 사용자 이름(프라이빗 리포지토리).
- 암호 - 사용자 암호(프라이빗 리포지토리).
- 리전 - [선택 사항] 여러 개정이 있는 경우, 개정 이름.

브랜치 또는 태그를 지정하려면 리포지토리 URL을 수정해야 합니다(예: **`http://repository_domain/repos/myapp/branches/my-apps-branch`** 또는 **`http://repository_domain_name/repos/calc/myapp/my-apps-tag`**).

사용자 지정 쿡북 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Stacks에 사용자 지정 AWS OpsWorks 쿡북을 제공하면 기본 제공 Setup 레시피는 새로 시작된 각 인스턴스에 로컬 캐시를 생성하고 쿡북을 캐시에 다운로드합니다. 그러면 AWS OpsWorks Stacks는 리포지토리가 아닌 캐시에서 레시피를 실행합니다. 리포지토리에서 사용자 지정 쿡북을 수정하는 경우 업데이트된 쿡북이 인스턴스의 로컬 캐시에 설치되어 있는지 확인해야 합니다. AWS OpsWorks Stacks는 최신 쿡북이 시작되면 새 인스턴스에 자동으로 배포합니다. 하지만 기존 인스턴스에서는 상황이 다릅니다.

- 업데이트된 사용자 지정 쿡북은 수동으로 온라인 인스턴스에 배포해야 합니다.
- 로드 기반 인스턴스와 시간 기반 인스턴스를 비롯하여 오프라인 인스턴스 스토어 지원 인스턴스에는 업데이트된 사용자 지정 쿡북을 배포할 필요가 없습니다.

  AWS OpsWorks Stacks는 인스턴스가 다시 시작될 때 현재 쿡북을 자동으로 배포합니다.
- 로드 기반 또는 시간 기반이 아니라 오프라인 EBS 지원 24/7 인스턴스를 시작해야 합니다.
- 오프라인 EBS 지원 로드 기반 및 시간 기반 인스턴스는 사용자가 시작할 수 없으므로, 가장 간단한 방법은 오프라인 인스턴스를 삭제하고 새 인스턴스를 추가해 대체하는 것입니다.

  이제 새 인스턴스이기 때문에 AWS OpsWorks Stacks는 인스턴스가 시작될 때 현재 사용자 지정 쿡북을 자동으로 배포합니다.

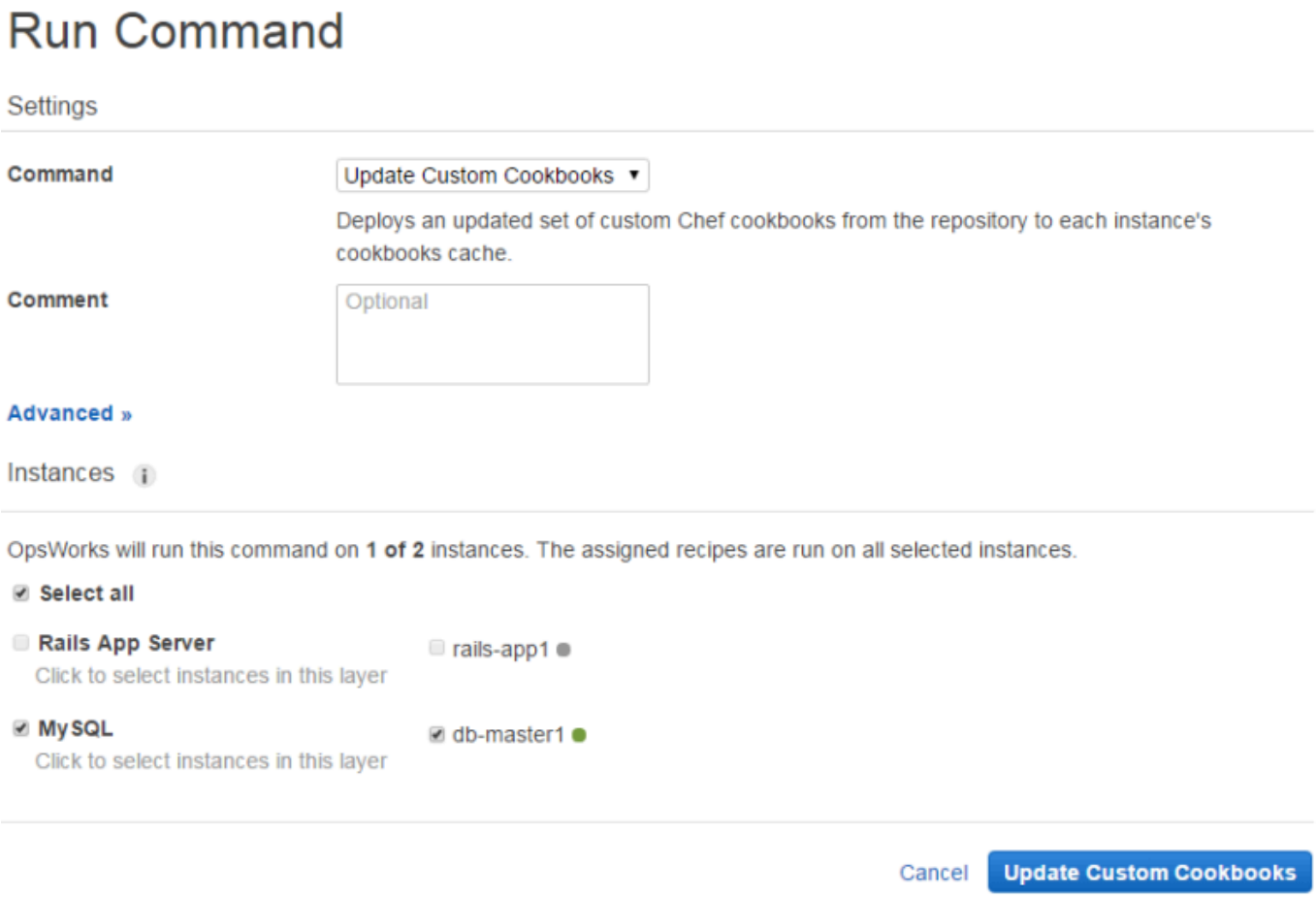
사용자 지정 쿡북을 수동으로 업데이트하려면

1. 수정된 쿡북으로 리포지토리를 업데이트합니다. AWS OpsWorks Stacks는 쿡북을 처음 설치할 때 제공한 캐시 URL을 사용하므로 쿡북 루트 파일 이름, 리포지토리 위치 및 액세스 권한이 변경되지 않아야 합니다.

   - Amazon S3 또는 HTTP 리포지토리의 경우 원래 .zip 파일을 같은 이름의 새 .zip 파일로 바꿉니다.

- Git 또는 하위 버전 리포지토리의 경우, 스택 설정을 편집하여 [분기/개정] 필드를 새 버전으로 변경합니다.

2. 스택 페이지에서 [명령 실행]을 클릭하고 [사용자 지정 쿡북 업데이트] 명령을 선택합니다.



3. 필요한 경우 설명을 추가합니다.
4. 선택적으로 명령에 대한 사용자 지정 JSON 객체를 지정하여 AWS OpsWorks Stacks가 인스턴스에 설치하는 스택 구성 및 배포 속성에 사용자 지정 속성을 추가합니다. 자세한 내용은 사용자 지정 JSON 사용 및 속성 재정의 섹션을 참조하세요.
5. 기본적으로 AWS OpsWorks Stacks는 모든 인스턴스에서 쿡북을 업데이트합니다. 업데이트할 인스턴스를 지정하려면 페이지 끝에 있는 목록에서 해당 인스턴스를 선택합니다. 계층의 인스턴스를 모두 선택하려면 왼쪽 열에서 해당 계층 확인란을 선택합니다.
6. 사용자 지정 쿡북 업데이트를 클릭하여 업데이트된 쿡북을 설치합니다. AWS OpsWorks 스택은 지정된 인스턴스에서 캐시된 사용자 지정 쿡북을 삭제하고 리포지토리에서 새 쿡북을 설치합니다.

> **Note**
>
> 이 절차는 캐시에 이전 버전의 쿡북이 있는 기존 인스턴스에만 필요합니다. 이후에 인스턴스를 계층에 추가하면 AWS OpsWorks Stacks는 현재 리포지토리에 있는 쿡북을 배포하여 자동으로 최신 버전을 가져옵니다.

레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피를 두 가지 방법으로 실행할 수 있습니다.

- 적절한 계층 수명 주기 이벤트에 할당하여 자동으로.
- 레시피 실행 스택 명령을 실행하거나 에이전트 CLI를 사용하여 수동으로.

주제

- AWS OpsWorks Stacks 수명 주기 이벤트
- 자동으로 레시피 실행
- 수동으로 레시피 실행

AWS OpsWorks Stacks 수명 주기 이벤트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

각 계층마다 5개 수명 주기 이벤트가 있으며, 각 이벤트에는 레시피 세트가 연결됩니다. 연결되는 레시피는 계층마다 다릅니다. 계층의 인스턴스에서 이벤트가 발생하면 AWS OpsWorks Stacks는 적절한 레시피 세트를 자동으로 실행합니다. 이러한 이벤트에 대한 사용자 지정 응답을 제공하려면 사용자 지정 레시피를 구현하고 각 계층의 적절한 이벤트에 할당합니다. AWS OpsWorks Stacks는 이벤트의 내장 레시피 이후에 해당 레시피를 실행합니다.

Setup

이 이벤트는 시작된 인스턴스가 부팅을 완료하면 발생합니다. Setup stack 명령을 사용하여 Setup 이벤트를 수동으로 트리거할 수도 있습니다. AWS OpsWorks Stacks는 계층에 따라 인스턴스를 설정하는 레시피를 실행합니다. 예를 들어 인스턴스가 Rails 앱 서버 계층에 속하는 경우 Setup 레시피가 Apache, Ruby Enterprise Edition, Passenger 및 Ruby on Rails를 설치합니다.

> **Note**
>
> [설정] 이벤트는 인스턴스를 서비스에서 제외시킵니다. 설정 수명 주기 이벤트가 실행될 때 인스턴스가 온라인 상태가 아니므로 설정 이벤트를 실행하는 인스턴스는 로드 밸런서에서 제거됩니다.

Configure

이 이벤트는 다음과 같은 경우에 스택의 모든 인스턴스에서 발생합니다.

- 인스턴스가 온라인 상태로 전환하거나 온라인 상태를 벗어나는 경우
- 탄력적 IP 주소를 인스턴스에 연결하거나 인스턴스에서 연결 해제하는 경우.
- Elastic Load Balancing 로드 밸런서를 계층에 연결하거나 계층에서 분리하는 경우.

예를 들어 스택에 인스턴스 A, B, C가 있고 새 인스턴스 D를 시작한다고 가정해 보겠습니다. D가 설정 레시피 실행을 완료한 후 AWS OpsWorks Stacks는 A, B, C, D에서 Configure 이벤트를 트리거합니다. 이후에 A를 중지하면 AWS OpsWorks Stacks는 B, C, D에서 Configure 이벤트를 트리거합니다. AWS OpsWorks Stacks는 각 계층의 Configure 레시피를 실행하여 Configure 이벤트에 응답합니다. 그러면 인스턴스의 구성이 업데이트되어 현재 온라인 인스턴스 세트를 반영합니다. Configure 이벤트가 구성 파일을 재생성하기에 아주 좋은 시점입니다. 예를 들어 HAProxy Configure 레시피가 온라인 애플리케이션 서버 인스턴스 집합의 변경을 수용하도록 로드 밸런서를 재구성합니다.

Configure 스택 명령을 사용하여 수동으로 Configure 이벤트를 트리거할 수도 있습니다.

Deploy

이 이벤트는 일반적으로 애플리케이션 서버 인스턴스 집합에 애플리케이션을 배포하기 위해 Deploy 명령을 실행할 때 발생합니다. 인스턴스는 애플리케이션 및 관련 파일을 리포지토리에서 계층의 인스턴스로 배포하는 레시피를 실행합니다. 예를 들어 Rails Application Server 인스턴스의 경우 Deploy 레시피가 지정된 Ruby 애플리케이션을 체크아웃하고 Phusion Passenger에게 재로드하도록 지시합니다. Deploy를 다른 인스턴스에서도 실행할 수 있습니다. 그러면 예를 들어 다른 인스턴스가 새로 배포된 앱을 수용하도록 구성을 업데이트할 수 있습니다.

> **Note**
>
> 설정은 Deploy를 포함합니다. 설정이 완료되면 Deploy 레시피를 실행합니다.

Undeploy

이 이벤트는 앱을 삭제하거나 Undeploy 명령을 실행하여 애플리케이션 서버 인스턴스 집합에 앱을 제거할 때 발생합니다. 지정된 인스턴스가 레시피를 실행하여 모든 애플리케이션 버전을 제거하고 필요한 정리 작업을 수행합니다.

Shutdown

이 이벤트는 AWS OpsWorks Stacks에 인스턴스를 종료하도록 지시한 후 연결된 Amazon EC2 인스턴스가 실제로 종료되기 전에 발생합니다. AWS OpsWorks Stacks는 레시피를 실행하여 서비스 종료와 같은 정리 작업을 수행합니다.

Elastic Load Balancing 로드 밸런서를 계층에 연결하고 연결 드레이닝에 대한 지원을 활성화한 경우 AWS OpsWorks Stacks는 연결 드레이닝이 완료될 때까지 기다렸다가 Shutdown 이벤트를 트리거합니다.

Shutdown 이벤트를 트리거한 후 AWS OpsWorks Stacks는 Shutdown 레시피가 작업을 수행하는 데 지정된 시간을 허용한 다음 Amazon EC2 인스턴스를 중지하거나 종료합니다. Shutdown의 기본 제한 시간 값은 120초입니다. Shutdown 레시피가 더 오랜 시간을 필요로 하는 경우 계층 구성을 편집하여 제한 시간 값을 변경할 수 있습니다. 인스턴스 종료(Shutdown)에 대한 자세한 정보는 인스턴스 중지 단원을 참조하세요.

> **Note**
>
> 인스턴스 재부팅은 수명 주기 이벤트를 트리거하지 않습니다.

Deploy 및 Undeploy 앱 명령에 대한 자세한 설명은 앱 배포 단원을 참조하세요.

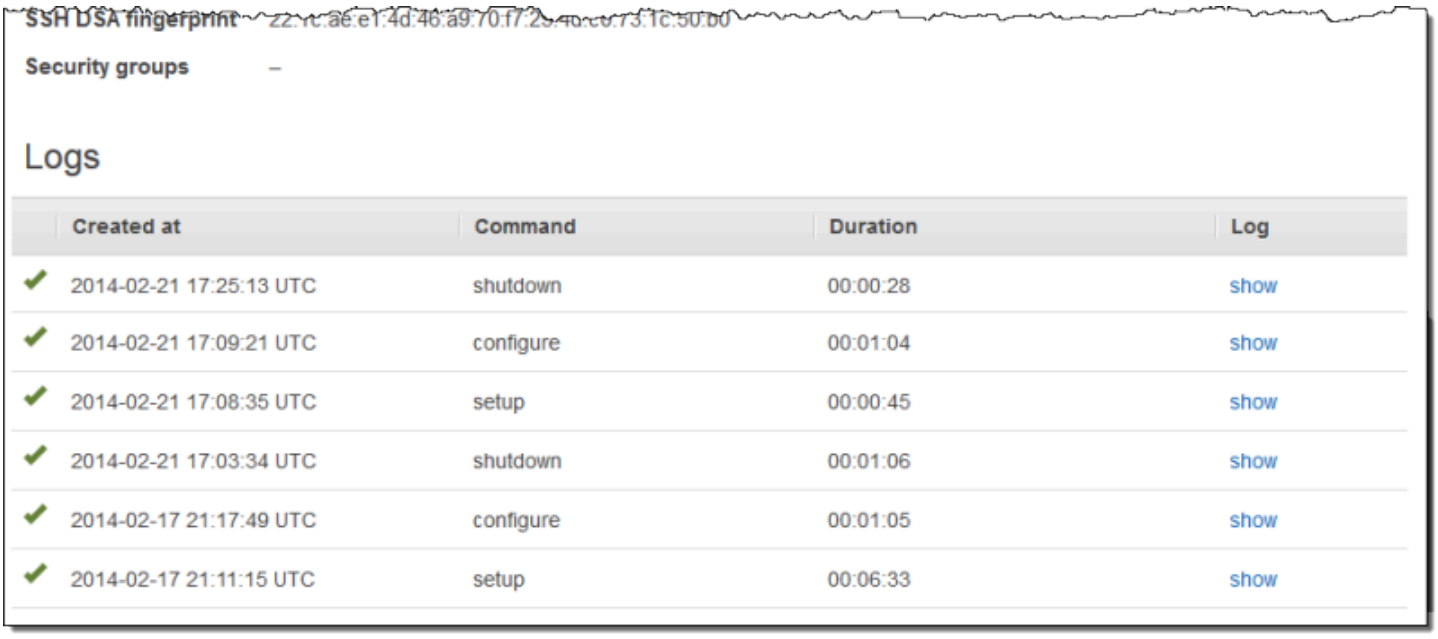
시작된 인스턴스가 부팅을 완료한 후 나머지 시작 시퀀스는 다음과 같습니다.

1. AWS OpsWorks Stacks는 인스턴스의 기본 제공 Setup 레시피를 실행한 다음 사용자 지정 Setup 레시피를 실행합니다.
2. AWS OpsWorks Stacks는 인스턴스의 기본 제공 Deploy 레시피를 실행한 다음 사용자 지정 Deploy 레시피를 실행합니다.

   이제 인스턴스는 온라인 상태입니다.
3. AWS OpsWorks Stacks는 새로 시작된 인스턴스를 포함하여 스택의 모든 인스턴스에서 Configure 이벤트를 트리거합니다.

   AWS OpsWorks Stacks는 인스턴스의 기본 제공 Configure 레시피를 실행한 다음 사용자 지정 Configure 레시피를 실행합니다.

> **Note**
>
> 특정 인스턴스에서 발생한 수명 주기 이벤트를 보려면 [인스턴스] 페이지로 이동하고 인스턴스의 이름을 클릭하여 세부 정보 페이지를 엽니다. 이벤트 목록은 페이지 하단의 [로그] 섹션에 있습니다. 로그 열에서 표시를 클릭하여 Chef 로그에서 이벤트를 검사할 수 있습니다. 로그는 실행된 레시피를 포함하여 이벤트가 처리된 방법에 대한 상세한 정보를 제공합니다. Chef 로그를 해석하는 방법에 대한 자세한 정보는 Chef 로그 단원을 참조하세요.



각 수명 주기 이벤트에 대해 AWS OpsWorks Stacks는 현재 스택 상태와 이벤트의 경우 배포에 대한 정보가 포함된 스택 구성 및 배포 속성 세트를 각 인스턴스에 설치합니다. Deploy 속성은 사용 가능한 인스턴스, 해당 IP 주소 등에 대한 정보를 포함합니다. 자세한 내용은 스택 구성 및 배포 속성 단원을 참조하십시오.

> **Note**
>
> 다수의 인스턴스를 동시에 시작 또는 중지하면 다수의 Configure 이벤트가 빠르게 연달아 생성될 수 있습니다. 불필요한 처리를 방지하기 위해 AWS OpsWorks Stacks는 마지막 이벤트에만 응답합니다. 해당 이벤트의 스택 구성 및 배포 속성에 스택의 인스턴스를 전체 변경 사항에 대해 업데이트하는 데 필요한 모든 정보가 포함됩니다. 이렇게 하면 이전 Configure 이벤트도 처리할 필요가 없습니다. AWS OpsWorks Stacks는 처리되지 않은 Configure 이벤트에 대체된 것으로 레이블을 지정합니다.

자동으로 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

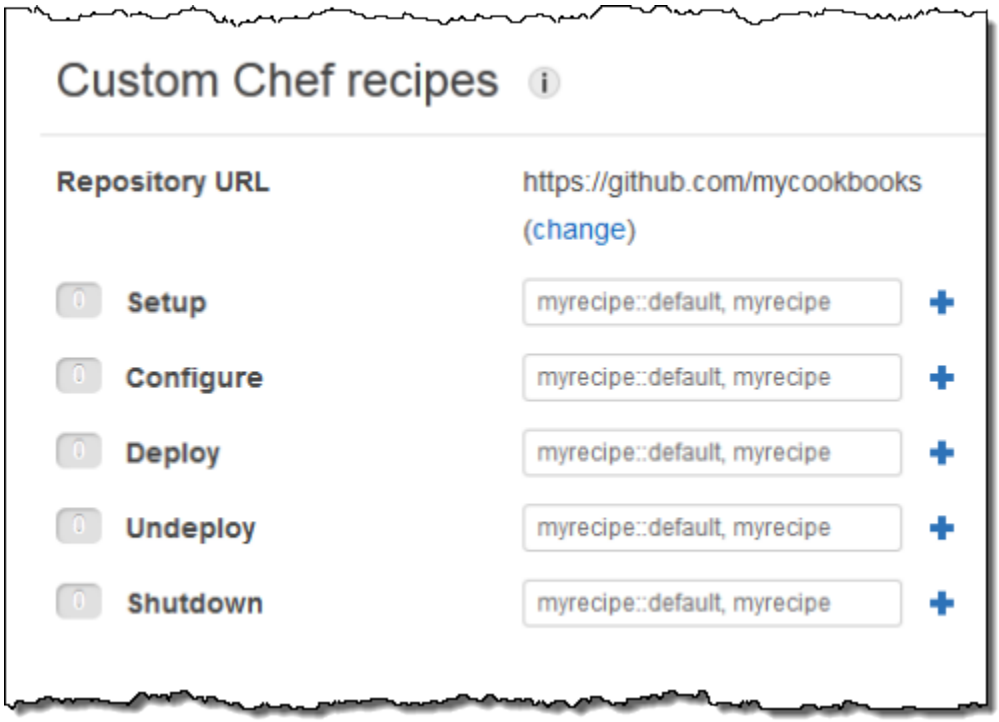
하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

각 계층은 내장 레시피가 각 수명 주기 이벤트에 할당되어 있으며, 일부 계층에는 Undeploy 레시피가 없습니다. 인스턴스에서 수명 주기 이벤트가 발생하면 AWS OpsWorks Stacks는 연결된 계층에 적합한 레시피 세트를 실행합니다.

사용자 지정 AWS OpsWorks 쿡북을 설치한 경우 각 레시피를 계층의 수명 주기 이벤트에 할당하여 Stacks가 레시피의 일부 또는 전체를 자동으로 실행하도록 할 수 있습니다. 이벤트가 발생하면 AWS OpsWorks Stacks는 계층의 기본 제공 레시피 이후에 지정된 사용자 지정 레시피를 실행합니다.

계층 이벤트에 사용자 지정 레시피를 할당하려면

1. [계층] 페이지에서 해당 계층에 대해 [레시피]를 클릭하고 [편집]을 클릭합니다. 사용자 지정 쿡북을 아직 활성화하지 않은 경우 [쿡북 구성]을 클릭하여 스택의 [설정] 페이지를 엽니다. [사용자 지정 Chef 쿡북 사용]을 [예]로 전환하고 쿡북의 리포지토리 정보를 입력합니다. 그런 다음 [저장]을 클릭하고 편집 페이지의 [레시피] 탭으로 돌아갑니다. 자세한 내용은 사용자 지정 쿡북 설치 단원을 참조하십시오.
2. [레시피] 탭에서 해당하는 이벤트 필드에 사용자 지정 레시피를 각각 입력하고 [+]를 클릭하여 목록에 해당 레시피를 추가합니다. 다음과 같이 레시피를 지정합니다. *cookbook*::*somerecipe*(.rb 확장명 제외)



새 인스턴스를 시작하면 AWS OpsWorks Stacks는 표준 레시피를 실행한 후 각 이벤트에 대해 사용자 지정 레시피를 자동으로 실행합니다.

> **Note**
>
> 사용자 지정 레시피는 콘솔에 입력된 순서대로 실행됩니다. 실행 순서를 제어하는 대안적 방법은 레시피를 정확한 순서로 실행하는 메타 레시피를 구현하는 것입니다.

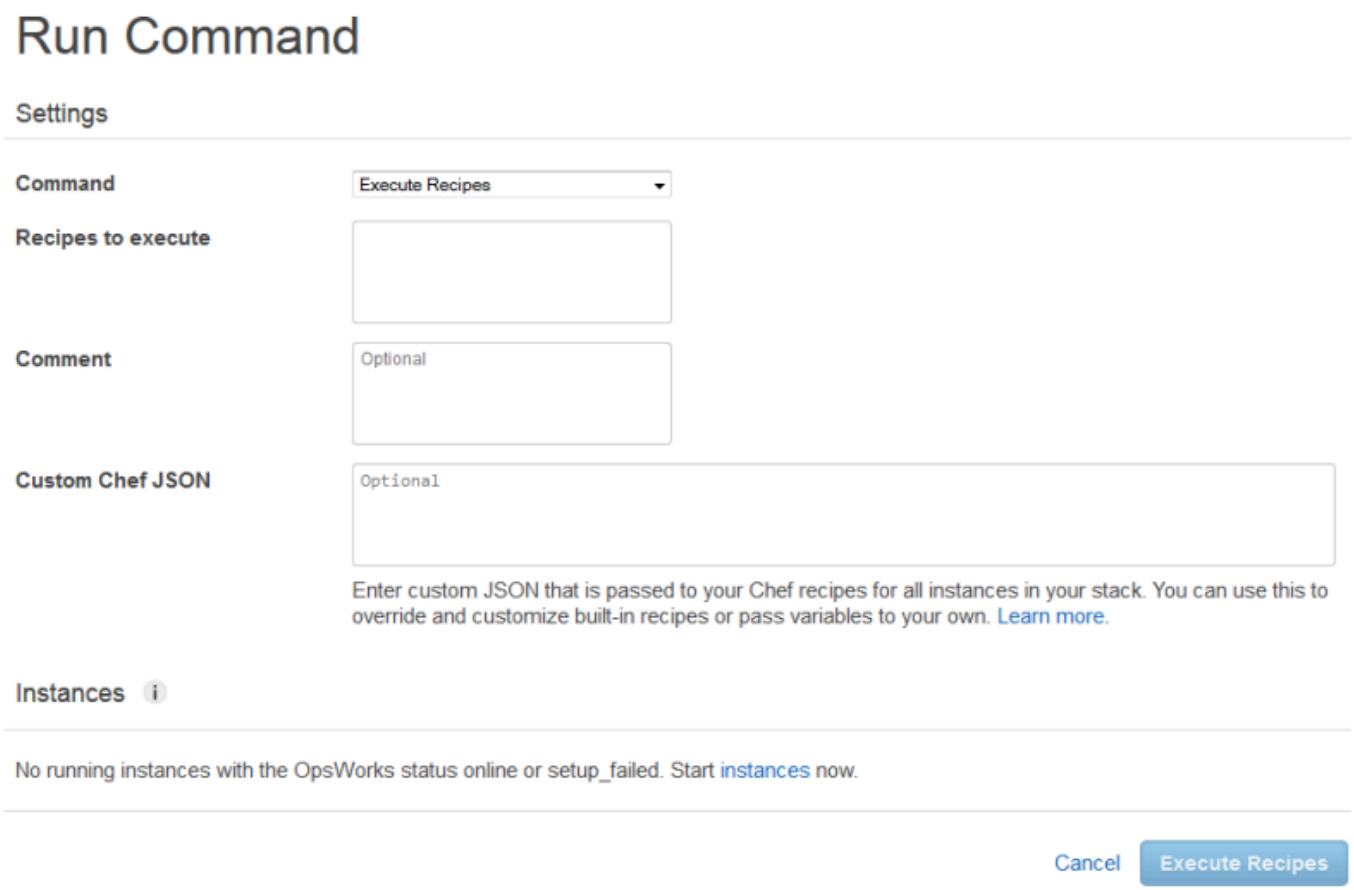
수동으로 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피는 수명 주기 이벤트에서 자동으로 실행되는 것이 일반적이지만, 언제든지 임의의 스택 인스턴스 또는 모든 스택 인스턴스에서 수동으로 실행할 수 있습니다. 이 기능은 인스턴스 백업과 같이 수명 주기 이벤트와 잘 매핑되지 않는 작업에 주로 사용됩니다. 수동으로 사용자 지정 레시피를 실행하려면 해당 레시피가 사용자 지정 쿡북 중 하나여야 하지만 수명 주기 이벤트에 할당될 필요는 없습니다. 레시피를 수동으로 실행하면 AWS OpsWorks Stacks는 Deploy 이벤트에 대해 수행하는 것과 동일한 `deploy` 속성을 설치합니다.

스택 인스턴스에서 레시피를 수동으로 실행하려면

1. 스택 페이지에서 명령 실행을 클릭합니다. 명령에 대해 레시피 실행을 선택합니다.



2. [실행할 레시피] 상자에 실행하려는 레시피를 *cookbookname*::*recipename* 형식으로 입력합니다. 레시피가 여러 개이면 쉼표를 사용하여 구분하고, 레시피는 나열한 순서대로 실행됩니다.
3. 경우에 따라 [사용자 지정 Chef JSON] 상자를 사용하여 사용자 지정 속성을 정의하는 사용자 지정 JSON 객체를 추가합니다. 이러한 속성은 인스턴스에 설치된 스택 구성 및 배포 속성과 병합됩니다. 사용자 지정 JSON 객체 사용에 대한 자세한 정보는 사용자 지정 JSON 사용 및 속성 재정의 단원을 참조하세요.
4. 인스턴스에서 AWS OpsWorks Stacks가 레시피를 실행해야 하는 인스턴스를 선택합니다.

수명 주기 이벤트가 발생하면 AWS OpsWorks Stacks 에이전트는 연결된 레시피를 실행하는 명령을 받습니다. 적절한 스택 명령을 실행하거나 에이전트 CLI의 run_command 명령을 사용하여 특정 인스턴스에서 이러한 명령을 수동으로 실행할 수 있습니다. 에이전트 CLI를 사용하는 방법에 대한 자세한 정보는 AWS OpsWorks Stacks 에이전트 CLI 단원을 참조하세요.

리소스 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

리소스 페이지를 사용하면 AWS OpsWorks Stacks 스택에서 계정의 탄력적 IP 주소, Amazon EBS 볼륨 또는 Amazon RDS 인스턴스 리소스를 사용할 수 있습니다. [리소스]를 사용하여 다음을 수행할 수 있습니다.

- 스택에 리소스를 등록하여 스택의 인스턴스 중 하나에 리소스를 연결할 수 있습니다.
- 스택의 인스턴스 중 하나에 리소스를 연결합니다.
- 한 인스턴스에서 다른 인스턴스로 리소스를 이동합니다.
- 인스턴스에서 리소스를 분리합니다. 리소스는 등록된 상태를 유지하며 다른 인스턴스에 연결할 수 있습니다.
- 리소스 등록 해제. 등록되지 않은 리소스는 AWS OpsWorks Stacks에서 사용할 수 없지만 삭제하지 않는 한 계정에 남아 있으며 다른 스택에 등록할 수 있습니다.

다음과 같은 제약에 유의하세요.

- 등록된 Amazon EBS 볼륨은 Windows 인스턴스에 연결할 수 없습니다.
- 리소스 페이지는 표준, PIOPS, 처리량에 최적화된 HDD, 콜드 HDD 또는 범용(SSD) Amazon EBS 볼륨을 관리하지만 RAID 어레이는 관리하지 않습니다.
- Amazon EBS 볼륨은 xfs 형식이어야 합니다.

  AWS OpsWorks Stacks는 ext4와 같은 다른 파일 형식을 지원하지 않습니다. Amazon EBS 볼륨 준비에 대한 자세한 내용은 Amazon EBS 볼륨을 사용할 수 있도록 만들기를 참조하세요.
- Amazon EBS 볼륨은 실행 중인 인스턴스에 연결하거나 실행 중인 인스턴스에서 분리할 수 없습니다.

오프라인 인스턴스에서만 작업이 가능합니다. 예를 들어 사용 중인 볼륨을 스택에 등록하여 오프라인 인스턴스에 연결할 수 있지만 새 인스턴스를 시작하기 전에 원래 인스턴스를 중지하고 볼륨을 분리해야 합니다. 그렇지 않으면 시작 프로세스가 실패합니다.

- 등록된 모든 리소스는 에서만 관리됩니다 AWS OpsWorks. 이렇게 하면 EC2 볼륨을 위한 `DeleteOnTermination` 같은 리소스 수명 주기 속성을 재정의할 수 있습니다.
- 탄력적 IP 주소는 실행 중인 인스턴스에 연결하거나 실행 중인 인스턴스에서 분리할 수 있습니다.

  온라인 또는 오프라인 인스턴스에서 작업이 가능합니다. 예를 들어 사용 중인 주소를 등록하여 실행 중인 인스턴스에 할당하면 AWS OpsWorks Stacks가 자동으로 주소를 재할당합니다.
- 리소스를 등록 및 등록 해제하기 위해서는 IAM 정책이 다음 작업에 대한 권한을 부여해야 합니다.

Amazon EBS 볼륨	탄력적 IP 주소	Amazon RDS 인스턴스
RegisterVolume	RegisterElasticIp	RegisterRdsDbInstance
UpdateVolume	UpdateElasticIp	UpdateRdsDbInstance
DeregisterVolume	DeregisterElasticIp	DeregisterRdsDbInstance

권한 수준 관리는 이러한 모든 작업에 대한 권한을 부여합니다. Manage 사용자가 특정 리소스를 등록 또는 등록 해제하는 것을 방지하려면 적절한 작업에 대한 권한을 거부하도록 IAM 정책을 편집하세요. 자세한 내용은 보안 및 권한 단원을 참조하십시오.

주제

- 스택에 리소스 등록
- 리소스 연결 및 이동
- 리소스 분리
- 리소스 등록 해제

스택에 리소스 등록

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon EBS 볼륨이나 탄력적 IP 주소는 인스턴스에 연결하기 전에 스택에 등록해야 합니다. AWS OpsWorks Stacks가 스택에 대한 리소스를 생성하면 해당 스택에 자동으로 등록됩니다. 외부에서 생성된 리소스를 사용하려면 리소스를 명시적으로 등록해야 합니다. 다음 사항에 유의하세요.

- 한 번에 하나의 스택에만 리소스를 등록할 수 있습니다.
- 스택을 삭제하면 AWS OpsWorks Stacks가 모든 리소스의 등록을 취소합니다.

주제

- 스택에 Amazon EBS 볼륨 등록
- 스택에 탄력적 IP 주소 등록
- 스택에 Amazon RDS 인스턴스 등록

스택에 Amazon EBS 볼륨 등록

> **Note**
>
> 이 리소스는 Linux 스택에서만 사용할 수 있습니다. Windows 스택에 Amazon EBS 볼륨을 등록할 수 있지만 인스턴스에 연결할 수는 없습니다.

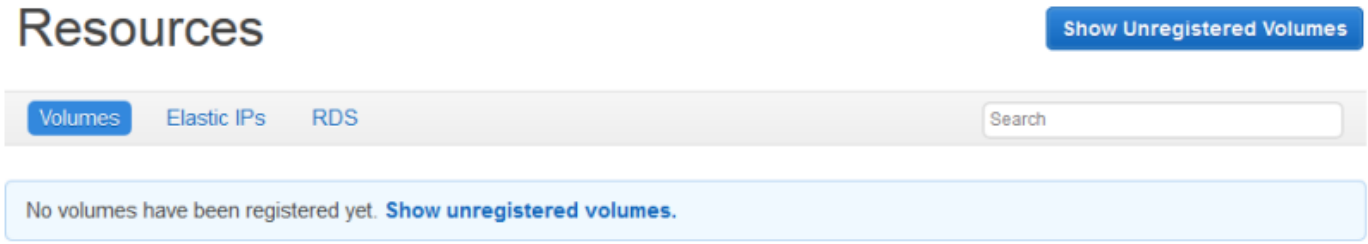
리소스 페이지에서 스택에 Amazon EBS 볼륨을 등록할 수 있으며, 이때 다음의 제약이 적용됩니다.

- 연결되는 비루트 Amazon EBS 볼륨은 표준, 처리량에 최적화된 HDD, 콜드 HDD, PIOPS 또는 범용(SSD)이어야 하며, RAID 어레이가 아니어야 합니다. 최대 및 최소 볼륨 크기에 대한 자세한 내용은 이 설명서의 EBS 볼륨 단원을 참조하세요.
- 볼륨은 XFS 형식이어야 합니다.

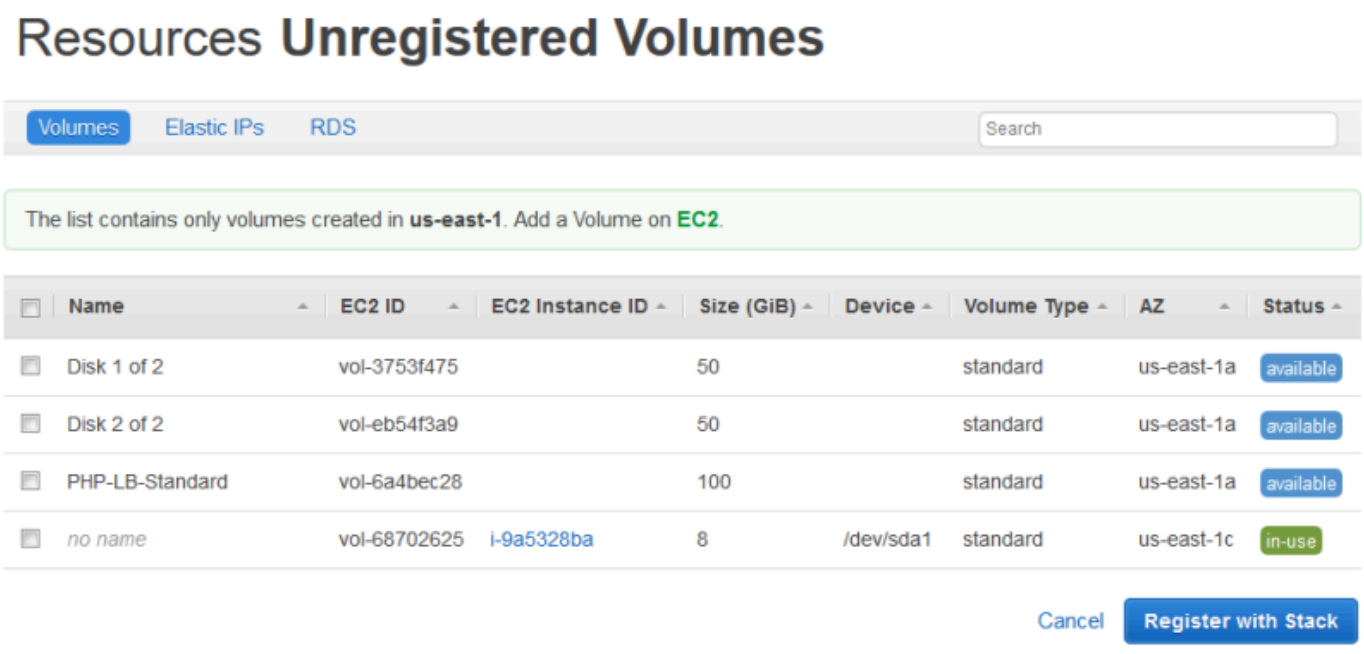
- AWS OpsWorks Stacks는 비루트 Amazon EBS 볼륨에 ext4(fourth extended file system) 등 다른 파일 형식을 지원하지 않습니다. Amazon EBS 볼륨 준비에 대한 자세한 내용은 Amazon EBS 볼륨을 사용할 수 있도록 만들기를 참조하세요. 이 주제의 예에서는 ext4 기반 볼륨 생성 방법을 설명하고 있지만 XFS 기반 볼륨도 같은 단계에 따르면 됩니다.

Amazon EBS 볼륨을 등록하려면

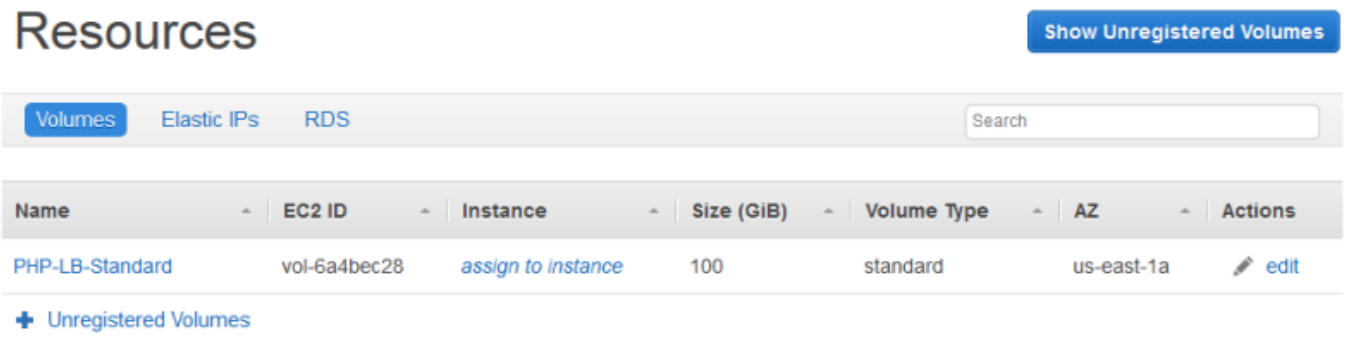
1. 원하는 스택을 열고 탐색 창에서 리소스를 클릭합니다.
2. 볼륨을 클릭하여 사용 가능한 Amazon EBS 볼륨을 표시합니다. 다음 그림과 같이 처음에는 스택에 등록된 볼륨이 없습니다.



3. 미등록 볼륨 표시를 클릭하여 스택의 리전 및 VPC(해당하는 경우)에 있는 사용자 계정의 Amazon EBS 볼륨을 표시합니다. [상태] 열에는 볼륨을 사용할 수 있는지 여부가 나타납니다. 볼륨 유형은 볼륨이 표준(`standard`), 범용 SSD(`gp2`), PIOPS(`io1`, 뒤에 디스크당 IOPS 값이 괄호로 묶여 표시됨), 처리량에 최적화된 HDD(`st1`) 또는 콜드 HDD(`sc1`) 중에서 어떤 것인지를 나타냅니다.



4. 적절한 볼륨을 선택한 다음 [스택에 등록]을 클릭합니다. 이제 [리소스] 페이지에 새로 등록된 볼륨이 표시됩니다.



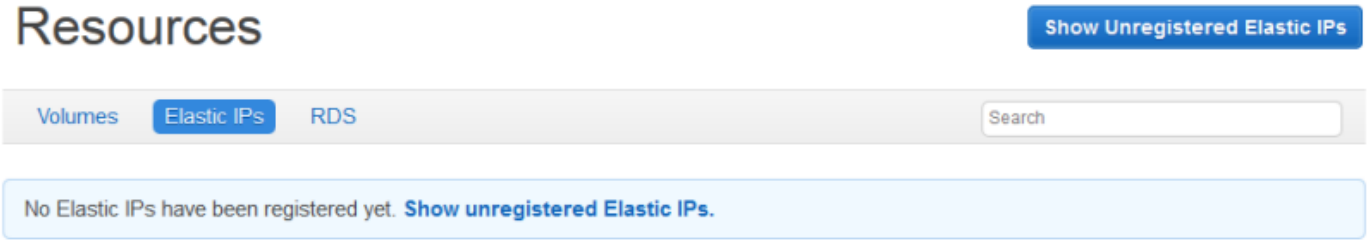
추가 볼륨을 등록하려면 [미등록 볼륨 표시] 또는 [+ 미등록 볼륨]을 클릭하고 이 절차를 반복합니다.

스택에 탄력적 IP 주소 등록

탄력적 IP 주소를 등록하려면 다음 절차를 사용합니다.

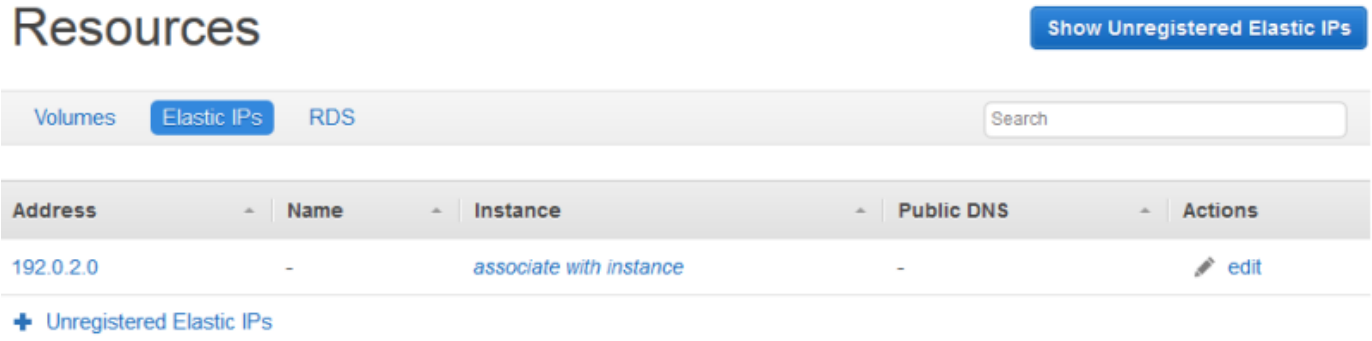
탄력적 IP 주소를 등록하려면

1. 스택의 [리소스] 페이지를 열고 [탄력적 IP]를 클릭하여 사용 가능한 탄력적 IP 주소를 표시합니다. 다음 그림과 같이 처음에 스택에는 등록된 주소가 없습니다.



2. [미등록 탄력적 IP 표시]를 클릭하여 스택의 리전에 있는 계정에서 사용 가능한 탄력적 IP 주소를 표시합니다.

3. 적절한 주소를 선택하고 [스택에 등록]을 클릭합니다. 그러면 리소스 페이지로 돌아옵니다. 이제 이 페이지에 새로 등록된 주소가 표시됩니다.



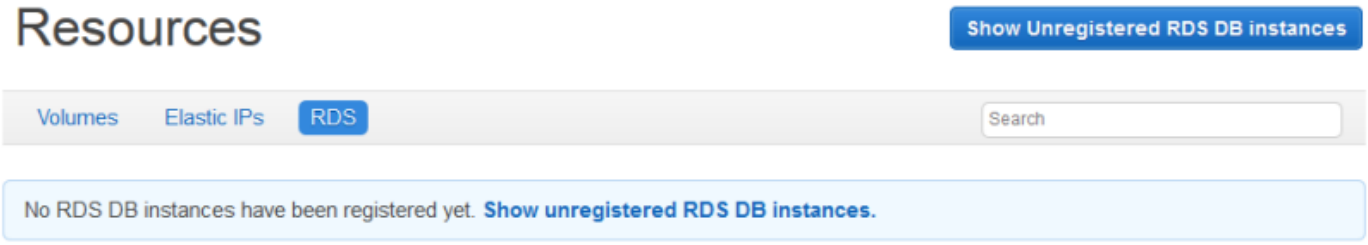
추가 주소를 등록하려면 [미등록 탄력적 IP 표시] 또는 [+ 미등록 탄력적 IP]를 클릭하고 이 절차를 반복합니다.

스택에 Amazon RDS 인스턴스 등록

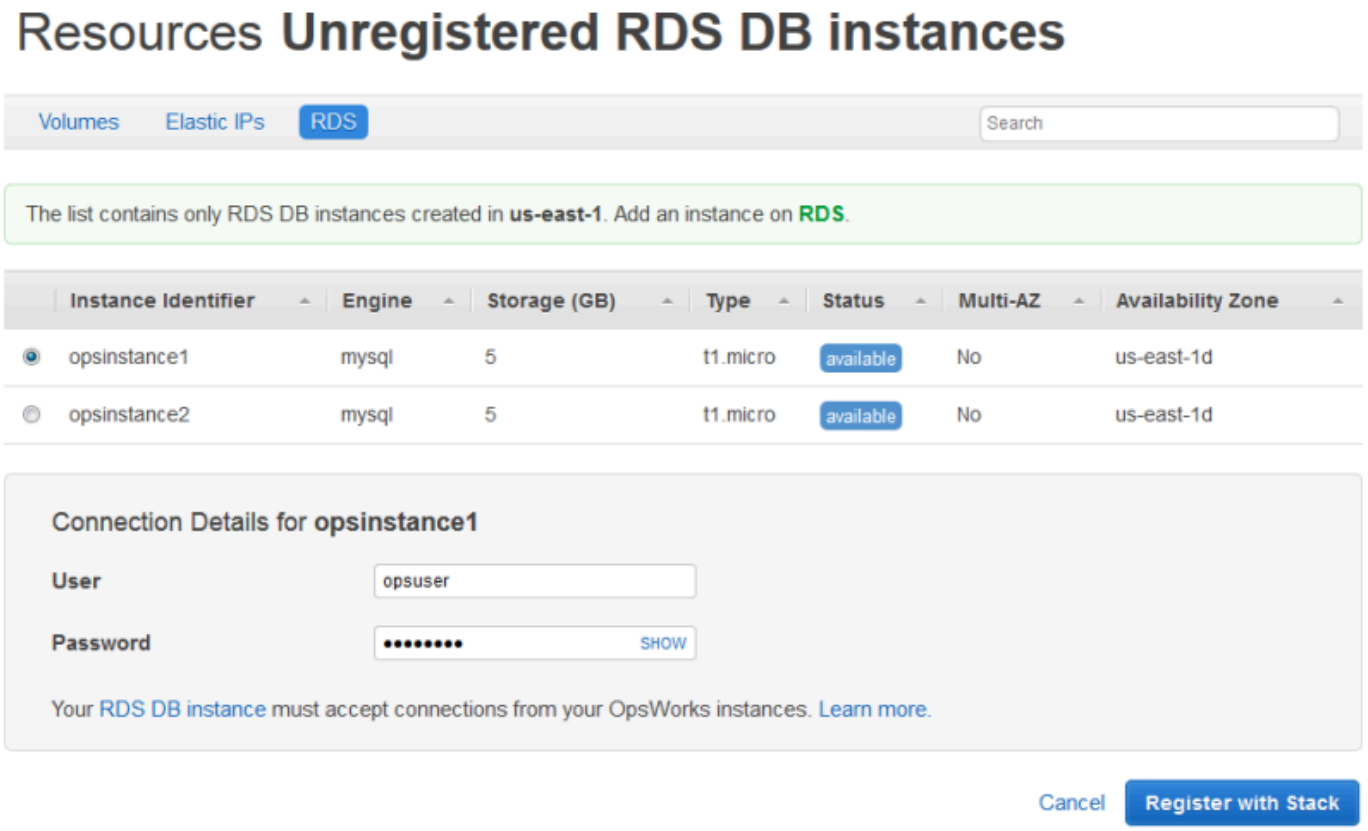
다음 절차에 따라 Amazon RDS 인스턴스를 등록합니다.

Amazon RDS 인스턴스를 등록하려면

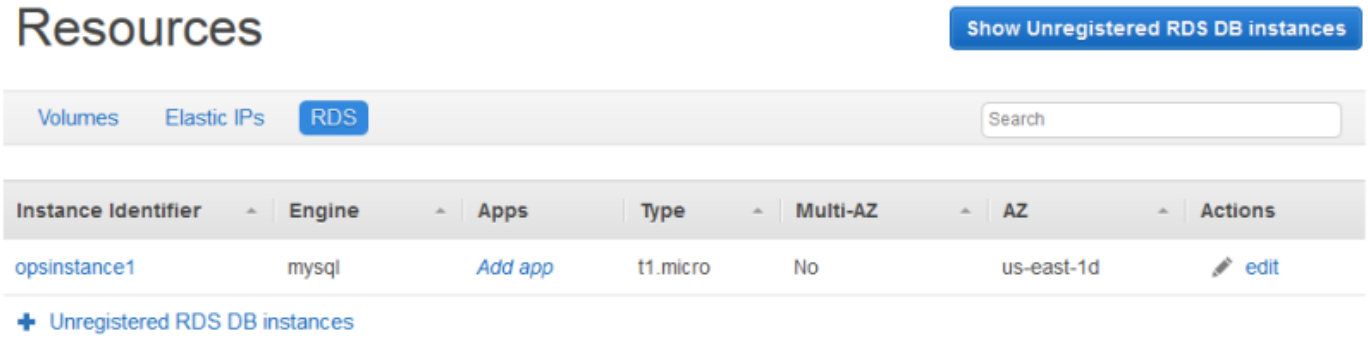
1. 스택의 리소스 페이지를 열고 RDS를 클릭하여 사용 가능한 Amazon RDS 인스턴스를 표시합니다. 다음 그림과 같이 처음에 스택에는 등록된 인스턴스가 없습니다.



2. 미등록 RDS DB 인스턴스 표시를 클릭하여 스택의 리전에 있는 사용자 계정에서 사용 가능한 Amazon RDS 인스턴스를 표시합니다.



3. 적절한 인스턴스를 선택하고, [사용자] 및 [암호]에 마스터 사용자 및 마스터 암호 값을 입력한 다음 [스택에 등록]을 클릭합니다. 그러면 리소스 페이지로 돌아옵니다. 이제 이 페이지에 새로 등록된 인스턴스가 표시됩니다.



> **Important**
>
> Amazon RDS 인스턴스 등록에 사용할 사용자 및 암호가 유효한 것인지 확인해야 합니다. 그렇지 않으면 애플리케이션에서 인스턴스에 연결할 수 없게 됩니다.

추가 주소를 등록하려면 [미등록 RDS DB 인스턴스 표시] 또는 [+ 미등록 RDS DB 인스턴스]를 클릭하고 이 절차를 반복합니다. AWS OpsWorks Stacks에서 Amazon RDS 인스턴스를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요Amazon RDS 서비스 계층.

> **Note**
>
> 계층 페이지를 통해 Amazon RDS 인스턴스를 등록할 수도 있습니다. 자세한 내용은 Amazon RDS 서비스 계층 단원을 참조하십시오.

리소스 연결 및 이동

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택에 리소스를 등록한 후 스택의 인스턴스 중 하나에 리소스를 연결할 수 있습니다. 연결된 리소스를 다른 인스턴스로 이동할 수도 있습니다. 다음 사항에 유의하세요.

- Amazon EBS 볼륨을 연결하거나 이동할 때는 작업에 관련된 인스턴스가 오프라인 상태여야 합니다. 관심 대상 인스턴스가 [리소스] 페이지에 없다면 [인스턴스] 페이지로 가서 인스턴스를 중지합니다. 인스턴스가 중지된 후 [리소스] 페이지로 돌아와 인스턴스를 연결하거나 이동할 수 있습니다.
- 탄력적 IP 주소는 인스턴스가 온라인 상태이거나 오프라인 상태일 때 연결하거나 이동할 수 있습니다.
- 인스턴스를 삭제하면 연결된 리소스는 스택에 등록된 상태를 유지합니다. 그런 다음 리소스를 다른 인스턴스에 연결하거나 더 이상 필요하지 않으면 리소스 등록을 해제할 수 있습니다.

주제

- 인스턴스에 Amazon EBS 볼륨 할당
- 인스턴스에 탄력적 IP 주소 연결
- Amazon RDS 인스턴스를 앱에 연결

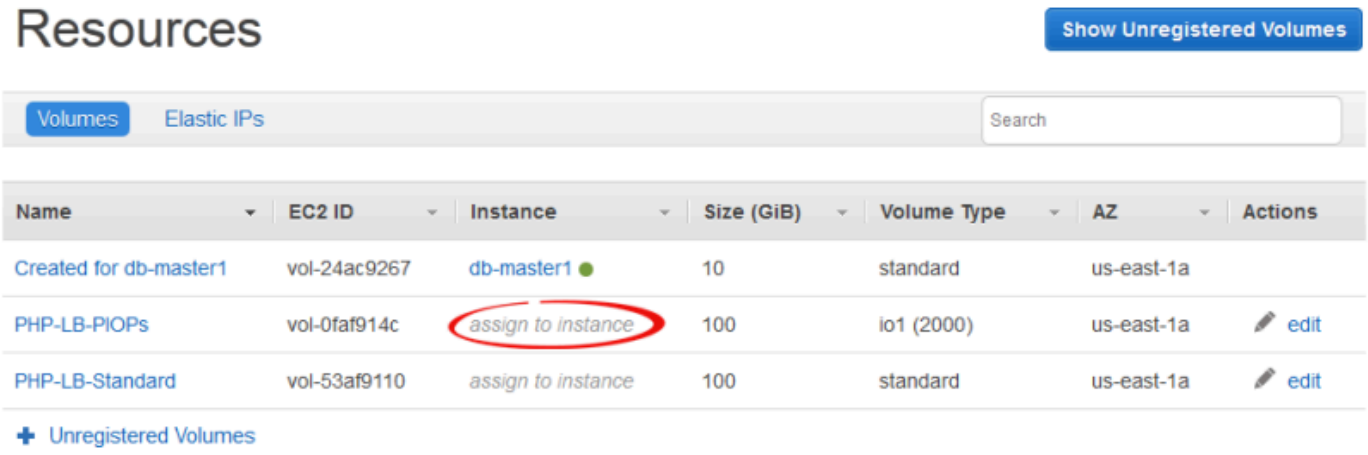
인스턴스에 Amazon EBS 볼륨 할당

> **Note**
>
> Amazon EBS 볼륨은 Windows 인스턴스에 할당할 수 없습니다.

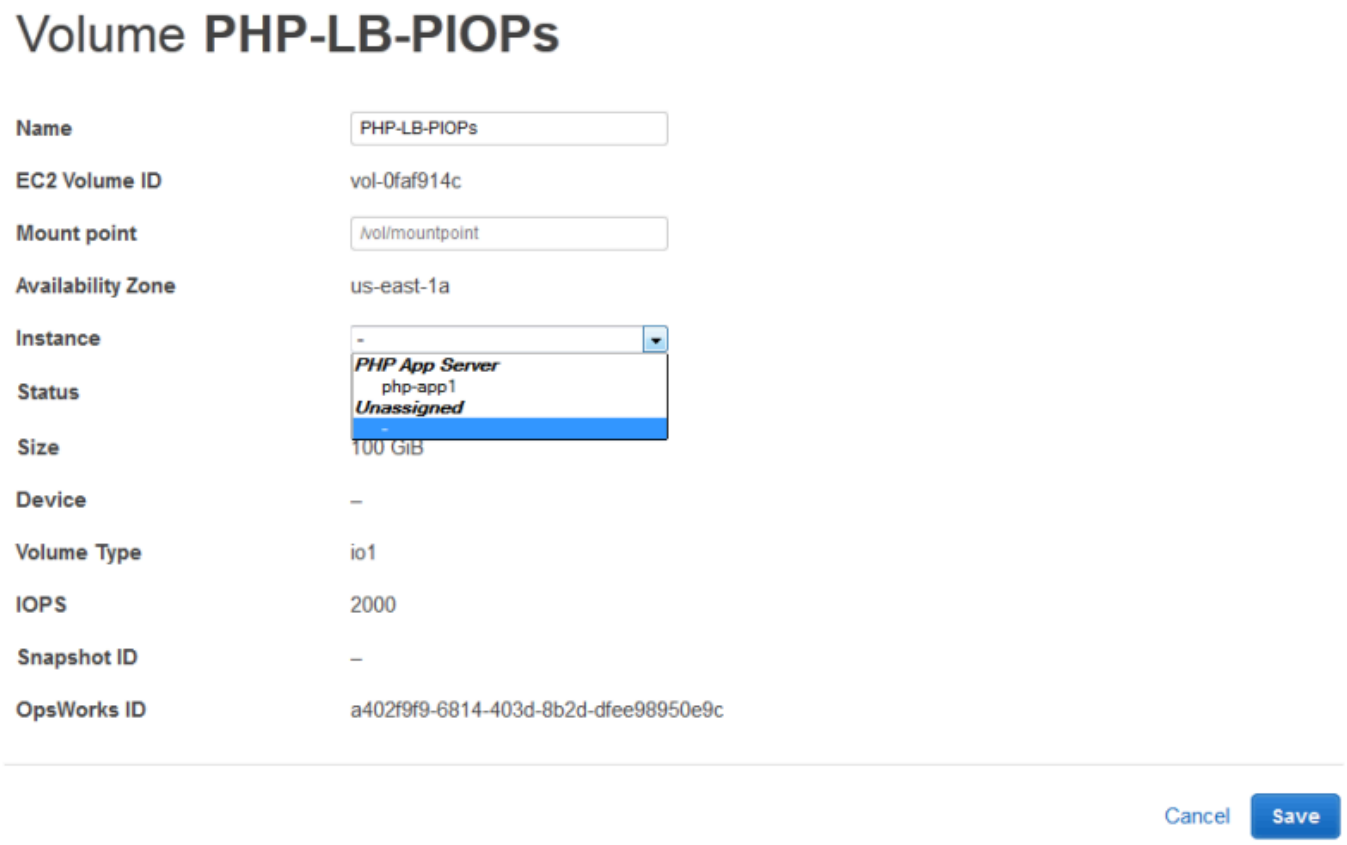
등록된 Amazon EBS 볼륨은 인스턴스에 할당하고 한 인스턴스에서 다른 인스턴스로 이동할 수 있습니다. 단, 두 인스턴스 모두 오프라인 상태여야 합니다.

Amazon EBS 볼륨을 인스턴스에 할당하려면

1. 리소스 페이지에서 해당 볼륨의 인스턴스 열에서 인스턴스에 할당을 클릭합니다.



2. 볼륨의 세부 정보 페이지에서 해당 인스턴스를 선택하고, 볼륨의 이름 및 탑재 지점을 지정하고, [저장]을 클릭하여 볼륨을 인스턴스에 연결합니다.



> **Important**
>
> 사용 중인 외부 볼륨을 인스턴스에 할당한 경우, Amazon EC2 콘솔, API 또는 CLI를 사용하여 원래 인스턴스에서 볼륨 할당을 해제해야 합니다. 그렇지 않으면 시작 프로세스가 실패합니다.

세부 정보 페이지를 사용하여 할당된 Amazon EBS 볼륨을 스택 내의 다른 인스턴스로 이동할 수도 있습니다.

Amazon EBS 볼륨을 다른 인스턴스로 이동하려면

1. 두 인스턴스가 오프라인 상태인지 확인합니다.
2. 리소스 페이지에서 볼륨을 클릭한 다음 볼륨의 작업 열에서 편집을 클릭합니다.
3. 다음 중 하나를 수행합니다.

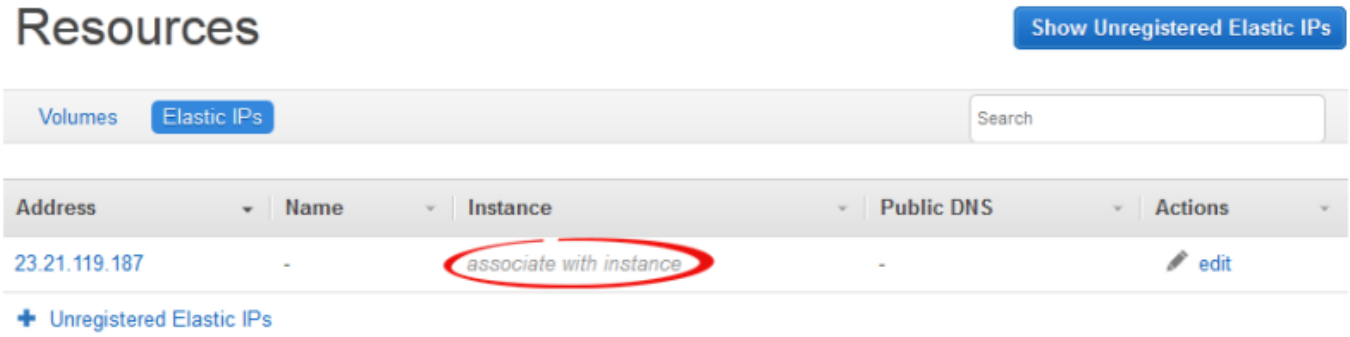
- 스택의 다른 인스턴스로 볼륨을 이동하려면 [인스턴스] 목록에서 해당 인스턴스를 선택하고 [저장]을 클릭합니다.
- 다른 스택의 인스턴스로 볼륨을 이동하려면 볼륨의 등록을 취소하고, 새 스택에 볼륨을 등록한 다음 새 인스턴스에 볼륨을 연결합니다.

인스턴스에 탄력적 IP 주소 연결

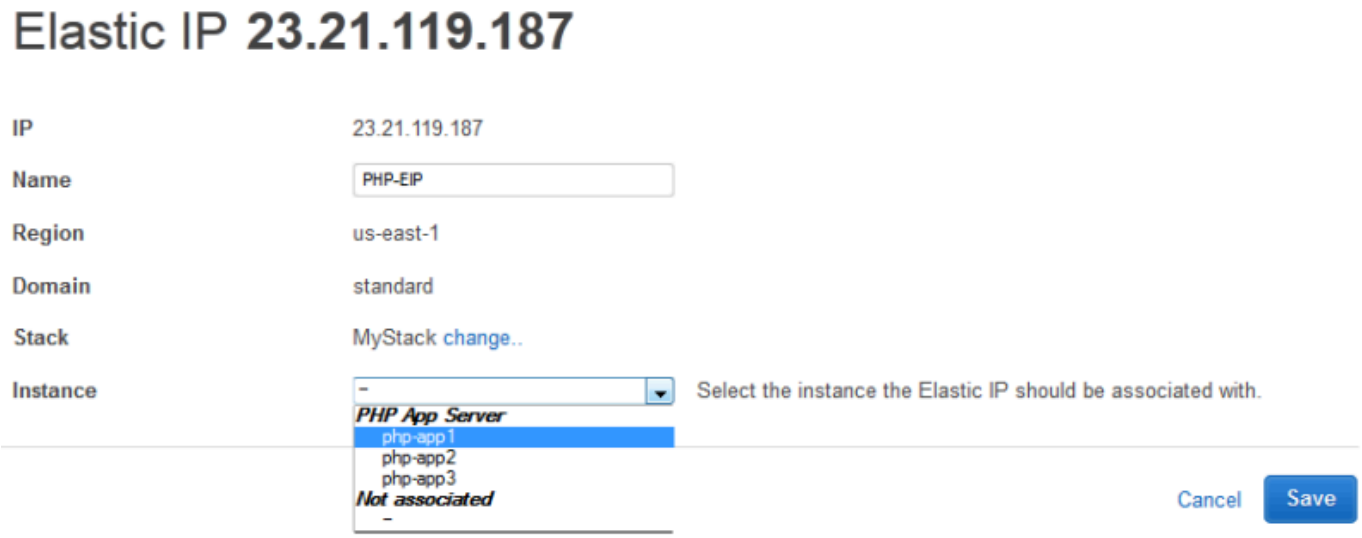
등록된 탄력적 IP 주소는 인스턴스에 할당하고 다른 스택에 있는 인스턴스를 비롯한 다른 인스턴스로 이동할 수 있습니다. 이러한 인스턴스는 온라인 상태이거나 오프라인 상태일 수 있습니다.

인스턴스와 탄력적 IP 주소 연결

1. 리소스 페이지에서 해당 주소의 인스턴스 열에서 인스턴스와 연결을 클릭합니다.



2. 주소의 세부 정보 페이지에서 해당 인스턴스를 선택하고, 주소의 이름을 지정하고, [저장]을 클릭하여 주소를 인스턴스와 연결합니다.

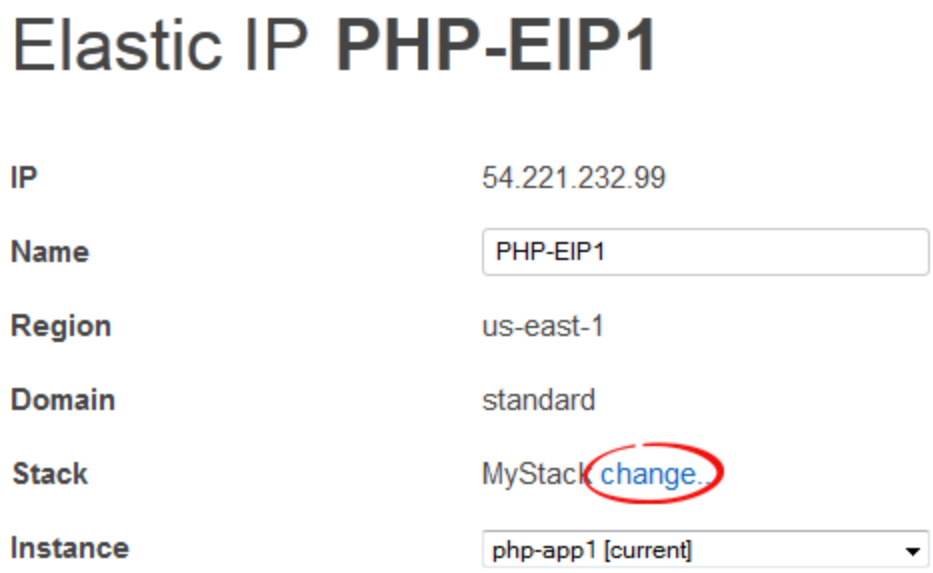


> **Note**
>
> 탄력적 IP 주소가 현재 다른 온라인 인스턴스와 연결되어 있는 경우 AWS OpsWorks Stacks는 주소를 새 인스턴스에 자동으로 재할당합니다.

세부 정보 페이지를 사용하여 연결된 탄력적 IP 주소를 다른 인스턴스로 이동할 수도 있습니다.

탄력적 IP 주소를 다른 인스턴스로 이동하려면

1. 리소스 페이지에서 탄력적 IP를 클릭한 다음 주소의 작업 열에서 편집을 클릭합니다.
2. 다음 중 하나를 수행합니다.
   - 스택의 다른 인스턴스로 주소를 이동하려면 [인스턴스] 목록에서 해당 인스턴스를 선택하고 [저장]을 클릭합니다.
   - 주소를 다른 스택의 인스턴스로 이동하려면 스택 설정에서 변경을 클릭하여 사용 가능한 스택 목록을 확인하세요. [스택] 목록에서 스택을 선택하고 [인스턴스] 목록에서 인스턴스를 선택합니다. 그런 다음 저장을 클릭합니다.



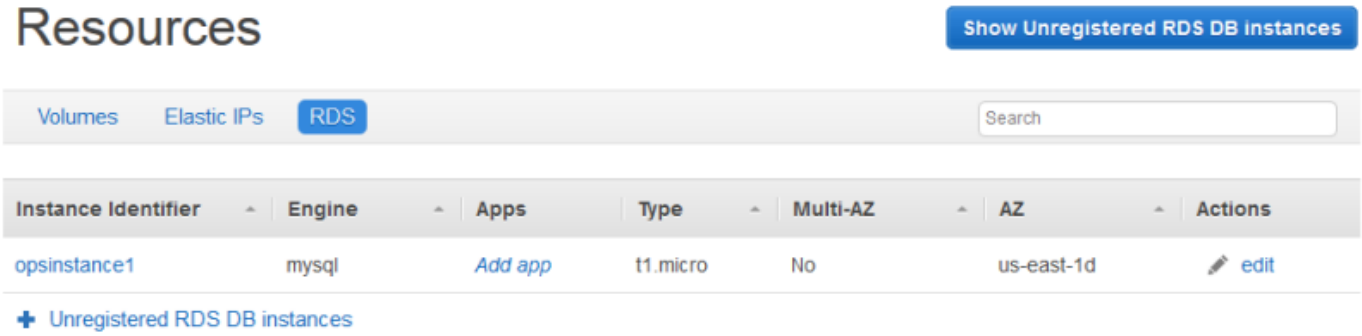
주소를 연결하거나 이동한 후 AWS OpsWorks Stacks는 수명 주기 구성 이벤트를 트리거하여 스택의 인스턴스에 변경 사항을 알립니다.

Amazon RDS 인스턴스를 앱에 연결

하나 이상의 앱에 Amazon RDS 인스턴스를 연결할 수 있습니다.

앱에 Amazon RDS 인스턴스를 연결하려면

1. 리소스 페이지에서 해당 인스턴스의 앱 열에서 앱 추가을 클릭합니다.



2. 앱 추가 페이지를 사용하여 Amazon RDS 인스턴스를 연결합니다. 자세한 내용은 앱 추가 단원을 참조하십시오.

Amazon RDS는 여러 앱에 연결할 수 있으므로 한 앱에서 다른 앱으로 인스턴스를 이동하기 위한 특별한 절차는 없습니다. 첫 번째 앱을 편집하여 RDS 인스턴스를 제거하거나 두 번째 앱을 편집하여 RDS 인스턴스를 추가하면 됩니다. 자세한 내용은 앱 편집 단원을 참조하십시오.

리소스 분리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

더 이상 필요 없는 리소스는 분리할 수 있습니다. 이 리소스는 스택에 등록된 상태를 유지하며 다른 곳에 연결할 수 있습니다.

주제

- Amazon EBS 볼륨 할당 취소
- 탄력적 IP 주소 연결 해제

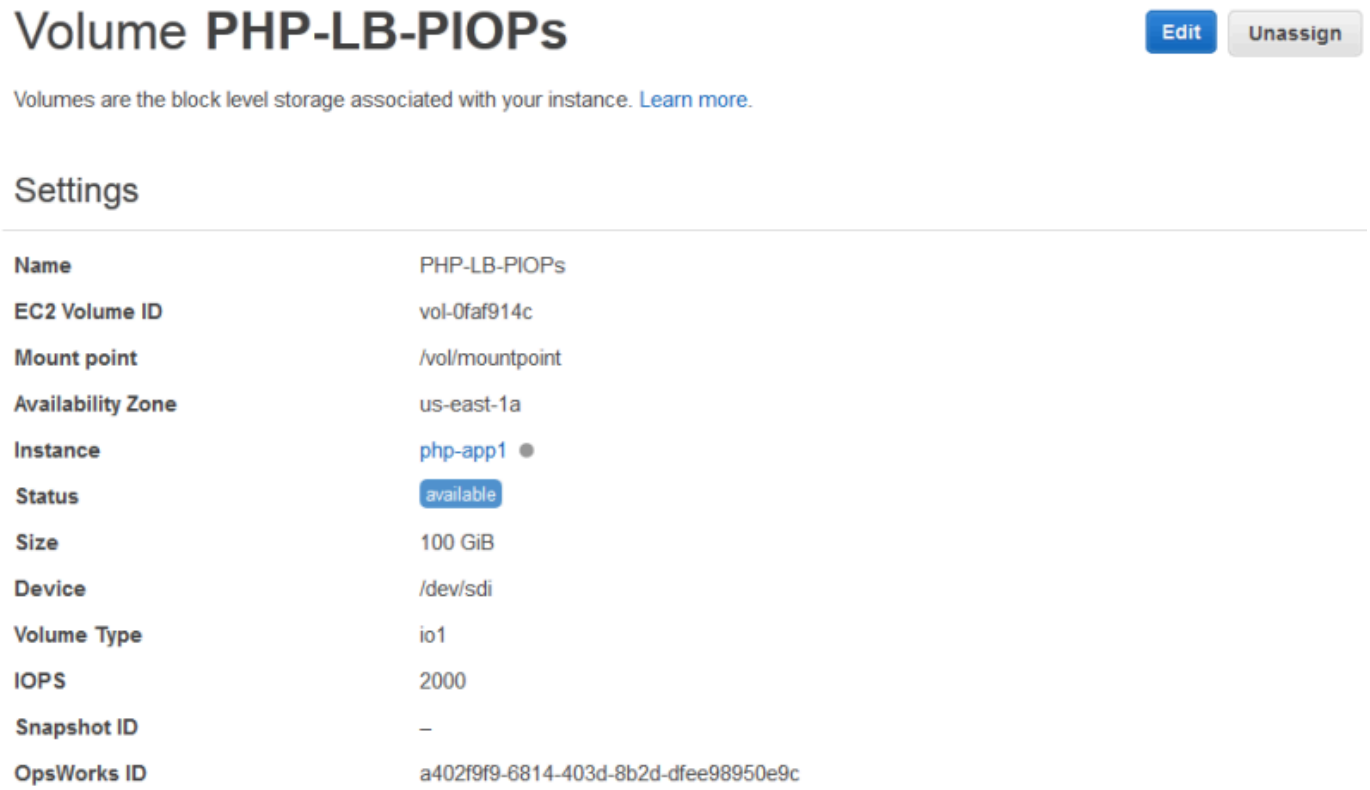
- Amazon RDS 인스턴스 분리

Amazon EBS 볼륨 할당 취소

인스턴스에서 Amazon EBS 볼륨 할당을 해제하려면 다음 절차를 사용합니다.

Amazon EBS 볼륨을 할당 해제하려면

1. 인스턴스가 오프라인 상태인지 확인합니다.
2. [리소스] 페이지에서 [볼륨] 및 볼륨 이름을 차례로 클릭합니다.
3. 볼륨의 세부 정보 페이지에서 [할당 취소]를 클릭합니다.

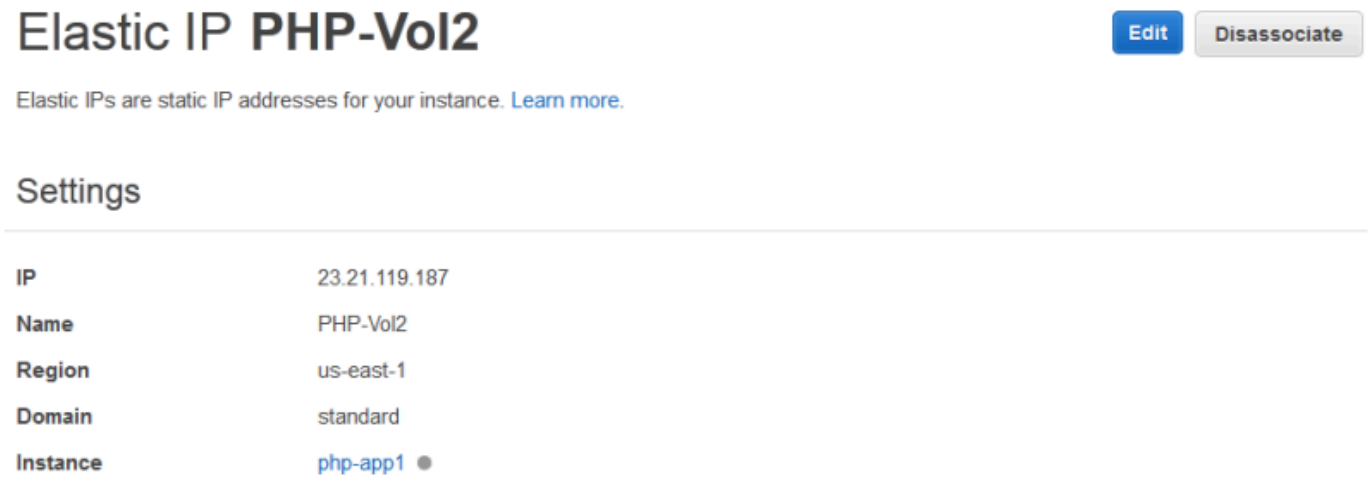


탄력적 IP 주소 연결 해제

인스턴스에서 탄력적 IP 주소 연결을 해제하려면 다음 절차를 사용합니다.

탄력적 IP 주소를 연결 해제하려면

1. 리소스 페이지에서 탄력적 IP를 클릭한 다음 주소의 작업 열에서 편집을 클릭합니다.
2. 주소의 세부 정보 페이지에서 [연결 해제]를 클릭합니다.



주소 연결을 해제하면 AWS OpsWorks 스택은 수명 주기 구성 이벤트를 트리거하여 스택의 인스턴스에 변경 사항을 알립니다.

Amazon RDS 인스턴스 분리

앱에서 Amazon RDS를 분리하려면 다음 절차를 사용합니다.

Amazon RDS 인스턴스를 분리하려면

1. [리소스] 페이지에서 [RDS]를 클릭하고 [앱] 열에서 해당 앱을 클릭합니다.
2. [편집]을 클릭하고 앱 구성을 편집해 인스턴스를 분리합니다. 자세한 내용은 앱 편집 단원을 참조하십시오.

> **Note**
>
> 이 절차는 단일 앱에서 Amazon RDS를 분리합니다. 인스턴스가 여러 앱에 연결된 경우, 앱마다 이 절차를 반복해야 합니다.

리소스 등록 해제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택에 등록된 리소스가 더 이상 필요하지 않으면 등록을 해제할 수 있습니다. 등록 취소는 계정에서 리소스를 삭제하지 않습니다. 리소스는 해당 계정에서 유지되며 다른 스택에 등록하거나 AWS OpsWorks Stacks 외부에서 사용할 수 있습니다. 리소스를 완전히 삭제하려면 다음과 같은 두 가지 방법이 있습니다.

- 탄력적 IP 또는 Amazon EBS 리소스가 인스턴스에 연결되어 있는 경우, 인스턴스를 삭제할 때 해당 리소스를 삭제할 수 있습니다.

  인스턴스 페이지로 이동하여 인스턴스의 작업 열에서 삭제를 클릭한 다음 인스턴스의 EBS 볼륨 삭제 또는 인스턴스의 탄력적 IP 삭제를 선택합니다.
- 리소스 등록을 해제한 다음 Amazon EC2 또는 Amazon RDS 콘솔, API 또는 CLI를 사용하여 리소스를 삭제합니다.

주제

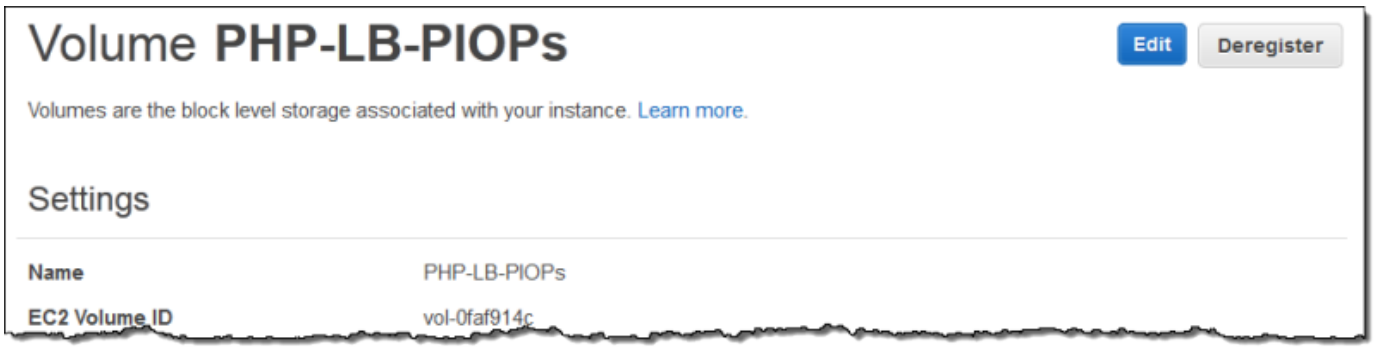
- Amazon EBS 볼륨 등록 취소
- 탄력적 IP 주소 등록 해제
- Amazon RDS 인스턴스 등록 취소

Amazon EBS 볼륨 등록 취소

다음 절차를 사용하여 Amazon EBS 볼륨 등록을 해제합니다.

Amazon EBS 볼륨을 등록 취소하려면

1. 볼륨이 인스턴스에 연결된 경우 Amazon EBS 볼륨 할당 취소 단원에 설명된 것처럼 할당을 해제합니다.
2. [리소스] 페이지의 [이름] 열에서 볼륨 이름을 클릭합니다.
3. 볼륨의 세부 정보 페이지에서 [등록 취소]를 클릭합니다.

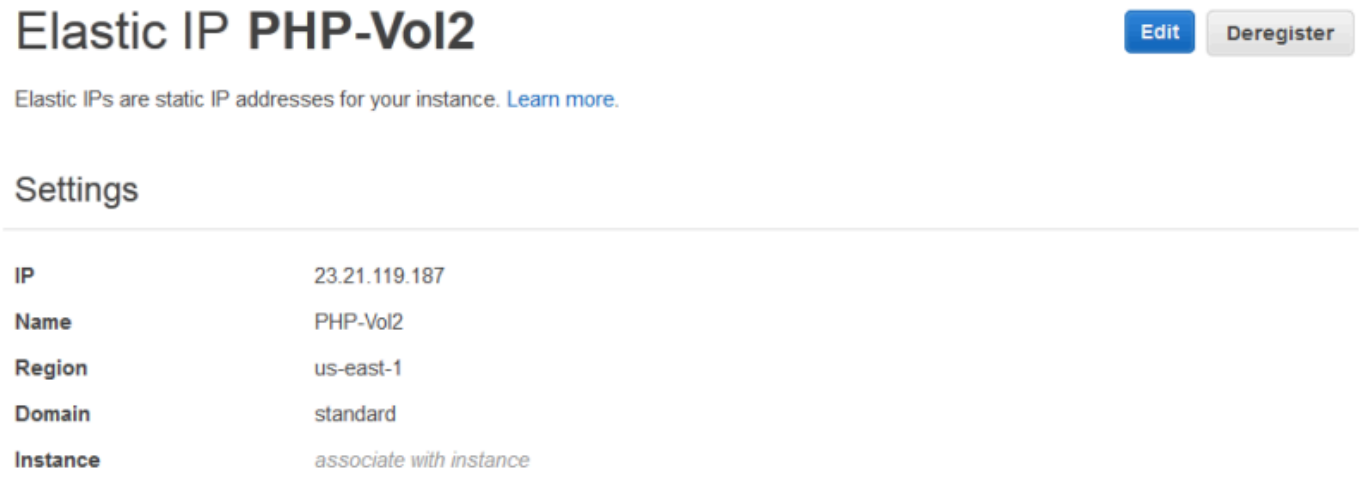


탄력적 IP 주소 등록 해제

탄력적 IP 주소를 등록 해제하려면 다음 절차를 사용합니다.

탄력적 IP 주소를 등록 취소하려면

1. 주소가 인스턴스와 연결된 경우 탄력적 IP 주소 연결 해제 단원의 설명에 따라 연결 해제합니다.
2. [리소스] 페이지에서 [탄력적 IP]를 클릭한 다음 [주소] 열에서 IP 주소를 클릭합니다.
3. 주소의 세부 정보 페이지에서 [등록 취소]를 클릭합니다.



> **Note**
>
> 단지 다른 스택에 탄력적 IP 주소를 등록하려는 경우, 현재 스택에서 등록을 해제한 다음 새 스택에 등록해야 합니다. 다만 연결된 탄력적 IP 주소를 다른 스택에 있는 인스턴스로 직접 이동할 수 있습니다. 자세한 내용은 리소스 연결 및 이동 단원을 참조하십시오.

Amazon RDS 인스턴스 등록 취소

다음 절차를 사용하여 Amazon RDS 인스턴스 등록을 해제합니다.

Amazon RDS 인스턴스를 등록 취소합니다.

1. 인스턴스가 앱과 연결된 경우 리소스 분리 단원의 설명에 따라 분리합니다.
2. [리소스] 페이지에서 [RDS] 및 인스턴스 이름을 차례로 클릭합니다.
3. 인스턴스의 세부 정보 페이지에서 [등록 취소]를 클릭합니다.

Tags

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

태그는 Chef 11.10, Chef 12, Chef 12.2 스택에서 리소스를 그룹화하고 AWS 결제 및 비용 관리에서 이러한 리소스를 실행하는 비용을 추적하는 데 도움이 될 수 있습니다.

스택 및 계층 수준에서 태그를 적용할 수 있습니다. 태그를 생성하면 태그가 지정된 구조 내의 모든 리소스에 태그가 적용됩니다. 예를 들어 계층에 태그를 적용하면 계층에 있는 모든 인스턴스, Amazon EBS 볼륨(루트 제외) 또는 Elastic Load Balancing 로드 밸런서에 태그가 적용됩니다. 현재는 루트 또는 인스턴스의 기본 EBS 볼륨에 태그를 적용할 수 없습니다.

태그는 AWS OpsWorks Stacks의 스택 또는 계층에 할당하는 키-값 페어입니다. 태그를 생성한 후 Billing and Cost Management 콘솔을 열어 사용자 정의 태그를 활성화합니다. 태그를 활성화하고 이를

사용하여 AWS OpsWorks Stacks 리소스 비용을 추적하고 관리하는 방법에 대한 자세한 내용은 Billing and Cost Management 사용 설명서의 비용 할당 태그 사용 및 사용자 정의 비용 할당 태그 활성화를 참조하세요.

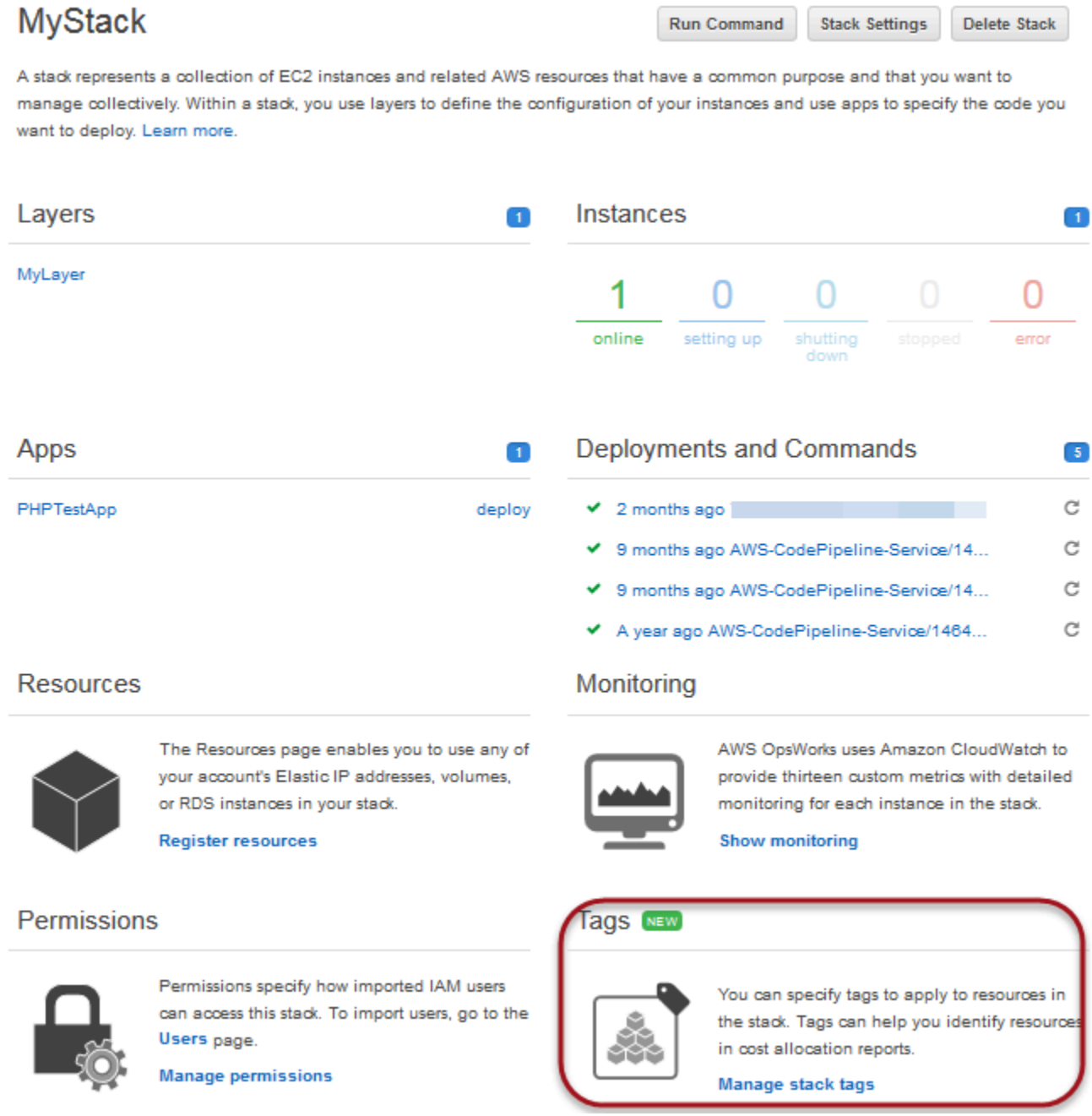
태그는 AWS OpsWorks Stacks의 사용자 지정 속성과 유사한 방식으로 작동합니다. 스택에 적용하는 태그는 스택의 각 계층에 의해 상속됩니다. 계층 수준에서 상속된 태그의 값(키 이름은 아님)을 재정의하고 새 계층별 태그를 추가할 수 있습니다.는 계층의 모든 리소스에 결과 태그 세트를 AWS OpsWorks 적용합니다. 새 리소스를 생성하거나 기존 리소스를 계층에 추가하면 계층의 새 리소스는 같은 태그 세트로 태그 지정됩니다.

주제

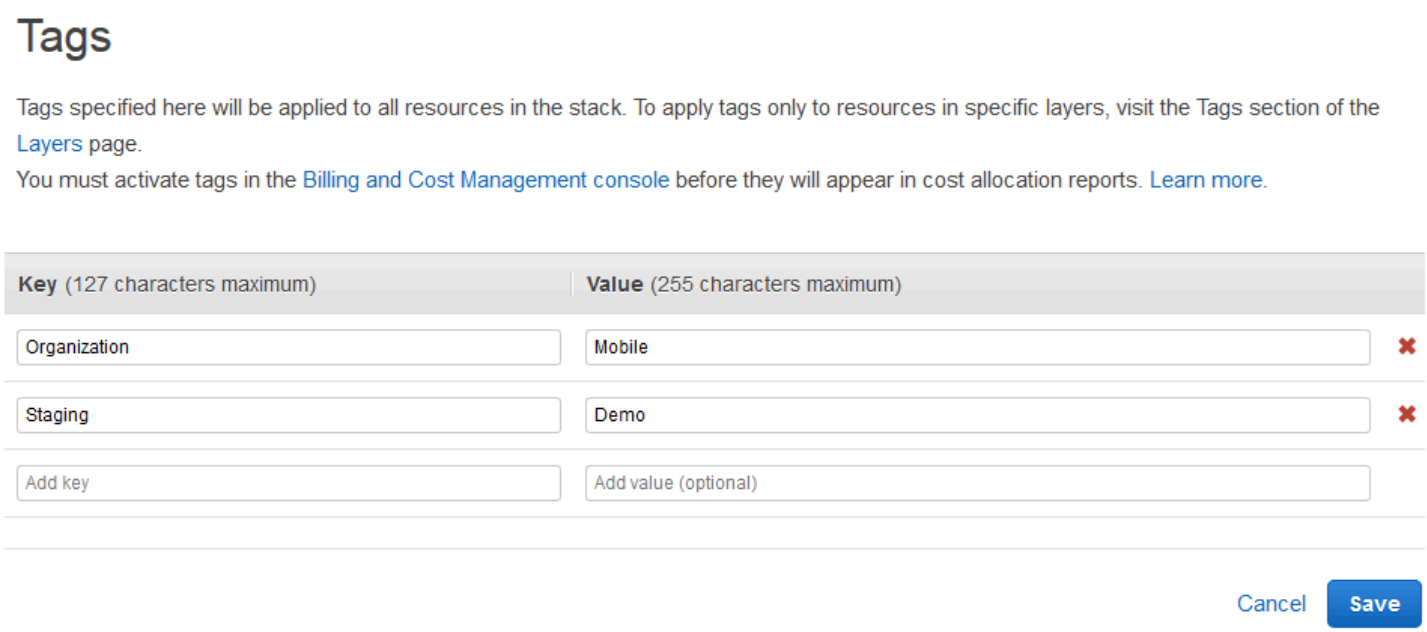
- 스택 수준에서 태그 설정
- 계층 수준에서 태그 설정
- 를 사용하여 태그 관리 AWS CLI
- 태그 제한

스택 수준에서 태그 설정

스택 수준에서는 스택의 홈 페이지에서 [태그]를 선택하여 태그를 추가하고 관리할 수 있습니다.

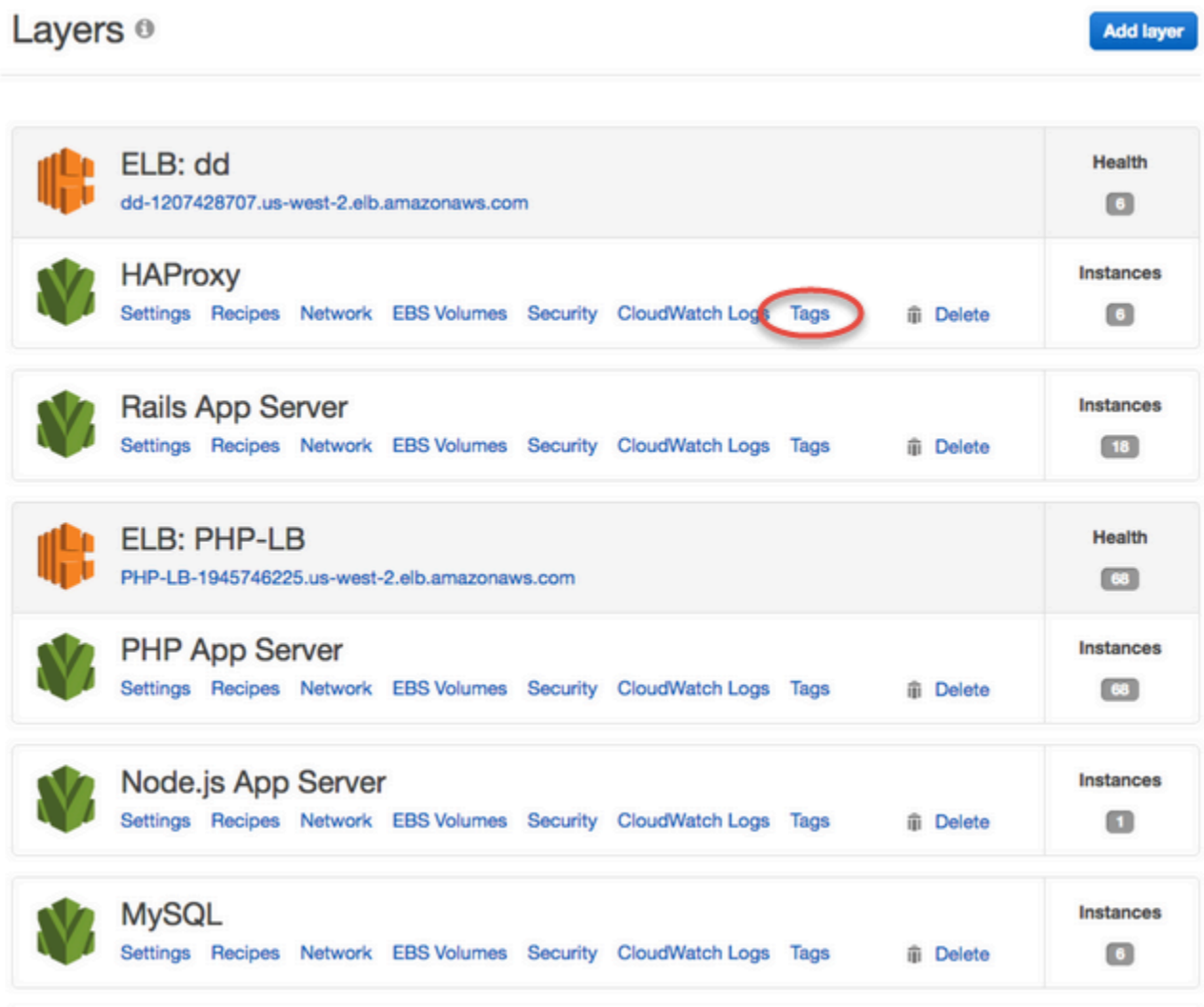


[태그] 페이지에서 키-값 쌍으로 태그를 추가하세요. 다음 스크린샷에는 몇몇 예시 태그가 나와 있습니다. 키-값 쌍의 오른쪽에 있는 빨간색 [X]를 선택하여 태그를 삭제할 수 있습니다.

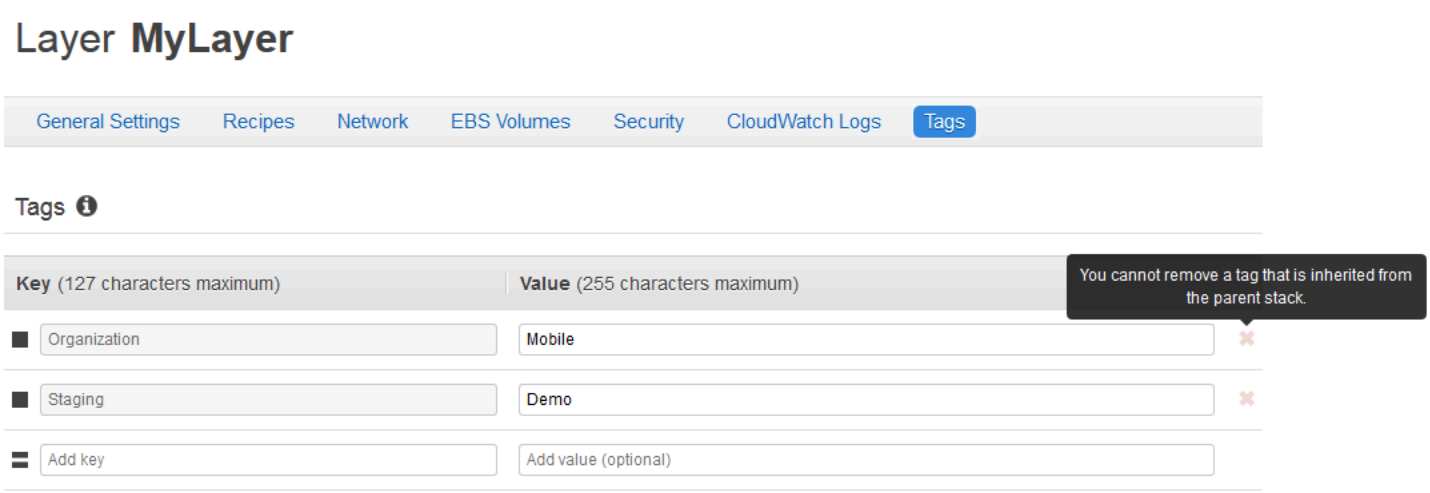


계층 수준에서 태그 설정

계층 수준에서 [태그] 탭을 선택하여 태그를 설정합니다. 이 탭은 계층 홈 페이지와 각 개별 계층의 홈 페이지에서 찾을 수 있습니다.



계층 수준에서 태그를 변경하거나 추가할 때는 상위 스택 수준에서 추가된 태그가 계층 및 리소스에 의해 상속된다는 점에 유의하세요. 상속된 태그의 값은 변경할 수 있지만 키 이름을 변경하거나 상속된 태그를 삭제할 수는 없습니다. 키 이름 변경이나 상위 스택에서 상속된 태그 삭제는 스택 설정에서 수행하세요. 다음 스크린샷은 스택 수준에서 상속된 태그의 예를 보여 줍니다. 상속된 태그는 회색으로 표시되어 있습니다.



스택에 태그를 추가하는 방법에 대한 자세한 내용은 새 스택 생성 단원을 참조하세요. 계층에 태그를 추가하는 방법에 대한 자세한 정보는 OpsWorks 계층의 구성 편집 단원을 참조하세요.

를 사용하여 태그 관리 AWS CLI

AWS CLI 명령을 사용하여 스택 및 계층 수준에서 태그를 추가하고 제거할 수도 있습니다. 다운로드 및 설치에 대한 자세한 내용은 명령줄 인터페이스 설치를 AWS CLI참조하세요. AWS 태그를 지정하려는 스택이 기본 리전에 있지 않은 경우, 명령에 `--region` 파라미터를 추가해야 함을 명심하세요. 계층 ARNs 현재에 표시되지 않습니다 AWS Management Console. 계층의 ARN을 가져오려면 describe-layers 명령을 실행합니다.

를 사용하여 태그를 추가하려면 AWS CLI

- AWS CLI 명령 프롬프트에서 다음 명령을 입력하여 *stack_or_layer_ARN*을 바꾸고 키-값 페어 태그를 지정한 다음 Enter 키를 누릅니다. 큰따옴표는 백슬래시로 이스케이프됩니다.

```
aws opsworks tag-resource --resource-arn stack_or_layer_ARN --tags "{\"key\":
\"value\",\"key\":\"value\"}"
```

다음은 예입니다.

```
aws opsworks tag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tags "{\"Stage\":
\"Production\",\"Organization\":\"Mobile\"}"
```

를 사용하여 태그를 제거하려면 AWS CLI

- AWS CLI 명령 프롬프트에서 다음을 입력한 다음 Enter 키를 누릅니다.

```
aws opsworks untag-resource --resource-arn stack_or_layer_ARN --tag-keys "[\"key\",
\"key\"]"
```

태그를 제거할 때는 제거하려는 태그의 키만 지정합니다. 다음은 예입니다.

```
aws opsworks untag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tag-keys "[\"Stage
\",\"Organization\"]"
```

> **Note**
>
> 계층에서 상속된 태그(상위 스택 레벨에서 추가된 태그)는 제거할 수 없습니다. 대신에 그 스택에서 상속된 태그를 제거하세요.

태그 제한

태그를 생성할 때는 다음의 제한을 염두에 두십시오.

- AWS OpsWorks Stacks는 상위 수준에서 상속된 사용자 정의 태그를 포함하여 스택 및 계층 수준에서 사용자 정의 태그 수를 40개로 제한합니다. 이렇게 하면 앞에가 붙은 기본 태그opsworks:와 다른 AWS 프로세스에 의해 설정된 태그에 사용 가능한 슬롯 10개가 남습니다. 리소스에는에서 생성한 사용자 정의 태그와 기본 태그를 모두 포함하여 최대 50개의 태그가 허용됩니다 AWS.
- 태그 키는 **aws:**, **opsworks:** 또는 **rds:**로 시작할 수 없습니다. **Name**는 Stacks에서 예약하므로 **name** 또는 AWS OpsWorks 를 태그 키**Name**로 사용하지 마십시오.
- 키는 최대 127자로서 유니코드 문자, 숫자 또는 구분 기호 또는 특수 문자(+ - = . _ : /)만 포함할 수 있습니다.
- 값은 최대 255자로서 유니코드 문자, 숫자 또는 구분 기호 또는 특수 문자(+ - = . _ : /)만 포함할 수 있습니다.

모니터링

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택을 모니터링하는 방법은 다음과 같습니다.

- AWS OpsWorks Stacks는 Amazon CloudWatch를 사용하여 스택의 각 인스턴스에 대한 세부 모니터링과 함께 13개의 사용자 지정 지표를 제공합니다.
- AWS OpsWorks Stacks는와 통합되어 모든 AWS OpsWorks Stacks API 호출을 AWS CloudTrail 기록하고 Amazon S3 버킷에 데이터를 저장합니다.
- Amazon CloudWatch Logs를 사용하여 스택의 시스템, 애플리케이션 및 사용자 지정 로그를 모니터링할 수 있습니다.

주제

- Amazon CloudWatch를 사용하여 스택 모니터링
- 를 사용하여 AWS OpsWorks Stacks API 호출 로깅 AWS CloudTrail
- AWS OpsWorks Stacks에서 Amazon CloudWatch Logs 사용
- Amazon CloudWatch Events를 사용하여 스택 모니터링

Amazon CloudWatch를 사용하여 스택 모니터링

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Amazon CloudWatch(CloudWatch)를 사용하여 스택에 대한 모니터링을 제공합니다.

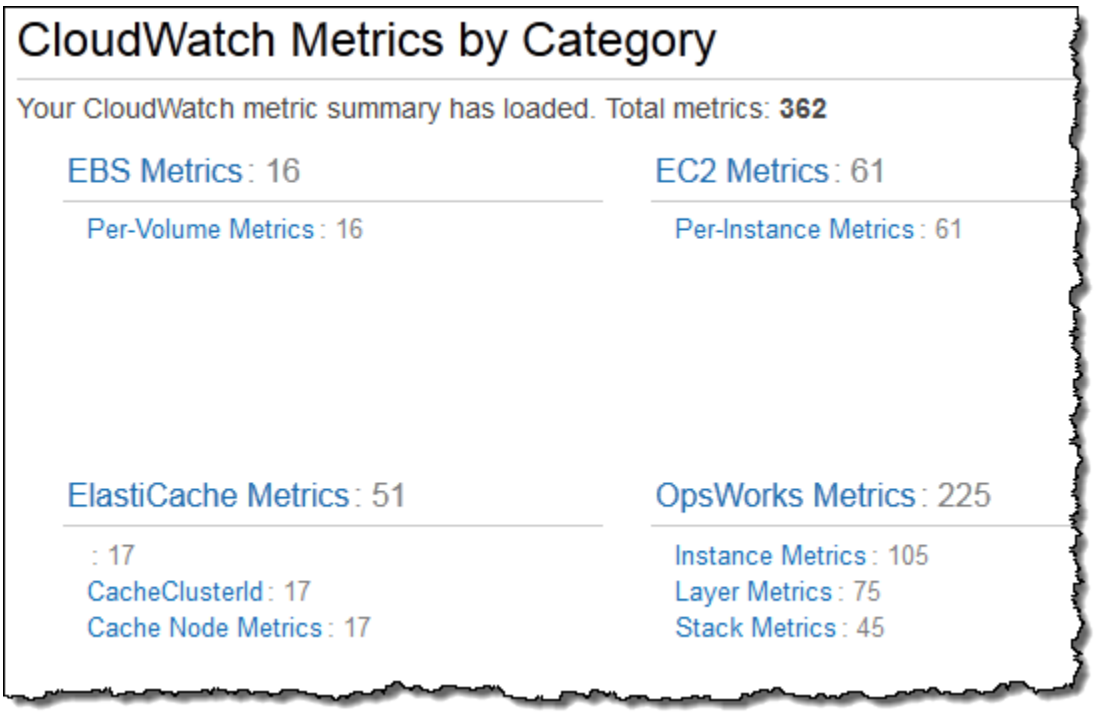
- Linux 스택의 경우 AWS OpsWorks , 스택은 스택의 각 인스턴스에 대한 세부 모니터링을 제공하는 13개의 사용자 지정 지표를 지원하고 모니터링 페이지에서 사용자의 편의를 위해 데이터를 요약합니다.
- Windows 스택의 경우 CloudWatch 콘솔을 사용하여 인스턴스에 대한 표준 Amazon EC2 지표를 모니터링할 수 있습니다.

  모니터링 페이지에 Windows 지표는 표시되지 않습니다.

모니터링 페이지에는 전체 스택, 계층 또는 인스턴스에 대한 지표가 표시됩니다. AWS OpsWorks 스택 지표는 Amazon EC2 지표와 다릅니다. CloudWatch 콘솔을 통해 추가 지표를 활성화할 수도 있지만, 일반적으로 추가 요금이 부과됩니다. 또한 다음과 같이 CloudWatch 콘솔에서 기본 데이터를 볼 수 있습니다.

CloudWatch에서 OpsWorks 사용자 지정 지표를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 모음에서 스택의 리전을 선택합니다.
3. 탐색 창에서 [지표]를 선택합니다.
4. OpsWorks 지표에서 인스턴스 지표, 계층 지표 또는 스택 지표를 선택합니다.



> **Note**
>
> AWS OpsWorks Stacks는 각 인스턴스(인스턴스 에이전트)에서 프로세스를 실행하여 지표를 수집합니다. CloudWatch는 하이퍼바이저를 사용하여 지표를 다르게 수집하므로 CloudWatch 콘솔의 값이 AWS OpsWorks Stacks 콘솔의 모니터링 페이지에 있는 해당 값과 약간 다를 수 있습니다.

CloudWatch 콘솔을 사용하여 경보를 설정할 수도 있습니다. 경보를 생성하는 방법에 대한 자세한 내용은 Amazon CloudWatch 경보 생성을 참조하세요. CloudWatch 사용자 지정 지표 목록은 AWS OpsWorks 지표 및 차원을 참조하세요. 자세한 내용은 Amazon CloudWatch를 참조하세요.

주제

- AWS OpsWorks 스택 지표
- AWS OpsWorks 스택 지표의 차원
- 스택 지표
- 계층 지표
- 인스턴스 지표

AWS OpsWorks 스택 지표

AWS OpsWorks Stacks는 5분마다 CloudWatch로 다음 지표를 전송합니다.

CPU 지표

지표	설명
cpu_idle	CPU가 유휴 상태인 시간 비율 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
cpu_nice	CPU가 양의 nice 값을 갖는, 즉 예약 우선순위가 비교적 낮은 프로세스를 처리하는 시간 비율. 이 지표에 대한 자세한 내용은 nice(Unix)를 참조하세요. 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
cpu_steal	AWS에서는 점차 증가하는 인스턴스에 하이퍼바이저 CPU 리소스를 할당하기 때문에 가상화 로드의 문제가 발생할 뿐 아니라 하이퍼바이저가 요청된 작업을 인스턴스에서 실행할 수 있는 빈도에도 영향을 미칠 수 있습니다. cpu_steal 은 하이퍼바이저가 물리적 CPU 리소스를 할당할 때까지 인스턴스가 대기하는 시간 비율을 측정합니다. 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId)

지표	설명
	유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
cpu_system	CPU가 시스템 작업을 처리하는 시간 비율 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
cpu_user	CPU가 사용자 작업을 처리하는 시간 비율 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
cpu_waitio	CPU가 입/출력 작업을 위해 대기하는 시간 비율 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음

메모리 지표

지표	설명
memory_buffers	버퍼 메모리 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
memory_cached	캐시 메모리 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
memory_free	여유 메모리 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
memory_swap	스왑 공간 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples

지표	설명
	단위: 없음
memory_total	총 메모리 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
memory_used	사용 중인 메모리 크기 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음

로드 지표

지표	설명
load_1	1분간 평균 로드 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: Average, Minimum, Maximum, Sum 또는 Data Samples 단위: 없음
load_5	5분간 평균 로드

지표	설명
	유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: `Average`, `Minimum`, `Maximum`, `Sum` 또는 `Data Samples` 단위: 없음
`load_15`	15분간 평균 로드 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: `Average`, `Minimum`, `Maximum`, `Sum` 또는 `Data Samples` 단위: 없음

프로세스 지표

지표	설명
`procs`	활성 프로세스 수 유효 차원: 지표가 표시되는 각 리소스의 ID(StackId, LayerId 또는 InstanceId) 유효한 통계: `Average`, `Minimum`, `Maximum`, `Sum` 또는 `Data Samples` 단위: 없음

AWS OpsWorks 스택 지표의 차원

AWS OpsWorks Stacks 지표는 AWS OpsWorks Stacks 네임스페이스를 사용하고 다음 차원에 대한 지표를 제공합니다.

차원	설명
StackId	특정 스택의 평균값입니다.
LayerId	특정 계층의 평균값입니다.
InstanceId	특정 인스턴스의 평균값입니다.

스택 지표

전체 스택에 대한 지표 요약을 보려면 스택 대시보드에서 AWS OpsWorks 스택을 선택한 다음 탐색 창에서 모니터링을 클릭합니다. 다음은 PHP 및 DB 계층을 포함하는 스택에 대한 예제입니다.

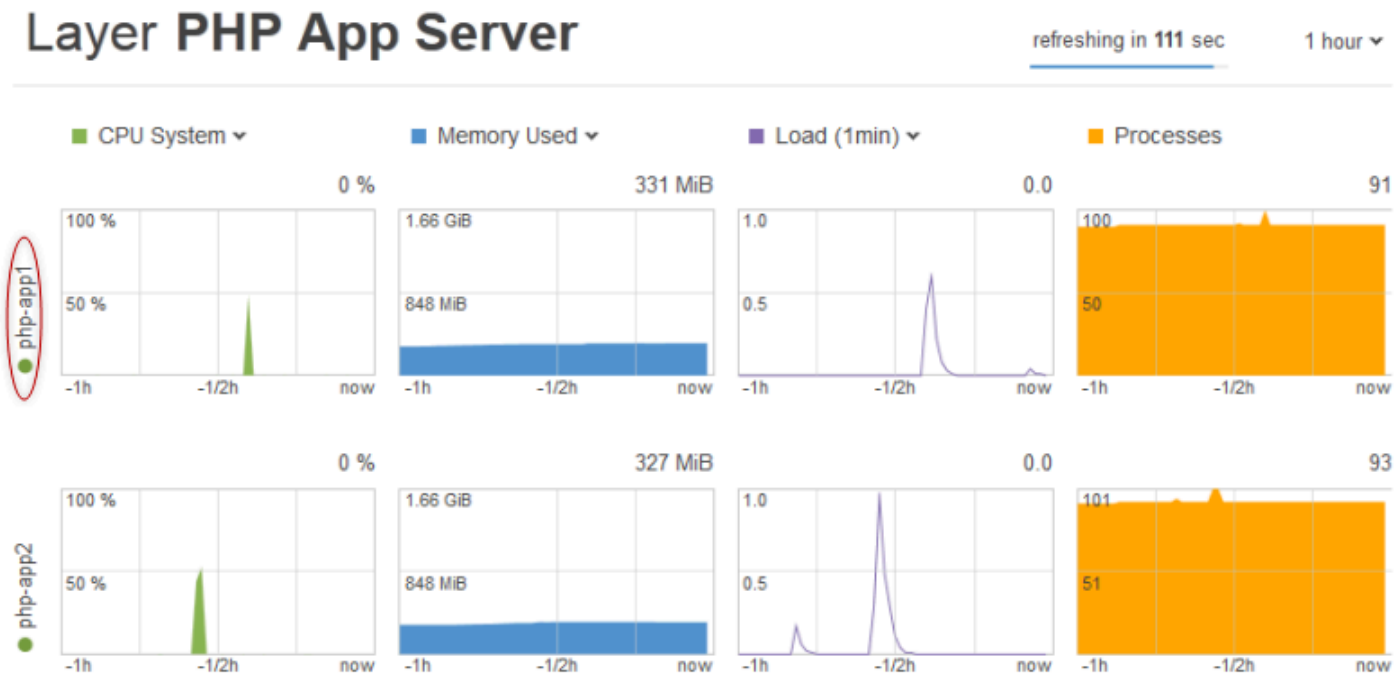
스택 보기는 지정된 기간(1시간, 8시간, 24시간, 1주 또는 2주) 동안 각 계층에 대해 4개 유형의 지표를 그래프로 표시합니다. 다음 사항에 유의하세요.

- AWS OpsWorks Stacks는 그래프를 주기적으로 업데이트합니다. 오른쪽 상단의 카운트다운 타이머는 다음 업데이트까지 남은 시간을 나타냅니다.
- 계층에 여러 인스턴스가 있는 경우 그래프에는 해당 계층의 평균값이 표시됩니다.
- 오른쪽 상단에서 목록을 클릭하고 원하는 값을 선택하여 기간을 지정할 수 있습니다.

지표 유형별로 그래프 상단의 목록을 사용하여 보고자 하는 특정 지표를 선택할 수 있습니다.

계층 지표

특정 계층의 지표를 보려면 계층 모니터링 보기에서 계층 이름을 클릭합니다. 다음 예는 인스턴스 두 개가 있는 PHP 계층의 지표를 보여줍니다.

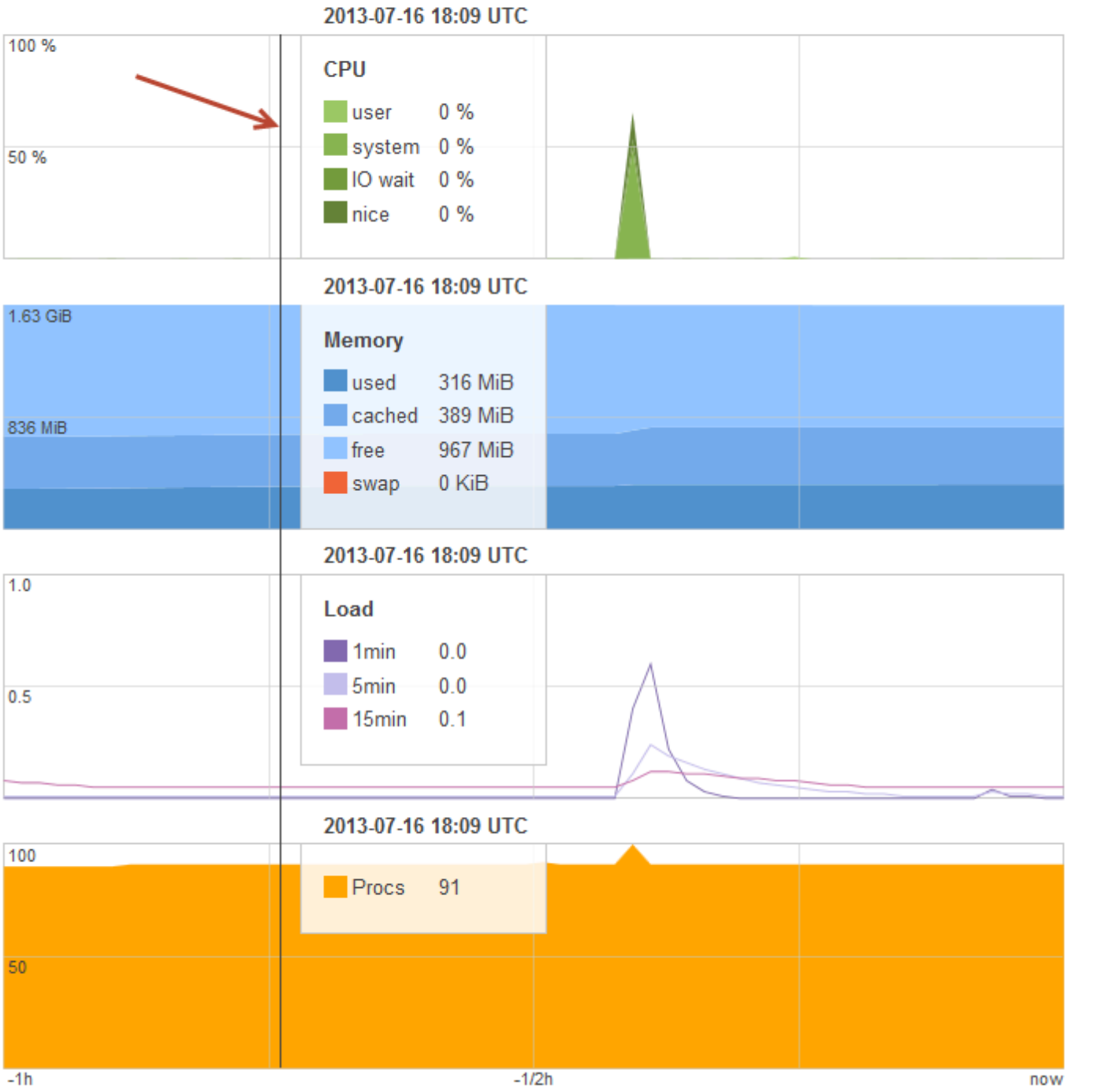


지표 유형은 스택 지표와 동일하며, 유형별로 그래프 상단의 목록을 사용하여 보고자 하는 특정 지표를 선택할 수 있습니다.

> **Note**
>
> 또한 계층 세부 정보 페이지로 이동하여 오른쪽 상단에서 모니터링을 클릭하면 계층 지표를 표시할 수 있습니다.

인스턴스 지표

특정 인스턴스의 지표를 보려면 계층 모니터링 보기에서 인스턴스 이름을 클릭합니다. 다음 예는 PHP 계층 내 php-app1 인스턴스의 지표를 보여줍니다.


Instance php-app1
refreshing in
2013-07-16 18:09 UTC
100 %
50 %
CPU
user 0 %
system 0 %
IO wait 0 %
nice 0 %
2013-07-16 18:09 UTC
1.63 GiB
836 MiB
Memory
used 316 MiB
cached 389 MiB
free 967 MiB
swap 0 KiB
2013-07-16 18:09 UTC
1.0
0.5
Load
1min 0.0
5min 0.0
15min 0.1
2013-07-16 18:09 UTC
100
50
Procs 91
-1h
-1/2h
now

이 그래프는 지표 유형별로 사용 가능한 모든 지표를 요약한 것입니다. 특정 시점의 정확한 값을 확인하려면 마우스를 사용하여 (이전 그림에서 빨간색 화살표로 표시된) 슬라이더를 원하는 위치로 이동합니다.

> **Note**
>
> 또한 인스턴스 세부 정보 페이지로 이동하여 오른쪽 상단에서 모니터링을 클릭하면 인스턴스 지표를 표시할 수 있습니다.

를 사용하여 AWS OpsWorks Stacks API 호출 로깅 AWS CloudTrail

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 IAM 자격 증명 AWS CloudTrail에서 수행한 작업의 레코드를 제공하는 서비스 또는 AWS OpsWorks Stacks의 AWS 서비스와 통합됩니다. CloudTrail은 AWS OpsWorks Stacks 콘솔의 호출 및 AWS OpsWorks Stacks API에 대한 코드 호출을 포함하여 AWS OpsWorks Stacks에 대한 모든 APIs. 추적을 생성하면 AWS OpsWorks Stacks 이벤트를 포함하여 CloudTrail 이벤트를 Amazon S3 버킷으로 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS OpsWorks Stacks에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

AWS OpsWorks CloudTrail의 스택 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. AWS OpsWorks Stacks에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

AWS OpsWorks Stacks에 대한 이벤트를 포함하여 AWS 계정의 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 AWS OpsWorks Stacks 작업은 CloudTrail에서 로깅되며 AWS OpsWorks Stacks API 참조에 문서화됩니다. 예를 들어 `CreateLayer`, `DescribeInstances` 및 `StartInstance` 작업을 직접적으로 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 IAM 사용자 보안 인증 정보로 했는지 여부.
- 역할 또는 페더레이션 사용자의 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청을 했는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

AWS OpsWorks Stacks 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음은 `CreateLayer` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예시입니다.

```
    {
  "Records": [
```

```
        {
            "awsRegion": "us-west-2",
            "eventID": "342cd1ec-8214-4a0f-a68f-8e6352feb5af",
            "eventName": "CreateLayer",
            "eventSource": "opsworks.amazonaws.com",
            "eventTime": "2014-05-28T16:05:29Z",
            "eventVersion": "1.01"ed,
            "requestID": "e3952a2b-e681-11e3-aa71-81092480ee2e",
            "requestParameters": {
                "attributes": {},
                "customRecipes": {},
                "name": "2014-05-28 16:05:29 +0000 a073",
                "shortname": "customcf4571d5c0d6",
                "stackId": "a263312e-f937-4949-a91f-f32b6b641b2c",
                "type": "custom"
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1 x86_64-linux",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        },
        {
            "awsRegion": "us-west-2",
            "eventID": "a860d8f8-c1eb-449b-8f55-eafc373b49a4",
            "eventName": "DescribeInstances",
            "eventSource": "opsworks.amazonaws.com",
            "eventTime": "2014-05-28T16:05:31Z",
            "eventVersion": "1.01",
            "requestID": "e4691bfd-e681-11e3-aa71-81092480ee2e",
            "requestParameters": {
                "instanceIds": [
                    "218289c4-0492-473d-a990-3fbe1efa25f6"
                ]
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1x86_64-linux",
```

```
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        }
    ]
}
```

AWS OpsWorks Stacks에서 Amazon CloudWatch Logs 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

여러 인스턴스에서 로그를 모니터링하는 프로세스를 간소화하기 위해 AWS OpsWorks Stacks는 Amazon CloudWatch Logs를 지원합니다. AWS OpsWorks Stacks의 계층 수준에서 CloudWatch Logs를 활성화합니다. CloudWatch Logs 통합은 Chef 11.10 및 Chef 12 Linux 기반 스택에서 작동합니다. CloudWatch Logs를 활성화하면 추가 비용이 발생하므로 시작하기 전에 Amazon CloudWatch 요금을 검토하세요.

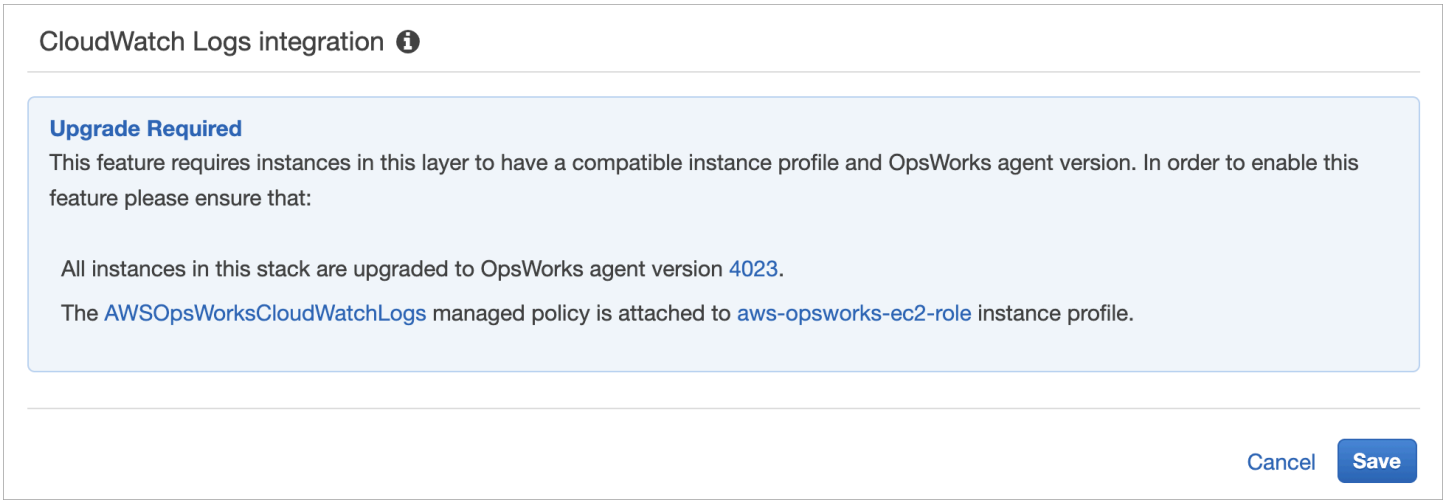
CloudWatch Logs는 선택한 로그에서 사용자가 지정한 패턴의 발생을 모니터링합니다. 예를 들어 로그에서 `NullReferenceException`과 같은 리터럴 용어의 발생을 모니터링하거나 이러한 발생 횟수를 계산할 수 있습니다. AWS OpsWorks Stacks에서 CloudWatch Logs를 활성화하면 AWS OpsWorks Stacks 에이전트가 CloudWatch Logs로 로그를 전송합니다. CloudWatch Logs에 대한 자세한 내용은 CloudWatch Logs 시작하기를 참조하세요.

사전 조건

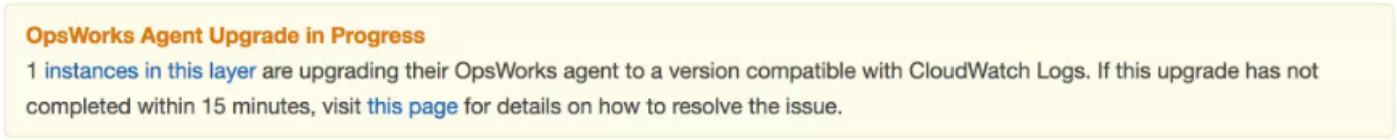
CloudWatch Logs를 활성화하려면 먼저 인스턴스가 Chef 11.10 스택에서는 AWS OpsWorks Stacks 에이전트 버전 3444 이상을, Chef 12 스택에서는 버전 4023 이상을 실행 중이어야 합니다. 또한 CloudWatch Logs를 사용하여 모니터링하는 인스턴스에 호환되는 인스턴스 프로파일을 사용해야 합니다.

사용자 지정 인스턴스 프로파일(스택을 생성할 때 AWS OpsWorks Stacks에서 제공하지 않은 프로파일)을 사용하는 경우 AWS OpsWorks Stacks는 인스턴스 프로파일을 자동으로 업그레이드할 수 없습니다. IAM을 사용하여 AWSOpsWorksCloudWatchlogs 정책을 프로파일에 수동으로 연결해야 합니다. 자세한 내용은 IAM 사용 설명서의 관리형 IAM 정책을 참조하세요.

에이전트 버전 또는 인스턴스 프로파일을 업그레이드해야 하는 경우 계층 페이지에서 CloudWatch Logs 탭을 열 때 AWS OpsWorks Stacks는 다음 스크린샷과 유사한 알림을 표시합니다.



계층의 모든 인스턴스에서 에이전트를 업데이트하려면 시간이 다소 소요될 수 있습니다. 에이전트 업그레이드가 완료되기 전에 계층에서 CloudWatch Logs를 활성화하면 다음과 비슷한 메시지가 표시됩니다.



CloudWatch Logs 활성화

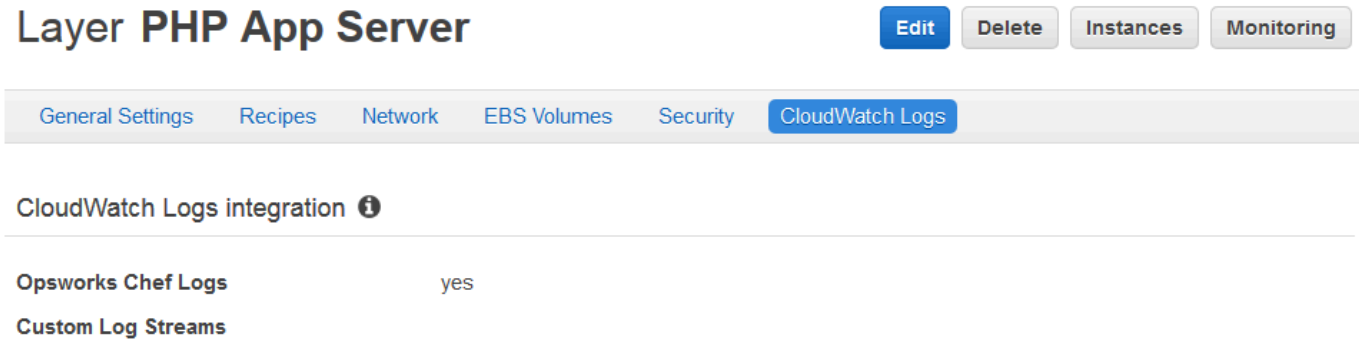
1. 필요한 에이전트 및 인스턴스 프로파일 업그레이드가 완료되면 CloudWatch Logs 탭의 슬라이더 컨트롤을 On으로 설정하여 CloudWatch Logs를 활성화할 수 있습니다.

2. 명령 로그를 스트리밍하려면 [명령 로그 스트리밍] 슬라이더를 [켜기]로 설정합니다. 이렇게 하면 계층의 인스턴스에서 Chef 활동 로그 및 사용자가 시작한 명령 로그가 CloudWatch Logs로 전송됩니다.

   이러한 로그에 포함된 데이터는 로그 URL의 대상을 열었을 때 DescribeCommands 작업의 결과에 표시되는 내용과 거의 일치합니다. 여기에는 `setup`, `configure`, `deploy`, `undeploy`, `start`, `stop` 및 레시피 실행 명령에 대한 데이터가 포함되어 있습니다.

3. 계층의 인스턴스에 있는 사용자 지정 위치에 저장되는 활동 로그(예: `/var/log/apache/myapp/mylog*`)를 스트리밍하려면 [명령 로그 스트리밍] 문자열 상자에 사용자 지정 위치를 입력한 다음[추가](+)를 선택합니다.

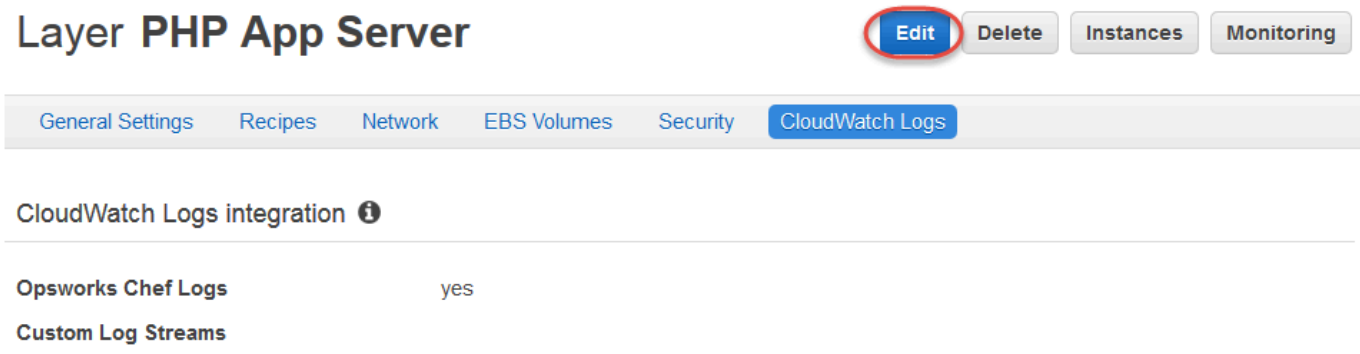
4. 저장(Save)을 선택합니다. 몇 분 내에 CloudWatch Logs 콘솔에 AWS OpsWorks 스택 로그 스트림이 표시되어야 합니다.



CloudWatch 로그 끄기

CloudWatch Logs를 끄려면 계층 설정을 편집하세요.

1. 계층의 속성 페이지에서 [편집]을 선택합니다.



2. 편집 페이지에서 [CloudWatch Logs] 탭을 선택합니다.
3. [CloudWatch Logs] 영역에서 [명령 로그 스트리밍]을 끕니다. 해당하는 경우 사용자 지정 로그에 표시된 X를 클릭하여 로그 스트림에서 해당 로그를 삭제합니다.
4. 저장을 선택합니다.

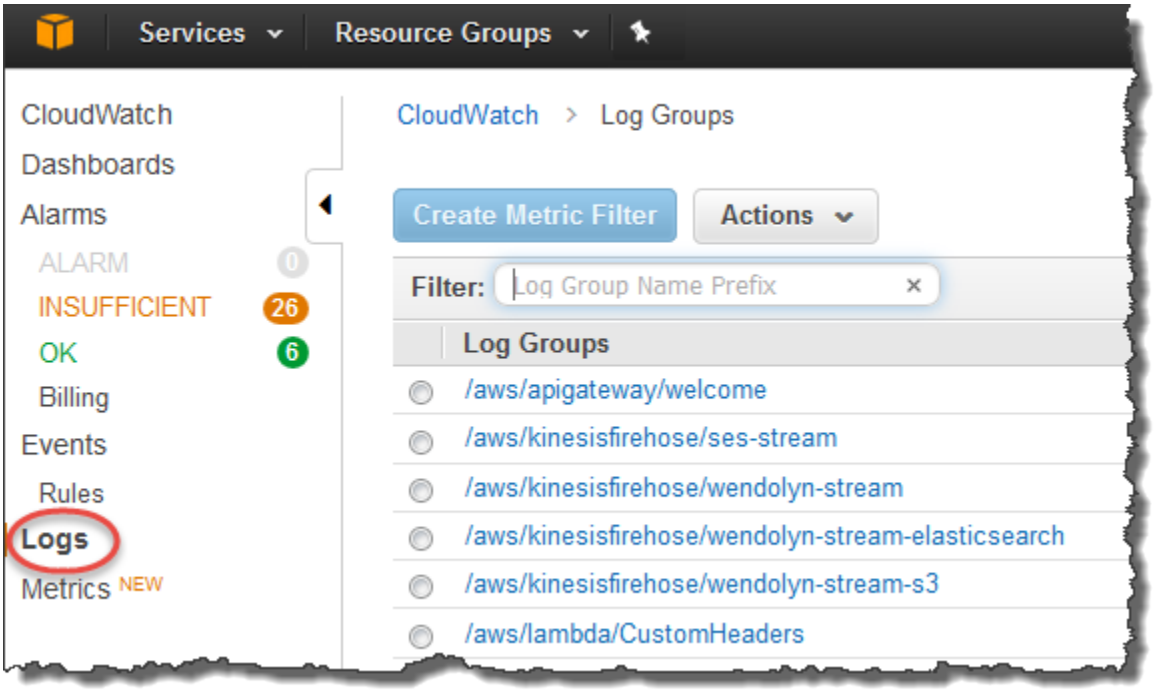
CloudWatch Logs에서 스트리밍된 로그 삭제

AWS OpsWorks Stacks에서 CloudWatch Logs 스트리밍을 끄면 CloudWatch Logs 관리 콘솔에서 기존 로그를 계속 사용할 수 있습니다. 로그를 Amazon S3로 내보내거나 삭제하지 않으면 저장된 로그에 대해 계속 비용이 발생합니다. S3로 로그를 내보내는 방법에 대한 자세한 내용은 Amazon S3로 로그 데이터 내보내기를 참조하세요.

CloudWatch Logs 관리 콘솔에서 또는 `delete-log-stream` 및 `delete-log-group` AWS CLI 명령을 실행하여 로그 스트림 및 로그 그룹을 삭제할 수 있습니다. 로그 보존 기간 변경에 대한 자세한 내용은 CloudWatch Logs에서 로그 데이터 보존 변경을 참조하세요.

CloudWatch Logs의 로그 관리

사용자가 스트리밍하는 로그는 CloudWatch Logs 콘솔에서 관리됩니다.



AWS OpsWorks 는 기본 로그 그룹 및 로그 스트림을 자동으로 생성합니다. AWS OpsWorks Stacks 데이터용 로그 그룹의 이름은 다음 패턴과 일치합니다.

stack_name/*layer_name*/*chef_log_name*

사용자 지정 로그의 이름은 다음 패턴과 일치합니다.

/stack_name/layer_short_name/file_path_name. 경로 이름은 별표(*) 같은 특수 문자를 제거하여 사람이 보다 읽기 쉽게 만들어집니다.

CloudWatch Logs에서 로그를 찾으면 로그를 그룹으로 구성하고, 지표 필터를 생성하여 로그를 검색 및 필터링하며, 사용자 지정 경보.를 생성할 수 있습니다.

CloudWatch Logs를 사용하도록 Chef 12.2 Windows 계층 구성

CloudWatch Logs 자동 통합은 Windows 기반 인스턴스에서는 지원되지 않습니다. Chef 12.2 스택의 계층에서는 [CloudWatch Logs] 탭을 사용할 수 없습니다. Windows 기반 인스턴스에 대한 CloudWatch Logs로의 스트리밍을 수동으로 활성화하려면 다음을 수행합니다.

- CloudWatch Logs 에이전트가 적절한 권한을 갖도록 Windows 기반 인스턴스의 인스턴스 프로파일을 업데이트합니다. [AWSOpsWorksCloudWatchLogs] 정책 설명에 어떤 권한이 필요한지 나와 있습니다.

일반적으로 이 작업은 한 번만 하면 됩니다. 그러면 계층의 모든 Windows 인스턴스에 업데이트된 인스턴스 프로파일을 사용할 수 있습니다.

- 각 인스턴스에서 다음의 JSON 구성 파일을 편집합니다. 이 파일에는 예컨대 어떤 로그를 모니터링할지와 같은 로그 스트림 기본 설정이 포함되어 있습니다.

  ```
  %PROGRAMFILES%\Amazon\Ec2ConfigService\Settings
  \AWS.EC2.Windows.CloudWatch.json
  ```

필요한 작업을 처리할 사용자 지정 레시피를 생성하여 Chef 12.2 계층의 [설정] 이벤트에 할당하면 앞의 두 작업을 자동화할 수 있습니다. 이러한 계층에서 새 인스턴스를 시작할 때마다 AWS OpsWorks Stacks는 인스턴스 부팅이 완료된 후 레시피를 자동으로 실행하여 CloudWatch Logs를 활성화합니다.

Windows 기반 인스턴스에서 CloudWatch Logs를 끄려면 이 프로세스를 거꾸로 수행합니다. EC2 서비스 속성 대화 상자에서 CloudWatch Logs 통합 활성화 확인란의 선택을 취소하고, `AWS.EC2.Windows.CloudWatch.json` 파일에서 로그 스트림 기본 설정을 삭제하고, Chef 12.2 계층의 새 인스턴스에 CloudWatch Logs 권한을 자동으로 할당하는 모든 Chef 레시피의 실행을 중지하세요.

Amazon CloudWatch Events를 사용하여 스택 모니터링

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon CloudWatch Events에서 규칙을 구성하여 AWS OpsWorks Stacks 리소스의 변경 사항을 알리고 CloudWatch Events가 이벤트 콘텐츠를 기반으로 작업을 수행하도록 지시할 수 있습니다. CloudWatch Events 시작하기 및 규칙 설정 방법에 대한 자세한 내용은 CloudWatch Events 사용자 가이드의 CloudWatch Events 시작하기를 참조하세요.

CloudWatch Events에서는 다음 AWS OpsWorks Stacks 이벤트 유형이 지원됩니다.

인스턴스 상태 변경

AWS OpsWorks Stacks 인스턴스의 상태 변경을 나타냅니다.

명령 상태 변경

AWS OpsWorks Stacks 명령의 상태에서 변경 사항이 발생했음을 나타냅니다.

배포 상태 변경

AWS OpsWorks Stacks 배포 상태에서 변경 사항이 발생했음을 나타냅니다.

알림

AWS OpsWorks Stacks 서비스 오류가 발생했음을 나타냅니다.

CloudWatch Events에서 지원하는 AWS OpsWorks Stacks 이벤트 유형에 대한 자세한 내용은 CloudWatch Events 사용 설명서의 AWS OpsWorks Stacks Events를 참조하세요. CloudWatch

보안 및 권한

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계정의 리소스에 액세스하려면 각 사용자에게 적절한 AWS 자격 증명이 AWS 있어야 합니다. 사용자에게 자격 증명을 제공하는 권장 방법은 AWS Identity and Access Management (IAM)을 사용하는 것입니다. AWS OpsWorks Stacks는 IAM과 통합되어 다음을 제어할 수 있습니다.

- 개별 사용자가 AWS OpsWorks Stacks와 상호 작용하는 방법.

  예를 들어 일부 사용자에게는 모든 스택에 앱을 배포할 수 있지만 스택 자체는 수정할 수 없도록 권한을 부여하고 다른 사용자에게는 일부 스택에 대해서만 전체 액세스를 허용하는 등의 제어가 가능합니다.
- AWS OpsWorks Stacks가 사용자를 대신하여 Amazon EC2 인스턴스 및 Amazon S3 버킷과 같은 스택 리소스에 액세스하는 방법.

  AWS OpsWorks Stacks는 이러한 작업에 대한 권한을 부여하는 서비스 역할을 제공합니다.
- AWS OpsWorks Stacks에서 제어하는 Amazon EC2 인스턴스에서 실행되는 앱이 Amazon S3 버킷에 저장된 데이터와 같은 다른 AWS 리소스에 액세스할 수 있는 방법.

인스턴스 프로파일을 계층의 인스턴스에 할당하여 해당 인스턴스에서 실행되는 앱에 다른 AWS 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다.

- 사용자 기반 SSH 키를 관리하고 SSH 또는 RDP를 사용하여 인스턴스에 연결하는 방법.

  각 스택에 대해, 관리 사용자는 각 사용자에게 개인 SSH 키를 할당하거나 사용자가 자신의 키를 지정하도록 허용할 수 있습니다. 또한 각 사용자에게 스택의 인스턴스에 대한 SSH 또는 RDP 액세스 및 sudo 또는 관리자 권한을 승인할 수 있습니다.

다음은 보안의 다른 측면입니다.

- 최신 보안 패치를 사용한 인스턴스 운영 체제 업데이트를 관리하는 방법.

  자세한 내용은 보안 업데이트 관리 단원을 참조하십시오.
- 인스턴스에 대한 양방향 네트워크 트래픽을 제어하기 위해 Amazon EC2 보안 그룹을 구성하는 방법.

  AWS OpsWorks Stacks 기본 보안 그룹 대신 사용자 지정 보안 그룹을 지정하는 방법. 자세한 내용은 보안 그룹 사용 단원을 참조하십시오.

주제

- AWS OpsWorks Stacks 사용자 권한 관리
- AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용
- AWS OpsWorks Stacks의 교차 서비스 혼동된 대리자 방지
- EC2 인스턴스에서 실행되는 앱에 대한 권한 지정
- SSH 액세스 관리
- Linux 보안 업데이트 관리
- 보안 그룹 사용

AWS OpsWorks Stacks 사용자 권한 관리

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

가장 좋은 방법은 AWS OpsWorks Stacks 사용자를 지정된 작업 세트 또는 스택 리소스 세트로 제한하는 것입니다. AWS OpsWorks Stacks 권한 페이지를 사용하고 적절한 IAM 정책을 적용하여 두 가지 방법으로 AWS OpsWorks Stacks 사용자 권한을 제어할 수 있습니다.

OpsWorks 권한 페이지 또는 이에 상응하는 CLI 또는 API 작업을 통해 각 사용자에게 여러 권한 수준 중 하나를 할당하여 다중 사용자 환경의 사용자 권한을 스택별로 제어할 수 있습니다. 각 수준은 특정 스택 리소스에 대해 표준적인 작업 세트를 수행할 수 있는 권한을 부여합니다. [권한] 페이지를 사용하여 다음을 제어할 수 있습니다.

- 각 스택에 액세스할 수 있는 사용자
- 각 사용자가 각 스택에서 수행하도록 허용된 작업

  예를 들어 일부 사용자는 스택을 볼 수만 있는 반면, 다른 사용자는 애플리케이션 배포, 인스턴스 추가 등의 작업을 수행할 수 있습니다.
- 각 스택을 관리할 수 있는 사용자

  하나 이상의 지정된 사용자에게 각 스택의 관리를 위임할 수 있습니다.
- 각 스택의 Amazon EC2 인스턴스에 대한 사용자 레벨 SSH 액세스 및 sudo 권한(Linux) 또는 RDP 액세스 및 관리자 권한(Windows)을 보유하는 사용자.

  언제라도 각 사용자별로 이러한 권한을 부여하거나 제거할 수 있습니다.

> **Important**
>
> SSH/RDP 액세스를 거부할 경우 반드시 해당 사용자가 인스턴스에 로그인할 수 없는 것은 아닙니다. 특정 인스턴스에 대해 Amazon EC2 키 페어를 지정하면 해당되는 프라이빗 키를 보유한 모든 사용자는 로그인하거나 Windows 관리자 암호를 검색할 수 있습니다. 자세한 내용은 SSH 액세스 관리 단원을 참조하십시오.

IAM 콘솔, CLI 또는 API를 사용하여 다양한 AWS OpsWorks Stacks 리소스 및 작업에 대한 명시적 권한을 부여하는 정책을 사용자에게 추가할 수 있습니다.

- IAM 정책으로 권한을 지정하는 것이 권한 수준을 사용하는 것보다 유연합니다.

- IAM ID(사용자, 사용자 그룹, 역할)를 설정하여 사용자 및 사용자 그룹과 같은 IAM ID에 권한을 부여하거나 페더레이션 사용자와 연결할 수 있는 역할을 정의할 수 있습니다.
- IAM 정책은 특정 키 AWS OpsWorks Stacks 작업에 대한 권한을 부여하는 유일한 방법입니다.

  예를 들어 각각 스택을 생성 및 복제하는 데 사용되는 `opsworks:CreateStack` 및 `opsworks:CloneStack`에 대한 권한을 부여하려면 IAM을 사용해야 합니다.

콘솔에서 페더레이션 사용자를 명시적으로 가져올 수는 없지만 페더레이션 사용자는 AWS OpsWorks Stacks 콘솔의 오른쪽 상단에서 내 설정을 선택한 다음 오른쪽 상단에서도 사용자를 선택하여 암시적으로 사용자 프로필을 생성할 수 있습니다. 사용자 페이지에서 페더레이션 사용자(API 또는 CLI를 사용하여 생성된 계정 또는 콘솔을 통해 묵시적으로 생성된 계정의 사용자)는 비 페더레이션 사용자와 비슷하게 계정을 관리할 수 있습니다.

두 방법은 상호 배타적이 아니며, 두 방법을 결합하는 것이 유용할 경우도 있습니다. 그러면 AWS OpsWorks Stacks가 두 권한 세트를 모두 평가합니다. 예를 통해 인스턴스 추가 또는 삭제는 허용하지만 계층 추가 또는 삭제는 허용하지 않으려는 경우를 가정해 봅시다. AWS OpsWorks Stacks 권한 수준은 해당 특정 권한 집합을 부여하지 않습니다. 하지만 권한 페이지를 사용하여 사용자에게 대부분의 스택 작업을 허용하는 관리 권한 수준을 부여한 후 계층을 추가 또는 제거할 수 있는 권한을 거부하는 IAM 정책을 적용할 수 있습니다. 자세한 내용은 정책을 사용한 AWS 리소스 액세스 제어를 참조하세요.

다음은 일반적인 사용자 권한 관리 모델입니다. 각 사례에서 사용자를 관리 사용자라고 가정합니다.

1. IAM 콘솔을 사용하여 AWSOpsWorks_FullAccess 정책을 하나 이상의 관리 사용자에게 연결합니다.
2. 어떤 AWS OpsWorks Stacks 권한도 부여하지 않는 정책을 사용하여 각 비관리 사용자에 대한 사용자를 생성합니다.

   사용자가 AWS OpsWorks Stacks에만 액세스해야 하는 경우 정책을 전혀 적용할 필요가 없을 수 있습니다. 대신 AWS OpsWorks 스택 권한 페이지를 사용하여 권한을 관리할 수 있습니다.
3. AWS OpsWorks Stacks 사용자 페이지를 사용하여 비관리 사용자를 AWS OpsWorks Stacks로 가져옵니다.
4. 각 스택에 대해 스택의 [권한] 페이지를 사용하여 각 사용자에게 권한 수준을 할당합니다.
5. 필요에 따라 적절히 구성된 IAM 정책을 적용하여 사용자의 권한 수준을 사용자 지정합니다.

사용자 관리에 대한 추가 권장 사항은 모범 사례: 권한 관리 섹션을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

주제

- AWS OpsWorks Stacks 사용자 관리
- AWS OpsWorks 스택 사용자 스택당 권한 부여
- IAM 정책을 연결하여 AWS OpsWorks 스택 권한 관리
- 정책 예제
- AWS OpsWorks 스택 권한 수준

AWS OpsWorks Stacks 사용자 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자를 AWS OpsWorks Stacks로 가져오고 권한을 부여하려면 먼저 각 개인에 대해 사용자를 생성해야 합니다. IAM 사용자를 생성하려면 IAMFullAccess 정책에 정의된 권한이 부여된 사용자로 AWS에 로그인합니다. 그런 다음 IAM 콘솔을 사용하여 AWS OpsWorks Stacks에 액세스해야 하는 모든 사용자를 위한 IAM 사용자를 생성합니다. 그런 다음 이러한 사용자를 AWS OpsWorks Stacks로 가져오고 다음과 같이 사용자 권한을 부여할 수 있습니다.

일반 AWS OpsWorks 스택 사용자

일반 사용자는 연결된 정책이 필요하지 않습니다. Stacks 권한이 있는 경우 일반적으로 AWS OpsWorks Stacks 권한이 포함되지 않습니다. 대신 AWS OpsWorks 스택 권한 페이지를 사용하여 stack-by-stack 일반 사용자에게 다음 권한 수준 중 하나를 할당합니다.

- [표시] 권한은 사용자가 스택을 볼 수는 있지만 어떤 작업도 수행할 수 없습니다.
- [배포] 권한은 [표시] 권한을 포함하며 사용자가 앱을 업데이트할 수 있습니다.
- [관리] 권한은 [배포] 권한을 포함하며 사용자가 계층 또는 인스턴스 추가와 같은 스택 관리 작업을 수행하고, [권한] 페이지를 사용하여 사용자 권한을 설정하고, 자체 SSH/RDP 및 sudo/admin 권한을 활성화할 수 있습니다.
- [거부] 권한은 스택 액세스를 거부합니다.

이러한 권한 수준은 특정 사용자에게 필요한 것이 아닐 경우 IAM 정책을 적용하여 사용자의 권한을 사용자 지정할 수 있습니다. 예를 들어 AWS OpsWorks 스택 권한 페이지를 사용하여 사용자에게 권한 관리 수준을 할당할 수 있습니다. 그러면 사용자에게 스택을 생성하거나 복제하지 않고 모든 스택 관리 작업을 수행할 수 있는 권한을 부여합니다. 그런 다음 계층 추가 또는 삭제는 권한을 거부하여 권한을 제한하거나 스택 생성 또는 복제를 허용하여 권한을 보강하는 정책을 적용할 수 있습니다. 자세한 내용은 IAM 정책을 연결하여 AWS OpsWorks 스택 권한 관리 단원을 참조하십시오.

AWS OpsWorks Stacks 관리 사용자

관리 사용자는 계정 소유자 또는 AWSOpsWorks_FullAccess 정책에 정의된 권한을 가진 IAM 사용자입니다. [관리] 사용자에게 부여된 권한 이외에 이 정책은 다음과 같이 [권한] 페이지를 통해 부여할 수 없는 작업에 대한 권한을 포함합니다.

- AWS OpsWorks Stacks로 사용자 가져오기
- 스택 생성 및 복제

전체 정책은 정책 예제 단원을 참조하세요. 사용자에게 IAM 정책을 적용하여 부여할 수 있는 권한의 상세 목록은 AWS OpsWorks 스택 권한 수준 섹션을 참조하세요.

주제

- 사용자 및 리전
- AWS OpsWorks Stacks 관리 사용자 생성
- AWS OpsWorks Stacks에 대한 IAM 사용자 생성
- AWS OpsWorks Stacks로 사용자 가져오기
- AWS OpsWorks Stacks 사용자 설정 편집

사용자 및 리전

AWS OpsWorks Stacks 사용자는 생성된 리전 엔드포인트 내에서 사용할 수 있습니다. 사용자를 다음 리전 중 하나에서 생성할 수 있습니다.

- US East (Ohio) Region
- 미국 동부(버지니아 북부) 리전
- US West (Oregon) Region
- US West (N. California) Region
- 캐나다(중부) 리전(API만 해당, 에서는 사용할 수 없음 AWS Management Console
- Asia Pacific (Mumbai) Region

- Asia Pacific (Singapore) Region
- 아시아 태평양(시드니) 리전
- 아시아 태평양(도쿄) 리전
- Asia Pacific (Seoul) Region
- Europe (Frankfurt) Region
- Europe (Ireland) Region
- Europe (London) Region
- Europe (Paris) Region
- South America (São Paulo) Region

사용자를 AWS OpsWorks Stacks로 가져올 때 리전 엔드포인트 중 하나로 가져옵니다. 둘 이상의 리전에서 사용자를 사용하려면 해당 리전으로 사용자를 가져와야 합니다. 한 리전에서 다른 리전으로 AWS OpsWorks Stacks 사용자를 가져올 수도 있습니다. 이미 이름이 같은 사용자가 있는 리전으로 사용자를 가져오면 가져온 사용자가 기존 사용자를 대체합니다. 사용자 가져오기에 대한 자세한 정보는 사용자 가져오기 단원을 참조하세요.

AWS OpsWorks Stacks 관리 사용자 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

정책을 사용자에게 추가하여 AWS OpsWorks Stacks 관리 사용자를 생성할 수 있습니다. `AWSOpsWorks_FullAccess` 그러면 해당 사용자에게 AWS OpsWorks Stacks 전체 액세스 권한이 부여됩니다. 관리 사용자 생성에 대한 자세한 내용은 관리 사용자 생성을 참조하세요.

> **Note**
>
> AWSOpsWorks_FullAccess 정책은 사용자가 AWS OpsWorks Stacks 스택을 생성 및 관리하도록 허용하지만 사용자가 스택에 대해 IAM 서비스 역할을 생성할 수 없습니다. 따라서 기존 역할을 사용해야 합니다. 관리 권한 섹션에서 설명한 대로 스택을 생성하는 첫 번째 사용자는 추가 IAM 권한이 있어야 합니다. 이 사용자가 첫 번째 스택을 생성하면 AWS

OpsWorks Stacks는 필요한 권한이 있는 IAM 서비스 역할을 생성합니다. 그런 다음에는 `opsworks:CreateStack` 권한을 가진 모든 사용자가 해당 역할을 사용해 추가 스택을 생성할 수 있습니다. 자세한 내용은 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 단원을 참조하십시오.

사용자를 생성할 때 필요에 따라 추가 고객 관리형 정책을 추가하여 사용자 권한을 세부적으로 조정할 수 있습니다. 예를 들어, 관리 사용자가 스택을 생성 또는 삭제할 수 있지만 새 사용자는 가져올 수 없도록 할 수 있습니다. 자세한 내용은 IAM 정책을 연결하여 AWS OpsWorks 스택 권한 관리 단원을 참조하십시오.

여러 관리 사용자가 있는 경우 각 사용자마다 권한을 설정하는 대신 AWSOpsWorks_FullAccess 정책을 IAM 그룹에 추가하고 사용자를 이 그룹에 추가할 수 있습니다.

그룹 생성에 대한 자세한 내용은 IAM 사용자 그룹 생성을 참조하세요. 그룹을 생성할 때 AWSOpsWorks_FullAccess 정책을 추가하세요. AWSOpsWorks_FullAccess 권한을 포함하는 AdministratorAccess 정책을 추가할 수도 있습니다.

기존 그룹에 권한을 추가하는 방법에 대한 자세한 내용은 정책을 IAM 사용자 그룹에 연결을 참조하세요.

AWS OpsWorks Stacks에 대한 IAM 사용자 생성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

IAM 사용자를 AWS OpsWorks Stacks로 가져오려면 먼저 사용자를 생성해야 합니다. 이 작업은 IAM 콘솔, 명령줄 또는 API를 사용하여 수행할 수 있습니다. 전체 지침은 AWS 계정에서 IAM 사용자 생성을 참조하세요.

관리 사용자와 달리, 정책을 연결하여 권한을 정의할 필요가 없습니다. 사용자 권한 관리에 설명된 대로 사용자를 AWS OpsWorks 스택으로 가져온 후 권한을 설정할 수 있습니다.

IAM 사용자 및 그룹 생성에 대한 자세한 내용은 IAM 시작하기를 참조하세요.

AWS OpsWorks Stacks로 사용자 가져오기

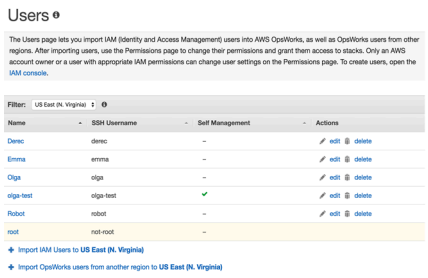
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

관리 사용자는 사용자를 AWS OpsWorks Stacks로 가져올 수 있으며, AWS OpsWorks Stacks 사용자를 한 리전 엔드포인트에서 다른 리전 엔드포인트로 가져올 수도 있습니다. 사용자를 AWS OpsWorks Stacks로 가져올 때 AWS OpsWorks Stacks 리전 엔드포인트 중 하나로 가져옵니다. 사용자를 여러 리전에서 사용할 수 있게 하려면 사용자를 해당 리전으로 가져와야 합니다.

콘솔에서 페더레이션 사용자를 명시적으로 가져올 수는 없지만 페더레이션 사용자는 AWS OpsWorks Stacks 콘솔의 오른쪽 상단에서 내 설정을 선택한 다음 오른쪽 상단에서도 사용자를 선택하여 암시적으로 사용자 프로필을 생성할 수 있습니다. 사용자 페이지에서 페더레이션 사용자(API 또는 CLI를 사용하여 생성된 계정 또는 콘솔을 통해 묵시적으로 생성된 계정의 사용자)는 비 페더레이션 사용자와 비슷하게 계정을 관리할 수 있습니다.

사용자를 AWS OpsWorks Stacks로 가져오려면

1. 관리 사용자 또는 계정 소유자로 AWS OpsWorks Stacks에 로그인합니다.
2. 오른쪽 위에서 [사용자]를 선택하여 [사용자] 페이지를 엽니다.



3. <## ##>으로 IAM 사용자 가져오기를 선택하여 사용 가능하지만 아직 가져오지 않은 사용자를 표시합니다.

4. [모두 선택] 확인란을 선택하거나 개별 사용자를 한 명 이상 선택합니다. 마치면 [OpsWorks에 가져오기]를 선택합니다.

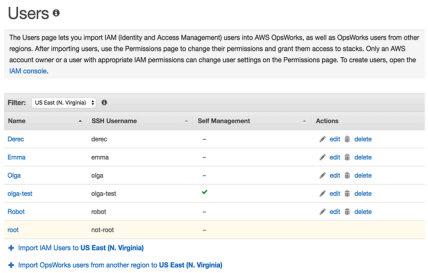
> **Note**
>
> 사용자를 AWS OpsWorks Stacks로 가져온 후 IAM 콘솔 또는 API를 사용하여 계정에서 사용자를 삭제하는 경우 사용자는 AWS OpsWorks Stacks를 통해 부여한 SSH 액세스 권한을 자동으로 잃지 않습니다. 또한 사용자 페이지를 열고 사용자의 작업 열에서 삭제를 선택하여 AWS OpsWorks Stacks에서 사용자를 삭제해야 합니다.

한 리전에서 다른 리전으로 AWS OpsWorks Stacks 사용자를 가져오려면

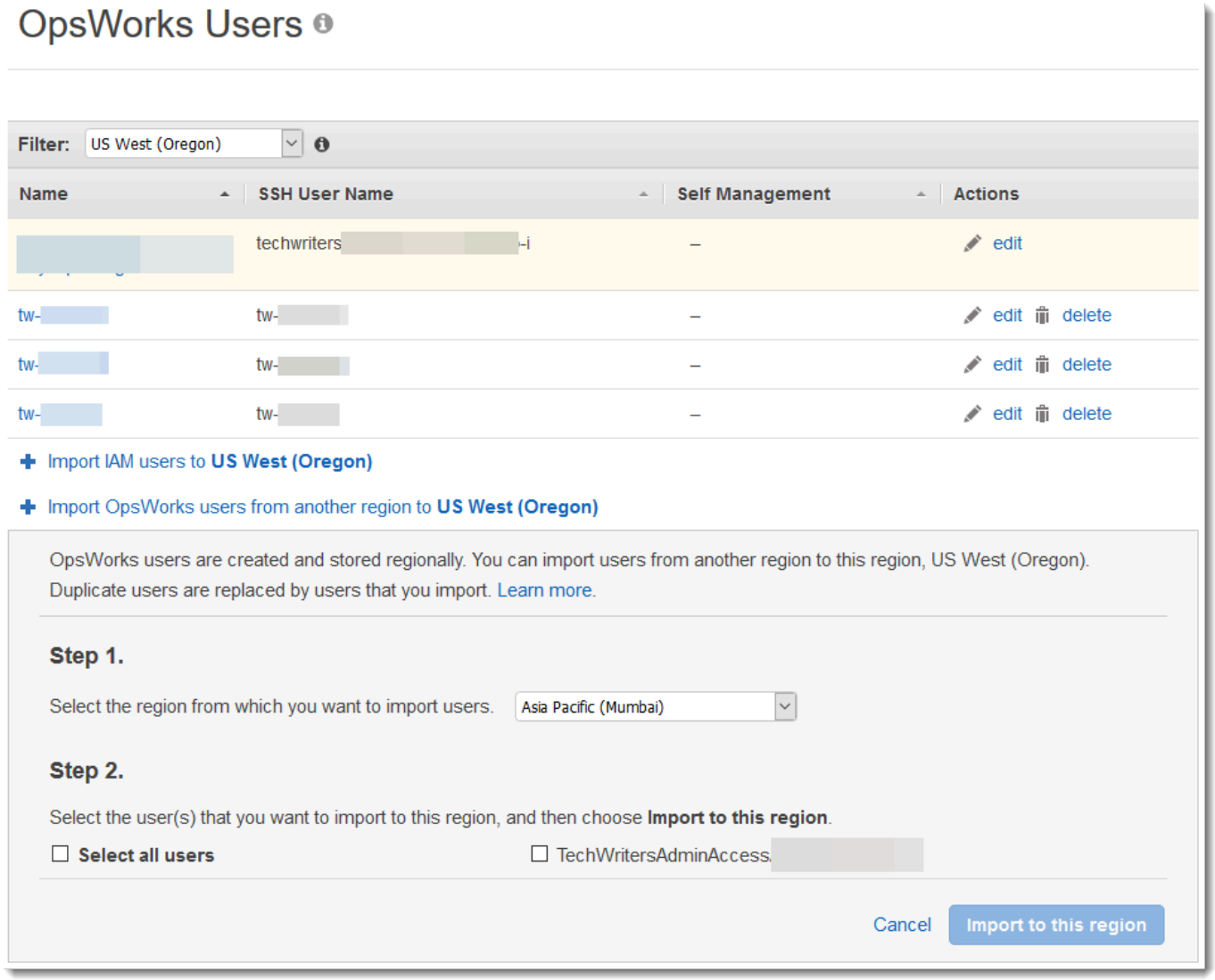
AWS OpsWorks Stacks 사용자는 생성된 리전 엔드포인트 내에서 사용할 수 있습니다. 사용자 및 리전에 표시된 리전 중 하나에서 사용자를 생성할 수 있습니다.

AWS OpsWorks Stacks 사용자를 한 리전에서 사용자 목록이 현재 필터링된 리전으로 가져올 수 있습니다. 사용자를 이미 동일한 이름의 사용자가 있는 리전으로 가져올 경우 가져온 사용자가 기존 사용자를 대체합니다.

1. 관리 사용자 또는 계정 소유자로 AWS OpsWorks Stacks에 로그인합니다.
2. 오른쪽 위에서 [사용자]를 선택하여 [사용자] 페이지를 엽니다. 둘 이상의 리전에 AWS OpsWorks Stacks 사용자가 있는 경우 필터 컨트롤을 사용하여 사용자를 가져오려는 리전을 필터링합니다.



3. 다른 리전에서 <## ##>으로 AWS OpsWorks 스택 사용자 가져오기를 선택합니다.



4. AWS OpsWorks Stacks 사용자를 가져올 리전을 선택합니다.
5. 가져올 사용자를 한 명 이상 선택하거나 모든 사용자를 선택한 다음 [이 리전으로 가져오기]을 선택합니다. AWS OpsWorks Stacks가 가져온 사용자를 사용자 목록에 표시할 때까지 기다립니다.

Unix IDs 및 외부 AWS OpsWorks 스택에서 생성된 사용자

AWS OpsWorks 는 AWS OpsWorks Stacks 인스턴스에서 2000~4000 사이의 Unix ID(UID) 값을 사용자에게 할당합니다. 는 2000-4000 범위의 UIDs를 AWS OpsWorks 예약하므로 외부에서 생성하는 사용자 AWS OpsWorks (예: 쿡북 레시피를 사용하거나 IAM AWS OpsWorks 에서 로 사용자를 가져옴)는 다른 사용자를 위해 Stacks에서 AWS OpsWorks 덮어쓰는 UIDs를 가질 수 있습니다. 이로 인해 AWS OpsWorks Stacks 외부에서 생성한 사용자가 데이터 백 검색 결과에 표시되지 않거나 AWS OpsWorks Stacks 기본 제공 `sync_remote_users` 작업에서 제외될 수 있습니다.

외부 프로세스는 AWS OpsWorks Stacks가 덮어쓸 수 있는 UIDs를 사용하여 사용자를 생성할 수도 있습니다. 예를 들어 일부 운영 체제 패키지는 설치 후 프로세스에서 사용자를 생성할 수 있습니다. 사용자 또는 소프트웨어 프로세스가 기본값인 UID를 명시적으로 지정하지 않고 Linux 기반 운영 체제에서 사용자를 생성하는 경우 AWS OpsWorks Stacks에서 할당한 UID는 *<## AWS OpsWorks UID# ## ##>* + 1입니다.

모범 사례는 AWS OpsWorks Stacks 콘솔에서 AWS CLI또는 AWS SDK를 사용하여 AWS OpsWorks Stacks 사용자를 생성하고 액세스를 관리하는 것입니다. 외부의 AWS OpsWorks Stacks 인스턴스에서 사용자를 생성하는 경우 4 AWS OpsWorks 000보다 큰 *UnixID* 값을 사용합니다.

AWS OpsWorks Stacks 사용자 설정 편집

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자를 가져온 후 다음과 같이 설정을 편집할 수 있습니다.

사용자 설정을 편집하려면

1. 사용자 페이지의 사용자 작업 열에서 편집을 선택합니다.
2. 다음 설정을 지정할 수 있습니다.

   자체 관리

   [예]를 선택하여 사용자가 [내 설정] 페이지에서 자신의 개인 SSH 키를 지정할 수 있도록 합니다.

   > **Note**
   >
   > 또한 DescribeMyUserProfile 및 UpdateMyUserProfile 작업에 대한 권한을 부여하는 IAM 정책을 IAM ID에 추가하여 자체 관리를 활성화할 수도 있습니다.

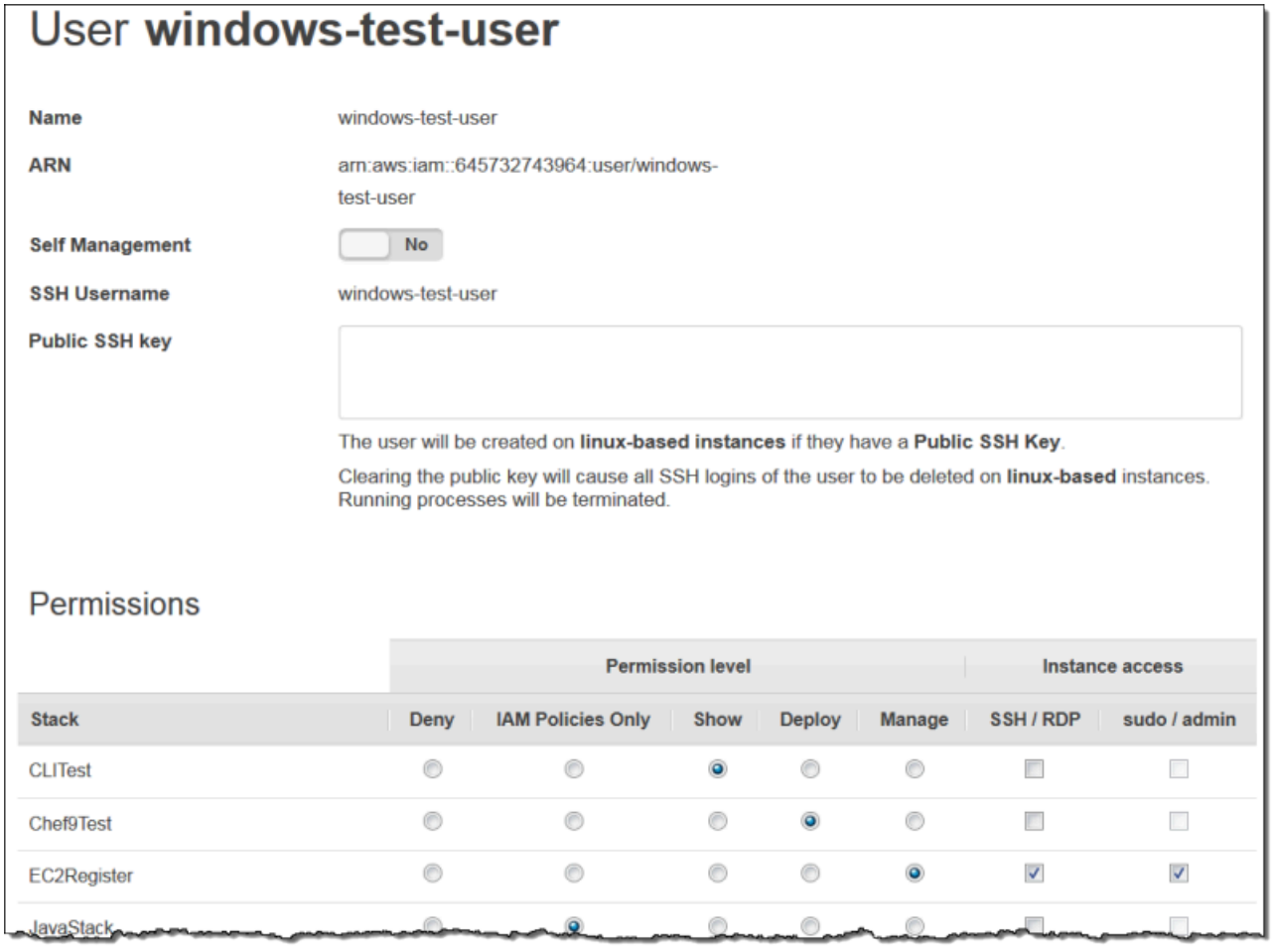
퍼블릭 SSH 키

(선택 사항) 사용자의 퍼블릭 SSH 키를 입력합니다. 이 키는 사용자의 [내 설정] 페이지에 나타납니다. 자체 관리를 활성화하면 해당 사용자가 [내 설정]을 편집하고 자신의 고유한 키를 지정할 수 있습니다. 자세한 내용은 사용자의 퍼블릭 SSH 키 등록 단원을 참조하십시오.

AWS OpsWorks Stacks는 모든 Linux 인스턴스에이 키를 설치합니다. 사용자는 연결된 프라이빗 키를 사용하여 로그인할 수 있습니다. 자세한 내용은 SSH를 사용하여 로그인 단원을 참조하십시오. 이 키는 Windows 스택에서는 사용할 수 없습니다.

권한

(선택 사항) 각 스택에 대한 사용자 권한 레벨은 개별적으로 설정할 필요 없이 각 스택의 [권한] 페이지에서 한 번에 설정할 수 있습니다. 권한 레벨에 대한 자세한 정보는 스택별 권한 부여 단원을 참조하세요.



AWS OpsWorks 스택 사용자 스택당 권한 부여

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 사용자 권한을 관리하는 가장 간단한 방법은 스택의 권한 페이지를 사용하는 것입니다. 각 스택마다 해당 스택에 대한 권한을 부여하는 페이지가 있습니다.

권한 설정을 수정하려면 관리 사용자 또는 [관리] 사용자로 로그인해야 합니다. 목록에는 AWS OpsWorks Stacks로 가져온 사용자만 표시됩니다. 사용자를 생성하고 가져오는 방법에 대한 자세한 정보는 사용자 관리 단원을 참조하세요.

기본 권한 수준은 사용자에게 IAM 정책의 권한만 부여하는 IAM 정책만입니다.

- IAM 또는 다른 리전에서 사용자를 가져오면 사용자는 IAM 정책만 권한 수준으로 모든 기존 스택의 목록에 추가됩니다.
- 기본적으로 다른 리전에서 방금 가져온 사용자는 대상 리전 내 스택에 액세스할 수 없습니다. 다른 리전에서 가져온 사용자가 대상 리전에서 스택을 관리할 수 있게 하려면 사용자를 가져온 후 해당 사용자에게 스택에 대한 권한을 부여해야 합니다.
- 새 스택을 생성하면 모든 현재 사용자가 IAM 정책만 권한 수준으로 목록에 추가됩니다.

주제

- 사용자 권한 설정
- 권한 보기
- IAM 조건 키를 사용하여 임시 자격 증명 확인

사용자 권한 설정

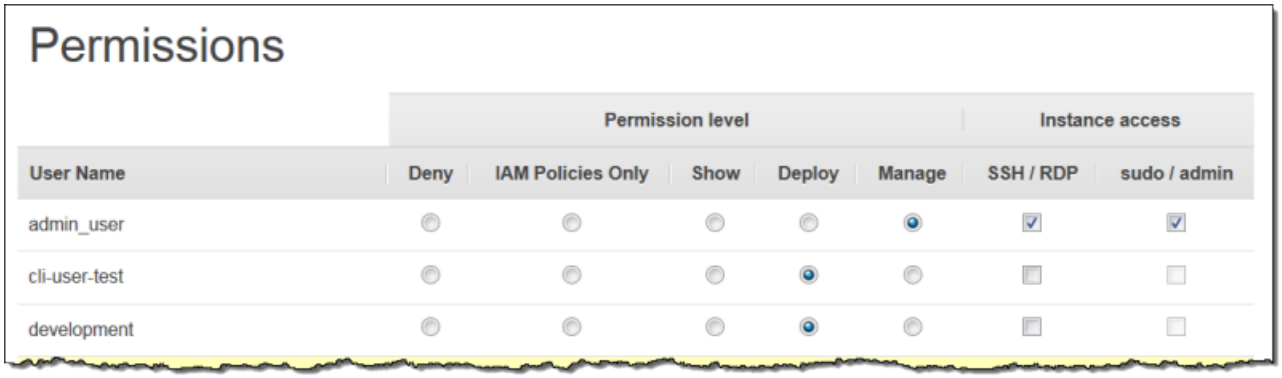
사용자 권한을 설정하려면

1. 탐색 창에서 권한을 선택합니다.

2. 권한 페이지에서 편집을 선택합니다.
3. 다음과 같이 권한 수준 및 인스턴스 액세스 설정을 변경합니다.

   - [권한 수준] 설정을 사용하여 각 사용자에게 표준 권한 레벨 중 하나를 할당합니다. 이 레벨은 사용자가 이 스택에 액세스할 수 있는지 여부와 사용자가 수행할 수 있는 작업을 결정합니다. 사용자에게 IAM 정책이 있는 경우 AWS OpsWorks Stacks는 두 권한 세트를 모두 평가합니다. 관련 예제는 정책 예제 단원을 참조하세요.
   - [인스턴스 액세스]의 [SSH/RDP] 설정은 사용자가 스택의 인스턴스에 대해 SSH(Linux) 또는 RDP(Windows) 액세스 권한을 가지고 있는지 여부를 지정합니다.

     SSH/RDP 액세스를 승인하면 경우에 따라 스택 인스턴스에서 사용자에게 sudo(Linux) 또는 administrative(Windows) 권한을 부여하는 sudo/admin을 선택할 수 있습니다.



각 사용자를 다음 권한 수준 중 하나에 할당할 수 있습니다. 각 수준에서 허용되는 작업의 목록은 AWS OpsWorks 스택 권한 수준 단원을 참조하세요.

거부

AWS OpsWorks Stacks에 전체 액세스 권한을 부여하는 IAM 정책이 있더라도 사용자는 스택에서 AWS OpsWorks Stacks 작업을 수행할 수 없습니다. 예를 들어 이 수준을 사용하여 일부 사용자가 릴리스 전 제품의 스택에 액세스하는 것을 거부할 수 있습니다.

IAM 정책만

새로 가져온 사용자에 할당되며 새로 생성된 스택에 대해 모든 사용자에게 할당되는 기본 수준입니다. 사용자의 권한은 IAM 정책에 따라 결정됩니다. 사용자에게 IAM 정책이 없거나 정책에 명시적 AWS OpsWorks Stacks 권한이 없는 경우 스택에 액세스할 수 없습니다. 관리 사용자에게는 일반적으로 이 수준이 할당됩니다. 관리 사용자에게 IAM 정책이 이미 전체 액세스 권한을 부여하기 때문입니다.

표시

사용자가 스택을 볼 수 있지만 작업을 수행할 수는 없습니다. 예를 들어 계정의 스택을 모니터링해야 하지만 앱을 배포하거나 어떤 식으로든 스택을 수정할 필요는 없는 관리자에게 부여할 수 있습니다.

배포

[표시] 권한을 포함하며 사용자가 앱을 배포할 수 있도록 허용합니다. 예를 들어 스택의 인스턴스에 업데이트를 배포해야 하지만 스택에 계층 또는 인스턴스를 추가할 필요는 없는 앱 개발자에게 부여할 수 있습니다.

관리

[배포] 권한을 포함하며 사용자에게 다음을 포함하여 다양한 스택 관리 작업을 수행하도록 허용합니다.

- 계층 및 인스턴스 추가 또는 삭제
- 스택의 [권한] 페이지를 사용하여 사용자에게 권한 수준을 할당
- 리소스 등록 또는 등록 해제

예를 들어 각 스택에 전담 관리자가 지명될 수 있습니다. 이 관리자의 책임은 스택에서 적절한 수 및 유형의 인스턴스가 실행되는지 확인하고, 패키지 및 운영 체제 업데이트를 처리하는 등입니다.

> **Note**
>
> [관리] 권한 수준은 사용자가 스택을 생성 또는 복제하도록 허용하지 않습니다. 이러한 권한은 IAM 정책을 통해 부여되어야 합니다. 예시는 권한 관리에서 확인하십시오.

사용자에게 IAM 정책도 있는 경우 AWS OpsWorks Stacks는 두 권한 세트를 모두 평가합니다. 그러므로 사용자에게 한 권한 수준을 할당한 다음 해당 수준에서 허용된 작업을 제한 또는 보강하는 정책을 적용할 수 있습니다. 예를 들어 관리 사용자가 스택을 생성 또는 복제하도록 허용하거나 리소스를 등록 또는 등록 해제하는 권한을 거부하는 정책을 적용할 수 있습니다. 이러한 정책의 몇 가지 예는 정책 예제 단원을 참조하세요.

> **Note**
>
> 사용자 정책이 추가 작업을 허용할 경우 결과가 [권한] 페이지 설정을 재정의하는 것으로 나타날 수 있습니다. 예를 들어 한 사용자에게 CreateLayer 작업을 허용하는 정책이 있지만 권한 페이지를 사용하여 배포 권한을 지정하는 경우 해당 사용자는 계속해서 계층을 생성할 수 있습니

다. 이 규칙의 예외는 거부 옵션입니다. 이 옵션은 AWSOpsWorks_FullAccess 정책이 연결된 사용자에게 스택 액세스조차 거부합니다. 자세한 내용은 정책을 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

권한 보기

자기 관리가 활성화된 경우 오른쪽 상단에서 [내 설정]을 선택하면 사용자 자신의 스택별 권한 수준의 요약을 볼 수 있습니다. 정책에서 DescribeMyUserProfile 및 UpdateMyUserProfile 작업에 대한 권한을 부여하는 경우에도 사용자는 내 설정에 액세스할 수 있습니다.

IAM 조건 키를 사용하여 임시 자격 증명 확인

AWS OpsWorks Stacks에는 추가 권한 부여 사례(예: 개별 사용자의 스택에 대한 읽기 전용 또는 읽기-쓰기 액세스의 간소화된 관리)를 지원하는 권한 부여 계층이 내장되어 있습니다. 이 권한 부여 계층은 임시 자격 증명을 사용합니다. 이 때문에 IAM 설명서의 IAM JSON 정책 요소 참조에서 설명된 대로 `aws:TokenIssueTime` 조건을 사용하여 해당 사용자가 장기 자격 증명을 사용하는지 확인하거나 임시 자격 증명을 사용하는 사용자의 작업을 차단할 수 없습니다.

IAM 정책을 연결하여 AWS OpsWorks 스택 권한 관리

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

IAM 정책을 연결하여 사용자의 AWS OpsWorks Stacks 권한을 지정할 수 있습니다. 다음과 같은 일부 권한에는 연결된 정책이 필요합니다.

- 사용자 가져오기와 같은 관리 사용자 권한
- 스택 생성 또는 복제와 같은 일부 작업에 대한 권한

연결된 정책을 필요로 하는 작업의 전체 목록은 AWS OpsWorks 스택 권한 수준 단원을 참조하세요.

또한 정책을 사용하여 권한 페이지를 통해 생성된 권한 수준을 사용자 지정할 수도 있습니다. 이 섹션에서는 사용자에게 IAM 정책을 적용하여 AWS OpsWorks Stacks 권한을 지정하는 방법에 대한 간략한 요약을 제공합니다. 자세한 내용은 AWS 리소스에 대한 액세스 관리를 참조하세요.

IAM 정책은 하나 이상의 문장으로 구성된 JSON 객체입니다. 각 문 요소에는 자체로 세 개의 기본 요소를 갖는 권한의 목록이 포함됩니다.

작업

권한이 영향을 미치는 작업. AWS OpsWorks Stacks 작업을 로 지정합니다`opsworks:`*`action`*. `Action`은 `opsworks:CreateStack`과 같은 특정 작업으로 설정될 수 있습니다. 이 작업은 사용자가 `CreateStack`을 호출하도록 허용되는지 여부를 지정합니다. 와일드카드를 사용하여 작업 그룹을 지정할 수도 있습니다. 예를 들어 `opsworks:Create*`는 모든 생성 작업을 지정합니다. AWS OpsWorks Stacks 작업의 전체 목록은 AWS OpsWorks Stacks API 참조를 참조하세요.

효과

지정된 작업을 허용 또는 거부하는지 여부.

리소스

권한이 영향을 미치는 AWS 리소스. AWS OpsWorks Stacks에는 하나의 리소스 유형인 스택이 있습니다. 특정 스택 리소스에 대해 권한을 지정하려면 `Resource`를 다음 형식을 따르는 스택 ARN으로 설정합니다. `arn:aws:opsworks:`*`region`*`:`*`account_id`*`:stack/`*`stack_id`*`/`.

와일드카드를 사용할 수도 있습니다. 예를 들어 `Resource`를 `*`로 설정하면 모든 리소스에 대해 권한이 부여됩니다.

예를 들어 다음 정책은 사용자에게서 ID가 `2860-2f18b4cb-4de5-4429-a149-ff7da9f0d8ee`인 스택에서 인스턴스를 정지할 수 있는 권한을 거부합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "opsworks:StopInstance",
      "Effect": "Deny",
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

IAM 사용자에게 권한을 추가하는 방법에 대한 자세한 내용은 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console 섹션을 참조하세요.

IAM 정책을 생성 또는 수정하는 방법에 대한 자세한 내용은 IAM의 권한 및 정책을 참조하세요. AWS OpsWorks Stacks 정책의 몇 가지 예는 섹션을 참조하세요정책 예제.

정책 예제

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 단원에서는 AWS OpsWorks Stacks 사용자에게 적용할 수 있는 IAM 정책 예제를 설명합니다.

- 관리 권한은(는) 관리 사용자에게 권한을 부여하는 데 사용할 수 있는 정책을 설명합니다.
- 권한 관리 및 [배포] 권한 섹션에서는 관리 및 배포 권한 수준을 보강 또는 제한하기 위해 사용자에게 적용할 수 있는 정책을 예시합니다.

  AWS OpsWorks Stacks는 IAM 정책에서 부여한 권한과 권한 페이지에서 부여한 권한을 평가하여 사용자의 권한을 결정합니다. 자세한 내용은 정책을 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요. [권한] 페이지에 대한 자세한 정보는 AWS OpsWorks 스택 권한 수준 단원을 참조하세요.

관리 권한

IAM 콘솔 https://console.aws.amazon.com/iam/ 사용하여 AWSOpsWorks_FullAccess 정책에 액세스합니다.이 정책을 사용자에게 연결하여 모든 AWS OpsWorks Stacks 작업을 수행할 수 있는 권한을 부여합니다. IAM 권한이 필요합니다(특히 관리 사용자가 사용자를 가져오도록 허용하는 경우).

AWS OpsWorks Stacks가 사용자를 대신하여 Amazon EC2 인스턴스와 같은 다른 AWS 리소스에 액세스할 수 있도록 허용하는 IAM 역할을 생성해야 합니다. 일반적으로 관리 사용자가 첫 번째 스택을 생성하고 Stacks가 사용자를 대신하여 역할을 생성하도록 AWS OpsWorks 하여이 작업을 처리합니다. 그러면 이후의 모든 작업에 이 역할을 사용할 수 있습니다. 자세한 내용은 AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용 단원을 참조하십시오.

첫 번째 스택을 생성하는 관리 사용자에게는 다음과 같이 AWSOpsWorks_FullAccess 정책에 포함되지 않은 일부 IAM 작업에 대한 권한을 부여해야 합니다. 정책 `Actions` 섹션에 다음 권한을 추가합니다. 적절한 JSON 구문을 위해 작업 사이에 쉼표를 추가하고 작업 목록 끝에 있는 후행 쉼표를 제거해야 합니다.

```
"iam:PutRolePolicy",
"iam:AddRoleToInstanceProfile",
"iam:CreateInstanceProfile",
"iam:CreateRole"
```

권한 관리

[관리] 권한 수준은 사용자가 계층 생성 또는 삭제를 포함한 다양한 스택 관리 작업을 수행하도록 허용합니다. 이 주제에서는 관리 사용자에게 사용하여 표준 권한을 보강 또는 제한할 수 있는 여러 정책을 설명합니다.

[관리] 사용자에게서 계층을 추가 또는 삭제할 수 있는 권한을 거부

다음 IAM 정책을 사용하여 사용자가 계층 추가 또는 삭제를 제외한 모든 관리 작업을 수행할 수 있도록 관리 권한 수준을 제한할 수 있습니다. *##*, *account_id*, *stack_id*를 구성에 적합한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
}
```

[관리] 사용자가 스택을 생성 또는 복제할 수 있도록 허용

관리 권한 수준은 사용자가 스택을 생성하거나 복제하도록 허용하지 않습니다. 다음 IAM 정책을 적용하여 사용자가 스택을 생성하거나 복제할 수 있도록 관리 권한을 변경할 수 있습니다. *##* 및 *account_id*를 구성에 적합한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:opsworks::account_id:stack/*/",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

[관리] 사용자에게서 리소스를 등록 또는 등록 해제할 수 있는 권한을 거부

관리 권한 수준은 사용자가 스택에 Amazon EBS 및 탄력적 IP 주소 리소스를 등록 또는 등록 해제할 수 있도록 허용합니다. 다음 정책을 적용하여 사용자가 리소스 등록을 제외한 모든 관리 작업을 수행할 수 있도록 관리 권한을 제한할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
```

```
        "opsworks:RegisterVolume",
        "opsworks:RegisterElasticIp"
      ],
      "Resource": "*"
    }
  ]
}
```

[관리] 사용자가 사용자를 가져오도록 허용

권한 관리 수준에서는 사용자가 사용자를 AWS OpsWorks Stacks로 가져올 수 없습니다. 다음 IAM 정책을 적용하여 사용자가 사용자를 가져오고 삭제할 수 있도록 관리 권한을 강화할 수 있습니다. *##* 및 *account_id*를 구성에 적합한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "iam:PassRole",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "arn:aws:iam:region:account_id:user/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

[배포] 권한

[배포] 권한 수준은 사용자가 앱을 생성하거나 복제하도록 허용하지 않습니다. 다음 IAM 정책을 적용하여 사용자가 앱을 만들고 삭제할 수 있도록 배포 권한을 강화할 수 있습니다. *##*, *account_id*, *stack_id*를 구성에 적합한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "opsworks:CreateApp",
        "opsworks:DeleteApp"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
}
```

AWS OpsWorks 스택 권한 수준

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 AWS OpsWorks 스택 권한 페이지의 권한 수준 표시, 배포 및 관리에서 허용하는 작업을 나열합니다. 또한 IAM 정책을 적용해야만 권한을 부여할 수 있는 작업의 목록도 포함되어 있습니다.

표시

[표시] 수준은 `Describe`*XYZ* 명령을 허용하지만 다음의 예외가 있습니다.

```
DescribePermissions
DescribeUserProfiles
DescribeMyUserProfile
DescribeStackProvisioningParameters
```

관리 사용자가 사용자에 대해 자기 관리를 활성화한 경우 [표시] 사용자는 `DescribeMyUserProfile` 및 `UpdateMyUserProfile`도 사용할 수 있습니다. 자기 관리에 대한 자세한 정보는 사용자 설정 편집 단원을 참조하세요.

배포

[배포] 수준에서는 [표시] 수준에서 허용되는 작업 이외에 다음 작업이 허용됩니다.

```
CreateDeployment
UpdateApp
```

관리

[관리] 수준에서는 [배포] 및 [표시] 수준에서 허용되는 작업 이외에 다음 작업이 허용됩니다.

```
AssignInstance
AssignVolume
AssociateElasticIp
AttachElasticLoadBalancer
CreateApp
CreateInstance
CreateLayer
DeleteApp
DeleteInstance
DeleteLayer
DeleteStack
DeregisterElasticIp
DeregisterInstance
DeregisterRdsDbInstance
DeregisterVolume
DescribePermissions
DetachElasticLoadBalancer
DisassociateElasticIp
GrantAccess
GetHostnameSuggestion
RebootInstance
RegisterElasticIp
RegisterInstance
RegisterRdsDbInstance
RegisterVolume
SetLoadBasedAutoScaling
SetPermission
SetTimeBasedAutoScaling
```

```
StartInstance
StartStack
StopInstance
StopStack
UnassignVolume
UpdateElasticIp
UpdateInstance
UpdateLayer
UpdateRdsDbInstance
UpdateStack
UpdateVolume
```

IAM 정책을 요구하는 권한

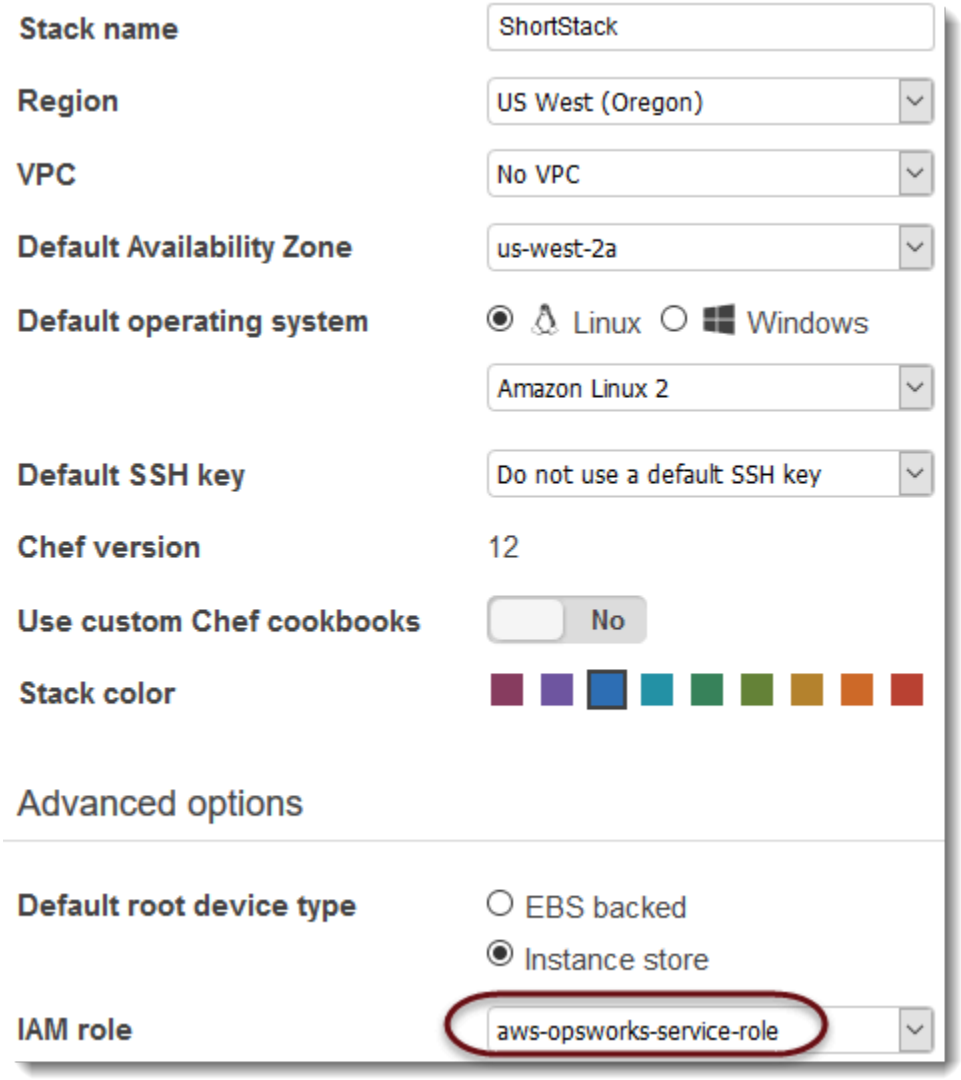
다음 작업에 대한 권한을 부여하려면 사용자에게 적절한 IAM 정책을 연결해야 합니다. 몇 가지 예는 정책 예제 섹션을 참조하세요.

```
CloneStack
CreateStack
CreateUserProfile
DeleteUserProfile
DescribeUserProfiles
UpdateUserProfile
```

AWS OpsWorks Stacks가 사용자를 대신하여 작동하도록 허용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 사용자를 대신하여 다양한 AWS 서비스와 상호 작용해야 합니다. 예를 들어 AWS OpsWorks Stacks는 Amazon EC2와 상호 작용하여 인스턴스를 생성하고 Amazon CloudWatch와 상호 작용하여 모니터링 통계를 가져옵니다. 스택을 생성할 때 AWS OpsWorks Stacks에 적절한 권한을 부여하는 일반적으로 서비스 역할이라고 하는 IAM 역할을 지정합니다.



새 스택의 서비스 역할을 지정할 때 다음 중 하나를 수행할 수 있습니다.

- 앞서 생성한 표준 서비스 역할을 지정합니다.

  첫 번째 스택을 만들 때 표준 서비스 역할을 생성한 후 이후 모든 스택에 대해 해당 역할을 사용하는 것이 일반적입니다.
- IAM 콘솔 또는 API를 사용하여 생성한 사용자 지정 서비스 역할을 지정합니다.

  이 접근 방식은 AWS OpsWorks Stacks에 표준 서비스 역할보다 더 제한된 권한을 부여하려는 경우에 유용합니다.

> **Note**
>
> 첫 번째 스택을 생성하려면 IAM AdministratorAccess 정책 템플릿에서 정의된 권한이 있어야 합니다. 앞서 설명한 것처럼 이러한 권한을 통해 AWS OpsWorks Stacks에서 새 IAM 서비스 역할을 생성하고 사용자를 가져올 수 있습니다. 이후 모든 스택에 대해 사용자가 첫 번째 스택에 대해 생성된 서비스 역할을 선택할 수 있습니다. 스택을 생성하는 데 전체 관리 권한이 필요한 것은 아닙니다.

표준 서비스 역할이 부여하는 권한은 다음과 같습니다.

- 모든 Amazon EC2 작업(`ec2:*`)을 수행합니다.
- 클라우드워치 통계 가져오기(`cloudwatch:GetMetricStatistics`).
- Elastic Load Balancing을 사용하여 서버에 트래픽을 분산합니다(`elasticloadbalancing:*`).
- Amazon RDS 인스턴스를 데이터베이스 서버로 사용합니다(`rds:*`).
- IAM 역할(`iam:PassRole`)을 사용하여 AWS OpsWorks Stacks와 Amazon EC2 인스턴스 간에 안전한 통신을 제공합니다.

사용자 지정 서비스 역할을 생성하는 경우 AWS OpsWorks Stacks가 스택을 관리하는 데 필요한 모든 권한을 부여해야 합니다. 다음 JSON 샘플은 표준 서비스 역할에 대한 정책 문입니다. 사용자 지정 서비스 역할은 최소한 다음 권한을 정책 문에 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:*",
                "iam:PassRole",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "ecs:*",
                "elasticloadbalancing:*",
                "rds:*"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        }
    ]
}
```

서비스 역할에는 신뢰 관계도 있습니다. AWS OpsWorks Stacks에서 생성한 서비스 역할은 다음과 같은 신뢰 관계를 갖습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StsAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS OpsWorks Stacks가 사용자를 대신하여 작업하려면 서비스 역할에이 신뢰 관계가 있어야 합니다. 기본 서비스 역할을 사용하는 경우, 신뢰 관계를 수정하지 마십시오. 사용자 지정 서비스 역할을 생성하는 경우, 다음과 같이 신뢰 관계를 지정합니다.

- IAM 콘솔에서 역할 생성 마법사를 사용하는 경우 사용 사례 선택에서 Opsworks를 선택합니다. 역할에 적절한 신뢰 관계가 있지만 암시적으로 연결된 정책은 없습니다. AWS OpsWorks Stacks에 사용자 대신 작업을 수행할 권한을 부여하려면 다음을 포함하는 고객 관리형 정책을 생성하여 새 역할에 연결하세요.

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
  ```

```
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricStatistics",
        "ec2:*",
        "ecs:*",
        "elasticloadbalancing:*",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfiles",
        "iam:ListRoles",
        "iam:ListUsers",
        "rds:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "ec2.amazonaws.com"
        }
      }
    }
  ]
}
```

- AWS CloudFormation 템플릿을 사용하는 경우 템플릿의 리소스 섹션에 다음과 같은 항목을 추가할 수 있습니다.

```
"Resources": {
  "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
            "Statement": [ {
              "Effect": "Allow",
              "Principal": {
                  "Service": [ "opsworks.amazonaws.com" ]
              },
```

```
              "Action": [ "sts:AssumeRole" ]
            } ]
        },
        "Path": "/",
        "Policies": [ {
            "PolicyName": "opsworks-service",
            "PolicyDocument": {
              ...
            } ]
        }
      },
   }
}
```

AWS OpsWorks Stacks의 교차 서비스 혼동된 대리자 방지

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 위장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

스택 액세스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 AWS OpsWorks Stacks가 스택에 다른 서비스를 제공하는 권한을 제한하는 것이 좋습니다. 만약 `aws:SourceArn` 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 전역 조건 컨텍스트 키를 모두 사용해야 합니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 `aws:SourceArn` 값을 모두 사용하는 경우, `aws:SourceAccount` 값 및 `aws:SourceArn` 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다. 하나의 스택만 교

차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을(를) 사용하세요. 해당 계정의 모든 스택이 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount을(를) 사용하세요.

의 값은 AWS OpsWorks 스택의 ARN이어야 aws:SourceArn 합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 AWS OpsWorks Stacks 스택의 전체 ARN이 포함된 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 스택의 전체 ARN을 모를 경우 또는 여러 스택 ARN을 지정하는 경우 ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예: arn:aws:*servicename*:*:*123456789012*:*.

다음 섹션에서는 AWS OpsWorks Stacks에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

AWS OpsWorks Stacks에서 혼동된 대리자 악용 방지

이 섹션에서는 Stacks에서 AWS OpsWorks 혼동된 대리자 악용을 방지하는 방법을 설명하고 AWS OpsWorks Stacks에 액세스하는 데 사용하는 IAM 역할에 연결할 수 있는 권한 정책의 예를 포함합니다. 최상의 보안을 위해, IAM 역할이 다른 서비스와 가지는 신뢰 관계에 aws:SourceArn 및 aws:SourceAccount 조건 키를 추가하는 것이 좋습니다. 신뢰 관계를 통해 AWS OpsWorks Stacks는 역할을 수임하여 AWS OpsWorks Stacks 스택을 생성하거나 관리하는 데 필요한 다른 서비스에서 작업을 수행할 수 있습니다.

신뢰 관계를 편집하여 **aws:SourceArn** 및 **aws:SourceAccount** 조건 키를 추가하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 검색 상자에서 AWS OpsWorks Stacks에 액세스하는 데 사용하는 역할을 검색합니다. AWS 관리형 역할은 입니다aws-opsworks-service-role.
4. 역할의 요약 페이지에서 신뢰 관계 탭을 선택합니다.
5. 신뢰 관계 탭에서 신뢰 정책 편집을 선택합니다.
6. 신뢰 정책 편집 페이지에서 하나 이상의 aws:SourceArn 또는 aws:SourceAccount 조건 키를 정책에 추가합니다. aws:SourceArn를 사용하여 교차 서비스(예: Amazon EC2)와 AWS OpsWorks Stacks 간의 신뢰 관계를 특정 AWS OpsWorks Stacks 스택으로 제한합니다. 이는 더 제한적입니다. aws:SourceAccount를 추가하여 교차 서비스와 AWS OpsWorks Stacks 간의 신뢰 관계를 특정 계정의 스택으로 제한합니다. 이는 덜 제한적입니다. 다음은 예입니다. 두 조건 키를 모두 사용하는 경우 계정 ID가 동일해야 한다는 점에 유의하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "arn:aws:opsworks:us-east-2:123456789012:stack/
EXAMPLEd-5699-40a3-80c3-22c32EXAMPLE/"
        }
      }
    }
  ]
}
```

7. 조건 키 추가가 끝나면 정책 업데이트를 선택합니다.

다음은 aws:SourceArn 및 aws:SourceAccount를 사용하여 스택에 대한 액세스를 제한하는 역할의 추가 예입니다.

주제

- 예제: 특정 리전에서 스택 액세스
- 예: 두 개 이상의 스택 ARN을 aws:SourceArn에 추가

예제: 특정 리전에서 스택 액세스

다음 역할 신뢰 관계 문은 미국 동부(오하이오) 리전()의 모든 AWS OpsWorks Stacks 스택에 액세스합니다us-east-2. 참고로 리전은 aws:SourceArn의 ARN 값에서 지정되지만 스택 ID 값은 와일드카드 (*) 입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks:us-east-2:123456789012:stack/*"
        }
      }
    }
  ]
}
```

예: 두 개 이상의 스택 ARN을 **aws:SourceArn**에 추가

다음 예제에서는 계정 ID 123456789012의 두 AWS OpsWorks Stacks 스택 배열에 대한 액세스를 제한합니다.

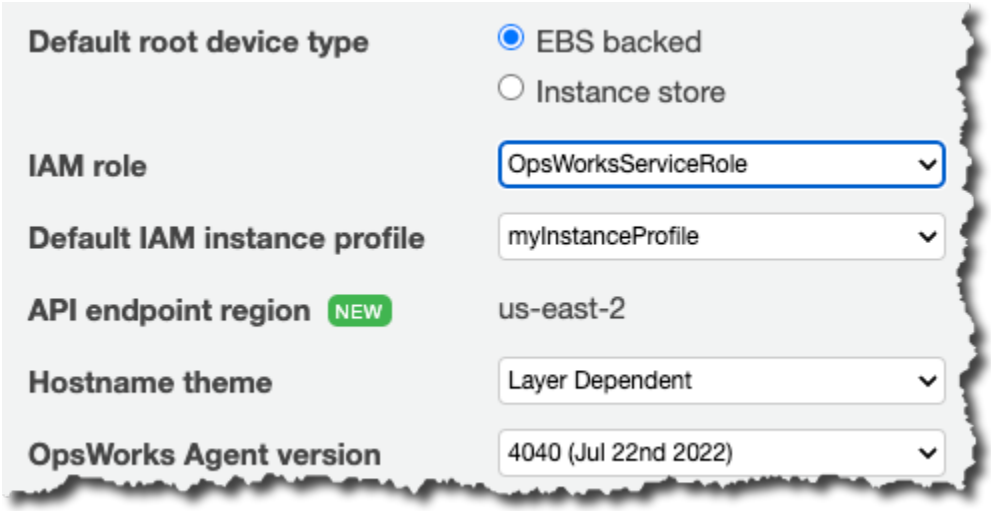
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID1",
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID2"
           ]
       }
      }
```

```
    }
  ]
}
```

EC2 인스턴스에서 실행되는 앱에 대한 권한 지정

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 Amazon EC2 인스턴스에서 실행되는 애플리케이션이 Amazon S3 버킷 같은 다른 AWS 리소스에 액세스해야 하는 경우, 해당 애플리케이션에 적절한 권한이 있어야 합니다. 이러한 권한을 부여하려면 인스턴스 프로파일을 사용합니다. AWS OpsWorks Stacks 스택을 생성할 때 각 인스턴스에 대한 인스턴스 프로파일을 지정할 수 있습니다.



계층 구성을 편집하여 계층의 인스턴스에 대한 프로파일을 지정할 수도 있습니다.

인스턴스 프로파일은 IAM 역할을 지정합니다. 인스턴스에서 실행되는 애플리케이션은 해당 역할을 수행하여 역할의 정책이 부여하는 권한에 따라 AWS 리소스에 액세스할 수 있습니다. 애플리케이션이 역할을 수행하는 방식에 대한 자세한 내용은 API 호출을 사용하여 역할 수행을 참조하세요.

다음 방법 중 하나를 사용하여 인스턴스 프로파일을 생성할 수 있습니다.

- IAM 콘솔 또는 API를 사용하여 프로파일을 생성합니다.

자세한 내용은 역할(위임 및 연동)을 참조하세요.

- AWS CloudFormation 템플릿을 사용하여 프로필을 생성합니다.

  템플릿의 IAM 리소스를 포함시키는 방법의 몇 가지 예제는 Identity and Access Management(IAM) 템플릿 조각을 참조하세요.

인스턴스 프로파일에는 신뢰 관계와 AWS 리소스에 액세스할 수 있는 권한을 부여하는 연결된 정책이 있어야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS OpsWorks Stacks가 사용자를 대신하여 작동하려면 인스턴스 프로파일에이 신뢰 관계가 있어야 합니다. 기본 서비스 역할을 사용하는 경우, 신뢰 관계를 수정하지 마십시오. 사용자 지정 서비스 역할을 생성하는 경우, 다음과 같이 신뢰 관계를 지정합니다.

- IAM 콘솔에서 역할 생성 마법사를 사용하는 경우 마법사의 두 번째 페이지에 있는 AWS 서비스 역할에서 Amazon EC2 역할 유형을 지정합니다.
- AWS CloudFormation 템플릿을 사용하는 경우 템플릿의 리소스 섹션에 다음과 같은 항목을 추가할 수 있습니다.

```
"Resources": {
      "OpsWorksEC2Role": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
```

```
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "OpsWorksEC2Role"
            }
         ]
      }
   }
}
```

자체 인스턴스 프로파일을 생성하는 경우, 그 시점의 프로파일의 역할에 적절한 정책을 연결할 수 있습니다. 스택을 생성한 후 IAM 콘솔 또는 API를 사용하여 프로파일의 역할에 적절한 정책을 연결해야 합니다. 예를 들어 다음 정책은 amzn-s3-demo-bucket이라는 Amazon S3 버킷의 모든 객체에 대한 전체 액세스 권한을 부여합니다. *region* 및 amzn-s3-demo-bucket을 구성에 적합한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:region::amzn-s3-demo-bucket/*"
    }
  ]
}
```

인스턴스 프로파일을 생성하고 사용하는 방법의 예제는 Amazon S3 버킷 사용을 참조하세요.

애플리케이션이 인스턴스 프로파일을 사용하여 EC2 인스턴스에서 AWS OpsWorks Stacks API를 호출하는 경우 정책은 AWS OpsWorks Stacks 및 기타 AWS 서비스에 대한 적절한 `iam:PassRole` 작업 외에도 작업을 허용해야 합니다. `iam:PassRole` 권한은 AWS OpsWorks Stacks가 사용자를 대신

하여 서비스 역할을 수행하도록 허용합니다. AWS OpsWorks Stacks API에 대한 자세한 내용은 AWS OpsWorks API 참조를 참조하세요.

다음은 EC2 인스턴스에서 모든 AWS OpsWorks Stacks 작업과 Amazon EC2 또는 Amazon S3 작업을 호출할 수 있는 IAM 정책의 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "s3:*",
        "opsworks:*",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:ec2:region:account_id:instance/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

Note

를 허용하지 않으면 AWS OpsWorks Stacks 작업을 호출하려는 `iam:PassRole`시도가 실패하고 다음과 같은 오류가 발생합니다.

```
User: arn:aws:sts::123456789012:federated-user/Bob is not authorized
to perform: iam:PassRole on resource:
arn:aws:sts::123456789012:role/OpsWorksStackIamRole
```

EC2 인스턴스의 역할을 사용하여 권한을 부여하는 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 Amazon EC2 인스턴스에서 실행하는 애플리케이션에 AWS 리소스에 대한 액세스 권한 부여를 참조하세요.

SSH 액세스 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Linux 스택과 Windows 스택 모두에 대해 SSH 키를 지원합니다.

- Linux 인스턴스의 경우, SSH를 사용하여 인스턴스에 로그인한 다음 예컨대 agent CLI 명령을 실행합니다.

  자세한 내용은 SSH를 사용하여 로그인 단원을 참조하십시오.
- Windows 인스턴스의 경우, SSH 키를 사용하여 인스턴스의 관리자 암호를 가져온 다음 RDP를 사용하여 로그인할 수 있습니다.

  자세한 내용은 RDP를 사용하여 로그인 단원을 참조하십시오.

인증은 퍼블릭 키와 프라이빗 키로 구성된 SSH 키 페어에 기반합니다.

- 인스턴스에 퍼블릭 키를 설치합니다.

  위치는 특정 운영 체제에 따라 다르지만 AWS OpsWorks Stacks는 세부 정보를 처리합니다.
- 프라이빗 키를 로컬에 저장하고 인스턴스에 액세스할 수 있도록 `ssh.exe` 같은 SSH 클라이언트에 제공합니다.

  SSH 클라이언트는 프라이빗 키를 사용하여 인스턴스에 연결합니다.

스택의 사용자에게 SSH 액세스 권한을 제공하려면 SSH 키 페어를 생성하고 스택의 인스턴스에 퍼블릭 키를 설치하고 프라이빗 키를 안전하게 관리할 방법이 필요합니다.

Amazon EC2는 인스턴스에 퍼블릭 SSH 키를 설치하는 간단한 방법을 제공합니다. Amazon EC2 콘솔 또는 API를 사용하여 사용하려는 각 AWS 리전에 대해 하나 이상의 키 페어를 생성할 수 있습니다. AWS의 퍼블릭 키는 Amazon EC2에 저장되며 프라이빗 키는 사용자가 현지에 저장합니다. 인스턴스를 시작할 때 리전의 키 페어 중 하나를 지정하면 Amazon EC2가 자동으로 인스턴스에 키 페어를 설

치합니다. 그런 다음 해당 프라이빗 키를 사용하여 인스턴스에 로그인합니다. 자세한 내용은 Amazon EC2 키 페어를 참조하세요.

AWS OpsWorks 스택을 사용하면 스택을 생성할 때 리전의 Amazon EC2 키 페어 중 하나를 지정하고, 선택적으로 각 인스턴스를 생성할 때 다른 키 페어로 재정의할 수 있습니다. AWS OpsWorks Stacks는 해당 Amazon EC2 인스턴스를 시작할 때 키 페어를 지정하며 Amazon EC2는 인스턴스에 퍼블릭 키를 설치합니다. 그러면 표준 Amazon EC2 인스턴스처럼 프라이빗 키를 사용하여 로그인하거나 관리자 암호를 검색할 수 있습니다. 자세한 내용은 Amazon EC2 키 설치 단원을 참조하십시오.

Amazon EC2 키 페어는 사용이 편리하지만 두 가지 중요한 제한이 있습니다.

- Amazon EC2 키 페어는 특정 AWS 리전에 연결됩니다.

  여러 리전에서 작업하는 경우, 여러 개의 키 페어를 관리해야 합니다.
- 인스턴스 하나에 Amazon EC2 키 페어 하나만 설치할 수 있습니다.

  여러 사용자가 로그인하도록 하려면 사용자 모두에게 프라이빗 키의 사본이 있어야 하는데, 이는 권장되는 보안 관행이 아닙니다.

Linux 스택의 경우 AWS OpsWorks Stacks는 SSH 키 페어를 관리하는 더 간단하고 유연한 방법을 제공합니다.

- 각 사용자가 개인 키 페어를 등록합니다.

  에 설명된 대로 프라이빗 키를 로컬에 저장하고 퍼블릭 키를 AWS OpsWorks Stacks에 등록합니다사용자의 퍼블릭 SSH 키 등록.
- 스택에 대한 사용자 권한을 설정할 때 스택의 인스턴스에 대한 SSH 액세스 권한을 가질 사용자를 지정합니다.

  AWS OpsWorks Stacks는 권한 있는 각 사용자에 대해 스택의 인스턴스에 시스템 사용자를 자동으로 생성하고 퍼블릭 키를 설치합니다. 그러면 이 사용자는 SSH를 사용하여 로그인에 설명된 것처럼 해당 프라이빗 키를 사용하여 로그인할 수 있습니다.

개인 SSH 키를 사용하면 다음과 같은 장점이 있습니다.

- 인스턴스에서 키를 수동으로 구성할 필요가 없습니다. AWS OpsWorks Stacks는 모든 인스턴스에 적절한 퍼블릭 키를 자동으로 설치합니다.
- AWS OpsWorks Stacks는 승인된 사용자의 개인 퍼블릭 키만 설치합니다.

권한이 없는 사용자는 개인 프라이빗 키를 사용하여 인스턴스에 액세스할 수 없습니다. Amazon EC2 키 페어를 사용하면 해당 프라이빗 키가 있는 사용자는 SSH 액세스 권한 부여 여부에 상관없이 로그인할 수 있습니다.

- 사용자에게 더 이상 SSH 액세스 권한이 필요 없는 경우, [권한] 페이지를 사용해 해당 사용자의 SSH/RDP 권한을 취소할 수 있습니다.

  AWS OpsWorks Stacks는 스택의 인스턴스에서 퍼블릭 키를 즉시 제거합니다.

- 어떤 AWS 리전에도 동일한 키를 사용할 수 있습니다.

  사용자는 하나의 프라이빗 키만 관리하면 됩니다.

- 프라이빗 키를 공유할 필요가 없습니다.

  사용자 각자에게 자신의 프라이빗 키가 있습니다.

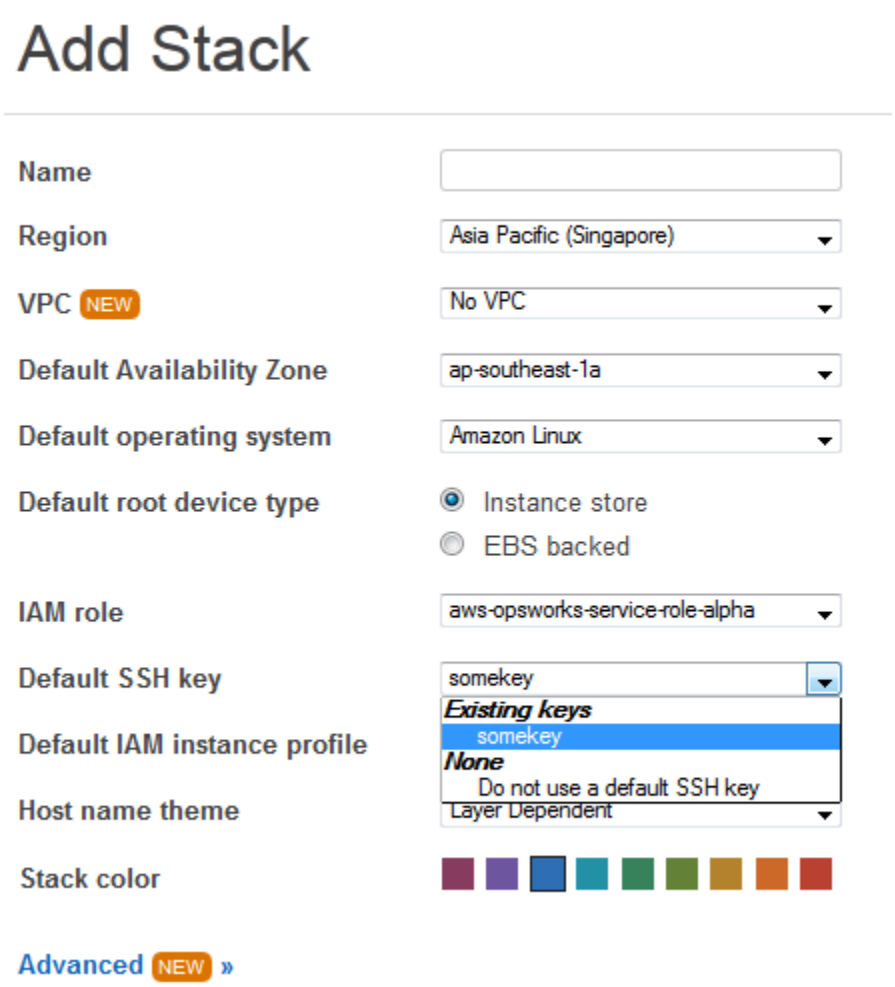
- 키를 교체하는 것은 쉽습니다.

  본인이 직접 또는 해당 사용자가 내 설정에서 퍼블릭 키를 업데이트하면 AWS OpsWorks Stacks가 자동으로 인스턴스를 업데이트합니다.

Amazon EC2 키 설치

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

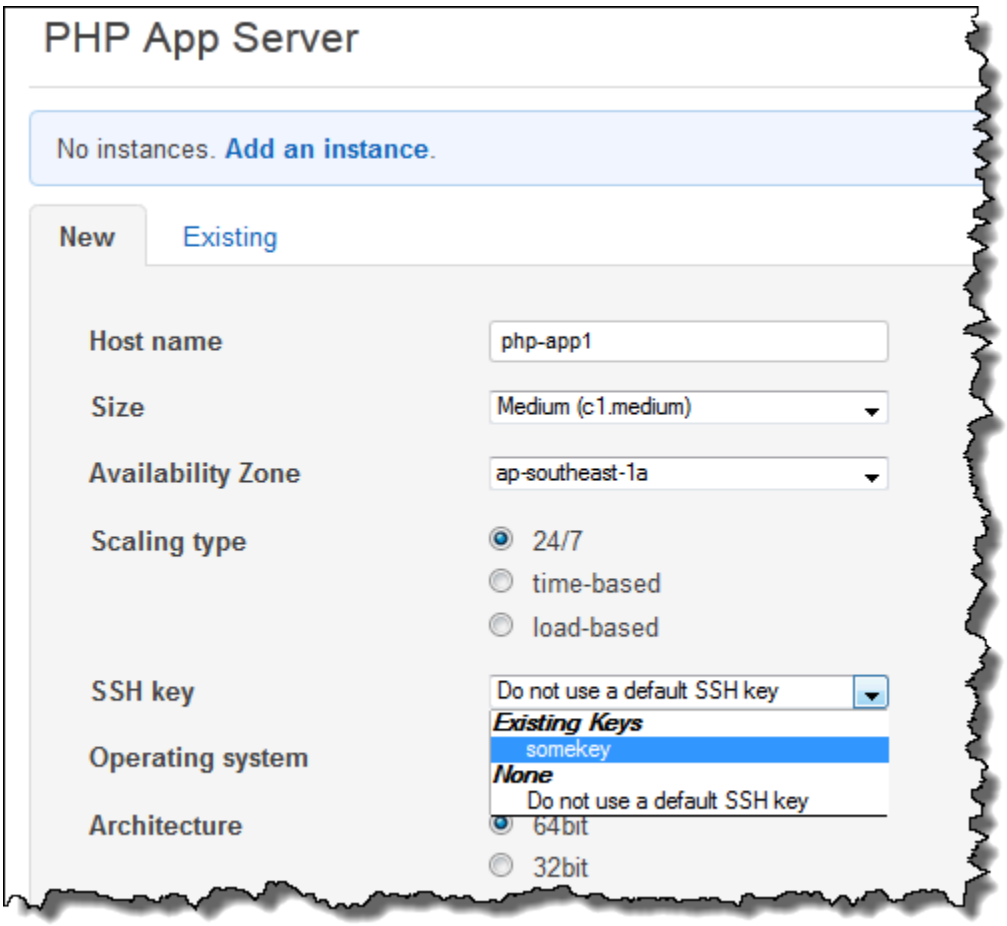
스택을 생성할 때 스택의 모든 인스턴스에 기본적으로 설치되는 Amazon EC2 SSH 키를 지정할 수 있습니다.



기본 SSH 키 목록은 사용자 AWS 계정의 Amazon EC2 키를 보여 줍니다. 다음 중 하나를 수행할 수 있습니다.

- 목록에서 적절한 키를 선택합니다.
- 아무 키도 지정하지 않으려면 [기본 SSH 키 사용 금지]를 선택합니다.

[기본 SSH 키 사용 금지]를 선택했거나 스택의 기본 키를 재정의하려는 경우, 인스턴스를 생성할 때 키를 지정할 수 있습니다.



인스턴스를 시작하면 AWS OpsWorks Stacks가 `authorized_keys` 파일에 퍼블릭 키를 설치합니다.

사용자의 퍼블릭 SSH 키 등록

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자의 퍼블릭 SSH 키를 등록하는 방법은 다음 두 가지가 있습니다.

- 관리 사용자는 한 명 이상의 사용자에게 퍼블릭 SSH 키를 할당하고 해당 프라이빗 키를 제공할 수 있습니다.
- 관리 사용자는 한 명 이상의 사용자에 대해 자기 관리를 활성화할 수 있습니다.

그러면 이 사용자들이 자신의 퍼블릭 SSH 키를 지정할 수 있습니다.

관리 사용자가 자기 관리를 활성화하거나 사용자에게 퍼블릭 키를 할당하는 방법에 대한 자세한 정보는 사용자 설정 편집 단원을 참조하세요.

PuTTY 터미널에서 SSH를 사용하여 Linux 기반 인스턴스에 연결하려면 추가 단계가 필요합니다. 자세한 정보는 AWS 설명서의 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결과 인스턴스 연결 문제 해결 단원을 참조하세요.

다음은 자기 관리가 활성화된 사용자가 퍼블릭 키를 지정하는 방법을 설명합니다.

SSH 퍼블릭 키를 지정하려면

1. SSH 키 페어를 생성합니다.

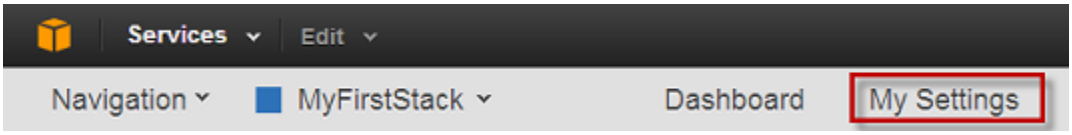
   키 페어를 로컬에 생성하는 것이 가장 간단합니다. 자세한 내용은 고유 키를 생성하여 Amazon EC2로 가져오는 방법을 참조하세요.

   > **Note**
   >
   > PuTTYgen을 사용하여 키 페어를 생성하는 경우 OpenSSH authorized_keys 파일 상자에 붙여넣기 위해 퍼블릭 키에서 퍼블릭 키를 복사합니다. [퍼블릭 키 저장]을 클릭하면 MindTerm에서 지원하지 않는 형식으로 퍼블릭 키가 저장됩니다.

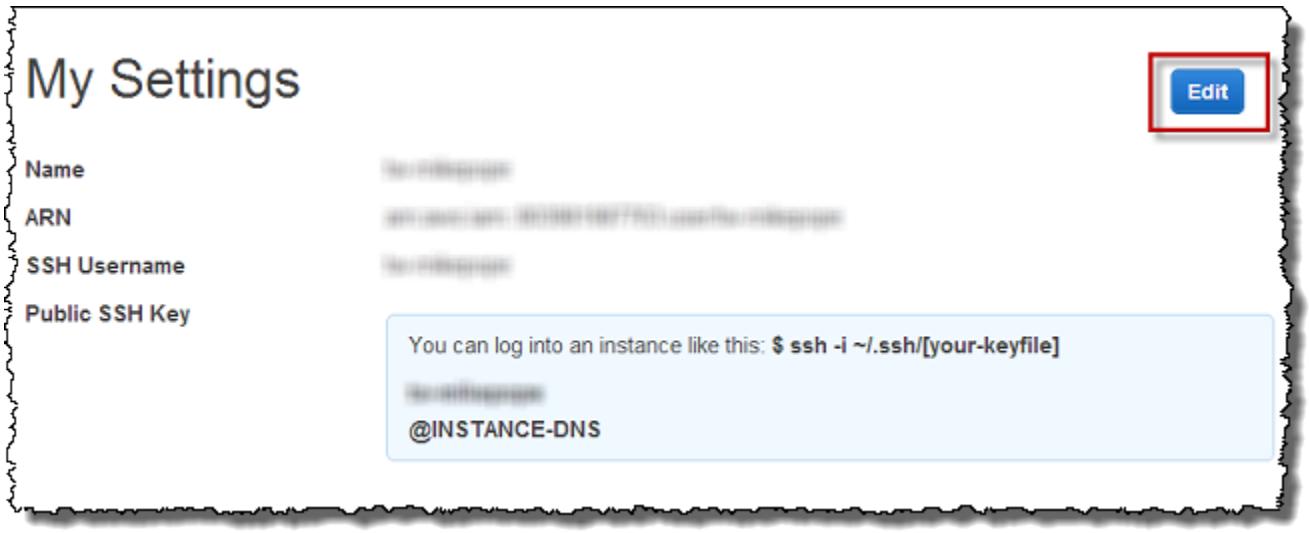
2. 자체 관리가 활성화된 IAM 사용자로 AWS OpsWorks Stacks 콘솔에 로그인합니다.

   > **Important**
   >
   > 계정 소유자 또는 자체 관리가 활성화되지 않은 IAM 사용자로 로그인하는 경우 AWS OpsWorks Stacks는 내 설정을 표시하지 않습니다. 관리 사용자 또는 계정 소유자인 경우 [사용자] 페이지로 이동하여 사용자 설정을 편집해 SSH 키를 대신 지정할 수 있습니다.

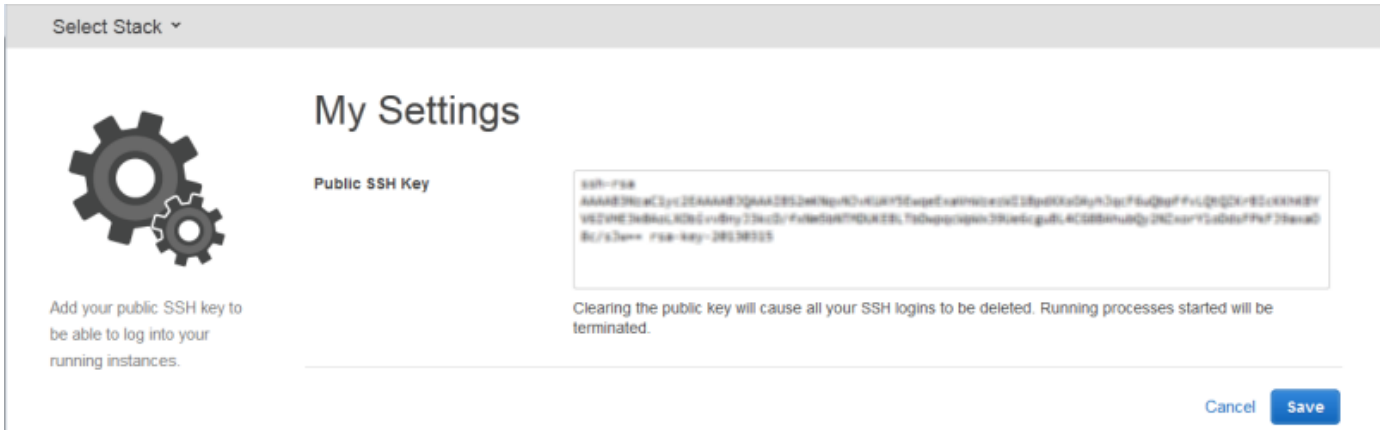
3. 내 설정을 선택합니다. 그러면 로그인된 사용자에 대한 설정이 표시됩니다.

4. 내 설정 페이지에서 편집을 클릭합니다.



5. [퍼블릭 SSH 키] 상자에 SSH 퍼블릭 키를 입력한 다음 [저장]을 클릭합니다.



> **Important**
>
> 기본 제공 MindTerm SSH 클라이언트를 사용하여 Amazon EC2 인스턴스에 연결하려면 사용자가 IAM 사용자로 로그인하고 AWS OpsWorks Stacks에 등록된 퍼블릭 SSH 키가 있어야 합니다. 자세한 내용은 내장 MindTerm SSH 클라이언트를 사용 단원을 참조하십시오.

Linux 보안 업데이트 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

보안 업데이트

Linux 운영 체제 공급자는 정기적 업데이트를 제공합니다. 이러한 업데이트는 대부분 운영 체제 보안 패치이지만 설치된 패키지에 대한 업데이트도 포함될 수 있습니다. 인스턴스의 운영 체제에 항상 최신 보안 패치가 적용되도록 해야 합니다.

기본적으로 AWS OpsWorks Stacks는 인스턴스 부팅이 완료된 후 설정 중에 최신 업데이트를 자동으로 설치합니다. AWS OpsWorks Stacks는 애플리케이션 서버 재시작과 같은 중단을 방지하기 위해 인스턴스가 온라인 상태가 된 후 업데이트를 자동으로 설치하지 않습니다. 그 대신 사용자가 온라인 인스턴스에 대한 업데이트를 직접 관리합니다. 그러므로 중단을 최소화할 수 있습니다.

다음 방법 중 하나를 사용하여 온라인 인스턴스를 업데이트할 것을 권장합니다.

- 새 인스턴스를 생성하여 현재 온라인 인스턴스를 대체합니다. 그런 다음 현재 인스턴스를 삭제합니다.

  새 인스턴스에는 설정 중 최신 보안 패치가 설치됩니다.
- Chef 11.10 또는 이전 스택의 Linux 기반 인스턴스에서는 Update Dependencies 스택 명령을 실행합니다. 이 명령은 지정된 인스턴스에 최신 보안 패치 및 기타 업데이트를 설치합니다.

이러한 두 가지 접근 방식 모두에서 AWS OpsWorks Stacks는 `yum update` Amazon Linux 및 Red Hat Enterprise Linux(RHEL) 또는 Ubuntu`apt-get update`에 대해를 실행하여 업데이트를 수행합니다. 각 배포는 업데이트를 약간 다르게 처리합니다. 그러므로 연결된 링크에서 정보를 확인하여 업데이트가 인스턴스에 어떤 영향을 미치는지 정확히 파악해야 합니다.

- Amazon Linux – Amazon Linux 업데이트는 보안 패치를 설치하며 패키지 업데이트를 포함한 기능 업데이트도 설치할 수 있습니다.

  자세한 내용은 Amazon Linux AMI FAQ를 참조하세요.
- Ubuntu – Ubuntu 업데이트는 주로 보안 패치 설치로 국한되지만, 일부 중요 수정을 위한 패키지 업데이트도 설치할 수 있습니다.

  자세한 정보는 LTS - Ubuntu Wiki를 참조하세요.
- CentOS – CentOS는 전반적으로 이전 버전과의 바이너리 호환성을 유지합니다.

- RHEL – RHEL은 전반적으로 이전 버전과의 바이너리 호환성을 유지합니다.

  자세한 정보는 Red Hat Enterprise Linux Life Cycle 단원을 참조하세요.

특정 패키지 버전을 지정하는 등 업데이트를 더 세밀하게 제어하려면 CreateInstance, UpdateInstance, CreateLayer, 또는 UpdateLayer 작업 또는 이에 상응하는 AWS SDK 방법 또는 AWS CLI 명령을 사용하여 `InstallUpdatesOnBoot` 파라미터를 `false`로 설정하여 자동 업데이트를 비활성화할 수 있습니다. 다음 예제는 AWS CLI를 사용하여 기존 계층의 기본 설정으로 `InstallUpdatesOnBoot`를 비활성화하는 방법을 보여줍니다.

```
aws opsworks update-layer --layer-id layer ID --no-install-updates-on-boot
```

그러면 사용자가 업데이트를 직접 관리해야 합니다. 예를 들어 다음 전략 중 하나를 채택할 수 있습니다.

- 적절한 shell 명령을 실행하여 선호하는 업데이트를 설치하는 사용자 지정 레시피를 구현합니다.

  시스템 업데이트가 자연스럽게 수명 주기 이벤트와 매핑되지 않으므로 사용자 지정 쿡북에 레시피를 포함하되 수동으로 레시피를 실행합니다. 패키지 업데이트의 경우 shell 명령 대신 yum_package(Amazon Linux) 또는 apt_package(Ubuntu) 리소스를 사용할 수도 있습니다.
- SSH를 사용하여 각 인스턴스에 로그인하고 수동으로 적절한 명령을 실행합니다.

보안 그룹 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

보안 그룹

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

각각의 Amazon EC2 인스턴스에는 방화벽과 비슷하게 인스턴스의 네트워크 트래픽을 관리하는 하나 이상의 보안 그룹이 연결되어 있습니다. 보안 그룹에는 허용되는 특정 트래픽 범주를 각각 지정하는 하나 이상의 규칙이 있습니다. 규칙이 지정하는 것은 다음과 같습니다.

- SSH 또는 HTTP 등 허용되는 트래픽 유형
- TCP 또는 UDP 등 트래픽의 프로토콜
- 트래픽이 발생할 수 있는 IP 주소 범위
- 트래픽의 허용되는 포트 범위

보안 그룹의 규칙 유형은 다음 두 가지입니다.

- 인바운드 규칙은 인바운드 네트워크 트래픽에 적용됩니다.

  예를 들어 애플리케이션 서버 인스턴스에는 일반적으로 모든 IP 주소에서 포트 80으로 오는 인바운드 HTTP 트래픽을 허용하는 인바운드 규칙이 있으며, 지정된 IP 주소 세트에서 포트 22로 오는 인바운드 SSH 트래픽을 허용하는 또 다른 인바운드 규칙이 있습니다.
- 아웃바운드 규칙은 아웃바운드 네트워크 트래픽에 적용됩니다.

  일반적 관행은 모든 아웃바운드 트래픽을 허용하는 기본 설정을 사용하는 것입니다.

보안 그룹에 대한 자세한 내용은 Amazon EC2 보안 그룹을 참조하세요.

리전에서 스택을 처음 생성할 때 AWS OpsWorks Stacks는 적절한 규칙 세트를 사용하여 각 계층에 대해 기본 제공 보안 그룹을 생성합니다. 모든 그룹에는 모든 아웃바운드 트래픽을 허용하는 기본 아웃바운드 규칙이 있습니다. 일반적으로 인바운드 규칙은 다음을 허용합니다.

- 적절한 AWS OpsWorks Stacks 계층의 인바운드 TCP, UDP 및 ICMP 트래픽

- 포트 22(SSH 로그인)의 인바운드 TCP 트래픽

> **Warning**
>
> 기본 보안 그룹 구성은 어느 네트워크 위치(0.0.0.0/0)에 대해서도 SSH(포트 22)를 엽니다. 이를 통해 모든 IP 주소에서 SSH를 사용하여 인스턴스에 액세스할 수 있습니다. 프로덕션 환경에서는 특정 IP 주소나 주소 범위로부터의 SSH 액세스만 허용하는 구성을 사용해야 합니다. 기본 보안 그룹을 생성 직후 업그레이드하거나 사용자 지정 보안 그룹을 대신 사용하세요.

- 웹 서버 계층의 경우, 포트 80(HTTP) 및 443(HTTPS)으로 오는 모든 인바운드 TCP 및 UDP 트래픽.

> **Note**
>
> 내장 `AWS-OpsWorks-RDP-Server` 보안 그룹이 Windows 인스턴스에 할당되어 RDP 액세스를 허용합니다. 하지만 기본적으로 이 보안 그룹에는 규칙이 없습니다. Windows 스택을 실행 중이고 RDP를 사용하여 인스턴스에 액세스하려는 경우, RDP 액세스를 허용하는 인바운드 규칙을 추가해야 합니다. 자세한 내용은 RDP를 사용하여 로그인 단원을 참조하십시오.

각 그룹의 세부 정보를 보려면 Amazon EC2 콘솔로 가서 탐색 창에서 보안 그룹을 선택한 다음 적절한 계층의 보안 그룹을 선택합니다. 예를 들어 [AWS-OpsWorks-Default-Server]는 모든 스택의 기본 내장 보안 그룹이고, [AWS-OpsWorks-WebApp]은 Chef 12 샘플 스택의 기본 내장 보안 그룹입니다.

> **Note**
>
> 실수로 AWS OpsWorks Stacks 보안 그룹을 삭제한 경우 AWS OpsWorks Stacks가 작업을 수행하도록 하는 것이 좋습니다. 동일한 AWS 리전에 새 스택을 생성하고 VPC가 있는 경우 AWS OpsWorks Stacks는 삭제한 보안 그룹을 포함하여 모든 기본 제공 보안 그룹을 자동으로 다시 생성합니다. 그런 다음 더 이상 사용할 일이 없으면 스택을 삭제할 수 있습니다. 보안 그룹은 그대로 남습니다. 보안 그룹을 수동으로 다시 생성하려는 경우, 이 보안 그룹은 그룹 이름의 대문자를 포함하여 원본의 정확한 복제본이어야 합니다.
>
> 또한 AWS OpsWorks Stacks는 다음 중 하나가 발생하는 경우 모든 기본 제공 보안 그룹을 다시 생성하려고 시도합니다.
>
> - 스택 콘솔에서 AWS OpsWorks 스택의 설정 페이지를 변경합니다.
> - 스택의 인스턴스 중 하나를 시작하는 경우.

- 새 스택을 생성하는 경우.

다음 방법 중 하나를 사용하여 보안 그룹을 지정할 수 있습니다. 스택을 생성할 때 [OpsWorks 보안 그룹 사용] 설정을 사용하여 기본 설정을 지정합니다.

- 예(기본 설정) - AWS OpsWorks 스택은 적절한 내장 보안 그룹을 각 계층에 자동으로 연결합니다.

  원하는 설정으로 사용자 지정 보안 그룹을 추가하여 계층의 내장 보안 그룹을 세부 조정할 수 있습니다. 하지만 Amazon EC2는 여러 보안 그룹을 평가할 때 가장 제한적인 규칙을 사용하므로 이 방법을 사용하여 내장 그룹보다 더 제한적인 규칙을 지정할 수는 없습니다.
- 아니요 - AWS OpsWorks Stacks는 내장 보안 그룹을 계층과 연결하지 않습니다.

  적절한 보안 그룹을 생성하고 생성하는 각 계층에 적어도 하나의 보안 그룹을 연결해야 합니다. 내장 그룹보다 더 제한적인 규칙을 지정하려면 이 방법을 사용하세요. 원한다면 수동으로 내장 보안 그룹을 계층에 연결할 수 있습니다. 사용자 지정 보안 그룹은 사용자 지정 설정이 필요한 계층에만 필요합니다.

> **Important**
>
> 내장 보안 그룹을 사용하는 경우, 그룹의 설정을 수동으로 수정하여 보다 제한적인 규칙을 생성할 수 없습니다. 스택을 생성할 때마다 AWS OpsWorks Stacks는 기본 제공 보안 그룹의 구성을 덮어쓰므로 다음에 스택을 생성할 때 변경한 내용이 모두 손실됩니다. 계층에 내장 보안 그룹보다 더 제한적인 보안 그룹 설정이 필요한 경우, [OpsWorks 보안 그룹 사용]을 [아니요]로 설정하고 기본 설정으로 사용자 지정 보안 그룹을 생성한 다음 생성 시 계층에 할당합니다.

AWS OpsWorks Chef 12 Linux에 대한 Stacks 지원

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 Chef AWS OpsWorks 12 Linux용 Stacks에 대한 간략한 개요를 제공합니다. Windows용 Chef 12에 대한 자세한 정보는 시작하기: Windows 단원을 참조하세요. 이전의 Linux용 Chef 버전에 대한 자세한 정보는 Linux용 Chef 11.10 및 이전 버전 단원을 참조하세요.

개요

AWS OpsWorks Stacks는 Linux 스택용 Chef의 최신 버전인 Chef 12를 지원합니다. 자세한 정보는 Learn Chef를 참조하세요.

AWS OpsWorks Stacks는 Linux 스택용 Chef 11.10을 계속 지원합니다. 하지만 커뮤니티 쿡북을 다수 선택하여 이득을 얻고자 하거나 본인만의 사용자 지정 쿡북을 작성하려는 고급 Chef 사용자인 경우, Chef 12를 사용할 것을 권장합니다. Chef 12 스택은 Linux용 Chef 11.10 이하 스택에 비해 다음과 같은 장점이 있습니다.

- 두 개의 별도 Chef 실행 - 인스턴스에서 명령이 실행되면 AWS OpsWorks Stacks 에이전트는 이제 두 개의 격리된 Chef 실행을 실행합니다. 하나는 인스턴스를 AWS Identity and Access Management (IAM)과 같은 다른 AWS 서비스와 통합하는 작업에 대한 실행이고 다른 하나는 사용자 지정 쿡북에 대한 실행입니다. 첫 번째 Chef 실행은 인스턴스에 AWS OpsWorks Stacks 에이전트를 설치하고 사용자 설정 및 관리, 볼륨 설정 및 구성, CloudWatch 지표 구성 등과 같은 시스템 작업을 수행합니다. 두 번째 실행은 AWS OpsWorks Stacks 수명 주기 이벤트에서 사용자 지정 레시피를 실행하기 위한 전용 실행입니다. 이 두 번째 실행으로 사용자가 자체 Chef 쿡북 또는 커뮤니티 쿡북을 사용할 수 있습니다.
- 네임스페이스 충돌 해결 - Chef 12 이전에는 AWS OpsWorks Stacks가 공유 환경에서 시스템 작업을 수행하고 내장 및 사용자 지정 레시피를 실행했습니다. 이로 인해 네임스페이스 충돌이 발생했고 AWS OpsWorks Stacks가 실행한 레시피에 대한 명확성이 부족했습니다. 원치 않는 기본 구성을 수동으로 덮어써야 했기 때문에 시간이 오래 걸리고 실수가 발생할 수 있었습니다. Linux용 Chef 12에서 AWS OpsWorks Stacks는 더 이상 PHP, Node.js 또는 Rails와 같은 애플리케이션 서버 환경을 위한 내장 Chef 쿡북을 지원하지 않습니다. 기본 제공 레시피를 제거하여 AWS OpsWorks Stacks는 기본 제공 레시피와 사용자 지정 레시피 간의 이름 지정 충돌 문제를 제거합니다.
- Chef 커뮤니티 쿡북에 대한 강력한 지원 - AWS OpsWorks Stacks Chef 12 Linux는 Chef 수퍼마켓의 커뮤니티 쿡북에 대한 더 큰 호환성과 지원을 제공합니다. 이제 Stacks가 이전에 제공한 내장 쿡북, 즉 최신 애플리케이션 서버 환경 및 프레임워크와 함께 사용하도록 설계된 쿡북보다 우수한 커뮤니티 AWS OpsWorks 쿡북을 사용할 수 있습니다. 이러한 쿡북은 대부분 수정 없이 Linux용 Chef 12에서 실행할 수 있습니다. 자세한 내용은 Learn Chef 웹사이트의 Chef 슈퍼마켓, Chef 슈퍼마켓 웹사이트, GitHub의 Chef 쿡북 리포지토리를 참조하세요.
- 적시 Chef 12 업데이트 - AWS OpsWorks Stacks는 각 Chef 릴리스 직후 Chef 환경을 최신 Chef 12 버전으로 업데이트합니다. Chef 12에서는 마이너 Chef 업데이트와 새로운 AWS OpsWorks Stacks

에이전트 릴리스가 동시에 적용됩니다. 그러므로 Chef 릴리스를 직접 테스트할 수 있고, Chef 레시피 및 애플리케이션이 최신 Chef 기능을 활용할 수 있습니다.

Chef 12 이전의 지원되는 Chef 버전에 대한 자세한 정보는 Linux용 Chef 11.10 및 이전 버전 단원을 참조하세요.

Chef 12로 전환

이전 Chef 버전 AWS OpsWorks 11.10, 11.4 및 0.9에 대한 지원과 비교하여 Chef 12 Linux에 대한 Key Stacks 변경 사항은 다음과 같습니다.

- Linux용 Chef 12 스택에서 내장 계층이 더 이상 제공 또는 지원되지 않습니다. 사용자 지정 레시피만 실행되므로 이 지원을 생략함으로써 인스턴스 설정 방식이 완전히 투명해지고 사용자 지정 쿡북을 훨씬 간편하게 작성 및 유지관리할 수 있습니다. 예를 들어 더 이상 내장 AWS OpsWorks Stacks 레시피의 속성을 덮어쓸 필요가 없습니다. 또한 내장 계층을 제거하면 AWS OpsWorks Stacks가 Chef 커뮤니티에서 개발하고 유지 관리하는 쿡북을 더 잘 지원할 수 있으므로 이를 최대한 활용할 수 있습니다. Linux용 Chef 12에서 더 이상 사용할 수 없는 내장 계층 유형은 AWS Flow(Ruby), Ganglia, HAProxy, Java 앱 서버, Memcached, MySQL, Node.js 앱 서버, PHP 앱 서버, Rails 앱 서버 및 Static Web Server입니다.
  - Stacks는 사용자가 제공하는 레시피를 실행하므로 더 이상 사용자 지정 AWS OpsWorks 쿡북을 실행하여 기본 제공 AWS OpsWorks Stacks 속성을 재정의할 필요가 없습니다. 자체 또는 커뮤니티 레시피에서 속성을 재정의하려면 Chef 12 설명서의 About Attributes에서 제공하는 지침 및 예제를 따릅니다.
- AWS OpsWorks Stacks는 Chef 12 Linux 스택에 대해 다음 계층에 대한 지원을 계속 제공합니다.
  - 사용자 지정 계층
  - Amazon RDS 서비스 계층
  - ECS 클러스터 계층
- Chef 12 Linux용 스택 구성 및 데이터 백이 Chef 12.2 Windows의 스택 구성 및 데이터 백과 매우 비슷하게 변경되었습니다. 그러므로 특히 운영 체제 유형이 서로 다른 스택에서 작업할 경우 이러한 데이터 백을 보다 간편하게 쿼리, 분석 및 문제 해결할 수 있습니다. AWS OpsWorks Stacks는 암호화된 데이터 백을 지원하지 않습니다. 암호나 인증서 등 암호화된 형식의 민감한 데이터를 저장해야 하는 경우, 프라이빗 S3 버킷에 저장하는 것이 좋습니다. 그런 다음 Ruby용 Amazon SDK를 사용하는 사용자 지정 레시피를 생성해 데이터를 검색할 수 있습니다. 자세한 정보는 SDK for Ruby 사용 단원 및 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하세요.
- Chef 12 Linux에서는 더 이상 Berkshelf가 스택 인스턴스에 설치되지 않습니다. 그 대신 권장되는 방법으로, 로컬 개발 시스템에서 Berkshelf를 사용하여 로컬에서 쿡북 종속성을 패키징합니다. 그런 다

음 종속성이 포함된 패키지를 Amazon Simple Storage Service로 업로드합니다. 마지막으로 쿡북 소스로 업로드된 패키지를 사용하도록 Chef 12 Linux 스택을 수정합니다. 자세한 내용은 로컬로 쿡북 종속성 패키징 단원을 참조하십시오.

- 더 이상 EBS 볼륨에 대한 RAID 구성이 지원되지 않습니다. 성능을 높이려면 Amazon Elastic Block Store(Amazon EBS)에 대한 프로비저닝된 IOPS를 사용할 수 있습니다.
- 더 이상 autofs가 지원되지 않습니다.
- 더 이상 하위 버전 리포지토리가 지원되지 않습니다.
- 이제 계층별 OS 패키지 설치가 사용자 지정 레시피를 통해 이루어져야 합니다. 자세한 내용은 계층별 패키지 설치 단원을 참조하십시오.

지원되는 운영 체제

Chef 12는 이전 버전의 Chef와 동일한 Linux 운영 체제를 지원합니다. Chef 12 Linux 스택이 사용할 수 있는 Linux 운영 체제 유형 및 버전은 Linux 운영 체제 단원을 참조하세요.

지원되는 인스턴스 유형

AWS OpsWorks Stacks는 고성능 컴퓨팅(HPC) 클러스터 컴퓨팅, 클러스터 GPU, 고 메모리 클러스터 인스턴스 유형과 같은 특수 인스턴스 유형을 제외하고 Chef 12 Linux 스택에 대한 모든 인스턴스 유형을 지원합니다.

추가 정보

Linux용 Chef 12 스택에서 작업하는 방법을 자세히 알아보려면 다음 단원을 참조하세요.

- 시작하기: 샘플

  AWS OpsWorks Stacks 콘솔을 사용한 간단한 실습을 통해 Node.js 애플리케이션 환경을 생성하여 AWS OpsWorks Stacks를 소개합니다.

- 시작하기: Linux

  AWS OpsWorks Stacks 콘솔을 사용한 실습을 통해 트래픽을 제공하는 Node.js 앱으로 간단한 계층이 포함된 기본 Chef 12 Linux 스택을 생성하여 AWS OpsWorks Stacks 및 Chef 12 Linux를 소개합니다.

- 사용자 지정 계층

쿡북 및 레시피를 포함하는 계층을 Chef 12 Linux 스택에 추가하는 지침을 제공합니다. Chef 커뮤니티에서 제공하는 즉시 사용 가능한 쿡북 및 레시피를 사용하거나 자체 쿡북 및 레시피를 생성할 수 있습니다.

- 데이터 백으로 이전

  Chef 11 이하 버전을 실행하는 Linux 스택에서 사용하는 인스턴스 JSON과 Chef 12를 실행하는 Linux 스택에서 사용하는 인스턴스 JSON을 비교합니다. 또한 Chef 12 인스턴스 JSON 형식의 참조 설명서에 대한 링크도 제공합니다.

스택 설정을 속성에서 데이터 백으로 이전

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 다양한 스택 설정을 Chef 레시피에 노출합니다. 이러한 스택 설정에는 다음과 같은 값이 포함됩니다.

- 스택 쿡북 소스 URL
- 계층 볼륨 구성
- 인스턴스 호스트 이름
- Elastic Load Balancing DNS 이름
- 앱 소스 URL
- 사용자 이름

레시피로부터 스택 설정 단원을 참조하면 레시피 코드가 레시피에서 직접 설정을 하드코딩하는 것에 비해 더 강력해지고 오류 발생이 낮아집니다. 이 주제에서는 이러한 스택 설정에 액세스하는 방법과 Linux용 Chef 11.10 및 이전 버전의 속성에서 Chef 12 Linux의 데이터 백으로 스택 설정을 이전하는 방법을 설명합니다.

Linux용 Chef 11.10 및 이전 버전에서는 스택 설정이 Chef 속성으로 제공되며 Chef node 객체 또는 Chef 검색을 통해 액세스됩니다. 이러한 속성은 `/var/lib/aws/opsworks/chef` 디렉터리의 JSON 파일 세트에 있는 AWS OpsWorks Stacks 인스턴스에 저장됩니다. 자세한 내용은 스택 구성 및 배포 속성: Linux 단원을 참조하십시오.

Chef 12 Linux에서는 스택 설정이 Chef 데이터 백으로 제공되며 Chef 검색을 통해서만 액세스됩니다. 데이터 백은 `/var/chef/runs/`*`run-ID`*`/data_bags` 디렉터리의 JSON 파일 세트에 있는 AWS OpsWorks Stacks 인스턴스에 저장됩니다. 여기서 *`run-ID`*는 AWS OpsWorks Stacks가 인스턴스에서 실행되는 각 Chef에 할당하는 고유 ID입니다. 스택 설정은 더 이상 Chef 속성으로 제공되지 않으며, 따라서 더 이상 Chef node 객체를 통해 스택 설정에 액세스할 수 없습니다. 자세한 정보는 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하십시오.

예를 들어 Linux용 Chef 11.10 및 이전 버전에서 다음 레시피 코드는 Chef node 객체를 사용하여 앱의 짧은 이름과 소스 URL을 표시하는 속성을 가져옵니다. 그런 다음 Chef 로그를 사용하여 두 속성 값을 기록합니다.

```
Chef::Log.info ("********** The app's short name is '#{node['opsworks']
['applications'].first['slug_name']}' **********")
Chef::Log.info("********** The app's URL is '#{node['deploy']['simplephpapp']['scm']
['repository']}' **********")
```

Chef 12 Linux에서는 다음 코드가 `aws_opsworks_app` 검색 인덱스를 사용하여 `aws_opsworks_app` 데이터 백에서 첫 번째 데이터 백 항목의 콘텐츠를 가져옵니다. 그런 다음 코드는 Chef 로그에 메시지 2개를 기록합니다. 한 메시지는 앱의 짧은 이름 데이터 백 콘텐츠를 포함하고 다른 메시지는 앱의 소스 URL 데이터 백 콘텐츠를 포함합니다.

```
app = search("aws_opsworks_app").first

Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

Linux용 Chef 11.10 및 이전 버전의 스택 설정에 액세스하는 레시피 코드를 Chef 12 Linux로 마이그레이션하려면 코드를 다음과 같이 수정해야 합니다.

- Chef 속성이 아니라 Chef 데이터 백에 액세스
- Chef node 객체 대신 Chef 검색을 사용
- `opsworks` 및와 같은 AWS OpsWorks Stacks 속성 이름을 사용하는 `aws_opsworks_app`대신와 같은 AWS OpsWorks Stacks 데이터 백 이름을 사용합니다`deploy`.

자세한 정보는 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하십시오.

AWS OpsWorks Stacks에서 이전 Chef 버전 지원

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 이전 Chef 버전의 AWS OpsWorks Stacks 설명서에 대한 간략한 개요를 제공합니다.

Linux용 Chef 11.10 및 이전 버전

Linux 스택용 Chef AWS OpsWorks 11.10, 11.4 및 0.9에 대한 Stacks 지원에 대한 설명서를 제공합니다.

Linux용 Chef 11.10 및 이전 버전

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 Linux용 Chef AWS OpsWorks 11.10, 11.4 및 0.9용 Stacks 설명서에 대한 간략한 개요를 제공합니다.

Chef 11 Linux 스택 시작하기

간단하지만 실용적인 PHP 애플리케이션 서버 스택을 생성하는 방법을 안내합니다.

첫 번째 Node.js 스택 생성

Node.js 애플리케이션 서버를 지원하는 Linux 스택을 생성하고 간단한 애플리케이션을 배포하는 방법을 설명합니다.

[AWS OpsWorks 스택 사용자 지정]

특정 요구 사항에 맞게 AWS OpsWorks Stacks를 사용자 지정하는 방법을 설명합니다.

[쿡북 101]

AWS OpsWorks Stacks 인스턴스에 대한 레시피를 구현하는 방법을 설명합니다.

[계층 로드 밸런싱]

사용 가능한 AWS OpsWorks Stacks 로드 밸런싱 옵션을 사용하는 방법을 설명합니다.

[VPC에서 스택 실행]

가상 프라이빗 클라우드에서 스택을 생성하고 실행하는 방법을 설명합니다.

[Chef Server에서 마이그레이션]

Chef Server에서 AWS OpsWorks Stacks로 마이그레이션하기 위한 지침을 제공합니다.

[AWS OpsWorks Stacks 계층 참조]

사용 가능한 AWS OpsWorks Stacks 기본 제공 계층을 설명합니다.

[쿡북 구성 요소]

세 표준 쿡북 구성 요소인 속성, 템플릿 및 레시피를 설명합니다.

[스택 구성 및 배포 속성: Linux]

Linux용 스택 구성과 배포 속성을 설명합니다.

[내장 쿡북 속성]

내장 레시피 속성을 사용하여 설치된 소프트웨어의 구성을 제어하는 방법을 설명합니다.

[Linux용 Chef 11.10 및 이전 버전 문제 해결]

AWS OpsWorks Stacks에서 다양한 문제를 해결하기 위한 접근 방식을 설명합니다.

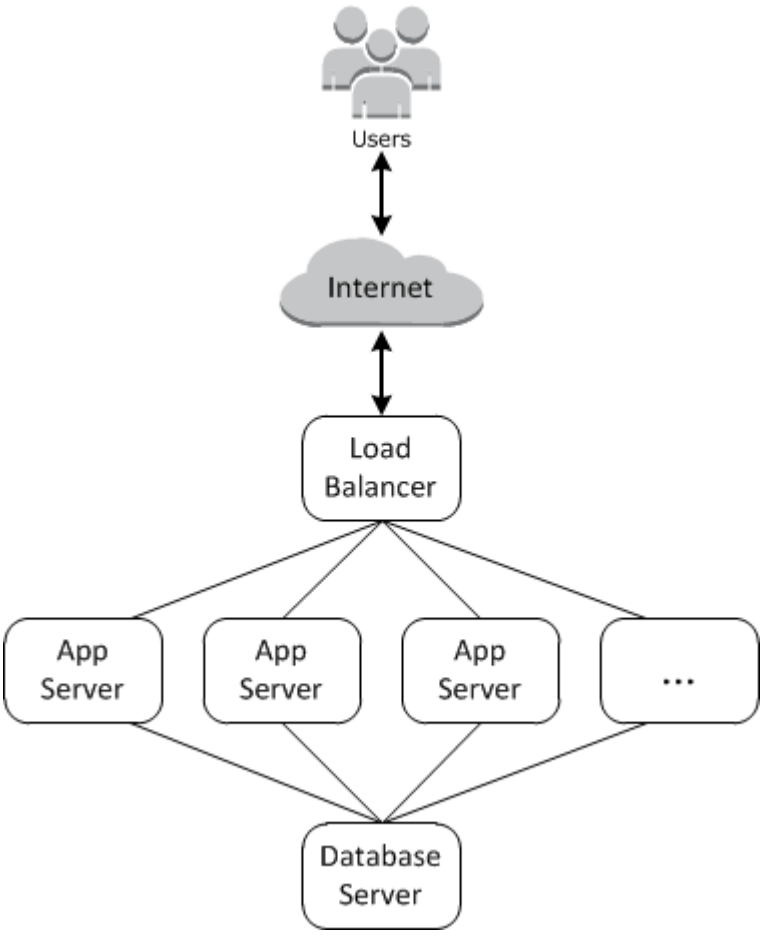
Chef 11 Linux 스택 시작하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 섹션에서는 Chef 11을 사용하여 Linux를 시작하는 방법을 설명합니다. Chef 12 Linux 스택 시작하기에 대한 자세한 정보는 시작하기: Linux 단원을 참조하세요. Chef 12 Windows 스택 시작하기에 대한 자세한 정보는 시작하기: Windows 단원을 참조하세요.

클라우드 기반 애플리케이션에는 일반적으로 애플리케이션 서버, 데이터베이스 서버 등과 같은 관련 리소스 그룹이 필요하며 이러한 리소스를 공동으로 생성하고 관리해야 합니다. 이러한 인스턴스의 모음을 스택이라고 합니다. 다음은 간단한 애플리케이션 스택의 예입니다.



기본 아키텍처는 다음과 같이 구성됩니다.

- 사용자로부터 트래픽을 수신하여 애플리케이션 서버 간에 고르게 분산하는 로드 밸런서 1개

- 트래픽을 처리하는 데 필요한 만큼의 애플리케이션 서버 인스턴스 집합
- 애플리케이션 서버에 백엔드 데이터 스토어를 제공하기 위한 데이터베이스 서버

또한, 일반적으로 애플리케이션을 애플리케이션 서버에 배포하고 스택을 모니터링하는 등의 기능이 필요합니다.

AWS OpsWorks Stacks는 스택과 관련 애플리케이션 및 리소스를 생성하고 관리하는 간단하고 간단한 방법을 제공합니다. 이 장에서는 다이어그램에서 애플리케이션 서버 스택을 생성하는 프로세스를 단계별로 안내하여 AWS OpsWorks Stacks의 기본 사항과 더욱 정교한 기능을 소개합니다. AWS OpsWorks Stacks가 따르기 쉽게 만드는 증분 개발 모델을 사용합니다. 기본 스택을 설정하고 올바르게 작동하면 모든 기능을 갖춘 구현에 도달할 때까지 구성 요소를 추가합니다.

- 1단계: 사전 조건 완료에서는 이 안내서를 시작하기 위해 필요한 사전 준비를 설명합니다.
- 2단계: 간단한 애플리케이션 서버 스택 생성 - Chef 11에서는 단일 애플리케이션 서버로 구성된 최소 스택을 생성하는 방법을 설명합니다.
- 3단계; 백엔드 데이터 스토어 추가에서는 데이터베이스 서버를 추가하고 애플리케이션 서버에 연결하는 방법을 설명합니다.
- 4단계: MyStack 확장에서는 증가하는 로드를 처리하기 위해 더 많은 애플리케이션 서버와 수신 트래픽을 분배할 로드 밸런서를 추가하여 스택을 확장하는 방법을 설명합니다.

주제

- 1단계: 사전 조건 완료
- 2단계: 간단한 애플리케이션 서버 스택 생성 - Chef 11
- 3단계; 백엔드 데이터 스토어 추가
- 4단계: MyStack 확장
- 5단계: MyStack 삭제

1단계: 사전 조건 완료

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

안내서를 시작하기 전에 다음 설정 단계를 완료합니다. 이러한 설정 단계에는 AWS 계정 가입, 관리 사용자 생성, AWS OpsWorks Stacks에 액세스 권한 할당이 포함됩니다.

이미 AWS OpsWorks Stacks 시작하기 안내서 중 하나를 마쳤다면 이 안내서의 사전 조건을 충족한 것이며, 2단계: 간단한 애플리케이션 서버 스택 생성 - Chef 11으 단원으로 건너뛸 수 있습니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- 사용자에게 서비스 액세스 권한 할당

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

사용자에게 서비스 액세스 권한 할당

역할 또는 사용자에게 `AWSOpsWorks_FullAccess` 및 `AmazonS3FullAccess` 권한을 추가하여 AWS OpsWorks Stacks 서비스(및 AWS OpsWorks Stacks가 의존하는 관련 서비스)에 대한 액세스를 활성화합니다.

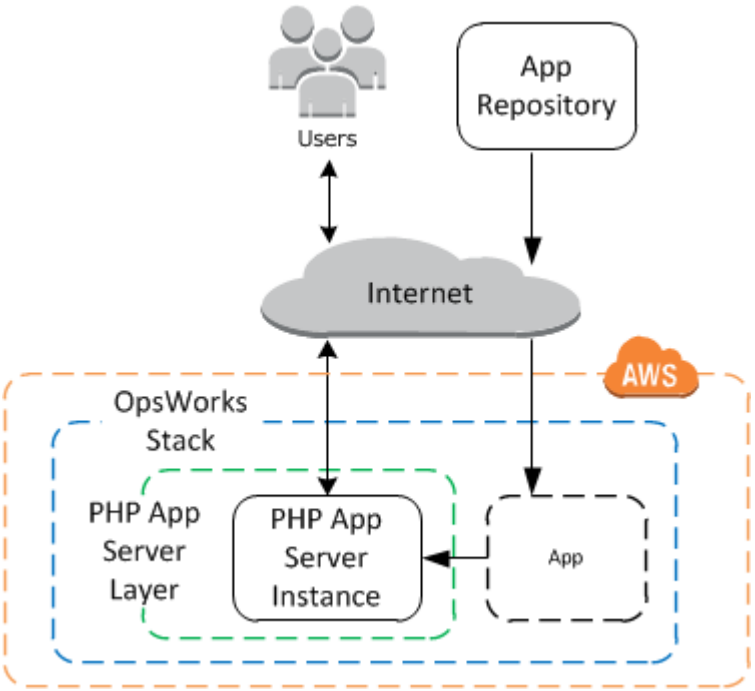
권한 추가에 대한 자세한 내용은 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

이제 모든 설정 단계를 마쳤으므로 이 안내서를 시작할 수 있습니다.

2단계: 간단한 애플리케이션 서버 스택 생성 - Chef 11

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

기본 애플리케이션 서버 스택은 사용자 요청을 수신하기 위한 퍼블릭 IP 주소를 가진 단일 애플리케이션 서버 인스턴스로 구성됩니다. 애플리케이션 코드 및 관련 파일은 별도의 리포지토리에 저장되다가 서버로 배포됩니다. 다음 다이어그램은 이러한 스택을 보여줍니다.



스택은 다음 구성 요소를 갖습니다.

- 계층 - 인스턴스의 그룹을 나타내며 인스턴스 그룹을 어떻게 구성할지를 지정합니다.

  이 예제에서 계층은 PHP 앱 서버 인스턴스의 그룹을 나타냅니다.

- Amazon EC2 인스턴스를 나타내는 인스턴스입니다.

  이 예제에서 인스턴스는 PHP 앱 서버를 실행하도록 구성됩니다. 계층에는 원하는 수의 인스턴스가 있을 수 있습니다. AWS OpsWorks Stacks는 다른 여러 앱 서버도 지원합니다. 자세한 내용은 애플리케이션 서버 계층 단원을 참조하십시오.

- 앱 - 애플리케이션 서버에 애플리케이션을 설치하는 데 필요한 정보를 포함합니다.

  코드는 원격 리포지토리(예: Git 리포지토리 또는 Amazon S3 버킷)에 저장됩니다.

다음 섹션에서는 AWS OpsWorks Stacks 콘솔을 사용하여 스택을 생성하고 애플리케이션을 배포하는 방법을 설명합니다. AWS CloudFormation 템플릿을 사용하여 스택을 프로비저닝할 수도 있습니다. 이 항목에서 설명하는 스택을 프로비저닝하는 템플릿 예제는 AWS OpsWorks 조각을 참조하세요.

주제

- 2.1 단계: 스택 생성 - Chef 11
- 2.2단계: PHP 앱 서버 계층 추가 - Chef 11
- 2.3단계: PHP 앱 서버 계층에 인스턴스 추가 - Chef 11
- 2.4단계: 앱 생성 및 배포 - Chef 11

2.1 단계: 스택 생성 - Chef 11

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스 및 기타 리소스의 컨테이너 역할을 하는 스택을 생성하여 AWS OpsWorks Stacks 프로젝트를 시작합니다. 스택 구성은 AWS 리전, 기본 운영 체제 등 스택의 모든 인스턴스가 공유하는 일부 기본 설정을 지정합니다.

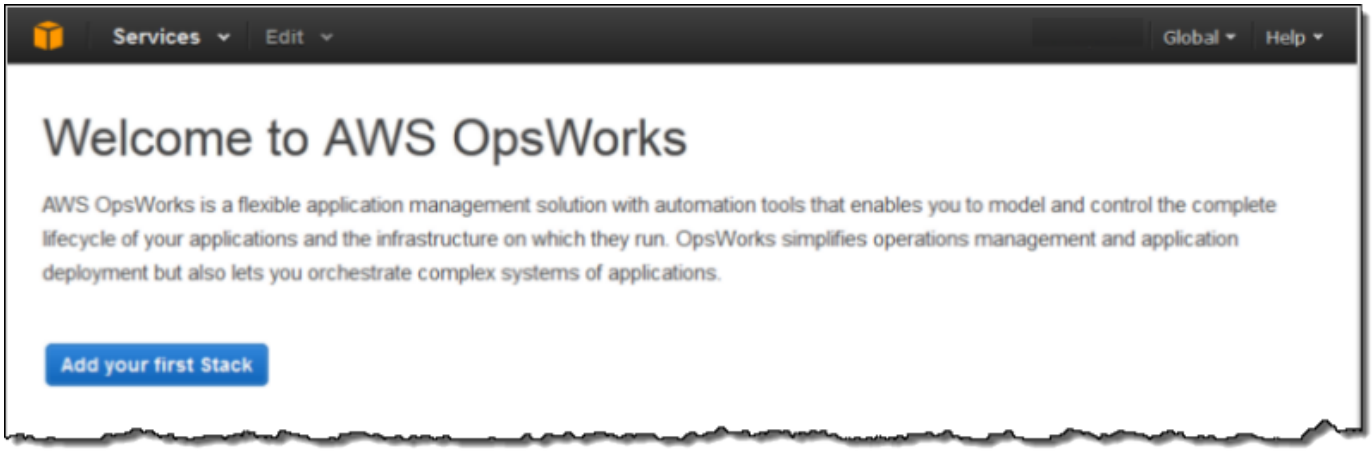
> **Note**
>
> 이 페이지에서는 Chef 11 스택을 생성하는 방법을 설명합니다. Chef 12 스택을 생성하는 방법에 대한 자세한 정보는 스택 생성 단원을 참조하세요.

이 페이지에서는 Chef 11에서 스택을 생성하는 방법을 설명합니다.

새 스택을 생성하려면

1. stack 추가

   AWS OpsWorks Stacks 콘솔에 로그인합니다. 계정에 기존 스택이 없는 경우에는 [AWS OpsWorks 시작하기] 페이지가 표시되는데 [첫 번째 스택 추가]를 클릭합니다. 그렇지 않으면 계정의 AWS OpsWorks 스택을 나열하는 Stacks 대시보드가 표시됩니다. 스택 추가를 클릭합니다.



2. 스택 구성

   [스택 추가] 페이지에서 [Chef 11 스택]을 선택하고 다음 설정을 지정합니다.

   스택 이름

   스택의 이름을 입력합니다. 이 이름에는 영숫자(a-z, A-Z 및 0-9) 및 하이픈(-)을 사용할 수 있습니다. 이 연습에서 사용되는 예제 스택의 이름은 **`MyStack`**입니다.

   리전

   스택 리전으로 미국 서부(오레곤)를 선택합니다.

다른 설정에 대해서는 기본값을 수락하고 [스택 추가]를 클릭합니다. 다양한 스택 설정에 대한 자세한 정보는 새 스택 생성 단원을 참조하세요.

2.2단계: PHP 앱 서버 계층 추가 - Chef 11

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택은 기본적으로 인스턴스의 컨테이너이지만 인스턴스를 스택에 직접 추가하는 것은 아닙니다. 관련된 인스턴스의 그룹을 나타내는 계층을 추가한 후 인스턴스를 계층에 추가합니다.

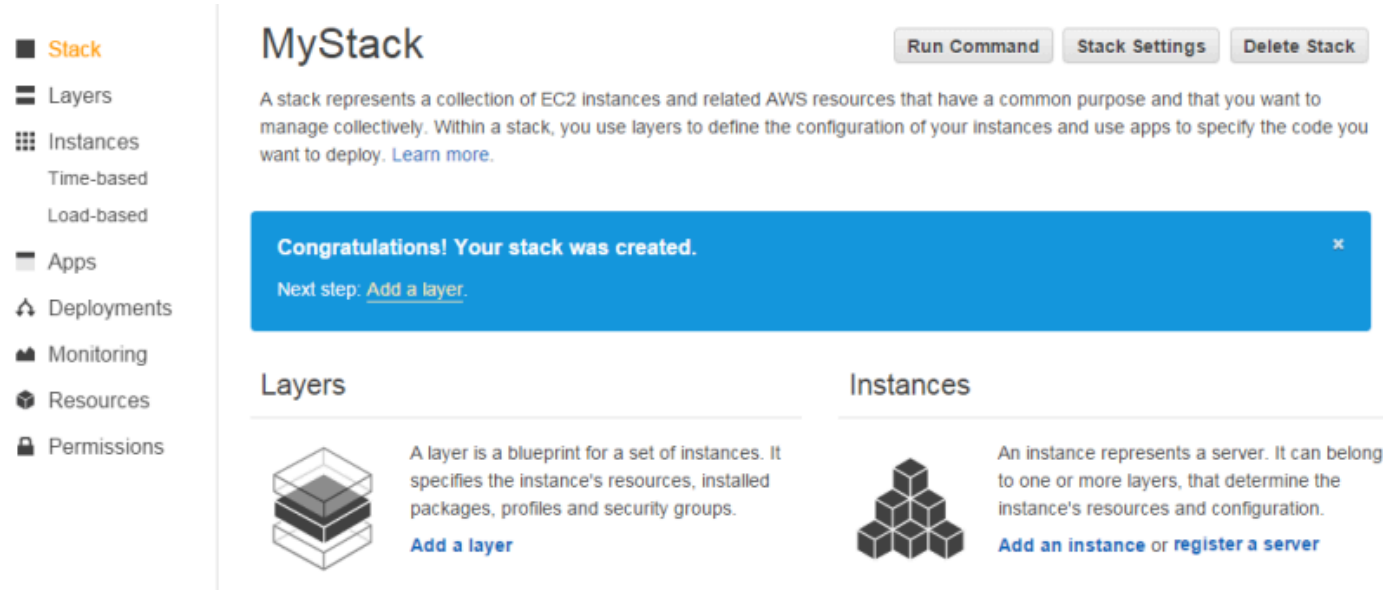
계층은 기본적으로 AWS OpsWorks Stacks가 동일한 구성으로 Amazon EC2 인스턴스 세트를 생성하는 데 사용하는 블루프린트입니다. 관련된 인스턴스의 그룹 각각에 대해 하나의 계층을 스택에 추가합니다. AWS OpsWorks Stacks에는 MySQL 데이터베이스 서버 또는 PHP 애플리케이션 서버와 같은 표준 소프트웨어 패키지를 실행하는 인스턴스 그룹을 나타내는 기본 제공 계층 세트가 포함되어 있습니다. 또한 사용자가 특정 요구 사항에 맞춰 부분적으로 또는 완전히 사용자 지정된 계층을 생성할 수 있습니다. 자세한 내용은 AWS OpsWorks 스택 사용자 지정 섹션을 참조하세요.

MyStack에는 PHP 애플리케이션 서버로 작동하는 인스턴스의 그룹을 나타내는 내장 PHP 앱 서버 계층 하나가 포함되어 있습니다. 내장 계층에 대한 설명을 포함하여 자세한 정보는 계층 단원을 참조하세요.

MyStack에 PHP 앱 서버 계층을 추가하려면

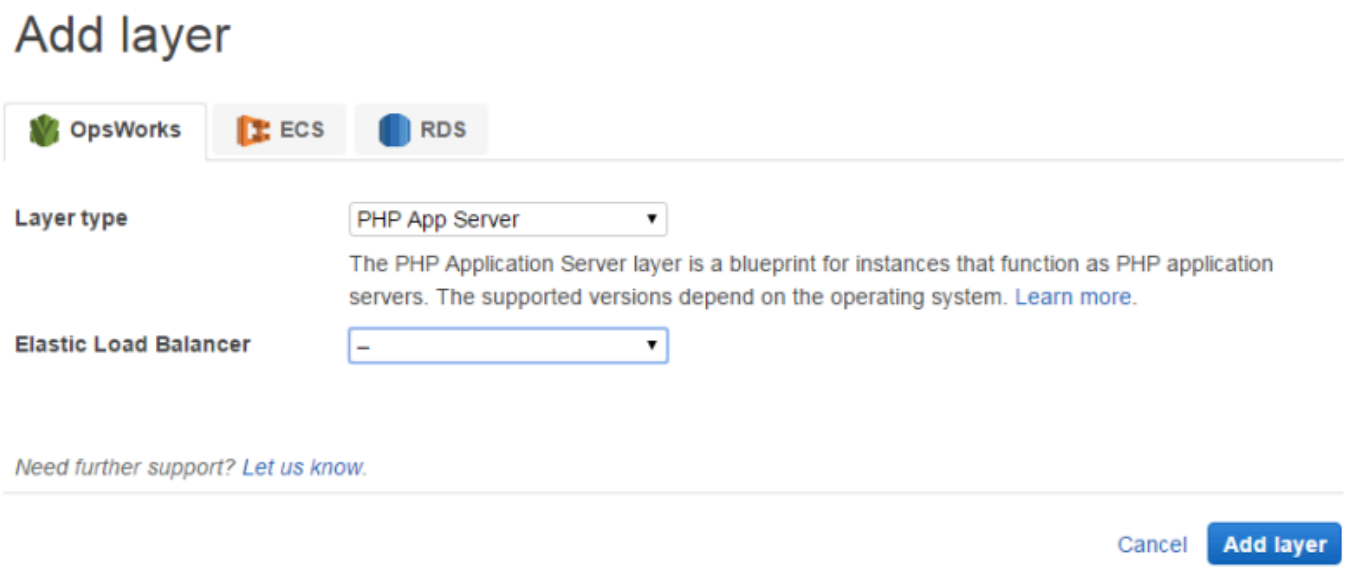
1. 계층 추가 페이지 열기

   스택 생성을 완료하면 AWS OpsWorks 스택에 스택 페이지가 표시됩니다. [계층 추가]를 클릭하여 첫 번째 계층을 추가합니다.



2. 계층 유형 지정 및 계층 구성

   계층 유형 상자에서 PHP 앱 서버를 선택하고 기본 Elastic Load Balancer 설정을 수락한 다음 계층 추가를 클릭합니다. 계층을 생성한 후에는 계층을 편집하여 EBS 볼륨 구성과 같은 다른 속성을 지정할 수 있습니다.



2.3단계: PHP 앱 서버 계층에 인스턴스 추가 - Chef 11

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 인스턴스는 특정 Amazon EC2 인스턴스를 나타냅니다.

- 인스턴스의 구성은 Amazon EC2 운영 체제 및 크기와 같은 일부 기본 설정을 지정하며, 실행은 되지만 그다지 많은 역할을 수행하지는 않습니다.
- 인스턴스의 계층은 설치할 패키지, 인스턴스가 탄력적 IP 주소를 갖는지 여부 등을 결정하여 인스턴스에 기능을 추가합니다.

AWS OpsWorks Stacks는 서비스와 상호 작용하는 각 인스턴스에 에이전트를 설치합니다. 인스턴스에 계층의 기능을 추가하기 위해 AWS OpsWorks Stacks는 에이전트에게 Chef 레시피라는 작은 애플리케이션을 실행하도록 지시합니다.이 레시피는 애플리케이션 및 패키지를 설치하고 구성 파일을 생성하는 등의 작업을 수행할 수 있습니다. AWS OpsWorks Stacks는 인스턴스 수명 주기의 주요 지점에서 레시피를 실행합니다. 예를 들어 OpsWorks는 인스턴스가 소프트웨어 설치와 같은 작업을 처리하기 위해 부팅을 완료하면 설정 레시피를 실행하고, 사용자가 코드 및 관련 파일을 설치하기 위해 앱을 배포할 때 Deploy 레시피를 실행합니다.

> **Note**
>
> 레시피의 작동 방식에 대해 궁금한 점이 있는 경우 모든 AWS OpsWorks Stacks 내장 레시피는 퍼블릭 GitHub 리포지토리인 OpsWorks 쿡북에 있습니다. 또한 자체 사용자 지정 레시피를 생성하고, 나중에 설명하는 것처럼 AWS OpsWorks Stacks가 이러한 레시피를 실행하게 할 수 있습니다.

MyStack에 PHP 애플리케이션 서버를 추가하려면 이전 단계에서 생성한 PHP 앱 서버 계층에 인스턴스를 추가합니다.

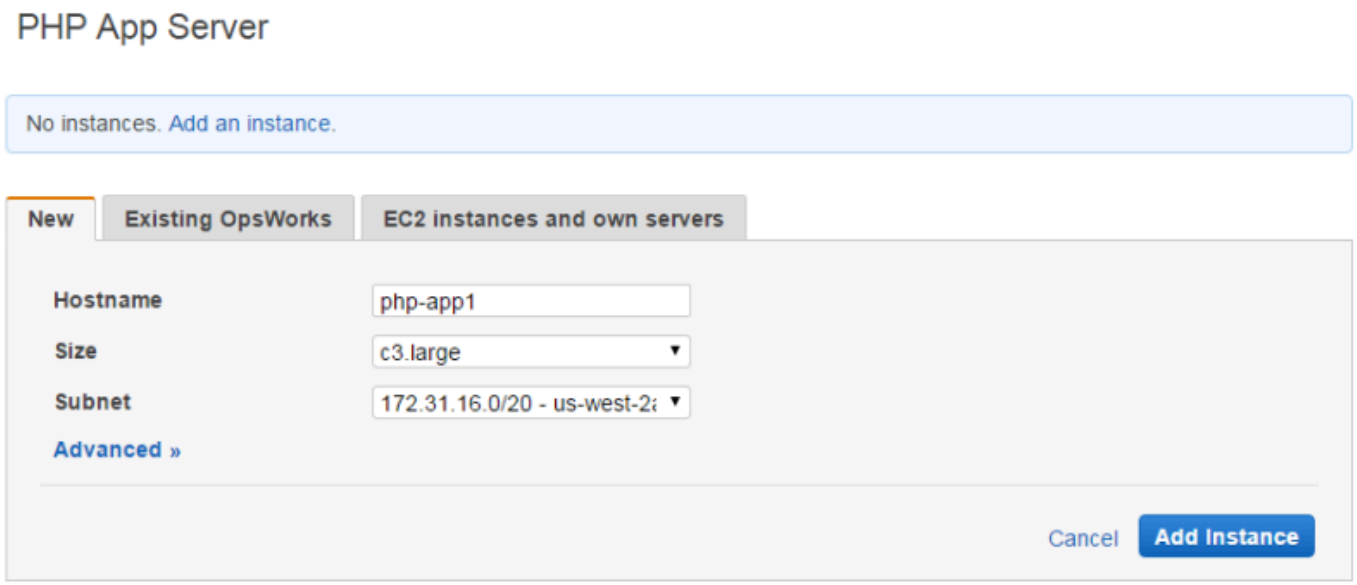
PHP 앱 서버 계층에 인스턴스를 추가하려면

1. 인스턴스 추가 열기

   계층 추가를 완료하면 AWS OpsWorks Stacks에 계층 페이지가 표시됩니다. 탐색 창에서 인스턴스를 클릭한 다음 PHP 앱 서버에서 인스턴스 추가를 클릭합니다.

2. 인스턴스 구성

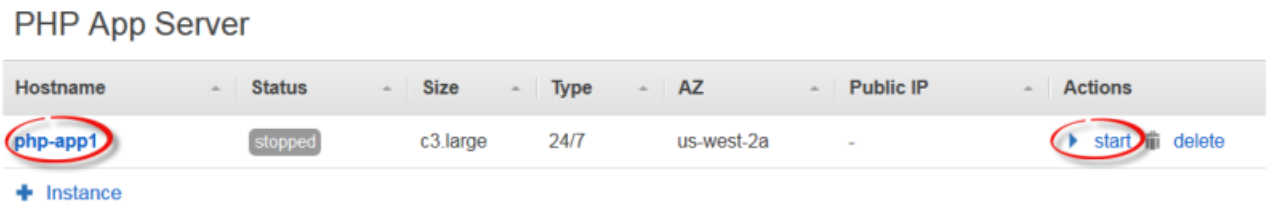
   각 인스턴스에는 AWS OpsWorks Stacks에서 생성한 기본 호스트 이름이 있습니다. 이 예제에서 AWS OpsWorks Stacks는 계층의 짧은 이름에 숫자를 추가하기만 하면 됩니다. 스택 생성 시 지정한 몇 가지 기본 설정(예: 가용 영역 또는 운영 체제) 재정의를 비롯하여 각 인스턴스를 개별적으로 구성할 수 있습니다. 이 연습의 경우 기본 설정을 수락하고 [인스턴스 추가]를 클릭하여 계층에 인스턴스를 추가합니다. 자세한 내용은 인스턴스 섹션을 참조하세요.



3. 인스턴스 시작

   지금까지 인스턴스의 구성을 지정했습니다. 실행 중인 Amazon EC2 인스턴스를 생성하려면 인스턴스를 시작해야 합니다. 그런 다음 AWS OpsWorks Stacks는 구성 설정을 사용하여 지정된 가용 영역에서 Amazon EC2 인스턴스를 시작합니다. 자세한 인스턴스 시작 방법은 인스턴스의 조정 유형에 따라 다릅니다. 이전 단계에서는 기본 조정 유형인 24/7을 사용하여 인스턴스를 생성했습니다. 이 조정 유형은 수동으로 시작해야 하며 시작되면 수동으로 중지될 때까지 실행됩니다. AWS OpsWorks Stacks가 일정 또는 현재 로드에 따라 자동으로 시작하고 중지하는 시간 기반 및 로드 기반 조정 유형을 생성할 수도 있습니다. 자세한 내용은 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리 단원을 참조하십시오.

PHP 앱 서버의 php-app1로 이동한 다음 행의 작업 열에서 시작을 클릭하여 인스턴스를 시작합니다.

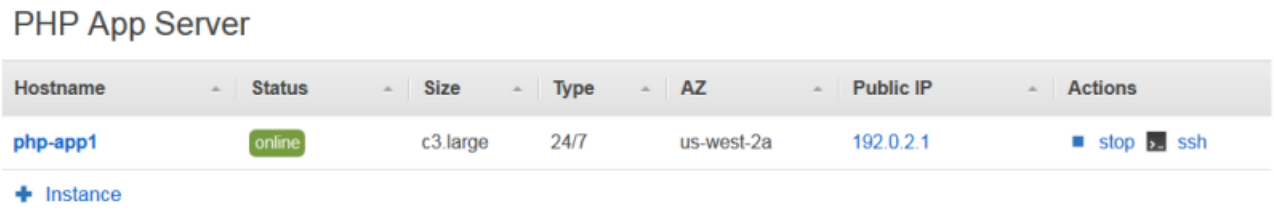


4. 시작 중 인스턴스의 상태 모니터링

Amazon EC2 인스턴스를 부팅하고 패키지를 설치하는 데는 일반적으로 몇 분 가량 소요됩니다. 시작 프로세스가 진행되는 동안 인스턴스의 [상태] 필드에 다음 값이 차례로 표시됩니다.

1. 요청됨 - AWS OpsWorks Stacks가 Amazon EC2 서비스를 호출하여 Amazon EC2 인스턴스를 생성했습니다.
2. 보류 중 - AWS OpsWorks Stacks가 Amazon EC2 인스턴스가 시작될 때까지 기다리고 있습니다.
3. booting - Amazon EC2 인스턴스가 부팅 중입니다.
4. running_setup - AWS OpsWorks Stacks 에이전트가 패키지 구성 및 설치와 같은 작업을 처리하는 계층의 Setup 레시피와 인스턴스에 앱을 배포하는 Deploy 레시피를 실행하고 있습니다.
5. online - 인스턴스를 사용할 준비가 되었습니다.

php-app1이 온라인 상태가 되면 [인스턴스] 페이지의 모양은 다음과 같아야 합니다.



이 페이지는 스택의 모든 인스턴스에 대한 간단한 요약으로 시작합니다. 이제 이 페이지에 온라인 인스턴스가 하나 표시됩니다. php-app1 [작업] 열에서 인스턴스를 중지하는 [중지]가 [시작] 및 [삭제]로 바뀝니다.

2.4단계: 앱 생성 및 배포 - Chef 11

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

MyStack을 보다 유용하게 만들려면 PHP 앱 서버 인스턴스에 앱을 배포해야 합니다. 앱의 코드 및 관련 파일은 리포지토리(예: Git)에 저장합니다. 이러한 파일을 애플리케이션 서버로 보내기 위해서는 두 단계가 필요합니다.

> **Note**
>
> 이 섹션에서 설명하는 절차는 Chef 11 스택에 적용됩니다. Chef 12 스택에서 계층에 앱을 추가하는 방법에 대한 자세한 정보는 앱 추가 단원을 참조하세요.

1. 앱을 생성합니다.

   앱에는 AWS OpsWorks 리포지토리에서 코드 및 관련 파일을 다운로드하는 데 필요한 정보가 포함되어 있습니다. 앱의 도메인과 같은 추가 정보를 지정할 수도 있습니다.

2. 애플리케이션 서버에 앱을 배포합니다.

   앱을 배포하면 AWS OpsWorks Stacks가 배포 수명 주기 이벤트를 트리거합니다. 그러면 에이전트가 파일을 적절한 디렉터리로 다운로드하고 서버 구성, 서비스 재시작 등의 관련 작업을 수행하는 인스턴스의 Deploy 레시피를 실행합니다.

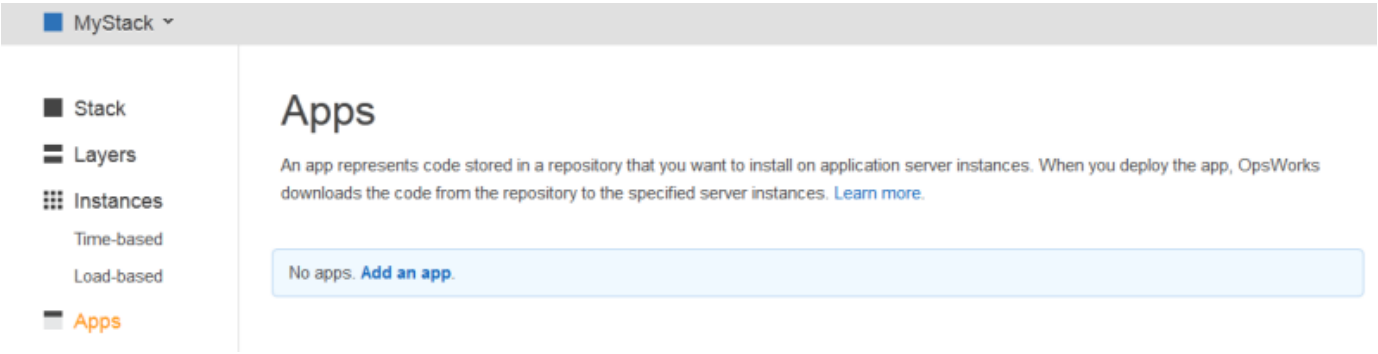
> **Note**
>
> 새 인스턴스를 생성하면 AWS OpsWorks Stacks가 인스턴스에 기존 앱을 자동으로 배포합니다. 하지만 새 앱을 생성하거나 기존 앱을 업데이트하는 경우에는 수동으로 모든 기존 인스턴스에 앱을 배포하거나 업데이트해야 합니다.

이 단계는 수동으로 예제 앱을 퍼블릭 Git 리포지토리에서 애플리케이션 서버로 배포하는 방법을 보여줍니다. 애플리케이션을 검사하려면 https://github.com/amazonwebservices/opsworks-demo-php-simple-app으로 이동하세요. 이 예제에서 사용되는 애플리케이션은 version1 브랜치에 있습니다. AWS OpsWorks Stacks는 다른 여러 리포지토리 유형도 지원합니다. 자세한 내용은 애플리케이션 소스 단원을 참조하십시오.

앱을 생성 및 배포하려면

1. 앱 페이지 열기

   탐색 창에서 [앱]을 클릭한 다음 [앱] 페이지에서 [앱 추가]를 클릭합니다.



2. 앱 구성

   [앱] 페이지에서 다음 값을 지정합니다.

   명칭

   AWS OpsWorks Stacks가 표시 목적으로 사용하는 앱의 이름입니다. 예제 앱의 이름은 입니다**`SimplePHPApp`**. AWS OpsWorks 스택은이 예제의 경우 짧은 이름인 단순phpapp도 생성합니다.이 이름은 나중에 설명하듯이 내부적으로 그리고 Deploy 레시피에 의해 사용됩니다.

   유형

   앱을 배포할 위치를 결정하는 앱 유형. 이 예제에서는 PHP 앱 서버 인스턴스에 앱을 배포하는 PHP를 사용합니다.

   데이터 소스 유형

   연결된 데이터베이스 서버. 지금은 [없음]을 선택합니다. 데이터베이스 서버에 대해서는 3단계; 백엔드 데이터 스토어 추가 단원에서 소개합니다.

   리포지토리 유형

   앱의 리포지토리 유형. 이 예제의 앱은 Git 리포지토리에 저장됩니다.

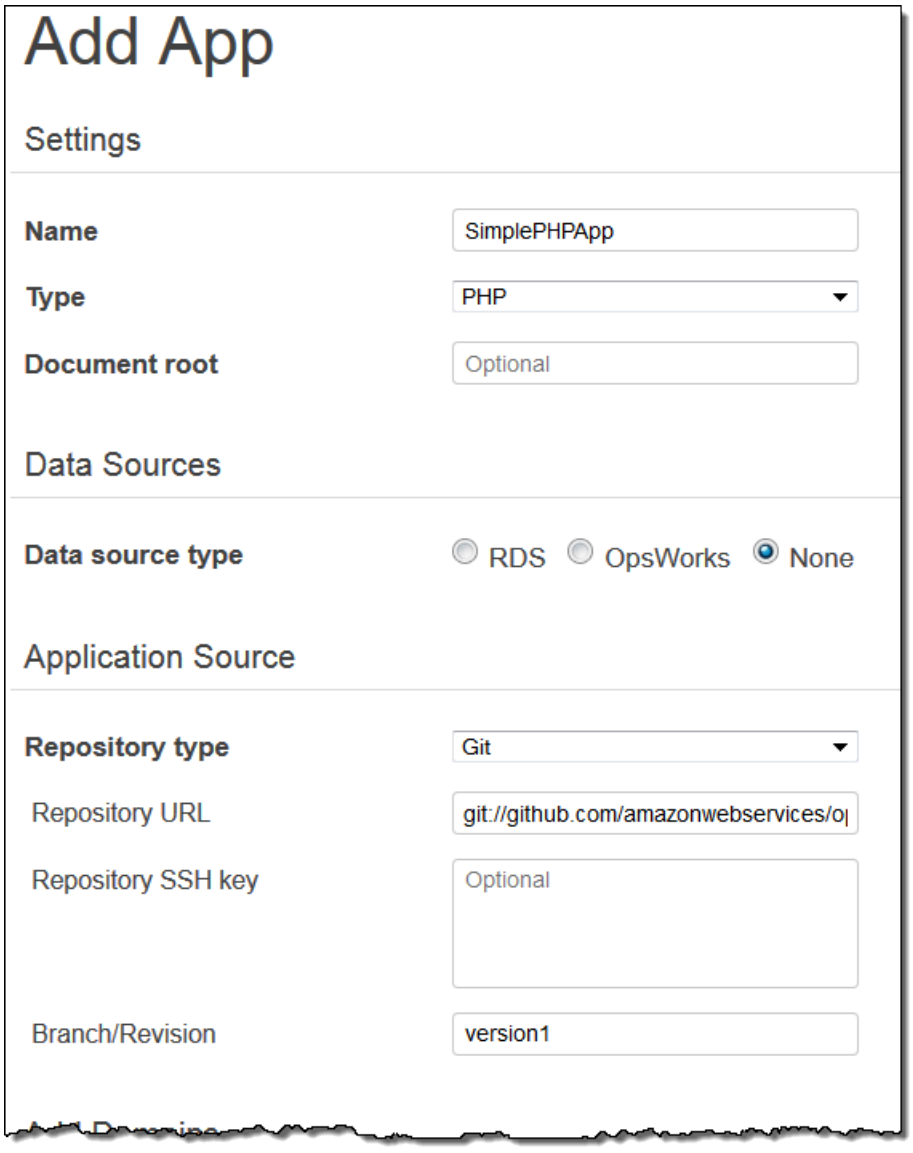
리포지토리 URL

앱의 리포지토리 URL. 예제 URL은 **`git://github.com/awslabs/opsworks-demo-php-simple-app.git`**입니다.

브랜치/개정

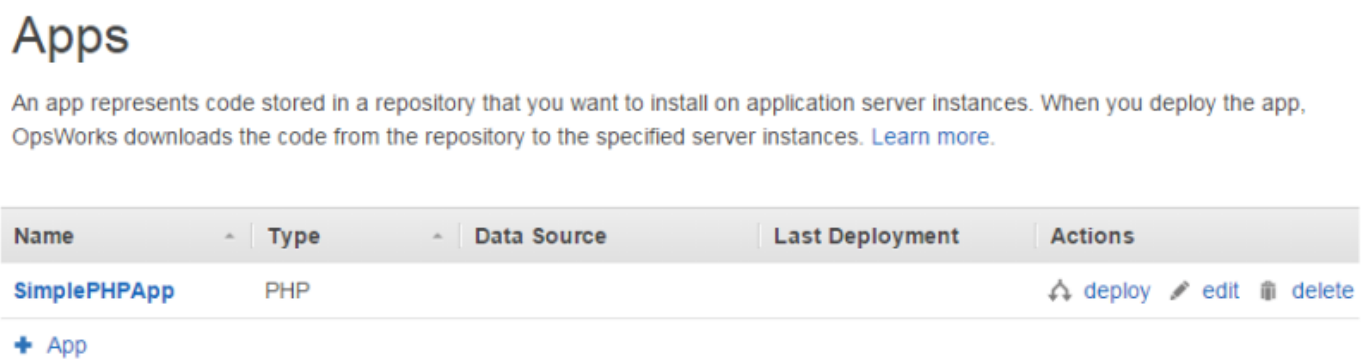
앱의 브랜치 또는 버전. 연습의 이 부분에서는 **`version1`** 브랜치를 사용합니다.

나머지 설정에 대해서는 기본값을 유지하고 [앱 추가]를 클릭합니다. 자세한 내용은 앱 추가 섹션을 참조하세요.



3. 배포 페이지 열기

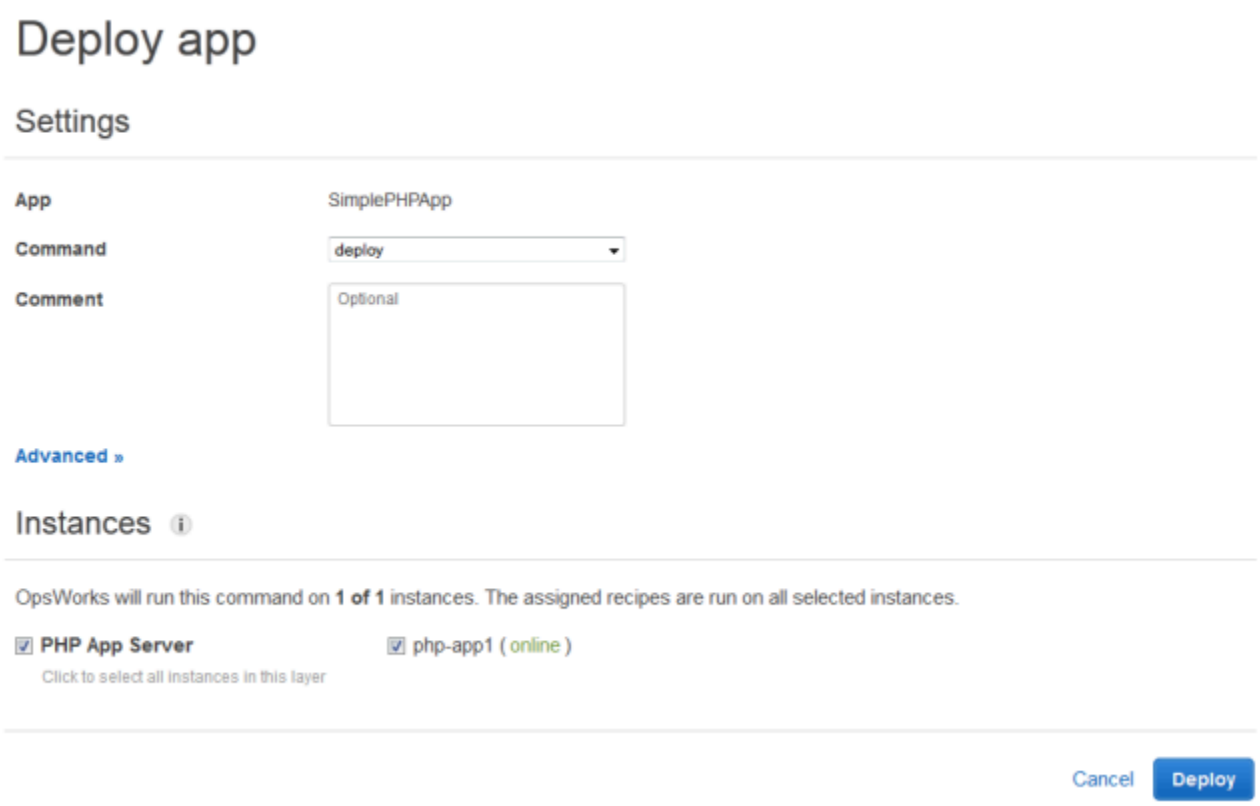
   서버에 코드를 설치하려면 앱을 배포해야 합니다. 이렇게 하려면 단순 PHPApp 작업 열에서 배포를 클릭합니다.



4. 앱 배포

   앱을 배포하면 에이전트는 애플리케이션을 다운로드하여 구성하는 Deploy 레시피를 PHP 앱 서버 인스턴스에서 실행합니다.

   [명령]은 이미 [배포]로 설정되어 있어야 합니다. 다른 설정은 기본값으로 유지하고 [배포]를 클릭하여 앱을 배포합니다.

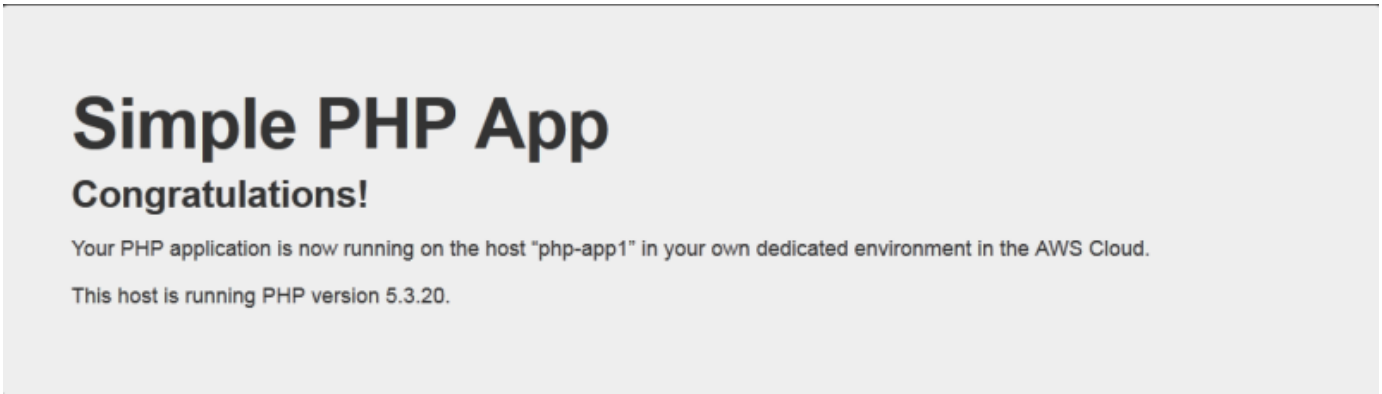


배포가 완료되면 배포 페이지에 성공 상태가 표시되고 php-app1 옆에 녹색 확인 표시가 나타납니다.

5. SimplePHPApp 실행

이제 SimplePHPApp이 설치되어 실행할 준비가 되었습니다. 이 앱을 실행하려면 탐색 창에서 [인스턴스]를 클릭하여 [인스턴스]로 이동합니다. 그런 다음 php-app1 인스턴스의 퍼블릭 IP 주소를 클릭합니다.

브라우저에 다음과 같은 페이지가 표시되어야 합니다.

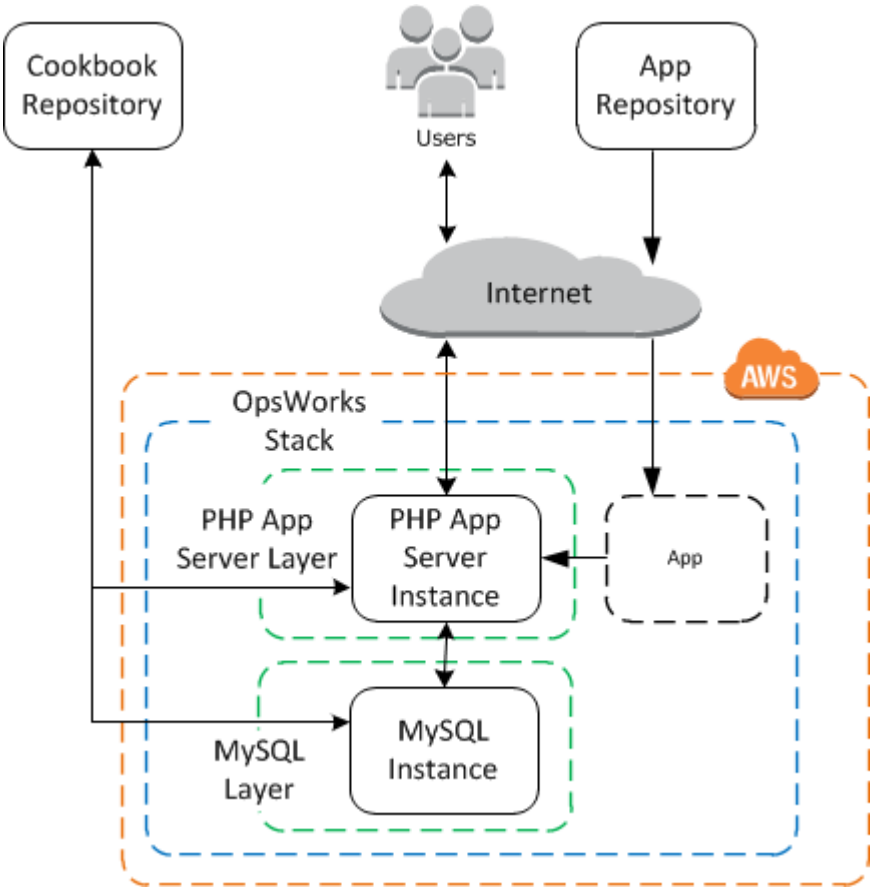


> **Note**
>
> 이 연습에서는 사용자가 다음 섹션으로 이동하여 최종적으로 단일 세션에서 전체 연습을 완료하는 것으로 가정합니다. 원하는 경우 언제든지 중지했다가 나중에 AWS OpsWorks Stacks에 로그인하고 스택을 열어 계속 진행할 수 있습니다. 단, 사용하는 AWS 리소스(예: 온라인 인스턴스)에 대해 요금이 부과됩니다. 불필요한 요금이 부과되지 않도록 인스턴스를 중지할 수 있습니다. 그러면 해당하는 EC2 인스턴스가 종료됩니다. 계속할 준비가 되었을 때 인스턴스를 다시 시작할 수 있습니다.

3단계; 백엔드 데이터 스토어 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

2.1 단계: 스택 생성 - Chef 11에서는 PHP 애플리케이션을 서비스하는 스택을 생성하는 방법을 설명했습니다. 하지만 이 애플리케이션을 일부 정적 텍스트를 표시하는 수준의 매우 간단한 애플리케이션이었습니다. 프로덕션 애플리케이션은 흔히 백엔드 데이터 스토어를 사용하며 다음과 비슷한 스택 구성을 보입니다.



이 섹션에서는 MyStack을 확장하여 백엔드 MySQL 데이터베이스 서버를 포함시키는 방법을 보여줍니다. 하지만 단순히 MySQL 서버를 스택에 추가하는 것보다는 많은 작업이 필요합니다. 또한 데이터베이스 서버와 제대로 통신하도록 앱을 구성해야 합니다. AWS OpsWorks Stacks는 이 작업을 대신 수행하지 않으므로 해당 작업을 처리하기 위해 일부 사용자 지정 레시피를 구현해야 합니다.

주제

- 3.1단계; 백엔드 데이터베이스 추가
- 3.2단계: SimplePHPApp 업데이트
- 간략한 차이: AWS OpsWorks 쿡북, 레시피 및 스택 속성
- 3.3단계: MyStack에 사용자 지정 쿡북 추가
- 3.4단계: 레시피 실행
- 3.5단계: SimplePHPApp 버전 2 배포
- 3.6단계: SimplePHPApp 실행

3.1단계; 백엔드 데이터베이스 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

새로운 버전의 SimplePHPApp은 백엔드 데이터베이스에 데이터를 저장합니다. AWS OpsWorks Stacks는 두 가지 유형의 데이터베이스 서버를 지원합니다.

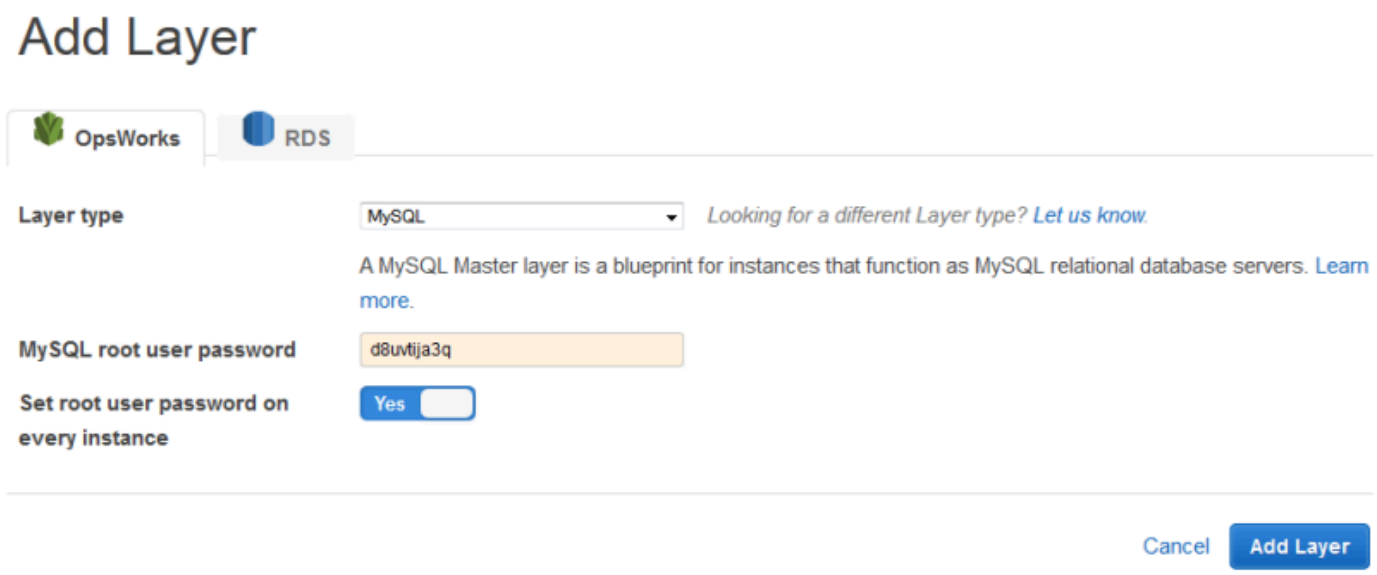
- MySQL AWS OpsWorks Stacks 계층은 MySQL 데이터베이스 마스터를 호스팅하는 Amazon EC2 인스턴스를 생성하기 위한 청사진입니다.
- Amazon RDS 서비스 계층은 Amazon RDS 인스턴스를 스택으로 통합하는 방법을 제공합니다.

Amazon DynamoDB와 같은 다른 데이터베이스를 사용할 수도 있으며, MongoDB와 같은 데이터베이스를 지원하는 사용자 지정 계층을 생성할 수도 있습니다. 자세한 내용은 the section called “백엔드 데이터 스토어 사용” 섹션을 참조하세요.

이 예제에서는 MySQL 계층을 사용합니다.

MyStack에 MySQL 계층을 추가하려면

1. [계층] 페이지에서 [+ 계층]을 클릭합니다.
2. [계층 추가] 페이지에서 [계층 유형]에 대해 [MySQL]을 선택하고, 기본 설정을 수락하고, [계층 추가]를 클릭합니다.



MySQL 계층에 인스턴스를 추가하려면

1. [계층] 페이지의 [MySQL] 행에서 [인스턴스 추가]를 클릭합니다.
2. [인스턴스] 페이지의 [MySQL]에서 [인스턴스 추가]를 클릭합니다.
3. 기본값을 수락하고 [인스턴스 추가]를 클릭하되 아직 시작하지는 마십시오.

> **Note**
>
> AWS OpsWorks Stacks는이 예제에서 앱의 짧은 이름인 simplephpapp을 사용하여 라는 데이터베이스를 자동으로 생성합니다. Chef 레시피를 사용하여 데이터베이스와 상호 작용하려면 이 이름이 필요합니다.

3.2단계: SimplePHPApp 업데이트

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

시작하려면 백엔드 데이터 스토어를 사용하는 새 버전의 SimplePHPApp이 필요합니다. AWS OpsWorks Stacks에서는 애플리케이션을 업데이트하기가 쉽습니다. Git 또는 하위 버전 리포지토리를 사용하는 경우 각 앱 버전마다 다른 리포지토리 브랜치를 지정할 수 있습니다. 예제 앱에서는 백엔드 데이터베이스를 사용하는 앱 버전을 Git 리포지토리의 version2 브랜치에 저장합니다. 앱 구성을 업데이트하여 새 브랜치를 지정하고 앱을 재배포하기만 하면 됩니다.

SimplePHPApp을 업데이트하려면

1. 이 앱의 편집 페이지를 엽니다.

   탐색 창에서 앱을 클릭한 다음 SimplePhpApp 행의 작업 열에서 편집을 클릭합니다.
2. 앱의 구성을 업데이트합니다.

   다음 설정을 변경합니다.

   브랜치/개정

   이 설정은 앱의 리포지토리 브랜치를 나타냅니다. SimplePHPApp의 첫 번째 버전은 데이터베이스에 연결하지 않습니다. 앱의 데이터베이스 지원 버전을 사용하려면 이 값을 **version2**로 설정합니다.

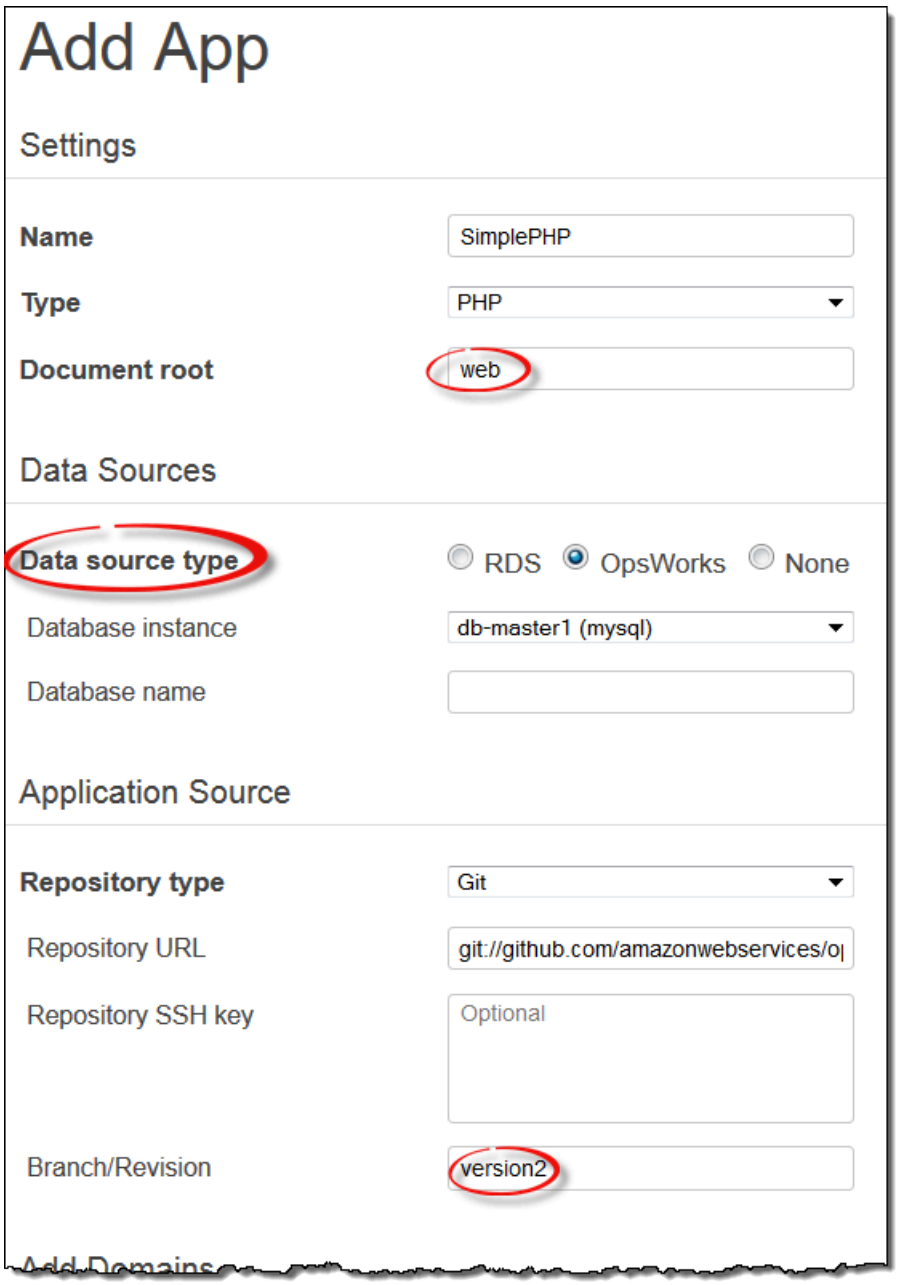
   [문서 루트]

   이 설정은 앱의 루트 폴더를 지정합니다. SimplePHPApp의 첫 번째 버전에서는 기본 설정을 사용했습니다. 즉, `index.php`를 서버의 표준 루트 폴더(PHP의 경우 `/srv/www`)에 설치했습니다. 여기서 이름만 지정하고 선행 '/'는 지정하지 않는 하위 폴더를 지정하면AWS OpsWorks 스택이 표준 폴더 경로에 추가합니다. SimplePHPApp의 버전 2는 `/srv/www/web`에 저장되어야 하므로 문서 루트를 **web**으로 설정합니다.

   데이터 원본 유형

   이 설정은 데이터베이스 서버를 앱과 연결합니다. 이 예제에서는 이전 단계에서 생성한 MySQL 인스턴스를 사용하므로 데이터 원본 유형을 OpsWorks로, 데이터베이스 인스턴스를 이전 단계에서 생성한 인스턴스인 db-master1(mysql)로 설정합니다. 데이터베이스 이름을 비워 둡니다. AWS OpsWorks Stacks는 앱의 짧은 이름인 simplephpapp을 사용하여 라는 이름의 서버에 데이터베이스를 생성합니다.

   [저장]을 클릭하여 새로운 구성을 저장합니다.



3. MySQL 인스턴스를 시작합니다.

앱을 업데이트하면 시작 시 AWS OpsWorks Stacks가 새 앱 서버 인스턴스에 새 앱 버전을 자동으로 배포합니다. 그러나 AWS OpsWorks Stacks는 기존 서버 인스턴스에 새 앱 버전을 자동으로 배포하지 않으므로에 설명된 대로 수동으로 배포해야 합니다2.4단계: 앱 생성 및 배포 - Chef 11. 이제 업데이트된 SimplePHPApp을 배포할 수 있지만, 이 예제에서는 약간 기다리는 것이 좋습니다.

간략한 차이: AWS OpsWorks 쿡북, 레시피 및 스택 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제 앱 및 데이터베이스 서버는 생성되었지만 사용할 준비가 되려면 아직 많은 작업이 필요합니다. 여전히 데이터베이스를 설정하고 앱의 연결 설정을 구성해야 합니다. AWS OpsWorks Stacks는 이러한 작업을 자동으로 처리하지 않지만 Chef 쿡북, 레시피 및 동적 속성을 지원합니다. 한 쌍의 레시피, 즉 데이터베이스를 설정하는 레시피와 앱의 연결 설정을 구성하는 레시피를 구현하고 Stacks에서 자동으로 실행하도록 AWS OpsWorks 할 수 있습니다.

필요한 레시피가 포함된 phpapp 쿡북이 이미 구현되어 사용 가능한 상태입니다. 원할 경우 3.3단계: MyStack에 사용자 지정 쿡북 추가 섹션으 단원으로 건너뛸 수 있습니다. 자세히 알아보기를 원할 경우 이 섹션에서 쿡북 및 레시피에 대한 배경 지식을 제공하고 레시피가 작동하는 방식을 설명합니다. 쿡북 자체에 대해 알아보려면 phpapp cookbook 단원을 참조하세요.

주제

- 레시피와 속성
- 데이터베이스 설정
- 데이터베이스에 애플리케이션 연결

레시피와 속성

Chef 레시피는 기본적으로 패키지 설치, 구성 파일 생성, shell 명령 실행 등 인스턴스에 대한 작업을 수행하는 특수한 Ruby 애플리케이션입니다. 관련된 레시피의 그룹은 쿡북으로 구성됩니다. 여기에는 구성 파일을 생성하기 위한 템플릿과 같은 지원 파일도 포함됩니다.

AWS OpsWorks Stacks에는 내장 계층을 지원하는 쿡북 세트가 있습니다. 인스턴스에 대한 사용자 지정 작업을 수행하기 위해 자체 레시피로 구성된 사용자 지정 쿡북을 생성할 수도 있습니다. 이 주제에서는 레시피를 간략히 소개하고 레시피를 사용하여 데이터베이스를 설정하고 앱의 연결 설정을 구성하는 방법을 살펴봅니다. 쿡북 및 레시피에 대한 자세한 정보는 쿡북과 레시피 또는 AWS OpsWorks 스택 사용자 지정 단원을 참조하세요.

레시피는 일반적으로 Chef 속성을 입력 데이터로 사용합니다.

- 일부 속성은 Chef에 의해 정의되며 운영 체제와 같이 인스턴스에 대한 기본 정보를 제공합니다.
- AWS OpsWorks Stacks는 계층 구성과 같은 스택과 앱 리포지토리와 같은 배포된 앱에 대한 정보를 포함하는 속성 세트를 정의합니다.

  사용자 지정 JSON을 스택 또는 배포에 할당하여 여기에 사용자 지정 속성을 추가할 수 있습니다.
- 또한 쿡북에서 쿡북 고유 속성을 정의할 수 있습니다.

  phpapp 쿡북 속성은 `attributes/default.rb`에서 정의됩니다.

AWS OpsWorks Stacks 속성의 전체 목록은 스택 구성 및 배포 속성: Linux 및 섹션을 참조하세요내장 쿡북 속성. 자세한 내용은 속성 재정의 단원을 참조하십시오.

속성은 계층 구조로 구성되며, JSON 객체로 표현될 수 있습니다.

다음과 같이 Chef 노드 구문을 사용하여 이 데이터를 애플리케이션에 통합합니다.

```
[:deploy][:simplephpapp][:database][:username]
```

`deploy` 노드에는 앱의 데이터베이스, Git 리포지토리 등에 대한 정보를 포함하는 단일 앱 노드 `simplephpapp`이 있습니다. 예제는 데이터베이스 사용자 이름의 값을 나타냅니다. 이 값은 `root`로 확인됩니다.

데이터베이스 설정

MySQL 계층의 내장 설정 레시피는 앱의 짧은 이름으로 명명된 데이터베이스를 자동으로 생성합니다. 그러므로 이 예제를 위해 simplephpapp이라는 데이터베이스가 이미 생성되어 있습니다. 하지만 앱이 데이터를 저장할 테이블을 생성하여 설정을 마쳐야 합니다. 테이블을 수동으로 생성할 수 있지만 더 좋은 방법은 작업을 처리하기 위해 사용자 지정 레시피를 구현하고 AWS OpsWorks Stacks에서 실행하도록 하는 것입니다. 이 섹션에서는 어떻게 레시피 `dbsetup.rb`가 구현되는지 설명합니다. Stacks가 레시피를 실행하도록 AWS OpsWorks 하는 절차는 나중에 설명합니다.

리포지토리의 레시피를 보려면 db설정.rb를 참조하세요. 다음 예제는 `dbsetup.rb` 코드를 보여줍니다.

`execute`는 지정된 명령을 실행하는 Chef 리소스입니다. 여기서는 테이블을 생성하는 MySQL 명령입니다. `not_if` 명령은 지정된 테이블이 이미 존재할 경우 명령이 실행되지 않게 합니다. Chef 리소스에 대한 자세한 정보는 About Resources and Providers를 참조하세요.

레시피는 앞서 설명한 노드 구문을 사용하여 속성 값을 명령 문자열에 삽입합니다. 예를 들어 다음 레시피는 데이터베이스의 사용자 이름을 삽입합니다.

```
#{deploy[:database][:username]}
```

약간은 퍼즐 같은 이 코드를 풀어 보겠습니다.

- 각 반복에서, deploy는 현재 앱 노드로 설정됩니다. 따라서 [:deploy][:*app_name*]으로 확인됩니다. 이 예제에서는 [:deploy][:simplephpapp]으로 확인됩니다.
- 앞서 제시한 배포 속성 값을 사용하여 전체 노드가 root로 확인됩니다.
- 노드를 #{ }로 묶어 문자열에 삽입합니다.

다른 노드도 대부분 비슷한 방식으로 확인됩니다. #{node[:phpapp][:dbtable]}은 예외입니다. 이 노드는 사용자 지정 쿡북의 속성 파일에 의해 정의되며 테이블 이름 urler로 확인됩니다. 따라서 MySQL 인스턴스에 실행되는 실제 명령은 다음과 같습니다.

```
"/usr/bin/mysql
    -uroot
    -pvjud1hw5v8
    simplephpapp
    -e'CREATE TABLE urler(
       id INT UNSIGNED NOT NULL AUTO_INCREMENT,
       author VARCHAR(63) NOT NULL,
       message TEXT,
       PRIMARY KEY (id))'
"
```

이 명령은 배포 속성의 자격 증명 및 데이터베이스 이름을 사용하여 id, author 및 message 필드를 포함하는 테이블 urler을 생성합니다.

데이터베이스에 애플리케이션 연결

퍼즐의 두 번째 부분은 테이블에 액세스하기 위해 데이터베이스 암호 같은 연결 정보가 필요한 애플리케이션입니다. SimplePHPApp은 작업 파일이 app.php 하나뿐입니다. index.php가 수행하는 작업은 app.php를 로드하는 것 뿐입니다.

app.php에는 데이터베이스 연결을 처리하는 db-connect.php가 포함됩니다. 하지만 이 파일은 리포지토리에 없습니다. db-connect.php를 미리 생성할 수는 없습니다. 이 파일은 특정 인스턴스를

기반으로 데이터베이스를 정의하기 때문입니다. 그 대신, appsetup.rb 레시피가 배포 속성의 연결 데이터를 사용하여 db-connect.php를 생성합니다.

리포지토리의 레시피를 보려면 app설정.rb를 참조하세요. 다음 예제는 appsetup.rb 코드를 보여줍니다.

dbsetup.rb처럼, appsetup.rb는 deploy 노드에 있는 앱을 반복합니다. 다시 말하지만 단순한 phappapp일 뿐입니다. 이 레시피는 script 리소스 및 template 리소스를 포함하는 코드 블록을 실행합니다.

이 script 리소스는 PHP 애플리케이션용 종속성 관리자인 Composer를 설치합니다. 그런 다음 Composer의 install 명령을 실행하여 샘플 애플리케이션의 종속 파일을 앱의 루트 디렉터리에 설치합니다.

template 리소스는 db-connect.php를 생성하여 /srv/www/simplephpapp/current에 저장합니다. 다음 사항에 유의하세요.

- 이 레시피는 조건문을 사용하여 인스턴스의 운영 체제에 따라 달라지는 파일 소유자를 지정합니다.
- only_if 명령은 Chef에게 지정된 디렉터리가 존재하는 경우에만 템플릿을 생성하도록 지시합니다.

template 리소스는 연결된 파일과 기본적으로 콘텐츠 및 구조가 동일한 템플릿에서 작동하지만 다양한 데이터 값을 위한 자리 표시자를 포함합니다. source 파라미터는 phpapp 쿡북의 db-connect.php.erb 디렉터리에 위치하는 템플릿 templates/default를 지정하며 다음을 포함합니다.

Chef가 템플릿을 처리할 때 <%= => 자리 표시자를 템플릿 리소스 해당 변수의 값으로 바꿉니다. 이들 값은 배포 속성에서 가져온 것입니다. 생성된 파일은 다음과 같습니다.

3.3단계: MyStack에 사용자 지정 쿡북 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 쿡북은 리포지토리에 앱과 거의 마찬가지로 저장합니다. 각 스택은 일련의 사용자 지정 쿡북으로 구성된 리포지토리를 가질 수 있습니다. 그런 다음 AWS OpsWorks Stacks에 스택의 인스턴스에 사용자 지정 쿡북을 설치하도록 지시합니다.

1. 탐색 창에서 [스택]을 클릭하여 현재 스택에 대한 페이지를 표시합니다.
2. [스택 설정]을 클릭하고 [편집]을 클릭합니다.
3. 스택 구성을 다음과 같이 수정합니다.
   - 사용자 지정 Chef 쿡북 사용 – 예
   - Repository type - Git
   - 리포지토리 URL – **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**
4. [저장]을 클릭하여 스택 구성을 업데이트합니다.

AWS OpsWorks 그런 다음 Stacks는 스택의 모든 인스턴스에 쿡북 리포지토리의 콘텐츠를 설치합니다. 새 인스턴스를 생성하면 AWS OpsWorks Stacks가 쿡북 리포지토리를 자동으로 설치합니다.

> **Note**
>
> 쿡북을 업데이트하거나 리포지토리에 새 쿡북을 추가해야 하는 경우 스택 설정을 터치하지 않고도 쿡북을 추가할 수 있습니다. AWS OpsWorks Stacks는 업데이트된 쿡북을 모든 새 인스턴스에 자동으로 설치합니다. 그러나 AWS OpsWorks Stacks는 스택의 온라인 인스턴스에 업데이트된 쿡북을 자동으로 설치하지 않습니다. 스택 명령을 실행하여 AWS OpsWorks 쿡북을 업데이트하도록 `Update Cookbooks` Stacks에 명시적으로 지시해야 합니다. 자세한 내용은 스택 명령 실행 단원을 참조하십시오.

3.4단계: 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 쿡북이 준비되었으면 적절한 인스턴스에서 레시피를 실행해야 합니다. 레시피를 수동으로 실행할 수 있습니다. 하지만 일반적으로 레시피는 인스턴스의 수명 주기에서 인스턴스 부팅 후 또는 앱 배포 시와 같은 예측 가능한 시점에 실행해야 합니다. 이 섹션에서는 훨씬 더 간단한 접근 방식을 설명합니다. AWS OpsWorks Stacks가 적절한 시간에 자동으로 실행하도록 합니다.

AWS OpsWorks Stacks는 레시피 실행을 간소화하는 수명 주기 이벤트 세트를 지원합니다. 예를 들어 설정 이벤트는 인스턴스 부팅이 완료된 후 발생하고, Deploy 이벤트는 앱을 배포할 때 발생합니다. 각 계층에는 각 수명 주기 이벤트에 연결된 내장 레시피 세트가 있습니다. 인스턴스에서 수명 주기 이벤트가 발생하면 에이전트가 인스턴스의 각 계층에서 연결된 레시피를 실행합니다. Stacks가 사용자 지정 레시피를 자동으로 실행하도록 AWS OpsWorks 하려면 해당 계층의 적절한 수명 주기 이벤트에 추가하면 기본 제공 레시피가 완료된 후 에이전트가 레시피를 실행합니다.

이 예제에서는 두 개의 레시피를 실행해야 합니다. 즉 MySQL 인스턴스에서 `dbsetup.rb`, PHP 앱 서버 인스턴스에서 `appsetup.rb`를 실행합니다.

> **Note**
>
> 콘솔에서 *`cookbook_name`*::*`recipe_name`* 형식을 사용하여 레시피를 지정합니다. 여기서 *`recipe_name`*에는 .rb 확장자가 포함되지 않습니다. 예를 들어 `dbsetup.rb`를 **`phpapp::dbsetup`**으로 참조합니다.

수명 주기 이벤트에 사용자 지정 레시피를 할당하려면

1. 계층 페이지에서 MySQL에 대해 레시피를 클릭하고 편집을 클릭합니다.
2. 사용자 지정 Chef 레시피 섹션에서 배포에 **`phpapp::dbsetup`**를 입력합니다.

3. [+] 아이콘을 클릭하여 이벤트에 레시피를 할당하고 [저장]을 클릭하여 새 계층 구성을 저장합니다.
4. 계층 페이지로 돌아가 해당 절차를 반복하여 **phpapp::appsetup**을 PHP 앱 서버 계층의 배포 이벤트에 할당합니다.

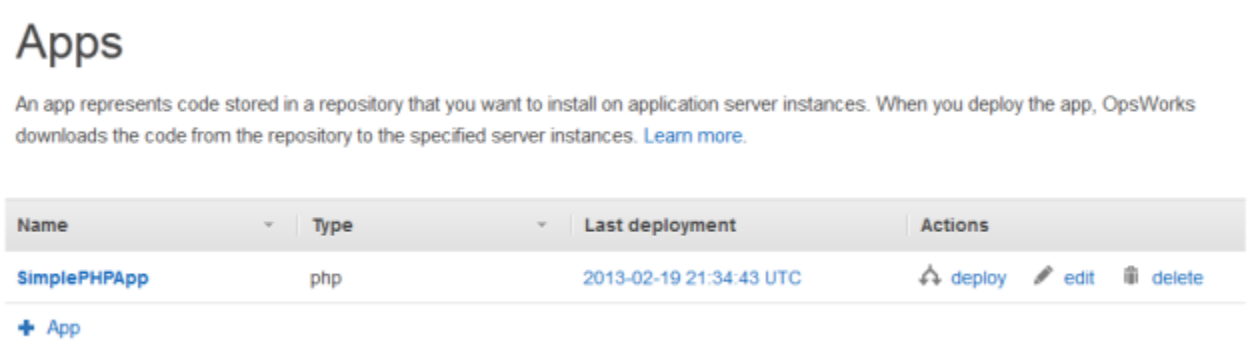
3.5단계: SimplePHPApp 버전 2 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

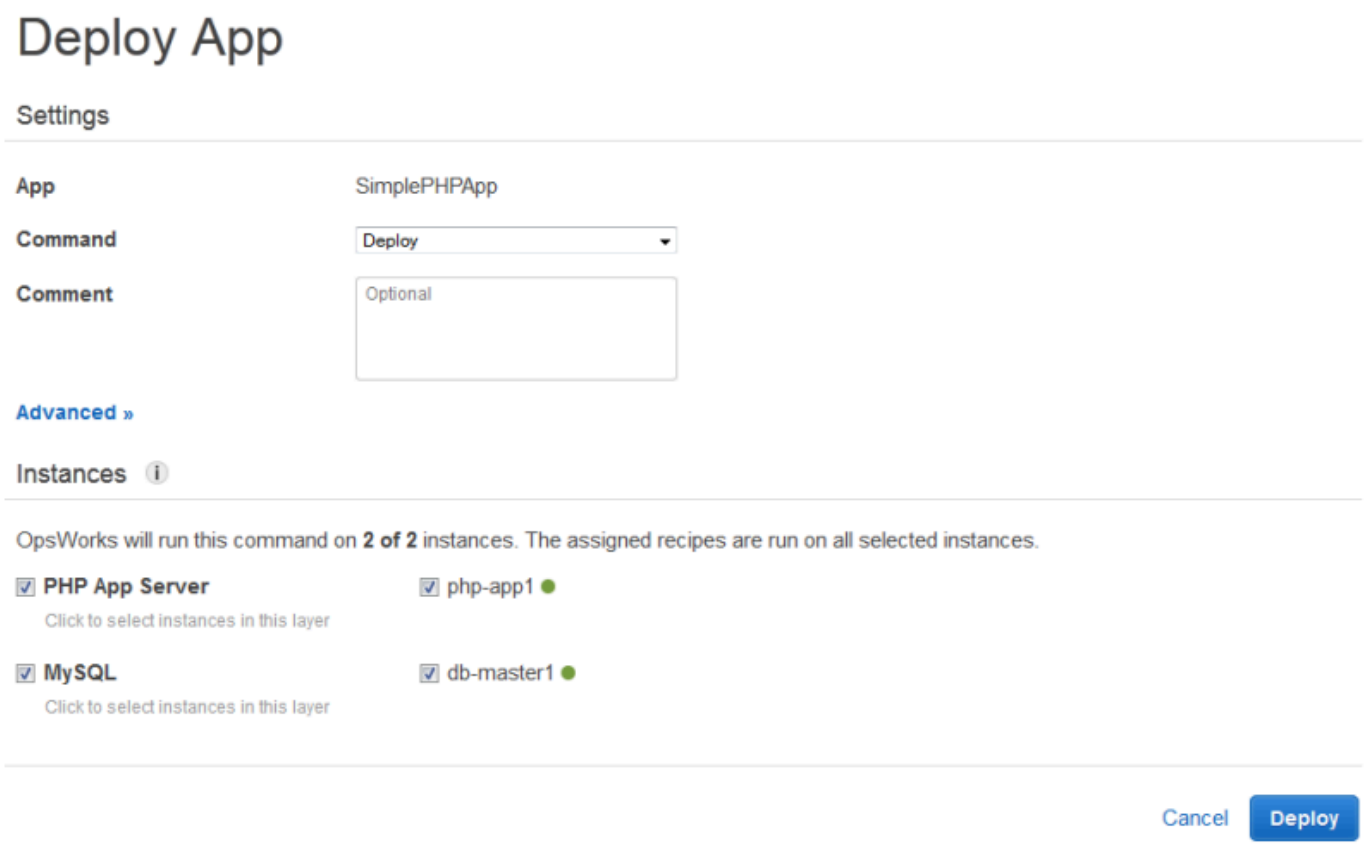
마지막 단계는 새 버전의 SimplePHPApp을 배포하는 것입니다.

SimplePHPApp을 배포하려면

1. 앱 페이지의 SimplePHPApp 앱의 작업에서 배포를 클릭합니다.



2. 기본값을 수락하고 [배포]를 클릭합니다.



앱 배포 페이지에서 배포를 클릭하면 배포 수명 주기 이벤트가 트리거되어 에이전트에게 배포 레시피를 실행하도록 알립니다. 기본적으로 스택의 모든 인스턴스에서 이 이벤트를 트리거합니다. 내장 Deploy 레시피는 앱 유형에 적절한 인스턴스(이 경우에는 PHP 앱 서버 인스턴스)에만 앱을 배포합니다. 그러나 일반적으로 다른 인스턴스에서 Deploy 이벤트를 트리거하여 앱 배포에 응답하도록 하는 데에도 유용합니다. 이 경우에는 MySQL 인스턴스에서도 Deploy를 트리거하여 데이터베이스를 설정하려고 할 수도 있습니다.

다음 사항에 유의하세요.

- PHP 앱 서버 인스턴스의 에이전트는 계층의 내장 레시피를 실행하고 뒤이어 앱의 데이터베이스 연결을 구성하는 `appsetup.rb`를 실행합니다.
- MySQL 인스턴스의 에이전트는 어떠한 것도 설치하지 않지만 `dbsetup.rb`를 실행하여 urler 테이블을 생성합니다.

배포가 완료되면 배포 페이지의 상태가 성공으로 변경됩니다.

3.6단계: SimplePHPApp 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

배포 상태가 [성공]으로 전환되면 다음과 같이 새 SimplePHPApp 버전을 실행할 수 있습니다.

SimplePHPApp을 실행하려면

1. [인스턴스] 페이지의 [php-app1] 행에서 퍼블릭 IP 주소를 클릭합니다.

   브라우저에 다음 페이지가 표시되어야 합니다.

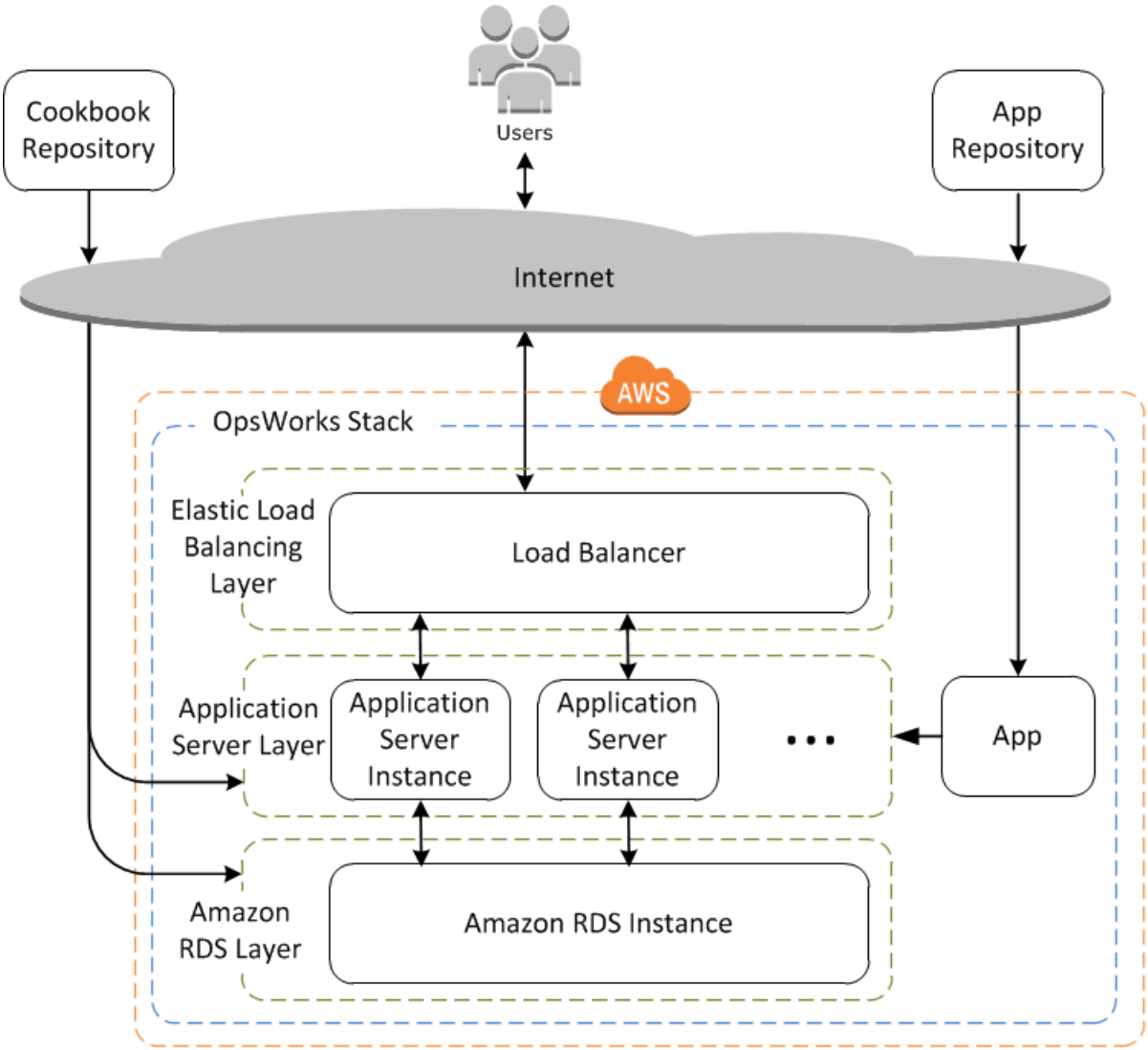
2. 생각 공유하기를 클릭하고 생각에는 **`Hello world!`**을 입력하고 이름에는 이름을 입력하세요. 그런 다음 [의견 제출]을 클릭하여 데이터베이스에 메시지를 추가합니다.

3. [뒤로 이동]을 클릭하여 데이터베이스의 메시지를 모두 확인합니다.

4단계: MyStack 확장

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

현재 MyStack은 애플리케이션 서버가 하나뿐입니다. 프로덕션 스택에서는 수신 트래픽을 처리하기 위한 애플리케이션 서버 여러 개와 수신 트래픽을 애플리케이션 서버 간에 균일하게 분산시킬 로드 밸런서가 필요할 수 있습니다. 아키텍처는 다음과 같습니다.



AWS OpsWorks 스택을 사용하면 스택을 쉽게 확장할 수 있습니다. 이 섹션에서는 MyStack에 두 번째 24/7 PHP 앱 서버 인스턴스를 추가하고 두 인스턴스 모두 Elastic Load Balancing 로드 밸런서 뒤에 배치하여 스택을 확장하는 기본적인 방법을 설명합니다. 절차를 쉽게 확장하여 임의의 수의 24/7 인스턴스를 추가하거나 시간 기반 또는 로드 기반 인스턴스를 사용하여 AWS OpsWorks Stacks가 스택을 자동으로 확장하도록 할 수 있습니다. 자세한 내용은 시간 기반 또는 로드 기반 인스턴스를 사용하여 로드 관리 단원을 참조하십시오.

4.1단계: 로드 밸런서 추가

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Elastic Load Balancing은 수신 애플리케이션 트래픽을 여러 Amazon EC2 인스턴스에 자동으로 분산하는 AWS 서비스입니다. 트래픽을 분산하는 것에 더하여 Elastic Load Balancing은 다음 기능을 수행합니다.

- 비정상 Amazon EC2 인스턴스를 탐지합니다.

  또한 비정상 인스턴스가 복원될 때까지 트래픽을 나머지 정상 인스턴스로 다시 라우팅합니다.
- 수신 트래픽에 맞춰 요청 처리 용량을 자동으로 조정합니다.

Note

로드 밸런서는 두 가지 목적을 충족할 수 있습니다. 명백한 목적은 애플리케이션 서버 간에 로드를 평활화하는 것입니다. 또한 대부분의 사이트에서는 애플리케이션 서버 및 데이터베이스를 직접 사용자 액세스로부터 격리하기를 선호합니다. AWS OpsWorks Stacks를 사용하면 다음과 같이 퍼블릭 및 프라이빗 서브넷이 있는 Virtual Private Cloud(VPC)에서 스택을 실행하여 이 작업을 수행할 수 있습니다.

- 애플리케이션 서버 및 데이터베이스를 프라이빗 서브넷에 배치합니다. 그러면 VPC 내 다른 인스턴스는 이들에 액세스할 수 있지만 사용자는 액세스할 수 없습니다.
- 사용자 트래픽을 퍼블릭 서브넷의 로드 밸런서로 보냅니다. 그러면 로드 밸런서가 프라이빗 서브넷의 애플리케이션 서버로 트래픽을 전달하고 응답을 사용자에게 반환합니다.

자세한 내용은 VPC에서 스택 실행 단원을 참조하십시오. 이 연습의 예제를 확장하여 VPC에서 실행하는 AWS CloudFormation 템플릿의 경우 `OpsWorksVPCtemplates.zip` 파일을 다운로드합니다.

Elastic Load Balancing은 종종 계층으로 불리지만 다른 내장 계층과는 약간 다르게 작동합니다. 계층을 생성하고 여기에 인스턴스를 추가하는 대신 Amazon EC2 콘솔을 사용하여 Elastic Load Balancing 로드 밸런서를 생성한 다음 기존 계층 중 하나, 일반적으로 애플리케이션 서버 계층에 연결합니다. 그런 다음 AWS OpsWorks Stacks는 계층의 기존 인스턴스를 서비스에 등록하고 새 인스턴스를 자동으로 추가합니다. 다음 절차에서는 로드 밸런서를 MyStack의 PHP 앱 서버 계층에 추가하는 방법을 설명합니다.

Note

AWS OpsWorks Stacks는 Application Load Balancer를 지원하지 않습니다. Classic Load Balancer는 AWS OpsWorks Stacks에서만 사용할 수 있습니다.

PHP 앱 서버 계층에 로드 밸런서를 연결하려면

1. Amazon EC2 콘솔을 사용하여 MyStack의 새 로드 밸런서를 생성합니다. 사용자 계정에서 EC2 Classic을 지원하는지 여부에 따라 세부 정보가 달라집니다. 자세한 내용은 Elastic Load Balancing 시작하기를 참조하세요. [로드 밸런서 생성] 마법사를 실행하면 다음과 같이 로드 밸런서를 구성합니다.

   로드 밸런서 정의

   Stacks 콘솔에서 쉽게 찾을 수 있도록 로드 AWS OpsWorks 밸런서에 PHP-LB와 같이 쉽게 알아볼 수 있는 이름을 할당합니다. [계속]을 선택하여 나머지 설정에 대해서는 기본값을 수락합니다.

   [LB 내부 생성] 메뉴에서 서브넷이 하나 이상 있는 VPC를 선택한 경우 로드 밸런서를 통해 트래픽을 라우트하려는 각 가용 영역의 서브넷을 선택해야 합니다.

   보안 그룹 할당

   사용자 계정에서 기본 VPC를 지원하는 경우 마법사에 이 페이지가 표시되어 로드 밸런서의 보안 그룹을 확인합니다. EC2 Classic의 경우에는 이 페이지가 표시되지 않습니다.

이 연습에서는 [기본 VPC 보안 그룹]을 선택합니다.

보안 설정을 구성합니다

로드 밸런서 정의 페이지에서 로드 밸런서 프로토콜로 HTTPS를 선택한 경우 이 페이지에서 인증서, 암호 및 SSL 프로토콜 설정을 구성하세요. 이 연습에서는 기본값을 수락하고 [상태 검사 구성]을 선택합니다.

상태 확인 구성

ping 경로를 **/**로 설정하고 나머지 설정에 대해 기본값을 수락합니다.

EC2 인스턴스 추가

계속을 선택합니다. AWS OpsWorks 스택은 로드 밸런서에 인스턴스를 자동으로 등록합니다.

태그 추가

찾은 도움말에 태그를 추가합니다. 각 태그는 키-값 페어입니다. 예를 들어 이 연습에서는 **Description**을 키로 **Test LB**를 값으로 지정할 수 있습니다.

검토

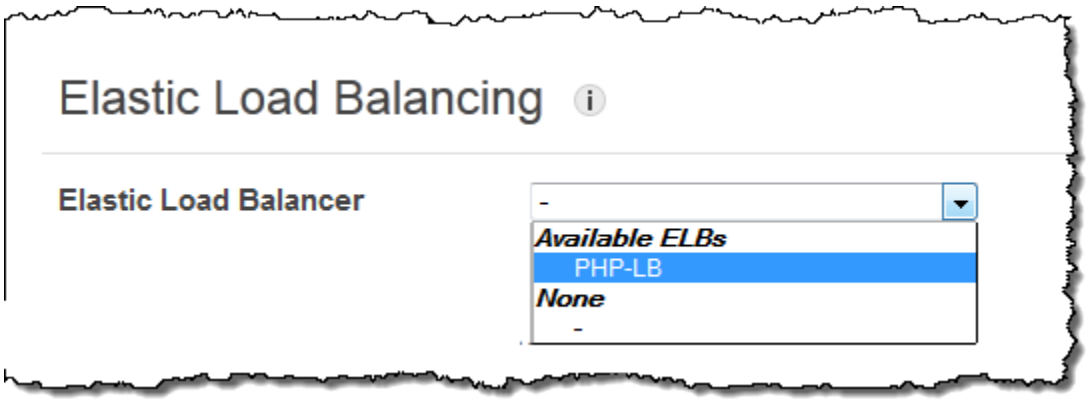
선택 항목을 검토하고 [만들기]를 선택한 다음 [닫기]를 선택합니다. 그러면 로드 밸런서가 시작됩니다.

2. 계정이 기본 VPC를 지원하는 경우 로드 밸런서를 시작한 후 로드 밸런서의 보안 그룹에 적절한 수신 규칙이 있는지 확인해야 합니다. 기본 규칙은 어떠한 인바운드 트래픽도 허용하지 않습니다.

   1. Amazon EC2 탐색 창에서 보안 그룹을 선택합니다.
   2. [기본 VPC 보안 그룹]을 선택합니다.
   3. 인바운드 탭에서 편집을 선택합니다.
   4. 이 연습에서는 Source를 Anywhere로 설정합니다. 그러면 로드 밸런서에 모든 IP 주소에서의 수신 트래픽을 허용하도록 지시합니다.

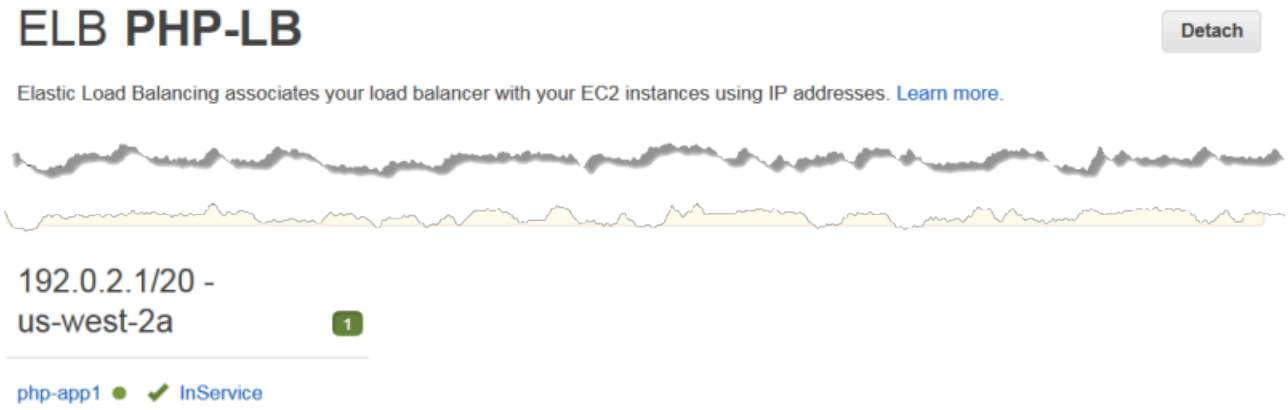
3. AWS OpsWorks Stacks 콘솔로 돌아갑니다. [계층] 페이지에서 계층의 [네트워크] 링크를 클릭한 다음 [편집]을 클릭합니다.

4. [Elastic Load Balancing]에서 1단계에서 생성한 로드 밸런서를 선택한 다음 [저장]을 선택합니다.



로드 밸런서를 계층에 연결한 후 AWS OpsWorks Stacks는 계층의 현재 인스턴스를 자동으로 등록하고 온라인 상태가 되면 새 인스턴스를 추가합니다.

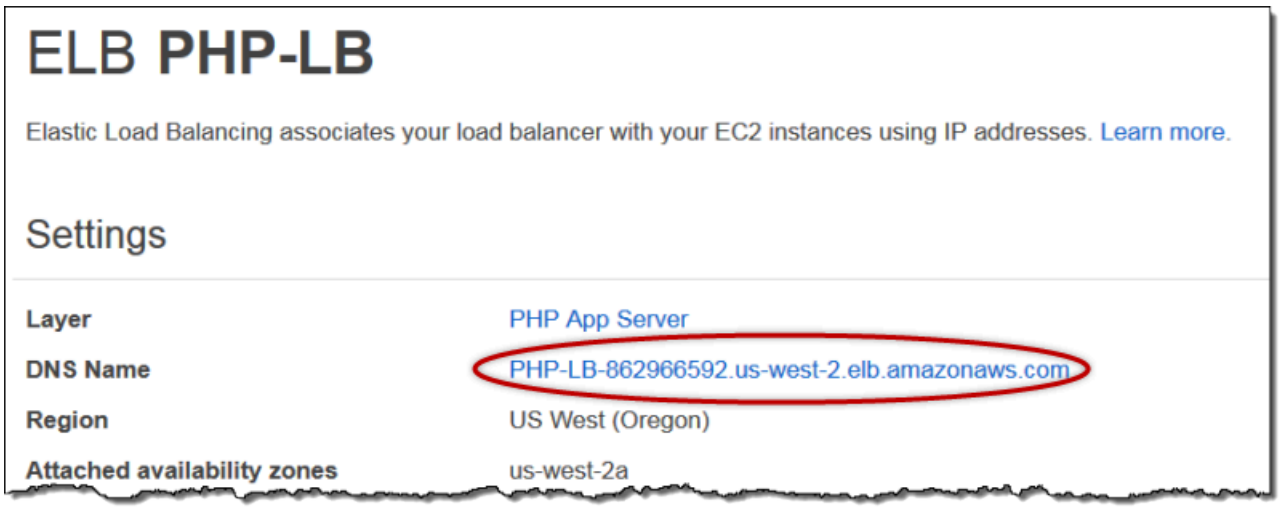
5. [계층] 페이지에서 로드 밸런서의 이름을 클릭하여 로드 밸런서의 세부정보 페이지를 엽니다. 등록이 완료되고 인스턴스가 상태 확인을 통과하면 AWS OpsWorks Stacks는 로드 밸런서 페이지의 인스턴스 옆에 녹색 확인 표시를 표시합니다.



이제 로드 밸런서로 요청을 전송하여 SimplePHPApp을 실행할 수 있습니다.

로드 밸런서를 통해 SimplePHPApp을 실행하려면

1. 아직 열지 않은 경우 로드 밸런서의 세부 정보 페이지를 엽니다.
2. 속성 페이지에서 인스턴스의 상태 확인 상태를 확인하고 로드 밸런서의 DNS 이름을 클릭하여 SimplePHPApp을 실행합니다. 로드 밸런서는 PHP 앱 서버 인스턴스에 요청을 전달하고 응답을 반환합니다. 이 응답은 PHP 앱 서버 인스턴스의 퍼블릭 IP 주소를 클릭했을 때 얻는 응답과 정확히 동일해야 합니다.



> **Note**
>
> AWS OpsWorks 또한 Stacks는 HAProxy 로드 밸런서를 지원하므로 일부 애플리케이션에는 이점이 있을 수 있습니다. 자세한 내용은 HAProxy AWS OpsWorks 스택 계층 단원을 참조하십시오.

4.2단계: PHP 앱 서버 인스턴스 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이재 로드 밸런서가 설치되었으므로 PHP 앱 서버 계층에 더 많은 인스턴스를 추가하여 스택을 확장할 수 있습니다. 사용자 관점에서 작업은 심리스합니다. 새 PHP 앱 서버 인스턴스가 온라인 상태가 될 때마다 AWS OpsWorks Stacks는 자동으로 이를 로드 밸런서에 등록하고 SimplePHPApp을 배포하므로 서버가 수신 트래픽 처리를 즉시 시작할 수 있습니다. 간략한 설명을 위해 이 항목에PHP 앱 서버서는 인스턴스 하나를 추가하는 방법을 설명하지만, 동일한 접근 방식을 사용하여 원하는 만큼 인스턴스를 추가할 수 있습니다.

PHP 앱 서버 계층에 다른 인스턴스를 추가하려면

1. 인스턴스 페이지의 PHP 앱 서버에서 + 인스턴스를 클릭합니다.
2. 기본 설정을 수락하고 [인스턴스 추가]를 클릭합니다.
3. [시작]을 클릭하여 인스턴스를 시작합니다.

4.3단계: MyStack 모니터링

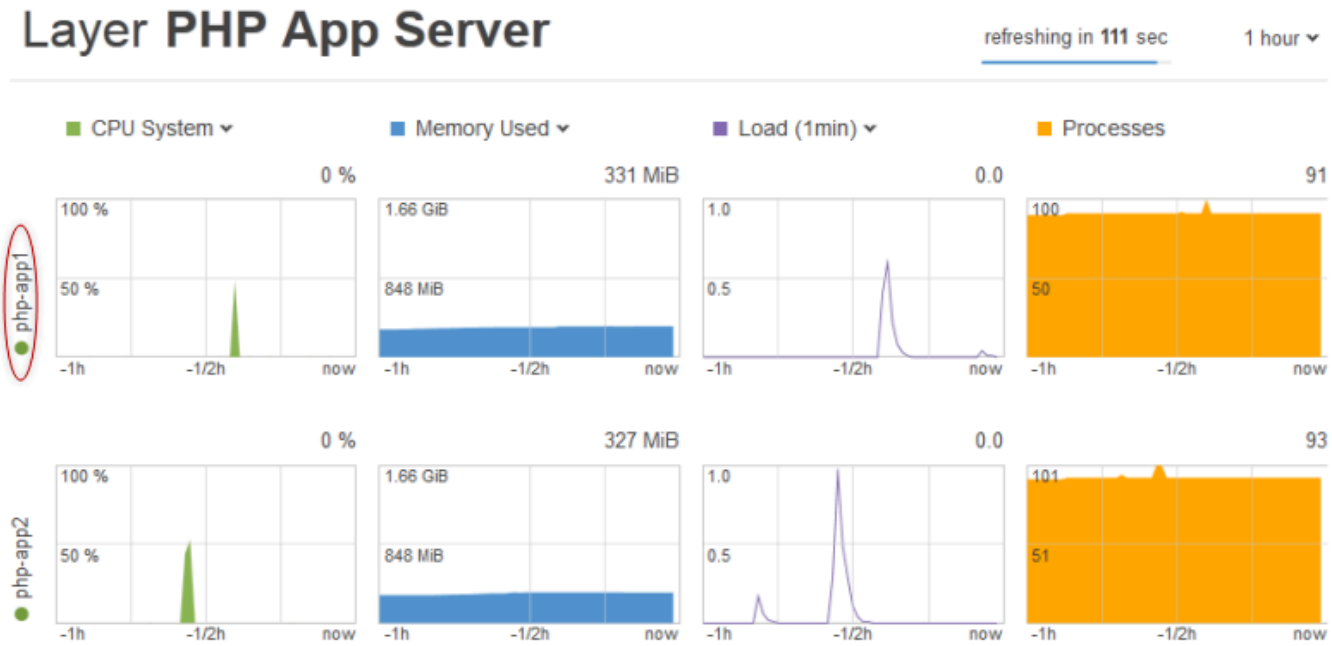
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Amazon CloudWatch를 사용하여 스택에 대한 지표를 제공하고 모니터링 페이지에서 사용자의 편의를 위해 이를 요약합니다. 전체 스택, 지정된 계층 또는 지정된 인스턴스에 대한 측정치를 볼 수 있습니다.

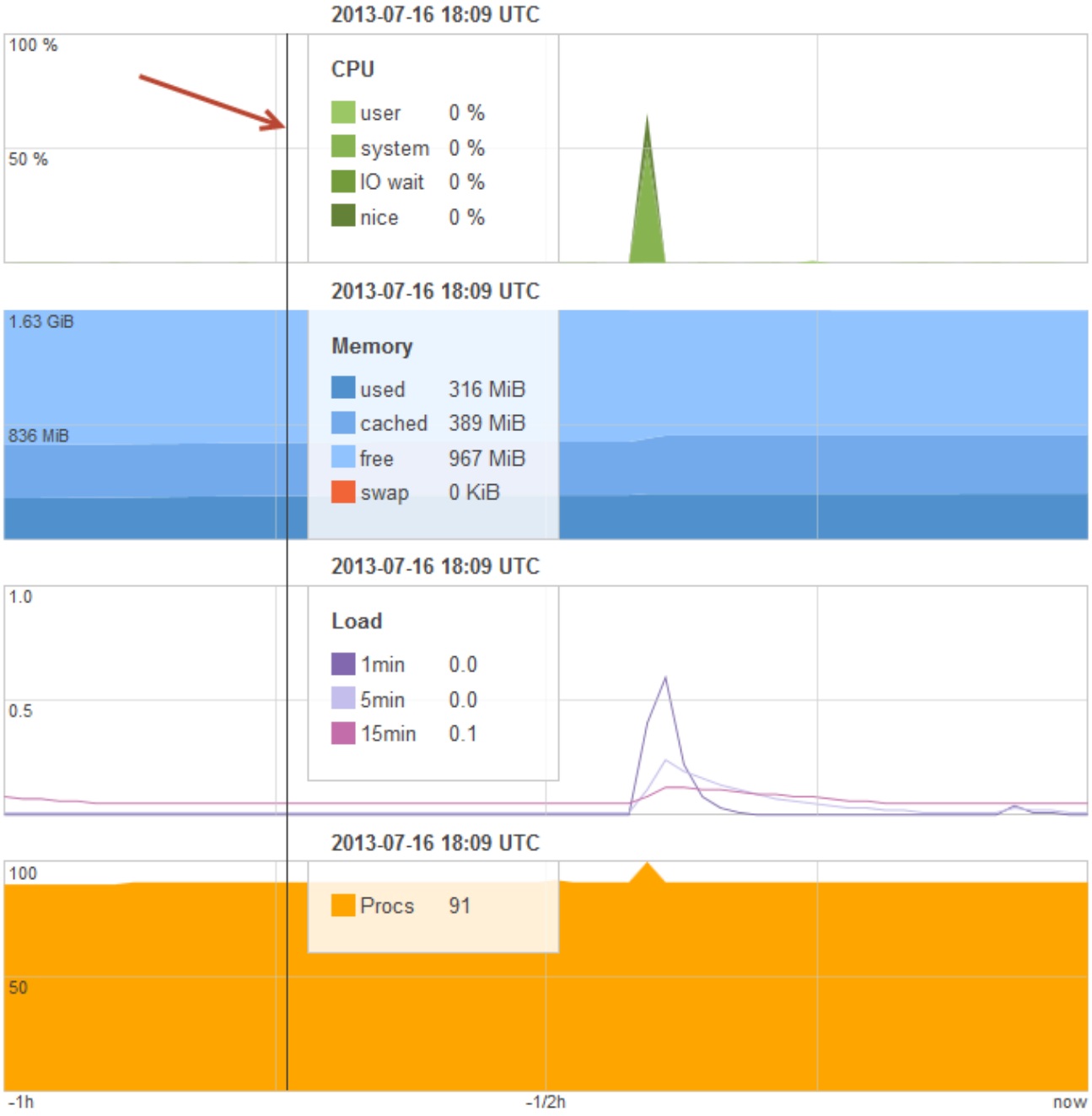
MyStack을 모니터링하려면

1. 탐색 창에서 [모니터링]을 클릭합니다. 그러면 각 계층에 대한 평균 측정치와 함께 일련의 그래프가 표시됩니다. [CPU 시스템], [사용 메모리] 및 [로드]에 대한 메뉴를 사용해 여러 가지 관련 측정치를 표시할 수 있습니다.

2. [PHP 앱 서버]를 클릭하면 계층의 각 인스턴스에 대한 측정치가 표시됩니다.



3. [php-app1]를 클릭하면 해당 인스턴스에 대한 측정치가 표시됩니다. 슬라이드를 이동하면 특정 시점에 대한 측정치를 볼 수 있습니다.


Instance php-app1
refreshing in
2013-07-16 18:09 UTC
100 %
50 %
CPU
user 0 %
system 0 %
IO wait 0 %
nice 0 %
2013-07-16 18:09 UTC
1.63 GiB
836 MiB
Memory
used 316 MiB
cached 389 MiB
free 967 MiB
swap 0 KiB
2013-07-16 18:09 UTC
1.0
0.5
Load
1min 0.0
5min 0.0
15min 0.1
2013-07-16 18:09 UTC
100
50
Procs 91
-1h
-1/2h
now

> **Note**
>
> AWS OpsWorks Stacks는 Ganglia 모니터링 서버도 지원하므로 일부 애플리케이션에는 이점이 있을 수 있습니다. 자세한 내용은 Ganglia 계층 단원을 참조하십시오.

5단계: MyStack 삭제

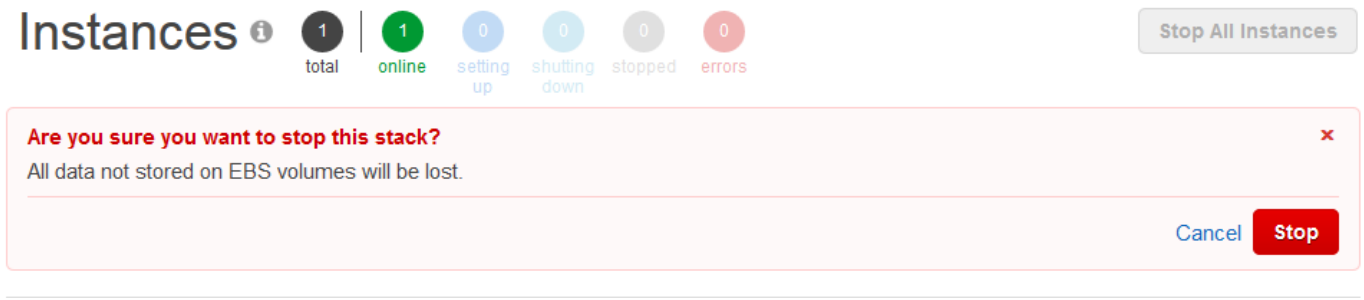
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon EC2 인스턴스와 같은 AWS 리소스를 사용하기 시작하는 순간부터 사용량에 따라 요금이 부과됩니다. 잠시 동안 사용하지 않을 경우 원치 않는 요금이 발생하지 않도록 인스턴스를 중지해야 합니다. 스택이 더 이상 필요하지 않으면 삭제할 수 있습니다.

MyStack을 삭제하려면

1. 모든 인스턴스 중지

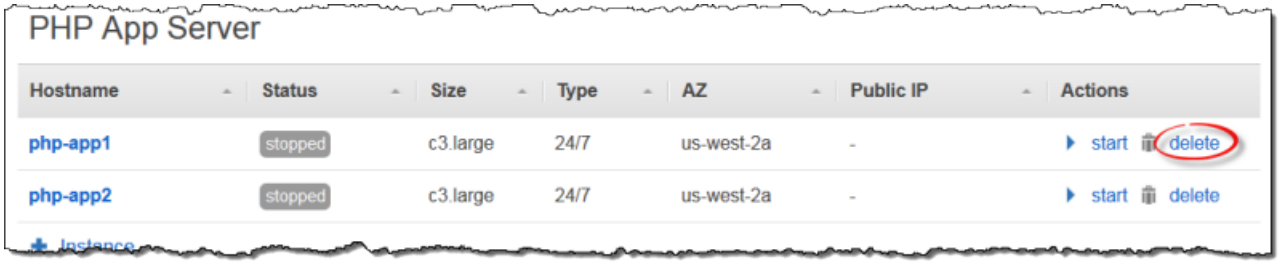
   [인스턴스] 페이지에서 [모든 인스턴스 중지]를 클릭하고, 작업을 확인하라는 메시지가 표시되면 [중지]를 클릭합니다.



   중지를 클릭하면 AWS OpsWorks Stacks는 연결된 Amazon EC2 인스턴스를 종료하지만 탄력적 IP 주소 또는 Amazon EBS 볼륨과 같은 연결된 리소스는 종료하지 않습니다.

2. 모든 인스턴스 삭제

   인스턴스를 종료하면 연결된 Amazon EC2 인스턴스만 종료됩니다. 인스턴스가 중지됨 상태가 된 뒤에는 각각의 인스턴스를 삭제해야 합니다. PHP 앱 서버 계층의 php-app1 인스턴스의 작업 열에서 삭제를 클릭합니다.

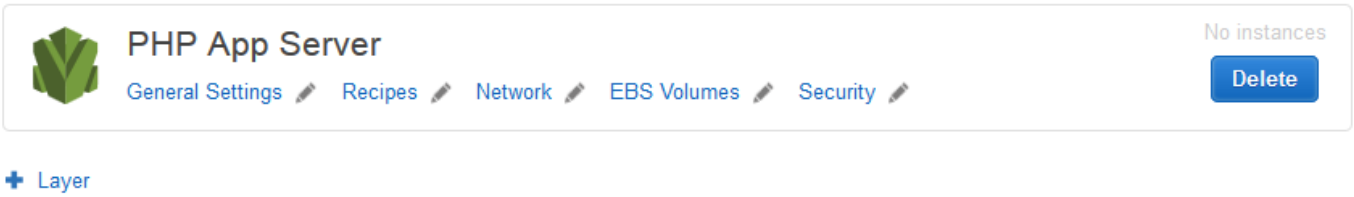


   AWS OpsWorks 그러면 Stacks가 삭제 확인을 요청하고 종속 리소스를 표시합니다. 이러한 리소스 중 일부 또는 전부를 유지하도록 설정할 수 있습니다. 이 예에는 종속 리소스가 없으므로 [삭제]를 클릭하기만 하면 됩니다.

   php-app2 및 [MySQL] 인스턴스인 db-master1에 대해 이 프로세스를 반복합니다. db-master1에는 연결된 Amazon Elastic Block Store 볼륨이 기본적으로 선택되어 있습니다. 선택된 상태로 두면 이 볼륨이 인스턴스와 함께 삭제됩니다.

3. 계층 삭제

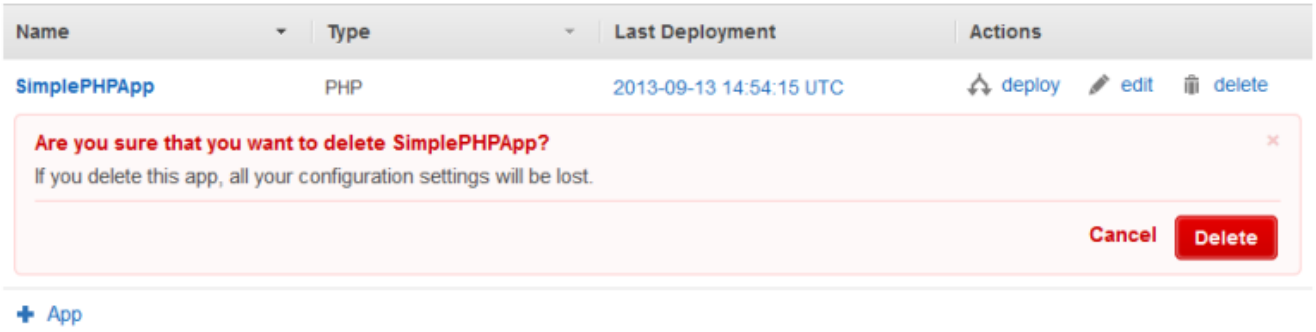
   [계층] 페이지에서 [삭제]를 클릭한 다음 [삭제]를 클릭하여 확인합니다.



   MySQL 계층에 대해 이 프로세스를 반복합니다.

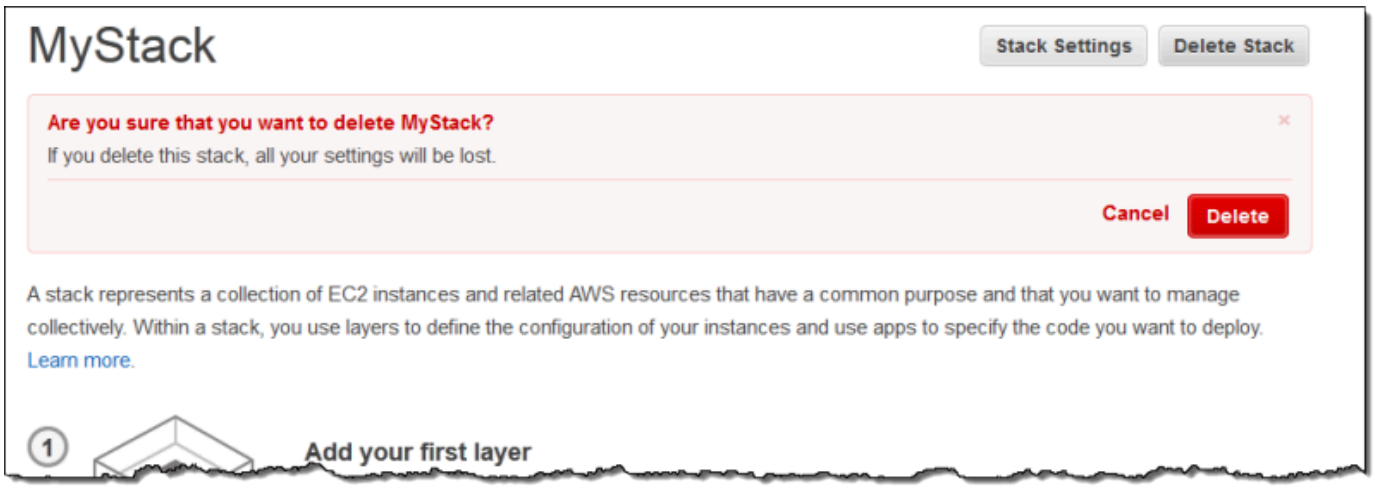
4. 앱 삭제

   앱 페이지에서 SimplePHPapp 앱의 작업 열에서 삭제를 클릭한 다음 삭제를 클릭하여 확인합니다.



5. MyStack 삭제

   [스택] 페이지에서 [스택 삭제]를 클릭한 다음 [삭제]를 클릭하여 확인합니다.



이 연습을 마칩니다.

첫 번째 Node.js 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 예제는 Node.js 애플리케이션 서버를 지원하는 Linux 스택을 생성하고 간단한 애플리케이션을 배포하는 방법을 설명합니다. 스택의 구성 요소는 다음과 같습니다.

- 두 개의 인스턴스가 있는 Node.js 앱 서버 계층
- 애플리케이션 서버 인스턴스 간에 트래픽을 분산하는 Elastic Load Balancing 로드 밸런서
- 백엔드 데이터베이스를 제공하는 Amazon Relational Database Service(RDS) 서비스 계층

주제

- 사전 조건
- 애플리케이션 구형
- 데이터베이스 서버 및 로드 밸런서 생성
- 스택 생성
- 애플리케이션 배포
- 다음 단계

사전 조건

이 연습에서는 다음과 같이 가정합니다.

- AWS 계정과 AWS OpsWorks Stacks 사용 방법에 대한 기본적인 이해가 있어야 합니다.

  AWS OpsWorks Stacks 또는 AWS를 처음 사용하는 경우의 입문 자습서를 완료하여 기본 사항을 알아봅니다Chef 11 Linux 스택 시작하기.
- Node.js 애플리케이션 구현 방법에 대한 기본적인 이해가 있습니다.

  Node.js가 생소하다면 Node: Up and Running과 같은 입문용 자습서를 완료하여 기본 사항에 대해 알아보십시오.
- 이 예제를 사용할 AWS 리전에서 이미 하나 이상의 스택이 생성되어 있습니다.

  리전에서 첫 번째 스택을 생성하면 AWS OpsWorks Stacks는 각 계층 유형에 대해 Amazon Elastic Compute Cloud(Amazon EC2) 보안 그룹을 생성합니다. Amazon RDS 데이터베이스(DB) 인스턴스를 생성하려면 이러한 보안 그룹이 필요합니다. AWS OpsWorks Stacks를 처음 사용하는 경우의 자습서를 따를 때와 동일한 리전을이 예제에 사용하는 것이 좋습니다Chef 11 Linux 스택 시작하기. 새 리전을 사용하려면 리전에서 새 스택을 생성합니다. 스택이 계층 또는 인스턴스를 포함할 필요는 없습니다. 스택을 생성하는 즉시 AWS OpsWorks Stacks는 리전에 보안 그룹 세트를 자동으로 추가합니다.
- 기본 VPC에서 스택을 생성합니다.

이 연습에서 EC2-Classic을 사용할 수 있지만 일부 세부 내용이 약간 다릅니다. 예를 들어 EC2-Classic에서는 서브넷 대신 인스턴스의 가용 영역(AZ)을 지정합니다.

- IAM 사용자에게는 AWS OpsWorks Stacks에 대한 전체 액세스 권한이 있습니다.

  보안상의 이유로 이 연습에서 계정의 루트 자격 증명은 사용하지 않는 것이 좋습니다. 대신 AWS OpsWorks Stacks 전체 액세스 권한이 있는 사용자를 생성하고 AWS OpsWorks Stacks에서 해당 자격 증명을 사용합니다. 자세한 내용은 관리 사용자 생성 단원을 참조하십시오.

애플리케이션 구형

이 안내에서는 Amazon RDS DB 인스턴스에 연결하고 인스턴스의 데이터베이스를 나열하는 간단한 Express 애플리케이션을 사용합니다.

애플리케이션을 구현하려면 워크스테이션의 편리한 위치에 nodedb라는 디렉터리를 생성하고 다음 세 파일을 이 디렉터리에 추가합니다.

주제

- 패키지 설명자
- 레이아웃 파일
- 코드 파일

패키지 설명자

애플리케이션의 패키지 설명자를 생성하려면 다음 콘텐츠가 포함된 `package.json`이라는 파일을 nodedb 디렉터리에 추가합니다. `package.json`은 Express 애플리케이션에 필요하며 애플리케이션의 루트 디렉터리에 위치해야 합니다.

```
{
  "name": "Nodejs-DB",
  "description": "Node.js example application",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "ejs": "*",
    "mysql": "*"
  }
}
```

이 package.json 예제는 최소한으로 구성되어 있습니다. 필수 name 및 version 속성을 정의하고 종속 패키지를 나열합니다.

- express는 Express 패키지를 참조합니다.
- ejs는 애플리케이션이 텍스트를 HTML 레이아웃 파일에 삽입할 때 사용하는 EJS 패키지를 참조합니다.
- mysql은 애플리케이션이 RDS 인스턴스에 연결할 때 사용하는 node-mysql 패키지를 참조합니다.

패키지 설명자에 대한 자세한 정보는 package.json 단원을 참조하세요.

레이아웃 파일

애플리케이션의 레이아웃 파일을 생성하려면 views 디렉터리에 nodedb 디렉터리를 추가하고, 콘텐츠가 포함된 views이라는 파일을 index.html에 추가합니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>AWS Opsworks Node.js Example</title>
</head>
<body>
  <h1>AWS OpsWorks Node.js Example</h1>
    <p>Amazon RDS Endpoint: <i><%= hostname %></i></p>
    <p>User: <i><%= username %></i></p>
    <p>Password: <i><%= password %></i></p>
    <p>Port: <i><%= port %></i></p>
    <p>Database: <i><%= database %></i></p>

    <p>Connection: <%= connectionerror %></p>
    <p>Databases: <%= databases %></p>
</body>
</html>
```

이 예제에서는 레이아웃 파일이 Amazon RDS로부터의 일부 데이터를 표시하는 간단한 HTML 문서입니다. 각 <%= ... => 요소는 이 다음에 생성할 애플리케이션의 코드 파일에서 정의되는 변수의 값을 나타냅니다.

코드 파일

애플리케이션의 코드 파일을 생성하려면 다음 콘텐츠가 포함된 `server.js` 파일을 nodedb 디렉터리에 추가합니다.

> **Important**
>
> AWS OpsWorks Stacks를 사용하면 Node.js 애플리케이션의 기본 코드 파일의 이름을 `server.js` 지정하고 애플리케이션의 루트 폴더에 위치해야 합니다.

```
var express = require('express');
var mysql = require('mysql');
var dbconfig = require('opsworks'); //[1] Include database connection data
var app = express();
var outputString = "";

app.engine('html', require('ejs').renderFile);

//[2] Get database connection data
app.locals.hostname = dbconfig.db['host'];
app.locals.username = dbconfig.db['username'];
app.locals.password = dbconfig.db['password'];
app.locals.port = dbconfig.db['port'];
app.locals.database = dbconfig.db['database'];
app.locals.connectionerror = 'successful';
app.locals.databases = '';

//[3] Connect to the Amazon RDS instance
var connection = mysql.createConnection({
    host: dbconfig.db['host'],
    user: dbconfig.db['username'],
    password: dbconfig.db['password'],
    port: dbconfig.db['port'],
    database: dbconfig.db['database']
});

connection.connect(function(err)
{
    if (err) {
        app.locals.connectionerror = err.stack;
```

```
        return;
    }
});

// [4] Query the database
connection.query('SHOW DATABASES', function (err, results) {
    if (err) {
        app.locals.databases = err.stack;
    }

    if (results) {
        for (var i in results) {
            outputString = outputString + results[i].Database + ', ';
        }
        app.locals.databases = outputString.slice(0, outputString.length-2);
    }
});

connection.end();

app.get('/', function(req, res) {
    res.render('./index.html');
});

app.use(express.static('public'));

//[5] Listen for incoming requests
app.listen(process.env.PORT);
```

이 예제는 데이터베이스 연결 정보를 표시하며 데이터베이스 서버에 쿼리하여 서버의 데이터베이스를 표시합니다. 손쉽게 이를 일반화하여 필요에 따라 데이터베이스와 상호 작용할 수 있습니다. 다음 참고 사항은 이전 코드에서 번호가 매겨진 주석 단원을 참조합니다.

[1] Include database connection data

이 `require` 문은 데이터베이스 연결 데이터를 포함시킵니다. 나중에 설명하듯이, 데이터베이스에 데이터베이스 인스턴스를 앱에 연결하면 AWS OpsWorks Stacks는 연결 데이터를 이라는 파일에 넣습니다`opsworks.js`.이 파일은 다음과 유사합니다.

```
exports.db = {
  "host":"nodeexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com",
  "database":"nodeexampledb",
```

```
  "port":3306,
  "username":"opsworksuser",
  "password":"your_pwd",
  "reconnect":true,
  "data_source_provider":"rds",
  "type":"mysql"}
```

opsworks.js는 애플리케이션의 shared/config 디렉터리인 /srv/www/*app_shortname*/shared/config에 있습니다. 그러나 AWS OpsWorks Stacks는 애플리케이션의 루트 디렉터리opsworks.js에에 symlink를 배치하므로 만 사용하여 객체를 포함할 수 있습니다require 'opsworks'.

[2] Get database connection data

이 문 세트는 db 객체에서 app.locals 속성 세트로 값을 할당하여 opsworks.js의 연결 데이터를 표시합니다. 각 속성은 index.html 파일 내 <%= ... %> 요소 중 하나에 매핑됩니다. 렌더링된 문서에서는 <%= ... %> 요소가 해당 속성 값으로 바뀝니다.

[3] Connect to the Amazon RDS instance

이 예제는 node-mysql을 사용하여 데이터베이스에 액세스합니다. 데이터베이스에 연결하려면 연결 데이터를 connection에 전달하여 createConnection 객체를 생성한 후 connection.connect를 호출하여 연결을 설정합니다.

[4] Query the database

연결이 설정되면 이 예제가 connection.query를 호출하여 데이터베이스를 쿼리합니다. 이 예제는 단순히 서버의 데이터베이스 이름을 쿼리합니다. query가 데이터베이스 이름이 Database 속성에 할당된 results 객체를 데이터베이스마다 하나씩 반환합니다. 이 예제는 이름을 연결하여 그 결과를 렌더링된 HTML 페이지에 목록을 표시하는 app.locals.databases,에 할당합니다.

이 예제에는 RDS 인스턴스를 생성할 때 지정된 nodeexampledb 데이터베이스와 Amazon RDS가 자동으로 생성한 4개의 데이터베이스 등 5개의 데이터베이스가 있습니다.

[5] Listen for incoming requests

이 마지막 문은 지정된 포트에서 수신 요청을 수신 대기합니다. 명시적 포트 값을 지정할 필요는 없습니다. 스택에 앱을 추가할 때 애플리케이션이 HTTP 또는 HTTPS 요청을 지원하는지 여부를 지정합니다. 그러면 AWS OpsWorks Stacks는 PORT 환경 변수를 80(HTTP) 또는 443(HTTPS)으로 설정하고 애플리케이션에서 해당 변수를 사용할 수 있습니다.

다른 포트에서도 수신이 가능하지만 Node.js 앱 서버 계층의 내장 보안 그룹인 AWS-OpsWorks-nodejs-App-Server는 포트 80, 443, 22(SSH)로의 인바운드 사용자 트래픽을 허용합니다. 다른 포

트로의 인바운드 사용자 트래픽을 허용하려면 적절한 인바운드 규칙을 사용하여 보안 그룹을 생성하고 Node.js 앱 서버 계층에 할당합니다. 내장 보안 그룹을 편집하여 인바운드 규칙을 수정하지 마십시오. 스택을 생성할 때마다 AWS OpsWorks Stacks는 기본 제공 보안 그룹을 표준 설정으로 덮어쓰므로 변경 사항이 손실됩니다.

> **Note**
>
> 연결된 앱을 생성 또는 업데이트할 때 애플리케이션에 사용자 지정 환경 변수를 연결할 수 있습니다. 또한 사용자 지정 JSON과 사용자 지정 레시피를 사용하여 데이터를 애플리케이션으로 전달할 수도 있습니다. 자세한 내용은 애플리케이션으로 데이터 전달 섹션을 참조하세요.

데이터베이스 서버 및 로드 밸런서 생성

이 예제는 Amazon RDS 데이터베이스 서버 및 Elastic Load Balancing 로드 밸런서 인스턴스를 사용합니다. 각 인스턴스를 별도로 생성한 후 스택으로 가져와야 합니다. 이 섹션에서는 새 데이터베이스 및 로드 밸런서 인스턴스를 생성하는 방법을 설명합니다. 기존 인스턴스를 대신 사용할 수 있지만 인스턴스를 올바로 구성할 수 있도록 절차를 끝까지 읽는 것이 좋습니다.

다음 절차는 이 예제에는 충분한, 최소한으로 구성된 RDS DB 인스턴스를 생성하는 방법을 설명합니다. 자세한 내용은 Amazon RDS 사용 설명서를 참조하세요.

RDS DB 인스턴스를 생성하려면

1. 콘솔을 엽니다.

   Amazon RDS 콘솔을 열고 지역을 미국 서부(오레곤)로 설정합니다. 탐색 창에서 [RDS 대시보드]를 선택한 다음 [DB 인스턴스 시작]을 선택합니다.

2. 데이터베이스 엔진을 지정합니다.

   데이터베이스 엔진으로 [MySQL 커뮤니티 에디션]을 선택합니다.

3. 다중 AZ 배포를 거부합니다.

   [아니요, 이 인스턴스...]를 선택한 다음, [다음]을 선택합니다. 이 예제에는 다중 AZ 배포가 필요 없습니다.

4. 기본 설정을 구성합니다.

   [DB 인스턴스 세부 정보] 페이지에서 다음 설정을 지정합니다.

- [DB 인스턴스 클래스]: [db.t2.micro]
- [다중 AZ 배포]: [No]
- 할당된 스토리지: **5** GB
- DB 인스턴스 식별자: **nodeexample**
- 마스터 사용자 이름: **opsworksuser**.
- Master Password:: 선택한 암호

나중에 사용하기 위해 인스턴스 식별자, 사용자 이름 및 암호를 적어 두고 다른 옵션에 대해서는 기본 설정을 수락한 다음 [다음]을 선택합니다.

5. 고급 설정을 구성합니다.

   [고급 설정 구성] 페이지에서 다음 설정을 지정합니다.

   - 데이터베이스 이름: **nodeexampledb**
   - [DB 보안 그룹]: [AWS-OpsWorks-DB-Master-Server]

   > **Note**
   >
   > [AWS-OpsWorks-DB-Master-Server] 보안 그룹에서는 스택의 인스턴스만 데이터베이스에 액세스하도록 허용합니다. 데이터베이스에 직접 액세스하려면 적절한 인바운드 규칙을 사용하여 RDS DB 인스턴스에 추가 보안 그룹을 연결합니다. 자세한 내용은 Amazon RDS 보안 그룹을 참조하세요. 또한 VPC에 인스턴스를 배치하여 액세스를 제어할 수도 있습니다. 자세한 내용은 VPC에서 스택 실행 섹션을 참조하세요.

   나중에 사용하기 위해 데이터베이스 이름을 적어 두고 다른 설정에 대해서는 기본값을 수락한 다음 [DB 인스턴스 시작]을 선택합니다.

다음 절차는 이 예제를 위한 Elastic Load Balancing 로드 밸런서를 생성하는 방법을 설명합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서를 참조하세요.

로드 밸런서를 생성하려면

1. Amazon EC2 콘솔을 엽니다.

   Amazon EC2 콘솔을 열고 리전이 미국 서부(오레곤)로 설정되어 있는지 확인합니다. 탐색 창에서 [로드 밸런서]를 선택한 다음 [로드 밸런서 만들기]를 선택합니다.

2. 로드 밸런서를 정의합니다.

   [로드 밸런서 정의] 페이지에서 다음 설정을 지정하세요.

   - 명칭 – **Node-LB**
   - LB 내부 생성 – 내 기본 VPC

   다른 옵션에 대해서는 기본 설정을 수락하고 [다음]을 선택합니다.

3. 보안 그룹을 할당합니다.

   [보안 그룹 할당] 페이지에서 다음 그룹을 지정합니다.

   - 기본 VPC 보안 그룹
   - AWS-OpsWorks-nodejs-App-Server

   다음을 선택합니다. [보안 설정 구성] 페이지에서 [다음]을 선택합니다. 이 예제에는 보안 리스너가 필요 없습니다.

4. 상태 확인을 구성합니다.

   상태 확인 구성 페이지에서 Ping 경로를 **/**로 설정하고 다른 설정에 대해서는 기본값을 수락합니다. 다음을 선택합니다. [EC2 인스턴스 추가] 페이지에서 [다음]을 선택합니다. 태그 추가 페이지에서 검토 및 생성을 선택합니다. 스택은 로드 밸런서에 EC2 인스턴스를 추가하는 작업을 처리하므로이 예제에서는 태그가 필요하지 않습니다. AWS OpsWorks

5. 로드 밸런서를 생성합니다.

   [검토] 페이지에서 [만들기]를 선택하여 로드 밸런서를 생성합니다.

스택 생성

이제 스택을 생성하는 데 필요한 모든 구성 요소가 준비되었습니다.

스택을 생성하는 방법

1. AWS OpsWorks Stacks 콘솔에 로그인합니다.

   AWS OpsWorks Stacks 콘솔에 로그인한 다음 스택 추가를 선택합니다.
2. 스택을 생성합니다.

   새 스택을 생성하려면 [Chef 11 스택]을 선택하고 다음 설정을 지정합니다.

   - – **`NodeStack`**
   - 리전 – 미국 서부(오레곤)

     스택은 모든 AWS 리전에서 생성할 수 있지만 자습서의 경우 미국 서부(오레곤)를 선택하는 것이 좋습니다.

   [스택 추가]를 선택합니다. 스택 구성 설정에 대한 자세한 정보는 새 스택 생성 단원을 참조하세요.
3. 연결된 로드 밸런서와 함께 Node.js 앱 서버 계층을 추가합니다.

   [NodeStack] 페이지에서 [계층 추가]를 선택하고 다음 설정을 지정합니다.

   - 계층 유형 — Node.js 앱 서버
   - 탄력적 로드 밸런서 - 노드-LB

   다른 설정에 대해서는 기본값을 수락하고 [계층 추가]를 선택합니다.
4. 인스턴스를 계층에 추가하고 시작합니다.

   탐색 창에서 [인스턴스]를 선택하고 다음과 같이 Rails 앱 서버 계층에 인스턴스 2개를 추가합니다.

   1. Node.js 앱 서버에서 인스턴스 추가를 선택합니다.

      [크기]를 [t2.micro]로 설정하고, 다른 설정에 대해서는 기본값을 수락하고 [인스턴스 추가]를 선택합니다.
   2. [+인스턴스]를 선택한 다음 두 번째 t2.micro 인스턴스를 다른 서브넷의 계층에 추가합니다.

      이렇게 하면 해당 인스턴스가 다른 AZ(가용 영역)에 배치됩니다.
   3. [인스턴스 추가]를 선택합니다.
   4. 두 인스턴스를 모두 시작하려면 [모든 인스턴스 시작]을 선택합니다.

이 계층에 Elastic Load Balancing 로드 밸런서를 할당했습니다. 인스턴스가 온라인 상태가 되거나 온라인 상태에서 나가면 AWS OpsWorks Stacks는 로드 밸런서에 인스턴스를 자동으로 등록하거나 등록 취소합니다.

> **Note**
>
> 프로덕션 스택의 경우 다중 AZ에 애플리케이션 서버 인스턴스를 배포하는 것이 좋습니다. 사용자가 AZ에 연결할 수 없는 경우 로드 밸런서가 나머지 영역의 인스턴스로 수신 트래픽을 라우트하므로 사이트가 계속해서 작동합니다.

5. 스택에 RDS DB 인스턴스를 등록합니다.

   탐색 창에서 [리소스]를 선택하고 다음과 같이 스택에 RDS DB 인스턴스를 등록합니다.

   1. [RDS] 탭을 선택한 다음 [미등록 RDS DB 표시] 인스턴스를 선택합니다.
   2. [nodeexampledb] 인스턴스를 선택하고 다음 설정을 지정합니다.
      - 사용자 - 인스턴스를 생성할 때 지정한 마스터 사용자 이름입니다(이 예제에서는). **`opsworksuser`**.
      - 암호 - 인스턴스를 생성할 때 지정한 마스터 암호입니다.
   3. 스택에 등록을 선택하여 스택에 RDS DB 인스턴스를 Amazon RDS 서비스 계층으로 추가합니다.

   > **Warning**
   >
   > AWS OpsWorks Stacks는 사용자 또는 암호 값을 검증하지 않고 단순히 애플리케이션에 전달합니다. 이러한 값을 잘못 입력하면 애플리케이션이 데이터베이스에 연결할 수 없습니다.

   스택에 등록을 선택하여 스택에 RDS DB 인스턴스를 Amazon RDS 서비스 계층으로 추가합니다.

애플리케이션 배포

애플리케이션은 원격 리포지토리에 저장해야 합니다. 배포 시 AWS OpsWorks Stacks는 코드 및 관련 파일을 리포지토리에서 애플리케이션 서버 인스턴스로 배포합니다. 편의를 위해 이 예제는 리포지토

리로 퍼블릭 Amazon Simple Storage Service(S3) 아카이브를 사용하지만, Git 및 하위 버전을 비롯해 다수의 다른 리포지토리 유형을 사용할 수 있습니다. 자세한 내용은 애플리케이션 소스 섹션을 참조하세요.

애플리케이션을 배포하려면

1. 애플리케이션을 아카이브 파일에 패키지로 포함합니다.

   `.zip` 디렉터리와 하위 디렉터리 nodedb.zip의 nodedb 아카이브를 생성합니다. 또한 gzip, bzip2 및 tarball을 비롯하여 다른 아카이브 파일 유형을 사용할 수도 있습니다. AWS OpsWorks Stacks는 압축되지 않은 tarball을 지원하지 않습니다. 자세한 내용은 애플리케이션 소스 단원을 참조하십시오.

2. Amazon S3로 아카이브 파일을 업로드합니다.

   `nodedb.zip`을 Amazon S3 버킷에 업로드하고, 이 파일을 퍼블릭으로 지정한 다음 나중에 사용하기 위해 파일의 URL을 복사해 둡니다. 버킷 생성 및 파일 업로드 방법에 대한 자세한 내용은 Amazon Simple Storage Service 시작하기를 참조하세요.

   > **Note**
   >
   > AWS OpsWorks Stacks는 Amazon S3 버킷에서 프라이빗 파일을 배포할 수도 있지만 간소화를 위해이 예제에서는 퍼블릭 파일을 사용합니다. 자세한 내용은 애플리케이션 소스 단원을 참조하십시오.

3. AWS OpsWorks Stacks 앱을 생성합니다.

   AWS OpsWorks Stacks 콘솔로 돌아가서 탐색 창에서 앱을 선택한 다음 앱 추가를 선택합니다. 다음 설정을 지정합니다.

   - 이름 – `NodeDB`.

     이 문자열은 앱의 표시 이름입니다. 대부분의 경우 AWS OpsWorks Stacks가 모든 문자를 소문자로 변환하고 구두점을 제거하여 표시 이름에서 생성하는 앱의 짧은 이름이 필요합니다. 이 예제에서 짧은 이름은 nodedb입니다. 앱을 생성한 후 앱의 짧은 이름을 확인하려면 [앱] 페이지에서 앱을 선택하여 앱의 세부 정보 페이지를 표시합니다.

   - 유형 – `Node.js`.
   - 데이터 소스 유형 – `RDS`.
   - 데이터베이스 인스턴스 - 앞서 등록한 Amazon RDS DB 인스턴스를 선택합니다.

- 데이터베이스 이름 - 앞서 생성한 데이터베이스 이름을 지정합니다. 이 예제에서는 `nodeexampledb`입니다.
- 리포지토리 유형 – `Http Archive`.

  이 리포지토리 유형은 퍼블릭 Amazon S3 파일에 사용해야 합니다. `S3 Archive` 유형은 프라이빗 아카이브에만 사용됩니다.
- 리포지토리 URL - 아카이브 파일의 Amazon S3 URL.

나머지 설정에 대해서는 기본값을 사용하고 [앱 추가]를 클릭하여 앱을 생성합니다.

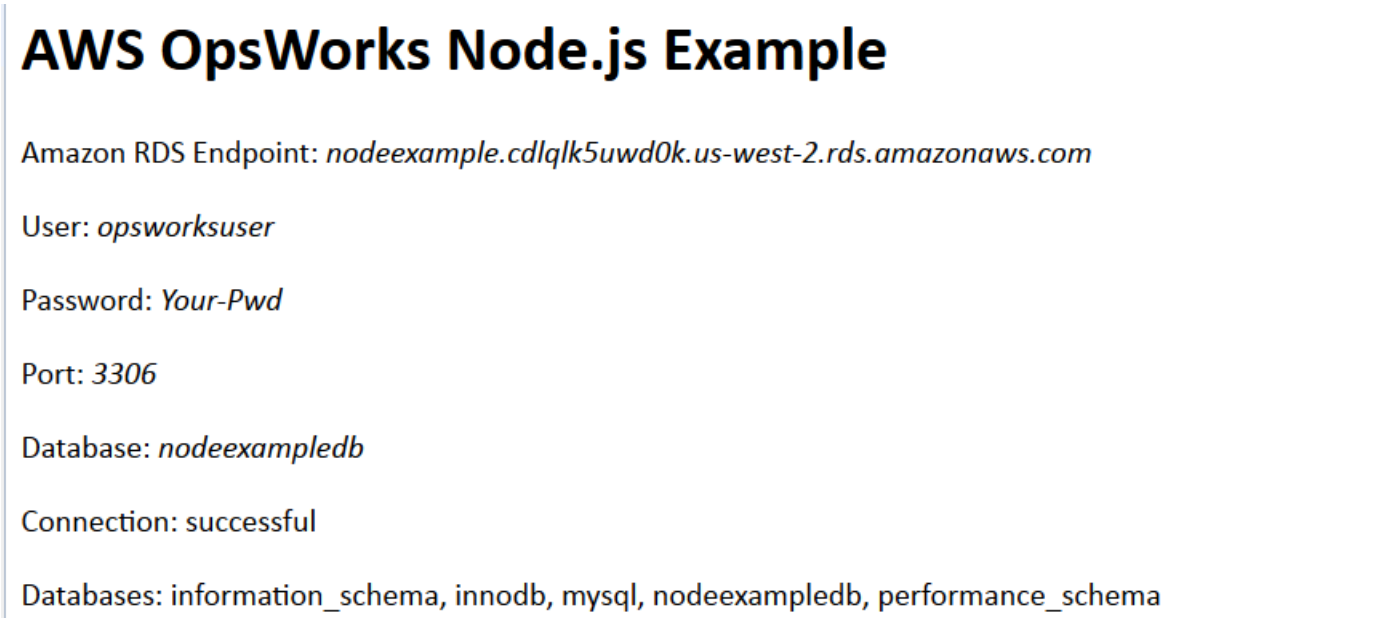
4. 앱을 배포합니다.

   앱 페이지로 이동한 다음 NodeDB 앱의 작업 열에서 배포를 선택합니다. 그런 다음 배포를 선택하여 앱을 서버 인스턴스에 배포합니다. AWS OpsWorks Stacks는 각 인스턴스에서 Deploy 레시피를 실행하여 리포지토리에서 애플리케이션을 다운로드하고 서버를 다시 시작합니다. 각 인스턴스에 녹색 확인 표시가 나타나고 [상태]가 [성공]이면 배포가 완료되고 애플리케이션이 요청 처리를 시작할 준비가 된 것입니다.

   > **Note**
   >
   > 배포에 실패한 경우 로그 열에서 표시를 선택하여 배포의 Chef 로그를 표시합니다. 맨 아래 근처에 오류 정보가 표시되어 있습니다.

5. 애플리케이션을 엽니다.

   애플리케이션을 열려면 [계층]를 선택하고 로드 밸런서를 선택한 다음 로드 밸런서의 DNS 이름을 선택합니다. 그러면 로드 밸런서로 HTTP 요청이 전송됩니다. 다음과 같은 내용이 표시되어야 합니다.



> **Note**
>
> AWS OpsWorks Stacks는 설정 중에 새 인스턴스에 앱을 자동으로 배포합니다. 수동 배포는 온라인 인스턴스에서만 필요합니다. 자세한 내용은 앱 배포 섹션을 참조하세요. 보다 복잡한 배포 전략을 비롯해 배포에 대한 전반적인 설명은 앱과 쿡북의 관리 및 배포 단원을 참조하세요.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 연습에서는 간단한 Node.js 애플리케이션 서버 스택을 설정하는 기본 사항을 살펴보았습니다. 아래는 다음 단계에 대한 몇 가지 제안입니다.

Node.js 내장 쿡북 살펴보기

인스턴스 구성 방법을 자세히 알아보려면 Stacks가 소프트웨어를 설치 및 구성하는 데 사용하는 레시피와 관련 파일이 포함된 계층의 내장 쿡북인 opsworks_nodejs와 Stacks가 앱을 배포하는 데 사용하는 레시피가 포함된 내장 배포 AWS OpsWorks 쿡북을 참조하세요. AWS OpsWorks

서버 구성 사용자 지정

이 예제 스택은 매우 기본적입니다. 프로덕션용으로 사용하려면 스택을 사용자 지정해야 할 것입니다. 자세한 내용은 AWS OpsWorks 스택 사용자 지정 섹션을 참조하세요.

SSL 지원을 추가

앱에 대한 SSL 지원을 활성화하고 앱을 생성할 때 AWS OpsWorks Stacks에 적절한 인증서를 제공할 수 있습니다. 그러면 AWS OpsWorks Stacks가 적절한 디렉터리에 인증서를 설치합니다. 자세한 내용은 SSL 사용 단원을 참조하십시오.

인 메모리 데이터 캐싱을 추가

프로덕션 수준 사이트에서 Redis 또는 Memcache와 같은 인 메모리 키-값 저장소에 데이터를 캐시하여 성능을 개선하는 경우가 종종 있습니다. 둘 중 하나를 AWS OpsWorks Stacks 스택과 함께 사용할 수 있습니다. 자세한 내용은 ElastiCache Redis 및 Memcached 섹션을 참조하세요.

보다 복잡한 배포 전략을 사용

이 예제에서는 모든 인스턴스를 동시에 업데이트하는 간단한 앱 배포 전략을 사용했습니다. 이 접근 방식은 간단하고 빠르지만 오류의 여지가 없습니다. 배포가 실패하거나 업데이트에 문제가 있을 경우 프로덕션 스택의 모든 인스턴스가 영향을 받을 수 있으므로 문제를 해결할 때까지는 사이트가 중단 또는 비활성화될 수 있습니다. 배포 전략에 대한 자세한 정보는 앱과 쿡북의 관리 및 배포 단원을 참조하세요.

Node.js 앱 서버 계층을 확장하세요.

다양한 방법으로 계층을 확장할 수 있습니다. 예를 들어 인스턴스에서 스크립트를 실행하는 레시피를 구현하거나 앱 배포를 사용자 지정하기 위한 Chef 배포 후크를 구현할 수 있습니다. 자세한 내용은 계층 확장 섹션을 참조하세요.

환경 변수를 정의

연결된 앱의 환경 변수를 정의하여 애플리케이션에 데이터를 전달할 수 있습니다. 앱을 배포할 때 AWS OpsWorks Stacks는 사용자가 앱에서 액세스할 수 있도록 해당 변수를 내보냅니다. 자세한 내용은 환경 변수 사용 단원을 참조하십시오.

AWS OpsWorks 스택 사용자 지정

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 내장 계층은 다양한 용도로 충분한 표준 기능을 제공합니다. 하지만 다음과 같은 상황을 마주할 수 있습니다.

- 내장 계층의 표준 구성이 적당하지만 이상적이지는 않아 특정 요구 사항에 맞게 최적화하기를 원할 경우

  예를 들어 최대 worker 프로세스 수 또는 `keepalivetimeout` 값과 같은 설정에 자체 값을 지정하여 Static Web Server 계층의 Nginx 서버 구성을 조정할 수 있습니다.
- 내장 계층의 기능이 훌륭하지만 추가 패키지를 설치하거나 몇몇 사용자 지정 설치 스크립트를 실행하여 확장하기를 원할 경우

  예를 들어 Redis 서버도 설치하여 PHP 앱 서버 계층을 확장할 수 있습니다.
- 내장 계층으로 처리할 수 없는 요구 사항이 있는 경우

  예를 들어 AWS OpsWorks Stacks에는 일부 인기 있는 데이터베이스 서버에 대한 기본 제공 계층이 포함되어 있지 않습니다. 계층의 인스턴스에 이러한 서버를 설치하는 사용자 지정 계층을 생성할 수 있습니다.
- 사용자 지정 계층만 지원하는 Windows 스택을 실행하는 경우

AWS OpsWorks Stacks는 특정 요구 사항에 맞게 계층을 사용자 지정하는 다양한 방법을 제공합니다. 다음 예제는 복잡성 및 파워가 증가하는 순서로 나열되어 있습니다.

> **Note**
>
> 일부 접근 방식은 Linux 스택에서만 작동합니다. 자세한 정보는 이하의 주제를 참조하세요.

- 사용자 지정 JSON을 사용하여 기본 AWS OpsWorks Stacks 설정을 재정의합니다.

- 기본 스택 설정을 재정의하는 속성 파일을 사용하여 사용자 지정 Chef AWS OpsWorks 쿡북을 구현합니다.
- 기본 Stacks 템플릿을 재정의하거나 확장하는 템플릿을 사용하여 사용자 지정 Chef AWS OpsWorks 쿡북을 구현합니다.
- shell 스크립트를 실행하는 간단한 레시피를 사용하여 사용자 지정 Chef 쿡북을 구현합니다.
- 디렉터리 구성, 패키지 설치, 구성 파일 생성, 앱 배포 등의 작업을 수행하는 레시피를 사용하여 사용자 지정 Chef 쿡북을 구현합니다.

스택의 Chef 버전 및 운영 체제에 따라 레시피를 재정의할 수도 있습니다.

- Chef 0.9 및 11.4 스택에서는 동일한 쿡북 및 레시피 이름의 사용자 지정 레시피를 구현하여 내장 레시피를 재정의할 수 없습니다.

  각 수명 주기 이벤트에 대해 AWS OpsWorks Stacks는 항상 기본 제공 레시피를 먼저 실행한 다음 사용자 지정 레시피를 실행합니다. 이러한 Chef 버전은 동일한 쿡북 및 레시피 이름을 두 번 실행하지 않으므로 내장 레시피가 우선 순위를 가져 사용자 지정 레시피를 실행되지 않습니다.
- Chef 11.10 스택에서는 내장 레시피를 재정의할 수 있습니다.

  자세한 내용은 쿡북 설치 및 우선 순위 섹션을 참조하세요.
- Windows 스택에서는 내장 레시피를 재정의할 수 없습니다.

  AWS OpsWorks Stacks가 Windows 스택용 Chef 실행을 처리하는 방식에서는 내장 레시피를 재정의할 수 없습니다.

> **Note**
>
> 많은 기법이 사용자 지정 쿡북을 사용하기 때문에 쿡북 구현에 익숙하지 않은 경우 먼저 쿡북과 레시피를 읽어야 합니다. 쿡북 기본 사항는 사용자 지정 쿡북 구현에 대한 자세한 자습서를 제공하고 Stacks 인스턴스용 AWS OpsWorks 쿡북을 구현하는 방법에 대한 몇 가지 세부 정보를 AWS OpsWorks Stacks용 쿡북 구현 다룹니다.

주제

- 속성을 재정의하여 AWS OpsWorks 스택 구성 사용자 지정
- 사용자 지정 템플릿을 사용하여 AWS OpsWorks Stacks 구성 파일 확장
- 계층 확장

- 사용자 지정 Tomcat 서버 계층 생성
- 스택 구성 및 배포 속성

속성을 재정의하여 AWS OpsWorks 스택 구성 사용자 지정

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> Windows 스택과 Chef 12 Linux 스택의 경우 AWS OpsWorks Stacks는 기본 제공 레시피와 사용자 지정 레시피에 별도의 Chef 실행을 사용합니다. 즉, 이 섹션에 설명된 기법을 사용해서는 Windows 스택과 Chef 12 Linux 스택의 내장 속성을 재정의할 수 없습니다.

레시피와 템플릿은 계층 구성 또는 애플리케이션 서버 설정 같은 인스턴스 또는 스택별 정보를 다양한 Chef 속성에 의존합니다. 이들 속성에는 몇 가지 소스가 있습니다.

- 사용자 지정 JSON - 필요한 경우, 스택을 생성, 업데이트 또는 복제하거나 앱을 배포할 때 사용자 지정 JSON 속성을 지정할 수 있습니다.
- 스택 구성 속성 -AWS OpsWorks 스택은 콘솔 설정을 통해 지정하는 정보를 포함하여 스택 구성 정보를 보유하도록 이러한 속성을 정의합니다.
- 배포 속성 — AWS OpsWorks는 Deploy 이벤트의 배포 관련 속성을 정의합니다.
- 쿡북 속성 - 내장 쿡북과 사용자 지정 쿡북은 일반적으로 애플리케이션 서버 구성 설정 등 쿡북별 값을 나타내는 속성이 포함된 속성 파일을 포함하고 있습니다.
- Chef - Chef의 Ohai 도구는 CPU 유형과 설치된 메모리 같은 다양한 시스템 구성 설정을 나타내는 속성을 정의합니다.

스택 구성 및 배포 속성과 내장 쿡북 속성의 완전한 목록은 스택 구성 및 배포 속성: Linux 및 내장 쿡북 속성를 참조하세요. Ohai 속성에 대한 자세한 정보는 Ohai를 참조하세요.

배포 또는 구성과 같은 수명 주기 이벤트가 발생하거나 `execute_recipes` 또는 `update_packages`같은 스택 명령을 실행하면 AWS OpsWorks Stacks는 다음을 수행합니다.

- 해당 명령을 각각의 해당 인스턴스의 에이전트에 전송합니다.

  이 에이전트는 적절한 레시피를 실행합니다. 예를 들어 Deploy 이벤트의 경우, 에이전트는 내장 Deploy 레시피를 실행한 다음 사용자 지정 Deploy 레시피를 실행합니다.
- 사용자 지정 JSON 및 배포 속성을 스택 구성 속성에 병합하고 인스턴스에 설치합니다.

사용자 지정 JSON의 속성, 스택 구성 및 배포 속성, 쿡북 속성 및 Ohai 속성은 레시피에 속성 값을 제공하는 노드 객체에 병합됩니다. 인스턴스는 사용자 지정 JSON을 비롯한 스택 구성 속성에 관한 한 기본적으로 상태 비저장입니다. 배포 또는 스택 명령을 실행하면 연결된 레시피는 명령과 함께 다운로드된 스택 구성 속성을 사용합니다.

주제

- 속성 우선 순위
- 사용자 지정 JSON을 사용한 속성 재정의
- 사용자 지정 AWS OpsWorks 쿡북 속성을 사용하여 스택 속성 재정의

속성 우선 순위

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

속성이 고유하게 정의되는 경우, Chef는 해당 속성을 노드 객체에 단순히 통합합니다. 하지만 모든 속성 소스는 어떤 속성도 정의할 수 있으므로 동일한 속성이 값이 각기 다른 여러 정의를 가질 수 있습니다. 예를 들어 내장 `apache2` 쿡북은 `node[:apache][:keepalive]`를 정의하지만 사용자 지정 JSON 또는 사용자 지정 쿡북에서도 해당 속성을 정의할 수 있습니다. 한 속성에 여러 정의가 있는 경우, 이 정의들은 뒤에 설명하는 순서대로 평가되며, 노드 객체는 우선 순위가 가장 높은 정의를 수신합니다.

속성은 다음과 같이 정의됩니다.

```
node.type[:attribute][:sub_attribute][:...]=value
```

속성에 여러 정의가 있는 경우 유형에 따라 우선 순위가 있는 정의가 결정되고 해당 정의가 노드 객체에 통합됩니다. AWS OpsWorks Stacks는 다음 속성 유형을 사용합니다.

- default - 이것은 가장 일반적인 유형이며, 기본적으로 "속성이 아직 정의되지 않았다면 이 값을 사용하라"는 뜻입니다. 속성의 모든 정의가 `default` 유형인 경우, 평가 순서에서 첫 번째 정의가 우선하며 후속 값들은 무시됩니다. Stacks는 모든 스택 구성 및 배포 속성 AWS OpsWorks 정의를 `default` 유형으로 설정합니다.
- normal - 이 유형의 속성은 모든 `default` 속성 또는 평가 순서에서 앞서 정의된 `normal` 속성을 재정의합니다. 예를 들어 첫 번째 속성이 내장 쿡북의 속성이고 `default` 유형을 가지고 있으며 두 번째는 사용자가 정의한 속성으로 `normal` 유형을 가지고 있다면 두 번째 속성이 우선합니다.
- set - 이전의 쿡북에서 볼 수 있는 사용되지 않는 유형입니다. 이 유형은 같은 우선 순위를 갖는 `normal`로 대체되었습니다.

Chef는 다른 모든 속성 정의에 우선하는 `automatic` 유형을 비롯한 몇 가지 추가 속성 유형을 지원합니다. Chef의 Ohai 도구에 의해 생성되는 속성 정의는 모두 `automatic` 유형이므로 사실상 읽기 전용입니다. 이는 일반적으로 문제가 되지 않습니다. 재정의할 이유가 없고 AWS OpsWorks Stacks의 속성과 별개이기 때문입니다. 다만 사용자 지정 쿡북 속성의 이름을 지정할 때는 Ohai 속성과 구별되도록 주의해야 합니다. 자세한 정보는 속성 정보를 참조하세요.

> **Note**
>
> Ohai 도구는 명령줄에서 실행할 수 있는 실행 파일입니다. 인스턴스의 Ohai 속성을 나열하려면 인스턴스에 로그인하고 터미널 창에서 `ohai`를 실행합니다. 매우 긴 출력이 생성되므로 유의하세요.

다음은 다양한 속성 정의를 노드 객체에 통합하는 단계입니다.

1. 모든 사용자 지정 스택 구성 속성을 스택 구성 및 배포 속성에 병합합니다.

   스택이나 특정 배포에 대해 사용자 지정 JSON 속성을 설정할 수 있습니다. 사용자 지정 JSON 속성은 평가 순서에서 첫 번째이며 사실상 `normal` 유형입니다. 사용자 지정 JSON에서 하나 이상의 스택 구성 속성도 정의되는 경우, 사용자 지정 JSON 값이 우선합니다. 그렇지 않으면 AWS OpsWorks Stacks는 단순히 사용자 지정 JSON 속성을 스택 구성에 통합합니다.
2. 모든 배포 사용자 지정 JSON 속성을 스택 구성 및 배포 속성에 병합합니다.

배포 사용자 지정 JSON 속성도 사실상 `normal` 유형이므로 내장 및 사용자 지정 스택 구성 JSON과 내장 배포 JSON보다 우선합니다.

3. 스택 구성 및 배포 속성을 인스턴스의 노드 객체에 병합합니다.
4. 인스턴스의 내장 쿡북 속성을 노드 객체에 병합합니다.

   내장 쿡북 속성은 모두 `default` 유형입니다. 일반적으로 사용자 지정 JSON으로 정의했기 때문에 스택 구성 및 배포 속성에도 하나 이상의 내장 쿡북 속성이 정의된 경우 스택 구성 정의가 내장 쿡북 정의보다 우선합니다. 다른 모든 내장 쿡북 속성은 단순히 노드 객체에 통합됩니다.
5. 인스턴스의 사용자 지정 쿡북 속성을 노드 객체에 병합합니다.

   사용자 지정 쿡북 속성은 일반적으로 `normal` 또는 `default` 유형입니다. 고유한 속성은 노드 객체에 통합됩니다. 1~3단계에서 사용자 지정 쿡북 속성도 정의한 경우(일반적으로 사용자 지정 JSON으로 정의했기 때문에) 우선 순위는 사용자 지정 쿡북 속성 유형에 따라 달라집니다.

   - 1-3단계에서 정의된 속성은 사용자 지정 쿡북 `default` 속성보다 우선합니다.
   - 사용자 지정 쿡북 `normal` 속성은 1-3단계의 정의보다 우선합니다.

> **Important**
>
> 사용자 지정 쿡북 `default` 속성을 사용하여 스택 구성 또는 내장 쿡북 속성을 재정의하지 마십시오. 사용자 지정 쿡북 속성은 마지막으로 평가되기 때문에 `default` 속성은 우선 순위가 가장 낮으며 아무것도 재정의할 수 없습니다.

사용자 지정 JSON을 사용한 속성 재정의

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> AWS OpsWorks Stacks는 Windows 스택의 Chef 실행을 Linux 스택의 경우와 다르게 처리하므로 Windows 스택의 경우이 섹션에서 설명하는 기술을 사용할 수 없습니다.

Stacks 속성을 재정의하는 가장 간단한 방법은 기본 제공 및 사용자 지정 AWS OpsWorks 쿡북 속성뿐만 아니라 스택 구성 및 배포 속성보다 우선하는 사용자 지정 JSON으로 정의하는 것입니다`default`. 자세한 내용은 속성 우선 순위 단원을 참조하십시오.

> **Important**
>
> 스택 구성 및 배포 속성을 재정의할 때는 주의해야 합니다. 예를 들어 `opsworks` 네임스페이스에서 속성을 재정의하면 내장 속성을 방해할 수 있습니다. 자세한 내용은 스택 구성 및 배포 속성 섹션을 참조하세요.

사용자 지정 JSON을 사용하여 고유한 속성을 정의할 수도 있습니다(일반적으로 데이터를 사용자 지정 레시피에 전달하기 위해). 속성은 노드 객체에 통합되며, 레시피는 표준 Chef 노드 구문을 사용하여 속성 단원을 참조할 수 있습니다.

사용자 지정 JSON 지정 방법

사용자 지정 JSON을 사용하여 속성 값을 재정의하려면 먼저 속성의 정규화된 속성 이름을 확인해야 합니다. 그런 다음 재정의하려는 속성을 포함한 JSON 객체를 생성하고 선호하는 값으로 설정합니다. 편의상 스택 구성 및 배포 속성: Linux 및 내장 쿡북 속성 문서에는 일반적으로 사용되는 스택 구성, 배포 및 내장 쿡북 속성과 그 정규화된 이름이 나와 있습니다.

객체의 상위-하위 관계는 적절한 정규화된 Chef 노드와 일치해야 합니다. 예를 들어 다음과 같은 Apache 속성을 변경하려 한다고 가정하겠습니다.

- 노드가 `node[:apache][:keepalivetimeout]`이고 3 기본값을 가진 `keepalivetimeout` 속성입니다.
- 노드가 `node[:apache][:logrotate][:schedule]`이고 `"daily"` 기본값을 가진 `logrotate schedule` 속성입니다.

이 속성들을 재정의하고 값을 각각 5와 `"weekly"`로 설정하려면 다음 사용자 지정 JSON을 사용합니다.

```
{
  "apache" : {
    "keepalivetimeout" : 5,
    "logrotate" : {
       "schedule" : "weekly"
    }
  }
}
```

사용자 지정 JSON을 지정할 시기

다음 작업에 사용자 지정 JSON 구조를 지정할 수 있습니다.

- 새 스택 생성
- 스택 업데이트
- 스택 명령 실행
- 스택 복제
- 앱 배포

각 작업에 대해 AWS OpsWorks Stacks는 사용자 지정 JSON 속성을 스택 구성 및 배포 속성과 병합하고 인스턴스로 전송하여 노드 객체에 병합합니다. 다만 다음을 참고하세요.

- 스택을 생성, 복제 또는 업데이트할 때 사용자 지정 JSON을 지정하는 경우, 속성은 모든 후속 수명 주기 이벤트 및 스택 명령의 스택 구성 및 배포 속성에 병합됩니다.
- 배포를 위해 사용자 지정 JSON을 지정하는 경우, 속성은 해당 이벤트의 스택 구성 및 배포 속성에만 병합됩니다.

  후속 배포에 이러한 사용자 지정 속성을 사용하려면 사용자 지정 JSON을 다시 명시적으로 지정해야 합니다.

속성은 레시피에 의해 사용될 때 인스턴스에만 영향을 미친다는 점을 명심해야 합니다. 속성 값을 재정의해도 해당 속성 단원을 참조하는 후속 레시피가 없다면 변경은 효과가 없습니다. 연결된 레시피가 실행되기 전에 사용자 지정 JSON을 전송하거나 적절한 레시피를 다시 실행해야 합니다.

사용자 지정 JSON 모범 사례

사용자 지정 JSON을 사용하여 AWS OpsWorks Stacks 속성을 재정의할 수 있지만 정보를 수동으로 입력하는 것은 다소 번거롭고 어떤 종류의 소스 제어도 하지 않습니다. 사용자 지정 JSON은 다음과 같은 용도로 가장 잘 활용할 수 있습니다.

- 소수의 속성만 재정의하고 그 밖에는 사용자 지정 쿡북을 사용할 필요가 없을 때.

  사용자 지정 JSON을 사용하면 단지 2가지 속성을 재정의하기 위해 쿡북 리포지토리를 설정하고 유지 관리하는 부담을 피할 수 있습니다.
- 암호 또는 인증 키 같은 중요한 값.

  쿡북 속성은 리포지토리에 저장되므로 중요한 정보가 침해될 위험이 있습니다. 그 대신 더미 값으로 기본값을 정의하고 사용자 지정 JSON을 사용하여 실제 값을 설정하세요.
- 값은 다양할 것으로 예상됩니다.

  예를 들어 권장되는 관행은 프로덕션 스택을 별도의 개발 및 스테이징 스택으로 지원하는 것입니다. 이 스택들이 지불을 받는 애플리케이션을 지원한다고 가정해 보십시오. 사용자 지정 JSON을 사용하여 지불 엔드포인트를 지정하는 경우, 스테이징 스택에 테스트 URL을 지정할 수 있습니다. 업데이트된 스택을 프로덕션 스택으로 마이그레이션할 준비가 되면 같은 쿡북을 사용하고 사용자 지정 JSON을 사용하여 지불 엔드포인트를 프로덕션 URL로 설정할 수 있습니다.
- 특정 스택 또는 배포 명령에 고유한 값.

사용자 지정 AWS OpsWorks 쿡북 속성을 사용하여 스택 속성 재정의

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> Windows 스택의 경우 AWS OpsWorks Stacks는 기본 제공 레시피와 사용자 지정 레시피에 별도의 Chef 실행을 사용합니다. 즉, 이 섹션에 설명된 기법을 사용해서는 Windows 스택의 내장 속성을 재정의할 수 없습니다.

사용자 지정 JSON은 Stacks 스택 구성 및 내장 AWS OpsWorks 쿡북 속성을 재정의하는 편리한 방법이지만 몇 가지 제한이 있습니다. 특히 사용 시마다 사용자 지정 JSON을 수동으로 입력해야 하므로 정의를 관리할 확실한 방법이 없습니다. 더 좋은 방법은 사용자 지정 쿡북 속성 파일을 사용하여 내장 속성을 재정의하는 것입니다. 이렇게 하면 소스 제어 아래에 정의를 배치할 수 있습니다.

사용자 지정 속성 파일을 사용하여 AWS OpsWorks Stacks 정의를 재정의하는 절차는 간단합니다.

Stacks 속성 AWS OpsWorks 정의를 재정의하려면

1. 쿡북과 레시피 단원의 설명에 따라 쿡북 리포지토리를 설정합니다.
2. 재정의할 속성이 포함된 내장 쿡북과 같은 이름으로 쿡북을 생성합니다. 예를 들어 Apache 속성을 재정의하려면 쿡북의 이름을 apache2로 지정해야 합니다.
3. 쿡북에 `attributes` 폴더를 추가하고 `customize.rb` 폴더에 파일을 추가합니다.
4. 재정의할 내장 쿡북의 속성별로 이 파일에 속성 정의를 추가하고 원하는 값으로 설정합니다. 속성은 `normal` 유형 이상이어야 하며 해당 AWS OpsWorks Stacks 속성과 노드 이름이 정확히 동일해야 합니다. 노드 이름을 포함한 AWS OpsWorks Stacks 속성의 자세한 목록은 스택 구성 및 배포 속성: Linux 및 섹션을 참조하세요내장 쿡북 속성. 속성 및 속성 파일에 대한 자세한 정보는 속성 파일 정보를 참조하세요.

   > **Important**
   >
   > AWS OpsWorks Stacks 속성을 재정의하려면 속성`normal`이 유형이어야 합니다. `default` 유형에는 우선 순위가 없습니다. 예를 들어, `customize.rb` 파일에 `default[:apache][:keepalivetimeout] = 5` 속성 정의가 있어도 내장된 `apache.rb` 속성 파일의 해당 속성이 먼저 평가되면 그 속성이 우선 적용됩니다. 자세한 내용은 속성 재정의 섹션을 참조하세요.

5. 재정의할 속성이 포함된 각 내장 쿡북에 대해 2 - 4단계를 반복합니다.

6. 스택에 대해 사용자 지정 쿡북을 활성화하고 Stacks가 AWS OpsWorks 쿡북을 스택의 인스턴스에 다운로드하는 데 필요한 정보를 제공합니다. 자세한 내용은 사용자 지정 쿡북 설치 단원을 참조하십시오.

> **Note**
>
> 이 절차에 대한 완전한 안내는 내장 속성 재정의 단원을 참조하세요.

이제 후속 수명 주기 이벤트, 배포 명령 및 스택 명령에 사용되는 노드 객체에 AWS OpsWorks Stacks 값 대신 속성 정의가 포함됩니다.

예를 들어 `keepalivetimeout`에서 설명한 내장 `logrotate schedule` 및 사용자 지정 JSON 지정 방법 설정을 재정의하려면 `apache2apache` 쿡북을 리포지토리에 추가하고 `customize.rb` 파일을 다음 콘텐츠와 함께 쿡북의 `attributes` 폴더에 추가합니다.

```
normal[:apache][:keepalivetimeout] = 5
normal[:apache][:logrotate][:schedule] = 'weekly'
```

> **Important**
>
> 연결된 내장 속성 파일의 사본을 수정하여 AWS OpsWorks Stacks 속성을 재정의해서는 안 됩니다. 예를 들어 `apache.rb`를 `apache2/attributes` 폴더에 복사하고 일부 설정을 수정하는 경우, 기본적으로 내장 파일의 모든 속성이 재정의됩니다. 레시피는 사본의 속성 정의를 사용하고 내장 파일은 무시합니다. AWS OpsWorks Stacks가 나중에 내장 속성을 수정하는 경우, 수동으로 사본을 업데이트하지 않는 한 레시피는 변경 사항에 액세스하지 못합니다.
>
> 이런 상황을 피하기 위해 모든 내장 쿡북에는 `customize.rb` 명령을 통해 모든 모듈에 필요한 빈 `include_attribute` 속성이 포함되어 있습니다. `customize.rb` 사본의 속성을 재정의하면 이러한 특정 속성에만 영향을 미칠 수 있습니다. 레시피는 그 밖의 모든 속성 값을 내장 속성 파일에서 가져오며, 재정의하지 않은 속성의 현재 값을 자동으로 가져옵니다.
>
> 이 방법은 쿡북 리포지토리의 속성 수를 적게 유지하도록 도움으로써 유지 관리 부담이 줄어들고 향후 업그레이드 관리가 쉬워집니다.

사용자 지정 템플릿을 사용하여 AWS OpsWorks Stacks 구성 파일 확장

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> AWS OpsWorks Stacks는 Windows 스택의 Chef 실행을 Linux 스택의 경우와 다르게 처리하므로 Windows 스택의 경우이 섹션에서 설명하는 기술을 사용할 수 없습니다.

AWS OpsWorks Stacks는 템플릿을 사용하여 일반적으로 많은 설정의 속성에 의존하는 구성 파일과 같은 파일을 생성합니다. 사용자 지정 JSON 또는 사용자 지정 AWS OpsWorks 쿡북 속성을 사용하여 Stacks 정의를 재정의하는 경우 기본 설정이 AWS OpsWorks Stacks 설정 대신 구성 파일에 통합됩니다. 그러나 AWS OpsWorks Stacks는 가능한 모든 구성 설정에 대한 속성을 반드시 지정하지는 않습니다. 일부 설정의 기본값을 수락하고 템플릿에서 직접 다른 설정을 하드코딩합니다. 해당하는 Stacks 속성이 없는 경우 사용자 지정 JSON 또는 사용자 지정 AWS OpsWorks 쿡북 속성을 사용하여 기본 설정을 지정할 수 없습니다.

사용자 지정 템플릿을 생성해 추가 구성 설정을 포함하도록 구성 파일을 확장할 수 있습니다. 그런 다음 필요한 구성 설정이나 그 밖의 콘텐츠를 파일에 추가하고 하드코딩된 설정을 재정의할 수 있습니다. 템플릿에 대한 자세한 정보는 템플릿 단원을 참조하세요.

> **Note**
>
> opsworks-agent.monitrc.erb를 제외한 어떤 내장 템플릿도 재정의할 수 있습니다.

사용자 지정 템플릿을 생성하려면

1. 내장 쿡북과 동일한 구조 및 디렉터리 이름으로 쿡북을 생성합니다. 그런 다음 해당 디렉터리에서 사용자 지정하려는 내장 템플릿과 동일한 이름을 가진 템플릿 파일을 만듭니다. 예를 들어 사용자 지정 템플릿을 사용하여 Apache `httpd.conf` 구성 파일을 확장하려는 경우 리포지

토리에서 apache2 쿡북을 구현해야 하고 템플릿 파일은 apache2/templates/default/apache.conf.erb여야 합니다. 정확히 동일한 이름을 사용하면 AWS OpsWorks Stacks가 사용자 지정 템플릿을 인식하고 기본 제공 템플릿 대신 사용할 수 있습니다.

내장 쿡북의 GitHub 리포지토리에서 쿡북으로 내장 템플릿 파일을 복사한 다음 필요에 따라 수정하는 것이 가장 간단합니다.

> **Important**
>
> 사용자 지정하려는 템플릿 파일을 제외하고는 내장 쿡북에서 어떠한 파일도 복사하지 마십시오. 다른 유형의 쿡북 파일(예: 레시피)을 복사하면 중복 Chef 리소스가 생성되어 오류가 발생할 수 있습니다.

또한 쿡북에는 사용자 지정 속성, 레시피 및 관련 파일이 포함될 수 있지만 파일 이름이 내장 파일 이름과 중복되면 안 됩니다.

2. 템플릿 파일을 사용자 지정하여 요구 사항을 충족하는 구성 파일을 생성합니다. 설정을 더 추가하고, 기존 설정을 삭제하고, 하드코딩된 설정을 대체하는 등의 작업을 수행할 수 있습니다.
3. 아직 수행하지 않은 경우에는 스택 설정을 편집해 사용자 지정 쿡북을 사용하고 쿡북 리포지토리를 지정할 수 있습니다. 자세한 내용은 사용자 지정 쿡북 설치 섹션을 참조하세요.

> **Note**
>
> 이 절차에 대한 완전한 안내는 내장 템플릿 재정의 단원을 참조하세요.

템플릿을 재정의하기 위해 레시피를 구현하거나 계층 구성에 레시피를 추가할 필요가 없습니다. AWS OpsWorks Stacks는 항상 기본 제공 레시피를 실행합니다. 구성 파일을 생성하는 레시피를 실행할 때는 내장 템플릿 대신 사용자 지정 템플릿을 자동으로 사용합니다.

> **Note**
>
> AWS OpsWorks Stacks가 기본 제공 템플릿을 변경하면 사용자 지정 템플릿이 동기화되지 않아 더 이상 제대로 작동하지 않을 수 있습니다. 예를 들어 템플릿이 종속 파일을 참조하고 파일 이름이 변경된다고 가정해 보겠습니다. AWS OpsWorks Stacks는 이러한 변경을 자주 수행하지 않으며 템플릿이 변경되면 변경 사항이 나열되고 새 버전으로 업그레이드할 수 있는 옵션이

제공됩니다. Stacks AWS OpsWorks 리포지토리의 변경 사항을 모니터링하고 필요에 따라 템플릿을 수동으로 업데이트해야 합니다.

계층 확장

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 속성 수정 또는 템플릿 사용자 지정으로 처리할 수 있는 것 이상으로 내장 계층을 사용자 지정해야 할 경우가 가끔 있습니다. 예를 들어 symlink를 생성하거나 파일 또는 폴더 모드를 설정하거나 추가 패키지를 설치하는 등의 작업이 필요하다고 가정해 봅시다. 최소 기능 이상을 제공하려면 사용자 지정 계층을 확장해야 합니다. 이 경우 사용자 지정 작업을 처리하기 위한 레시피로 하나 이상의 사용자 지정 쿡북을 구현해야 합니다. 이 항목에서는 레시피를 사용하여 계층을 확장하는 방법을 예시합니다.

Chef를 처음 사용하는 경우 먼저 다양한 일반적인 작업을 수행하기 위한 쿡북을 구현하는 기본적인 방법을 소개하는 쿡북 101 섹션을 읽어야 합니다. 사용자 지정 계층을 구현하는 방법의 상세한 예는 사용자 지정 Tomcat 서버 계층 생성 단원을 참조하세요.

주제

- 레시피를 사용하여 스크립트 실행
- Chef 배포 후크 사용
- Linux 인스턴스에서 Cron 작업 실행
- Linux 인스턴스에서 패키지 설치 및 구성

레시피를 사용하여 스크립트 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

필요한 사용자 지정 작업을 수행하는 스크립트가 이미 있는 경우 가장 간편하게 계층을 확장하는 방법은 종종 스크립트를 실행하기 위한 간단한 레시피를 구현하는 것입니다. 그런 다음 레시피를 적절한 수명 주기 이벤트(일반적으로 설정 또는 Deploy)에 할당하거나 `execute_recipes` 스택 명령을 사용하여 수동으로 레시피를 실행할 수 있습니다.

다음 예제는 Linux 인스턴스에서 shell 스크립트를 실행하지만, Windows PowerShell 스크립트 등 다른 유형의 스크립트에도 동일한 접근 방식을 사용할 수 있습니다.

```
cookbook_file "/tmp/lib-installer.sh" do
  source "lib-installer.sh"
  mode 0755
end

execute "install my lib" do
  command "sh /tmp/lib-installer.sh"
end
```

`cookbook_file` 리소스는 쿡북의 `files` 디렉터리 내 하위 디렉터리에 저장된 파일을 나타내며, 파일을 인스턴스의 지정된 위치로 전송합니다. 이 예제에서는 shell 스크립트 `lib-installer.sh`를 인스턴스의 `/tmp` 디렉터리로 전송하고 파일의 모드를 `0755`로 설정합니다. 자세한 정보는 cookbook_file 단원을 참조하세요.

`execute` 리소스는 명령을 나타냅니다(예: shell 명령). 이 예제에서는 `lib-installer.sh`를 실행합니다. 자세한 정보는 실행을 참조하세요.

또한 스크립트를 레시피에 통합하여 레시피를 실행할 수도 있습니다. 다음 예제는 bash 스크립트를 실행하지만 Chef는 Csh, Perl, Python 및 Ruby도 지원합니다.

```
script "install_something" do
  interpreter "bash"
  user "root"
  cwd "/tmp"
  code <<-EOH
    #insert bash script
```

```
  EOH
end
```

`script` 리소스는 스크립트를 나타냅니다. 예제에서는 bash 인터프리터를 지정하고, 사용자를 `"root"`로 설정하고, 작업 디렉터리를 `/tmp`로 설정합니다. 그런 다음 `code` 블록의 bash 스크립트를 실행합니다. 이 스크립트는 필요한 만큼 많은 줄을 포함할 수 있습니다. 자세한 정보는 script를 참조하세요.

레시피를 사용하여 스크립트를 실행하는 방법에 대한 자세한 정보는 예제 7: 명령 및 스크립트 실행 단원을 참조하세요. Windows 인스턴스에서 PowerShell을 실행하는 방법에 대한 예제는 Windows PowerShell 스크립트 실행 단원을 참조하세요.

Chef 배포 후크 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

필요한 작업을 수행하기 위한 사용자 지정 레시피를 구현하고 적절한 계층의 설정 이벤트에 레시피를 할당하여 배포를 사용자 지정할 수 있습니다. 대안적이고 때로는 더 간단한 접근 방식(특히 다른 목적으로 쿡북을 구현할 필요가 없는 경우)은 Chef 배포 후크를 사용하여 사용자 지정 코드를 실행하는 것입니다. 또한 사용자 지정 Deploy 레시피는 내장 레시피가 배포를 이미 완료한 후에 실행됩니다. 배포 후크를 사용하면 배포 도중(예를 들어 앱의 코드가 리포지토리에서 체크아웃되었지만 Apache가 재시작되기 전) 상호 작용이 가능합니다.

Chef는 앱을 4개 단계로 배포합니다.

- 체크아웃 - 리포지토리에서 파일을 다운로드합니다.
- 마이그레이션 -필요에 따라 마이그레이션을 실행합니다.
- Symlink - symlink를 생성합니다.
- 재시작 - 애플리케이션을 다시 시작합니다.

Chef 배포 후크는 각 단계 완료 후 선택적으로 사용자 제공 Ruby 애플리케이션을 실행하여 배포를 간편하게 사용자 지정할 수 있는 방법을 제공합니다. 배포 후크를 사용하려면 Ruby 애플리케이션을 하나

이상 구현하여 앱의 /deploy 디렉터리에 배치합니다. (앱에 /deploy 디렉터리가 없으면 APP_ROOT 수준에서 디렉터리를 생성하세요.) 애플리케이션은 다음 이름 중 하나를 가져야 합니다. 이 이름은 실행 시점을 결정합니다.

- before_migrate.rb는 체크아웃 단계 완료 후, 마이그레이션 단계 전에 실행됩니다.
- before_symlink.rb는 마이그레이션 단계 완료 후, Symlink 단계 전에 실행됩니다.
- before_restart.rb는 Symlink 단계 완료 후, 재시작 단계 전에 실행됩니다.
- after_restart.rb는 재시작 단계가 완료된 후에 실행됩니다.

Chef 배포 후크는 레시피와 마찬가지로 표준 노드 구문을 사용하여 노드 객체에 액세스할 수 있습니다. 또한 배포 후크는 사용자가 지정한 모든 앱 환경 변수의 값에 액세스할 수 있습니다. 그러나 ENV["*VARIABLE_NAME*"] 대신 new_resource.environment["*VARIABLE_NAME*"] 을 사용하여 변수 값에 액세스해야 합니다.

Linux 인스턴스에서 Cron 작업 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Linux cron 작업은 cron 데몬이 지정된 일정으로 하나 이상의 명령을 실행하도록 지시합니다. 예를 들어 스택이 PHP 전자 상거래 애플리케이션을 지원한다고 가정하겠습니다. 서버가 매주 지정된 시간에 판매 보고서를 전송하도록 cron 작업을 설정할 수 있습니다. Cron에 대한 자세한 내용은 Wikipedia에서 cron을 참조하세요. Linux 기반 컴퓨터 또는 인스턴스에서 cron을 직접 실행하는 방법에 대한 자세한 정보는 Indiana University 지식 기반 웹 사이트에서 What are cron and crontab, and how do I use them? 단원을 참조하세요.

SSH로 cron 작업에 연결하고 crontab 항목을 편집하여 개별 Linux 기반 인스턴스에서 cron 작업을 수동으로 설정할 수는 있지만, AWS OpsWorks Stacks의 주된 장점 중 하나는 전체 인스턴스 계층에서 작업을 실행하도록 지시할 수 있다는 점입니다. 다음 절차에서는 PHP 앱 서버 계층의 인스턴스에서 cron 작업을 설정하는 방법을 설명하지만, 어떤 계층에서도 동일한 접근 방식을 사용할 수 있습니다.

계층의 인스턴스에 대한 **cron** 작업을 설정하려면

1. 이 작업을 설정하는 cron 리소스가 포함된 레시피를 사용하여 쿡북을 구현합니다. 이 예제에서는 레시피의 이름이 `cronjob.rb`라고 가정하고, 자세한 구현 정보는 뒤에서 설명합니다. 쿡북 및 레시피에 대한 자세한 내용은 쿡북과 레시피 단원을 참조하세요.
2. 스택에 쿡북을 설치합니다. 자세한 내용은 사용자 지정 쿡북 설치 단원을 참조하십시오.
3. AWS OpsWorks Stacks가 다음 수명 주기 이벤트에 할당하여 계층의 인스턴스에서 레시피를 자동으로 실행하도록 합니다. 자세한 내용은 자동으로 레시피 실행 단원을 참조하십시오.

   - 설정 -이 이벤트에 할당`cronjob.rb`하면 AWS OpsWorks Stacks가 모든 새 인스턴스에서 레시피를 실행하도록 지시합니다.
   - 배포 - `cronjob.rb`이 이벤트에 할당하면 앱을 계층에 배포하거나 재배포할 때 AWS OpsWorks Stacks가 모든 온라인 인스턴스에서 레시피를 실행하도록 지시합니다.

   `Execute Recipes` 스택 명령을 사용하여 온라인 인스턴스에서 레시피를 수동으로 실행할 수도 있습니다. 자세한 내용은 스택 명령 실행 섹션을 참조하세요.

다음은 서버로부터 판매 데이터를 수집하여 보고서를 전송하는 사용자 구현 PHP 애플리케이션을 매주 한 번 실행하도록 cron 작업을 설정하는 `cronjob.rb` 예제입니다. Cron 리소스를 사용하는 방법의 추가 예제는 cron을 참조하세요.

```
cron "job_name" do
  hour "1"
  minute "10"
  weekday "6"
  command "cd /srv/www/myapp/current && php .lib/mailing.php"
end
```

`cron`은 cron 작업을 나타내는 Chef 리소스입니다. AWS OpsWorks Stacks가 인스턴스에서 레시피를 실행하면 연결된 공급자가 작업 설정의 세부 정보를 처리합니다.

- *`job_name`*은 cron 작업의 사용자 정의 이름입니다(예: `weekly report`).
- `hour`/`minute`/`weekday`는 명령을 실행할 시점을 지정합니다. 이 예제는 명령을 매주 토요일 오전 1시 10분에 실행합니다.
- `command`는 실행할 명령을 지정합니다.

이 예제는 2개의 명령을 실행합니다. 첫 번째 명령은 `/srv/www/myapp/current` 디렉터리로 이동합니다. 두 번째 명령은 판매 데이터를 수집하고 보고서를 전송하는 사용자 구현 `mailing.php` 애플리케이션을 실행합니다.

> **Note**
>
> `bundle` 명령은 기본적으로 `cron` 작업에 사용되지 않습니다. 그 이유는 AWS OpsWorks Stacks가 `/usr/local/bin` 디렉터리에 번들러를 설치하기 때문입니다. `bundle` 작업에서 `cron`을 사용하려면 명시적으로 `/usr/local/bin` 경로를 cron 작업에 추가해야 합니다. 또한 `cron` 작업에서 $PATH 환경 변수가 확장되지 않을 수 있으므로, $PATH 변수의 확장에 의존하지 않고 필요한 경로 정보를 모두 명시적으로 추가하는 것이 모범 사례입니다. 다음 예제는 `cron` 작업에서 `bundle`을 사용하는 두 가지 방법을 보여줍니다.
>
> ```
> cron "my first task" do
>   path "/usr/local/bin"
>   minute "*/10"
>   command "cd /srv/www/myapp/current && bundle exec my_command"
> end
> ```
>
> ```
> cron_env = {"PATH" => "/usr/local/bin"}
> cron "my second task" do
>   environment cron_env
>   minute "*/10"
>   command "cd /srv/www/myapp/current && /usr/local/bin/bundle exec my_command"
> end
> ```

스택에 여러 애플리케이션 서버가 있는 경우 PHP 앱 서버 계층의 수명 주기 이벤트에 `cronjob.rb`를 할당하는 접근 방식은 이상적이지 않을 수 있습니다. 예를 들어 레시피가 계층의 모든 인스턴스에서 실행되므로 여러 보고서가 전송됩니다. 이보다 나은 접근 방식은 사용자 지정 계층을 사용하여 한 서버만 보고서를 전송하도록 하는 것입니다.

계층의 인스턴스 중 하나에서만 레시피를 실행하려면

1. 예를 들어, PHPAdmin이라는 사용자 지정 계층을 만들고 이 계층의 설정 및 Deploy 이벤트에 `cronjob.rb`를 할당합니다. 사용자 지정 계층에서 반드시 많은 레시피를 실행할 필요는 없습니다. 이 경우, PHPAdmin은 인스턴스에서 사용자 지정 레시피 하나를 실행하기만 하면 됩니다.
2. PHP 앱 서버 인스턴스 중 하나를 AdminLayer에 할당합니다. 인스턴스가 둘 이상의 계층에 속하는 경우 AWS OpsWorks Stacks는 각 계층의 내장 및 사용자 지정 레시피를 실행합니다.

인스턴스 하나만 PHP 앱 서버 및 PHPAdmin 계층에 속하므로 `cronjob.rb`는 해당 인스턴스에서만 실행되고, 따라서 하나의 보고서만 전송됩니다.

Linux 인스턴스에서 패키지 설치 및 구성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

내장 계층은 특정 패키지만 지원합니다. 자세한 내용은 계층 섹션을 참조하세요. 연결된 설정, 구성 및 배포 작업을 처리할 사용자 지정 레시피를 구현하여 Redis 서버와 같은 다른 패키지를 설치할 수 있습니다. 경우에 따라서는 내장 계층을 확장하여 패키지를 계층의 표준 패키지와 함께 인스턴스에 설치하도록 하는 것이 최선의 접근 방식입니다. 예를 들어 PHP 애플리케이션을 지원하는 스택이 있고 Redis 서버를 포함시키기를 원할 경우 PHP 앱 서버 계층을 확장하고 계층의 인스턴스에서 PHP 애플리케이션 서버 이외에 Redis 서버를 구성할 수 있습니다.

일반적으로 패키지 설치 레시피는 다음과 같은 작업을 수행합니다.

- 하나 이상의 디렉터리를 생성하고 각각 모드를 설정
- 템플릿에서 구성 파일을 생성
- 설치 관리자를 실행하여 인스턴스에 패키지를 설치
- 하나 이상의 서비스를 시작

Tomcat 서버를 설치하는 방법의 예제는 사용자 지정 Tomcat 서버 계층 생성 단원을 참조하세요. 이 항목에서는 사용자 지정 Redis 계층을 설정하는 방법을 설명하지만, 동일한 코드를 사용하여 내장 계

층에서 Redis를 설치하고 구성할 수 있습니다. 다른 패키지를 설치하는 방법에 대한 예는 내장된 쿡북 (https://github.com/aws/opsworks-cookbooks)을 참조하세요.

사용자 지정 Tomcat 서버 계층 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 항목에서는 Linux 스택용 사용자 지정 계층을 구성하는 방법을 설명합니다. 하지만 기본 원칙 및 일부 코드(특히 앱 배포 섹션)를 수정하면 Windows 스택용 사용자 지정 계층도 구현할 수 있습니다.

AWS OpsWorks Stacks 인스턴스에서 비표준 패키지를 사용하는 가장 간단한 방법은 기존 계층을 확장하는 것입니다. 하지만 이 접근 방식은 계층이 인스턴스에 표준 및 비표준 패키지를 모두 설치하고 실행하므로 항상 바람직한 것은 아닙니다. 약간 까다롭지만 더 강력한 접근 방식은 사용자 지정 계층을 구현하는 것입니다. 이는 다음을 비롯해 계층의 인스턴스에 대해 거의 완벽한 제어를 제공합니다.

- 설치될 패키지
- 패키지가 구성되는 방법
- 리포지토리에서 인스턴스로 앱을 배포하는 방법

콘솔 또는 API를 사용하여 사용자 지정 계층 섹션에 설명된 대로 다른 계층과 거의 똑같이 사용자 지정 계층을 생성하고 관리합니다. 하지만 사용자 지정 계층의 내장 레시피는 Ganglia 클라이언트를 설치하여 Ganglia 마스터로 측정치를 보고하는 것과 같은 매우 기본적인 일부 작업만 수행합니다. 사용자 지정 계층의 인스턴스가 최소 기능 이상을 발휘하게 하려면 패키지 설치와 구성, 앱 배포 등의 작업을 처리하도록 Chef 레시피 및 관련 파일을 포함하는 하나 이상의 사용자 지정 쿡북을 구현해야 합니다. 하지만 모든 것을 처음부터 구현해야 하는 것은 아닙니다. 예를 들어 애플리케이션을 표준 리포지토리 중 하나에 저장하는 경우 내장 deploy 레시피를 사용하여 계층의 인스턴스에 애플리케이션을 설치하는 작업을 대부분 처리할 수 있습니다.

> **Note**
>
> Chef를 처음 사용하는 경우 먼저 다양한 일반적인 작업을 수행하기 위한 쿡북을 구현하는 기본적인 방법을 소개하는 쿡북 101 섹션을 읽어야 합니다.

다음 안내서에서는 Tomcat 애플리케이션 서버를 지원하는 사용자 지정 계층을 구현하는 방법을 설명합니다. 이 계층은 Tomcat이라는 사용자 지정 쿡북을 기반으로 하며, 패키지 설치, 배포 등을 처리하는 레시피를 포함합니다. 이 안내서는 Tomcat 쿡북에서 발췌된 코드를 포함하고 있습니다. 전체 쿡북은 GitHub 리포지토리에서 다운로드할 수 있습니다. Opscode Chef에 대해 잘 알지 못하는 경우 먼저 쿡북과 레시피 섹션을 읽어야 합니다.

> **Note**
>
> AWS OpsWorks Stacks에는 프로덕션용 전체 기능을 갖춘 Java 앱 서버 계층이 포함되어 있습니다. Tomcat 쿡북은 사용자 지정 계층을 구현하는 방법을 예시하는 것이 목적이므로 SSL과 같은 기능을 포함하지 않는 제한적 버전의 Tomcat만 지원합니다. 전체 기능을 제공하는 구현의 예는 내장 opsworks_java 쿡북 단원을 참조하세요.

Tomcat 쿡북은 인스턴스가 다음과 같은 특성을 갖는 사용자 지정 계층을 지원합니다.

- Apache 프런트 엔드를 포함하는 Tomcat Java 애플리케이션 서버를 지원함
- 애플리케이션이 JDBC `DataSource` 객체를 사용하여 백 엔드 데이터 스토어로 사용되는 별도의 MySQL 인스턴스에 연결하는 것을 허용하도록 Tomcat이 구성됨

이 프로젝트의 쿡북에는 다수의 주요 구성 요소가 포함됩니다.

- 속성 파일은 다양한 레시피가 사용하는 구성 파일을 포함하고 있습니다.
- 설정 레시피는 계층의 설정 수명 주기 이벤트에 할당됩니다. 이 레시피는 인스턴스 부팅 후 실행되며 패키지 설치, 구성 파일 생성과 같은 작업을 수행합니다.
- Configure 레시피는 계층의 Configure 수명 주기 이벤트에 할당됩니다. 스택의 구성이 변경된 후(주로 인스턴스가 온라인 상태가 되거나 오프라인 상태가 될 때) 실행되며 필요한 구성 변경을 모두 처리합니다.
- Deploy 레시피는 계층의 Deploy 수명 주기 이벤트에 할당됩니다. 이 레시피는 설정 레시피 이후 또한 사용자가 수동으로 앱을 배포하여 코드 및 관련 파일을 계층의 인스턴스에 설치할 때 실행되며, 서비스 재시작과 같은 관련 작업을 처리합니다.

마지막 섹션 스택 생성 및 애플리케이션 실행에서는 Tomcat 쿡북을 기반으로 한 사용자 지정 계층을 포함하는 스택을 생성하는 방법과 별도의 MySQL 계층에 속하는 인스턴스에서 실행되는 MySQL 데이터베이스로부터 데이터를 표시하는 간단한 JSP 애플리케이션을 배포하고 실행하는 방법을 설명합니다.

> **Note**
>
> Tomcat 쿡북 레시피는 일부 AWS OpsWorks Stacks 내장 레시피에 따라 달라집니다. 각 레시피의 출처를 명확히 하기 위해 이 항목에서는 Chef cookbookname::recipename 규칙을 사용하여 레시피를 식별합니다.

주제

- 속성 파일
- 설정 레시피
- Configure 레시피
- Deploy 레시피
- 스택 생성 및 애플리케이션 실행

속성 파일

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피에 대해 알아보기 전에 먼저 Tomcat 쿡북의 속성 파일을 살펴보는 것이 유용합니다. 이 파일에는 레시피가 사용하는 다양한 구성 설정이 들어 있습니다. 속성은 필수는 아닙니다. 레시피 또는 템플릿에서 이들 값을 하드코딩할 수도 있습니다. 그러나 속성을 사용하여 구성 설정을 정의하는 경우 AWS OpsWorks Stacks 콘솔 또는 API를 사용하여 설정을 변경할 때마다 레시피 또는 템플릿 코드를 다시 작성하는 것보다 더 간단하고 유연한 사용자 지정 JSON 속성을 정의하여 값을 수정할 수 있습니다. 이 접근 방식을 사용하면 예를 들어 여러 스택에 동일한 쿡북을 사용하되 각 스택마다 Tomcat 서버

를 다르게 구성할 수 있습니다. 속성과 속성 재정의 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

다음 예제는 Tomcat 쿡북의 `default.rb` 디렉터리에 위치하는 전체 속성 파일 `attributes`입니다.

```
default['tomcat']['base_version'] = 6
default['tomcat']['port'] = 8080
default['tomcat']['secure_port'] = 8443
default['tomcat']['ajp_port'] = 8009
default['tomcat']['shutdown_port'] = 8005
default['tomcat']['uri_encoding'] = 'UTF-8'
default['tomcat']['unpack_wars'] = true
default['tomcat']['auto_deploy'] = true
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['java_opts'] = ''
when 'debian', 'ubuntu'
  default['tomcat']['java_opts'] = '-Djava.awt.headless=true -Xmx128m -XX:
+UseConcMarkSweepGC'
end
default['tomcat']['catalina_base_dir'] = "/etc/tomcat#{node['tomcat']['base_version']}"
default['tomcat']['webapps_base_dir'] = "/var/lib/tomcat#{node['tomcat']
['base_version']}/webapps"
default['tomcat']['lib_dir'] = "/usr/share/tomcat#{node['tomcat']['base_version']}/lib"
default['tomcat']['java_dir'] = '/usr/share/java'
default['tomcat']['mysql_connector_jar'] = 'mysql-connector-java.jar'
default['tomcat']['apache_tomcat_bind_mod'] = 'proxy_http' # or: 'proxy_ajp'
default['tomcat']['apache_tomcat_bind_config'] = 'tomcat_bind.conf'
default['tomcat']['apache_tomcat_bind_path'] = '/tc/'
default['tomcat']['webapps_dir_entries_to_delete'] = %w(config log public tmp)
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['user'] = 'tomcat'
  default['tomcat']['group'] = 'tomcat'
  default['tomcat']['system_env_dir'] = '/etc/sysconfig'
when 'debian', 'ubuntu'
  default['tomcat']['user'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['group'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['system_env_dir'] = '/etc/default'
end
```

설정 자체에 대해서는 나중에 관련 섹션에서 설명합니다. 다음 참고 사항은 전반적으로 적용됩니다.

- 노드 정의는 모두 `default` 유형이므로 사용자 지정 JSON 속성으로 재정의할 수 있습니다.
- 파일은 `case` 문을 사용하여 인스턴스의 운영 체제에 따라 일부 속성 값을 조건적으로 설정합니다.

  `platform` 노드는 Chef의 Ohai 도구에 의해 생성되며 인스턴스의 운영 체제를 표시합니다.

설정 레시피

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

설정 레시피는 계층의 설정 수명 주기 이벤트에 할당되고 인스턴스 부팅 후 실행됩니다. 이 레시피는 패키지 설치, 구성 파일 생성, 서비스 시작과 같은 작업을 수행합니다. Setup 레시피 실행이 완료되면 AWS OpsWorks Stacks는 Deploy 레시피를 실행하여 앱을 새 인스턴스에 배포합니다.

주제

- tomcat::설정
- tomcat::install
- tomcat::service
- tomcat::container_config
- tomcat::apache_tomcat_bind

tomcat::설정

`tomcat::setup` 레시피는 계층의 설정 수명 주기 이벤트에 할당됩니다.

```
include_recipe 'tomcat::install'
include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end
```

```
# for EBS-backed instances we rely on autofs
bash '(re-)start autofs earlier' do
  user 'root'
  code <<-EOC
    service autofs restart
  EOC
  notifies :restart, resources(:service => 'tomcat')
end

include_recipe 'tomcat::container_config'
include_recipe 'apache2'
include_recipe 'tomcat::apache_tomcat_bind'
```

`tomcat::setup` 레시피는 대체로 메타 레시피입니다. Tomcat 및 관련 패키지를 설치하고 구성하는 세부 작업을 대부분 처리하는 일련의 종속 레시피가 포함됩니다. `tomcat::setup`의 첫 번째 부분은 다음 레시피를 실행합니다(각 레시피에 대해서는 나중에 설명).

- tomcat::install 레시피는 Tomcat 서버 패키지를 설치합니다.
- tomcat::service 레시피는 Tomcat 서비스를 설정합니다.

`tomcat::setup`의 중간 부분은 Tomcat 서비스를 활성화하고 시작합니다.

- Chef 서비스 리소스는 부팅 시 Tomcat 서비스를 활성화합니다.
- Chef bash 리소스는 Bash 스크립트를 실행하여 Amazon EBS 지원 인스턴스에 필요한 autofs 대몬(daemon)을 시작합니다. 그런 다음 이 리소스는 `service` 리소스에 Tomcat 서비스를 재시작하라고 알립니다.

  자세한 정보는 autofs(Amazon Linux) 또는 Autofs(Ubuntu)를 참조하세요.

`tomcat::setup`의 마지막 부분은 구성 파일을 생성하고 프런트 엔드 Apache 서버를 설치하고 구성합니다.

- tomcat::container_config 레시피는 구성 파일을 생성합니다.
- `apache2` 레시피(의 약어`apache2::default`)는 Apache 서버를 설치하고 구성하는 AWS OpsWorks Stacks 기본 제공 레시피입니다.
- tomcat::apache_tomcat_bind 레시피는 Apache 서버가 Tomcat 서버의 프런트 엔드로 기능하도록 구성합니다.

> **Note**
>
> 내장 레시피를 사용하여 일부 필요한 작업을 수행하면 시간과 노력을 아낄 수 있는 경우가 자주 있습니다. 이 레시피는 처음부터 Apache를 구현하는 것이 아니라 내장 `apache2::default` 레시피를 사용하여 Apache를 설치합니다. 내장 레시피를 사용하는 방법의 또 하나의 예는 Deploy 레시피 단원을 참조하세요.

다음 섹션에서는 Tomcat 쿡북의 설정 레시피에 대해 보다 자세히 설명합니다. `apache2` 레시피에 대한 자세한 정보는 opsworks-cookbooks/apache2를 참조하세요.

tomcat::install

`tomcat::install` 레시피는 Tomcat 서버, OpenJDK, 그리고 MySQL 서버 연결을 처리하는 Java 커넥터 라이브러리를 설치합니다.

```
tomcat_pkgs = value_for_platform(
  ['debian', 'ubuntu'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'libtcnative-1',
 'libmysql-java']
  },
  ['centos', 'redhat', 'fedora', 'amazon'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'tomcat-native', 'mysql-
connector-java']
  },
  'default' => ["tomcat#{node['tomcat']['base_version']}"]
)

tomcat_pkgs.each do |pkg|
  package pkg do
    action :install
  end
end

link ::File.join(node['tomcat']['lib_dir'], node['tomcat']['mysql_connector_jar']) do
  to ::File.join(node['tomcat']['java_dir'], node['tomcat']['mysql_connector_jar'])
  action :create
end

# remove the ROOT webapp, if it got installed by default
```

```
include_recipe 'tomcat::remove_root_webapp'
```

이 레시피가 수행하는 작업은 다음과 같습니다.

1. 인스턴스의 운영 체제에 따라 설치할 패키지의 목록을 생성합니다.
2. 목록의 각 패키지를 설치합니다.

   Chef 패키지 리소스는 Amazon Linux의 경우 yum와 Ubuntu의 경우 apt-get인 적절한 공급자를 사용하여 설치를 처리합니다. 패키지 공급자가 OpenJDK를 Tomcat 종속성으로 설치하지만, MySQL 커넥터 라이브러리는 명시적으로 설치되어야 합니다.
3. Chef 링크 리소스를 사용하여 Tomcat 서버의 라이브러리 디렉터리에서 JDK 내 MySQL 커넥터 라이브러리에 대한 symlink를 생성합니다.

   기본 속성 값을 사용할 경우 Tomcat 라이브러리 디렉터리는 /usr/share/tomcat6/lib이고, MySQL 커넥터 라이브러리(mysql-connector-java.jar)는 /usr/share/java/에 위치합니다.

tomcat::remove_root_webapp 레시피는 ROOT 웹 애플리케이션(기본적으로 /var/lib/tomcat6/webapps/ROOT)을 제거하여 일부 보안 문제를 방지합니다.

```
ruby_block 'remove the ROOT webapp' do
  block do
    ::FileUtils.rm_rf(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'), :secure
 => true)
  end
  only_if { ::File.exists?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'))
 && !::File.symlink?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT')) }
end
```

only_if 문은 파일이 존재하는 경우에만 레시피가 파일을 제거하도록 합니다.

> **Note**
>
> Tomcat 버전은 ['tomcat']['base_version'] 속성에 의해 지정되는데, 속성 파일에서 6으로 설정되어 있습니다. Tomcat 7을 설치하려면 사용자 지정 JSON 속성을 사용하여 속성을 재정의할 수 있습니다. 스택 설정을 편집하고 다음 JSON을 [사용자 지정 Chef JSON] 상자에 입력하거나 기존의 사용자 지정 JSON에 추가하면 됩니다.

```
{
  'tomcat' : {
    'base_version' : 7
  }
}
```

사용자 지정 JSON 속성은 기본 속성을 재정의하고 Tomcat 버전을 7로 설정합니다. 속성 재정의에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

tomcat::service

`tomcat::service` 레시피는 Tomcat 서비스 정의를 생성합니다.

```
service 'tomcat' do
  service_name "tomcat#{node['tomcat']['base_version']}"

  case node[:platform]
  when 'centos', 'redhat', 'fedora', 'amazon'
    supports :restart => true, :reload => true, :status => true
  when 'debian', 'ubuntu'
    supports :restart => true, :reload => false, :status => true
  end

  action :nothing
end
```

이 레시피는 Chef 서비스 리소스를 사용하여 Tomcat 서비스 이름(기본적으로 tomcat6)을 지정하고 `supports` 속성을 설정하여 Chef가 운영 체제별로 서비스의 restart, reload 및 status 명령을 관리하는 방법을 정의합니다.

- `true`는 Chef가 init 스크립트 또는 그 밖의 서비스 공급자를 사용하여 명령을 실행할 수 있음을 나타냅니다.
- `false`는 Chef가 다른 수단을 사용하여 명령을 실행하려 시도해야 함을 나타냅니다.

`action`은 `:nothing`로 설정됩니다. 각 수명 주기 이벤트에 대해 AWS OpsWorks Stacks는 Chef 실행을 시작하여 적절한 레시피 세트를 실행합니다. Tomcat 쿡북은 레시피가 서비스 정의를 생성하지만 서비스를 재시작하지는 않는 일반 패턴을 따릅니다. Chef 실행의 다른 레시피가 일반적으로 구성 파일을 생성하는 데 사용된 `notifies` 리소스에 `template` 명령을 포함시켜 재시작을 처리합니다. 알림은

서비스를 재시작하는 편리한 방법입니다. 구성이 변경된 경우에만 서비스를 재시작하기 때문입니다. 또한 Chef 실행에 특정 서비스에 대한 재시작 알림이 여러 개일 경우 Chef는 서비스를 한 번만 재시작합니다. 이 방법은 완전히 작동하지 않는 서비스를 재시작하려고 시도할 때 발생할 수 있는 문제를 방지하기 위한 것으로, 이 문제는 Tomcat 오류의 일반적인 원인입니다.

Tomcat 서비스는 재시작 알림을 사용하는 모든 Chef 실행에 대해 정의되어야 합니다. `tomcat::service`는 여러 레시피에 포함되어 서비스가 모든 Chef 실행에 대해 정의되도록 합니다. Chef 실행에 `tomcat::service` 인스턴스가 여러 번 포함되더라도 Chef는 포함 횟수와 상관없이 Chef 실행당 한 번만 레시피를 실행하므로 페널티는 없습니다.

tomcat::container_config

`tomcat::container_config` 레시피는 쿡북 템플릿 파일에서 구성 파일을 생성합니다.

```
include_recipe 'tomcat::service'

template 'tomcat environment configuration' do
  path ::File.join(node['tomcat']['system_env_dir'], "tomcat#{node['tomcat']
['base_version']}")
  source 'tomcat_env_config.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end

template 'tomcat server configuration' do
  path ::File.join(node['tomcat']['catalina_base_dir'], 'server.xml')
  source 'server.xml.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end
```

이 레시피는 먼저 `tomcat::service`를 호출합니다. 필요할 경우 이 레시피가 서비스를 정의합니다. 레시피의 대부분은 2개의 템플릿 리소스로 구성되는데, 각각 쿡북의 템플릿 파일 중 하나에서 구성 파일을 생성하고, 파일 속성을 설정하고, Chef에 서비스를 재시작하라고 알립니다.

Tomcat 환경 구성 파일

첫 번째 `template` 리소스는 `tomcat_env_config.erb` 템플릿 파일을 사용하여 Tomcat 환경 구성 파일을 생성합니다. 이 파일은 `JAVA_HOME`과 같은 환경 변수를 설정하는 데 사용됩니다. 기본 파일 이름은 `template` 리소스의 인수입니다. `tomcat::container_config`는 `path` 속성을 사용하여 기본값을 재정의하고 구성 파일을 `/etc/sysconfig/tomcat6`(Amazon Linux) 또는 `/etc/default/tomcat6` (Ubuntu)로 명명합니다. 또한 `template` 리소스는 파일의 소유자, 그룹 및 모드 설정을 지정하고 Chef에게 백업 파일을 생성하지 않도록 지시합니다.

소스 코드를 보면 실제로 세 버전의 `tomcat_env_config.erb`가 각각 `templates` 디렉터리의 서로 다른 하위 디렉터리에 위치합니다. `ubuntu` 및 `amazon` 디렉터리에는 해당 운영 체제용 템플릿이 들어 있습니다. `default` 폴더에는 명령줄 하나로 구성된 더미 템플릿이 들어 있습니다. 이 템플릿은 지원되지 않는 운영 체제를 사용하는 인스턴스에서 이 레시피를 실행하려고 시도할 경우에만 사용됩니다. `tomcat::container_config` 레시피는 사용할 `tomcat_env_config.erb`를 지정할 필요가 없습니다. Chef가 File Specificity에 기술된 규칙에 따라 인스턴스의 운영 체제에 적합한 디렉터리를 선택합니다.

이 예제의 `tomcat_env_config.erb` 파일은 대부분 주석으로 구성되어 있습니다. 추가 환경 변수를 설정하려면 해당 줄의 주석 처리를 해제하고 원하는 값을 제공하면 됩니다.

> **Note**
>
> 변경될 수 있는 구성 설정은 템플릿에서 하드코딩하는 것보다는 속성으로 정의해야 합니다. 그러면 설정을 변경하기 위해 템플릿을 재작성할 필요 없이 속성을 재정의하기만 하면 됩니다.

Amazon Linux 템플릿은 다음 코드에 나와 있듯이 하나의 환경 변수만 설정합니다.

```
...
# Use JAVA_OPTS to set java.library.path for libtcnative.so
#JAVA_OPTS="-Djava.library.path=/usr/lib"

JAVA_OPTS="${JAVA_OPTS} <%= node['tomcat']['java_opts'] %>"

# What user should run tomcat
#TOMCAT_USER="tomcat"
...
```

JAVA_OPTS를 사용하여 라이브러리 경로와 같은 Java 옵션을 지정할 수 있습니다. 기본 속성 값을 사용할 경우 템플릿은 Amazon Linux에 대해 어떤 Java 옵션도 설정하지 않습니다. `['tomcat']['java_opts']` 속성을 재정의하여(예를 들어 사용자 지정 JSON 속성을 사용하여) 자체 Java 옵션을 설정할 수 있습니다. 예제는 스택 생성 섹션을 참조하세요.

Ubuntu 템플릿은 다음 템플릿 코드에 나와 있듯이 여러 환경 변수를 설정합니다.

```
# Run Tomcat as this user ID. Not setting this or leaving it blank will use the
# default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_USER=tomcat<%= node['tomcat']
['base_version'] %>
...
# Run Tomcat as this group ID. Not setting this or leaving it blank will use
# the default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_GROUP=tomcat<%= node['tomcat']
['base_version'] %>
...
JAVA_OPTS="<%= node['tomcat']['java_opts'] %>"

<% if node['tomcat']['base_version'].to_i < 7 -%>
# Unset LC_ALL to prevent user environment executing the init script from
# influencing servlet behavior.  See Debian bug #645221
unset LC_ALL
<% end -%>
```

기본 속성 값을 사용하면 템플릿은 다음과 같이 Ubuntu 환경 변수를 설정합니다.

- `TOMCAT6_USER` 및 `TOMCAT6_GROUP`(Tomcat 사용자 및 그룹을 표시)이 모두 `tomcat6`으로 설정됩니다.

  ['tomcat']['base_version']을 `tomcat7`로 설정할 경우 변수 이름이 `TOMCAT7_USER` 및 `TOMCAT7_GROUP`로 확인되고, 모두 `tomcat7`로 설정됩니다.
- `JAVA_OPTS`가 `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC`로 설정됩니다.
  - `-Djava.awt.headless`를 `true`로 설정하면 그래픽 엔진에게 인스턴스가 헤드리스이고 콘솔이 없음을 알려줍니다. 이는 일부 그래픽 애플리케이션의 동작 오류를 해결합니다.
  - `-Xmx128m`은 JVM이 적절한 메모리 리소스를 사용하도록 지정합니다(이 예제에서는 128MB).
  - `-XX:+UseConcMarkSweepGC`는 동시 마크 스윕 가비지 수집을 지정합니다. 이는 가비지 수집으로 유발되는 일시 중지를 제한하는 데 도움이 됩니다.

자세한 정보는 Concurrent Mark Sweep Collector Enhancements를 참조하세요.

- Tomcat 버전이 7 미만일 경우 템플릿이 LC_ALL을 설정 해제합니다. 이는 Ubuntu 버그를 해결합니다.

> **Note**
>
> 기본 속성을 사용할 경우 이러한 환경 변수 중 일부가 기본값으로 설정됩니다. 하지만 명시적으로 환경 변수를 속성으로 설정하면 사용자 지정 JSON 속성을 정의하여 기본 속성을 재정의하고 사용자 지정 값을 제공할 수 있습니다. 속성 재정의에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

전체 템플릿 파일은 source code를 참조하세요.

Server.xml 구성 파일

두 번째 `template` 리소스는 `server.xml.erb`를 사용하여 서블릿/JSP 컨테이너를 구성하는 `system.xml` 구성 파일을 생성합니다. `server.xml.erb`에는 운영 체제별 설정이 없으므로 `template` 디렉터리의 `default` 하위 디렉터리에 위치합니다.

템플릿은 표준 설정을 사용하지만 Tomcat 6 또는 Tomcat 7에 대해 `system.xml` 파일을 생성할 수 있습니다. 예를 들어 템플릿의 서버 섹션에서 발췌된 다음 코드는 리스너를 지정된 버전에 따라 적절하게 구성합니다.

```
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
<% end -%>
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-
howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<% if node['tomcat']['base_version'].to_i < 7 -%>
  <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
```

```
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
<% end -%>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
<% end -%>
```

템플릿은 하드코딩된 설정 대신 속성을 사용합니다. 따라서 사용자 지정 JSON 속성을 정의하여 간편하게 설정을 변경할 수 있습니다. 예시:

```
<Connector port="<%= node['tomcat']['port'] %>" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="<%= node['tomcat']['uri_encoding'] %>"
           redirectPort="<%= node['tomcat']['secure_port'] %>" />
```

자세한 정보는 source code를 참조하세요.

tomcat::apache_tomcat_bind

`tomcat::apache_tomcat_bind` 레시피는 Apache 서버가 Tomcat의 프런트 엔드로 동작하여 수신 요청을 수신하고, 요청을 Tomcat으로 전달하고, 응답을 클라이언트로 반환하도록 설정합니다. 이 예제에서는 mod_proxy를 Apache 프록시/게이트웨이로 사용합니다.

```
execute 'enable mod_proxy for apache-tomcat binding' do
  command '/usr/sbin/a2enmod proxy'
  not_if do
    ::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled', 'proxy.load'))
 || node['tomcat']['apache_tomcat_bind_mod'] !~ /\Aproxy/
  end
end

execute 'enable module for apache-tomcat binding' do
  command "/usr/sbin/a2enmod #{node['tomcat']['apache_tomcat_bind_mod']}"
  not_if {::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled',
 "#{node['tomcat']['apache_tomcat_bind_mod']}.load"))}
end

include_recipe 'apache2::service'

template 'tomcat thru apache binding' do
```

```
  path ::File.join(node['apache']['dir'], 'conf.d', node['tomcat']
['apache_tomcat_bind_config'])
  source 'apache_tomcat_bind.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'apache2')
end
```

mod_proxy를 활성화하려면 proxy 모듈과 프로토콜 기반 모듈을 활성화해야 합니다. 프로토콜 모듈에는 두 가지 옵션을 사용할 수 있습니다.

- HTTP: proxy_http
- Apache JServ Protocol(AJP): proxy_ajp

  AJP는 내부 Tomcat 프로토콜입니다.

레시피의 두 실행 리소스가 모두 a2enmod 명령을 실행합니다. 따라서 필요한 symlink를 생성하여 지정된 모듈을 활성화할 수 있습니다.

- 첫 번째 execute 리소스는 proxy 모듈을 활성화합니다.
- 두 번째 execute 리소스는 프로토콜 모듈을 활성화합니다. 이 모듈은 기본적으로 proxy_http로 설정됩니다.

  AJP를 사용할 경우 사용자 지정 JSON을 정의하여 apache_tomcat_bind_mod 속성을 재정의하고 proxy_ajp로 설정할 수 있습니다.

apache2::service 레시피는 Apache 서비스를 정의하는 AWS OpsWorks Stacks 기본 제공 레시피입니다. 자세한 내용은 AWS OpsWorks Stacks GitHub 리포지토리의 레시피를 참조하세요.

template 리소스는 apache_tomcat_bind.conf.erb를 사용하여 구성 파일을 생성하며, 이 파일의 기본 이름은 tomcat_bind.conf입니다. 그리고 리소스는 이 구성 파일을 ['apache']['dir']/.conf.d 디렉터리에 저장합니다. ['apache']['dir'] 속성은 내장된 apache2 속성 파일에 정의되어 있으며, 기본적으로 /etc/httpd(Amazon Linux) 또는 /etc/apache2(Ubuntu)로 설정됩니다. template 리소스가 구성 파일을 생성하거나 변경할 경우 notifies 명령이 Apache 서비스 재시작을 예약합니다.

```
<% if node['tomcat']['apache_tomcat_bind_mod'] == 'proxy_ajp' -%>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
<% else %>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
<% end -%>
```

템플릿은 ProxyPass 및 ProxyPassReverse 명령을 사용하여 Apache와 Tomcat 간에 트래픽을 전달하는 데 사용되는 포트를 구성합니다. 두 서버는 동일한 인스턴스에서 실행되므로 localhost URL을 사용할 수 있으며 모두 기본적으로 `http://localhost:8080`으로 설정됩니다.

Configure 레시피

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Configure 레시피는 계층의 Configure 수명 주기 이벤트에 할당됩니다. 이 이벤트는 특정 인스턴스가 온라인 상태로 전환되거나 해제될 때마다 스택의 모든 인스턴스에서 발생합니다. 변경 사항에 대응하여 적절히 인스턴스의 구성을 조정하려면 Configure 레시피를 사용합니다. Configure 레시피를 구현할 때 스택 구성 변경이 이 계층과 관계가 없는 인스턴스도 포함할 수 있다는 점을 염두에 두어야 합니다. 이 레시피는 적절히 대응할 수 있어야 합니다. 이는 경우에 따라 아무 작업도 하지 않음을 의미하기도 합니다.

tomcat::configure

`tomcat::configure` 레시피는 계층의 Configure 수명 주기 이벤트를 위한 것입니다.

```
include_recipe 'tomcat::context'
# Optional: Trigger a Tomcat restart in case of a configure event, if relevant
```

```
# settings in custom JSON have changed (e.g. java_opts/JAVA_OPTS):
#include_recipe 'tomcat::container_config'
```

`tomcat::configure` 레시피는 기본적으로 두 개의 종속 레시피를 실행하는 메타 레시피입니다.

1. `tomcat::context` 레시피는 웹 앱 컨텍스트 구성 파일을 생성합니다.

   이 파일은 애플리케이션이 MySQL 인스턴스와 통신하는 데 사용하는 JDBC 리소스를 구성합니다. 이에 대해서는 다음 섹션에서 설명합니다. Configure 이벤트에 대응하여 이 레시피를 실행할 경우 데이터베이스 계층이 변경되었을 때 웹 앱 컨텍스트 구성 파일을 업데이트할 수 있습니다.
2. `tomcat::container_config` 설정 레시피가 다시 실행되어 컨테이너 구성 변경 사항을 캡처합니다.

이 예제에서는 `include`의 `tomcat::container_config`가 코멘트 아웃됩니다. 사용자 지정 JSON을 사용하여 Tomcat 설정을 수정하려는 경우 이 주석을 제거할 수 있습니다. 그런 다음 Configure 수명 주기 이벤트가 `tomcat::container_config`를 실행합니다. 이는 Tomcat 관련 구성 파일을 업데이트하고(tomcat::container_config 섹션 참조) Tomcat 서비스를 재시작합니다.

tomcat::context

Tomcat 쿡북은 애플리케이션이 J2EE DataSource 객체를 사용하여 별도의 인스턴스에서 실행될 수 있는 MySQL 데이터베이스 서버에 액세스하도록 설정합니다. Tomcat에서는 각 애플리케이션에 대해 웹 앱 컨텍스트 구성 파일을 생성 및 설치하여 연결을 활성화할 수 있습니다. 이 파일은 애플리케이션과 애플리케이션이 데이터베이스와 통신하는 데 사용할 JDBC 리소스 간의 관계를 정의합니다. 자세한 정보는 The Context Container를 참조하세요.

`tomcat::context` 레시피의 주된 목적은 이 구성 파일을 생성하는 것입니다.

```
include_recipe 'tomcat::service'

node[:deploy].each do |application, deploy|
  context_name = deploy[:document_root].blank? ? application : deploy[:document_root]

  template "context file for #{application} (context name: #{context_name})" do
    path ::File.join(node['tomcat']['catalina_base_dir'], 'Catalina', 'localhost',
 "#{context_name}.xml")
    source 'webapp_context.xml.erb'
    owner node['tomcat']['user']
    group node['tomcat']['group']
```

```
    mode 0640
    backup false
    only_if { node['datasources'][context_name] }
    variables(:resource_name => node['datasources'][context_name], :webapp_name =>
 application)
    notifies :restart, resources(:service => 'tomcat')
  end
end
```

Tomcat AWS OpsWorks 쿡북 속성 외에도이 레시피는 Stacks가 구성 이벤트와 함께 설치하는 스택 구성 및 배포 속성을 사용합니다. AWS OpsWorks Stacks 서비스는 레시피가 일반적으로 데이터 백을 사용하거나 각 인스턴스에 속성을 검색하여 얻는 정보를 포함하는 속성을 각 인스턴스의 노드 객체에 추가합니다. 이러한 속성에는 스택 구성, 배포된 앱, 사용자가 포함시키기 원하는 사용자 지정 데이터에 대한 세부 정보가 포함됩니다. 레시피는 표준 Chef 노드 구문을 사용하여 스택 구성 및 배포 속성에서 데이터를 가져옵니다. 자세한 내용은 스택 구성 및 배포 속성 섹션을 참조하세요. Chef 11.10 스택에서는 Chef 검색을 사용하여 스택 구성 및 배포 데이터를 가져올 수도 있습니다. 자세한 내용은 Chef 검색 사용 단원을 참조하십시오.

deploy 속성은 콘솔 또는 API를 통해 정의되거나 AWS OpsWorks Stacks 서비스에 의해 생성되는 배포 관련 속성을 포함하는 [:deploy] 네임스페이스를 나타냅니다. deploy 속성에는 배포된 각 앱마다 앱의 짧은 이름으로 명명된 속성이 하나씩 포함됩니다. 각각의 앱 속성에는 문서 루트([:deploy][:*appname*][:document_root])와 같이 앱의 특성을 정의하는 속성 세트가 포함됩니다.

context 레시피는 먼저 tomcat::service를 호출하여 서비스가 이 Chef 실행에 대해 정의되도록 합니다. 그런 다음 구성 파일의 이름(context_name 확장명 제외)을 표시하는 .xml 변수를 정의합니다. 기본 문서 루트를 사용하는 경우, context_name이 앱의 짧은 이름으로 설정됩니다. 그렇지 않은 경우 지정된 문서 루트로 설정됩니다. 스택 생성 및 애플리케이션 실행 섹션에 설명된 예제는 문서 루트를 "ROOT"로 설정합니다. 따라서 컨텍스트는 ROOT이고 구성 파일 이름은 ROOT.xml입니다.

이 레시피는 배포된 앱의 목록이 많은 부분을 차지하며, 각 앱에 대해 webapp_context.xml.erb 템플릿을 사용하여 컨텍스트 구성 파일을 생성합니다. 예제는 앱을 하나만 배포하지만 deploy 속성의 정의는 앱 수와 상관없이 이를 앱 목록으로 처리하도록 요구합니다.

webapp_context.xml.erb 템플릿은 운영 체제를 구별하지 않으므로 templates 디렉터리의 default 하위 디렉터리에 위치합니다.

이 레시피는 다음과 같이 구성 파일을 생성합니다.

- 기본 속성 값을 사용할 경우 구성 파일 이름은 *context_name*.xml로 설정되고 /etc/tomcat6/Catalina/localhost/ 디렉터리에 설치됩니다.

스택 구성 속성의 ['datasources'] 노드에는 각각 연결된 애플리케이션이 데이터베이스와 통신하는 데 사용할 JDBC 데이터 리소스에 컨텍스트 이름을 매핑하는 속성이 하나 이상 포함됩니다. 노드 및 그 콘텐츠는 스택 생성 및 애플리케이션 실행 섹션에 설명된 대로 스택을 생성할 때 사용자 지정 JSON을 통해 정의됩니다. 예제에는 ROOT 컨텍스트 이름을 jdbc/mydb라는 JDBC 리소스와 연결하는 속성 하나가 있습니다.

- 기본 속성 값을 사용하면 파일의 사용자 및 그룹 모두 Tomcat 패키지에 의해 설정됩니다. tomcat(Amazon Linux) 또는 tomcat6(Ubuntu).
- template 리소스는 ['datasources'] 노드가 존재하고 context_name 속성을 포함하는 경우에만 구성 파일을 생성합니다.
- template 리소스는 resource_name 및 webapp_name, 두 개의 변수를 정의합니다.

  resource_name은 context_name과 연결된 리소스 이름으로 설정되고, webapp_name은 앱의 짧은 이름으로 설정됩니다.
- 템플릿 리소스는 Tomcat 서비스를 재시작하여 변경 사항을 로드하고 활성화합니다.

webapp_context.xml.erb 템플릿은 자체 속성 세트를 갖는 Context 요소를 포함하는 Resource 요소로 구성됩니다.

Resource 속성은 컨텍스트 구성의 특성을 정의합니다.

- name - JDBC 리소스 이름. tomcat::context에서 정의된 resource_name 값으로 설정됩니다.

  예제에서는 리소스 이름이 jdbc/mydb로 설정됩니다.
- auth 및 type - 이들은 JDBC DataSource 연결의 표준 설정입니다.
- maxActive, maxIdle 및 maxWait - 최대 활성 연결 수, 최대 유휴 연결 수 및 연결이 반환되기까지 최대 대기 시간.
- username 및 password - 데이터베이스의 사용자 이름 및 암호. deploy 속성에서 가져옵니다.
- driverClassName - JDBC 드라이버의 클래스 이름. MySQL 드라이버로 설정됩니다.
- url - 연결 URL.

  접두사는 데이터베이스에 따라 다릅니다. MySQL의 경우 jdbc:mysql, Postgres의 경우 jdbc:postgresql, SQL Server의 경우 jdbc:sqlserver로 설정되어야 합니다. 예제에서는 URL을 jdbc:mysql://*host_IP_Address*:3306:simplejsp로 설정합니다. 여기서 *simplejsp*는 앱의 짧은 이름입니다.
- factory - DataSource 팩토리. MySQL 데이터베이스의 경우 필수입니다.

이 구성 파일에 대한 자세한 정보는 Tomcat wiki의 Using DataSources 항목 단원을 참조하세요.

Deploy 레시피

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Deploy 레시피는 계층의 Deploy 수명 주기 이벤트에 할당됩니다. 이는 일반적으로 앱을 배포할 때마다 스택의 모든 인스턴스에서 발생하지만 선택적으로 이벤트를 지정된 인스턴스로만 제한할 수 있습니다. AWS OpsWorks 스택은 설정 레시피가 완료된 후 새 인스턴스에서 배포 레시피도 실행합니다. Deploy 레시피의 주된 목적은 코드와 관련 파일을 리포지토리에서 애플리케이션 서버 계층의 인스턴스에 배포하는 것입니다. 하지만 다른 계층에서도 Deploy 레시피를 실행하는 경우가 종종 있습니다. 그러면 예를 들어 해당 계층의 인스턴스가 구성을 업데이트하여 새로 개발된 앱을 수용할 수 있습니다. Deploy 레시피를 구현할 때 Deploy 이벤트가 반드시 앱이 인스턴스에 배포되는 것을 의미하지는 않음을 염두에 두어야 합니다. 단지 인스턴스가 필요한 업데이트를 할 수 있도록 스택 내 다른 인스턴스에 앱이 배포되고 있다는 알림일 수도 있습니다. 이 레시피는 적절히 대응할 수 있어야 합니다. 이는 아무 작업도 하지 않음을 의미하기도 합니다.

AWS OpsWorks Stacks는 표준 앱 유형의 앱을 해당 내장 애플리케이션 서버 계층에 자동으로 배포합니다. 앱을 사용자 지정 계층에 배포하려면 앱의 파일을 리포지토리에서 인스턴스 내 적절한 위치로 다운로드하는 사용자 지정 Deploy 레시피를 구현해야 합니다. 하지만 내장 배포 쿡북을 사용하여 배포의 일부 측면을 처리함으로써 작성할 코드의 양을 줄일 수도 있습니다. 예를 들어 파일을 지원되는 리포지토리 중 하나에 저장할 경우 내장 쿡북이 리포지토리에서 계층의 인스턴스로 파일을 다운로드하는 세부 작업을 처리할 수 있습니다.

`tomcat::deploy` 레시피는 Deploy 수명 주기 이벤트에 할당됩니다.

```
include_recipe 'deploy'

node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
```

```
  end

  opsworks_deploy do
    deploy_data deploy
    app application
  end
...
```

tomcat::deploy 레시피는 애플리케이션에 고유하지 않은 배포에 대해 내장 배포 쿡북을 사용합니다. deploy 레시피(내장 deploy::default 레시피의 약칭)는 deploy 속성의 데이터를 기반으로 사용자, 그룹 등을 설정하는 세부 작업을 처리하는 내장 레시피입니다.

이 레시피는 두 개의 내장 Chef 정의 opsworks_deploy_dir 및 opworks_deploy를 사용하여 애플리케이션을 설치합니다.

opsworks_deploy_dir 정의는 앱의 배포 JSON에 포함된 데이터를 기반으로 디렉터리 구조를 설정합니다. 정의는 기본적으로 리소스 정의를 패키징하는 편리한 방법으로, definitions 디렉터리에 위치합니다. 레시피는 사용자 정의를 리소스와 거의 비슷하게 사용할 수 있지만, 정의 자체에는 연결된 공급자가 없고 정의에 포함된 리소스만 있습니다. 레시피에서 변수를 정의할 수 있습니다. 그러면 해당 변수가 기본 리소스 정의로 전달됩니다. tomcat::deploy 레시피는 배포 JSON의 데이터를 기반으로 user, group 및 path 변수를 설정합니다. 이들 변수는 정의의 디렉터리를 관리하는 디렉터리 리소스로 전달됩니다.

> **Note**
>
> 배포된 앱의 사용자 및 그룹은 [:opsworks][:deploy_user][:user] 및 [:opsworks][:deploy_user][:group] 속성을 통해 결정되며, 각 속성은 내장 배포 쿡북의 deploy.rb 속성 파일에서 정의됩니다. [:opsworks][:deploy_user][:user]의 기본값은 deploy입니다. [:opsworks][:deploy_user][:group]의 기본값은 인스턴스의 운영 체제에 따라 다릅니다.
>
> - Ubuntu 인스턴스의 경우, 기본 그룹은 www-data입니다.
> - Nginx와 Unicorn을 사용하는 Rails 앱 서버 계층에 속하는 Amazon Linux 인스턴스의 경우, 기본 그룹은 nginx입니다.
> - 다른 모든 Amazon Linux 인스턴스의 경우, 기본 그룹은 apache입니다.
>
> 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 해당 속성을 재정의하면 설정을 변경할 수 있습니다. 자세한 내용은 속성 재정의 섹션을 참조하세요.

다른 정의 opsworks_deploy는 deploy 속성에 포함된 데이터를 기반으로 앱의 코드와 관련 파일을 리포지토리에서 체크아웃하고 인스턴스에 배포하는 세부 작업을 처리합니다. 모든 앱 유형에서 이 정의를 사용할 수 있습니다. 디렉터리 이름과 같은 배포 세부 정보는 콘솔 또는 API를 통해 지정되며 deploy 속성에 포함됩니다. 하지만 opsworks_deploy는 Git, 하위 버전, S3 및 HTTP 등 네 개의 지원되는 리포지토리 유형에서만 사용할 수 있습니다. 다른 리포지토리 유형을 사용하려면 이 코드를 직접 구현해야 합니다.

앱의 파일은 Tomcat webapps 디렉터리에 설치합니다. 일반적인 방법은 파일을 webapps에 직접 복사하는 것입니다. 그러나 AWS OpsWorks Stacks 배포는 인스턴스에 최대 5개의 앱 버전을 유지하도록 설계되었으므로 필요한 경우 이전 버전으로 롤백할 수 있습니다. 따라서 AWS OpsWorks Stacks는 다음을 수행합니다.

1. 이름에 타임스탬프가 포함된 고유 디렉터리(예: /srv/www/my_1st_jsp/releases/20130731141527)에 앱을 배포합니다.
2. 이 고유 디렉터리에 대한 symlink를 생성하고 current로 명명합니다(예: /srv/www/my_1st_jsp/current).
3. 이미 존재하지 않으면 webapps 디렉터리에서 2단계에서 생성된 current symlink까지의 symlink를 생성합니다.

이전 버전으로 롤백해야 할 경우 current symlink를 해당하는 타임스탬프가 있는 고유 디렉터리를 가리키도록 수정합니다(예를 들어 /srv/www/my_1st_jsp/current의 링크 대상 변경).

tomcat::deploy의 중간 부분은 symlink를 설정합니다.

```
 ...
 current_dir = ::File.join(deploy[:deploy_to], 'current')
 webapp_dir = ::File.join(node['tomcat']['webapps_base_dir'],
deploy[:document_root].blank? ? application : deploy[:document_root])

 # opsworks_deploy creates some stub dirs, which are not needed for typical webapps
 ruby_block "remove unnecessary directory entries in #{current_dir}" do
   block do
     node['tomcat']['webapps_dir_entries_to_delete'].each do |dir_entry|
       ::FileUtils.rm_rf(::File.join(current_dir, dir_entry), :secure => true)
     end
   end
 end
```

```
  link webapp_dir do
    to current_dir
    action :create
  end
  ...
```

이 레시피는 먼저 두 개의 변수 current_dir 및 webapp_dir을 생성하여 각각 current 및 webapp 디렉터리를 표시합니다. 그런 다음 link 리소스를 사용하여 webapp_dir를 current_dir에 링크합니다. AWS OpsWorks Stacks deploy::default 레시피는이 예제에 필요하지 않은 일부 스텁 디렉터리를 생성하므로 발췌문의 중간 부분이 이를 제거합니다.

tomcat::deploy의 마지막 부분은 필요할 경우 Tomcat 서비스를 재시작합니다.

```
  ...
  include_recipe 'tomcat::service'

  execute 'trigger tomcat service restart' do
    command '/bin/true'
    not_if { node['tomcat']['auto_deploy'].to_s == 'true' }
    notifies :restart, resources(:service => 'tomcat')
  end
end

include_recipe 'tomcat::context'
```

이 레시피는 먼저 tomcat::service를 실행하여 서비스가 이 Chef 실행에 대해 정의되도록 합니다. 그런 다음 실행 리소스를 사용하여 ['tomcat']['auto_deploy']가 'true'로 설정된 경우에만 서비스를 재시작하라고 알립니다. 그렇지 않은 경우에는 Tomcat이 webapps 디렉터리의 변경 사항을 수신 대기합니다. 그러므로 명시적 Tomcat 서비스 재시작은 필요하지 않습니다.

> **Note**
>
> execute 리소스가 실제로는 아무 것도 실행하지 않습니다. /bin/true는 단지 성공 코드를 반환하는 더미 shell 스크립트입니다. 여기서는 단지 재시작 알림을 생성하는 간편한 방법으로 사용됩니다. 앞서 언급한 대로, 알림을 사용하면 서비스가 너무 빈번히 재시작되지 않습니다.

마지막으로, tomcat::deploy는 tomcat::context를 실행합니다. 이는 백 엔드 데이터베이스가 변경된 경우 웹 앱 컨텍스트 구성 파일을 업데이트합니다.

스택 생성 및 애플리케이션 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션은 SimpleJSP라는 간단한 Java 서버 페이지(JSP) 애플리케이션을 실행하는 기본적인 스택 설정을 구현하기 위해 Tomcat 쿡북을 사용하는 방법을 보여줍니다. 스택은 TomCustom이라는 Tomcat 기반 사용자 지정 계층과 MySQL 계층으로 구성됩니다. SimpleJSP는 TomCustom에 배포되고 MySQL 데이터베이스로부터 일부 정보를 표시합니다. AWS OpsWorks Stacks 사용 방법의 기본 사항에 아직 익숙하지 않은 경우 먼저을 읽어야 합니다Chef 11 Linux 스택 시작하기.

SimpleJSP 애플리케이션

SimpleJSP 애플리케이션은 데이터베이스 연결을 설정하고 스택의 MySQL 데이터베이스에서 데이터를 검색하는 방법의 기초를 보여줍니다.

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
    <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
    <%
      StringBuffer output = new StringBuffer();
      DataSource ds = null;
      Connection con = null;
      Statement stmt = null;
      ResultSet rs = null;
      try {
        Context initCtx = new InitialContext();
        ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
        con = ds.getConnection();
        output.append("Databases found:<br>");
        stmt = con.createStatement();
        rs = stmt.executeQuery("show databases");
        while (rs.next()) {
```

```
          output.append(rs.getString(1));
          output.append("<br>");
        }
      }
      catch (Exception e) {
        output.append("Exception: ");
        output.append(e.getMessage());
        output.append("<br>");
      }
      finally {
        try {
          if (rs != null) {
            rs.close();
          }
          if (stmt != null) {
            stmt.close();
          }
          if (con != null) {
            con.close();
          }
        }
        catch (Exception e) {
          output.append("Exception (during close of connection): ");
          output.append(e.getMessage());
          output.append("<br>");
        }
      }
    %>
    <%= output.toString() %>
  </body>
</html>
```

SimpleJSP는 `DataSource` 객체를 사용하여 MySQL 데이터베이스와 통신합니다. Tomcat은 웹 앱 컨텍스트 구성 파일의 데이터를 사용하여 `DataSource` 객체를 생성 및 초기화하고 논리적 이름에 바인딩합니다. 그런 다음 이 논리적 이름을 Java Naming and Directory Interface(JNDI) 이름 서비스에 등록합니다. `DataSource` 객체의 인스턴스를 가져오려면 `InitialContext` 객체를 생성하고 리소스의 논리적 이름을 객체의 `lookup` 메서드로 전달합니다. 이 메서드가 해당 객체를 검색합니다. SimpleJSP 예제의 논리적 이름 `java:comp/env/jdbc/mydb`는 다음 요소로 구성됩니다.

- 이름의 나머지 부분과 콜론(:)으로 구분되는 루트 네임스페이스 `java`.
- 슬래시(/)로 구분되는 추가 네임스페이스.

Tomcat은 자동으로 리소스를 comp/env 네임스페이스에 추가합니다.

- 웹 앱 컨텍스트 구성 파일에서 정의되고 슬래시로 네임스페이스와 구분되는 리소스 이름.

  이 예제에서는 리소스 이름이 jdbc/mydb입니다.

데이터베이스와 연결을 설정하기 위해 SimpleJSP는 다음을 수행합니다.

1. DataSource 객체의 getConnection 메서드를 호출합니다. 이 메서드는 Connection 객체를 반환합니다.
2. Connection 객체의 createStatement 메서드를 호출하여 데이터베이스와 통신하는 데 사용할 Statement 객체를 생성합니다.
3. Statement 메서드를 호출하여 데이터베이스와 통신합니다.

   SimpleJSP는 executeQuery를 호출하여 서버의 데이터베이스를 나열하는 SHOW DATABASES 쿼리를 실행합니다.

executeQuery 메서드는 쿼리 결과를 포함하는 ResultSet 객체를 반환합니다. SimpleJSP는 반환된 ResultSet 객체에서 데이터베이스 이름을 가져오고 이들을 연결하여 출력 문자열을 생성합니다. 마지막으로 예제는 ResultSet, Statement 및 Connection 객체를 닫습니다. JSP 및 JDBC에 대한 자세한 정보는 각각 JavaServer Pages Technology 및 JDBC Basics를 참조하세요.

SimpleJSP를 스택에서 사용하려면 리포지토리에 배치해야 합니다. 지원되는 리포지토리를 모두 사용할 수 있지만, 다음 섹션에서 설명하는 예제 스택에서 SimpleJSP를 사용하려면 퍼블릭 S3 아카이브에 배치해야 합니다. 다른 표준 리포지토리를 사용하는 방법에 대한 자세한 정보는 쿡북 리포지토리 단원을 참조하세요.

S3 아카이브 리포지토리에 SimpleJSP를 저장하려면

1. 예제 코드를 simplejsp.jsp 파일에 복사한 다음 이 파일을 simplejsp 디렉터리에 저장합니다.
2. simplejsp 디렉터리의 .zip 아카이브를 생성합니다.
3. 퍼블릭 Amazon S3 버킷을 만들고, simplejsp.zip을 이 버킷에 업로드하고, 이 파일을 퍼블릭으로 설정합니다.

   이 작업을 수행하는 방법에 대한 자세한 설명은 Amazon Simple Storage Service 시작하기를 참조하세요.

스택 생성

SimpleJSP를 실행하려면 다음 계층이 포함된 스택이 필요합니다.

- 백 엔드 MySQL 서버를 지원하는 MySQL 계층.
- Tomcat 쿡북을 사용하여 Tomcat 서버 인스턴스를 지원하는 사용자 지정 계층.

스택을 생성하는 방법

1. AWS OpsWorks 스택 대시보드에서 스택 추가를 클릭하여 새 스택을 생성하고 고급 >>를 클릭하여 모든 옵션을 표시합니다. 다음과 같이 스택을 구성합니다.

   - 이름 - 사용자 정의 스택 이름. 이 예에서는 TomStack을 사용합니다.
   - 사용자 지정 Chef 쿡북 사용 - 토글을 예로 설정합니다. 그러면 몇 가지 추가 옵션이 표시됩니다.
   - 리포지토리 유형 - Git
   - 리포지토리 URL – `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`.
   - Custom Chef JSON - 다음 JSON을 추가합니다.

     ```
     {
       "tomcat": {
         "base_version": 7,
         "java_opts": "-Djava.awt.headless=true -Xmx256m"
       },
       "datasources": {
         "ROOT": "jdbc/mydb"
       }
     }
     ```

   나머지 옵션의 경우 기본값을 수락할 수 있습니다.

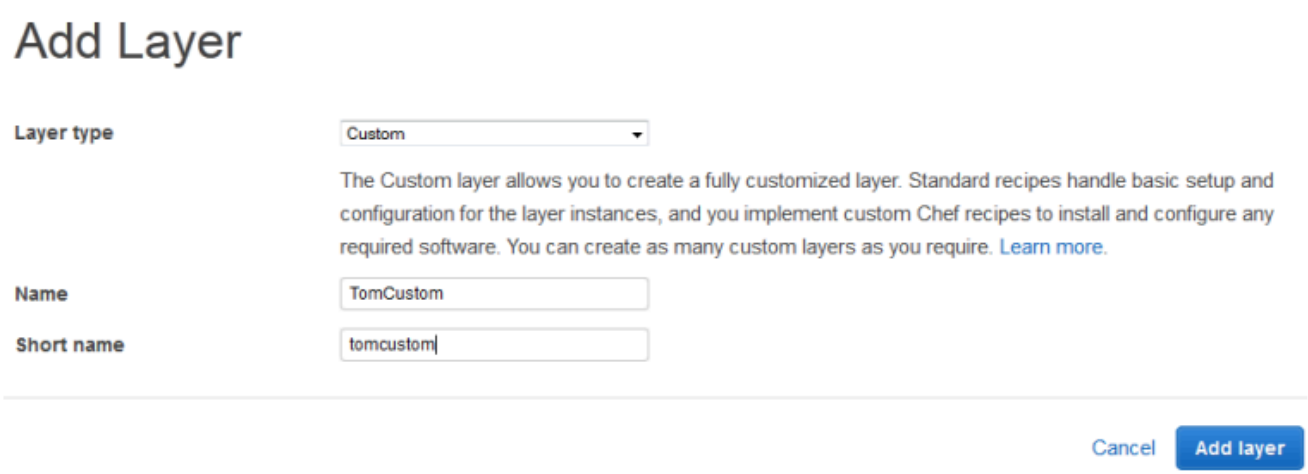
   사용자 지정 JSON에서는 다음 작업을 수행합니다.

   - Tomcat 쿡북의 `['base_version']` 속성을 재정의해 Tomcat 버전을 7로 설정합니다. 기본값은 6입니다.

- Tomcat 쿡북의 `['java_opts']` 속성을 재정의해 인스턴스가 헤드리스가 되도록 지정하고 JVM 최대 힙 크기를 256MB로 설정합니다. 기본값은 Amazon Linux를 실행하는 인스턴스에 대해 옵션을 설정하지 않는 것입니다.
- `['datasources]` 속성 값을 지정합니다. 이 속성 값은 tomcat::context 단원에서 설명하는 것처럼 JDBC 리소스 이름(jdbc/mydb)을 웹 앱 컨텍스트 이름(ROOT)에 할당합니다.

  이 마지막 속성에는 기본값이 없습니다. 따라서 사용자 지정 JSON을 사용하여 값을 설정해야 합니다.

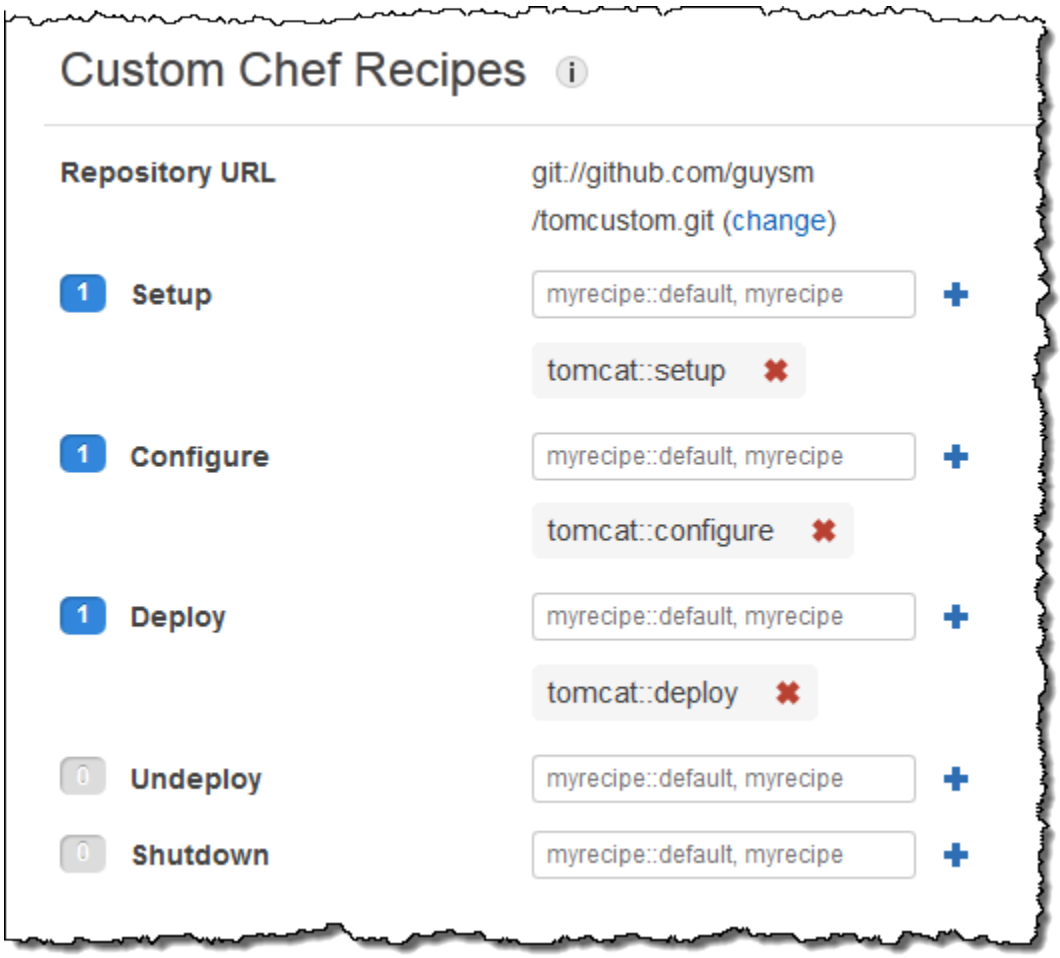
2. [계층 추가]를 클릭합니다. [계층 유형]으로 MySQL을 선택합니다. 그런 다음 [계층 추가]를 클릭합니다.
3. 탐색 창에서 [인스턴스]를 클릭한 다음 [인스턴스 추가]를 클릭합니다. 인스턴스 추가를 클릭하고 기본값을 수락합니다. 인스턴스에 해당하는 라인에서 [시작]을 클릭합니다.
4. [계층] 페이지로 돌아가 [+ 계층]을 클릭하여 계층을 추가합니다. Layer type(계층 유형)으로 사용자 지정을 클릭합니다. 이 예제에서는 계층 이름 및 짧은 이름으로 각각 **`TomCustom`** 및 **`tomcustom`**을 사용합니다.



5. [계층] 페이지에서 사용자 지정 계층으로 [레시피]를 클릭하고 [편집]을 클릭합니다. [사용자 지정 Chef 레시피]에서 다음과 같이 계층의 수명 주기 이벤트에 Tomcat 쿡북 레시피를 할당합니다.

   - 설정에 **`tomcat::setup`**을 입력하고 +를 클릭합니다.
   - 구성에 **`tomcat::configure`**을 입력하고 +를 클릭합니다.
   - 배포에 **`tomcat::deploy`**를 입력하고 +를 클릭합니다. 그런 다음 저장을 클릭합니다.

   .



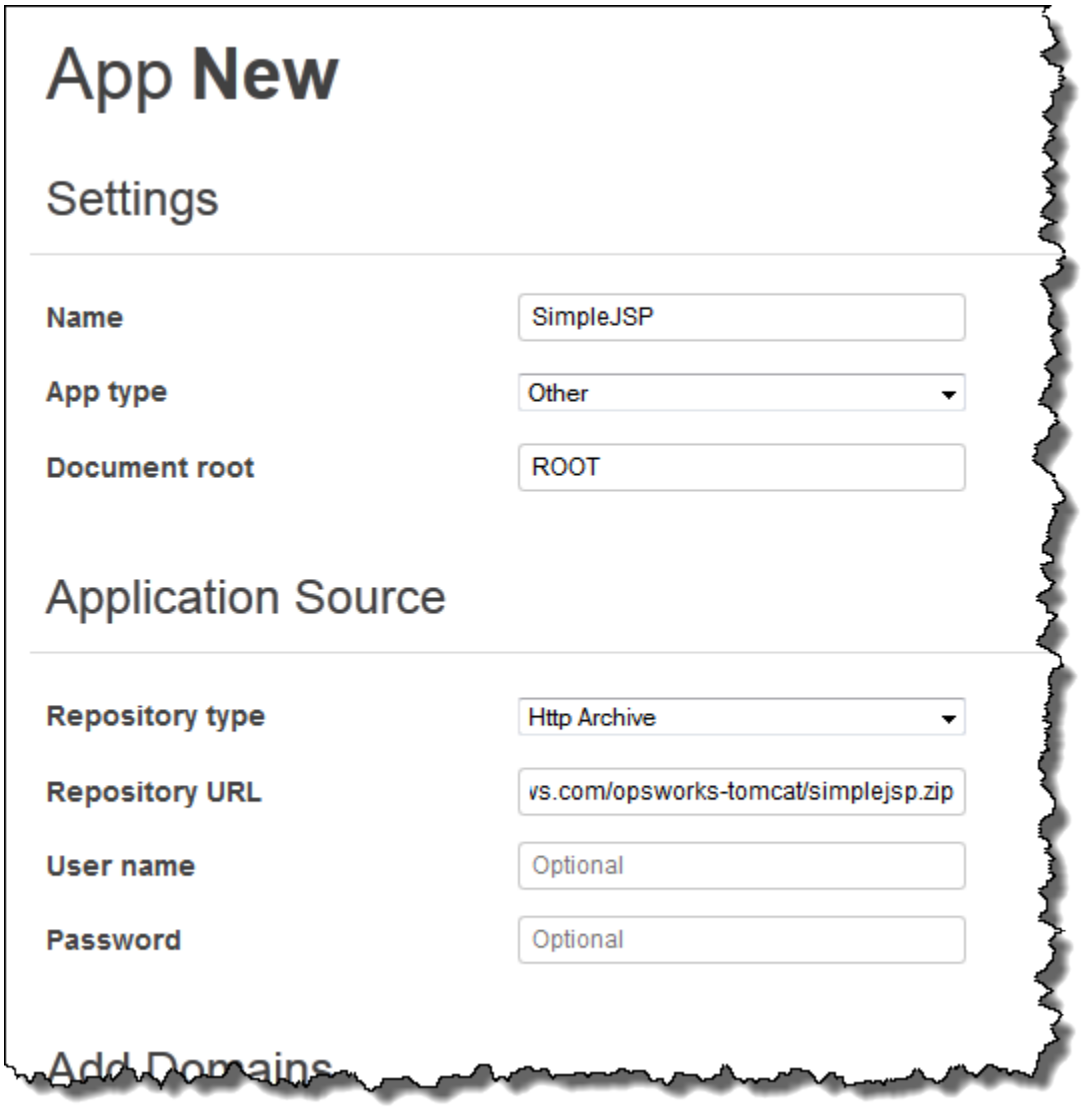
6. 탐색 창에서 [앱]을 클릭한 다음 [앱 추가]를 클릭합니다. 다음 옵션을 지정하고 [앱 추가]를 클릭합니다.

   - 이름 - 앱의 이름입니다. 예제에서는 SimpleJSP를 사용하며 AWS OpsWorks Stacks에서 생성된 짧은 이름은 simplejsp입니다.
   - 앱 유형 - 이 옵션은 기타로 설정합니다.

     AWS OpsWorks Stacks는 연결된 서버 인스턴스에 표준 앱 유형을 자동으로 배포합니다. 앱 유형을 기타로 설정하면 AWS OpsWorks Stacks에서는 Deploy 레시피를 실행하고 이 레시피에 따라 배포가 처리되도록 합니다.
   - [문서 루트] - 이 옵션은 **ROOT**으로 설정합니다.

     [문서 루트] 값은 컨텍스트 이름을 지정합니다.
   - 리포지토리 유형 - 이 옵션은 S3 아카이브로 설정합니다.
   - 리포지토리 URL - 이 옵션은 앞에서 생성한 앱의 Amazon S3 URL로 설정합니다.

다른 옵션에는 기본 설정을 사용합니다.



7. 인스턴스 페이지를 사용하여 TomCustom 계층에 인스턴스를 추가하고 시작합니다. AWS OpsWorks 스택은 Setup 레시피가 완료된 후 새 인스턴스에서 Deploy 레시피를 자동으로 실행하므로 인스턴스를 시작하면 SimpleJSP도 배포됩니다.
8. TomCustom 인스턴스가 온라인 상태이면 [인스턴스] 페이지에서 인스턴스 이름을 클릭하여 세부 정보를 확인합니다. 퍼블릭 IP 주소를 복사합니다. 그런 다음 URL을 http://*publicIP*/tc/*appname.jsp* 형식으로 구성합니다. 예를 들어, 이 URL은 **`http://50.218.191.172/tc/simplejsp.jsp`**와 같습니다.

> **Note**
>
> Tomcat에 요청을 전달하는 Apache URL은 기본 `['tomcat']['apache_tomcat_bind_path']` 속성이 `/tc/`로 설정되어 있습니다. SimpleJSP 문서 루트는 특수값인 `ROOT`로 설정되어 있으며 `/`로 해석됩니다. 따라서 URL은 ".../tc/simplejsp.jsp"입니다.

9. 이전 단계의 URL을 브라우저에 붙여 넣습니다. 다음과 같은 모양이어야 합니다.

```
Databases found:
information_schema
simplejsp
test
```

> **Note**
>
> 스택에 MySQL 인스턴스가 있는 경우 AWS OpsWorks Stacks는 앱의 짧은 이름으로 이름이 지정된 각 앱에 대한 데이터베이스를 자동으로 생성합니다.

스택 구성 및 배포 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 인스턴스에서 명령을 실행하면 예를 들어 수명 주기 배포 이벤트에 대한 응답으로 배포 명령이 스택의 현재 구성을 설명하는 인스턴스의 노드 객체에 속성 집합을 추가합니다. Deploy events 및 Execute Recipes 스택 명령의 경우 AWS OpsWorks Stacks는 몇 가지 추가 배포 정보를 제공하는 배포 속성을 설치합니다. 노드 객체에 대한 자세한 정보는 속성 재정의 단원을 참조하세요. 정규화된 도메인 이름을 포함하여 일반적으로 사용되는 스택 구성 및 배포 속성의 목록은 스택 구성 및 배포 속성: Linux 및 내장 쿡북 속성를 참조하세요.

> **Note**
>
> Linux 스택에서는 에이전트 CLI의 get_json 명령을 사용하여 JSON 객체로 형식 지정된 이러한 속성의 완전한 목록을 가져올 수 있습니다.

다음 섹션에서는 다음으로 구성된 간단한 스택의 Configure 이벤트 및 Deploy 이벤트에 연결된 속성을 보여 줍니다.

- 두 개의 인스턴스가 있는 PHP 앱 서버 계층
- 1개의 인스턴스가 포함된 HAProxy 계층

예제는 PHP 앱 서버 인스턴스 중 하나인 php-app1의 예제입니다. 편의상 속성은 JSON 객체로 형식이 지정되어 있습니다. 객체의 구조는 속성의 정규화된 이름에 매핑됩니다. 예를 들어 `node[:opsworks][:ruby_version]` 속성은 다음과 같이 JSON 표시로 나타납니다.

```
{
  "opsworks": {
    ...
    "ruby_version": "1.8.7",
    ...
  }
}
```

주제

- Configure 속성
- 배포 속성

Configure 속성

다음 JSON 객체는 인스턴스가 온라인 상태가 되거나 오프라인 상태가 될 때 스택의 모든 인스턴스에서 발생하는 Configure 이벤트의 속성을 보여 줍니다. 속성에는 내장 스택 구성 속성과 이벤트 전에 스택에 대해 구성된 사용자 지정 JSON 속성(이 예제에는 없음)이 포함됩니다. 속성은 길이를 위해 편집되었습니다. 다양한 속성에 대한 상세한 설명은 스택 구성 및 배포 속성: Linux 및 내장 쿡북 속성를 참조하세요.

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "192.0.2.0",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
          },
          "php-app1": {
            ...
          }
        },
        "name": "PHP Application Server"
      },
      "lb": {
        "id": "15c86142-d836-4191-860f-f4d310440f14",
        "instances": {
          "lb1": {
           ...
          }
        },
        "name": "Load Balancer"
      }
    },
    "agent_version": "104",
    "applications": [

    ],
    "stack": {
      "name": "MyStack"
    },
```

```
    "ruby_version": "1.8.7",
    "sent_at": 1361911623,
    "ruby_stack": "ruby_enterprise",
    "instance": {
      "layers": [
        "php-app"
      ],
      "region": "us-west-2",
      "ip": "192.0.2.0",
      "id": "45ef378d-b87c-42be-a1b9-b67c48edafd4",
      "aws_instance_id": "i-32037f00",
      "availability_zone": "us-west-2a",
      "private_dns_name": "ip-10-252-84-253.us-west-2.compute.internal",
      "instance_type": "c1.medium",
      "hostname": "php-app1",
      "private_ip": "10.252.84.253",
      "backends": 8,
      "architecture": "i386",
      "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
    },
    "activity": "configure",
    "rails_stack": {
      "name": null
    },
    "deployment": null,
    "valid_client_activities": [
      "reboot",
      "stop",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes"
    ]
  },
  "opsworks_custom_cookbooks": {
    "recipes": [

    ],
    "enabled": false
  },
  "recipes": [
    "opsworks_custom_cookbooks::load",
```

```
    "opsworks_ganglia::configure-client",
    "ssh_users",
    "agent_version",
    "mod_php5_apache2::php",
    "php::configure",
    "opsworks_stack_state_sync",
    "opsworks_custom_cookbooks::execute",
    "test_suite",
    "opsworks_cleanup"
  ],
  "opsworks_rubygems": {
    "version": "1.8.24"
  },
  "ssh_users": {
  },
  "opsworks_bundler": {
    "manage_package": null,
    "version": "1.0.10"
  },
  "deploy": {
  }
}
```

대부분의 정보는 흔히 네임스페이스라고 하는 opsworks 속성 아래에 있습니다. 다음 목록은 주요 속성을 설명합니다.

- layers 속성 - 각각 스택의 계층 중 하나의 구성을 설명하는 속성의 집합.

  계층은 짧은 이름(이 예제의 경우, php-app 및 lb)으로 식별됩니다. 기타 계층의 짧은 이름에 대한 자세한 정보는 AWS OpsWorks Stacks 계층 참조를 참조하세요.
- instances 속성 - 모든 계층에는 인스턴스의 짧은 이름으로 명명된, 계층의 온라인 인스턴스 각각의 속성을 포함하는 instances 요소가 있습니다.

  PHP 앱 서버 계층에는 php-app1과 php-app2라는 2개의 인스턴스가 있습니다. HAProxy 계층에는 1개의 인스턴스(lb1)가 있습니다.

> **Note**
>
> instances 요소에는 특정 스택 및 배포 속성이 생성될 때 온라인 상태인 인스턴스만이 포함됩니다.

- 인스턴스 속성 - 각각의 인스턴스 속성에는 인스턴스의 프라이빗 IP 주소와 프라이빗 DNS 이름 등 인스턴스를 특징짓는 속성 집합이 포함됩니다. 간결하게 하기 위해 예제에서는 php-app2 속성만 자세히 표시합니다. 나머지 속성에 포함된 정보도 비슷합니다.
- `applications` – 배포된 앱의 목록(이 예제에서는 사용되지 않음).
- `stack` - 스택 이름(이 예제에서는 `MyStack`).
- `instance` - 이 속성들이 설치되는 인스턴스(이 예제에서는 php-app1). 레시피는 이 속성을 사용하여 인스턴스의 퍼블릭 IP 주소 등 레시피가 실행되고 있는 인스턴스에 대한 정보를 얻을 수 있습니다.
- `activity` – 속성을 생성한 활동(이 예제에서는 Configure 이벤트).
- `rails_stack` – Rails 앱 서버 계층을 포함하는 스택의 Rails 스택.
- `deployment` – 이 속성들이 배포에 연결되어 있는지 여부. 이 예제에서는 Configure 이벤트에 연결되므로 `null`로 설정되어 있습니다.
- `valid_client_activities` – 유효한 클라이언트 활동의 목록.

`opsworks` 속성 뒤에는 다음을 포함한 그 밖의 몇몇 최상위 속성이 옵니다.

- `opsworks_custom_cookbooks` – 사용자 지정 쿡북이 활성화되었는지 여부. 활성화된 경우, 이 속성에는 사용자 지정 레시피 목록이 포함됩니다.
- `recipes` – 이 활동에 의해 실행된 레시피.
- `opsworks_rubygems` – 인스턴스의 RubyGems 버전.
- `ssh_users` – SSH 사용자 목록(이 예제에는 없음).
- `opsworks_bundler` – bundler 버전 및 활성화 여부.
- `deploy` – 배포 활동에 대한 정보(이 예제에는 없음).

배포 속성

Deploy 이벤트 또는 레시피 실행 스택 명령의 속성은 내장 스택 구성 및 배포 속성과 사용자 지정 스택 또는 배포 속성(이 예제에는 없음)으로 구성됩니다. 다음 JSON 객체는 SimplePHP 앱을 스택의 PHP 인스턴스에 배포한 배포 이벤트에 연결된 [php-app1]의 속성을 보여 줍니다. 객체 대부분은 앞 섹션에서 설명한 Configure 이벤트의 속성과 비슷한 스택 구성 속성으로 구성되므로 예제에서는 주로 배포에 고유한 속성에 집중합니다. 다양한 속성에 대한 상세한 설명은 스택 구성 및 배포 속성: Linux 및 내장 쿡북 속성를 참조하세요.

```
{
```

```
    ...
  "opsworks": {
    ...
    "activity": "deploy",
    "applications": [
      {
        "slug_name": "simplephp",
        "name": "SimplePHP",
        "application_type": "php"
      }
    ],
    "deployment": "5e6242d7-8111-40ee-bddb-00de064ab18f",
    ...
  },
  ...
{
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
      "auto_bundle_on_deploy": true,
      "deploy_to": "/srv/www/simplephpapp",
      "deploying_user": "arn:aws:iam::123456789012:user/guysm",
      "document_root": null,
      "domains": [
        "simplephpapp"
      ],
      "migrate": false,
      "mounted_at": null,
      "rails_env": null,
      "restart_command": "echo 'restarting app'",
      "sleep_before_restart": 0,
      "ssl_support": false,
      "ssl_certificate": null,
      "ssl_certificate_key": null,
      "ssl_certificate_ca": null,
      "scm": {
        "scm_type": "git",
```

```
        "repository": "git://github.com/amazonwebservices/opsworks-demo-php-simple-
app.git",
        "revision": "version1",
        "ssh_key": null,
        "user": null,
        "password": null
      },
      "symlink_before_migrate": {
        "config/opsworks.php": "opsworks.php"
      },
      "symlinks": {
      },
      "database": {
      },
      "memcached": {
        "host": null,
        "port": 11211
      },
      "stack": {
        "needs_reload": false
      }
    }
  },
}
```

opsworks 속성은 앞 섹션의 예제와 대체로 동일합니다. 다음 섹션들은 배포와 가장 관련이 깊습니다.

- activity – 이 속성들과 연결된 이벤트(이 예제에서는 Deploy 이벤트).
- applications – 앱의 이름, 슬러그 이름, 유형을 제공하는 각 앱의 속성 집합을 포함합니다.

  슬러그 이름은 AWS OpsWorks Stacks가 앱 이름에서 생성하는 앱의 짧은 이름입니다. SimplePHP의 슬러그 이름은 simplephp입니다.
- deployment – 배포를 고유하게 식별하는 배포 ID.

deploy 속성은 배포 중인 앱에 대한 정보를 포함합니다. 예를 들어 내장 Deploy 레시피는 deploy 속성의 데이터를 사용하여 적절한 디렉터리에 파일을 설치하고 데이터베이스 연결 파일을 생성합니다. deploy 속성에는 배포된 각 앱마다 앱의 짧은 이름으로 명명된 속성이 하나씩 포함됩니다. 각 앱 속성은 다음 속성을 포함합니다.

- `environment_variables` – 앱에 대해 정의한 환경 변수를 포함합니다. 자세한 내용은 환경 변수 섹션을 참조하세요.
- `domains` – 기본적으로 도메인은 앱의 짧은 이름입니다(이 예제에서는 simplephpapp). 사용자 지정 도메인을 할당한 경우, 여기에 도메인도 표시됩니다. 자세한 내용은 사용자 지정 도메인 사용 섹션을 참조하세요.
- `application` – 앱의 짧은 이름.
- `scm` – 이 요소에는 앱의 리포지토리(이 예제에서는 Git 리포지토리)에서 앱의 파일을 다운로드하는 데 필요한 정보가 포함됩니다.
- `database` – 데이터베이스 정보(스택에 데이터베이스 계층이 포함된 경우)
- `document_root` - 문서 루트. 이 예제에서는 `null`로 설정되어 루트가 공개되어 있음을 나타냅니다.
- `ssl_certificate_ca`, `ssl_support`, `ssl_certificate_key` – 앱에 SSL 지원이 있는지 여부를 나타냅니다. SSL 지원이 있는 경우, `ssl_certificate_key` 및 `ssl_certificate_ca` 속성은 해당 인증서로 설정됩니다.
- `deploy_to` – 앱의 루트 디렉터리.

쿡북 101

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

프로덕션 수준 AWS OpsWorks Stacks 스택에는 일반적으로 사용자 지정이 필요합니다. 즉, 하나 이상의 레시피, 속성 파일 또는 템플릿 파일을 사용하여 사용자 지정 Chef 쿡북을 구현하는 경우가 많습니다. 이 주제는 Stacks용 쿡북 구현에 대한 AWS OpsWorks 자습서 소개입니다.

Stacks가 AWS OpsWorks 쿡북을 사용하는 방법에 대한 자세한 내용은 단원을 참조하십시오쿡북과 레시피. Chef 레시피를 구현하고 테스트하는 방법에 대한 자세한 정보는 Chef를 사용한 인프라 테스트 드라이브, 2판 단원을 참조하세요.

이 자습서의 예제는 다음 두 부분으로 나뉩니다.

- 쿡북 기본 사항는 Chef에 익숙하지 않은 사용자를 위한 예제 안내서입니다. 경험이 많은 Chef 사용자는 이 섹션을 건너뛰어도 됩니다.

  이 예제는 패키지 설치 또는 디렉터리 생성 등 일반적인 작업을 수행하기 위해 쿡북을 구현하는 방법의 기초를 안내합니다. 프로세스를 간단히 하기 위해 Vagrant와 Test Kitchen이라는 유용한 도구 2개를 사용하여 예제 대부분을 가상 머신에서 로컬로 실행하게 됩니다. 쿡북 기본 사항를 시작하기에 앞서 이들 도구의 설치 및 사용 방법을 배우기 위해 먼저 Vagrant와 Test Kitchen를 읽어야 합니다. Test Kitchen은 아직 Windows를 지원하지 않기 때문에 예제는 모두 Linux용이며 Windows에 맞게 조정하는 방법을 알려주는 참고가 포함되어 있습니다.
- AWS OpsWorks Stacks용 쿡북 구현 에서는 Windows 스택용를 포함하여 AWS OpsWorks Stacks용 레시피를 구현하는 방법을 설명합니다.

  Berkshelf를 사용하여 외부 쿡북을 관리하는 방법 등 몇 가지 고급 s도 포함되어 있습니다. 예제는 쿡북 기본 사항의 예제처럼 Chef가 처음인 사용자들을 위해 작성되었습니다. 그러나 AWS OpsWorks Stacks는 Chef 서버와 약간 다르게 작동하므로 숙련된 Chef 사용자는 최소한이 섹션을 읽어보는 것이 좋습니다.

Vagrant와 Test Kitchen

Linux 인스턴스용 레시피 작업을 하는 경우, Vagrant와 Test Kitchen은 학습과 초기 개발 및 테스트에 매우 유용한 도구입니다. 이것은 Vagrant와 Test Kitchen을 간략히 설명하고 기본적인 도구 사용 방법을 시작하고 익힐 수 있는 설치 지침과 안내서를 제공합니다. Vagrant는 Windows를 지원하지만 Test Kitchen은 지원하지 않으므로 이 도구들의 경우, Linux 예제만 나와 있습니다.

Vagrant

Vagrant는 가상 머신에서 코드를 실행하고 테스트할 수 있는 일관된 환경을 제공합니다. Vagrant 박스라고 하는 다양한 환경을 지원하며, 각 환경은 구성된 운영 체제를 나타냅니다. AWS OpsWorks Stacks의 경우 관심 환경은 Ubuntu, Amazon 또는 Red Hat Enterprise Linux(RHEL) 배포를 기반으로 하므로 예제에서는 주로 라는 이름의 Vagrant 상자를 사용합니다`opscode-ubuntu-12.04`.

Vagrant는 Linux, Windows, Macintosh 시스템에서 사용 가능하므로 원하는 워크스테이션을 사용하여 지원되는 어떤 운영 체제에서도 레시피를 구현하고 테스트할 수 있습니다. 이 장의 예제는 Ubuntu Linux 시스템에서 작성되었지만 Windows 또는 Macintosh 시스템에도 간단히 적용할 수 있습니다.

Vagrant는 기본적으로 가상화 공급자용 래퍼입니다. 대부분의 예제는 VirtualBox 공급자를 사용합니다. VirtualBox는 무료이며 Linux, Windows, Macintosh 시스템에서 사용할 수 있습니다. 아직 시스템에

VirtualBox이 설치되어 있지 않은 경우, Vagrant 안내서에 설치 지침이 나와 있습니다. VirtualBox에서 Ubuntu 기반 환경을 실행할 수 있지만 Amazon Linux는 Amazon EC2 인스턴스에만 사용할 수 있습니다. 다만 CentOS와 같은 유사한 운영 체제는 VirtualBox에서 실행할 수 있어 초기 개발과 테스트에 유용합니다.

다른 공급자에 대해서는 Vagrant 설명서를 참조하세요. 특히 `vagrant-aws` 플러그인 공급자를 사용하면 Amazon EC2 인스턴스에 Vagrant를 사용할 수 있습니다. 이 공급자는 Amazon EC2 인스턴스에서만 사용할 수 있는 Amazon Linux에서 레시피를 테스트하는 데 특히 유용합니다. `vagrant-aws` 공급자는 무료이지만 AWS 계정이 있어야 하며, 사용하는 AWS 리소스에 대한 요금을 지불해야 합니다.

이 시점에서 워크스테이션에 Vagrant를 설치하는 방법을 설명하고 Vagrant의 기본적 사용 방법을 알려 주는 Vagrant 시작하기 안내서를 읽어야 합니다. 이 장의 예제는 Git 리포지토리를 사용하지 않기 때문에 원한다면 안내서에서 해당 부분은 생략해도 됩니다.

Test Kitchen

Test Kitchen은 Vagrant에서 쿡북을 실행하고 테스트하는 프로세스를 간소화합니다. 실제로는 직접 Vagrant를 사용할 일은 거의 없습니다. 다음을 비롯한 대부분의 일반적 작업은 Test Kitchen이 수행합니다.

- Vagrant에서 인스턴스 시작.
- 쿡북을 인스턴스로 이전.
- 인스턴스에서 쿡북의 레시피 실행.
- 인스턴스에서 쿡북의 레시피 테스트.
- SSH를 사용하여 인스턴스에 로그인.

Test Kitchen gem을 직접 설치하는 대신 Chef DK를 설치하는 것이 좋습니다. 이 패키지에는 Chef 외에도 Test Kitchen, Berkshelf, ChefSpec 및 유용한 그 밖의 몇 가지 도구가 포함되어 있습니다.

이 시점에서 Test Kitchen을 사용하여 레시피를 실행하고 테스트하는 방법을 알려 주는 Test Kitchen 시작하기 안내서를 읽어야 합니다.

> **Note**
>
> 이 장의 예제에서는 레시피를 실행하는 편리한 방법으로 Test Kitchen을 사용합니다. 원한다면 예제에 대해 알아야 할 모든 내용을 다루고 있는 수동으로 확인(Manually Verifying) 섹션을 마친 다음 시작하기 안내서를 중단해도 됩니다. 하지만 Test Kitchen은 일차적으로 bash

automated test system(BATS) 같은 테스트 프레임워크를 지원하는 테스트 플랫폼입니다. 일정 시점이 되면 안내서의 나머지 부분을 끝마쳐 Test Kitchen을 사용해 레시피를 테스트하는 방법을 배워야 합니다.

쿡북 기본 사항

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

쿡북을 사용하여 다양한 작업을 수행할 수 있습니다. 다음 주제에서는 사용자가 Chef를 처음 사용하는 것으로 가정하며, 일부 공통 작업 수행에 쿡북을 사용하는 방법을 설명합니다. Test Kitchen은 아직 Windows를 지원하지 않기 때문에 예제는 모두 Linux용이며 Windows에 맞게 조정하는 방법을 알려주는 참고가 포함되어 있습니다. Chef가 처음인 경우, Windows 작업을 하게 되더라도 이 예제를 끝까지 살펴보는 것이 좋습니다. 이 주제의 예제 대부분은 예제에서 언급하는 몇 가지 적당한 변경을 거치면 Windows 인스턴스에서도 사용할 수 있습니다. 모든 예제는 가상 머신에서 실행되므로 Linux 컴퓨터가 없어도 됩니다. 평소 사용하는 워크스테이션에 Vagrant와 Test Kitchen을 설치하세요.

Note

이러한 레시피를 Windows 인스턴스에서 실행하는 가장 간단한 방법은 Windows 스택을 생성하고 스택의 인스턴스 중 하나에서 레시피를 실행하는 것입니다. AWS OpsWorks Stacks Windows 인스턴스에서 레시피를 실행하는 방법에 대한 자세한 내용은 섹션을 참조하세요Windows 인스턴스에서 레시피 실행.

계속하기 전에 Vagrant와 Test Kitchen을 설치하고 시작하기 안내서를 마치십시오. 자세한 내용은 Vagrant와 Test Kitchen 섹션을 참조하세요.

주제

- 레시피 구조
- 예제 1: 패키지 설치

- 예제 2: 사용자 관리
- 예제 3: 디렉터리 생성
- 예제 4: 레시피에 흐름 제어 추가
- 예제 5: 속성 사용
- 예제 6: 파일 생성
- 예제 7: 명령 및 스크립트 실행
- 예제 8: 서비스 관리
- 예제 9: Amazon EC2 인스턴스 사용
- 다음 단계

레시피 구조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

쿡북은 기본적으로 인스턴스에서 다양한 작업을 수행할 수 있는 레시피의 집합입니다. 레시피를 구현하는 방법을 명확히 알려면 간단한 예를 살펴보는 것이 도움이 됩니다. 다음은 내장 HAProxy 계층의 설정 레시피입니다. 지금은 전체적인 구조에 집중하고 세부적인 사항에 너무 신경 쓰지 마십시오. 세부적인 내용은 이후 예제에서 다룹니다.

```
package 'haproxy' do
  action :install
end

if platform?('debian','ubuntu')
  template '/etc/default/haproxy' do
    source 'haproxy-default.erb'
    owner 'root'
    group 'root'
    mode 0644
```

```
  end
end

include_recipe 'haproxy::service'

service 'haproxy' do
  action [:enable, :start]
end

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, "service[haproxy]"
end
```

> **Note**
>
> 이 예제 및 다른 작업 레시피와 관련 파일의 예제는 AWS OpsWorks Stacks 내장 레시피를 참조하세요.

이 예제는 다음 섹션에서 설명하는 주요 레시피 요소를 집중적으로 다룹니다.

주제

- 리소스
- 흐름 제어
- 포함된 레시피

리소스

레시피는 주로 Chef 리소스 집합으로 구성됩니다. 각각의 리소스는 설치할 패키지나 시작할 서비스 같은 인스턴스 최종 상태의 특정 측면을 지정합니다. 예제에는 다음 4개의 리소스가 있습니다.

- 설치된 패키지를 나타내는 `package` 리소스입니다. 이 예시에서는 HAProxy 서버입니다.
- 이 예제에서 HAProxy 서비스에 해당하는 서비스를 나타내는 `service` 리소스.
- 이 예제에서 2개의 HAProxy 구성 파일에 해당하는, 지정된 템플릿에서 생성할 파일을 나타내는 2개의 `template` 리소스.

리소스는 인스턴스 상태를 지정하는 선언적 방법을 제공합니다. 백그라운드에서는 각각의 리소스에 패키지 설치, 디렉터리 생성 및 구성, 서비스 시작 등등의 필요한 작업을 수행하는 연결된 공급자가 있습니다. 작업의 세부 사항이 특정 운영 체제에 따라 다른 경우, 리소스는 여러 개의 공급자를 가지며 시스템에 적합한 공급자를 사용합니다. 예를 들어 Red Hat Linux 시스템에서 package 공급자는 yum을 사용하여 패키지를 설치합니다. Ubuntu Linux 시스템에서 package 공급자는 apt-get을 사용합니다.

다음과 같은 일반적 형식을 사용하여 Ruby 코드 블록으로 리소스를 구현합니다.

```
resource_type "resource_name" do
  attribute1 'value1'
  attribute2 'value2'
  ...
  action :action_name
  notifies : action 'resource'
end
```

요소는 다음과 같습니다.

리소스 유형

(필수) 예제에는 세 가지 리소스 형식(package, service, template)이 포함됩니다.

리소스 이름

(필수) 이름은 특정 리소스를 식별하며, 경우에 따라 속성 중 하나의 기본값으로 사용됩니다. 예제에서 package는 haproxy라는 이름의 패키지 리소스를 나타내며, 첫 번째 template 리소스는 /etc/default/haproxy라는 이름의 구성 파일을 나타냅니다.

속성

(선택 사항) 속성은 리소스 구성을 지정하며, 리소스 유형 및 리소스 구성 방법에 따라 달라집니다.

- 예제의 template 리소스는 생성된 파일의 원본, 소유자, 그룹, 모드를 지정하는 속성 세트를 명시적으로 정의합니다.
- 예제의 package 및 service 리소스는 어떤 속성도 명시적으로 정의하지 않습니다.

  리소스 이름은 일반적으로 필수 속성의 기본값이며, 필요한 전부인 경우도 있습니다. 예를 들어 리소스 이름은 package 리소스의 package_name 속성에 대한 기본값이며, 이것이 유일한 필수 속성입니다.

가드 속성이라고 하는 몇 가지 특화된 속성도 있는데, 리소스 공급자가 언제 작업을 수행할지 지정합니다. 예를 들어 `only_if` 속성은 지정된 조건이 충족되는 경우에만 작업을 수행하라고 리소스 공급자에게 명령합니다. HAProxy 레시피는 가드 속성을 사용하지 않지만 다음 몇 가지 예제에서는 가드 속성을 사용합니다.

작업 및 알림

(선택 사항) 작업 및 알림은 공급자가 수행할 작업을 지정합니다.

- `action`은 설치 또는 생성 등 지정된 작업을 하도록 공급자에게 명령합니다.

  각 리소스에는 특정 리소스에 따라 달라지는 작업 세트가 있으며, 이 중 하나는 기본 작업입니다. 예제에서 `package` 리소스의 작업은 `install`인데, 공급자에게 패키지를 설치하라고 명령합니다. 첫 번째 `template` 리소스에는 `action` 요소가 없기 때문에 공급자는 기본 `create` 작업을 수행합니다.

- `notifies`는 다른 리소스의 공급자에게 작업을 수행할 것을 명령하지만 리소스의 상태가 변경된 경우에 한합니다.

  `notifies`는 일반적으로 `template` 및 `file` 같은 리소스와 함께 사용되어 구성 파일 수정 후 서비스 재시작 같은 작업을 수행합니다. 리소스에는 기본 알림이 없습니다. 알림을 원하는 경우, 리소스에 명시적인 `notifies` 요소가 있어야 합니다. HAProxy 레시피에서 두 번째 `template` 리소스는 연결된 구성 파일이 변경된 경우, haproxy `service` 리소스에 HAProxy 서비스를 다시 시작하라고 알립니다.

리소스가 운영 체제에 따라 달라지는 경우가 있습니다.

- 일부 리소스는 Linux 또는 Windows 시스템에서만 사용할 수 있습니다.

  예를 들어 패키지는 Linux 시스템에 패키지를 설치하고, windows_package는 Windows 시스템에 패키지를 설치합니다.

- 일부 리소스는 어떤 운영 체제에도 사용할 수 있지만 특정 시스템에 고유한 속성을 가지고 있습니다.

  예를 들어 파일 리소스는 Linux 시스템이나 Windows 시스템에서 사용할 수 있지만 구성 권한에 대한 속성을 별도로 가지고 있습니다.

각 리소스의 사용 가능한 속성, 작업, 알림을 포함한 표준 리소스에 대한 설명은 리소스 및 공급자에 대하여를 참조하세요.

흐름 제어

레시피는 Ruby 애플리케이션이므로 Ruby 제어 구조를 사용하여 흐름 제어를 레시피에 통합할 수 있습니다. 예를 들어 Ruby 조건부 논리를 사용하여 레시피가 시스템에 따라 다르게 동작하도록 할 수 있습니다. HAProxy 레시피에 포함된 `if` 블록은 레시피가 Debian 또는 Ubuntu 시스템에서 실행되는 경우에만 `template` 리소스를 사용하여 구성 파일을 생성합니다.

다른 일반적 시나리오는 루프를 사용하여 하나의 리소스를 상이한 속성 설정으로 여러 번 실행하는 것입니다. 예를 들어 루프를 사용해 `directory` 리소스를 서로 다른 디렉터리 이름으로 여러 번 실행하여 디렉터리 집합을 생성할 수 있습니다.

> **Note**
>
> Ruby에 익숙하지 않다면 대부분의 레시피에 대해 알아야 할 내용을 다루고 있는 Just Enough Ruby for Chef를 참조하세요.

포함된 레시피

`include_recipe`는 코드의 다른 레시피를 포함하며, 레시피를 모듈화하여 여러 레시피에서 같은 코드를 다시 사용할 수 있도록 해 줍니다. 호스트 레시피 실행 시 Chef는 호스트 레시피를 실행하기 전에 각각의 `include_recipe` 요소를 지정된 레시피의 코드로 대체합니다. 표준 Chef `cookbook_name::recipe_name` 구문을 사용하여 포함된 레시피를 식별합니다. 여기서 `recipe_name`에는 `.rb` 확장명이 생략되어 있습니다. 예제에는 HAProxy 서비스를 나타내는 하나의 레시피인 `haproxy::service`가 포함됩니다.

> **Note**
>
> Chef 11.10 이후 버전에서 실행되는 레시피에서 `include_recipe`를 사용하여 다른 쿡북의 레시피를 포함시키려면 `depends` 문을 사용하여 쿡북의 `metadata.rb` 파일에서 종속성을 선언해야 합니다. 자세한 내용은 레시피 구현: Chef 11.10 섹션을 참조하세요.

예제 1: 패키지 설치

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

패키지 설치는 가장 일반적인 레시피 용도 중 하나이며, 패키지에 따라서는 아주 간단할 수 있습니다. 예를 들어 다음 레시피는 Linux 시스템에 Git를 설치합니다.

```
package 'git' do
  action :install
end
```

package리소스는 패키지 설치를 처리합니다. 이 예제에서는 아무 속성도 지정할 필요가 없습니다. 리소스 이름은 패키지를 식별하는 package_name 속성의 기본값입니다. install 작업은 공급자에게 패키지를 설치하라고 명령합니다. install 리소스의 기본 작업인 package을 건너뛰면 코드를 더욱 간단하게 만들 수 있습니다. 레시피를 실행할 때 Chef는 적절한 공급자를 사용하여 패키지를 설치합니다. 예제에 사용할 Ubuntu 시스템에서는 공급자가 apt-get을 호출하여 Git를 설치합니다.

> **Note**
>
> Windows 시스템에 소프트웨어를 설치하려면 약간 다른 절차가 필요합니다. 자세한 내용은 Windows 소프트웨어 설치 섹션을 참조하세요.

Test Kitchen을 사용하여 Vagrant에서 이 레시피를 실행하려면 먼저 쿡북을 설정한 다음 Test Kitchen을 초기화하고 구성해야 합니다. 다음은 Linux 시스템용이지만 Windows 및 Macintosh 시스템과 절차는 기본적으로 비슷합니다. 먼저 Terminal 창을 엽니다. 이 장의 모든 예제는 명령줄 도구를 사용합니다.

쿡북을 준비하려면

1. 홈 디렉터리에서 opsworks_cookbooks라는 하위 디렉터리를 만듭니다. 이 디렉터리에는 이 장의 모든 쿡북이 저장됩니다. 그런 다음 이 쿡북의 하위 디렉터리로 installpkg를 만들고 이 디렉터리로 이동합니다.
2. installpkg에서 다음 코드가 포함된 metadata.rb라는 파일을 만듭니다.

   ```
   name "installpkg"
   version "0.1.0"
   ```

간단한 설명을 위해 이 장의 예제에서는 쿡북 이름 및 버전만 지정하지만, `metadata.rb`에 다양한 쿡북 메타데이터를 넣을 수 있습니다. 자세한 정보는 About Cookbook Metadata를 참조하세요.

> **Note**
>
> Test Kitchen에서는 `metadata.rb` 파일의 데이터를 사용하여 기본 구성 파일을 만듭니다. 그러므로 Test Kitchen을 초기화하기 전에 이 파일을 만드십시오.

3. `installpkg`에서 `kitchen init`를 실행합니다. 이 코드는 Test Kitchen을 초기화하고 기본 Vagrant 드라이버를 설치합니다.
4. `kitchen init` 명령은 `installpkg`에 YAML 구성 파일 `.kitchen.yml`을 만듭니다. 즐겨찾는 텍스트 편집기에서 파일을 엽니다. `.kitchen.yml` 파일에는 레시피를 실행할 시스템을 지정하는 `platforms` 섹션이 포함되어 있습니다. Test Kitchen은 인스턴스를 생성하고 지정된 레시피를 각 플랫폼에서 실행합니다.

> **Note**
>
> 기본적으로 Test Kitchen은 한 번에 한 플랫폼에서 레시피를 실행합니다. 인스턴스를 생성하는 명령에 `-p` 인수를 추가하면 Test Kitchen은 모든 플랫폼에서 동시에 레시피를 실행합니다.

이 예제에서는 플랫폼 하나로 충분하므로 `.kitchen.yml` 플랫폼을 빼고 `centos-6.4`을 편집합니다. `.kitchen.yml` 파일은 이제 다음과 같아야 합니다.

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
```

```
    - recipe[installpkg::default]
  attributes:
```

Test Kitchen은 `.kitchen.yml` 실행 목록에 있는 레시피만 실행합니다. [*cookbook_name*::*recipe_name*] 형식을 사용하여 레시피를 식별합니다. 여기서 *recipe_name*은 `.rb` 확장명을 제외한 이름입니다. 처음에 `.kitchen.yml` 실행 목록에는 쿡북의 기본 레시피 `installpkg::default`가 포함되어 있습니다. 이 레시피가 구현하려는 레시피이므로 실행 목록을 수정할 필요가 없습니다.

5. `installpkg`의 하위 디렉터리로 `recipes`를 만듭니다.

   쿡북에 레시피가 포함되어 있는 경우 (대부분 그렇죠) 해당 레시피는 `recipes` 하위 디렉터리에 있어야 합니다.

이제 레시피를 쿡북에 추가하고 Test Kitchen을 사용하여 인스턴스에서 레시피를 실행할 수 있습니다.

레시피를 실행하려면

1. 이 단원의 시작 부분에 나온 Git 설치 예제 코드가 포함된 `default.rb` 파일을 만들어 `recipes` 하위 디렉터리에 저장합니다.
2. `installpkg` 디렉터리에서 `kitchen converge`를 실행합니다. 이 명령은 Vagrant에서 새 Ubuntu 인스턴스를 시작하고 인스턴스에 쿡북을 복사하고, Chef 실행을 시작하여 `.kitchen.yml` 실행 목록에서 레시피를 실행합니다.
3. 레시피가 성공적으로 실행되었는지 확인하려면 인스턴스에 대한 SSH 연결을 여는 `kitchen login`을 실행합니다. 그런 다음 `git --version`을 실행하여 Git가 성공적으로 설치되었는지 확인합니다. 워크스테이션으로 돌아가려면 `exit`를 실행합니다.
4. 다 마치면 `kitchen destroy`를 실행해 인스턴스를 종료하세요. 다음 예제에서는 다른 쿡북을 사용합니다.

이 예제는 시작하기에는 좋은 방법이었지만 유독 단순합니다. 다른 패키지는 설치하기가 더 복잡할 수 있습니다. 이 경우, 다음 방법 일부 또는 전부를 수행해야 할 수 있습니다.

- 사용자를 생성하고 구성합니다.
- 데이터, 로그 등을 위한 하나 이상의 디렉터리를 생성합니다.
- 하나 이상의 구성 파일을 설치합니다.
- 운영 체제에 따라 서로 다른 패키지 이름이나 속성 값을 지정합니다.
- 서비스를 시작한 다음 필요하다면 다시 시작합니다.

다음 예제에서는 이러한 문제를 해결하는 방법과 그 밖의 몇 가지 유용한 작업을 설명합니다.

예제 2: 사용자 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

또 다른 간단한 작업은 인스턴스에서의 사용자 관리입니다. 다음 레시피는 Linux 인스턴스에 새 사용자를 추가합니다.

```
user "myuser" do
  home "/home/newuser"
  shell "/bin/bash"
end
```

user 리소스를 사용하여 Linux 시스템과 Windows 시스템에서 사용자를 관리할 수 있습니다(단, 일부 속성은 하나의 시스템에만 적용됩니다). 이 예제는 myuser라는 이름의 사용자를 생성하고 홈 디렉터리 및 셸을 지정합니다. 지정된 작업이 없기 때문에 리소스는 기본 create 작업을 사용합니다. user에 속성을 추가하여 암호나 그룹 ID 같은 다양한 다른 설정을 지정할 수 있습니다. 사용자 설정 수정이나 사용자 삭제 같은 관련된 사용자 관리 작업에 user를 사용할 수도 있습니다. 자세한 정보는 user를 참조하세요.

레시피를 실행하려면

1. opsworks_cookbooks 안에 newuser 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 코드가 포함된 metadata.rb 파일을 만든 다음 이 파일을 newuser에 저장합니다.

   ```
   name "newuser"
   version "0.1.0"
   ```

3. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성하고 recipes 디렉터리 내에서 newuser 디렉터리를 추가합니다.
4. 예제 레시피가 포함된 default.rb 파일을 쿡북의 recipes 디렉터리에 추가합니다.

5. `kitchen converge`를 실행하여 레시피를 실행합니다.
6. `kitchen login`을 사용하여 인스턴스에 로그인하고 `cat /etc/passwd`를 실행하여 새 사용자가 존재하는지 확인합니다. `myuser` 사용자는 파일의 맨 아래에 있어야 합니다.

예제 3: 디렉터리 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스에 패키지를 생성할 때 일부 구성 파일을 생성해 적절한 디렉터리에 넣어야 하는 경우가 많습니다. 하지만 이런 디렉터리가 아직 없을 수 있습니다. 데이터, 로그 파일 등을 위한 디렉터리를 생성해야 할 수도 있습니다. 예를 들어 대부분의 예제에 사용하는 Ubuntu 시스템을 처음 부팅하면 `/srv` 디렉터리에 하위 디렉터리가 없습니다. 애플리케이션 서버를 설치하는 경우, `/srv/www/` 디렉터리 이외에 데이터 파일, 로그 등을 저장하기 위한 몇몇 하위 디렉터리가 필요할 것입니다. 다음 레시피는 인스턴스에 `/srv/www/`를 생성합니다.

```
directory "/srv/www/" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

`directory` 리소스를 사용하여 Linux 시스템과 Windows 시스템에서 디렉터리를 생성하고 구성할 수 있습니다. 단, 일부 속성은 서로 다르게 사용됩니다. 리소스 이름은 리소스의 `path` 속성에 대한 기본값이므로 이 예제에서는 `/srv/www/`를 생성하고 이 디렉터리의 `mode`, `owner` 및 `group` 속성을 지정합니다.

레시피를 실행하려면

1. `opsworks_cookbooks` 안에 `createdir` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.

2. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성하고 `recipes` 디렉터리 내에서 `createdir` 디렉터리를 추가합니다.
3. 레시피 코드가 포함된 `default.rb` 파일을 쿡북의 `recipes` 디렉터리에 추가합니다.
4. `kitchen converge`를 실행하여 레시피를 실행합니다.
5. `kitchen login`을 실행하고, `/srv`로 이동하여 여기에 `www` 하위 디렉터리가 있는지 확인합니다.
6. `exit`를 실행하여 워크스테이션으로 돌아가되 인스턴스를 실행 중인 상태로 둡니다.

> **Note**
>
> 인스턴스에서 홈 디렉터리에 관련된 디렉터리를 생성하려면 `#{ENV['HOME']}`을 사용하여 홈 디렉터리를 나타냅니다. 예를 들어 다음은 `~/shared` 디렉터리를 생성합니다.
>
> ```
> directory "#{ENV['HOME']}/shared" do
>   ...
> end
> ```

`/srv/www/shared`와 같이 더 깊이 중첩된 디렉터리를 만들려 한다고 가정해 봅시다. 앞의 레시피를 다음과 같이 수정할 수 있습니다.

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

레시피를 실행하려면

1. `default.rb`의 코드를 이전 레시피로 바꿉니다.
2. `kitchen converge` 디렉터리에서 `createdir`를 실행합니다.
3. 해당 디렉터리가 생성되었는지 확인하려면 `kitchen login`을 실행하고 `/srv/www`로 이동하여 여기에 `shared` 하위 디렉터리가 있는지 확인합니다.

4. `kitchen destroy`를 실행하여 인스턴스를 종료합니다.

`kitchen converge` 명령이 훨씬 빠르게 실행되는 것을 알 수 있습니다. 그 이유는 인스턴스가 이미 실행되고 있어서 인스턴스를 부팅하고 Chef를 설치하는 등의 작업이 필요 없기 때문입니다. Test Kitchen은 업데이트된 쿡북을 인스턴스에 복사하고 Chef 실행을 시작하면 됩니다.

이제 `kitchen converge`를 다시 실행하면 새로운 인스턴스에서 레시피가 실행됩니다. 이제 다음과 같은 결과를 보게 됩니다.

```
Chef Client failed. 0 resources updated in 1.908125788 seconds
[2014-06-20T20:54:26+00:00] ERROR: directory[/srv/www/shared] (createdir::default line
 1) had an error: Chef::Exceptions::EnclosingDirectoryDoesNotExist: Parent directory /
srv/www does not exist, cannot create /srv/www/shared
[2014-06-20T20:54:26+00:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run
 process exited unsuccessfully (exit code 1)
>>>>>> Converge failed on instance <default-ubuntu-1204>.
>>>>>> Please see .kitchen/logs/default-ubuntu-1204.log for more details
>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: SSH exited (1) for command: [sudo -E chef-solo --config /tmp/kitchen/
solo.rb --json-attributes /tmp/kitchen/dna.json  --log_level info]
>>>>>> ----------------------
```

어떻게 된 걸까요? 문제는 기본적으로 `directory` 리소스가 한 번에 한 디렉터리만 생성할 수 있고 디렉터리의 체인은 생성할 수 없다는 것입니다. 앞에서 레시피가 작동했던 이유는 인스턴스에서 제일 먼저 실행한 레시피가 이미 `/srv/www`를 생성했기 때문에 `/srv/www/shared` 생성으로 하나의 하위 디렉터리만 생성됐기 때문입니다.

> **Note**
>
> `kitchen converge`를 실행할 때는 레시피를 새 인스턴스에서 실행하는지 기존 인스턴스에서 실행하는지 알아야 합니다. 결과가 상이할 수 있습니다.

하위 디렉터리 체인을 생성하려면 `recursive` 속성을 `directory`에 추가하고 `true`로 설정합니다. 다음 레시피는 깨끗한 인스턴스에 `/srv/www/shared` 디렉터리를 생성합니다.

```
directory "/srv/www/shared" do
```

```
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

예제 4: 레시피에 흐름 제어 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

일부 레시피는 일련의 Chef 리소스에 불과합니다. 이 경우, 레시피를 실행하면 레시피는 단순히 각각의 리소스 공급자를 순차적으로 실행합니다. 하지만 보다 복잡한 실행 경로가 있으면 유용한 경우가 많습니다. 다음은 두 가지 일반적인 시나리오입니다.

- 레시피가 같은 리소스를 서로 다른 속성 설정으로 여러 번 실행하도록 하려는 경우.
- 운영 체제에 따라 서로 다른 속성 설정을 사용하려는 경우.

이러한 시나리오는 Ruby 제어 구조를 레시피에 통합하여 해결할 수 있습니다. 이 섹션에서는 예제 3: 디렉터리 생성의 레시피를 수정하여 두 가지 시나리오를 모두 해결하는 방법을 살펴봅니다.

주제

- 반복
- 조건부 논리

반복

예제 3: 디렉터리 생성에서는 `directory` 리소스를 사용하여 디렉터리 또는 디렉터리 체인을 생성하는 방법을 살펴봤습니다. 하지만 `/srv/www/config`와 `/srv/www/shared`라는 2개의 별도의 디렉터리를 생성하려 한다고 가정해 봅시다. 각 디렉터리마다 별도의 디렉터리 리소스를 구성할 수도 있지만 아주 많은 디렉터리를 생성하려는 경우, 이 방법은 번거로울 수 있습니다. 다음 레시피는 이 작업을 처리하는 보다 간단한 방법을 보여 줍니다.

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

이 레시피는 각 하위 디렉터리에 별도의 디렉터리 리소스를 사용하는 대신 하위 디렉터리 경로가 포함된 문자열 모음을 사용합니다. Ruby each 메서드는 첫 번째 모음 요소부터 시작하여 각 모음 요소에 한 번씩 리소스를 실행합니다. 요소의 값은 리소스에서 path 변수에 의해 표시되며, 이 경우에는 디렉터리 경로를 나타냅니다. 이 예제를 쉽게 조정하여 얼마든지 많은 하위 디렉터리를 생성할 수 있습니다.

레시피를 실행하려면

1. createdir 디렉터리를 그대로 열어 두고, 다음 여러 예제에 이 쿡북을 사용합니다.
2. 아직 실행하지 않은 경우 kitchen destroy를 실행합니다. 그러면 깨끗한 인스턴스로 시작할 수 있습니다.
3. default.rb의 코드를 이 예제로 바꾸고 kitchen converge를 실행합니다.
4. 인스턴스에 로그인합니다. 그러면 /srv에 새로 생성된 디렉터리가 있습니다.

해시 테이블을 사용하여 각 반복에 2개의 값을 지정할 수 있습니다. 다음 레시피는 각각 다른 모드로 /srv/www/config와 /srv/www/shared를 생성합니다.

```
{ "/srv/www/config" => 0644, "/srv/www/shared" => 0755 }.each do |path, mode_value|
  directory path do
    mode mode_value
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

레시피를 실행하려면

1. 아직 실행하지 않은 경우 `kitchen destroy`를 실행합니다. 그러면 깨끗한 인스턴스로 시작할 수 있습니다.
2. `default.rb`의 코드를 이 예제로 바꾸고 `kitchen converge`를 실행합니다.
3. 인스턴스에 로그인합니다. 그러면 `/srv`에 지정된 모드로 새로 생성된 디렉터리가 있습니다.

> **Note**
>
> AWS OpsWorks Stacks 레시피는 일반적으로이 접근 방식을 사용하여 기본적으로 큰 해시 테이블인 스택 구성 및 배포 JSON에서 값을 추출하고 리소스에 삽입합니다. 예제는 Deploy 레시피 섹션을 참조하세요.

조건부 논리

Ruby 조건부 논리를 사용하여 여러 실행 분기를 생성할 수도 있습니다. 다음 레시피는 `if-elsif-else` 로직을 사용하여 이전 예제를 확장하므로 `/srv/www/shared`라는 이름의 하위 디렉터리를 생성하지만 Debian 및 Ubuntu 시스템에서만 가능합니다. 다른 모든 시스템의 경우, Test Kitchen 출력에 표시되는 오류 메시지를 로깅합니다.

```
if platform?("debian", "ubuntu")
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

예제 레시피를 실행하려면

1. 인스턴스가 여전히 실행 중인 경우 `kitchen destroy`를 실행하여 종료합니다.
2. `default.rb`의 코드를 이 예제 코드로 바꿉니다.

3. `.kitchen.yml`을 편집하여 CentOS 6.4 시스템을 플랫폼 목록에 추가합니다. 파일의 `platforms` 섹션은 다음과 같아야 합니다.

```
...
platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
...
```

4. `kitchen converge`를 실행합니다. 이 코드는 인스턴스를 생성하고 각 플랫폼에 대해 `.kitchen.yml`의 레시피를 순서대로 실행합니다.

> **Note**
>
> 인스턴스를 하나만 수렴하려는 경우 인스턴스 이름을 파라미터로 추가합니다. 예를 들어, Ubuntu 플랫폼에서만 레시피를 수렴하려면 `kitchen converge default-ubuntu-1204`를 실행합니다. 플랫폼 이름을 모르는 경우에는 `kitchen list`를 실행하면 됩니다.

Test Kitchen 출력의 CentOS 부분에 다음과 같은 로그 메시지가 표시되어야 합니다.

```
...
Converging 1 resources
Recipe: createdir::default
* log[Unsupported system] action write[2014-06-23T19:10:30+00:00] INFO: Processing
 log[Unsupported system] action write (createdir::default line 12)
[2014-06-23T19:10:30+00:00] INFO: Unsupported system

[2014-06-23T19:10:30+00:00] INFO: Chef Run complete in 0.004972162 seconds
```

이제 인스턴스에 로그인하여 디렉터리가 생성되었는지 여부를 확인할 수 있습니다. 다만 지금은 `kitchen login`을 실행할 수 없습니다. 플랫폼 이름(예: `kitchen login default-ubuntu-1204`)을 추가하여 어떤 인스턴스인지 지정해야 합니다.

> **Note**
>
> Test Kitchen 명령이 인스턴스 이름을 받는 경우, 완전한 이름을 입력할 필요가 없습니다. Test Kitchen은 인스턴스 이름을 Ruby 정규식으로 취급하므로 고유한 일치를 제공하기에 충분한 문자만 있으면 됩니다. 예를 들어 `kitchen converge ub`를 실행하여 Ubuntu 인스턴스만 수렴하거나 `kitchen login 64`를 실행하여 CentOS 인스턴스에 로그인할 수 있습니다.

아마도 지금 시점에서 의문은 어떻게 레시피가 어느 플랫폼에서 실행 중인지 아느냐일 것입니다. Chef는 플랫폼을 포함한 시스템 데이터를 수집하는 모든 실행에 대해 Ohai라는 도구를 실행하며, 이를 노드 객체라고 하는 구조에서 속성 집합으로 나타냅니다. Chef `platform?` 메서드는 괄호로 묶인 시스템들을 Ohai 플랫폼 값과 비교하고 그중 하나가 일치하면 true를 반환합니다.

`node['attribute_name']`을 사용하면 코드에서 직접 노드 속성의 값 단원을 참조할 수 있습니다. 예를 들어 플랫폼 값은 `node['platform']`으로 나타냅니다. 가령 이전 예제를 다음과 같이 작성했을 수 있습니다.

```
if node[:platform] == 'debian' or node[:platform] == 'ubuntu'
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

레시피에 조건부 논리를 포함시키는 일반적 이유는 Linux 제품군마다 가끔 패키지, 디렉터리 등에 사용되는 이름이 다르다는 점을 감안하기 위해서입니다. 예를 들어 Apache 패키지 이름은 CentOS 시스템에서는 `httpd`이고 Ubuntu 시스템에서는 `apache2`입니다.

시스템에 따라 다른 문자열만이 필요한 경우, Chef `value_for_platform` 메서드가 `if-elsif-else`보다 간단한 해법입니다. 다음 레시피는 CentOS 시스템에 `/srv/www/shared` 디렉터리, Ubuntu 시스템에 `/srv/www/data` 디렉터리, 그 밖의 모든 시스템에 `/srv/www/config` 디렉터리를 생성합니다.

```
data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

`value_for_platform`은 `data_dir`에 적절한 경로를 할당하며, `directory` 리소스는 이 값을 사용하여 디렉터리를 생성합니다.

예제 레시피를 실행하려면

1. 인스턴스가 여전히 실행 중인 경우 `kitchen destroy`를 실행하여 종료합니다.
2. `default.rb`의 코드를 이 예제 코드로 바꿉니다.
3. `kitchen converge`를 실행한 다음 각 인스턴스에 로그인하여 해당 디렉터리가 있는지 확인합니다.

예제 5: 속성 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

앞 섹션의 레시피는 플랫폼 이외의 모든 것에 하드코딩된 값을 사용했습니다. 이 방법은 예컨대 둘 이상의 레시피에 동일한 값을 사용하려는 경우, 불편할 수 있습니다. 쿡북에 속성 파일을 포함시키면 레시피와 별도로 값을 정의할 수 있습니다.

속성 파일은 하나 이상의 속성에 값을 할당하는 Ruby 애플리케이션입니다. 속성 파일은 쿡북의 `attributes` 폴더에 있어야 합니다. Chef는 속성을 노드 객체에 통합하며, 모든 레시피는 속성 단원을

참조하여 속성 값을 사용할 수 있습니다. 이 주제에서는 반복의 레시피를 수정하여 속성을 사용하는 방법을 살펴봅니다. 다음은 참조용 원래 레시피입니다.

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

다음은 하위 디렉터리 이름, 모드, 소유자 및 그룹 값에 대해 속성을 정의합니다.

```
default['createdir']['shared_dir'] = 'shared'
default['createdir']['config_dir'] = 'config'
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

다음 사항에 유의하세요.

- 각 정의는 속성 유형으로 시작합니다.

  속성이 여러 번 정의된 경우(아마도 다른 속성 파일에서) 속성 유형에 따라 속성의 우선 순위가 지정되며, 이에 따라 노드 개체에 통합되는 정의가 결정됩니다. 자세한 내용은 속성 우선 순위 섹션을 참조하세요. 이 예제의 모든 정의는 이 목적에 일반적인 유형인 `default` 속성 유형을 가지고 있습니다.
- 속성에는 중첩된 이름이 있습니다.

  기본적으로 노드 객체는 임의로 깊이 중첩될 수 있는 해시 테이블이므로 속성 이름은 중첩이 가능하고 일반적으로 중첩됩니다. 이 속성 파일은 쿡북 이름인 `createdir`이 있는 중첩된 이름을 첫 번째 요소로 사용하는 표준 관행을 따릅니다.

createdir을 속성의 첫 번째 요소로 사용하는 이유는 Chef 실행 시 Chef가 모든 쿡북의 속성을 노드 객체에 통합하기 때문입니다. Stacks를 사용하면 노드 객체에 사용자가 정의한 속성 외에도 내장 AWS OpsWorks 쿡북의 많은 속성이 포함됩니다. https://github.com/aws/opsworks-cookbooks 속성 이름

에 쿡북 이름을 포함시키면 특히 속성이 `port` 또는 `user` 같은 이름을 가지고 있는 경우, 다른 쿡북의 속성과의 이름 충돌 위험이 줄어듭니다. 속성 값을 재정의하려는 경우가 아니라면 속성 이름을 `[:apache2][:user]`와 같이 명명하지 마십시오. 자세한 내용은 사용자 지정 쿡북 속성 사용 섹션을 참조하세요.

다음 예제는 하드코딩된 값 대신 속성을 사용하여 원래 레시피를 보여 줍니다.

```
[ "/srv/www/#{node['createdir']['shared_dir']}", "/srv/www/#{node['createdir']
['config_dir']}" ].each do |path|
  directory path do
    mode node['createdir']['mode']
    owner node['createdir']['owner']
    group node['createdir']['group']
    recursive true
    action :create
  end
end
```

> **Note**
>
> 속성 값을 문자열에 통합하려면 `#{}`으로 묶습니다. 이전 예제에서 `#{node['createdir']['shared_dir']}`은 "shared"를 "/srv/www/"에 추가합니다.

레시피를 실행하려면

1. `kitchen destroy`를 실행하여 깨끗한 인스턴스로 시작합니다.
2. `recipes/default.rb`의 코드를 이전 레시피 예제로 바꿉니다.
3. `createdir`의 하위 디렉터리로 `attributes`를 만들고 속성 정의가 포함된 `default.rb` 파일을 추가합니다.
4. `.kitchen.yml`을 편집하여 플랫폼 목록에서 CentOS를 제거합니다.
5. `kitchen converge`를 실행한 다음 인스턴스에 로그인하여 `/srv/www/shared` 및 `/srv/www/config` 디렉터리가 있는지 확인합니다.

> **Note**
>
> AWS OpsWorks Stacks를 사용하면 값을 속성으로 정의하면 추가 이점이 있습니다. 사용자 지정 JSON을 사용하여 스택별 또는 배포별로 해당 값을 재정의할 수 있습니다. 이는 다음을 비롯한 다양한 목적에 유용할 수 있습니다.
>
> - 쿡북을 수정할 필요 없이 구성 설정이나 사용자 이름 등 레시피의 동작을 사용자 지정할 수 있습니다.
>
> 예를 들어 서로 다른 스택에 같은 쿡북을 사용하고 사용자 지정 JSON을 사용하여 특정 스택의 주요 구성 설정을 지정할 수 있습니다. 이렇게 하면 쿡북을 수정하거나 스택마다 다른 쿡북을 사용하는 데 드는 시간과 노력이 줄어듭니다.
> - 데이터베이스 암호와 같이 잠재적으로 중요한 정보를 쿡북 리포지토리에 넣지 않아도 됩니다.
>
> 그 대신 속성을 사용하여 기본값을 정의한 다음 사용자 지정 JSON을 사용하여 해당 값을 실제 값으로 재정의합니다.
>
> 사용자 지정 JSON을 사용하여 속성을 재정의하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

속성 파일은 다소 간단한 Ruby 애플리케이션이기 때문에 `default.rb`로 명명됩니다. 즉, 예컨대 조건부 논리를 사용하여 운영 체제를 기반으로 속성 값을 지정할 수 있습니다. 조건부 논리에서는 레시피의 다양한 Linux 제품군에 다양한 하위 디렉터리 이름을 지정했습니다. 속성 파일이 있으면 그 대신 조건부 논리를 속성 파일에 넣을 수 있습니다.

다음 속성 파일은 `value_for_platform`을 사용하여 운영 체제에 따라 다른 `['shared_dir']` 속성 값을 지정합니다. 다른 조건의 경우, Ruby `if-elsif-else` 논리 또는 `case` 문을 사용할 수 있습니다.

```
data_dir = value_for_platform(
  "centos" => { "default" => "shared" },
  "ubuntu" => { "default" => "data" },
  "default" => "user_data"
)
default['createdir']['shared_dir'] = data_dir
default['createdir']['config_dir'] = "config"
default['createdir']['mode'] = 0755
```

```
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

레시피를 실행하려면

1. `kitchen destroy`를 실행하여 깨끗한 인스턴스로 시작합니다.
2. `attributes/default.rb`의 코드를 이전 예제로 바꿉니다.
3. `.kitchen.yml` 단원에서 설명한 대로 조건부 논리을 편집하여 플랫폼 섹션에 CentOS 플랫폼을 추가합니다.
4. `kitchen converge`를 실행한 다음 인스턴스에 로그인하여 디렉터리가 있는지 확인합니다.

다 마치면 `kitchen destroy`를 실행해 인스턴스를 종료하세요. 다음 예제는 새 쿡북을 사용합니다.

예제 6: 파일 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

디렉터리를 생성한 후 구성 파일, 데이터 파일 등으로 디렉터리를 채워야 하는 경우가 많습니다. 이 주제에서는 인스턴스에 파일을 설치하는 두 가지 방법을 살펴봅니다.

주제

- 쿡북에서 파일 설치
- 템플릿에서 파일 생성

쿡북에서 파일 설치

인스턴스에 파일을 설치하는 가장 간단한 방법은 `cookbook_file` 리소스를 사용하는 것입니다. 이 방법은 Linux와 Windows 시스템 모두 쿡북에서 파일을 인스턴스의 지정된 위치로 복사합니다. 이 예제는 예제 3: 디렉터리 생성 단원의 레시피를 확장해 디렉터리 생성 후 데이터 파일을 `/srv/www/shared`에 추가합니다. 다음은 참조용 원래 레시피입니다.

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

쿡북을 설정하려면

1. opsworks_cookbooks 디렉터리 안에 createfile 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. createfile에 다음 콘텐츠가 포함된 metadata.rb 파일을 추가합니다.

   ```
   name "createfile"
   version "0.1.0"
   ```

3. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성하고 platforms 목록에서 CentOS를 제거합니다.
4. recipes 하위 디렉터리를 createfile에 추가합니다.

설치할 파일에는 다음 JSON 데이터가 포함됩니다.

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

데이터 파일을 설정하려면

1. files 하위 디렉터리는 createfile에, default 하위 디렉터리는 files에 추가합니다. cookbook_file을 사용하여 설치한 모든 파일은 files의 하위 디렉터리에 있어야 하며, 예를 들어 이 예에서는 files/default입니다.

> **Note**
>
> 시스템마다 다른 파일을 지정하려면 시스템의 이름을 딴 하위 폴더(예: `files/ubuntu`)에 각 시스템별 파일을 저장할 수 있습니다. `cookbook_file` 리소스는 해당하는 시스템별 파일이 있는 경우 복사하고 없는 경우 `default` 파일을 사용합니다. 자세한 정보는 cookbook_file 단원을 참조하세요.

2. 이전 예제의 JSON을 사용하여 `example_data.json` 파일을 만들어 `files/default`에 추가합니다.

다음 레시피는 `example_data.json`을 지정된 위치로 복사합니다.

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

디렉터리 리소스가 `/srv/www/shared`를 생성한 후 `cookbook_file` 리소스가 `example_data.json`을 해당 디렉터리로 복사하고 파일의 사용자, 그룹, 모드도 설정합니다.

> **Note**
>
> `cookbook_file` 리소스는 `create_if_missing`이라는 새로운 작업을 도입합니다. `create` 작업도 사용할 수 있지만 이 작업은 기존 파일을 덮어씁니다. 아무것도 덮어쓰지 않으려면 아직 존재하지 않는 경우에 한해 `create_if_missing`을 설치하는 `example_data.json`을 사용하세요.

레시피를 실행하려면

1. `kitchen destroy`를 실행하여 깨끗한 인스턴스로 시작합니다.
2. 이전 레시피가 포함된 `default.rb` 파일을 만든 다음 이 파일을 `recipes`에 저장합니다.
3. `kitchen converge`를 실행한 다음 인스턴스에 로그인하여 `/srv/www/shared` 디렉터리에 `example_data.json`이 포함되어 있는지 확인합니다.

템플릿에서 파일 생성

`cookbook_file` 리소스는 일부 목적에는 유용하지만 쿡북에 있는 파일이라면 무엇이든 설치합니다. `template` 리소스는 템플릿에서 동적으로 파일을 생성함으로써 Windows 또는 Linux 인스턴스에 파일을 설치하는 보다 유연한 방법을 제공합니다. 그러면 실행 시간에 파일의 콘텐츠의 세부 정보를 확인하고 필요에 따라 변경할 수 있습니다. 예를 들어 인스턴스를 시작할 때는 구성 파일이 특정 설정을 갖고 나중에 스택에 더 많은 인스턴스를 추가할 때 이 설정을 수정하기를 원할 수 있습니다.

이 예제는 `createfile` 리소스를 사용하여 약간 수정된 `template` 버전을 설치하도록 `example_data.json` 쿡북을 수정합니다.

설치된 파일은 다음과 같이 보입니다.

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true,
  "a_string" : "some string",
  "platform" : "ubuntu"
}
```

템플릿 리소스는 일반적으로 속성 파일과 함께 사용되므로 예제에서는 속성 파일을 사용하여 다음 값을 지정합니다.

```
default['createfile']['my_name'] = 'myname'
default['createfile']['your_name'] = 'yourname'
default['createfile']['install_file'] = true
```

쿡북을 설정하려면

1. `createfile` 쿡북의 `files` 디렉터리와 포함된 내용을 모두 삭제합니다.
2. `attributes` 하위 디렉터리를 `createfile`에 추가하고, `default.rb` 파일을 이전 속성 정의가 포함된 `attributes`에 추가합니다.

템플릿은 기본적으로 콘텐츠 일부가 자리 표시자에 의해 표시되는 최종 파일의 사본인 `.erb` 파일입니다. `template` 리소스는 파일을 생성할 때 템플릿의 콘텐츠를 지정된 파일에 복사하고 자리 표시자를 할당된 값으로 덮어씁니다. 다음은 `example_data.json`에 대한 템플릿입니다.

```
{
  "my_name" : "<%= node['createfile']['my_name'] %>",
  "your_name" : "<%= node['createfile']['your_name'] %>",
  "a_number" : 42,
  "a_boolean" : <%= @a_boolean_var %>,
  "a_string" : "<%= @a_string_var %>",
  "platform" : "<%= node['platform'] %>"
}
```

`<%=...%>` 값은 자리 표시자입니다.

- `<%=node[...]%>`은 노드 속성 값을 나타냅니다.

  이 예제에서 "your_name" 값은 쿡북 속성 파일의 속성 값 중 하나를 나타내는 자리 표시자입니다.
- `<%=@...%>`는 간략히 설명한 것처럼 템플릿 리소스에서 정의되는 변수의 값을 나타냅니다.

템플릿 파일을 생성하려면

1. `templates` 하위 디렉터리는 `createfile` 쿡북에, `default` 하위 디렉터리는 `templates`에 추가합니다.

> **Note**
>
> `templates` 디렉터리는 `files` 디렉터리처럼 작동합니다. 시스템별 템플릿은 시스템 이름을 딴 하위 디렉터리(예: ubuntu)에 저장할 수 있습니다. `template` 리소스는 해당하는 시스템별 템플릿이 있는 경우 사용하고 없는 경우 `default` 템플릿을 사용합니다.

2. example_data.json.erb라는 파일을 만들어 templates/default 디렉터리에 저장합니다. 템플릿 이름은 임의적이지만 일반적으로 파일 이름에 확장명을 포함해 .erb를 추가하여 템플릿 이름을 생성합니다.

다음 레시피는 template 리소스를 사용하여 /srv/www/shared/example_data.json을 생성합니다.

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
end
```

template 리소스는 템플릿에서 example_data.json을 생성하여 /srv/www/shared에 설치합니다.

- 템플릿 이름 /srv/www/shared/example_data.json은 설치된 파일의 경로와 이름을 지정합니다.
- source 속성은 파일 생성에 사용되는 템플릿을 지정합니다.
- mode 속성은 설치된 파일의 모드를 지정합니다.
- 리소스는 a_boolean_var와 a_string_var라는 두 가지 변수를 정의합니다.

  리소스는 example_data.json을 생성할 때 템플릿의 변수 자리 표시자를 리소스의 해당 값으로 덮어씁니다.
- only_if guard 속성은 ['createfile']['install_file']이 true로 설정되는 경우에만 파일을 생성하라고 리소스에 명령합니다.

레시피를 실행하려면

1. `kitchen destroy`를 실행하여 깨끗한 인스턴스로 시작합니다.
2. `recipes/default.rb`의 코드를 이전 예제로 바꿉니다.
3. `kitchen converge`를 실행한 다음 인스턴스에 로그인하여 파일이 `/srv/www/shared` 디렉터리에 있고 올바른 콘텐츠를 포함하고 있는지 확인합니다.

다 마치면 `kitchen destroy`를 실행해 인스턴스를 종료하세요. 다음 섹션에서는 새 쿡북을 사용합니다.

예제 7: 명령 및 스크립트 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 리소스는 인스턴스에서 다양한 작업을 처리할 수 있지만 경우에 따라 셸 명령이나 스크립트를 사용하는 것이 좋습니다. 예를 들어 특정 작업을 수행하는 데 사용하는 스크립트가 이미 있고 새 코드를 구현하는 것보다 이 스크립트를 계속 사용하는 것이 더 쉬울 수 있습니다. 이 섹션에서는 인스턴스에서 명령 또는 스크립트를 실행하는 방법을 살펴봅니다.

주제

- 명령 실행
- 스크립트 실행

명령 실행

`script` 리소스는 하나 이상의 명령을 실행합니다. 이 리소스는 csh, bash, Perl, Python 및 Ruby 명령 인터프리터를 지원하므로 적절한 인터프리터가 설치되어 있다면 Linux 또는 Windows 시스템에서 사용할 수 있습니다. 이 주제에서는 Linux 인스턴스에서 간단한 bash 명령을 실행하는 방법을 살펴봅니다. Chef는 Windows에서 스크립트를 실행하기 위해 powershell_script 및 batch 리소스도 지원합니다. 자세한 내용은 Windows PowerShell 스크립트 실행 섹션을 참조하세요.

시작하기

1. opsworks_cookbooks 디렉터리 안에 script 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. script에 다음 콘텐츠가 포함된 metadata.rb 파일을 추가합니다.

```
name "script"
version "0.1.0"
```

3. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성하고 platforms 목록에서 CentOS를 제거합니다.
4. script 안에 recipes 디렉터리를 만듭니다.

script 리소스 자체를 사용하여 명령을 실행할 수도 있지만 Chef는 인터프리터에 대해 명명되는 리소스의 명령 인터프리터별 버전 집합도 지원합니다. 다음 레시피는 bash 리소스를 사용하여 간단한 bash 스크립트를 실행합니다.

```
bash "install_something" do
  user "root"
  cwd "/tmp"
  code <<-EOH
    touch somefile
  EOH
  not_if do
    File.exists?("/tmp/somefile")
  end
end
```

bash 리소스는 다음과 같이 구성됩니다.

- bash 리소스는 code 블록에서 명령을 실행하는 기본 작업인 run을 사용합니다.

  이 예제에는 touch somefile이라는 명령 하나만 있지만 code 블록에는 여러 명령이 포함될 수 있습니다.
- user 속성은 명령을 실행하는 사용자를 지정합니다.
- cwd 속성은 작업 디렉터리를 지정합니다.

이 예제에서는 touch가 /tmp 디렉터리에 파일을 생성합니다.

- not_if guard 속성은 이 파일이 이미 존재하는 경우, 리소스에게 아무 작업도 하지 말라고 명령합니다.

레시피를 실행하려면

1. 이전 예제 코드가 포함된 default.rb 파일을 만든 다음 이 파일을 recipes에 저장합니다.
2. kitchen converge를 실행한 다음 인스턴스에 로그인하여 /tmp 디렉터리에 이 파일이 포함되어 있는지 확인합니다.

스크립트 실행

script 리소스는 특히 하나 또는 두 개의 명령만 실행해야 하는 경우에 편리하지만 대체로 스크립트를 파일에 저장하고 파일을 실행하는 것이 더 좋습니다. execute 리소스는 Linux 또는 Windows에서 스크립트 파일을 포함한 지정된 실행 파일을 실행합니다. 이 주제에서는 script를 사용하여 간단한 셸 스크립트를 실행하도록 앞의 예제의 execute 쿡북을 수정합니다. 더 복잡한 스크립트나 다른 실행 파일 유형까지 예제를 쉽게 확장할 수 있습니다.

스크립트 파일을 설정하려면

1. files 하위 디렉터리는 script에, default 하위 디렉터리는 files에 추가합니다.
2. 다음 코드가 포함된 touchfile이라는 파일을 만들어 files/default에 추가합니다. 이 예제에서는 일반적인 Bash 인터프리터 라인이 사용되지만 필요한 경우 사용자의 셸 환경에서 작동하는 인터프리터를 대체합니다.

   ```
   #!/usr/bin/env bash
   touch somefile
   ```

   스크립트 파일은 명령을 제한 없이 포함할 수 있습니다. 편의를 위해 이 예제 스크립트에는 touch 명령만 있습니다.

다음 레시피는 스크립트를 실행합니다.

```
cookbook_file "/tmp/touchfile" do
  source "touchfile"
  mode 0755
```

```
end

execute "touchfile" do
  user "root"
  cwd "/tmp"
  command "./touchfile"
end
```

cookbook_file 리소스는 스크립트 파일을 /tmp에 복사하고 파일을 실행 파일로 만들도록 모드를 설정합니다. 그런 다음 execute 리소스가 다음과 같이 파일을 실행합니다.

- user 속성은 명령의 사용자(이 예제에서는 root)를 지정합니다.
- cwd 속성은 작업 디렉터리(이 예제에서는 /tmp)를 지정합니다.
- command 속성은 작업 디렉터리에 위치한 실행할 스크립트(이 예제에서는 touchfile)를 지정합니다.

레시피를 실행하려면

1. recipes/default.rb의 코드를 이전 예제로 바꿉니다.
2. kitchen converge를 실행한 다음 인스턴스에 로그인하여 /tmp 디렉터리에 모드가 0755로 설정된 스크립트 파일과 somefile 디렉터리 파일이 있는지 확인합니다.

다 마치면 kitchen destroy를 실행해 인스턴스를 종료하세요. 다음 섹션에서는 새 쿡북을 사용합니다.

예제 8: 서비스 관리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

애플리케이션 서버와 같은 패키지에는 일반적으로 시작, 중지, 재시작해야 하는 연결된 서비스가 있습니다. 예를 들어 패키지를 설치한 후 또는 인스턴스 부팅이 완료된 후에 Tomcat 서비스를 시작하고 구성 파일을 수정할 때마다 서비스를 다시 시작해야 합니다. 이 주제에서는 Tomcat 애플리케이션 서버

를 예제로 사용하여 Linux 인스턴스에서 서비스를 관리하는 방법의 기초를 살펴봅니다. 서비스 리소스는 Windows 인스턴스에서와 거의 비슷한 방식으로 작동하지만 세부적으로는 몇 가지 차이점이 있습니다. 자세한 내용은 `service` 단원을 참조하십시오.

> **Note**
>
> 예제에서는 `service` 리소스 사용 방법의 기초를 보여 주기에 충분한 정도로만 Tomcat을 최소 설치합니다. 기능이 더 많은 Tomcat 서버를 위한 레시피 구현 방법의 예제는 사용자 지정 Tomcat 서버 계층 생성 단원을 참조하세요.

주제

- 서비스 정의 및 시작
- notifies를 사용하여 서비스 시작 또는 재시작

서비스 정의 및 시작

이 섹션에서는 서비스를 정의하고 시작하는 방법의 기초를 살펴봅니다.

시작하기

1. `opsworks_cookbooks` 디렉터리 안에 `tomcat` 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. `tomcat`에 다음 콘텐츠가 포함된 `metadata.rb` 파일을 추가합니다.

   ```
   name "tomcat"
   version "0.1.0"
   ```

3. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성하고 `platforms` 목록에서 CentOS를 제거합니다.
4. `recipes` 하위 디렉터리를 `tomcat`에 추가합니다.

`service` 리소스를 사용하여 서비스를 관리합니다. 다음의 기본 레시피는 Tomcat을 설치하고 서비스를 시작합니다.

```
execute "install_updates" do
```

```
  command "apt-get update"
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :start
end
```

이 레시피는 다음 작업을 수행합니다.

- `execute` 리소스는 `apt-get update`를 실행하여 최신 시스템 업데이트를 설치합니다.

  이 예제에서 사용하는 Ubuntu 인스턴스의 경우, Tomcat을 설치하기 전에 업데이트를 설치해야 합니다. 다른 시스템은 요구 사항이 다를 수 있습니다.
- `package` 리소스는 Tomcat 7을 설치합니다.
- 포함된 `tomcat::service` 레시피는 서비스를 정의하며, 나중에 설명합니다.
- `service` 리소스는 Tomcat 서비스를 시작합니다.

  이 리소스를 사용하여 서비스 중단 및 재시작 같은 다른 명령을 실행할 수도 있습니다.

다음 예제는 `tomcat::service` 레시피를 보여 줍니다.

```
service 'tomcat' do
  service_name "tomcat7"
  supports :restart => true, :reload => false, :status => true
  action :nothing
end
```

이 레시피는 다음과 같이 Tomcat 서비스 정의를 생성합니다.

- 리소스 이름인 `tomcat`은 다른 레시피가 서비스를 참조하는 데 사용합니다.

  예를 들어 `default.rb`는 서비스를 시작하기 위해 `tomcat` 단원을 참조합니다.
- `service_name` 리소스는 서비스 이름을 지정합니다.

인스턴스에서 서비스를 나열할 때 Tomcat 서비스는 tomcat7으로 명명됩니다.

- `supports`는 Chef가 서비스의 `restart`, `reload` 및 `status` 명령을 관리하는 방법을 지정합니다.
  - `true`는 Chef가 init 스크립트 또는 그 밖의 서비스 공급자를 사용하여 명령을 실행할 수 있음을 나타냅니다.
  - `false`는 Chef가 다른 수단을 사용하여 명령을 실행하려 시도해야 함을 나타냅니다.

`action`이 `:nothing`으로 설정되어 있어 아무 작업도 하지 말라고 리소스에게 명령한다는 점에 유의하세요. 서비스 리소스는 `start` 및 `restart`와 같은 작업을 지원합니다. 하지만 이 쿡북은 아무 작업도 하지 않는 서비스 정의를 사용하고 다른 곳에서 서비스를 시작하거나 다시 시작하는 표준 관행에 따릅니다. 서비스를 시작하거나 다시 시작하는 각각의 레시피는 먼저 서비스를 정의해야 하므로 가장 간단한 방법은 서비스 정의를 별도의 레시피에 넣고 필요에 따라 다른 레시피에 포함시키는 것입니다.

> **Note**
>
> 간단히 하기 위해 이 예제의 기본 레시피는 서비스 정의를 실행한 후 `service` 리소스를 사용하여 서비스를 시작합니다. 프로덕션 구현은 일반적으로 뒤에서 설명하는 것처럼 `notifies`를 사용하여 서비스를 시작하거나 다시 시작합니다.

레시피를 실행하려면

1. 기본 레시피 예제가 포함된 `default.rb` 파일을 만든 다음 이 파일을 `recipes`에 저장합니다.
2. 서비스 정의 예제가 포함된 `service.rb` 파일을 만든 다음 이 파일을 `recipes`에 저장합니다.
3. `kitchen converge`를 실행한 다음 인스턴스에 로그인하고 다음 명령을 실행해서 서비스가 실행 중인지 확인하세요.

```
sudo service tomcat7 status
```

> **Note**
>
> `service.rb`를 `default.rb`와 별도로 실행한 경우, `.kitchen.yml`을 편집하여 `tomcat::service`를 실행 목록에 추가해야 합니다. 다만 레시피를 포함시키면 레시피가 실행되기 전에 그 코드가 상위 레시피에 통합됩니다. 따라서 `service.rb`는 `default.rb`의 일부이며 별도의 실행 목록 항목이 필요하지 않습니다.

notifies를 사용하여 서비스 시작 또는 재시작

프로덕션 구현은 일반적으로 서비스를 시작하거나 다시 시작하는 데 `service`를 사용하지 않습니다. 그 대신 몇몇 리소스에 `notifies`를 추가합니다. 예를 들어 구성 파일을 수정한 후 서비스를 다시 시작하려면 연결된 `notifies` 리소스에 `template`를 포함시킵니다. `notifies`를 사용하면 `service` 리소스를 사용하여 명시적으로 서비스를 다시 시작하는 방법에 비해 다음과 같은 장점이 있습니다.

- `notifies` 요소는 연결된 구성 파일이 변경된 경우에만 서비스를 다시 시작하므로 불필요한 서비스 재시작을 초래할 위험이 없습니다.
- Chef는 실행에 얼마나 많은 `notifies`가 포함되는지에 상관없이 각 실행이 끝날 때 많아야 한 번 서비스를 다시 시작합니다.

  예를 들어 Chef 실행에는 각각 서로 다른 구성 파일을 수정하고 파일이 변경된 경우 서비스를 다시 시작해야 하는 여러 개의 템플릿 리소스가 포함될 수 있습니다. 하지만 사용자는 일반적으로 Chef 실행이 끝날 때 한 번만 서비스를 다시 시작하려 합니다. 그렇지 않다면 이전의 재시작으로 완전히 작동하지 않아 오류를 발생시킬 수 있는 서비스를 다시 시작하려 시도할 수 있습니다.

이 예제는 `tomcat::default`를 사용하여 서비스를 다시 시작하는 `template` 리소스를 포함하도록 `notifies`를 수정합니다. 현실적 예에서는 Tomcat 구성 파일 중 하나의 사용자 지정 버전을 생성하는 템플릿 리소스를 사용하겠지만 과정이 길고 복잡합니다. 간단히 하기 위해 이 예제는 템플릿에서 파일 생성의 템플릿 리소스를 사용합니다. 이 리소스는 Tomcat과 아무 관련이 없지만 `notifies` 사용 방법을 간단히 보여 줍니다. 템플릿을 사용해 Tomcat 구성 파일을 생성하는 방법의 예제는 설정 레시피 단원을 참조하세요.

쿡북을 설정하려면

1. `templates` 하위 디렉터리는 `tomcat`에, `default` 하위 디렉터리는 `templates`에 추가합니다.
2. `example_data.json.erb` 쿡북에서 `createfile` 디렉터리로 `templates/default` 템플릿을 복사합니다.
3. `attributes`에 `tomcat` 하위 디렉터리를 추가합니다.
4. `default.rb` 쿡북에서 `createfile` 디렉터리로 `attributes` 속성 파일을 복사합니다.

다음 레시피는 `notifies`를 사용하여 Tomcat 서비스를 다시 시작합니다.

```
execute "install_updates" do
  command "apt-get update"
```

```
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
  notifies :restart, resources(:service => 'tomcat')
end
```

이 예제는 템플릿에서 파일 생성의 레시피를 이전 섹션의 레시피에 병합하며, 다음과 같은 두 가지 중요한 변경이 있습니다.

- `service` 리소스는 그대로 있지만 이제는 약간 다른 용도로 사용됩니다.

  `:enable` 작업은 부팅 시 Tomcat 서비스를 활성화합니다.
- 이제 템플릿 리소스에는 `notifies`이 변경된 경우, Tomcat 서비스를 다시 시작하는 `example_data.json`가 포함됩니다.

  이로써 서비스는 Tomcat이 처음 설치될 때 시작되고 구성 파일 변경 후마다 다시 시작됩니다.

레시피를 실행하려면

1. `kitchen destroy`를 실행하여 깨끗한 인스턴스로 시작합니다.
2. `default.rb`의 코드를 이전 예제로 바꿉니다.
3. `kitchen converge`를 실행한 다음 인스턴스에 로그인하여 서비스가 실행 중인지 확인합니다.

> **Note**
>
> 서비스를 다시 시작하고 싶지만 레시피에 `template` 등 `notifies`를 지원하는 리소스가 없는 경우, 그 대신 더미 `execute` 리소스를 사용할 수 있습니다. 예
>
> ```
> execute 'trigger tomcat service restart' do
>   command 'bin/true'
>   notifies :restart, resources(:service => 'tomcat')
> end
> ```
>
> `execute` 리소스는 `command`를 실행하는 방법으로만 사용한다 하더라도 `notifies` 속성이 있어야 합니다. 이 예제에서는 단순히 성공 코드를 반환하는 셸 명령인 `/bin/true`를 실행함으로써 이러한 요구를 해결합니다.

예제 9: Amazon EC2 인스턴스 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

지금까지는 VirtualBox에서 로컬로 인스턴스를 실행했습니다. 이 방법이 빠르고 쉽기는 하지만 결국은 Amazon EC2 인스턴스에서 레시피를 테스트해야 합니다. 특히 Amazon Linux에서 레시피를 실행하려면 Amazon EC2에서만 가능합니다. 예비 구현 및 테스트를 위해 CentOS 같은 비슷한 시스템을 사용할 수 있지만 Amazon Linux에서 레시피를 완전히 테스트하는 유일한 방법은 Amazon EC2 인스턴스를 사용하는 것입니다.

이 주제에서는 Amazon EC2 인스턴스에서 레시피를 실행하는 방법을 살펴봅니다. 다음 두 가지 차이점만 제외하고 Test Kitchen과 Vagrant도 앞 섹션과 거의 같은 방법으로 사용합니다.

- 드라이버가 Vagrant 대신 `kitchen-ec2`입니다.
- Amazon EC2 인스턴스를 시작하는 데 필요한 정보를 사용하여 쿡북의 `.kitchen.yml` 파일을 구성해야 합니다.

> **Note**
>
> 대안적 방법은 `vagrant-aws` Vagrant 플러그인을 사용하는 것입니다. 자세한 정보는 Vagrant AWS 공급자를 참조하세요.

Amazon EC2 인스턴스를 생성하려면 AWS 자격 증명이 필요합니다. AWS 계정이 없는 경우, 다음과 같이 계정을 얻을 수 있습니다.

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

Amazon EC2에 액세스할 수 있는 권한이 있는 IAM 사용자를 생성하고 사용자의 액세스 및 보안 키를 워크스테이션의 안전한 위치에 저장해야 합니다. Test Kitchen은 이 자격 증명을 사용하여 인스턴스를 생성합니다. Test Kitchen에 자격 증명을 제공하는 선호되는 방법은 워크스테이션에서 다음 환경 변수에 키를 할당하는 것입니다.

> **Warning**
>
> IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

- AWS_ACCESS_KEY - 사용자의 액세스 키로,와 비슷합니다AKIAIOSFODNN7EXAMPLE.
- AWS_SECRET_KEY - 사용자의 보안 키로,와 비슷합니다wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY.

이 방법은 예컨대 자격 증명이 포함된 프로젝트를 퍼블릭 리포지토리에 업로드함으로써 뜻하지 않게 계정이 침해될 가능성을 줄입니다.

쿡북을 설정하려면

1. `kitchen-ec2` 드라이버를 사용하려면 시스템에 `ruby-dev` 패키지가 설치되어 있어야 합니다. 다음 예제 명령은 `aptitude`를 사용하여 Ubuntu 시스템에 패키지를 설치하는 방법을 보여줍니다.

   ```
   sudo aptitude install ruby1.9.1-dev
   ```

2. `kitchen-ec2` 드라이버는 다음과 같이 설치할 수 있는 Gem입니다.

   ```
   gem install kitchen-ec2
   ```

워크스테이션에 따라 이 명령에는 sudo가 필요할 수 있습니다. 또는 Ruby 환경 관리자(예: RVM)를 사용할 수도 있습니다. 이 절차는 kitchen-ec2 드라이버 버전 0.8.0에서 테스트했으나 최신 버전이 있습니다. 특정 버전을 설치하려면 gem install kitchen-ec2 -v <*version number*>를 실행합니다.

3. Test Kitchen이 인스턴스에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어를 지정해야 합니다. Amazon EC2 키 페어가 없는 경우 이러한 키 페어를 생성하는 방법은 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다. 이 예에서는 미국 서부(캘리포니아 북부)를 사용합니다.

   키 페어를 선택한 후에는 opsworks_cookbooks의 하위 디렉터리로 ec2_keys를 만들고 해당 키 페어의 프라이빗 키(.pem) 파일을 이 하위 디렉터리에 복사합니다. 프라이빗 키는 시스템의 모든 위치에 저장할 수 있지만 ec2_keys에 저장하면 코드가 간소화되어 편리합니다.

4. createdir-ec2의 하위 디렉터리 opsworks_cookbooks를 만들고 그 디렉터리로 이동합니다.

5. createdir-ec2에 다음 콘텐츠가 포함된 metadata.rb 파일을 추가합니다.

   ```
   name "createdir-ec2"
   version "0.1.0"
   ```

6. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화합니다. 다음 단원에서는 .kitchen.yml을 구성하는 방법에 대해 설명합니다. 이 파일은 Amazon EC2 인스턴스의 경우 훨씬 더 복잡합니다.

7. recipes 하위 디렉터리를 createdir-ec2에 추가합니다.

Amazon EC2용 .kitchen.yml 구성

kitchen-ec2 드라이버가 적절히 구성된 Amazon EC2 인스턴스를 시작하는 데 필요한 정보를 사용하여 .kitchen.yml을 구성합니다. 다음은 미국 서부(캘리포니아 북부) 리전의 Amazon Linux 인스턴스용 .kitchen.yml 파일의 예입니다.

```
driver:
  name: ec2
  aws_ssh_key_id: US-East1
  region: us-west-1
  availability_zone: us-west-1c
  require_chef_omnibus: true
  security_group_ids: sg........
```

```
  subnet_id: subnet-.........
  associate_public_ip: true
  interface: dns

provisioner:
  name: chef_solo

platforms:
  -name: amazon
  driver:
    image_id: ami-xxxxxxxx
  transport:
    username: ec2-user
    ssh_key: ../ec2_keys/US-East1.pem

suites:
  - name: default
    run_list:
      - recipe[createdir-ec2::default]
    attributes:
```

provisioner 및 suites 섹션의 기본 설정을 사용할 수 있지만 기본 driver 및 platforms 설정을 수정해야 합니다. 이 예제에서는 최소한의 설정 목록을 사용하며 나머지에 대해서는 기본값을 사용합니다. 완전한 kitchen-ec2 설정 목록은 Kitchen::Ec2: Amazon EC2용 Test Kitchen 드라이버를 참조하세요.

이 예제는 다음 driver 속성을 설정합니다. 이 예제에서는 앞서 설명한 것처럼 사용자의 액세스 키와 보안 키를 표준 환경 변수에 할당했다고 가정합니다. 드라이버는 기본적으로 이러한 키를 사용합니다. 그렇지 않은 경우, aws_access_key_id와 aws_secret_access_key를 driver 속성에 추가하여 명시적으로 키를 지정하고 적절한 키 값으로 설정해야 합니다.

name

(필수) 이 속성은 ec2로 설정해야 합니다.

aws_ssh_key_id

(필수) 이 예제에서 US-East1으로 명명된 Amazon EC2 SSH 키 페어 이름.

transport.ssh_key

(필수) aws_ssh_key_id에 대해 지정한 키의 프라이빗 키(.pem) 파일. 이 예제에서 이 파일은 US-East1.pem으로 명명되며 ../opsworks/ec2_keys 디렉터리에 있습니다.

리전

(필수) 인스턴스의 AWS 리전. 이 예에서는) 로 `us-west-1` 표시되는 미국 서부(캘리포니아 북부)를 사용합니다.

availability_zone

(선택 사항) 인스턴스의 가용 영역. 이 설정을 생략하면 Test Kitchen은 지정된 리전에 대해 기본 가용 영역(`us-west-1b`의 경우, 미국 서부(캘리포니아 북부))를 사용합니다. 하지만 이 가용 영역은 사용자 계정에서 사용하지 못할 수 있습니다. 이 경우, 가용 영역을 명시적으로 지정해야 합니다. 마침 이 예제를 준비하는 데 사용된 계정이 `us-west-1b`를 지원하지 않기 때문에 예제는 명시적으로 `us-west-1c`를 지정합니다.

require_chef_omnibus

이 설정을 `true`로 설정하면 omnibus installer를 사용하여 모든 플랫폼 인스턴스에 `chef-client`를 설치합니다.

security_group_ids

(선택 사항) 인스턴스에 적용할 보안 그룹 ID 목록. 이 설정은 `default` 보안 그룹을 인스턴스에 적용합니다. 보안 그룹 수신 규칙이 인바운드 SSH 연결을 허용하도록 하세요. 그렇지 않으면 Test Kitchen이 인스턴스와 통신할 수 없습니다. `default` 보안 그룹을 사용하는 경우, 그에 맞게 편집해야 할 수 있습니다. 자세한 내용은 Amazon EC2 보안 그룹을 참조하세요.

subnet_id

인스턴스의 대상 서브넷 ID(해당되는 경우).

associate_public_ip

인터넷에서 인스턴스에 액세스할 수 있도록 Amazon EC2 인스턴스가 해당 인스턴스에 퍼블릭 IP 주소를 연결하도록 할 수 있습니다.

인터페이스

인스턴스에 액세스하는 데 사용하는 호스트 이름 구성 형식. 유효한 값은 `dns`, `public`, `private` 또는 `private_dns`입니다. 이 속성의 값을 지정하지 않으면 `kitchen-ec2`이 다음 순서로 호스트 이름 구성을 설정합니다. 이 속성을 생략하면 구성 형식이 설정되지 않습니다.

1. DNS 이름
2. 퍼블릭 IP 주소
3. 프라이빗 IP 주소
4. 프라이빗 DNS 이름

> **Important**
>
> 액세스 키와 보안 키에 계정 자격 증명을 사용하는 대신 사용자를 생성해 이러한 자격 증명을 Test Kitchen에 제공해야 합니다. 자세한 내용은 AWS 액세스 키 관리 모범 사례를 참조하세요. `.kitchen.yml`을 공용 액세스가 가능한 위치에 저장하지 않도록 주의하세요(예: 퍼블릭 GitHub 또는 Bitbucket 리포지토리에 업로드). 그러면 자격 증명이 노출되고 계정의 보안이 훼손될 수 있습니다.

`kitchen-ec2` 드라이버는 다음 플랫폼에 대한 기본 지원을 제공합니다.

- ubuntu-10.04
- ubuntu-12.04
- ubuntu-12.10
- ubuntu-13.04
- ubuntu-13.10
- ubuntu-14.04
- centos-6.4
- debian-7.1.0
- windows-2012r2
- windows-2008r2

이런 플랫폼을 하나 이상 사용하려는 경우, 적절한 플랫폼 이름을 `platforms`에 추가합니다. `kitchen-ec2` 드라이버가 자동으로 적절한 AMI를 선택하고 SSH 사용자 이름을 생성합니다. 다른 플랫폼(이 예에서는 Amazon Linux 사용)을 사용할 수 있지만 다음 `platforms` 속성을 명시적으로 지정해야 합니다.

name

플랫폼 이름. 이 예제는 Amazon Linux를 사용하므로 `name`이 `amazon`으로 설정됩니다.

driver

다음을 포함하는 `driver` 속성.

- `image_id` – 플랫폼의 AMI로서 지정된 리전에 속해야 합니다. 이 예제는 미국 서부(캘리포니아 북부) 리전의 Amazon Linux AMI인 `ami-ed8e9284`를 사용합니다.

- `transport.username` – Test Kitchen이 인스턴스와의 통신에 사용할 SSH 사용자 이름.

  Amazon Linux의 경우, `ec2-user`를 사용합니다. 다른 AMI는 사용자 이름이 다를 수 있습니다.

`.kitchen.yml`의 코드를 예제로 바꾸고 `aws_access_key_id`와 같은 계정별 속성에 적절한 값을 할당합니다.

레시피 실행

이 예제는 반복의 레시피를 사용합니다.

레시피를 실행하려면

1. 다음 코드를 사용하여 `default.rb` 파일을 만들어 이 파일을 쿡북의 `recipes` 폴더에 저장합니다.

   ```
   directory "/srv/www/shared" do
     mode 0755
     owner 'root'
     group 'root'
     recursive true
     action :create
   end
   ```

2. `kitchen converge`를 실행하여 레시피를 실행합니다. Amazon EC2 인스턴스를 시작 및 초기화하는 데 필요한 시간 때문에 이전 예제보다 이 명령을 완료하는 데 더 많은 시간이 걸릴 수 있습니다.
3. Amazon EC2 콘솔로 이동하여 미국 서부(캘리포니아 북부) 리전을 선택하고 탐색 창에서 인스턴스를 클릭합니다. 목록에 새로 생성된 인스턴스가 보입니다.
4. VirtualBox에서 실행 중인 인스턴스에 대해 수행한 것처럼 `kitchen login`을 실행하여 인스턴스에 로그인합니다. `/srv`에 새로 생성된 디렉터리가 있습니다. 이제 즐겨 사용하는 SSH 클라이언트를 사용하여 인스턴스에 연결할 수 있습니다.

다음 단계

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 장에서 Chef 쿡북 구현 방법의 기초를 살펴봤지만 아직 많은 내용이 남아 있습니다.

- 예제에서 보다 일반적으로 사용되는 몇몇 리소스를 사용하는 방법을 보여 주었지만 아직 다루지 않은 내용이 많습니다.

  우리가 다룬 리소스의 경우, 예제에서는 사용할 수 있는 속성과 작업 일부만이 사용됐습니다. 완전한 참조는 리소스 및 공급자에 대하여를 참조하세요.
- 예제는 핵심 쿡북 요소인 `recipes`, `attributes`, `files`, `templates`만을 사용했습니다.

  쿡북에는 `libraries`, `definitions`, `specs`와 같은 그 밖의 다양한 요소도 포함될 수 있습니다. 자세한 정보는 Chef 설명서를 참조하세요.
- 예제에서는 인스턴스를 시작하고, 레시피를 실행하며, 인스턴스에 로그인하는 편리한 방법으로만 Test Kitchen을 사용했습니다.

  Test Kitchen은 기본적으로 레시피에서 다양한 테스트를 실행할 수 있는 테스트 플랫폼입니다. 아직 살펴보지 않았다면 Test Kitchen의 테스트 기능을 소개하는 Test Kitchen walkthrough를 살펴보십시오.
- AWS OpsWorks Stacks용 쿡북 구현는 몇 가지 고급 예제를 제공하며 Stacks용 AWS OpsWorks 쿡북을 구현하는 방법을 보여줍니다.

AWS OpsWorks Stacks용 쿡북 구현

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

쿡북 기본 사항 섹션에서는 쿡북 및 레시피에 대해 소개했습니다. 이 섹션의 예제는 설계상 간단했으며 AWS OpsWorks Stacks 인스턴스를 포함하여 Chef를 지원하는 모든 인스턴스에서 작동합니다. Stacks에 보다 정교한 AWS OpsWorks 쿡북을 구현하려면 일반적으로 여러 가지 면에서 표준 Chef와 다른 AWS OpsWorks Stacks 환경을 최대한 활용해야 합니다.

이 주제에서는 AWS OpsWorks Stacks 인스턴스용 레시피 구현의 기본 사항에 대해 설명합니다.

> **Note**
>
> 쿡북을 구현하는 방법을 잘 알지 못하는 경우 쿡북 기본 사항 섹션부터 시작해야 합니다.

주제

- AWS OpsWorks Stacks Linux 인스턴스에서 레시피 실행
- Windows 인스턴스에서 레시피 실행
- Windows PowerShell 스크립트 실행
- Vagrant에서 스택 구성 및 배포 속성 모의
- 스택 구성 및 배포 속성 값 사용
- Linux 인스턴스에서 외부 쿡북 사용: Berkshelf
- SDK for Ruby 사용: Amazon S3에서 파일 다운로드
- Windows 소프트웨어 설치
- 내장 속성 재정의
- 내장 템플릿 재정의

AWS OpsWorks Stacks Linux 인스턴스에서 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Test Kitchen과 Vagrant는 쿡북을 구현하는 간단하고 효율적인 방법을 제공하지만 쿡북의 레시피가 프로덕션 환경에서 올바르게 실행되는지 확인하려면 AWS OpsWorks Stacks 인스턴스에서 실행해야 합니다. 이 주제에서는 AWS OpsWorks Stacks Linux 인스턴스에 사용자 지정 쿡북을 설치하고 간단한 레시피를 실행하는 방법을 설명합니다. 또한 효율적으로 레시피 버그를 수정하기 위한 몇 가지 팁도 제공합니다.

Windows 인스턴스에서 레시피를 실행하는 방법에 대한 설명은 Windows 인스턴스에서 레시피 실행 단원을 참조하세요.

주제

- 레시피 생성 및 실행
- 자동으로 레시피 실행
- 레시피 문제 해결 및 수정

레시피 생성 및 실행

먼저 스택을 생성해야 합니다. 다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택 생성

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 클릭합니다.
2. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 클릭합니다.

   - 이름 – OpsTest
   - 기본 SSH 키 – Amazon EC2 키 페어

   Amazon EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다. 이 예에서는 기본 미국 서부(오레곤) 리전을 사용합니다.
3. [계층 추가]를 클릭하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – OpsTest

- 짧은 이름 - opstest

모든 계층 유형이 실제로 Linux 스택에서 작동하지만 이 예제에서는 다른 계층 유형에서 설치한 패키지가 필요하지 않기 때문에 사용자 지정 계층을 사용하는 것이 가장 간단합니다.

4. 기본 설정을 사용하여 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

인스턴스가 시작되는 동안(보통 몇 분 정도 소요됨) 쿡북을 생성할 수 있습니다. 이 예제에서는 조건부 논리을 약간 수정한 버전을 사용합니다. 이는 플랫폼에 따라 명명된 데이터 디렉터리를 생성합니다.

쿡북을 설정하려면

1. opsworks_cookbooks 안에 opstest 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 opstest에 저장합니다.

```
name "opstest"
version "0.1.0"
```

3. recipes 안에 opstest 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)

directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

레시피는 메시지를 기록하지만 이 파일은 `Chef::Log.info`를 호출해 기록합니다. 이 예제에서는 Test Kitchen을 사용하지 않으므로 `log` 메서드가 그다지 유용하지 않습니다.는 Chef 실행이 완료된 후 읽을 수 있는 Chef 로그에 메시지를 `Chef::Log.info` 넣습니다. AWS OpsWorks Stacks는 나중에 설명하듯이 이러한 로그를 쉽게 볼 수 있는 방법을 제공합니다.

> **Note**
>
> Chef 로그에는 일반적으로 일상적이고 상대적으로 관심도가 떨어지는 정보가 많이 포함되어 있습니다. 메시지 텍스트를 묶고 있는 '*' 문자 덕분에 메시지를 쉽게 식별할 수 있습니다.

5. `.zip`의 `opsworks_cookbooks` 아카이브를 생성합니다. AWS OpsWorks Stacks 인스턴스에 쿡북을 설치하려면 리포지토리에 쿡북을 저장하고 인스턴스에 쿡북을 다운로드하는 데 필요한 정보를 AWS OpsWorks Stacks에 제공해야 합니다. 지원되는 여러 리포지토리 유형에 쿡북을 저장할 수 있습니다. 이 예제에서는 쿡북이 포함된 아카이브 파일을 Amazon S3 버킷에 저장합니다. 쿡북 리포지토리에 대한 자세한 정보는 쿡북 리포지토리 단원을 참조하세요.

   > **Note**
   >
   > 간단한 설명을 위해 이 예제에서는 전체 `opsworks_cookbooks` 디렉터리를 보관합니다. 그러나 Stacks는 인스턴스에의 모든 AWS OpsWorks 쿡북`opsworks_cookbooks`을 다운로드합니다. 단, 인스턴스 중 하나만 사용해야 합니다. 예제 쿡북만 설치하려면 다른 상위 디렉터리를 만들어 `opstest`를 해당 디렉터리로 이동합니다. 그런 다음 상위 디렉터리의 `.zip` 아카이브를 만들어 `opsworks_cookbooks.zip` 대신 사용합니다.
   > Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

6. 아카이브를 Amazon S3 버킷에 업로드하고, 해당 아카이브를 퍼블릭으로 설정한 다음, 아카이브의 URL을 적어 둡니다.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

- 리포지토리 유형 – S3 아카이브
- 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

기타 설정에 기본값을 사용하고 [저장]을 클릭하여 스택 구성을 업데이트합니다.

2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.
3. 실행할 레시피가 **opstest::default**으로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 opstest::default로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

opstest를 확인하려면

1. 가장 먼저 Chef 로그를 확인합니다. opstest1 인스턴스의 로그 열에서 표시를 클릭하여 로그를 표시합니다. 아래로 스크롤하면 맨 아래 근처에 로그 메시지가 보입니다.

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. SSH를 사용하여 인스턴스에 로그인하고 /srv/www/의 내용을 표시합니다.

모든 단계를 수행했다면 예상했던 /srv/www/shared 디렉터리 대신 /srv/www/config 디렉터리가 보일 것입니다. 다음 섹션에서는 이러한 버그를 빠르게 수정하는 몇 가지 지침을 제공합니다.

자동으로 레시피 실행

[레시피 실행] 명령은 사용자 지정 레시피를 테스트하는 편리한 방법입니다. 이 때문에 다음 예제의 대부분에서 이 명령이 사용됩니다. 그러나 실제로는 인스턴스 부팅이 완료된 후 또는 앱을 배포할 때와 같이 인스턴스 수명 주기의 표준 지점에서 레시피를 실행합니다. AWS OpsWorks Stacks는 설정, 구

성, 배포, 배포 취소 및 종료와 같은 각 계층에 대한 수명 주기 이벤트 세트를 지원하여 인스턴스에서 레시피 실행을 간소화합니다. 레시피를 적절한 수명 주기 이벤트에 할당하여 AWS OpsWorks Stacks가 계층의 인스턴스에서 레시피를 자동으로 실행하도록 할 수 있습니다.

일반적으로는 인스턴스가 부팅을 완료하는 대로 디렉터리를 생성하는데, 이는 설정 이벤트에 해당합니다. 다음 절차는 예제의 이전 단계에서 생성한 것과 동일한 스택을 사용하여 설정 시 예제 레시피를 실행하는 방법을 보여줍니다. 다른 이벤트에도 동일한 절차를 사용할 수 있습니다.

설정 시 레시피를 자동으로 실행하려면

1. 탐색 창에서 [계층]를 선택한 다음 OpsTest 계층의 [레시피] 링크 옆에 있는 연필 아이콘을 선택합니다.
2. **`opstest::default`**을 계층의 설정 레시피에 추가하고, +를 클릭하여 계층에 추가하고, 저장을 선택하여 구성을 저장합니다.
3. [인스턴스]를 선택해 계층에 다른 인스턴스를 추가한 다음 시작합니다.

   이 인스턴스의 이름은 `opstest2`여야 합니다. 부팅이 완료되면 AWS OpsWorks Stacks가를 실행합니다`opstest::default`.
4. `opstest2` 인스턴스가 온라인 상태가 되면 `/srv/www/shared`가 있는지 확인합니다.

> **Note**
>
> 설정, Configure 또는 Deploy 이벤트에 레시피를 할당한 경우 스택 명령(설정 및 Configure) 또는 배포 명령(Deploy)을 사용하여 이벤트를 트리거함으로써 수동으로 레시피를 실행할 수도 있습니다. 한 이벤트에 여러 레시피를 할당하면 이러한 명령은 모든 레시피를 실행합니다.

레시피 문제 해결 및 수정

예상한 결과를 얻지 못하거나 레시피가 성공적으로 실행되지 않은 경우 일반적으로 Chef 로그를 검사하는 것으로 문제 해결이 시작됩니다. 로그에는 실행에 대한 상세한 설명과 레시피로부터의 모든 인라인 로그 메시지가 포함됩니다. 레시피가 그냥 실패한 경우 로그가 특히 유용합니다. 이러한 경우가 발생하면 Chef가 스택 트레이스를 포함하여 오류를 기록합니다.

이 예제에서와 같이 레시피가 성공한 경우에는 Chef 로그가 큰 도움이 안 됩니다. 그러면 레시피의 첫 몇 줄을 보다 면밀히 검토하여 문제를 파악할 수 있습니다.

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

Vagrant에서 레시피를 테스트할 경우 CentOS가 Amazon Linux의 합리적 대역이지만, 지금은 실제 Amazon Linux 인스턴스가 실행 중입니다. Amazon Linux용 플랫폼 값은 amazon입니다. value_for_platform 호출에 이 값이 포함되지 않으므로 기본적으로 레시피가 /srv/www/config를 생성합니다. 문제 해결에 대한 자세한 정보는 디버깅 및 문제 해결 안내서 단원을 참조하세요.

이제 문제를 식별했으므로 레시피를 업데이트하고 수정을 확인해야 합니다. 원래 소스 파일로 돌아가 default.rb를 업데이트하고, 새 아카이브를 Amazon S3로 업로드하는 등의 작업을 수행합니다. 하지만 이 프로세스는 약간 번거롭고 시간이 걸릴 수 있습니다. 다음은 인스턴스에서 레시피를 편집하는 방법으로, 예제 내 버그와 같은 간단한 레시피 버그를 수정하는 데 특히 유용한 훨씬 빠른 접근 방식입니다.

인스턴스에서 레시피를 편집하려면

1. SSH를 사용하여 인스턴스에 로그인한 다음 sudo su를 실행하여 권한을 승격시킵니다. 이 쿡북 디렉터리에 액세스하려면 루트 권한이 필요합니다.
2. AWS OpsWorks Stacks는 쿡북을에 저장/opt/aws/opsworks/current/site-cookbooks하므로 로 이동합니다/opt/aws/opsworks/current/site-cookbooks/opstest/recipes.

   > **Note**
   >
   > AWS OpsWorks 또한 Stacks는 쿡북의 사본을에 저장합니다/opt/aws/opsworks/current/merged-cookbooks. 이러한 쿡북은 편집하지 마십시오. 레시피를 실행하면 AWS OpsWorks Stacks가 쿡북을에서 .../site-cookbooks로 복사하므로 .../merged-cookbooks에서 변경한 내용을 .../merged-cookbooks 덮어씁니다.

3. 인스턴스에서 텍스트 편집기를 사용하여 default.rb를 편집하고 centos는 amazon으로 바꿉니다. 이제 레시피가 다음과 같이 보여야 합니다.

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "amazon" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

수정을 확인하려면 레시피 실행 스택 명령을 다시 실행하여 레시피를 실행합니다. 이제 인스턴스에 `/srv/www/shared` 디렉터리가 있을 것입니다. 추가로 레시피를 수정해야 할 경우 [레시피 실행]을 원하는 만큼 실행할 수 있으며, 명령을 실행할 때마다 인스턴스를 중지했다 재시작할 필요가 없습니다. 레시피가 올바로 작동하는 것으로 판단되면 소스 쿡북에서 코드를 업데이트할 필요가 없습니다.

> **Note**
>
> AWS OpsWorks Stacks가 자동으로 실행하도록 레시피를 수명 주기 이벤트에 할당한 경우 언제든지 레시피 실행을 사용하여 레시피를 다시 실행할 수 있습니다. AWS OpsWorks Stacks 콘솔을 사용하여 적절한 이벤트를 수동으로 트리거하여 인스턴스를 다시 시작하지 않고도 레시피를 원하는 횟수만큼 다시 실행할 수도 있습니다. 하지만 이 방법에서는 이벤트의 레시피가 모두 실행됩니다. 다음을 기억하세요.
>
> - 설정 또는 Configure 이벤트를 트리거하려면 스택 명령을 사용합니다.
> - Deploy 또는 Undeploy 이벤트를 트리거하려면 배포 명령을 사용합니다.

Windows 인스턴스에서 레시피 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제는 기본적으로 Linux 인스턴스에서 레시피 실행 섹션의 요약 버전으로, Windows 스택에서 레시피를 실행하는 방법을 보여줍니다. 두 운영 체제 모두와 관련이 있는 내용이 보다 자세히 설명되어 있으므로 먼저 Linux 인스턴스에서 레시피 실행 섹션을 읽는 것이 좋습니다.

AWS OpsWorks Stacks Linux 인스턴스에서 레시피를 실행하는 방법에 대한 설명은 섹션을 참조하세요Linux 인스턴스에서 레시피 실행.

주제

- RDP 액세스 활성화
- 레시피 생성 및 실행
- 자동으로 레시피 실행

RDP 액세스 활성화

시작하기 전에 인스턴스에 대해 RDP 액세스를 허용하는 인바운드 규칙을 포함하는 보안 그룹을 설정해야 합니다. 스택을 생성할 때 이 그룹이 필요합니다.

리전에서 첫 번째 스택을 생성하면 AWS OpsWorks Stacks가 보안 그룹 세트를 생성합니다. 여기에는과 같은 이름의 인스턴스가 포함되며`AWS-OpsWorks-RDP-Server`, AWS OpsWorks Stacks는 모든 Windows 인스턴스에 연결하여 RDP 액세스를 허용합니다. 하지만 이 보안 그룹은 기본적으로 아무 규칙도 포함하고 있지 않으므로, 인스턴스에 RDP 액세스를 허용하는 인바운드 규칙을 추가해야 합니다.

RDP 액세스를 허용하려면

1. Amazon EC2 콘솔을 열고, 콘솔을 스택의 리전으로 설정한 다음, 탐색 창에서 보안 그룹을 선택합니다.
2. [AWS-OpsWorks-RDP-Server], [인바운드] 탭, [편집]을 차례대로 선택합니다.
3. 다음 설정으로 규칙을 추가합니다.
   - 유형 – RDP
   - 소스 – 허용 가능한 소스 IP 주소.

     일반적으로 IP 주소 또는 지정된 IP 주소 범위(대개 회사 IP 주소)에서 들어오는 인바운드 RDP 요청을 허용합니다.

> **Note**
>
> 또한 나중에 설명하듯이 사용자 권한을 편집해 일반 사용자에게 RDP 액세스를 허용해야 합니다.

자세한 내용은 RDP를 사용하여 로그인 섹션을 참조하세요.

레시피 생성 및 실행

다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.

   - 이름 – WindowsRecipeTest
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – RecipeTest
   - 짧은 이름 - recipetest
3. 기본 설정을 사용하여 RecipeTest 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

   AWS OpsWorks Stacks는 이 인스턴스 `AWS-OpsWorks-RDP-Server`에를 자동으로 할당하므로 권한 있는 사용자가 인스턴스에 로그인할 수 있습니다.
4. [권한] 및 [편집]을 차례대로 선택하고 [SSH/RDP] 및 [sudo/admin]을 선택합니다. 인스턴스에 로그인하려면 일반 사용자에게는 `AWS-OpsWorks-RDP-Server` 보안 그룹 이외에 이러한 권한 부여가 필요합니다.

> **Note**
>
> Administrator로도 로그인할 수 있지만 로그인 절차가 다릅니다. 자세한 내용은 RDP를 사용하여 로그인 섹션을 참조하세요.

인스턴스가 시작되는 동안(보통 몇 분 정도 소요됨) 쿡북을 생성할 수 있습니다. 이 예제의 레시피는 데이터 디렉터리를 생성하며, 기본적으로 예제 3: 디렉터리 생성의 레시피를 Windows용으로 수정한 것입니다.

> **Note**
>
> Stacks Windows 인스턴스용 AWS OpsWorks 쿡북을 구현할 때는 Stacks Linux 인스턴스용 AWS OpsWorks 쿡북을 구현할 때와 약간 다른 디렉터리 구조를 사용합니다. 자세한 내용은 쿡북 리포지토리 단원을 참조하십시오.

쿡북을 설정하려면

1. `windowstest` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 `metadata.rb` 파일을 만들어 `windowstest`에 저장합니다.

```
name "windowstest"
version "0.1.0"
```

3. `recipes` 안에 `windowstest` 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 `default.rb` 파일을 만들어 `recipes` 디렉터리에 저장합니다.

```
Chef::Log.info("******Creating a data directory.******")

directory 'C:\data' do
  rights :full_control, 'instance_name\username'
  inherits false
  action :create
end
```

*username*을 사용자 이름으로 바꿉니다.

5. 쿡북을 리포지토리에 저장합니다.

   Stacks 인스턴스에 AWS OpsWorks 쿡북을 설치하려면 리포지토리에 쿡북을 저장하고 인스턴스에 쿡북을 다운로드하는 데 필요한 정보를 AWS OpsWorks Stacks에 제공해야 합니다. S3 버킷 또는 Git 리포지토리에 Windows 쿡북을 아카이브 파일로 저장할 수 있습니다. 이 예제에서는 S3 버킷을 사용하므로 `windowstest` 디렉터리의 .zip 아카이브를 생성해야 합니다. 쿡북 리포지토리에 대한 자세한 정보는 쿡북 리포지토리 단원을 참조하세요.

6. 아카이브를 S3 버킷에 업로드하고, 해당 아카이브를 퍼블릭으로 설정한 다음, 아카이브의 URL을 적어 둡니다. 또한 프라이빗 아카이브를 사용할 수도 있지만 이 예제에는 퍼블릭 아카이브면 충분합니다. 퍼블릭 아카이브가 작업하기 더 간단합니다.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.

2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스(온라인 인스턴스 포함)에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어씁니다.

3. 사용자 지정 쿡북 업데이트가 완료되면 실행할 레시피가 **`windowstest::default`**로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 레시피로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

windowstest를 확인하려면

1. Chef 로그를 확인합니다. opstest1 인스턴스의 로그 열에서 표시를 선택하여 로그를 표시합니다. 아래로 스크롤하면 맨 아래 근처에 로그 메시지가 보입니다.

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 인스턴스를 선택하고, 인스턴스의 작업 열에서 rdp를 선택한 다음 적절한 만료 시간을 가진 RDP 비밀번호를 요청합니다. DNS 이름, 사용자 이름 및 암호를 복사합니다. 그런 다음 RDP 클라이언트(예: Windows 원격 데스크톱 연결 클라이언트)에서 해당 정보를 사용해 인스턴스에 로그인하고 `c:\data`가 있는지 확인합니다. 자세한 내용은 RDP를 사용하여 로그인 섹션을 참조하세요.

> **Note**
>
> 레시피가 올바로 작동하지 않는다면 문제 해결 팁은 레시피 문제 해결 및 수정 단원을 참조하세요. Windows 인스턴스에도 대부분 적용됩니다. 인스턴스에서 레시피를 편집하여 수정 사항을 테스트하려면 `C:\chef\cookbooks` 디렉터리에서 쿡북을 찾습니다. 여기서 AWS OpsWorks Stacks는 사용자 지정 쿡북을 설치합니다.

자동으로 레시피 실행

[레시피 실행] 명령은 사용자 지정 레시피를 테스트하는 편리한 방법입니다. 이 때문에 다음 예제의 대부분에서 이 명령이 사용됩니다. 그러나 실제로는 인스턴스 부팅이 완료된 후 또는 앱을 배포할 때와 같이 인스턴스 수명 주기의 표준 지점에서 레시피를 실행합니다. AWS OpsWorks Stacks는 설정, 구성, 배포, 배포 취소 및 종료와 같은 각 계층에 대한 수명 주기 이벤트 세트를 지원하여 인스턴스에서 레시피 실행을 간소화합니다. 레시피를 적절한 수명 주기 이벤트에 할당하여 AWS OpsWorks Stacks가 계층의 인스턴스에서 레시피를 자동으로 실행하도록 할 수 있습니다.

일반적으로는 인스턴스가 부팅을 완료하는 대로 디렉터리를 생성하는데, 이는 설정 이벤트에 해당합니다. 다음 절차는 예제의 이전 단계에서 생성한 것과 동일한 스택을 사용하여 설정 시 예제 레시피를 실행하는 방법을 보여줍니다. 다른 이벤트에도 동일한 절차를 사용할 수 있습니다.

설정 시 레시피를 자동으로 실행하려면

1. 탐색 창에서 [계층]를 선택한 다음 RecipeTest 계층의 [레시피] 링크 옆에 있는 연필 아이콘을 선택합니다.
2. **`windowstest::default`**을 계층의 설정 레시피에 추가하고, +를 선택하여 계층에 추가하고, 저장을 선택하여 구성을 저장합니다.
3. [인스턴스]를 선택해 계층에 다른 인스턴스를 추가한 다음 시작합니다.

   이 인스턴스의 이름은 `recipetest2`여야 합니다. 부팅이 완료되면 AWS OpsWorks Stacks가를 실행합니다`windowstest::default`.
4. `recipetest2` 인스턴스가 온라인 상태가 되면 `c:\data`가 있는지 확인합니다.

> **Note**
>
> 설정, Configure 또는 Deploy 이벤트에 레시피를 할당한 경우 스택 명령(설정 및 Configure) 또는 배포 명령(Deploy)을 사용하여 이벤트를 트리거함으로써 수동으로 레시피를 실행할 수도 있습니다. 한 이벤트에 여러 레시피를 할당하면 이러한 명령은 모든 레시피를 실행합니다.

Windows PowerShell 스크립트 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 다음 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.

레시피가 Windows 인스턴스에서 작업, 특히 해당 Chef 리소스가 없는 작업을 수행하도록 하는 한 가지 방법은 레시피에서 Windows PowerShell 스크립트를 실행하도록 하는 것입니다. 이 섹션에서는 Windows PowerShell 스크립트를 사용하여 Windows 기능을 설치하는 방법을 설명하면서 기본 사항을 소개합니다.

`powershell_script` 리소스는 인스턴스에서 Windows PowerShell cmdlet을 실행합니다. 다음 예제는 Install-WindowsFeature cmdlet을 사용하여 인스턴스에 XPS 뷰어를 설치합니다.

다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 클릭합니다.

   - 이름 – PowerShellTest
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – PowerShell
   - 짧은 이름 - powershell
3. 기본 설정을 사용하여 PowerShell 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.
4. [권한] 및 [편집]을 차례대로 선택하고 [SSH/RDP] 및 [sudo/admin]을 선택합니다. 인스턴스에 일반 사용자로 로그인하려면 `AWS-OpsWorks-RDP-Server` 보안 그룹 이외에 이러한 권한 부여가 필요합니다.

인스턴스가 시작되는 동안(보통 몇 분 정도 소요됨) 쿡북을 생성할 수 있습니다. 이 예제의 레시피는 데이터 디렉터리를 생성하며, 기본적으로 예제 3: 디렉터리 생성의 레시피를 Windows용으로 수정한 것입니다.

쿡북을 설정하려면

1. `powershell` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 `metadata.rb` 파일을 만들어 `windowstest`에 저장합니다.

```
name "powershell"
version "0.1.0"
```

3. `recipes` 디렉터리 내에 `powershell` 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 `default.rb` 파일을 만들어 `recipes` 디렉터리에 저장합니다.

```
Chef::Log.info("******Installing XPS.******")

powershell_script "Install XPS Viewer" do
  code <<-EOH
    Install-WindowsFeature XPS-Viewer
  EOH
  guard_interpreter :powershell_script
  not_if "(Get-WindowsFeature -Name XPS-Viewer).installed"
end
```

   - `powershell_script` 리소스가 cmdlet을 실행해 XPS 뷰어를 설치합니다.

     이 예제에서는 cmdlet을 하나만 실행하지만 `code` 블록에는 명령줄을 무제한 포함할 수 있습니다.

   - `guard_interpreter` 속성은 Chef에 Windows PowerShell의 64비트 버전을 사용하도록 지시합니다.
   - `not_if` 보호 속성은 Chef가 이미 설치된 기능을 설치하지 않도록 합니다.

5. `powershell` 디렉터리의 `.zip` 아카이브를 생성합니다.
6. 아카이브를 Amazon S3 버킷에 업로드하고, 해당 아카이브를 퍼블릭으로 설정한 다음, 아카이브의 URL을 적어 둡니다. 또한 프라이빗 아카이브를 사용할 수도 있지만 이 예제에는 퍼블릭 아카이브면 충분합니다. 퍼블릭 아카이브가 작업하기 더 간단합니다.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.
2. [사용자 지정 쿡북 업데이트] 스택 명령을 실행하여 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다.
3. 사용자 지정 쿡북 업데이트가 완료되면 실행할 레시피가 **`powershell::default`**로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다.

> **Note**
>
> 이 예제에서는 편의상 레시피 실행을 사용하지만 일반적으로 AWS OpsWorks Stacks가 레시피를 적절한 수명 주기 이벤트에 할당하여 레시피를 자동으로 실행하도록 합니다. 수동으로 이벤트를 트리거하여 그러한 레시피를 실행할 수 있습니다. 스택 명령을 사용하여 설정 및 Configure 이벤트를 트리거할 수 있고, 배포 명령을 사용하여 Deploy 및 Undeploy 이벤트를 트리거할 수 있습니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

powershell 레시피를 확인하려면

1. Chef 로그를 확인합니다. powershell1 인스턴스의 로그 열에서 표시를 클릭하여 로그를 표시합니다. 아래로 스크롤하면 맨 아래 근처에 로그 메시지가 보입니다.

```
...
[2015-04-27T18:12:09+00:00] INFO: Storing updated cookbooks/powershell/metadata.rb
 in the cache.
[2015-04-27T18:12:09+00:00] INFO: ******Installing XPS.******
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Install XPS Viewer]
 action run (powershell::default line 3)
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Guard resource]
 action run (dynamically defined)
[2015-04-27T18:12:42+00:00] INFO: powershell_script[Install XPS Viewer] ran
 successfully
...
```

2. RDP를 사용하여 인스턴스에 로그인하고 [시작] 메뉴를 엽니다. XPS 뷰어가 [Windows 액세서리]와 함께 나열되어야 합니다.

Vagrant에서 스택 구성 및 배포 속성 모의

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제는 Linux 인스턴스에만 적용됩니다. Test Kitchen은 아직 Windows를 지원하지 않으므로 AWS OpsWorks Stacks 인스턴스에서 모든 Windows 예제를 실행합니다.

AWS OpsWorks Stacks는 모든 수명 주기 이벤트에 대해 스택의 각 인스턴스에 대한 노드 객체에 스택 구성 및 배포 속성을 추가합니다. 이러한 속성은 스택 구성의 스냅샷을 제공합니다. 여기에는 각 계층 및 해당 온라인 인스턴스의 구성, 배포된 각 앱의 구성 등이 포함됩니다. 이러한 속성은 노드 객체에 있으므로 모든 레시피에서 액세스할 수 있습니다. AWS OpsWorks 대부분의 Stacks 인스턴스용 레시피는 이러한 속성 중 하나 이상을 사용합니다.

Vagrant 상자에서 실행되는 인스턴스는 AWS OpsWorks Stacks에서 관리하지 않으므로 노드 객체에는 기본적으로 스택 구성 및 배포 속성이 포함되지 않습니다. 하지만 Test Kitchen 환경에 적절한 속성

세트를 추가할 수 있습니다. 그런 다음 Test Kitchen은 인스턴스의 노드 객체에 속성을 추가하고 레시피는 AWS OpsWorks Stacks 인스턴스에서와 마찬가지로 속성에 액세스할 수 있습니다.

이 주제는 적합한 스택 구성 및 배포 속성의 사본을 구하고, 인스턴스에 속성을 설치하고, 속성에 액세스하는 방법을 설명합니다.

> **Note**
>
> Test Kitchen을 사용하여 레시피를 테스트하려는 경우, fauxhai가 스택 구성 및 배포 JSON을 모의하는 대체 방법을 제공합니다.

쿡북을 설정하려면

1. `printjson`의 하위 디렉터리 `opsworks_cookbooks`를 만들고 그 디렉터리로 이동합니다.
2. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성합니다.
3. `printjson`에 하위 디렉터리 `recipes` 및 `environments`를 추가합니다.

적절한 정의를 포함하는 속성 파일을 쿡북에 추가하여 스택 구성 및 배포 속성을 모의할 수도 있지만, Test Kitchen 환경을 사용하는 것이 더 나은 접근 방식입니다. 기본적으로 두 가지 방법이 있습니다.

- `.kitchen.yml`에 속성 정의를 추가합니다.

  이 방법은 속성이 몇 개 정도일 경우 매우 유용합니다. 자세한 정보는 kitchen.yml 단원을 참조하세요.
- 환경 파일에서 속성을 정의하고 `.kitchen.yml`에서 파일 단원을 참조합니다.

  이 방법은 환경 파일이 이미 JSON 형식이므로 일반적으로 스택 구성 및 배포 속성에 대해 선호됩니다. 적절한 Stacks 인스턴스에서 JSON 형식의 속성 사본을 가져와 붙여넣기만 AWS OpsWorks 하면 됩니다. 다음 예제는 모두 환경 파일을 사용합니다.

쿡북에서 스택 구성 및 배포 속성을 생성하는 가장 간단한 방법은 적절히 구성된 스택을 생성하고 인스턴스에서 결과 속성을 JSON으로 복사하는 것입니다. Test Kitchen 환경 파일을 관리 가능하게 유지하기 위해 레시피에 필요한 속성만 포함하도록 JSON을 편집할 수 있습니다. 이 장에서 설명하는 예제는 Chef 11 Linux 스택 시작하기의 스택을 기반으로 합니다. 이 스택은 로드 밸런서, PHP 애플리케이션 서버 및 MySQL 데이터베이스 서버를 포함하는 PHP 애플리케이션 서버 스택입니다.

스택 구성 및 배포 JSON을 생성하려면

1. SimplePHPApp 배포를 비롯하여 Chef 11 Linux 스택 시작하기 단원에서 설명하는 대로 MyStack을 생성합니다. 원하는 경우 4단계: MyStack 확장 단원에서 호출한 두 번째 PHP 앱 서버 인스턴스를 생략할 수 있습니다. 이 예제에서는 이러한 속성을 사용하지 않기 때문입니다.
2. 아직 수행하지 않은 경우 php-app1 인스턴스를 시작한 다음 SSH로 로그인합니다.
3. 터미널 창에서 다음 agent cli 명령을 실행합니다.

   ```
   sudo opsworks-agent-cli get_json
   ```

   이 명령은 인스턴스의 최신 스택 구성 및 배포 속성을 JSON 형식으로 터미널 창에 인쇄합니다.
4. JSON을 .json 파일에 복사하고 워크스테이션의 원하는 위치에 저장합니다. 세부 정보는 SSH 클라이언트에 따라 달라집니다. 예를 들어, Windows에서 PuTTY를 사용하는 경우 터미널 창의 모든 텍스트를 Windows 클립보드에 복사하는 Copy All to Clipboard 명령을 실행할 수 있습니다. 그런 다음 해당 내용을 .json 파일에 붙여 넣고 이 파일을 편집해 관련 없는 텍스트를 제거합니다.
5. 필요에 따라 MyStack JSON을 편집합니다. 스택 구성 및 배포 속성은 수 없이 많지만 일반적으로 쿡북에서는 이 중 극히 일부만 사용합니다. 환경 파일을 관리하기 쉽게 유지하려면 원래 구조를 유지하되 쿡북에서 실제 사용하는 속성만 포함하도록 JSON을 편집할 수 있습니다.

   이 예제에서는 ['opsworks']['stack'] 속성 두 개, 즉 ['id] 및 ['name']만 포함하도록 크게 편집한 MyStack JSON 버전을 사용합니다. 다음과 유사한 MyStack JSON의 편집 버전을 생성합니다.

   ```
   {
     "opsworks": {
       "stack": {
         "name": "MyStack",
         "id": "42dfd151-6766-4f1c-9940-ba79e5220b58",
       },
     },
   }
   ```

이 JSON을 인스턴스의 노드 객체로 가져오려면 Test Kitchen 환경에 추가해야 합니다.

Test Kitchen 환경에 스택 구성 및 배포 속성을 추가하려면

1. 다음 내용이 포함된 환경 파일 `test.json`을 만들어 이 파일을 쿡북의 `environments` 폴더에 저장합니다.

```
{
  "default_attributes": {
    "opsworks" : {
      "stack" : {
        "name" : "MyStack",
        "id" : "42dfd151-6766-4f1c-9940-ba79e5220b58"
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

   환경 파일에는 다음 요소가 포함되어 있습니다.

   - `default_attributes` – JSON 형식의 기본 속성.

     이러한 속성은 속성 유형이 `default`인 노드 객체에 추가되는데, 이 속성 유형은 모든 스택 구성 및 배포 JSON 속성에서 사용합니다. 이 예제에서는 앞서 표시된 스택 구성 및 배포 JSON의 편집 버전을 사용합니다.
   - `chef_type` - 이 요소는 `environment`로 설정합니다.
   - `json_class` - 이 요소는 `Chef::Environment`로 설정합니다.
2. `.kitchen.yml`을 편집하여 Test Kitchen 환경을 다음과 같이 정의합니다.

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-12.04
```

```
suites:
  - name: printjson
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[printjson::default]
    attributes:
```

`kitchen init`이 생성한 기본 `.kitchen.yml`에 다음 요소를 추가하여 환경을 정의합니다.

provisioner

다음 요소를 추가합니다.

- `name` - 이 요소는 `chef_solo`로 설정합니다.

  AWS OpsWorks Stacks 환경을 더 가깝게 복제하려면 Chef 솔로 대신 Chef 클라이언트 로컬 모드를 사용할 수 있습니다. 로컬 모드는 원격 서버 대신 인스턴스에서 로컬로 실행되는 Chef 서버의 경량 버전(Chef Zero)을 사용하는 Chef 클라이언트 옵션입니다. 로컬 모드에서는 레시피가 원격 서버에 연결하지 않고 검색 또는 데이터 백과 같은 Chef 서버 기능을 사용할 수 있습니다.

- `environments_path` - 환경 파일(이 예제의 경우 `./environments`)을 포함하는 쿡북 하위 디렉터리

suites:provisioner

환경 파일의 이름(.json 확장명 제외)으로 설정된 `environment` 요소와 함께 `solo_rb` 요소를 추가합니다. 이 예제에서는 `environment`를 `test`로 설정합니다.

3. 다음 내용이 포함된 레시피 파일 `default.rb`를 만들어 이 파일을 쿡북의 `recipes` 디렉터리에 저장합니다.

```
log "Stack name: #{node['opsworks']['stack']['name']}"
log "Stack id: #{node['opsworks']['stack']['id']}"
```

이 레시피는 환경에 추가한 스택 구성 및 배포 값 2개를 기록하기만 합니다. 레시피가 가상 상자에서 로컬로 실행되지만 레시피가 AWS OpsWorks Stacks 인스턴스에서 실행 중일 때와 동일한 노드 구문을 사용하여 이러한 속성을 참조합니다.

4. `kitchen converge`을(를) 실행합니다. 다음 로그 출력과 유사해야 합니다.

```
...
Converging 2 resources
Recipe: printjson::default
  * log[Stack name: MyStack] action write[2014-07-01T23:14:09+00:00] INFO:
 Processing log[Stack name: MyStack] action write (printjson::default line 1)

[2014-07-01T23:14:09+00:00] INFO: Stack name: MyStack

  * log[Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58] action
 write[2014-07-01T23:14:09+00:00] INFO: Processing log[Stack id:
 42dfd151-6766-4f1c-9940-ba79e5220b58] action write (printjson::default line 2)

[2014-07-01T23:14:09+00:00] INFO: Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58

...
```

스택 구성 및 배포 속성 값 사용

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피는 스택 구성 또는 배포된 앱에 대한 정보를 자주 필요로 합니다. 예를 들어 구성 파일을 생성하려면 스택의 IP 주소 목록이 필요하고, 로그 디렉터리를 생성하려면 앱의 배포 디렉터리가 필요합니다. 이 데이터를 중앙 서버에 저장하는 대신 AWS OpsWorks Stacks는 각 수명 주기 이벤트에 대해 각 인스턴스의 노드 객체에 스택 구성 및 배포 속성 세트를 설치합니다. 이러한 속성은 배포된 앱을 포함하여 현재 스택 상태를 나타냅니다. 그러면 레시피가 노드 객체로부터 필요한 데이터를 가져옵니다.

Note

때로는 애플리케이션이 스택 구성 및 배포 속성 값과 같이 노드 객체에 저장된 정보를 필요로 합니다. 하지만 애플리케이션은 노드 객체에 액세스할 수 없습니다. 노드 객체 데이터를 애플

리케이션에 제공하기 위해 노드 객체에서 필요한 정보를 검색하여 편리한 형식의 파일에 기록하는 레시피를 구현할 수 있습니다. 그러면 애플리케이션이 파일에서 데이터를 읽을 수 있습니다. 자세한 내용과 예제는 애플리케이션으로 데이터 전달 단원을 참조하세요.

레시피는 다음과 같이 노드 객체로부터 스택 구성 및 배포 속성 값을 가져올 수 있습니다.

- 속성의 정규화된 이름을 사용하여 직접.

  이 방법은 모든 Linux 스택에서 사용할 수 있지만 Windows 스택에서는 사용할 수 없습니다.
- 노드 객체에서 속성 값을 쿼리하는 데 사용할 수 있는 Chef 검색을 사용.

  이 방법은 모든 Windows 스택 및 Chef 11.10 Linux 스택에서 사용할 수 있습니다.

> **Note**
>
> Linux 스택에서는 에이전트 CLI를 사용하여 인스턴스의 스택 구성 및 배포 속성의 사본을 JSON 형식으로 가져올 수 있습니다. 자세한 내용은 Vagrant에서 스택 구성 및 배포 속성 모의 섹션을 참조하세요.

주제

- 속성 값을 직접 가져오기
- Chef 검색을 사용하여 속성 값 가져오기

속성 값을 직접 가져오기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 방법은 Linux 스택에서만 사용할 수 있습니다.

Vagrant에서 스택 구성 및 배포 속성 모의 섹션에서는 노드 구문을 사용하여 특정 속성을 직접 참조함으로써 스택 구성 및 배포 데이터를 가져오는 방법을 설명합니다. 때로는 이 방법이 최선입니다. 하지만 많은 속성이 모음 또는 목록으로 정의되어 있으며, 그 콘텐츠 및 이름은 스택마다 또한 특정 스택에서도 시간 경과에 따라 다를 수 있습니다. 예를 들어 `deploy` 속성에는 앱의 짧은 이름으로 명명된 앱 속성의 목록이 포함됩니다. 앱 속성을 포함하여 이 목록은 일반적으로 스택마다 다르며, 심지어 배포마다 다릅니다.

목록 또는 모음에서 속성을 열거하여 필요한 데이터를 가져오는 것이 종종 더 유용하고 때로는 반드시 필요할 수도 있습니다. 예를 들어 스택 내 인스턴스의 퍼블릭 IP 주소를 알아야 할 경우 이 정보는 `['opsworks']['layers']` 속성에 포함되어 있습니다. 이 속성은 스택의 각 계층마다 계층의 짧은 이름으로 명명된 한 요소를 포함하는 해시 테이블로 설정됩니다. 각 계층 요소는 계층의 속성을 포함하는 해시 테이블로 설정되며, 이 중 하나가 `['instances']`입니다. 이 요소는 계층의 각 인스턴스마다 인스턴스의 짧은 이름으로 명명된 한 속성을 포함하는 또 하나의 해시 테이블로 설정됩니다. 각 인스턴스 속성은 `['ip']`를 포함하여 퍼블릭 IP 주소를 나타내는 인스턴스 속성을 포함하는 또 다른 해시 테이블로 설정됩니다. 이를 시각화하는 데 어려움이 있을 경우 참조할 수 있도록 다음 절차에 JSON 형식의 예제가 포함되어 있습니다.

이 예제는 스택의 계층에 대해 스택 구성 및 배포 JSON으로부터 데이터를 가져오는 방법을 보여줍니다.

쿡북을 설정하려면

1. `opsworks_cookbooks` 안에 `listip` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성합니다.
3. `listip`에 디렉터리 `recipes` 및 `environments`를 추가합니다.
4. 관련 속성이 포함된 MyStack 구성 및 배포 속성의 편집된 JSON 버전을 생성합니다. 다음과 같이 보여야 합니다.

```
{
  "opsworks": {
    "layers": {
      "php-app": {
```

```
        "name": "PHP App Server",
        "id": "efd36017-ec42-4423-b655-53e4d3710652",
        "instances": {
          "php-app1": {
            "ip": "192.0.2.0"
          }
        }
      },
      "db-master": {
        "name": "MySQL",
        "id": "2d8e0b9a-0d29-43b7-8476-a9b2591a7251",
        "instances": {
          "db-master1": {
            "ip": "192.0.2.5"
          }
        }
      },
      "lb": {
        "name": "HAProxy",
        "id": "d5c4dda9-2888-4b22-b1ea-6d44c7841193",
        "instances": {
          "lb1": {
            "ip": "192.0.2.10"
          }
        }
      }
    }
  }
}
```

5. 환경 파일 `test.json`을 만들고, 예제 JSON을 `default_attributes`에 붙여 넣은 다음 쿡북의 `environments` 폴더에 이 파일을 저장합니다. 이 파일은 다음과 유사해야 합니다(간단한 설명을 위해 대부분의 예제 JSON에서는 생략 부호 사용).

```
{
  "default_attributes" : {
    "opsworks": {
      "layers": {
        ...
      }
    }
  },
```

```
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

6. .kitchen.yml의 텍스트를 다음으로 바꿉니다.

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero
  environments_path: ./environment

platforms:
  - name: ubuntu-12.04

suites:
  - name: listip
    provisioner:
      client_rb:
        environment: test
    run_list:
      - recipe[listip::default]
    attributes:
```

쿡북을 설정한 후 다음 레시피를 사용하여 계층 ID를 기록할 수 있습니다.

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
end
```

레시피는 ['opsworks']['layers']에서 계층을 열거하고 각 계층의 이름 및 ID를 기록합니다.

계층 ID 로깅 레시피를 실행하려면

1. 예제 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.
2. kitchen converge을(를) 실행합니다.

해당 부분은 다음과 유사하게 출력됩니다.

```
Recipe: listip::default
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193
```

인스턴스의 IP 주소를 나열하려면 다음과 같은 중첩 루프가 필요합니다.

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
  layerdata['instances'].each do |instance, instancedata|
    log "Public IP: #{instancedata['ip']}"
  end
end
```

내부 루프는 각 계층의 인스턴스를 반복하고 IP 주소를 기록합니다.

인스턴스 IP 로깅 레시피를 실행하려면

1. `default.rb`의 코드를 예제 레시피로 바꿉니다.
2. `kitchen converge`를 실행하여 레시피를 실행합니다.

해당 부분은 다음과 유사하게 출력됩니다.

```
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[Public IP: 192.0.2.0] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.0] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.0


  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[Public IP: 192.0.2.5] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.5] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.5


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193


  * log[Public IP: 192.0.2.10] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.10] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.10
```

작업을 마쳤으면 `kitchen destroy`을 실행합니다. 다음 주제에서는 새 쿡북을 사용합니다.

> **Note**
>
> 스택 구성 및 배포 JSON 모음을 열거하는 가장 일반적인 이유는 배포된 앱에서 배포 디렉터리와 같은 데이터를 가져오는 것입니다. 예제는 Deploy 레시피 섹션을 참조하세요.

Chef 검색을 사용하여 속성 값 가져오기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 방법은 Windows 스택 및 Chef 11.10 Linux 스택에서 사용할 수 있습니다.

노드 객체로부터 직접 스택 구성 및 배포 속성 값을 가져오는 방법은 복잡할 수 있으며 Windows 스택에서는 사용할 수 없습니다. 대안적 방법은 Chef 검색을 사용하여 원하는 속성을 쿼리하는 것입니다. Chef 서버에 익숙하다면 Chef 검색이 AWS OpsWorks Stacks와 약간 다르게 작동하는 것을 알 수 있습니다. AWS OpsWorks Stacks는 로컬 모드에서 chef-client를 사용하기 때문에 Chef 검색은 chef-zero라는 로컬 버전의 Chef 서버에 의존하므로 검색은 원격 서버가 아닌 인스턴스의 노드 객체에 로컬로 저장된 데이터에 대해 작동합니다.

실제로는 AWS OpsWorks Stacks 인스턴스의 노드 객체에 스택 구성 및 배포 속성이 포함되어 있기 때문에 검색을 로컬로 저장된 데이터로 제한하는 것은 일반적으로 중요하지 않습니다. 레시피가 일반적으로 Chef 서버에서 가져오고 동일한 이름을 사용하는 모든 데이터가 아니라면 대부분 포함되므로 일반적으로 AWS OpsWorks Stacks 인스턴스에서 Chef 서버에 대해 작성된 검색 코드를 수정 없이 사용할 수 있습니다. 자세한 내용은 Chef 검색 사용 단원을 참조하십시오.

다음은 검색 쿼리의 기본의 구조를 보여줍니다.

```
result = search(:search_index, "key:pattern")
```

- 검색 인덱스는 쿼리가 적용되는 속성을 지정하고 반환되는 객체의 유형을 결정합니다.
- 키는 속성 이름을 지정합니다.
- 패턴은 검색하려는 속성의 값을 지정합니다.

  특정 속성 값을 쿼리할 수도 있고 와일드 카드를 사용하여 일련의 값을 쿼리할 수도 있습니다.
- 결과는 쿼리를 만족하는 객체의 목록이며, 각각 관련 속성이 여러 개 포함된 해시 테이블입니다.

예를 들어 node 검색 인덱스를 사용할 경우 쿼리는 쿼리를 만족하는 인스턴스마다 하나씩 인스턴스 객체의 목록을 반환합니다. 각 객체는 호스트 이름, IP 주소 등 인스턴스 구성을 정의하는 속성 세트를 포함하는 해시 테이블입니다.

예를 들어 다음 쿼리는 스택의 인스턴스(Chef 용어로는 노드)에 적용되는 표준 Chef 검색 인덱스인 node 검색 인덱스를 사용합니다. 이 쿼리는 호스트 이름이 `myhost`인 인스턴스를 검색합니다.

```
result = search(:node, "hostname:myhost")
```

검색은 호스트 이름이 `myhost`인 인스턴스 객체의 목록을 반환합니다. 예를 들어 첫 번째 인스턴스의 운영 체제를 원할 경우 `result[0][:os]`에 의해 표현됩니다. 쿼리가 여러 객체를 반환할 경우 객체를 열거하여 필요한 정보를 검색할 수 있습니다.

레시피에서 검색을 사용하는 세부 방법은 Linux 스택 또는 Windows 스택을 사용하는지에 따라 다릅니다. 다음 주제에서는 두 스택 유형 모두에 대한 예제를 제공합니다.

주제

- Linux 스택에서 검색 사용
- Windows 스택에서 검색 사용

Linux 스택에서 검색 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 예제는 단일 PHP 애플리케이션 서버를 포함하는 Linux 스택을 기반으로 합니다. Chef 검색을 사용하여 서버의 퍼블릭 IP 주소를 가져와 `/tmp` 디렉터리의 파일에 이 주소를 저장합니다. 기본적으로 속성 값을 직접 가져오기 와 동일한 정보를 노드 객체에서 검색하지만, 이 코드는 스택 구성 및 배포 속성 구조의 세부 정보에 의존하지 않습니다.

다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 단원을 참조하십시오.

> **Note**
>
> 이전에 AWS OpsWorks Stacks 인스턴스에서 사용자 지정 레시피를 실행하지 않은 경우 먼저 Linux 인스턴스에서 레시피 실행 예제를 검토해야 합니다.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 클릭합니다.
2. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 클릭합니다.

   - 이름 – SearchJSON
   - 기본 SSH 키 – Amazon EC2 키 페어

   Amazon EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다. 이 예에서는 미국 서부(오레곤) 리전을 사용합니다.
3. 계층 추가를 클릭하여 스택에 기본 설정으로 PHP 앱 서버 계층을 추가합니다.
4. 기본 설정을 사용하여 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

쿡북을 설정하려면

1. opsworks_cookbooks 안에 searchjson 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 opstest에 저장합니다.

```
name "searchjson"
version "0.1.0"
```

3. recipes 안에 searchjson 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.

```
phpserver = search(:node, "layers:php-app").first
Chef::Log.info("**********The public IP address is: '#{phpserver[:ip]}'**********")

file "/tmp/ip_addresses" do
  content "#{phpserver[:ip]}"
  mode 0644
```

```
  action :create
end
```

Linux 스택은 node 검색 인덱스만 지원합니다. 레시피는 이 인덱스를 사용하여 php-app 계층의 인스턴스 목록을 얻습니다. 이 계층에는 인스턴스가 하나 뿐인 것으로 알려져 있기 때문에 레시피는 첫 번째 인스턴스만 phpserver에 할당합니다. 계층에 인스턴스가 여러 개인 경우 인스턴스를 열거하여 필요한 정보를 검색할 수 있습니다. 각 목록 항목은 인스턴스 속성 세트가 포함된 해시 테이블입니다. ip 속성은 인스턴스의 퍼블릭 IP 주소로 설정되므로 후속 레시피 코드의 해당 주소를 phpserver[:ip]로 표현할 수 있습니다.

메시지를 Chef 로그에 추가하면 레시피는 file 리소스를 사용하여 ip_addresses 파일을 만듭니다. content 속성은 phpserver[:ip]의 문자열 표현으로 설정됩니다. Chef가 ip_addresses를 생성하면 이 파일에 해당 문자열이 추가됩니다.

5. opsworks_cookbooks의 .zip 아카이브를 생성하고, 이 아카이브를 Amazon S3 버킷에 업로드한 다음, 해당 아카이브를 퍼블릭으로 설정하고, 아카이브의 URL을 기록합니다. 쿡북 리포지토리에 대한 자세한 정보는 쿡북 리포지토리 단원을 참조하세요.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – Http Archive
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 기본값을 사용하고 [저장]을 클릭하여 스택 구성을 업데이트합니다.

2. 사용자 지정 계층 구성을 편집하고 계층의 Setup 이벤트에를 할당searchjson::default합니다. 인스턴스가 부팅된 후 또는 Setup 이벤트를 명시적으로 트리거한 경우 AWS OpsWorks Stacks가 레시피를 실행합니다.
3. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스에 사용자 지정 쿡북 리포지토리의 최신 버전을 설치합니다. 리포지토리의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.

4. [설정] 스택 명령을 실행하여 레시피를 실행합니다. 이 레시피는 인스턴스에서 설정 이벤트를 트리거하고 `searchjson::default`를 실행합니다. [실행 명령 설정] 페이지는 열어 둡니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

searchjson을 확인하려면

1. 가장 먼저, Chef 로그에서 최신 설정 이벤트를 확인합니다. 실행 중인 명령 설정 페이지에서 php-app1 인스턴스의 로그 열에 있는 표시를 클릭하여 로그를 표시합니다. 로그를 중간 지점 근처까지 스크롤하여 다음과 같은 로그 메시지를 찾습니다.

```
...
[2014-09-05T17:08:41+00:00] WARN: Previous
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] WARN: Current
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] INFO: **********The public IP address is:
 '192.0.2.0'**********
[2014-09-05T17:08:41+00:00] INFO: Processing directory[/etc/sysctl.d] action create
 (opsworks_initial_setup::sysctl line 1)
...
```

2. SSH를 사용하여 인스턴스에 로그인하고 `/tmp`의 내용을 나열합니다. 여기에는 IP 주소가 포함된 `ip_addresses` 파일이 있어야 합니다.

Windows 스택에서 검색 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Windows 스택에서 검색을 사용할 수 있는 두 가지 옵션을 제공합니다.

- 표준 Chef 속성을 쿼리하는 데 사용할 수 있는 node 검색 인덱스.

를 사용하는 검색 코드가 있는 기존 레시피가 있는 경우 node일반적으로 수정 없이 AWS OpsWorks Stacks 스택에서 작동합니다.

- AWS OpsWorks Stacks 고유 속성 및 일부 표준적이 특성을 쿼리하는 데 사용할 수 있는 추가 검색 인덱스 세트.

  이러한 인덱스는 Windows AWS OpsWorks Stacks에서 스택별 검색 인덱스 사용 섹션에 설명되어 있습니다.

호스트 이름, IP 주소 등 표준적인 정보를 검색할 때는 node를 사용하는 것이 좋습니다. 이 방법은 레시피를 표준 Chef 관행과 일관되게 유지해줍니다. AWS OpsWorks Stacks 검색 인덱스를 사용하여 AWS OpsWorks Stacks와 관련된 정보를 검색합니다.

주제

- Windows 스택에서 노드 검색 인덱스 사용
- Windows AWS OpsWorks Stacks에서 스택별 검색 인덱스 사용

Windows 스택에서 노드 검색 인덱스 사용

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.

이 예제는 단일 사용자 지정 계층 및 단일 인스턴스를 포함하는 Windows 스택을 기반으로 합니다. Chef 검색을 node 검색 인덱스와 함께 사용하여 서버의 퍼블릭 IP 주소를 가져와 `C:\tmp` 디렉터리에

서 파일에 기록합니다. 다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다.
2. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.
   - 이름 – NodeSearch
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
3. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.
   - 이름 – IPTest
   - 짧은 이름 - iptest
4. 기본 설정을 사용하여 IPTest 계층에 24/7 t2.micro 인스턴스를 추가하고 해당 인스턴스를 시작합니다. 이 인스턴스의 이름은 iptest1입니다.

   AWS OpsWorks Stacks는이 인스턴스AWS-OpsWorks-RDP-Server에를 자동으로 할당하므로 권한 있는 사용자가 인스턴스에 로그인할 수 있습니다.
5. [권한] 및 [편집]을 차례대로 선택하고 [SSH/RDP] 및 [sudo/admin]을 선택합니다. 인스턴스에 로그인하려면 일반 사용자에게는 AWS-OpsWorks-RDP-Server 보안 그룹 이외에 이러한 권한 부여가 필요합니다.

> **Note**
>
> Administrator로도 로그인할 수 있지만 로그인 절차가 다릅니다. 자세한 내용은 RDP를 사용하여 로그인 섹션을 참조하세요.

쿡북을 설정하려면

1. nodesearch 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 opstest에 저장합니다.

```
name "nodesearch"
version "0.1.0"
```

3. recipes 안에 nodesearch 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
  action :create
end

windowsserver = search(:node, "hostname:iptest*").first
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:ipaddress]}'**********")

file 'C:\tmp\addresses.txt' do
  content "#{windowsserver[:ipaddress]}"
  rights :full_control, 'Everyone'
  action :create
end
```

이 레시피는 다음 작업을 수행합니다.

1. 디렉터리 리소스를 사용하여 파일을 위한 C:\tmp 디렉터리를 만듭니다.

   이 리소스에 대한 자세한 정보는 예제 3: 디렉터리 생성 단원을 참조하세요.

2. node 검색 인덱스와 함께 Chef 검색을 사용하여 iptest로 시작하는 호스트 이름이 포함된 노드(인스턴스) 목록을 얻습니다.

   계층의 짧은 이름에 정수를 추가하여 호스트 이름을 생성하는 기본 테마를 사용하는 경우 이 쿼리는 IPTest 계층의 인스턴스를 모두 반환합니다. 예를 들어, 이 계층에는 인스턴스가 하나 뿐인 것으로 알려져 있기 때문에 레시피는 첫 번째 인스턴스만 windowsserver에 할당합니다. 인스턴스가 여러 개인 경우 전체 목록을 가져와 인스턴스를 열거할 수 있습니다.

3. 이 실행을 위해 Chef 로그에 IP 주소와 함께 메시지를 추가합니다.

`windowsserver` 객체는 `ipaddress` 속성이 인스턴스의 퍼블릭 IP 주소로 설정된 해시 테이블이므로, 후속 레시피 코드의 해당 주소를 `windowsserver[:ipaddress]`로 표현할 수 있습니다. 레시피가 해당 문자열을 메시지에 삽입하고 Chef 로그에 추가합니다.

4. `file` 리소스를 사용하여 IP 주소가 포함된 `C:\tmp\addresses.txt`라는 파일을 만듭니다.

   이 리소스의 `content` 속성은 파일에 추가할 내용을 지정하는데, 이 경우에는 퍼블릭 IP 주소입니다.

5. `nodesearch`의 `.zip` 아카이브를 생성하고, 이 아카이브를 S3 버킷에 업로드한 다음, 해당 아카이브를 퍼블릭으로 설정하고, 아카이브의 URL을 기록합니다.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

쿡북을 설치하고 레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.

2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스(온라인 인스턴스 포함)에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어씁니다.
3. 사용자 지정 쿡북 업데이트가 완료되면 실행할 레시피가 **`nodesearch::default`**로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 레시피로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다. execute_recipes 페이지는 그대로 열어 두십시오.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

nodesearch를 확인하려면

1. Chef 로그에서 최신 execute_recipes 이벤트를 확인합니다. 실행 중인 명령 execute_recipes 페이지에서 iptest1 인스턴스의 로그 열에 있는 표시를 선택하여 로그를 표시합니다. 로그를 거의 끝까지 아래로 스크롤하여 다음과 같은 로그 메시지를 찾습니다.

```
...
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb
 in the cache.
[2015-05-13T18:55:47+00:00] INFO: **********The public IP address is:
 '192.0.0.1'**********
[2015-05-13T18:55:47+00:00] INFO: Processing directory[C:\tmp] action create
 (nodesearch::default line 1)
[2015-05-13T18:55:47+00:00] INFO: Processing file[C:\tmp\addresses.txt] action
 create (nodesearch::default line 10)
...
```

2. RDP를 사용하여 인스턴스에 로그인하고 `C:\tmp\addresses.txt`의 내용을 확인합니다.

Windows AWS OpsWorks Stacks에서 스택별 검색 인덱스 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.

AWS OpsWorks Stacks는 외에도 node다음과 같은 검색 인덱스를 제공합니다.

- `aws_opsworks_stack` – 스택 구성.
- `aws_opsworks_layer` – 스택의 계층 구성.
- `aws_opsworks_instance` – 스택의 인스턴스 구성.
- `aws_opsworks_app` – 스택의 앱 구성.
- `aws_opsworks_user` – 스택의 사용자 구성.
- `aws_opsworks_rds_db_instance` – 등록된 RDS 인스턴스의 연결 정보.

이러한 인덱스에는 일부 표준 Chef 속성이 포함되지만 주로 AWS OpsWorks Stacks별 속성을 검색하기 위한 것입니다. 예를 들어 `aws_opsworks_instance`는 `online`과 같은 인스턴스 상태를 제공하는 `status` 포함합니다.

> **Note**
>
> 권장되는 방법은 가능할 경우 node를 사용하여 레시피를 표준 Chef 사용과 일관되게 유지하는 것입니다. 예제는 Windows 스택에서 노드 검색 인덱스 사용 섹션을 참조하세요.

이 예제에서는 AWS OpsWorks Stacks 인덱스를 사용하여 AWS OpsWorks Stacks별 속성의 값을 검색하는 방법을 보여줍니다. 이 예제는 단일 인스턴스의 단일 사용자 지정 계층을 포함하는 단순한 Windows 스택을 기반으로 합니다. Chef 검색을 사용하여 인스턴스의 AWS OpsWorks Stacks ID를 가져오고 결과를 Chef 로그에 넣습니다.

다음은 이 예제를 위한 스택을 생성하는 방법을 간략히 요약한 것입니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 + Stack(+ 스택)을 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.

   - 이름 – IDSearch
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

- 이름 – IDCheck
- 짧은 이름 - idcheck

3. 기본 설정을 사용하여 IDCheck 계층에 24/7 t2.micro 인스턴스를 추가하고 해당 인스턴스를 시작합니다. 이 인스턴스의 이름은 iptest1입니다.

   AWS OpsWorks Stacks는이 인스턴스AWS-OpsWorks-RDP-Server에 자동으로 할당합니다.는 권한 있는 사용자가 인스턴스에 로그인할 수 있도록 허용하는 인바운드 규칙을이 보안 그룹에 추가하는 방법을 RDP 액세스 활성화 설명합니다.

4. [권한] 및 [편집]을 차례대로 선택하고 [SSH/RDP] 및 [sudo/admin]을 선택합니다. 인스턴스에 로그인하려면 일반 사용자에게는 AWS-OpsWorks-RDP-Server 보안 그룹 이외에 이러한 권한 부여가 필요합니다.

   > **Note**
   >
   > Administrator로도 로그인할 수 있지만 로그인 절차가 다릅니다. 자세한 내용은 RDP를 사용하여 로그인 섹션을 참조하세요.

쿡북을 설정하려면

1. idcheck 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 opstest에 저장합니다.

```
name "idcheck"
version "0.1.0"
```

3. idcheck 안에 recipes 디렉터리를 만들고 다음 레시피가 포함된 default.rb 파일을 이 디렉터리에 추가합니다.

```
windowsserver = search(:aws_opsworks_instance, "hostname:idcheck*").first
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:instance_id]}'**********")
```

   이 레시피는 aws_opsworks_instance 검색 인덱스와 함께 Chef 검색을 사용하여 스택의 각 인스턴스에 대한 인스턴스 속성을 idcheck로 시작하는 호스트 이름과 함께 가져옵니다. 계층

의 짧은 이름에 정수를 추가하여 호스트 이름을 생성하는 기본 테마를 사용하는 경우 이 쿼리는 IDCheck 계층의 인스턴스를 모두 반환합니다. 예를 들어, 이 계층에는 인스턴스가 하나 뿐인 것으로 알려져 있기 때문에 레시피는 첫 번째 인스턴스만 `windowsserver`에 할당합니다. 인스턴스가 여러 개인 경우 전체 목록을 가져와 인스턴스를 열거할 수 있습니다.

이 레시피는 스택에는 이 호스트 이름을 가진 인스턴스가 하나 뿐이라는 사실을 활용하므로 첫 번째 결과는 올바른 인스턴스 하나입니다. 스택에 인스턴스가 여러 개인 경우 다른 속성을 검색하면 결과가 두 개 이상 반환될 수 있습니다. 인스턴스 속성의 목록은 인스턴스 데이터 백(aws_opsworks_instance) 단원을 참조하세요.

인스턴스 속성은 기본적으로 해시 테이블이며 인스턴스의 AWS OpsWorks Stacks ID가 `instance_id` 속성에 할당되므로 ID를 로 참조할 수 있습니다`windowsserver[:instance_id]`. 레시피가 해당 문자열을 메시지에 삽입하고 Chef 로그에 추가합니다.

4. `ipaddress` 쿡북의 `.zip` 아카이브를 생성하고, 이 아카이브를 Amazon S3 버킷에 업로드한 다음, 해당 아카이브의 URL을 기록합니다. 쿡북 리포지토리에 대한 자세한 정보는 쿡북 리포지토리 단원을 참조하세요.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

쿡북을 설치하고 레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.

2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스(온라인 인스턴스 포함)에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어씁니다.

3. 사용자 지정 쿡북 업데이트가 완료되면 실행할 레시피가 **idcheck::default**로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 레시피로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다. execute_recipes 페이지는 그대로 열어 두십시오.

레시피가 성공적으로 실행되면 Chef 로그에서 가장 최근의 execute_recipes 이벤트를 보고 이를 확인할 수 있습니다. 실행 중인 명령 execute_recipes 페이지에서 iptest1 인스턴스의 로그 열에 있는 표시를 선택하여 로그를 표시합니다. 로그를 거의 끝까지 아래로 스크롤하여 다음과 같은 로그 메시지를 찾습니다.

```
...
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb in
 the cache.
[2015-05-13T20:03:47+00:00] INFO: **********The instance ID is: 'i-8703b570'**********
[2015-05-13T20:03:47+00:00] INFO: Chef Run complete in 0.312518 seconds
...
```

Linux 인스턴스에서 외부 쿡북 사용: Berkshelf

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> Berkshelf는 Chef 11.10 Linux 스택에서만 사용할 수 있습니다.

쿡북을 구현하기 전에 Chef 커뮤니티의 구성원들이 매우 다양한 용도를 위해 생성한 쿡북이 포함되어 있는 Chef 커뮤니티 쿡북을 확인하세요. 이러한 쿡북은 대부분 수정 없이 AWS OpsWorks Stacks에서 사용할 수 있으므로 모든 코드를 직접 구현하는 대신 일부 작업에 활용할 수 있습니다.

인스턴스에서 외부 쿡북을 사용하려면 쿡북을 설치하고 종속성을 관리할 방법이 필요합니다. 선호되는 방법은 Berkshelf라는 종속성 관리자를 지원하는 쿡북을 구현하는 것입니다. Berkshelf는 AWS OpsWorks Stacks 인스턴스를 포함한 Amazon EC2 인스턴스에서 작동하지만 Test Kitchen 및 Vagrant에서도 작동하도록 설계되었습니다. 그러나 Vagrant의 사용량은 AWS OpsWorks Stacks와 약간 다르므로이 주제에는 두 플랫폼에 대한 예제가 포함되어 있습니다. Berkshelf를 사용하는 방법에 대한 자세한 정보는 Berkshelf를 참조하세요.

주제

- Test Kitchen 및 Vagrant에서 Berkshelf 사용
- AWS OpsWorks Stacks에서 Berkshelf 사용

Test Kitchen 및 Vagrant에서 Berkshelf 사용

이 예제는 Berkshelf를 사용하여 시작하기 커뮤니티 쿡북을 설치하고 해당 레시피를 실행하는 방법을 보여줍니다. 이 레시피는 인스턴스의 홈 디렉터리에 간단한 텍스트 파일을 설치합니다.

Berkshelf를 설치하고 쿡북을 초기화하려면

1. 워크스테이션에 다음과 같이 Berkshelf Gem을 설치합니다.

   ```
   gem install berkshelf
   ```

   워크스테이션에 따라 이 명령에는 sudo가 필요할 수 있습니다. 또는 Ruby 환경 관리자(예: RVM)를 사용할 수도 있습니다. Berkshelf가 성공적으로 설치되었는지 확인하려면 `berks --version`을 실행합니다.

2. 이 주제의 쿡북 이름은 external_cookbook입니다. 이전 주제에서 채택한 수동 접근 방식 대신 Berkshelf를 사용하여 초기화된 쿡북을 생성할 수 있습니다. 이렇게 하려면 `opsworks_cookbooks` 디렉터리로 이동하여 다음 명령을 실행합니다.

   ```
   berks cookbook external_cookbook
   ```

   이 명령은 `external_cookbook` 디렉터리와 `recipes` 및 `test`를 비롯하여 Chef 및 Test Kitchen 하위 디렉터리를 만듭니다. 또한 이 명령은 다음을 비롯하여 여러 표준 파일의 기본 버전을 생성합니다.

   - `metadata.rb`
   - Vagrant, Test Kitchen 및 Berkshelf용 구성 파일

- `default.rb` 디렉터리의 빈 `recipes` 레시피

> **Note**
>
> `kitchen init`를 실행할 필요가 없습니다. `berks cookbook` 명령이 이러한 작업을 처리하기 때문입니다.

3. `kitchen converge`을(를) 실행합니다. 새로 생성된 쿡북은 이 시점에서 어떠한 의미 있는 작업을 수행하지 않지만 수렴을 수행합니다.

> **Note**
>
> 또한 `berks init`를 사용하여 Berkshelf를 사용하도록 기존 쿡북을 초기화할 수도 있습니다.

Berkshelf를 사용하여 쿡북의 외부 종속성을 관리하려면 쿡북의 루트 디렉터리에 Berkshelf가 종속성을 관리하는 방법을 지정하는 구성 파일인 `Berksfile`이 포함되어야 합니다. `berks cookbook`을 사용하여 `external_cookbook` 쿡북을 생성한 경우 다음 콘텐츠를 포함하는 `Berksfile`이 생성되었습니다.

```
source "https://supermarket.chef.io"
metadata
```

이 파일에는 다음 선언이 포함됩니다.

- `source` – 쿡북 소스의 URL.

  Berksfile에는 각각 종속 쿡북의 기본 소스를 지정하는 `source` 선언이 여러 개 포함될 수 있습니다. 쿡북의 소스를 명시적으로 지정하지 않을 경우 Berkshelf는 기본 리포지토리에서 동일한 이름의 쿡북을 찾습니다. 기본 Berksfile에는 커뮤니티 쿡북 리포지토리를 지정하는 `source` 속성이 포함됩니다. 이 리포지토리에 시작하기 쿡북이 들어 있습니다. 따라서 이 줄은 그대로 둘 수 있습니다.

- `metadata` - Berkshelf에게 쿡북의 `metadata.rb` 파일에서 선언된 쿡북 종속성을 포함하도록 지시합니다.

  나중에 설명하듯이 `cookbook` 속성을 포함시켜 Berksfile에서 종속 쿡북을 선언할 수도 있습니다.

쿡북 종속성을 선언하는 방법은 두 가지입니다.

- Berksfile에 cookbook 선언을 포함.

  이는 AWS OpsWorks Stacks에서 사용하는 접근 방식입니다. 예를 들어 이 예제에서 사용되는 시작하기 쿡북을 지정하려면 Berksfile에 cookbook "getting-started"를 포함시킵니다. 그러면 Berkshelf가 기본 리포지토리에서 동일한 이름의 쿡북을 찾습니다. 또한 cookbook을 사용하여 쿡북 소스는 물론 특정 버전까지 명시적으로 지정할 수도 있습니다. 자세한 정보는 Berkshelf를 참조하세요.
- Berksfile에 metadata 선언을 포함하고 metadata.rb에서 종속성을 선언.

  이 선언은 Berkshelf에게 쿡북의 metadata.rb 파일에서 선언된 쿡북 종속성을 포함하도록 지시합니다. 예를 들어 시작하기 종속성을 선언하려면 depends 'getting-started' 선언을 쿡북의 metadata.rb 파일에 추가합니다.

이 예제에서는 AWS OpsWorks Stacks와의 일관성을 위해 첫 번째 접근 방식을 사용합니다.

getting-started 쿡북을 설치하려면

1. 기본 Berksfile을 편집하여 metadata 선언을 getting-started에 대한 cookbook 선언으로 바꿉니다. 내용은 다음과 같아야 합니다.

   ```
   source "https://supermarket.chef.io"

   cookbook 'getting-started'
   ```

2. 커뮤니티 쿡북 리포지토리에서 워크스테이션의 Berkshelf 디렉터리(일반적으로 ~/.berkshelf)로 getting-started 쿡북을 다운로드하는 berks install을 실행합니다. 일반적으로 이 디렉터리는 the Berkshelf라고 합니다. Berkshelf의 cookbooks 디렉터리를 살펴보면 getting-started 쿡북의 디렉터리가 있어야 하고 이 디렉터리의 이름은 getting-started-0.4.0과 유사합니다.
3. external_cookbook::default 실행 목록의 .kitchen.yml를 getting-started::default로 바꿉니다. 이 예제는 external_cookbook의 레시피를 실행하지 않고 기본적으로 getting-started 쿡북을 사용하는 방식입니다. .kitchen.yml 파일은 이제 다음과 유사해야 합니다.

   ```
   ---
   driver:
   ```

```
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
      - recipe[getting-started::default]
    attributes:
```

4. `kitchen converge`를 실행한 다음 `kitchen login`을 사용하여 인스턴스에 로그인합니다. 로그인 디렉터리에는 내용이 다음과 같은 `chef-getting-started.txt` 파일이 포함되어 있어야 합니다.

   ```
   Welcome to Chef!

   This is Chef version 11.12.8.
   Running on ubuntu.
   Version 12.04.
   ```

   Test Kitchen은 인스턴스의 `/tmp/kitchen/cookbooks` 디렉터리에 쿡북을 설치합니다. 이 디렉터리의 내용을 나열하면 external_cookbook 및 getting-started, 이렇게 2개의 쿡북이 있습니다.

5. `kitchen destroy`를 실행하여 인스턴스를 종료합니다. 다음 예제에서는 AWS OpsWorks Stacks 인스턴스를 사용합니다.

AWS OpsWorks Stacks에서 Berkshelf 사용

AWS OpsWorks Stacks는 선택적으로 Chef 11.10 스택용 Berkshelf를 지원합니다. 스택에서 Berkshelf를 사용하려면 다음을 수행해야 합니다.

- 스택에서 Berkshelf를 활성화합니다.

  AWS OpsWorks 그런 다음 Stacks는 스택의 인스턴스에 Berkshelf를 설치하는 세부 정보를 처리합니다.

- Berksfile을 리포지토리의 루트 디렉터리에 추가합니다.

Berksfile은 모든 종속 쿡북에 대해 `source` 및 `cookbook` 선언을 포함해야 합니다.

Stacks는 인스턴스에 사용자 지정 AWS OpsWorks 쿡북 리포지토리를 설치할 때 Berkshelf를 사용하여 리포지토리의 Berksfile에 선언된 종속 쿡북을 설치합니다. 자세한 내용은 Berkshelf 사용 단원을 참조하십시오.

이 예제에서는 Berkshelf를 사용하여 Stacks 인스턴스에 시작하기 커뮤니티 AWS OpsWorks 쿡북을 설치하는 방법을 보여줍니다. 또한 지정된 디렉터리에 파일을 생성하는 createfile 사용자 지정 쿡북도 설치합니다. createfile 파일이 작동하는 방식에 대한 자세한 정보는 쿡북에서 파일 설치 단원을 참조하세요.

> **Note**
>
> Stacks 스택에 사용자 지정 AWS OpsWorks 쿡북을 처음 설치한 경우 먼저 Linux 인스턴스에서 레시피 실행 예제를 살펴봐야 합니다.

다음에 요약된 대로 스택을 생성하여 시작합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 클릭합니다.
2. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 클릭합니다.

   - 이름 – BerksTest
   - 기본 SSH 키 – Amazon EC2 키 페어

   Amazon EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다. 이 예에서는 기본 미국 서부(오레곤) 리전을 사용합니다.
3. [계층 추가]를 클릭하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – BerksTest
   - 짧은 이름 - berkstest

이 예제에는 실제로 모든 계층 유형을 사용할 수 있습니다. 그러나 이 예제에서는 다른 계층에서 설치한 패키지가 필요하지 않기 때문에 사용자 지정 계층을 사용하는 것이 가장 간단합니다.

4. 기본 설정을 사용하여 BerksTest 계층에 24/7 인스턴스를 추가하되 해당 인스턴스를 아직 시작하지는 마십시오.

Stacks에서는 AWS OpsWorks 쿡북이 표준 디렉터리 구조의 원격 리포지토리에 있어야 합니다. 그런 다음 Stacks에 다운로드 정보를 제공합니다. 그러면 시작 시 AWS OpsWorks 스택의 각 인스턴스에 리포지토리가 자동으로 다운로드됩니다. 간소화를 위해이 예제의 리포지토리는 퍼블릭 Amazon S3 아카이브이지만 Stacks는 HTTP 아카이브, Git AWS OpsWorks 리포지토리 및 하위 버전 리포지토리도 지원합니다. 자세한 내용은 쿡북 리포지토리 단원을 참조하십시오.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

쿡북 리포지토리를 생성하려면

1. opsworks_cookbooks 디렉터리에서 berkstest_cookbooks라는 디렉터리를 만듭니다. 이 디렉터리는 리포지토리에 업로드할 것이므로 원하는 경우 편리한 위치면 어디에든 이 디렉터리를 만들 수 있습니다.
2. 다음 내용이 포함된 Berksfile이라는 파일을 berkstest_cookbooks에 추가합니다.

```
source "https://supermarket.chef.io"

cookbook 'getting-started'
```

   이 파일은 getting-started 쿡북 종속성을 선언하고 Berkshelf에 커뮤니티 쿡북 사이트에서 종속성을 다운로드하도록 지시합니다.

3. createfile 디렉터리를 다음 항목이 포함된 berkstest_cookbooks에 추가합니다.

   - 다음 내용이 포함된 metadata.rb 파일

```
name "createfile"
version "0.1.0"
```

- 다음 내용이 들어 있는 `files/default` 파일이 포함된 `example_data.json` 디렉터리

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

  파일의 이름 및 내용은 임의적입니다. 이 레시피는 지정된 위치에 파일을 복사하기만 합니다.

- 다음 레시피 코드가 들어 있는 `recipes` 파일이 포함된 `default.rb` 디렉터리

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

  이 레시피는 `/srv/www/shared`를 만들고, 쿡북의 `files` 디렉터리에서 이 디렉터리로 `example_data.json`을 복사합니다.

4. `berkstest_cookbooks`의 `.zip` 아카이브를 생성하고, 이 아카이브를 Amazon S3 버킷에 업로드한 다음, 해당 아카이브를 퍼블릭으로 설정하고, 아카이브의 URL을 기록합니다.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

쿡북을 설치하고 레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – Http Archive

- 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL
- Berkshelf 관리 – 예

처음 두 설정은 AWS OpsWorks Stacks에 쿡북 리포지토리를 인스턴스에 다운로드하는 데 필요한 정보를 제공합니다. 마지막 설정은 인스턴스로 getting-started 쿡북을 다운로드하는 Berkshelf 지원을 활성화합니다. 기타 설정에 대해 기본값을 수락하고 [저장]을 클릭하여 스택 구성을 업데이트합니다.

2. BerksTest 계층을 편집하여 다음 레시피를 계층의 설정 수명 주기 이벤트에 추가합니다.

   - `getting-started::default`
   - `createfile::default`

3. 인스턴스를 시작합니다. Setup 이벤트는 인스턴스 부팅이 완료된 후에 발생합니다. 그런 다음 AWS OpsWorks Stacks는 쿡북 리포지토리를 설치하고, Berkshelf를 사용하여 시작하기 쿡북을 다운로드하고, `getting-started::default` 및를 포함한 계층의 설정 및 배포 레시피를 실행합니다`createfile::default`.
4. 인스턴스가 온라인 상태가 되면 SSH를 사용하여 로그인합니다. 다음과 같은 모양이어야 합니다.

   - `/srv/www/shared`에는 `example_data.json`이 포함되어 있어야 합니다.
   - `/root`에는 `chef-getting-started.txt`이 포함되어 있어야 합니다.

     AWS OpsWorks Stacks는 레시피를 루트로 실행하므로 시작하기를 통해 홈 `/root` 디렉터리가 아닌 디렉터리에 파일이 설치됩니다.

SDK for Ruby 사용: Amazon S3에서 파일 다운로드

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 서비스와 통합과 같은 일부 작업은 Chef 리소스를 사용하여 처리할 수 없습니다. 예를 들어 파일을 원격에 저장하고 레시피가 인스턴스로 다운로드하게 하는 것이 더 좋을 때가 있습니다. remote_file 리소스를 사용하여 원격 서버에서 파일을 다운로드할 수 있습니다. 하지만 Amazon S3 버킷에 파일을

저장하려는 경우 `remote_file`는 ACL에서 작업을 허용하는 경우에만 해당 파일을 다운로드할 수 있습니다.

레시피는 AWS SDK for Ruby를 사용하여 대부분의 AWS 서비스에 액세스할 수 있습니다. 이 주제는 SDK for Ruby를 사용하여 S3 버킷에서 파일을 다운로드하는 방법을 보여줍니다.

> **Note**
>
> AWS SDK for Ruby를 사용하여 암호화 및 암호 해독을 처리하는 방법에 대한 자세한 내용은 AWS::S3::S3Object를 참조하세요. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

주제

- Vagrant 인스턴스에서 SDK for Ruby 사용
- AWS OpsWorks Stacks Linux 인스턴스에서 SDK for Ruby 사용
- AWS OpsWorks Stacks Windows 인스턴스에서 SDK for Ruby 사용

Vagrant 인스턴스에서 SDK for Ruby 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는 Vagrant 인스턴스에서 실행되는 레시피가 어떻게 AWS SDK for Ruby를 사용하여 Amazon S3에서 파일을 다운로드할 수 있는지에 대해 설명합니다. 시작하기 전에 먼저 레시피가 Amazon S3에 액세스할 수 있도록 허용하는 액세스 키와 보안 액세스 키라는 AWS 자격 증명 세트가 있어야 합니다.

> **Important**
>
> 이 목적으로 루트 계정 자격 증명은 사용하지 않는 것이 좋습니다. 그 대신, 적절한 정책으로 사용자를 생성하고 레시피에 해당 자격 증명을 제공합니다.

자격 증명을 포함한 파일을 공개 GitHub 또는 Bitbucket 저장소에 업로드하는 등 공개적으로 액세스할 수 있는 위치에 자격 증명(IAM 사용자 자격 증명 포함)을 보관하지 않도록 주의하세요. 그러면 자격 증명이 노출되고 계정의 보안이 훼손될 수 있습니다.
EC2Amazon EC2 인스턴스에서 실행되는 레시피는 더욱 뛰어난 방식인 IAM 역할을 사용할 수 있습니다(AWS OpsWorks Stacks Linux 인스턴스에서 SDK for Ruby 사용 단원 참조).
Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

적절한 사용자가 아직 없는 경우 다음과 같이 생성할 수 있습니다. 자세한 내용은 IAM이란?을 참조하세요.

Warning

IAM 사용자는 장기 자격 증명을 가지므로 보안 위험이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

IAM 사용자 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택하고 필요한 경우 사용자 추가를 선택하여 신규 관리 사용자를 만듭니다.
3. 권한 설정 페이지에서 정책을 직접 연결을 선택합니다.
4. 권한 정책 검색 상자에 **S3**을 입력하여 Amazon S3 정책을 표시합니다.

   AmazonS3ReadOnlyAccess를 선택합니다. 원하는 경우 AmazonS3FullAccess와 같이 더 넓은 권한을 부여하는 정책을 지정할 수 있지만 필요한 권한만 부여하는 것이 일반적입니다. 이 경우, 레시피는 파일만 다운로드하므로 읽기 전용 액세스로 충분합니다.
5. 다음을 선택합니다.
6. 사용자 생성을 선택합니다.
7. 그다음에 사용자에 대해 액세스 키를 생성합니다. 액세스 키 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 사용자를 위한 액세스 키 관리를 참조하세요.

다음에는 다운로드할 파일을 제공해야 합니다. 이 예제에서는 새로 생성된 `myfile.txt`이라는 S3 버킷에 `cookbook_bucket`라는 파일을 저장하는 것으로 가정합니다.

다운로드할 파일을 제공하려면

1. 다음 텍스트가 포함된 `myfile.txt` 파일을 만든 다음 워크스테이션의 원하는 위치에 저장합니다.

   ```
   This is the file that you just downloaded from Amazon S3.
   ```

2. Amazon S3 cookbook_bucket 콘솔에서 표준 리전에 라는 이름의 버킷을 생성하고 `myfile.txt`를 버킷에 업로드합니다.

다음과 같이 쿡북을 설정합니다.

쿡북을 설정하려면

1. `opsworks_cookbooks` 안에 `s3bucket` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 예제 1: 패키지 설치 단원에서 설명하는 대로 Test Kitchen을 초기화 및 구성합니다.
3. `.kitchen.yml`의 텍스트를 다음으로 바꿉니다.

   ```
   ---
   driver:
     name: vagrant

   provisioner:
     name: chef_solo
     environments_path: ./environments

   platforms:
     - name: ubuntu-14.04

   suites:
     - name: s3bucket
       provisioner:
         solo_rb:
           environment: test
       run_list:
         - recipe[s3bucket::default]
   ```

```
    attributes:
```

4. s3bucket에 디렉터리 recipes 및 environments를 추가합니다.
5. 다음 default_attributes 섹션을 사용하여 환경 파일 test.json을 만들고 사용자에 해당하는 키로 access_key 및 secret_key 값을 바꿉니다. 쿡북의 environments 폴더에 파일을 저장합니다.

```
{
  "default_attributes" : {
    "cookbooks_101" : {
      "access_key": "AKIAIOSFODNN7EXAMPLE",
      "secret_key" : "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

다양한 방법으로 인스턴스에서 실행되는 레시피에 자격 증명을 제공할 수 있습니다. 중요한 고려 사항은 잘못하여 키가 노출되어 계정 보안이 훼손될 가능성을 제한하는 것입니다. 이를 위해 코드에 명시적 키 값을 사용하는 것은 권장하지 않습니다. 예제에서는 키 값을 노드 객체에 저장합니다. 그러면 레시피가 리터럴 값을 노출하는 대신 노드 구문을 사용하여 참조할 수 있습니다. 노드 객체에 액세스하려면 루트 권한이 있어야 합니다. 이는 키가 노출될 가능성을 제한합니다. 자세한 내용은 AWS 액세스 키 관리 모범 사례를 참조하세요.

> **Note**
>
> 이 예제에서는 cookbooks_101을 첫 번째 요소로 하는 중첩 속성을 사용합니다. 이 방법은 노드 객체에 다른 access_key 또는 secret_key 속성이 있을 경우 이름 충돌의 가능성을 줄여 줍니다.

다음 레시피는 myfile.text 버킷에서 cookbook_bucket 파일을 다운로드합니다.

```
gem_package "aws-sdk ~> 3" do
  action :install
end
```

```
ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = Aws::S3::Client.new(
          :access_key_id => "#{node['cookbooks_101']['access_key']}",
          :secret_access_key => "#{node['cookbooks_101']['secret_key']}")

    myfile = s3.bucket['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.write(myfile.read)
      f.close
    end
  end
  action :run
end
```

레시피의 첫 번째 부분은 젬 패키지인 SDK for Ruby를 설치합니다. gem_package 리소스는 레시피 또는 다른 애플리케이션에서 사용될 젬을 설치합니다.

> **Note**
>
> 사용자의 인스턴스에는 일반적으로 두 개의 Ruby 인스턴스가 있으며, 이 두 개의 인스턴스는 일반적으로 버전이 다릅니다. 하나는 Chef 클라이언트가 사용하는 전용 인스턴스입니다. 다른 하나는 인스턴스에서 실행되는 애플리케이션 및 레시피가 사용합니다. 젬 설치를 위한 리소스는 gem_package 및 chef_gem 두 개가 있으므로 젬 패키지를 설치할 때 이 구분을 이해하는 것이 중요합니다. 애플리케이션 또는 레시피가 젬 패키지를 사용하는 경우 gem_package를 사용하여 젬 패키지를 설치합니다. chef_gem은 Chef 클라이언트가 사용하는 젬 패키지에만 사용합니다.

레시피의 나머지 부분은 ruby_block 리소스입니다. 여기에 파일을 다운로드하는 Ruby 코드가 포함됩니다. 레시피가 Ruby 애플리케이션이기 때문에 코드를 레시피에 직접 배치할 수 있다고 생각할 수 있습니다. 하지만 Chef 실행은 리소스를 실행하기 전에 모든 코드를 컴파일합니다. 예제 코드를 레시피에 직접 배치할 경우, Ruby는 gem_package 리소스를 실행하기 전에 `require 'aws-sdk'` 문을 해결하려고 시도합니다. SDK for Ruby가 아직 설치되지 않았으므로 컴파일이 실패합니다.

ruby_block 리소스 내 코드는 해당 리소스가 실행될 때까지는 컴파일되지 않습니다. 이 예제에서 gem_package 리소스가 SDK for Ruby 설치를 마친 후에 ruby_block 리소스가 실행됩니다. 그러므로 코드가 성공적으로 실행될 것입니다.

ruby_block 내 코드는 다음과 같이 작동합니다.

1. 새 Aws::S3 객체를 생성합니다. 이 객체는 서비스 인터페이스를 제공합니다.

   액세스 키와 보안 키는 노드 객체에 저장된 값 단원을 참조하여 지정됩니다.
2. S3 객체의 bucket.objects 연결을 호출하여 이를 myfile.txt을 나타내는 myfile 이름의 Aws::S3::Object 객체를 반환합니다.
3. Dir.chdir을 사용하여 작업 디렉터리를 /tmp로 설정합니다.
4. myfile.txt 파일을 열고, 파일에 myfile을 기록하고, 파일을 닫습니다.

레시피를 실행하려면

1. 예제 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.
2. kitchen converge을(를) 실행합니다.
3. kitchen login을 실행하여 인스턴스에 로그인한 다음, ls /tmp를 실행합니다. 여러 Test Kitchen 파일 및 디렉터리와 함께 myfile.txt가 있어야 합니다.

   ```
   vagrant@s3bucket-ubuntu-1204:~$ ls /tmp
   install.sh  kitchen  myfile.txt  stderr
   ```

   또한 cat /tmp/myfile.txt를 실행하여 파일의 내용이 올바른지 확인할 수 있습니다.

다 마치면 kitchen destroy를 실행해 인스턴스를 종료하세요.

AWS OpsWorks Stacks Linux 인스턴스에서 SDK for Ruby 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에서는 AWS OpsWorks Stacks Linux 인스턴스에서 SDK for Ruby를 사용하여 Amazon S3 버킷에서 파일을 다운로드하는 방법을 설명합니다. AWS OpsWorks Stacks는 모든 Linux 인스턴스에 SDK for Ruby를 자동으로 설치합니다. 하지만 서비스의 클라이언트 객체를 생성할 때는 적절한 AWS 자격 증명 `AWS::S3.new`(또는 다른 서비스의 동등한 값)를 제공해야 합니다.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

Vagrant 인스턴스에서 SDK for Ruby 사용 섹션은 노드 객체에 자격 증명을 저장하고 레시피 코드에서 속성 단원을 참조하는 식으로 자격 증명 노출 위험을 완화하는 방법을 보여줍니다. Amazon EC2 인스턴스에서 레시피를 실행하는 경우 IAM 역할이라는 훨씬 더 좋은 옵션을 사용할 수 있습니다.

IAM 역할은 IAM 사용자와 거의 비슷하게 작동합니다. 여기에는 다양한 AWS 서비스를 사용할 수 있는 권한을 부여하는 정책이 연결됩니다. 하지만 역할은 개인이 아니라 Amazon EC2 인스턴스에 할당합니다. 그러면 이 인스턴스에서 실행되는 애플리케이션은 연결된 정책이 부여하는 권한을 획득할 수 있습니다. 역할을 사용하면 자격 증명이 간접적으로라도 코드에 표시되지 않습니다. 이 주제에서는 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 Vagrant 인스턴스에서 SDK for Ruby 사용의 레시피를 실행하는 방법을 설명합니다.

예제 9: Amazon EC2 인스턴스 사용 섹션에 설명된 대로 kitchen-ec2 드라이버를 사용하여 이 레시피를 Test Kitchen과 함께 실행할 수 있습니다. 하지만 SDK for Ruby를 Amazon EC2 인스턴스에 설치하는 것은 다소 복잡하며 AWS OpsWorks Stacks에 대해 걱정할 필요가 없습니다. 모든 AWS OpsWorks Stacks Linux 인스턴스에는 기본적으로 SDK for Ruby가 설치되어 있습니다. 따라서 간소화를 위해이 예제에서는 AWS OpsWorks Stacks 인스턴스를 사용합니다.

첫 번째 단계는 IAM 역할을 설정하는 것입니다. 이 예제에서는 가장 간단한 접근 방식을 취합니다. 즉, AWS OpsWorks Stacks가 첫 번째 스택을 생성할 때 생성하는 Amazon EC2 역할을 사용하는 것입니다. 이 역할은 `aws-opsworks-ec2-role`로 명명됩니다. 그러나 AWS OpsWorks Stacks는 정책을 해당 역할에 연결하지 않으므로 기본적으로 권한을 부여하지 않습니다.

적절한 권한을 부여하려면 `AmazonS3ReadOnlyAccess` 정책을 `aws-opsworks-ec2-role` 역할에 연결해야 합니다. 정책을 역할에 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

스택을 생성하거나 업데이트할 때 역할을 지정합니다. Linux 인스턴스에서 레시피 실행 섹션에 설명된 대로 사용자 지정 계층을 포함하는 스택을 설정하되 한 가지를 추가합니다. 스택 추가 페이지에서 기본 IAM 인스턴스 프로파일이 aws-opsworks-ec2-role로 설정되어 있는지 확인합니다. 그러면 AWS OpsWorks 스택이 스택의 모든 인스턴스에 해당 역할을 할당합니다.

쿡북을 설정하는 절차는 Linux 인스턴스에서 레시피 실행 섹션에서 사용한 것과 비슷합니다. 다음은 간략한 요약이며, 자세한 정보는 해당 예제를 참조해야 합니다.

쿡북을 설정하려면

1. s3bucket_ops 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 s3bucket_ops에 저장합니다.

```
name "s3bucket_ops"
version "0.1.0"
```

3. recipes 안에 s3bucket_ops 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다.

```
Chef::Log.info("******Downloading a file from Amazon S3.******")

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = AWS::S3.new

    myfile = s3.buckets['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.syswrite(myfile.read)
      f.close
    end
  end
  action :run
end
```

5. s3bucket_ops의 .zip 아카이브를 만들고 그 아카이브를 Amazon S3 버킷에 업로드합니다. 간단한 설명을 위해 아카이브를 퍼블릭으로 설정한 다음 나중에 사용하기 위해 아카이브의 URL을 적어 둡니다. 프라이빗 Amazon S3 아카이브 또는 기타 여러 가지 리포지토리 유형에 쿡북을 저장할 수도 있습니다. 자세한 내용은 쿡북 리포지토리 섹션을 참조하세요.

이 레시피는 다음을 제외하면 이전 예제에서 사용된 것과 비슷합니다.

- AWS OpsWorks Stacks는 이미 SDK for Ruby를 설치했으므로 `chef_gem` 리소스가 삭제되었습니다.
- 레시피가 `AWS::S3.new`로 자격 증명을 전달하지 않습니다.

  자격 증명이 인스턴스의 역할에 따라 자동으로 애플리케이션에 할당됩니다.
- 레시피는 `Chef::Log.info`를 사용하여 Chef 로그에 메시지를 추가합니다.

다음과 같이 이 예제를 위한 스택을 생성합니다. 기존의 Windows 스택을 사용할 수 있습니다. 나중에 설명하듯이 쿡북을 업데이트하면 됩니다.

스택 생성

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 클릭합니다.
2. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 클릭합니다.

   - 이름 – RubySDK
   - 기본 SSH 키 – Amazon EC2 키 페어

   Amazon EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다. 이 예에서는 기본 미국 서부(오레곤) 리전을 사용합니다.
3. [계층 추가]를 클릭하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – S3Download
   - 짧은 이름 - s3download

   모든 계층 유형이 실제로 Linux 스택에서 작동하지만 이 예제에서는 다른 계층 유형에서 설치한 패키지가 필요하지 않기 때문에 사용자 지정 계층을 사용하는 것이 가장 간단합니다.
4. 기본 설정을 사용하여 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

이제 레시피를 설치하고 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

- 리포지토리 유형 – Http Archive
- 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

기타 설정에 기본값을 사용하고 [저장]을 클릭하여 스택 구성을 업데이트합니다.

2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.
3. 실행할 레시피가 **s3bucket_ops::default**으로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 `s3bucket_ops::default`로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다.

> **Note**
>
> 일반적으로 AWS OpsWorks Stacks가 레시피를 적절한 수명 주기 이벤트에 할당하여 레시피를 자동으로 실행하도록 합니다. 수동으로 이벤트를 트리거하여 그러한 레시피를 실행할 수 있습니다. 스택 명령을 사용하여 설정 및 Configure 이벤트를 트리거할 수 있고, 배포 명령을 사용하여 Deploy 및 Undeploy 이벤트를 트리거할 수 있습니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

s3bucket_ops를 확인하려면

1. 가장 먼저 Chef 로그를 확인합니다. 스택에는 opstest1이라는 인스턴스가 하나 있어야 합니다. 인스턴스 페이지에서 인스턴스의 로그 열에 있는 표시를 클릭하여 Chef 로그를 표시합니다. 아래로 스크롤하면 맨 아래 근처에 로그 메시지가 보입니다.

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Downloading a file from Amazon S3.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. SSH를 사용하여 인스턴스에 로그인하고 `/tmp`의 내용을 표시합니다.

AWS OpsWorks Stacks Windows 인스턴스에서 SDK for Ruby 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.
>
> Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

이 주제에서는 AWS OpsWorks Stacks Windows 인스턴스AWS SDK for Ruby에서를 사용하여 S3 버킷에서 파일을 다운로드하는 방법을 설명합니다.

Ruby 애플리케이션이 AWS 리소스에 액세스해야 하는 경우 적절한 권한을 가진 AWS 자격 증명을 제공해야 합니다. 레시피의 경우 AWS 자격 증명을 제공하는 가장 좋은 방법은 AWS Identity and Access Management (IAM) 역할을 사용하는 것입니다. IAM 역할은 IAM 사용자처럼 작동하며 다양한 AWS 서비스를 사용할 수 있는 권한을 부여하는 정책이 연결되어 있습니다. 하지만 역할은 개인이 아니라 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 할당합니다. 그러면 이 인스턴스에서 실행되는 애플리케이션은 연결된 정책이 부여하는 권한을 획득할 수 있습니다. 역할을 사용하면 자격 증명이 간접적으로라도 코드에 표시되지 않습니다.

첫 번째 단계는 IAM 역할을 설정하는 것입니다. 이 예제에서는 가장 간단한 접근 방식을 취합니다. 즉, AWS OpsWorks Stacks가 첫 번째 스택을 생성할 때 생성하는 Amazon EC2 역할을 사용하는 것입니다. 이 역할은 `aws-opsworks-ec2-role`로 명명됩니다. 그러나 AWS OpsWorks Stacks는 정책을 해당 역할에 연결하지 않으므로 기본적으로 권한을 부여하지 않습니다.

적절한 권한을 부여하려면 `AmazonS3ReadOnlyAccess` 정책을 `aws-opsworks-ec2-role` 역할에 연결해야 합니다. 정책을 역할에 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

스택을 생성하거나 업데이트할 때 역할을 지정합니다. Windows 인스턴스에서 레시피 실행 섹션에 설명된 대로 사용자 지정 계층을 포함하는 스택을 설정하되 한 가지를 추가합니다. 스택 추가 페이지에서 기본 IAM 인스턴스 프로파일이 aws-opsworks-ec2-role로 설정되어 있는지 확인합니다. 그러면 AWS OpsWorks 스택이 스택의 모든 인스턴스에 해당 역할을 할당합니다.

쿡북을 설정하는 절차는 Linux 인스턴스에서 레시피 실행 섹션에서 사용한 것과 비슷합니다. 다음은 간략한 요약이며, 자세한 정보는 해당 예제를 참조하세요.

쿡북을 설정하려면

1. s3bucket_ops 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. 다음 내용이 포함된 metadata.rb 파일을 만들어 s3bucket_ops에 저장합니다.

```
name "s3download"
version "0.1.0"
```

3. recipes 안에 s3download 디렉터리를 만듭니다.
4. 다음 레시피가 포함된 default.rb 파일을 만들어 recipes 디렉터리에 저장합니다. *windows-cookbooks*를 다운로드할 파일을 저장하는 데 사용할 S3 버킷의 이름으로 바꿉니다.

```
Chef::Log.info("******Downloading an object from S3******")

chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    Aws.use_bundled_cert!

    s3_client = Aws::S3::Client.new(region:'us-west-2')

    s3_client.get_object(bucket: 'windows-cookbooks',
                         key: 'myfile.txt',
                         response_target: '/chef/myfile.txt')
  end
```

```
  action :run
end
```

5. s3download의 .zip 아카이브를 만들어 이 파일을 S3 버킷에 업로드합니다. 이 파일을 퍼블릭으로 설정하고 나중에 사용하기 위해 URL을 적어 둡니다.
6. myfile.txt라는 텍스트 파일을 만들어 이 파일을 S3 버킷에 업로드합니다. 이 파일은 레시피에서 다운로드할 파일이므로 편리한 버킷은 어느 것이든 사용할 수 있습니다.

이 레시피가 수행하는 작업은 다음과 같습니다.

1: SDK for Ruby v2 설치

이 예에서는 SDK for Ruby를 이용하여 객체를 다운로드합니다. 그러나 AWS OpsWorks Stacks는 Windows 인스턴스에이 SDK를 설치하지 않으므로 레시피의 첫 번째 부분은 chef_gem 리소스를 사용하여 해당 작업을 처리합니다. 사용자는 이 리소스를 사용하여 레시피를 포함하여 Chef가 사용하는 젬을 설치합니다.

2: 파일을 다운로드합니다.

레시피의 세 번째 부분은 ruby_block 리소스를 사용하여 *windows-cookbooks*라는 이름의 S3 버킷 에서 인스턴스의 /chef 디렉터리로 myfile.txt을 다운로드하는 SDK for Rub v2 코드를 실행합니다. *windows-cookbooks*를 myfile.txt가 저장된 버킷의 이름으로 변경합니다.

> **Note**
>
> 레시피는 Ruby 애플리케이션입니다. 따라서 Ruby 코드를 레시피 본문에 배치할 수 있습니다. 코드가 반드시 ruby_block 리소스에 위치할 필요는 없습니다. 하지만 Chef는 Ruby 코드를 레시피 본문부터 실행한 다음 각 리소스를 순서대로 실행합니다. 이 예제에서는 다운로드 코드를 레시피 본문에 배치할 경우 코드가 실패합니다. 이 코드는 SDK for Ruby에 의존하는데 SDK를 설치하는 chef_gem 리소스가 아직 실행되지 않았기 때문입니다. 리소스가 실행될 때 ruby_block 리소스 내 코드가 실행되며, 이는 chef_gem 리소스가 SDK for Ruby를 설치한 후에 이루어집니다.

다음과 같이 이 예제를 위한 스택을 생성합니다. 기존의 Windows 스택을 사용할 수 있습니다. 나중에 설명하듯이 쿡북을 업데이트하면 됩니다.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.

   - 이름 – S3Download
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – S3Download
   - 짧은 이름 - s3download
3. 기본 설정을 사용하여 S3Download 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

이제 레시피를 설치하고 실행할 수 있습니다.

레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.
2. 사용자 지정 쿡북 업데이트 스택 명령을 실행하여 스택의 온라인 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.
3. 실행할 레시피가 **`s3download::default`**으로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 `s3download::default`로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다.

> **Note**
>
> 일반적으로 AWS OpsWorks Stacks가 레시피를 적절한 수명 주기 이벤트에 할당하여 레시피를 자동으로 실행하도록 합니다. 수동으로 이벤트를 트리거하여 그러한 레시피를 실행할 수도 있습니다. 스택 명령을 사용하여 설정 및 Configure 이벤트를 트리거할 수 있고, 배포 명령을 사용하여 Deploy 및 Undeploy 이벤트를 트리거할 수 있습니다.

레시피가 성공적으로 실행되면 이를 확인할 수 있습니다.

s3download를 확인하려면

1. 가장 먼저 Chef 로그를 확인합니다. 스택에는 s3download1이라는 인스턴스가 하나 있어야 합니다. 인스턴스 페이지에서 인스턴스의 로그 열에서 표시를 선택하여 Chef 로그를 표시합니다. 아래로 스크롤하면 맨 아래 근처에 로그 메시지가 보입니다.

```
...
[2015-05-01T21:11:04+00:00] INFO: Loading cookbooks [s3download@0.0.0]
[2015-05-01T21:11:04+00:00] INFO: Storing updated cookbooks/s3download/recipes/
default.rb in the cache.
[2015-05-01T21:11:04+00:00] INFO: ******Downloading an object from S3******
[2015-05-01T21:11:04+00:00] INFO: Processing chef_gem[aws-sdk] action install
 (s3download::default line 3)
[2015-05-01T21:11:05+00:00] INFO: Processing ruby_block[download-object] action run
 (s3download::default line 8)
...
```

2. RDP를 사용하여 인스턴스에 로그인하고 `c:\chef`의 내용을 확인합니다.

Windows 소프트웨어 설치

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 다음 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.

Windows 인스턴스는 Windows Server 2012 R2 Standard를 사용하여 시작하므로, 대개 일부 소프트웨어를 설치해야 합니다. 세부 정보는 소프트웨어 유형에 따라 다릅니다.

- Windows 기능은 .NET 프레임워크 및 인터넷 정보 서비스(IIS)를 비롯한 선택적 시스템 구성 요소이며 인스턴스로 다운로드할 수 있습니다.
- 타사 소프트웨어는 일반적으로 MSI 파일과 같은 설치 프로그램 관리자로 제공됩니다. 이 패키지를 인스턴스로 다운로드하여 실행해야 합니다.

  일부 Microsoft 소프트웨어도 설치 프로그램 패키지로 제공됩니다.

이 섹션에서는 Windows 기능 및 패키지를 설치하는 쿡북을 구현하는 방법을 설명합니다. 또한 Windows 인스턴스용 레시피 구현을 간소화하는 리소스 및 헬퍼 기능을 포함하는 Chef windows 쿡북도 소개합니다.

주제

- Windows 기능 설치: IIS
- Windows 인스턴스에 패키지 설치

Windows 기능 설치: IIS

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Windows 기능은 .NET 프레임워크 및 인터넷 정보 서비스(IIS)를 비롯한 일련의 선택적 시스템 구성 요소입니다. 이 주제에서는 일반적으로 사용되는 기능인 인터넷 정보 서비스(IIS)를 설치하는 쿡북을 구현하는 방법을 설명합니다.

> **Note**
>
> 패키지 설치 섹션은 MSI 파일과 같은 설치 프로그램 관리자로 제공되는 소프트웨어를 설치하는 방법을 보여줍니다. 이러한 패키지는 인스턴스로 다운로드하여 실행해야 합니다. IIS 쿡북

Windows 인스턴스에서 레시피 실행 섹션은`powershell_script` 리소스를 사용하여 Windows 기능을 설치하는 방법을 보여줍니다. 이 예제는 Chef Windows 쿡북의 `windows_feature` 리소스를 사용하는 대안적 방법을 보여줍니다. 이 쿡북에는 Deployment Image Servicing and Management를 사용하여 기능 설치를 비롯한 다양한 Windows 작업을 수행하는 일련의 리소스가 포함되어 있습니다.

> **Note**
>
> Chef에도 IIS를 관리하는 데 사용할 수 있는 IIS 쿡북이 있습니다. 자세한 정보는 IIS cookbook 단원을 참조하세요.

쿡북을 설정하려면

1. windows cookbook GitHub 리포지토리로 이동한 다음 `windows` 쿡북을 다운로드합니다.

   이 예제에서는 `windows` 리포지토리를 .zip 파일로 다운로드한다고 가정하지만 원하는 경우 이 리포지토리를 복제할 수도 있습니다.

2. chef_handler cookbook GitHub 리포지토리로 이동한 다음 `chef-handler` 쿡북을 다운로드합니다.

   `windows` 쿡북은 `chef_handler`에 따라 달라지므로 이 쿡북을 직접 사용하지는 않습니다. 이 예제에서는 `chef_handler` 리포지토리를 .zip 파일로 다운로드한다고 가정하지만 원하는 경우 이 리포지토리를 복제할 수도 있습니다.

3. `windows` 및 `chef_handler` 쿡북을 쿡북 디렉터리 `windows` 및 `chef_handler`에 각각 추출합니다.
4. 쿡북 디렉터리 `install-iis` 안에 디렉터리를 만들고 그 디렉터리로 이동합니다.
5. `install-iis`에 다음 콘텐츠가 포함된 `metadata.rb` 파일을 추가합니다.

```
name "install-iis"
version "0.1.0"

depends "windows"
```

depends 명령을 사용하면 레시피에서 windows 쿡북 리소스를 사용할 수 있습니다.

6. recipes에 install-iis 디렉터리를 추가하고 다음 레시피 코드가 포함된 default.rb 파일을 이 디렉터리에 추가합니다.

```
%w{ IIS-WebServerRole IIS-WebServer }.each do |feature|
  windows_feature feature do
    action :install
  end
end

service 'w3svc' do
  action [:start, :enable]
end
```

레시피에서는 windows 쿡북의 windows_feature 리소스를 사용하여 다음을 설치합니다.

1. The IIS 웹 서버 역할.
2. The IIS 웹 서버.

그런 다음 레시피에서는 service 리소스를 사용하여 IIS 서비스(W3SVC)를 시작 및 활성화합니다.

> **Note**
>
> 사용 가능한 Windows 기능의 전체 목록을 확인하려면 RDP를 사용하여 인스턴스에 로그인해 명령 프롬프트 창을 열고 다음 명령을 실행합니다. 이 목록은 상당히 깁니다.
>
> ```
> dism /online /Get-Features
> ```

7. `install-iis`, `chef_handler` 및 `windows` 쿡북이 포함된 `.zip` 아카이브를 만들어 S3 버킷에 업로드합니다. 이 아카이브를 퍼블릭으로 설정하고 나중에 사용하기 위해 URL을 적어 둡니다. 이 예제에서는 이 아카이브의 이름을 `install-iis.zip`이라고 가정합니다. 자세한 내용은 쿡북 리포지토리 섹션을 참조하세요.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

다음과 같이 이 예제를 위한 스택을 생성합니다. 기존의 Windows 스택을 사용할 수도 있습니다. 나중에 설명하듯이 쿡북을 업데이트하면 됩니다.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.

   - 이름 – InstallIIS
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 - IIS
   - 짧은 이름 - iis
3. 기본 설정을 사용하여 IIS 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

이제 쿡북을 설치하고 레시피를 실행할 수 있습니다.

쿡북을 설치하고 레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL – 앞에서 기록해 둔 쿡북 아카이브 URL

기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.

2. [사용자 지정 쿡북 업데이트] 스택 명령을 실행하여 스택의 온라인 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.
3. 실행할 레시피가 **`install-iis::default`**으로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 지정한 레시피를 실행하는 Chef 실행을 시작합니다.

> **Note**
>
> 이 예제에서는 편의상 레시피 실행을 사용하지만 일반적으로 AWS OpsWorks Stacks가 레시피를 적절한 수명 주기 이벤트에 할당하여 레시피를 자동으로 실행하도록 합니다. 수동으로 이벤트를 트리거하여 그러한 레시피를 실행할 수 있습니다. 스택 명령을 사용하여 설정 및 Configure 이벤트를 트리거할 수 있고, 배포 명령을 사용하여 Deploy 및 Undeploy 이벤트를 트리거할 수 있습니다.

4. 설치를 확인하려면 RDP를 사용하여 인스턴스에 연결한 다음 Windows 탐색기를 엽니다. 이제 파일 시스템에 `C:\inetpub` 디렉터리가 있을 것입니다. 관리 도구 제어판 애플리케이션에서 서비스 목록을 확인하는 경우 IIS는 맨 아래 근처에 있습니다. 그러나 이름이 IIS가 아니라 World Wide Web Publishing Service로 표시됩니다.

Windows 인스턴스에 패키지 설치

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 예제에서는 사용자가 이미 Windows 인스턴스에서 레시피 실행 예제를 완료한 것으로 가정합니다. 그렇지 않은 경우 먼저 그 예제를 수행해야 합니다. 특히, 그 예제는 인스턴스에 대한 RDP 액세스를 활성화하는 방법을 설명합니다.

소프트웨어가 MSI와 같은 설치 프로그램 관리자로 제공될 경우 파일을 인스턴스로 다운로드하여 실행해야 합니다. 이 예제는 연결된 환경 변수를 정의하는 방법을 포함하여 MSI 패키지 형태의 Python 런타임을 설치하는 쿡북을 구현하는 방법을 보여줍니다. IIS와 같은 Windows 기능을 설치하는 방법에 대한 자세한 정보는 Windows 기능 설치: IIS 단원을 참조하세요.

쿡북을 설정하려면

1. `installpython` 하위 디렉터리를 만들고 그 디렉터리로 이동합니다.
2. `installpython`에 다음 콘텐츠가 포함된 `metadata.rb` 파일을 추가합니다.

```
name "installpython"
version "0.1.0"
```

3. `installpython`에 `recipes` 및 `files` 디렉터리를 추가하고 파일에 `default` 디렉터리를 추가합니다.
4. Windows용 Python 버전에서 쿡북의 `files\default` 디렉터리로 Python 패키지를 다운로드합니다. 이 예제에서는 `python-3.4.3.amd64.msi`라는 MSI 설치 관리자를 사용하는 Python 3.5.0a3의 Windows x86-64 버전을 설치합니다.
5. 다음 레시피 코드가 포함된 `default.rb` 파일을 `recipes` 디렉터리에 추가합니다.

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
  action :create
end

cookbook_file 'C:\tmp\python-3.4.3.amd64.msi' do
  source "python-3.4.3.amd64.msi"
  rights :full_control, 'Everyone'
  action :create
end

windows_package 'python' do
  source 'C:\tmp\python-3.4.3.amd64.msi'
  action :install
end

env "PATH" do
  value 'c:\python34'
```

```
  delim ";"
  action :modify
end
```

이 레시피는 다음 작업을 수행합니다.

1. 디렉터리 리소스를 사용하여 C:\tmp 디렉터리를 만듭니다.

   이 리소스에 대한 자세한 정보는 예제 3: 디렉터리 생성 단원을 참조하세요.

2. cookbook_file 리소스를 사용하여 쿡북의 files\default 디렉터리에서 C:\tmp로 설치 관리자를 복사합니다.

   이 리소스에 대한 자세한 정보는 쿡북에서 파일 설치 단원을 참조하세요.

3. windows_package 리소스를 사용하여 c:\python34에 Python을 설치하는 MSI 설치 관리자를 실행합니다.

   이 설치 관리자는 필수 디렉터리를 만들고 파일을 설치하지만 시스템의 PATH 환경 변수를 수정하지는 않습니다.

4. env 리소스를 사용하여 시스템 경로에 c:\python34를 추가합니다.

   env 리소스를 사용하여 환경 변수를 정의합니다. 이 경우, 레시피를 사용하면 경로에 c:\python34를 추가하여 명령줄에서 Python 스크립트를 쉽게 실행할 수 있습니다.

   - 리소스 이름은 이 예제에 필요한 환경 변수의 이름인 PATH를 지정합니다.
   - value 특성은 이 예제에 필요한 변수 값 c:\\python34를 지정합니다(\ 문자를 이스케이프해야 함).
   - :modify 작업은 변수의 현재 값 앞에 지정된 값을 추가합니다.
   - delim 속성은 새 값을 기존 값과 구분하는 구분 기호를 지정합니다. 이 예제에서는 ;입니다.

6. installpython의 .zip 아카이브를 만들고 그 아카이브를 S3 버킷에 업로드하고 퍼블릭으로 지정합니다. 나중에 사용하기 위해 아카이브의 URL를 적어 둡니다. 자세한 내용은 쿡북 리포지토리 섹션을 참조하세요.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

다음과 같이 이 예제를 위한 스택을 생성합니다. 기존의 Windows 스택을 사용할 수도 있습니다. 나중에 설명하듯이 쿡북을 업데이트하면 됩니다.

스택을 만듭니다

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다. 다음 설정을 지정하고, 그 외 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택합니다.

   - 이름 – InstallPython
   - 리전 – 미국 서부(오레곤)

     이 예제는 모든 리전에서 작동하지만 자습서의 경우 미국 서부(오레곤)를 사용하는 것이 좋습니다.
   - 기본 운영 체제 - Microsoft Windows Server 2012 R2
2. [계층 추가]를 선택하여 스택에 다음 설정으로 사용자 지정 계층을 추가합니다.

   - 이름 – Python
   - 짧은 이름 - python
3. 기본 설정을 사용하여 Python 계층에 24/7 인스턴스를 추가하고 해당 인스턴스를 시작합니다.

인스턴스가 온라인 상태로 전환되면 쿡북을 설치하고 레시피를 실행할 수 있습니다.

쿡북을 설치하고 레시피를 실행하려면

1. 스택을 편집해 사용자 지정 쿡북을 활성화하고 다음 설정을 지정합니다.

   - 리포지토리 유형 – S3 아카이브
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   기타 설정에 대해 기본값을 수락하고 [저장]을 선택하여 스택 구성을 업데이트합니다.
2. [사용자 지정 쿡북 업데이트] 스택 명령을 실행하여 스택의 온라인 인스턴스에 사용자 지정 쿡북의 최신 버전을 설치합니다. 쿡북의 이전 버전이 있으면 이 명령이 이전 버전을 덮어 씁니다.
3. 실행할 레시피가 **`installpython::default`**으로 설정된 상태에서 레시피 실행 스택 명령을 실행하여 레시피를 실행합니다. 이 명령은 `installpython::default`로 구성된 실행 목록을 사용하여 Chef 실행을 시작합니다.

> **Note**
>
> 이 예제에서는 편의상 레시피 실행을 사용하지만 일반적으로 AWS OpsWorks Stacks가 레시피를 적절한 수명 주기 이벤트에 할당하여 레시피를 자동으로 실행하도록 합니다. 수

동으로 이벤트를 트리거하여 그러한 레시피를 실행할 수 있습니다. 스택 명령을 사용하여 설정 및 Configure 이벤트를 트리거할 수 있고, 배포 명령을 사용하여 Deploy 및 Undeploy 이벤트를 트리거할 수 있습니다.

4. 설치를 확인하려면 RDP를 사용하여 인스턴스에 연결한 다음 Windows 탐색기를 엽니다.

   - 이제 파일 시스템에 `C:\Python34` 디렉터리가 있을 것입니다.
   - 명령줄에서 `path`를 실행하면 그 결과는 `PATH=c:\python34;C:\Windows\system32;...`와 유사해야 합니다.
   - 명령줄에서 `python --version`을 실행하면 `Python 3.4.3`을 반환해야 합니다.

내장 속성 재정의

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 주제는 Linux 스택에만 적용됩니다. Windows 스택에서는 내장 속성을 재정의할 수 없습니다.

AWS OpsWorks Stacks는 각 인스턴스에 내장 쿡북 세트를 설치합니다. 내장 쿡북은 대부분 내장 계층을 지원하며, 속성 파일은 Apache 서버 구성 설정과 같은 다양한 기본 시스템 및 애플리케이션 설정을 정의합니다. 속성 파일에 이러한 설정을 포함시키면 다음 방법 중 하나를 사용하여 해당되는 내장 속성을 재정의함으로써 많은 구성 설정을 사용자 지정할 수 있습니다.

- 속성을 사용자 지정 JSON으로 정의합니다.

  이 방법은 간단하고 유연하다는 장점이 있습니다. 하지만 속성 정의를 관리할 확실한 방법이 없으므로 수동으로 사용자 지정 JSON을 입력해야 합니다.
- 사용자 지정 쿡북을 구현하고 속성을 `customize.rb` 속성 파일에 정의합니다.

이 방법은 사용자 지정 JSON을 사용하는 것보다 덜 유연하지만, 사용자 지정 쿡북을 소스 제어에 포함시킬 수 있으므로 보다 확실합니다.

이 주제는 사용자 지정 쿡북 속성 파일을 사용하여 내장 속성을 재정의하는 방법을 Apache 서버를 예로 하여 설명합니다. 사용자 지정 JSON을 사용하여 속성을 재정의하는 방법에 대한 자세한 정보는 사용자 지정 JSON 사용 단원을 참조하세요. 속성 재정의 대한 일반적 설명은 속성 재정의 단원을 참조하세요.

> **Note**
>
> 속성 재정의는 구성 설정을 사용자 지정하는 데 선호되는 방법이지만, 설정이 항상 속성으로 표현되는 것은 아닙니다. 이런 경우 종종 내장 레시피가 구성 파일을 생성하는 데 사용하는 템플릿을 재정의하여 구성 파일을 사용자 지정할 수 있습니다. 예제는 내장 템플릿 재정의 섹션을 참조하세요.

일반적으로 내장 속성은 설정 레시피가 구성 파일을 생성하는 데 사용하는 템플릿 파일의 값을 나타냅니다. 예를 들어 apache2 설정 레시피 중 하나인 `default.rb`는 `apache2.conf.erb` 템플릿을 사용하여 Apache 서버의 주 구성 파일 `httpd.conf`(Amazon Linux) 또는 `apache2.conf`(Ubuntu)를 생성합니다. 다음은 템플릿 파일에서 발췌한 코드입니다.

```
...
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
##
## Server-Pool Size Regulation (MPM specific)
##

...
```

이 예제의 KeepAliveTimeout 설정은 [:apache][:keepalivetimeout] 속성의 값입니다. 다음 코드에 나오듯이 이 속성의 기본값은 apache2 cookbook's apache.rb 속성 파일에서 정의됩니다.

```
...
# General settings
default[:apache][:listen_ports] = [ '80','443' ]
default[:apache][:contact] = 'ops@example.com'
default[:apache][:log_level] = 'info'
default[:apache][:timeout] = 120
default[:apache][:keepalive] = 'Off'
default[:apache][:keepaliverequests] = 100
default[:apache][:keepalivetimeout] = 3
...
```

> **Note**
>
> 일반적으로 사용되는 내장 속성에 대한 자세한 정보는 내장 쿡북 속성 단원을 참조하세요.

내장 속성 재정의를 지원하기 위해 모든 내장 쿡북은 customize.rb 속성 파일을 포함합니다. 이 파일은 include_attribute 명령을 통해 모든 모듈에 통합됩니다. 내장 쿡북의 customize.rb 파일은 속성 정의가 없으므로 내장 속성에 아무 영향을 미치지 않습니다. 내장 속성을 재정의하려면 내장 쿡북과 동일한 이름을 사용하여 사용자 지정 쿡북을 생성하고 사용자 지정 속성 정의를 마찬가지로 customize.rb로 명명된 속성 파일에 저장합니다. 이 파일은 내장 버전보다 우선하며 관련된 모든 모듈에 포함됩니다. customize.rb에서 내장 속성을 정의할 경우 이들이 해당되는 내장 속성을 재정의합니다.

이 예제는 내장 [:apache][:keepalivetimeout] 속성을 재정의하여 값을 3 대신 5로 설정하는 방법을 보여줍니다. 모든 내장 속성에서 비슷한 방법을 사용할 수 있습니다. 하지만 어느 속성을 재정의할지는 신중해야 선택해야 합니다. 예를 들어 opsworks 네임스페이스에서 속성을 재정의하면 일부 내장 레시피에 문제가 생길 수 있습니다.

> **Important**
>
> 내장 속성 파일의 사본 자체를 수정하여 내장 속성을 재정의하지 마십시오. 예를 들어 사용자 지정 쿡북의 폴더에 의 사본을 저장하고 일부 설정을 수정할 수 apache.rb있습니다apache2/attributes. 하지만 이 파일은 내장 버전에 우선하므로 이제부터 내장 레시피가 사용자 지정 버전의 apache.rb를 사용합니다. 나중에 AWS OpsWorks Stacks가 기본 제

공 apache.rb 파일을 수정하는 경우 버전을 수동으로 업데이트하지 않는 한 레시피는 새 값을 가져오지 않습니다. customize.rb를 사용하면 지정된 속성만 재정의되고 내장 레시피는 재정의되지 않은 모든 속성에 대해 계속해서 최신 값을 자동으로 가져옵니다.

시작하려면 사용자 지정 쿡북을 생성합니다.

쿡북을 생성하려면

1. opsworks_cookbooks 디렉터리 안에 쿡북 디렉터리 apache2를 만들고 그 디렉터리로 이동합니다.

   내장 속성을 재정의하려면 사용자 지정 쿡북의 이름이 내장 쿡북과 동일해야 합니다(이 예제에서는 apache2).
2. apache2 디렉터리 내에 attributes 디렉터리를 만듭니다.
3. customize.rb 파일을 attributes 디렉터리에 추가하고 이 파일을 사용하여 재정의하려는 내장 쿡북 속성을 정의합니다. 이 예제의 경우 파일에 다음 내용이 포함되어 있어야 합니다.

   ```
   normal[:apache][:keepalivetimeout] = 5
   ```

   Important

   내장 속성을 재정의하려면 사용자 지정 속성이 normal 유형 이상이어야 하며 해당하는 내장 속성과 노드 이름이 정확하게 같아야 합니다. normal 유형은 사용자 지정 속성이 모두 default 유형인 내장 속성보다 우선하도록 합니다. 자세한 내용은 속성 우선 순위 섹션을 참조하세요.
4. opsworks_cookbooks.zip 이름이 지정된 opsworks_cookbooks의 .zip 아카이브를 생성하고 아카이브를 Amazon Simple Storage Service(S3) 버킷에 업로드합니다. 간단하게 설명하기 위해 이 파일을 퍼블릭으로 설정합니다. 나중에 사용하기 위해 이 URL을 적어 둡니다. 프라이빗 Amazon S3 아카이브 또는 기타 리포지토리 유형에 쿡북을 저장할 수도 있습니다. 자세한 내용은 쿡북 리포지토리 섹션을 참조하세요.

   Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

사용자 지정 속성을 사용하려면 스택을 생성하고 쿡북을 설치합니다.

사용자 지정 속성을 사용하려면

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다.
2. 다음 표준 설정을 지정합니다.

   - 이름 - ApacheConfig
   - 리전 – 미국 서부(오레곤)

     모든 리전에서 스택을 저장할 수 있지만 자습서의 경우 미국 서부(오레곤)를 선택하는 것이 좋습니다.
   - 기본 SSH 키 - EC2 키 페어

     EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 스택과 동일한 AWS 리전에 속해야 합니다.

   [고급>>]을 선택하고 [사용자 지정 Chef 쿡북 사용]을 [예]로 설정한 후 다음 설정을 지정합니다.

   - 리포지토리 유형 – Http Archive
   - 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

   다른 설정에 대해서는 기본값을 수락한 다음 [스택 추가]를 선택해 스택을 생성합니다.

   > **Note**
   >
   > 이 예제에서는 기본 운영 체제인 Amazon Linux를 사용하지만, 원한다면 Ubuntu를 사용해도 됩니다. Ubuntu 시스템에서는 내장 설정 레시피가 설정이 동일한 구성 파일 `apache2.conf`를 생성해 `/etc/apache2` 디렉터리에 저장한다는 점만 다릅니다.
3. 계층 추가를 선택한 다음 스택에 기본 설정으로 Java 앱 서버 계층을 추가합니다.
4. 기본 설정을 사용하여 계층에 24/7 인스턴스를 추가한 다음 해당 인스턴스를 시작합니다.

   이 예제에는 t2.micro 인스턴스면 충분합니다.
5. 인스턴스가 온라인 상태가 되면 SSH를 사용하여 이 인스턴스에 연결합니다. `httpd.conf` 파일은 `/etc/httpd/conf` 디렉터리에 있습니다. 이 파일을 확인하는 경우 사용자 지정

KeepAliveTimeout 설정이 보여야 합니다. 나머지 설정은 내장 apache.rb 파일의 기본값을 갖습니다. httpd.conf의 관련 부분에 다음과 비슷한 내용이 표시됩니다.

```
...
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5
...
```

내장 템플릿 재정의

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제는 Linux 스택에만 적용됩니다. Windows 스택에서는 템플릿을 재정의할 수 없습니다.

AWS OpsWorks Stacks 내장 레시피는 템플릿을 사용하여 인스턴스에 파일을 생성하고, 주로 Apache와 같은 서버에 대한 구성 파일을 생성합니다. 예를 들어 apache2 레시피는 apache2.conf.erb 템플릿을 사용하여 Apache 서버의 주 구성 파일 httpd.conf(Amazon Linux) 또는 apache2.conf(Ubuntu)를 생성합니다.

이러한 설정에서 구성 설정은 대부분은 속성으로 표현되므로 해당 내장 속성을 재정하여 구성 파일을 사용자 지정하는 것이 선호되는 방법입니다. 예제는 내장 속성 재정의 섹션을 참조하세요. 하지만 사용자 지정할 설정이 내장 속성으로 표현되지 않거나 템플릿에 포함되지 않은 경우 템플릿 자체를 재정의해야 합니다. 이 주제에서는 내장 템플릿을 재정의하여 사용자 지정 Apache 구성 설정을 지정하는 방법을 설명합니다.

ErrorDocument 설정을 httpd.conf 파일에 추가하여 Apache에 사용자 지정 오류 응답을 제공할 수 있습니다. 다음 코드에서 보듯이 apache2.conf.erb는 일부 코멘트 아웃된 예제만 포함하고 있습니다.

```
...
#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

이러한 설정은 하드코딩된 주석이므로 속성을 재정의하여 사용자 지정 값을 지정할 수 없습니다. 템플릿 자체를 재정의해야 합니다. 하지만 속성과 달리 템플릿 파일의 특정 부분을 재정의할 수는 없습니다. 내장 버전과 동일한 이름으로 사용자 지정 쿡북을 생성하고, 템플릿 파일을 동일한 하위 디렉터리에 복사하고, 필요에 따라 파일을 수정해야 합니다. 이 주제는 apache2.conf.erb를 재정의하여 오류 500에 대한 사용자 지정 응답을 제공하는 방법을 설명합니다. 템플릿 재정의 대한 일반적 설명은 사용자 지정 템플릿 사용 단원을 참조하세요.

> **Important**
>
> 내장 템플릿을 재정의하면 내장 레시피가 내장 버전 대신 사용자 지정 버전의 템플릿을 사용합니다. AWS OpsWorks Stacks가 기본 제공 템플릿을 업데이트하면 사용자 지정 템플릿이 동기화되지 않아 올바르게 작동하지 않을 수 있습니다. AWS OpsWorks Stacks는 이러한 변경을 자주 수행하지 않으며 템플릿이 변경되면 AWS OpsWorks Stacks는 변경 사항을 나열하고 새 버전으로 업그레이드할 수 있는 옵션을 제공합니다. AWS OpsWorks Stacks 리포지토리에서 변경 사항을 모니터링하고 필요에 따라 사용자 지정 템플릿을 수동으로 업데이트하는 것이 좋습니다. 리포지토리는 지원되는 Chef 버전마다 별도의 브랜치가 있습니다. 따라서 올바른 브랜치에 위치하는지 확인해야 합니다.

시작하려면 사용자 지정 쿡북을 생성합니다.

쿡북을 생성하려면

1. opsworks_cookbooks 디렉터리 안에 쿡북 디렉터리 apache2를 만들고 그 디렉터리로 이동합니다. 내장 템플릿을 재정의하려면 사용자 지정 쿡북의 이름이 내장 쿡북과 동일해야 합니다(이 경우에는 apache2).

   > **Note**
   >
   > 내장 속성 재정의 연습을 이미 완료한 경우 이 예제에 동일한 apache2 쿡북을 사용해 2단계를 건너뛸 수 있습니다.

2. 다음 내용이 포함된 metadata.rb 파일을 만들어 apache2 디렉터리에 저장합니다.

   ```
   name "apache2"
   version "0.1.0"
   ```

3. apache2 디렉터리에서 templates/default 디렉터리를 만듭니다.

   > **Note**
   >
   > templates/default 디렉터리는 기본 apache2.conf.erb 템플릿을 사용하는 Amazon Linux 인스턴스에 대해 작동합니다. Ubuntu 14.04 인스턴스는 apache2.conf.erb 디렉터리에 있는 운영 체제별 templates/ubuntu-14.04 템플릿을 사용합니다. Ubuntu 14.04 인스턴스에도 사용자 지정 사항을 적용하려면 해당 템플릿도 재정의해야 합니다.

4. 내장 apache2.conf.erb 템플릿을 templates/default 디렉터리에 복사합니다. 템플릿 파일을 열고 ErrorDocument 500 행의 주석 처리를 해제하고 다음과 같이 사용자 지정 오류 메시지를 입력합니다.

   ```
   ...
   ErrorDocument 500 "A custom error message."
   #ErrorDocument 404 /missing.html
   ...
   ```

5. opsworks_cookbooks의 .zip 아카이브를 만들어 opsworks_cookbooks.zip으로 이름을 지정한 다음 해당 파일을 Amazon Simple Storage Service(S3) 버킷에 업로드합니다. 간단하게 설명하기 위해 이 아카이브를 퍼블릭으로 설정합니다. 나중에 사용하기 위해 아카이브의 URL를 적

어 둡니다. 프라이빗 Amazon S3 아카이브 또는 기타 리포지토리 유형에 쿡북을 저장할 수도 있습니다. 자세한 내용은 쿡북 리포지토리 섹션을 참조하세요.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

> **Note**
>
> 간결한 설명을 위해 이 예제에서는 템플릿에 하드코딩된 오류 메시지를 추가합니다. 이를 변경하려면 템플릿을 수정하고 쿡북을 재설치해야 합니다. 유연성을 높이기 위해 사용자 지정 쿡북의 `customize.rb` 속성 파일에서 오류 문자열의 기본 사용자 지정 속성을 정의하고 해당 속성의 값을 `ErrorDocument 500`에 할당할 수 있습니다. 예를 들어 속성을 `[:apache][:custom][:error500]`으로 명명할 경우 `apache2.conf.erb`에서 해당 줄이 다음과 같을 것입니다.
>
> ```
> ...
> ErrorDocument 500 <%= node[:apache][:custom][:error500] %>
> #ErrorDocument 404 /missing.html
> ...
> ```
>
> 그런 다음 언제라도 `[:apache][:custom][:error500]`을 재정의하여 사용자 지정 오류 메시지를 변경할 수 있습니다. 사용자 지정 JSON을 사용하여 속성을 재정의하는 경우 쿡북은 손댈 필요조차 없습니다.

사용자 지정 템플릿을 사용하려면 스택을 생성하고 쿡북을 설치합니다.

사용자 지정 템플릿을 사용하려면

1. AWS OpsWorks Stacks 콘솔을 열고 스택 추가를 선택합니다.
2. 다음 표준 설정을 지정합니다.
   - 이름 – ApacheTemplate
   - 리전 – 미국 서부(오레곤)
   - 기본 SSH 키 - Amazon Elastic Compute Cloud(Amazon EC2) 키 페어

Amazon EC2 키 페어를 생성해야 하는 경우 Amazon EC2 키 페어를 참조하세요. 키 페어는 인스턴스와 동일한 AWS 리전에 속해야 합니다.

[고급>>]를 선택한 다음 [사용자 지정 Chef 쿡북 사용]를 선택하여 다음 설정을 지정합니다.

- 리포지토리 유형 – Http Archive
- 리포지토리 URL - 앞에서 기록해 둔 쿡북 아카이브 URL

기타 설정에 대해 기본값을 수락하고 [스택 추가]를 선택하여 스택을 생성합니다.

3. 계층 추가를 선택한 다음 스택에 기본 설정으로 Java 앱 서버 계층을 추가합니다.
4. 기본 설정을 사용하여 계층에 24/7 인스턴스를 추가한 다음 해당 인스턴스를 시작합니다.

   이 예제에는 t2.micro 인스턴스면 충분합니다.
5. 인스턴스가 온라인 상태가 되면 SSH를 사용하여 이 인스턴스에 연결합니다. `httpd.conf` 파일은 `/etc/httpd/conf` 디렉터리에 있습니다. 이 파일에는 다음과 유사한 사용자 지정 `ErrorDocument` 설정이 포함되어 있어야 합니다.

```
...
# Some examples:
ErrorDocument 500 "A custom error message."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

계층 로드 밸런싱

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 두 가지 로드 밸런싱 옵션인 Elastic Load Balancing과 HAProxy를 제공합니다.이 옵션은 일반적으로 애플리케이션 서버 계층의 인스턴스에서 로드 밸런싱에 사용됩니다. 이 주제에서는 계층에 로드 밸런싱을 추가할 때 어떤 옵션을 선택할지 결정하는 데 도움이 되도록 각 옵션의 이점과 제약을 설명합니다. 일부의 경우 두 옵션을 모두 사용하는 것이 최선의 방법입니다.

SSL 종료

내장 HAProxy 계층은 SSL 종료를 처리하지 않습니다. 따라서 서버에서 SSL을 종료해야 합니다. 이 접근 방식의 장점은 트래픽이 서버에 도달할 때까지 암호화된다는 것입니다. 하지만 서버가 암호 해독을 처리해야 하므로 서버 로드가 증가합니다. 또한 애플리케이션 서버에 SSL 인증서를 배치해야 하는데, 애플리케이션 서버는 사용자가 액세스하기 더 쉽습니다.

Elastic Load Balancing을 사용하면 로드 밸런서에서 SSL을 종료할 수 있습니다. 이렇게 하면 애플리케이션 서버의 부하가 줄어들지만 로드 밸런서와 서버 간의 트래픽은 암호화되지 않습니다. Elastic Load Balancing을 사용하면 서버에서 SSL을 종료할 수도 있지만 설정하기가 다소 복잡합니다.

스케일링

수신 트래픽이 HAProxy 로드 밸런서의 용량을 초과할 경우 수동으로 용량을 증가해야 합니다.

Elastic Load Balancing은 수신 트래픽을 처리하기 위해 자동으로 확장됩니다. Elastic Load Balancing 로드 밸런서가 처음 온라인으로 전환될 때 예상 로드를 처리하는 데 충분한 용량을 갖추게 하려면 로드 밸런서를 사전 워밍할 수 있습니다.

로드 밸런서 장애

HAProxy 서버를 호스팅하는 인스턴스에 장애가 발생할 경우 인스턴스를 재시작할 수 있을 때까지 전체 사이트가 오프라인이 될 수 있습니다.

Elastic Load Balancing은 HAProxy보다 장애 방지 기능이 더 강합니다. 예를 들어, EC2 인스턴스가 등록된 각 가용 영역에 로드 밸런싱 노드를 프로비저닝합니다. 한 영역에서 서비스가 중단되면 다른 노드가 계속해서 수신 트래픽을 처리합니다. 자세한 내용은 Elastic Load Balancing Concepts를 참조하세요.

유휴 제한 시간

두 유형의 로드 밸런서는 모두 서버가 지정된 제한 시간 값보다 오래 유휴 상태를 유지할 경우 연결을 종료합니다.

- HAProxy - 유휴 제한 시간 값에 상한이 없습니다.
- Elastic Load Balancing – 기본 유휴 제한 시간 값이 60초이며 최대값은 3,600초(60분)입니다.

대부분의 경우 Elastic Load Balancing 유휴 제한 시간은 충분합니다. 이보다 긴 유휴 제한 시간이 필요할 경우 HAProxy를 사용하는 것이 좋습니다. 예시:

- 푸시 알림에 사용되는 장시간 HTTP 연결.
- 60분 이상 소요되는 작업을 수행하기 위해 사용하는 관리 인터페이스.

URL 기반 매핑

로드 밸런서가 수신 요청을 요청의 URL에 따라 특정 서버로 전달해야 할 수 있습니다. 예를 들어 온라인 상거래 애플리케이션을 지원하는 애플리케이션 서버 10개의 그룹이 있다고 가정합시다. 서버 중 8개는 카탈로그를 처리하고 2개는 결제를 처리합니다. 요청 URL을 기반으로 모든 결제 관련 HTTP 요청을 결제 서버로 보내려고 합니다. 이 경우, "payment" 또는 "checkout"을 포함하는 모든 URL을 결제 서버 중 하나로 리디렉션합니다.

HAProxy에서는 URL 기반 매핑을 사용하여 지정된 문자열을 포함하는 URL을 특정 서버로 리디렉션할 수 있습니다. AWS OpsWorks Stacks에서 URL 기반 매핑을 사용하려면 `haproxy` 기본 제공 쿡북의 `haproxy-default.erb` 템플릿을 재정의하여 사용자 지정 HAProxy 구성 파일을 생성해야 합니다. 자세한 정보는 HAProxy Configuration Manual 및 사용자 지정 템플릿 사용 단원을 참조하세요. HTTPS 요청에는 URL 기반 매핑을 사용할 수 없습니다. HTTPS 요청은 암호화됩니다. 따라서 HAProxy가 요청 URL을 검사할 방법이 없습니다.

Elastic Load Balancing은 URL 매핑을 제한적으로 지원합니다. 자세한 내용은 Elastic Load Balancing의 리스너 구성을 참조하세요.

권장 사항: HAProxy에서만 처리할 수 있는 요구 사항이 아니라면 로드 밸런싱에 Elastic Load Balancing을 사용할 것을 권장합니다. 이 경우 가장 좋은 방법은 Elastic Load Balancing을 HAProxy 서버 집합으로 수신 트래픽을 분산하는 프런트 엔드 로드 밸런서로 사용하여 두 유형을 결합하는 것이 최선의 방법일 수 있습니다. 방법:

- 스택의 각 가용 영역에서 HAProxy 인스턴스를 설정하여 요청을 영역의 애플리케이션 서버로 분산합니다.
- HAProxy 인스턴스를 Elastic Load Balancing 로드 밸런서에 할당합니다. 그러면 수신 요청이 HAProxy 로드 밸런서로 분산됩니다.

이 접근 방식은 HAProxy의 URL 기반 매핑을 사용하여 다양한 유형의 요청을 적절한 애플리케이션 서버로 분산할 수 있습니다. 하지만 HAProxy 서버 중 하나가 오프라인으로 전환될 경우 Elastic Load Balancing 로드 밸런서가 자동으로 수신 트래픽을 정상 상태의 HAProxy 서버로 분산하므로 사이트는 기능을 유지합니다. 단, Elastic Load Balancing을를 프런트 엔드 로드 밸런서로 사용해야 합니다. HAProxy 서버는 요청을 다른 HAProxy 서버로 분산할 수 없습니다.

Chef Server에서 AWS OpsWorks Stacks로 마이그레이션

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 Chef를 기반으로 하기 때문에 Chef Server에서 AWS OpsWorks Stacks로 마이그레이션하는 것은 비교적 간단합니다. 이 주제에서는 AWS OpsWorks Stacks와 연동되도록 Chef Server 코드를 수정하는 지침을 제공합니다.

> **Note**
>
> chef-solo를 기반으로 하고 검색 또는 데이터 백을 지원하지 않는 11.10 이전의 Chef 버전을 사용하는 스택으로 마이그레이션하는 것은 권장하지 않습니다.

주제

- 계층에 역할 매핑
- 데이터 백 사용
- Chef 검색 사용
- 쿡북 및 레시피 관리
- Chef 환경 사용

계층에 역할 매핑

Chef Server는 각각 Java 애플리케이션 서버를 호스팅하는 인스턴스 집합처럼 용도와 구성이 동일한 인스턴스를 나타내고 관리하는 데 역할을 사용합니다. AWS OpsWorks Stacks 계층은 기본적으로 Chef 역할과 같은 용도로 사용됩니다. 계층이란 구성, 설치된 패키지, 애플리케이션 배포 절차 등이 동일한 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 집합을 생성하기 위한 청사진입니다.

AWS OpsWorks Stacks에는 여러 유형의 애플리케이션 서버, HAProxy 로드 밸런서, MySQL 데이터베이스 마스터 및 Ganglia 모니터링 마스터를 위한 기본 제공 계층 세트가 포함되어 있습니다. 예를 들어 내장 Java 앱 서버 계층은 Tomcat 서버를 호스팅하는 인스턴스를 생성하기 위한 청사진입니다.

AWS OpsWorks Stacks로 마이그레이션하려면 각 역할을 동등한 기능을 제공하는 계층과 연결해야 합니다. 일부 역할은 내장 계층 중 하나를 그냥 사용할 수 있습니다. 일부 역할에는 다양한 정도의 사용자 지정이 필요할 수 있습니다. 연결된 레시피 등 내장 계층의 기능부터 검사하여 내장 계층이 최소한 역할의 기능 일부를 제공하는지 확인하세요. 내장 계층에 대한 자세한 정보는 계층 및 AWS OpsWorks Stacks 계층 참조 단원을 참조하세요. 내장 레시피를 검사하려면 AWS OpsWorks Stacks 퍼블릭 GitHub 리포지토리를 참조하세요.

이후의 진행은 다음과 같이 계층을 각 역할에 얼마나 가깝게 일치시킬 수 있는지에 따라 달라집니다.

내장 계층이 역할의 모든 기능을 지원

이 내장 계층을 직접 사용하거나 필요하다면 약간의 사용자 지정을 거쳐 사용할 수 있습니다. 예를 들어 역할이 Tomcat 서버를 지원하는 경우, Java 앱 서버 계층의 레시피는 이미 역할의 모든 작업과 어쩌면 일부 약간의 사용자 지정까지 처리할 수 있습니다. 예컨대 적절한 속성 또는 템플릿을 재정의하여 계층의 내장 레시피가 사용자 지정 Tomcat 또는 Apache 구성 설정을 사용하도록 할 수 있습니다.

내장 계층이 역할의 일부 기능을 지원하지만 모든 기능을 지원하지는 않음

계층을 확장하여 내장 계층을 사용할 수 있습니다. 그러려면 일반적으로 사용자 지정 레시피를 구현하여 빠진 기능을 지원하고 레시피를 계층의 수명 주기 이벤트에 할당해야 합니다. 예를 들어 역할이 Tomcat 서버를 호스팅하는 동일한 인스턴스에 Redis 서버를 설치한다고 가정해 보십시오. 이 경우, 사용자 지정 레시피를 구현하여 Redis를 Java 앱 서버 계층의 인스턴스에 설치하고 계층의 설정 이벤트에 레시피를 할당하면 역할의 기능에 일치하도록 계층을 확장할 수 있습니다.

역할의 모든 기능을 적절히 지원하는 내장 계층이 없음

사용자 지정 계층을 구현합니다. 예를 들어 어느 내장 계층도 지원하지 않는 MongoDB 데이터베이스 서버를 역할이 지원한다고 가정해 보십시오. 이러한 지원은 레시피를 구현하여 필요한 패키지를 설치하고 서버를 구성한 다음 레시피를 사용자 지정 계층의 수명 주기 이벤트에 할당함으로써 제공할 수 있습니다. 일반적으로 적어도 역할의 레시피 일부는 이러한 용도에 사용할 수 있습니다. 사용자 지정 계층을 구현하는 방법에 대한 자세한 정보는 사용자 지정 Tomcat 서버 계층 생성 단원을 참조하세요.

데이터 백 사용

Chef Server를 통해 데이터 백을 사용하여 사용자 정의 데이터를 레시피에 전달할 수 있습니다.

- 쿡북을 사용하여 데이터를 저장하면 Chef가 각 인스턴스에 데이터를 설치합니다.
- 비밀번호와 같은 민감한 데이터에는 암호화된 데이터 백을 사용할 수 있습니다.

AWS OpsWorks Stacks는 데이터 백을 지원합니다. 레시피는 Chef Server와 정확히 동일한 코드를 사용하여 데이터를 검색할 수 있습니다. 다만 이 지원에는 다음과 같은 제한과 차이점이 있습니다.

- 데이터 백은 Chef 11.10 Linux 및 이후의 스택에서만 지원됩니다.

  이보다 이전 버전의 Chef에서 실행되는 Windows 스택과 Linux 스택은 데이터 백을 지원하지 않습니다.
- 데이터 백을 쿡북 리포지토리에 저장하지 마십시오.

  그 대신 사용자 지정 JSON을 사용하여 데이터 백의 데이터를 관리하세요.
- AWS OpsWorks Stacks는 암호화된 데이터 백을 지원하지 않습니다.

  암호나 인증서 등 암호화된 형식의 민감한 데이터를 저장해야 하는 경우, 프라이빗 S3 버킷에 저장하는 것이 좋습니다. 그런 다음 Ruby용 Amazon SDK를 사용하는 사용자 지정 레시피를 생성해 데이터를 검색할 수 있습니다. 예제는 SDK for Ruby 사용 섹션을 참조하세요.

자세한 내용은 데이터 백 사용 섹션을 참조하세요.

Chef 검색 사용

Chef Server는 IP 주소와 역할 구성 같은 스택 구성 정보를 서버에 저장합니다. 레시피는 Chef 검색을 사용하여이 데이터를 검색합니다. AWS OpsWorks Stacks는 약간 다른 접근 방식을 사용합니다. 예를 들어 Chef 11.10 Linux 스택은 Chef Server의 경량 버전(흔히 Chef Zero라고 함)을 인스턴스에서 로컬로 실행하는 Chef 클라이언트 옵션인 Chef 클라이언트 로컬 모드에 기반합니다. Chef Zero는 인스턴스의 노드 객체에 저장된 데이터에 대한 검색을 지원합니다.

스택 데이터를 원격 서버에 저장하는 대신 AWS OpsWorks Stacks는 모든 수명 주기 이벤트에 대해 각 인스턴스의 노드 객체에 스택 구성 및 배포 속성 세트를 추가합니다. 이 속성들은 스택 구성의 스냅샷을 나타냅니다. 이 속성들은 Chef Server와 동일한 구문을 사용하며, 레시피가 서버에서 검색해야 하는 대부분의 데이터를 나타냅니다.

AWS OpsWorks Stacks에 대한 레시피의 검색 종속 코드를 수정할 필요가 없는 경우가 많습니다. Chef 검색은 스택 구성 및 배포 속성을 포함하는 노드 객체에서 작동하므로 AWS OpsWorks Stacks의 검색 쿼리는 일반적으로 Chef Server와 동일하게 작동합니다.

기본 예외는 스택 구성 및 배포 속성에 AWS OpsWorks Stacks가 인스턴스에 속성을 설치할 때 인식하는 데이터만 포함되어 있기 때문입니다. 특정 인스턴스에서 로컬로 속성을 생성하거나 수정하는 경우 이러한 변경 사항은 AWS OpsWorks Stacks로 다시 전파되지 않으며 다른 인스턴스에 설치된 스택 구

성 및 배포 속성에 통합되지 않습니다. 해당 인스턴스에서만 검색을 사용하여 속성을 검색할 수 있습니다. 자세한 내용은 Chef 검색 사용 단원을 참조하십시오.

Chef Server와의 호환성을 위해 AWS OpsWorks Stacks는 노드 객체에 `role` 속성 세트를 추가합니다. 각 객체에는 스택의 계층 속성 중 하나가 포함됩니다. 레시피가 `roles`를 검색 키로 사용하는 경우, 검색 코드를 변경할 필요가 없습니다. 쿼리는 자동으로 해당 계층에 대한 데이터를 반환합니다. 예를 들어 다음의 두 쿼리는 모두 php-app 계층의 속성을 반환합니다.

```
phpserver = search(:node, "layers:php-app").first
```

```
phpserver = search(:node, "roles:php-app").first
```

쿡북 및 레시피 관리

AWS OpsWorks Stacks와 Chef Server는 쿡북과 레시피를 약간 다르게 처리합니다. Chef Server의 경우:

- 직접 구현하거나 커뮤니티 쿡북을 사용하여 모든 쿡북을 제공합니다.
- 쿡북을 서버에 저장합니다.
- 수동으로 또는 정기적으로 레시피를 실행합니다.

AWS OpsWorks Stacks 사용:

- AWS OpsWorks Stacks는 각 내장 계층에 대해 하나 이상의 쿡북을 제공합니다. 이러한 쿡북은 내장 계층의 소프트웨어 설치 및 구성과 앱 배포 같은 표준 작업을 처리합니다.

  내장 쿡북이 수행하지 않는 작업을 처리하려면 스택에 사용자 지정 쿡북을 추가하거나 커뮤니티 쿡북을 사용해야 합니다.
- Stacks AWS OpsWorks 쿡북을 S3 버킷 또는 Git 리포지토리와 같은 원격 리포지토리에 저장합니다.

  자세한 내용은 쿡북 저장 단원을 참조하십시오.
- 레시피를 수동으로 실행할 수 있지만 일반적으로 인스턴스의 수명 주기 중 주요 지점에서 발생하는 수명 주기 이벤트 세트에 대한 응답으로 AWS OpsWorks Stacks가 레시피를 실행하도록 합니다.

  자세한 내용은 레시피 실행 단원을 참조하십시오.

- AWS OpsWorks Stacks는 Chef 11.10 스택에서만 Berkshelf를 지원합니다. Berkshelf를 사용하여 쿡북 종속성을 관리하는 경우, Chef 11.4 또는 이전 버전을 실행하는 스택은 사용할 수 없습니다.

  자세한 내용은 Berkshelf 사용 섹션을 참조하세요.

주제

- 쿡북 저장
- 레시피 실행

쿡북 저장

Chef Server의 경우, 쿡북을 서버에 저장하고 서버에서 인스턴스로 쿡북을 배포합니다. Stacks를 사용하면 S3 또는 HTTP 아카이브나 Git 또는 Subversion 리포지토리와 같은 리포지토리에 AWS OpsWorks 쿡북을 저장할 수 있습니다. 쿡북을 설치할 때 AWS OpsWorks Stacks가 리포지토리에서 스택의 인스턴스로 코드를 다운로드하는 데 필요한 정보를 지정합니다.

Chef Server에서 마이그레이션하려면 쿡북을 이러한 리포지토리 중 하나에 두어야 합니다. 쿡북 리포지토리를 구성하는 방법에 대해서는 쿡북 리포지토리를 참조하세요.

레시피 실행

AWS OpsWorks Stacks의 각 계층에는 설정, 구성, 배포, 배포 취소 및 종료와 같은 수명 주기 이벤트 세트가 있으며, 각 이벤트는 인스턴스의 수명 주기 동안 키 포인트에서 발생합니다. 사용자 지정 레시피를 실행하려면 일반적으로 적절한 계층에서 적절한 이벤트에 사용자 지정 레시피를 할당합니다. 이벤트가 발생하면 AWS OpsWorks Stacks는 연결된 레시피를 실행합니다. 예를 들어 설정 이벤트는 인스턴스 부팅이 완료된 후 발생하므로 일반적으로 이 이벤트에는 패키지 설치 및 구성과 서비스 시작을 수행하는 레시피를 할당합니다.

레시피 실행 스택 명령을 사용하면 수동으로 레시피를 실행할 수 있습니다. 이 명령은 개발 및 테스트에도 유용하지만 수명 주기 이벤트에 매핑되지 않는 레시피를 실행하는 데도 사용할 수 있습니다. 레시피 실행 명령을 사용하여 설정 및 Configure 이벤트를 수동으로 트리거할 수도 있습니다.

AWS OpsWorks Stacks 콘솔 외에도 AWS CLI 또는 SDKs를 사용하여 레시피를 실행할 수 있습니다. 이 도구들은 모든 AWS OpsWorks Stacks API 작업을 지원하지만 API보다 사용이 더 간단합니다. create-deployment CLI 명령을 사용하여 수명 주기 이벤트를 트리거하면 연결된 모든 레시피가 실행됩니다. 이 명령을 사용하여 이벤트를 트리거하지 않고 하나 이상의 레시피를 실행할 수도 있습니다. 이에 상응하는 SDK 코드는 특정 언어에 의존하지만 대체로 CLI 명령과 비슷합니다.

다음 예제는 `create-deployment` CLI 명령을 사용하여 애플리케이션 배포를 자동화하는 두 가지 방법을 설명합니다.

- 단일 인스턴스가 포함된 사용자 지정 계층을 스택에 추가하여 정기적으로 앱을 배포합니다.

  인스턴스에서 `cron` 작업을 생성해 지정된 일정에 명령을 실행하는 사용자 지정 설정 레시피를 계층에 추가합니다. 레시피를 사용하여 `cron` 작업을 생성하는 방법의 예제는 Linux 인스턴스에서 Cron 작업 실행 단원을 참조하세요.
- `create-deployment` CLI 명령을 사용하여 앱을 배포하는 작업을 지속적 통합 파이프라인에 추가합니다.

Chef 환경 사용

AWS OpsWorks Stacks는 Chef 환경을 지원하지 않으며 `node.chef_environment` 항상를 반환합니다`_default`.

AWS OpsWorks Stacks 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 배포하는 모든 인스턴스는 스택에서 인스턴스의 역할을 정의하고 인스턴스 설정 및 구성, 패키지 설치, 애플리케이션 배포 등의 세부 정보를 제어하는 하나 이상의 계층의 구성원이어야 합니다. AWS OpsWorks 스택을 사용하여 계층을 생성하고 관리하는 방법에 대한 자세한 내용은 섹션을 참조하세요계층.

각 계층 설명에는 AWS OpsWorks Stacks가 각 계층의 수명 주기 이벤트에 대해 실행하는 기본 제공 레시피 목록이 포함되어 있습니다. 이러한 레시피는 https://github.com/aws/opsworks-cookbooks에 저장되어 있습니다. 목록에는 AWS OpsWorks Stacks에서 직접 실행하는 레시피만 포함됩니다. 이러한 레시피는 때때로 목록에는 나열되지 않는 종속 레시피를 실행합니다. 종속 및 사용자 지정 레시피를 비롯해 특정 이벤트에 대한 레시피의 전체 목록을 보려면 해당 수명 주기 이벤트의 Chef 로그에서 실행 목록을 확인하세요.

주제

- HAProxy 계층 레퍼런스
- HAProxy AWS OpsWorks 스택 계층
- MySQL 계층 참조
- MySQL OpsWorks 계층
- 애플리케이션 서버 계층 참조
- 애플리케이션 서버 계층
- ECS 클러스터 계층 참조
- 사용자 지정 계층 참조
- 기타 계층 참조
- 기타 계층

HAProxy 계층 레퍼런스

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

HAProxy 계층은 신뢰할 수 있는 고성능 TCP/HTTP 로드 밸런서인 HAProxy를 사용하여 TCP 및 HTTP 기반 애플리케이션을 위한 고가용성 로드 밸런싱 및 프록시 서비스를 제공합니다. 이는 매우 고부하에서 크롤링하면서도 지속성 또는 7계층 처리를 요구하는 웹 사이트에 특히 유용합니다.

HAProxy는 트래픽을 모니터링하고 연결된 인스턴스의 통계 및 상태를 웹 페이지에 표시합니다. 기본적으로 URI는 http://*DNSName*/haproxy?stats입니다. 여기서 *DNSName*은 HAProxy인스턴스의 DNS 이름입니다.

짧은 이름: lb

호환성: HAProxy 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached.

개방 포트: HAProxy는 포트 22(SSH), 80(HTTP) 및 443(HTTPS)에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 On

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-LB-Server

구성: 계층을 구성하려면 다음을 지정해야 합니다.

- 상태 확인 URI(기본값: http://*DNSName*/).
- 통계 URI(기본값: http://*DNSName*/haproxy?stats).
- 통계 암호(선택 항목).
- 상태 확인 메서드(선택 항목). 기본적으로 HAProxy는 HTTP OPTIONS 메서드를 사용합니다. 사용자가 GET 또는 HEAD를 지정할 수도 있습니다.
- 통계 활성화(선택 항목)
- 포트. 기본적으로 AWS OpsWorks Stacks는 HTTP 트래픽과 HTTPS 트래픽을 모두 처리하도록 HAProxy를 구성합니다. Chef 구성 템플릿인 `haproxy.cfg.erb`를 재정의하여 이 중 하나만 처리하도록 HAProxy를 구성할 수 있습니다.

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- haproxy

Configure 레시피:

- opsworks_ganglia::configure-client

- ssh_users
- agent_version
- haproxy::configure

Deploy 레시피:

- deploy::default
- haproxy::configure

Shutdown 레시피:

- opsworks_shutdown::default
- haproxy::stop

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 기본 위치에 HAProxy를 설치합니다.
- 로그 파일을 지정된 위치로 보내도록 syslog를 설정해야 합니다. 자세한 내용은 HAProxy를 참조하세요.

HAProxy AWS OpsWorks 스택 계층

> **Note**
>
> 이 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

AWS OpsWorks Stacks HAProxy 계층은 안정적인 고성능 TCP/HTTP 로드 밸런싱인 HAProxy 서버를 호스팅하는 인스턴스에 대한 청사진을 제공하는 AWS OpsWorks Stacks 계층입니다. 일반적으로 스몰 인스턴스 하나로 모든 애플리케이션 서버 트래픽을 처리하는 데 충분합니다.

> **Note**
>
> 스택은 단일 리전으로 제한됩니다. 여러 리전으로 애플리케이션을 분산하려면 각 리전마다 따로 스택을 생성해야 합니다.

HAProxy 계층을 만들려면

1. 탐색 창에서 [계층]를 클릭합니다.
2. [계층] 페이지에서 [계층 추가] 또는 [+ 계층]을 클릭합니다. [계층 유형]으로 HAProxy를 선택합니다.

계층 구성 설정은 다음과 같습니다(모두 선택 사항).

HAProxy 통계

계층이 통계를 수집하고 표시하는지 여부. 기본 값은 [예]입니다.

통계 URL

통계 페이지의 URL 경로. 전체 URL은 http://*DNSName*`StatisticsPath`이며, 여기서 *DNSName*은 연결된 인스턴스의 DNS 이름입니다. 기본 *`StatisticsPath`* 값은 /haproxy?stats이며, 다음과 같은 값에 해당합니다. http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/haproxy?stats.

통계 사용자 이름

통계 페이지를 보려면 제공해야 하는 통계 페이지의 사용자 이름. 기본값은 “OpsWorks”입니다.

통계 암호

통계 페이지를 보려면 제공해야 하는 통계 페이지 암호. 기본값은 무작위로 생성된 문자열입니다.

상태 검사 URL

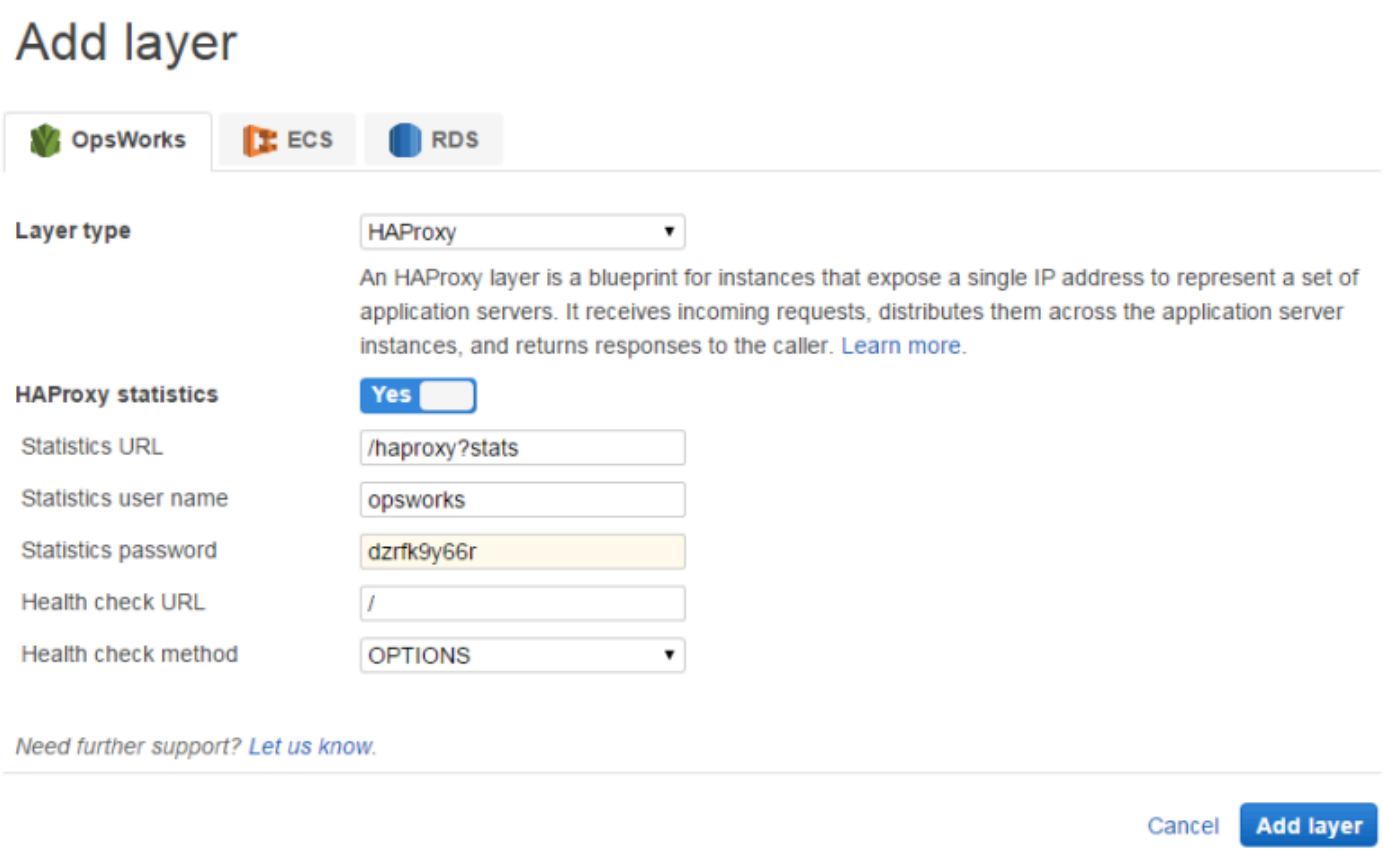
상태 검사 URL 접미사. HAProxy는 이 URL을 사용하여 주기적으로 각 애플리케이션 서버 인스턴스에서 HTTP 메서드를 호출하여 인스턴스가 작동하는지 판단합니다. 상태 검사가 실패할 경우 인스턴스가 수동으로 또는 자동 조정을 통해 재시작될 때까지 HAProxy는 해당 인스턴스로 트래픽 라우팅을 중단합니다. URL 접미사 기본값은 "/"이며, 서버 인스턴스의 홈 페이지 http://*DNSName*/에 해당합니다.

상태 검사 메서드

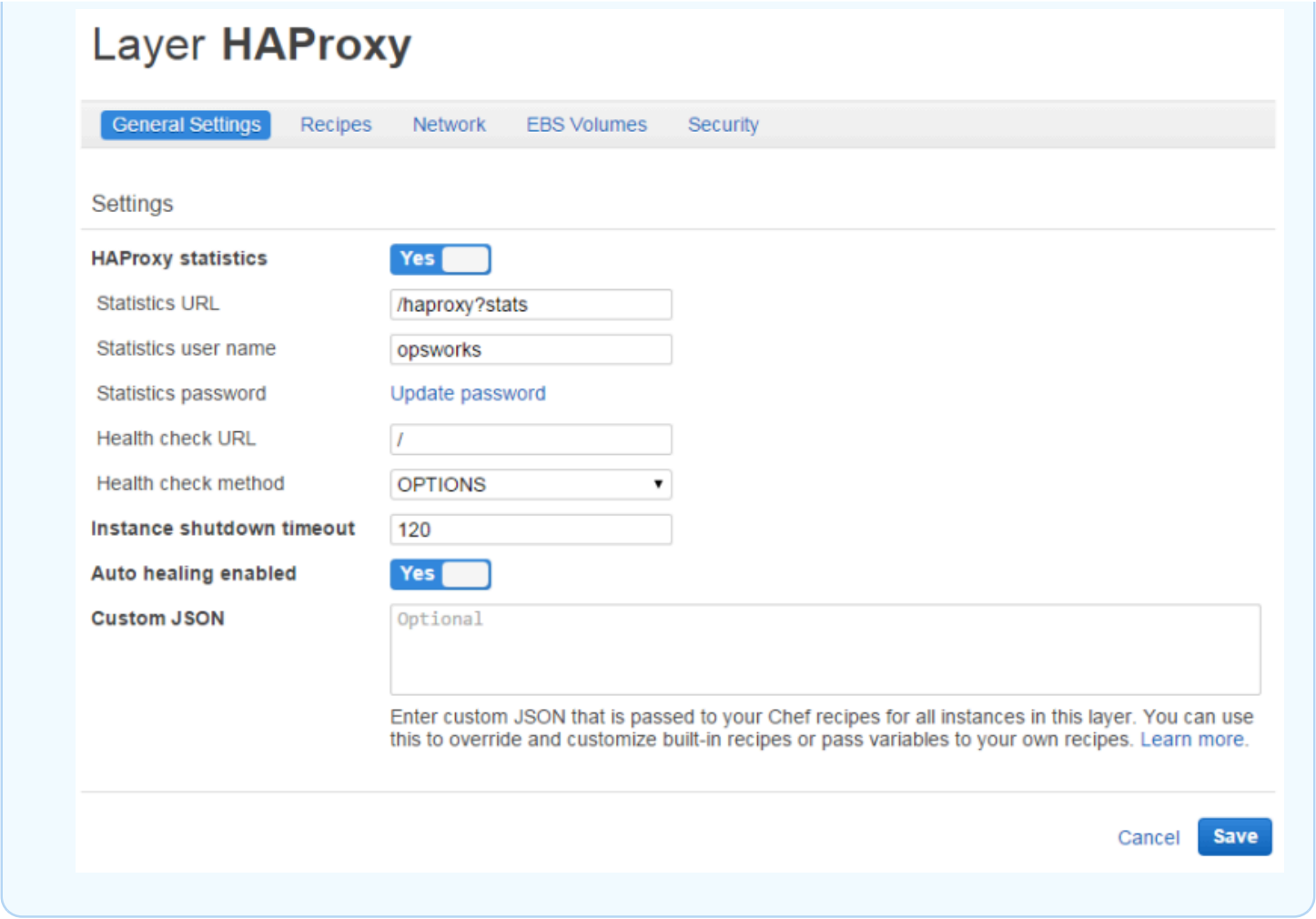
인스턴스가 작동하는지 검사하는 데 사용할 HTTP 메서드. 기본값은 [OPTIONS]이며 [GET] 또는 [HEAD]를 지정할 수도 있습니다. 자세한 정보는 httpchk를 참조하세요.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 계층 간 트래픽을 허용하도록 그룹의 설정이 올바른지 확인하세요. 자세한 내용은 새 스택 생성 단원을 참조하십시오.



> **Note**
>
> 나중에 사용할 수 있도록 암호를 기록합니다. AWS OpsWorks Stacks에서는 계층을 생성한 후 암호를 볼 수 없습니다. 하지만 계층의 편집 페이지로 이동하고 일반 설정 탭의 암호 업데이트를 클릭하면 암호를 업데이트할 수 있습니다.



HA프록시 계층 작동 방식

기본적으로 HAProxy는 다음 작업을 수행합니다.

- HTTP 및 HTTPS 포트에서 요청을 수신 대기합니다.

  Chef 구성 템플릿인 `haproxy.cfg.erb`를 재정의하여 HTTP 또는 HTTPS 포트 중 하나에서만 수신 대기하도록 HAProxy를 구성할 수 있습니다.
- 수신 트래픽을 애플리케이션 서버 계층의 멤버인 인스턴스로 라우팅합니다.

  기본적으로 AWS OpsWorks Stacks는 모든 애플리케이션 서버 계층의 멤버인 인스턴스에 트래픽을 분산하도록 HAProxy를 구성합니다. 예를 들어 Rails 앱 서버 및 PHP 앱 서버 계층이 모두 포함된 스택이 있고 HAProxy 마스터가 두 계층 모두의 인스턴스로 트래픽을 분산하는 경우가 있을 수 있습니다. 사용자 지정 레시피를 사용하여 기본 라우팅을 구성할 수 있습니다.
- 여러 가용 영역에 걸쳐 트래픽을 라우팅합니다.

한 가용 영역이 다운되면 로드 밸런서가 수신 트래픽을 다른 영역의 인스턴스로 라우팅하여 애플리케이션이 중단 없이 계속 실행될 수 있습니다. 이러한 이유로, 권장되는 방법은 애플리케이션 서버를 여러 가용 영역으로 분산하는 것입니다.

- 주기적으로 각 애플리케이션 서버 인스턴스에서 지정된 상태 검사 메서드를 실행하여 상태를 평가합니다

  메서드가 지정된 제한 시간 내에 반환되지 않으면 인스턴스가 실패한 것으로 간주되고 HAProxy가 인스턴스에 대한 요청 라우팅을 중지합니다. 또한 AWS OpsWorks Stacks는 실패한 인스턴스를 자동으로 대체하는 방법도 제공합니다. 자세한 내용은 자동 복구 사용 단원을 참조하십시오. 계층을 생성할 때 상태 검사 메서드를 변경할 수 있습니다.
- 통계를 수집하고 선택적으로 웹 페이지에 통계를 표시합니다.

> **Important**
>
> 기본 OPTIONS 메서드를 사용하여 상태 검사가 올바로 작동하려면 앱이 2xx 또는 3xx 상태 코드를 반환해야 합니다.

기본적으로 인스턴스를 HAProxy 계층에 추가하면 AWS OpsWorks Stacks는 인스턴스에 탄력적 IP 주소를 할당하여 애플리케이션을 나타내며, 이는 전 세계에 공개됩니다. HAProxy 인스턴스의 탄력적 IP 주소는 애플리케이션의 유일하게 공개된 URL이므로 기본 애플리케이션 서버 인스턴스에 대해 퍼블릭 도메인 이름을 생성하고 관리할 필요가 없습니다. 다음 그림에서 보듯이 [인스턴스] 페이지로 이동하여 인스턴스의 퍼블릭 IP 주소를 보면 이 주소를 확인할 수 있습니다. (EIP) 다음의 주소가 탄력적 IP 주소입니다. 탄력적 IP 주소에 대한 자세한 내용은 탄력적 IP 주소(EIP)를 참조하세요.

HAProxy 인스턴스를 중지하면 AWS OpsWorks Stacks는 탄력적 IP 주소를 유지하고 다시 시작할 때 인스턴스에 다시 할당합니다. HAProxy 인스턴스가 삭제될 경우 기본적으로 AWS OpsWorks Stacks는

인스턴스의 IP 주소를 삭제합니다. 주소를 보존하려면 다음 그림과 같이 [인스턴스의 탄력적 IP 삭제] 옵션을 선택 취소합니다.

이 옵션은 새 인스턴스를 계층에 추가하여 삭제된 인스턴스를 대체할 경우의 동작에 영향을 미칩니다.

- 삭제된 인스턴스의 탄력적 IP 주소를 유지한 경우 AWS OpsWorks Stacks는 새 인스턴스에 주소를 할당합니다.
- 그렇지 않으면 AWS OpsWorks Stacks가 인스턴스에 새 탄력적 IP 주소를 할당하므로 DNS 등록 기관 설정을 업데이트하여 새 주소에 매핑해야 합니다.

애플리케이션 서버 인스턴스가 수동으로 또는 자동 조정 또는 자동 복구의 결과로 온라인 상태가 되거나 오프라인 상태가 되면 트래픽을 현재 온라인 인스턴스 세트로 라우팅하도록 로드 밸런서 구성을 업데이트해야 합니다. 이 작업은 계층의 내장 레시피에 의해 자동으로 처리됩니다.

- 새 인스턴스가 온라인 상태가 되면 AWS OpsWorks Stacks는 수명 주기 구성 이벤트를 트리거합니다. HAProxy 계층의 내장 Configure 레시피가 새 애플리케이션 서버 인스턴스로도 요청을 분산하도록 로드 밸런서 구성을 업데이트합니다.
- 인스턴스가 오프라인 상태가 되거나 인스턴스가 상태 확인에 실패하면 AWS OpsWorks Stacks는 수명 주기 구성 이벤트도 트리거합니다. HAProxy Configure 레시피가 남은 온라인 인스턴스로만 트래픽을 라우팅하도록 로드 밸런서 구성을 업데이트합니다.

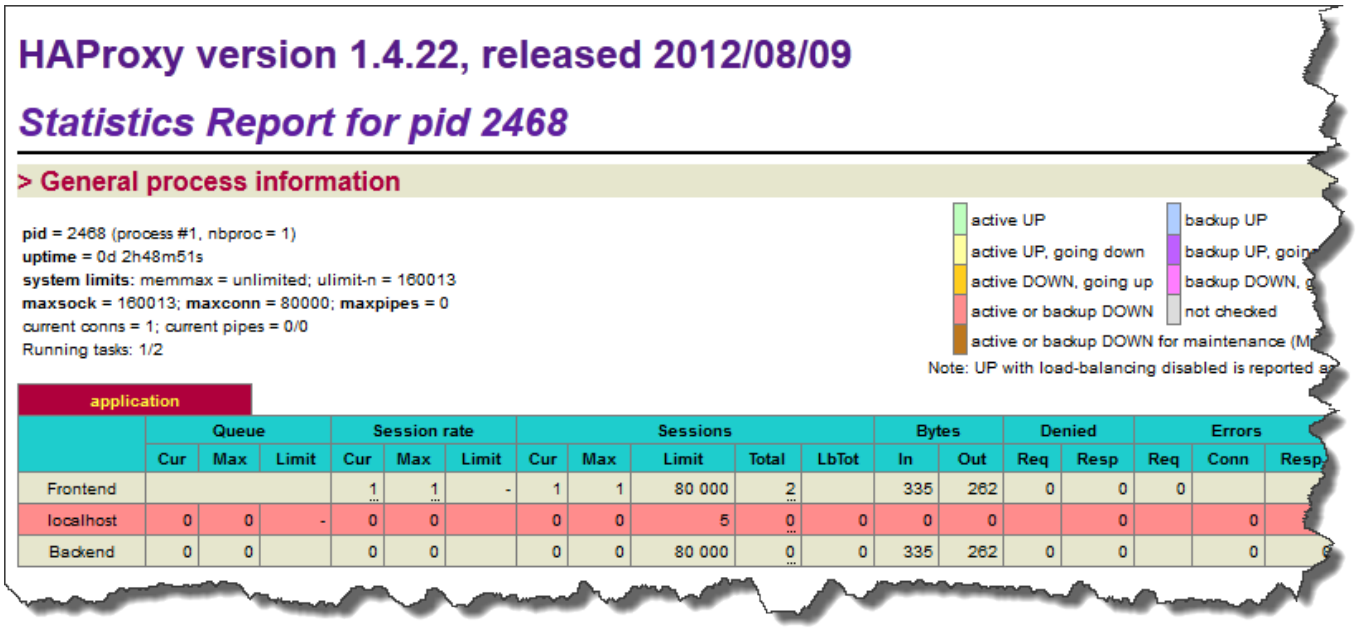
마지막으로 HAProxy 계층에서도 사용자 지정 도메인을 사용할 수 있습니다. 자세한 내용은 사용자 지정 도메인 사용 섹션을 참조하세요.

통계 페이지

통계 페이지를 활성화한 경우 HAProxy가 지정된 URL에서 다양한 측정치가 포함된 페이지를 표시합니다.

HAProxy 통계를 보려면

1. 인스턴스의 세부 정보 페이지에서 HAProxy 인스턴스의 퍼블릭 DNS 이름을 가져와 복사하세요.
2. 계층 페이지에서 HAProxy를 클릭하여 계층의 세부 정보 페이지를 엽니다.
3. 계층 세부 정보에서 통계 URL을 확인하여 퍼블릭 DNS 이름에 추가합니다. 예를 들면 **`http://ec2-54-245-102-172.us-west-2.compute.amazonaws.com/haproxy?stats`**를 추가합니다.
4. 이전 단계에서 복사한 URL을 브라우저에 붙여 넣고 계층 생성 시 지정한 사용자 이름 및 암호를 사용하여 통계 페이지를 엽니다.



MySQL 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

MySQL 계층은 널리 사용되는 관계형 데이터베이스 관리 시스템인 MySQL을 지원합니다. AWS OpsWorks Stacks는 운영 체제에 따라 사용 가능한 최신 버전을 설치합니다. MySQL 인스턴스를 추가할 경우 필요한 액세스 정보가 애플리케이션 서버 계층으로 제공됩니다. 사용자 지정 Chef 레시피를 작성하여 마스터-마스터 또는 마스터-슬레이브 구성을 설정해야 합니다.

짧은 이름: db-master

호환성: MySQL 계층은 다음 계층과 호환됩니다. 사용자 지정, lb, memcached, monitoring-master, nodejs-app, php-app, rails-app, web.

개방 포트: MySQL 계층은 포트 22(SSH) 그리고 스택의 웹 서버, 사용자 지정 서버 및 Rails, PHP 및 Node.js 애플리케이션 서버의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 예. 위치: `/vol/mysql`

기본 보안 그룹: AWS-OpsWorks-DB-Master-Server

구성: MySQL 계층을 구성하려면 다음을 지정해야 합니다.

- 루트 사용자 암호
- MySQL 엔진

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client

- dependencies
- ebs
- opsworks_ganglia::client
- mysql::server
- dependencies
- deploy::mysql

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- deploy::mysql

Deploy 레시피:

- deploy::default
- deploy::mysql

Shutdown 레시피:

- opsworks_shutdown::default
- mysql::stop

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 MySQL과 해당 로그 파일을 기본 위치에 설치합니다. 자세한 정보는 MySQL 설명서를 참조하세요.

MySQL OpsWorks 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

MySQL OpsWorks 계층은 MySQL 데이터베이스 마스터로 작동하는 Amazon EC2 인스턴스에 대한 청사진을 제공합니다. 내장 레시피는 애플리케이션 서버 계층에 배포된 각 애플리케이션의 데이터베이스를 생성합니다. 예를 들어 "myapp"이라는 PHP 애플리케이션을 배포하면 이 레시피가 “myapp” 데이터베이스를 생성합니다.

MySQL 계층에는 다음 구성 설정이 있습니다.

[MySQL 루트 사용자 암호]

(필수) 루트 사용자 암호.

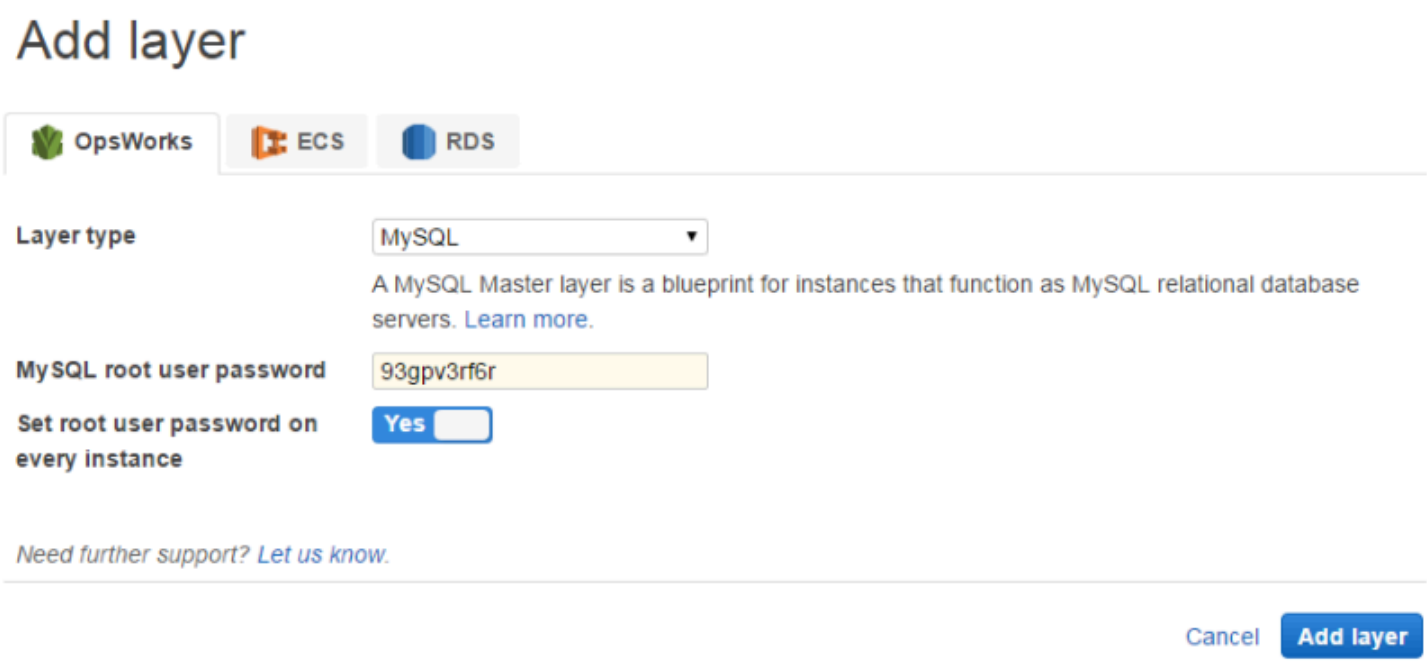
[인스턴스마다 루트 사용자 암호 설정]

(선택 사항) 스택의 모든 인스턴스에 설치되는 스택 구성 및 배포 속성에 루트 사용자 암호가 포함되는지 여부. 기본 설정은 [예]입니다.

이 값을 아니요로 설정하면 AWS OpsWorks Stacks는 루트 암호만 애플리케이션 서버 인스턴스에 전달합니다.

사용자 지정 보안 그룹

(선택 사항) 계층에 연결할 사용자 지정 보안 그룹. 자세한 내용은 새 스택 생성 섹션을 참조하세요.



각각이 별도의 MySQL 데이터베이스 마스터를 나타내는 하나 이상의 인스턴스를 계층에 추가할 수 있습니다. 그런 다음 인스턴스를 앱에 연결하면 앱의 애플리케이션 서버에 필요한 연결 정보가 설치됩니다. 그러면 애플리케이션이 연결 정보를 사용하여 인스턴스의 데이터베이스 서버에 연결할 수 있습니다.

애플리케이션 서버 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 여러 애플리케이션 및 정적 웹 페이지 서버를 지원합니다.

주제

- AWS Flow(루비) 계층 참조
- Java 앱 서버 계층 참조
- Node.js 앱 서버 계층 참조

- PHP 앱 서버 계층 참조
- Rails 앱 서버 계층 참조
- Static Web Server 계층 참조

AWS Flow(루비) 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

AWS Flow(Ruby) 계층은 Amazon Simple Workflow Service 활동 및 워크플로우 worker를 호스트하는 인스턴스의 청사진이 됩니다.

짧은 이름: aws-flow-ruby

호환성: AWS Flow(Ruby) 계층은 PHP 앱 서버, MySQL, Memcached, Ganglia 및 사용자 지정 계층과 호환됩니다.

개방 포트: 없음.

IAM 역할: aws-opsworks-ec2-role-with-swf는 요청 시 AWS OpsWorks Stacks가 자동으로 생성하는 표준 AWS Flow(Ruby) 역할입니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-AWS-Flow-Ruby-Server

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_aws_flow_ruby::설정

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version
- opsworks_aws_flow_ruby::configure

Deploy 레시피:

- deploy::default
- deploy::aws-flow-ruby

Undeploy 레시피:

- deploy::aws-flow-ruby-undeploy

Shutdown 레시피:

- opsworks_shutdown::default

Java 앱 서버 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Java 앱 서버 계층은 Apache Tomcat 7.0 애플리케이션 서버를 지원합니다.

짧은 이름: java-app

호환성: Java 앱 서버 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached.

개방 포트: Java 앱 서버 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 로드 밸런서의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-Java-App-Server

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs

- opsworks_ganglia::client
- opsworks_java::설정

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_java::configure

Deploy 레시피:

- deploy::default
- deploy::java

Undeploy 레시피:

- deploy::java-undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- deploy::java-stop

설치:

- Tomcat이 `/usr/share/tomcat7`에 설치됩니다.
- 로그 파일을 생성하는 방법에 대한 자세한 정보는 Tomcat에 로그인 단원을 참조하세요.

Node.js 앱 서버 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Node.js 앱 서버 계층은 고가용성 네트워크 애플리케이션 서버를 구현하기 위한 플랫폼인 Node.js 애플리케이션 서버를 지원합니다. 프로그램은 JavaScript로 작성되며, 이벤트 중심 비동기식 I/O를 사용하여 오버헤드를 최소화하고 확장성을 최대화합니다.

짧은 이름: nodejs-app

호환성: Node.js 앱 서버 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached, monitoring-master.

개방 포트: Node.js 앱 서버 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 로드 밸런서의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-nodejs-App-Server

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_nodejs
- opsworks_nodejs::npm

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_nodejs::configure

Deploy 레시피:

- deploy::default
- opsworks_nodejs
- opsworks_nodejs::npm
- deploy::nodejs

Undeploy 레시피:

- deploy::nodejs-undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- deploy::nodejs-stop

설치:

- Node.js가 `/usr/local/bin/node`에 설치됩니다.
- 로그 파일을 생성하는 방법에 대한 자세한 정보는 Nodejitsu 웹사이트의 node.js에 로그인하는 방법을 참조하세요.

Node.js 애플리케이션 구성:

- Node.js가 실행하는 메인 파일은 이름이 `server.js`여야 하며, 배포된 애플리케이션의 루트 디렉터리에 상주해야 합니다.
- Node.js 애플리케이션은 포트 80(또는 포트 443, 해당되는 경우)에서 수신 대기하도록 설정해야 합니다.

> **Note**
>
> Express를 실행하는 Node.js 앱은 공통적으로 다음 코드를 사용하여 수신 포트를 설정합니다. 여기서 `process.env.PORT`는 기본 포트를 나타내며 80으로 확인됩니다.
>
> ```
> app.set('port', process.env.PORT || 3000);
> ```
>
> AWS OpsWorks Stacks에서는 다음과 같이 포트 80을 명시적으로 지정해야 합니다.
>
> ```
> app.set('port', 80);
> ```

PHP 앱 서버 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

PHP 앱 서버 계층은 mod_php가 있는 Apache2를 사용하여 PHP 애플리케이션 서버를 지원합니다.

짧은 이름: php-app

호환성: PHP 앱 서버 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached, monitoring-master, rails-app.

개방 포트: PHP 앱 서버 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 로드 밸런서의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-PHP-App-Server

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::client
- dependencies
- mod_php5_apache2

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- mod_php5_apache2::php
- php::configure

Deploy 레시피:

- deploy::default
- deploy::php

Undeploy 레시피:

- deploy::php-undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- apache2::stop

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 Apache2, mod_php 및 관련 로그 파일을 기본 위치에 설치합니다. 설치에 대한 자세한 정보는 Apache를 참조하세요. 로깅에 대한 자세한 정보는 로그 파일 단원을 참조하세요.

Rails 앱 서버 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Rails 앱 서버 계층은 Ruby on Rails 애플리케이션 서버를 지원합니다.

짧은 이름: rails-app

호환성: Rails 앱 서버 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached, monitoring-master, php-app.

포트: Rails 앱 서버 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 로드 밸런서의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-Rails-App-Server

구성: Rails 앱 서버 계층을 구성하려면 다음을 지정해야 합니다.

- Ruby 버전
- Rails 스택
- Rubygems 버전
- Bundler 설치 및 관리 여부
- Bundler 버전

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- apache2 apache2::mod_deflate
- passenger_apache2
- passenger_apache2::mod_rails
- passenger_apache2::rails

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- rails::configure

Deploy 레시피:

- deploy::default
- deploy::rails

Undeploy 레시피:

- deploy::rails-undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- apache2::stop

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 mod_passenger, mod_rails 및 연결된 로그 파일이 있는 Apache2를 기본 위치에 설치합니다. 설치에 대한 자세한 정보는 Phusion Passenger를 참조하세요. 로깅에 대한 자세한 정보는 로그 파일 단원을 참조하세요.

Static Web Server 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Static Web Server 계층은 정적 HTML 페이지를 서비스하며, 여기에는 JavaScript 같은 클라이언트 측 코드가 포함될 수 있습니다. 이 계층은 오픈 소스 HTTP, 리버스 프록시 및 메일 프록시 서버인 Nginx를 기반으로 합니다.

짧은 이름: web

호환성: Static Web Server 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached.

개방 포트: Static Web Server 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 로드 밸런서의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-Web-Server

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- nginx

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 레시피:

- deploy::default
- deploy::web

Undeploy 레시피:

- deploy::web-undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- nginx::stop

설치:

- Nginx가 `/usr/sbin/nginx`에 설치됩니다.
- Nginx 로그 파일은 `/var/log/nginx`에 있습니다.

애플리케이션 서버 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이들 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

AWS OpsWorks Stacks는 여러 애플리케이션 서버를 지원합니다. 여기서 "애플리케이션"에는 정적 웹 페이지가 포함됩니다. 각 서버 유형에는 별도의 AWS OpsWorks Stacks 계층이 있으며, 각 계층의 인스턴스에 애플리케이션 서버 및 관련 패키지를 설치하고 앱을 배포하는 등의 작업을 처리하는 내장 레시피가 있습니다. 예를 들어 Java 앱 서버 계층은 Apache, Tomcat, OpenJDK를 비롯한 여러 패키지를 설치하고 각 계층의 인스턴스에 Java 앱을 배포합니다.

애플리케이션 서버 계층을 사용하는 기본적 절차는 다음과 같습니다.

1. 사용 가능한 앱 서버 계층 유형 중 하나를 생성합니다.
2. 계층에 하나 이상의 인스턴스를 추가합니다.
3. 앱을 생성해 인스턴스에 배포합니다. 자세한 내용은 앱 섹션을 참조하세요.
4. (선택 사항) 계층에 여러 인스턴스가 있는 경우, 수신 트래픽을 인스턴스에 분산하는 로드 밸런서를 추가할 수 있습니다. 자세한 내용은 HAProxy AWS OpsWorks 스택 계층 섹션을 참조하세요.

주제

- AWS Flow(Ruby) 계층
- Java 앱 서버 AWS OpsWorks 스택 계층
- Node.js 앱 서버 AWS OpsWorks 스택 계층
- PHP 앱 서버 AWS OpsWorks 스택 계층
- Rails 앱 서버 AWS OpsWorks 스택 계층
- 정적 웹 서버 AWS OpsWorks 스택 계층

AWS Flow(Ruby) 계층

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

AWS Flow(Ruby) 계층은 Amazon SWF 활동 및 워크플로 작업자를 호스팅하는 인스턴스에 대한 청사진을 제공하는 AWS OpsWorks Stacks 계층입니다. 작업자는 Amazon SWF의 모든 이점을 제공하면서 분산 비동기 애플리케이션을 구현하는 프로세스를 단순화하는 프로그래밍 프레임워크인 AWS Flow Framework for Ruby를 사용하여 구현됩니다. 이 프레임워크는 비즈니스 프로세스, 미디어 인코딩, 장기 실행 작업, 백그라운드 프로세싱 등 다양한 시나리오를 처리하는 애플리케이션을 구현하는 데 이상적입니다.

AWS Flow(Ruby) 계층에는 다음 구성 설정이 포함됩니다.

RubyGems 버전

프레임워크의 Gem 버전.

Bundler 버전

Bundler 버전.

EC2 인스턴스 프로파일

계층의 인스턴스가 사용할 사용자 정의 Amazon EC2 인스턴스 프로파일. 이 프로파일은 계층의 인스턴스에서 실행되는 애플리케이션에게 Amazon SWF에 액세스할 수 있는 권한을 부여해야 합니다.

계정에 적절한 프로필이 없는 경우 SWF 액세스 권한이 있는 새 프로필을 선택하여 AWS OpsWorks Stacks가 프로필을 업데이트하도록 하거나 IAM 콘솔을 사용하여 직접 업데이트할 수 있습니다. 그러면 이후의 모든 AWS Flow 계층에 업데이트된 프로파일을 사용할 수 있습니다. 다음은 IAM 콘솔을 사용해 프로파일을 생성하는 방법에 대한 간략한 설명입니다. 자세한 내용은 Identity and Access Management in Amazon Simple Workflow Service를 참조하세요.

AWS Flow(Ruby) 인스턴스용 프로파일 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택하고 정책 생성을 선택하여 새 고객 관리형 정책을 생성합니다.
3. 서비스에서 SWF를 선택합니다.
4. 활동에서 모든 SWF 활동(swf:*)을 선택합니다.
5. Amazon 리소스 이름(ARN)에 작업자가 액세스할 수 있는 Amazon SWF 도메인을 지정하는 ARN을 입력합니다. 모든 도메인에 액세스하도록 하려면 **`All resources`**를 선택합니다.
6. 다음을 선택합니다.
7. 정책을 식별하는 태그를 입력할 수도 있습니다.
8. 다음을 선택합니다.
9. 모두 마쳤으면 정책 생성을 선택합니다.
10. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
11. 역할 이름을 지정하고 Next Step을 클릭합니다. 역할이 생성된 이후에는 역할의 이름을 바꿀 수 없습니다.
12. AWS 서비스와 EC2를 차례대로 선택합니다.
13. 다음을 선택합니다.
14. 권한 정책 목록에서 이전에 생성한 정책을 선택합니다.
15. 다음을 선택합니다.

16. 역할 이름을 입력하고 역할 생성을 선택합니다. 역할이 생성된 이후에는 역할의 이름을 바꿀 수 없습니다.
17. AWS OpsWorks Stacks에서 AWS Flow(Ruby) 계층을 생성할 때이 프로파일을 지정합니다.

Java 앱 서버 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Java 앱 서버 계층은 Java 애플리케이션 서버 역할을 하는 인스턴스에 대한 블루프린트를 제공하는 AWS OpsWorks Stacks 계층입니다. 이 계층은 Apache Tomcat 7.0 및 Open JDK 7을 기반으로 합니다. AWS OpsWorks Stacks는 Java 커넥터 라이브러리도 설치하므로 Java 앱이 JDBC `DataSource` 객체를 사용하여 백엔드 데이터 스토어에 연결할 수 있습니다.

설치: Tomcat이 `/usr/share/tomcat7`에 설치됩니다.

[계층 추가] 페이지는 다음 구성 옵션을 제공합니다.

Java VM 옵션

이 설정을 사용하여 사용자 지정 Java VM 옵션을 지정할 수 있습니다. 기본 옵션은 없습니다. 예를 들어 공통 옵션 세트는 `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC`입니다. Java VM 옵션을 사용하는 경우 유효한 옵션 세트를 전달해야 합니다. AWS OpsWorks Stacks는 문자열을 검증하지 않습니다. 유효하지 않은 옵션을 전달하려고 하면 일반적으로 Tomcat 서버가 시작하지 못해 설정이 실패합니다. 이런 현상이 발생하는 경우, 인스턴스의 설정 Chef 로그에서 상세 정보를 검사할 수 있습니다. Chef 로그를 보고 해석하는 방법에 대한 자세한 정보는 Chef 로그 단원을 참조하세요.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다. 자세한 내용은 Elastic Load Balancing 계층 섹션을 참조하세요.

사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 다른 구성 설정을 지정할 수 있습니다. 자세한 내용은 사용자 지정 구성 섹션을 참조하세요.

> **Important**
>
> Java 애플리케이션이 SSL을 사용하는 경우, 가능하다면 SSLv3를 비활성화하여 CVE-2014-3566에 설명된 취약성을 해결하는 것이 좋습니다. 자세한 내용은 Apache 서버에서 SSLv3 비활성화 섹션을 참조하세요.

주제

- Apache 서버에서 SSLv3 비활성화
- 사용자 지정 구성
- Java 앱 배포

Apache 서버에서 SSLv3 비활성화

SSLv3를 비활성화하려면 Apache 서버 `ssl.conf` 파일의 `SSLProtocol` 설정을 수정해야 합니다. 이렇게 하려면 Java 앱 서버 계층의 설치 레시피가 `ssl.conf`을 생성하는 데 사용하는 내장 apache2 쿡북의 `ssl.conf.erb` 템플릿 파일을 재정의해야 합니다. 세부 사항은 계층의 인스턴스에 대해 지정하는 운영 체제에 따라 다릅니다. 다음은 Amazon Linux 및 Ubuntu 시스템에 필요한 수정을 요약한 것입니다. Red Hat Enterprise Linux(RHEL) 시스템의 경우, SSLv3가 자동으로 비활성화됩니다. 내장 템플릿을 재정의하는 방법에 대한 자세한 정보는 사용자 지정 템플릿 사용 를 참조하세요.

Amazon Linux

이러한 운영 체제의 `ssl.conf.erb` 파일은 `apache2` 쿡북의 `apache2/templates/default/mods` 디렉터리에 있습니다. 다음은 내장 파일의 관련 부분을 보여 줍니다.

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv2
</IfModule>
```

다음과 같이 ssl.conf.erb를 재정의하고 SSLProtocol 설정을 수정합니다.

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv3 -SSLv2
</IfModule>
```

Ubuntu 14.04 LTS

이 운영 체제의 ssl.conf.erb 파일은 apache2 쿡북의 apache2/templates/ubuntu-14.04/mods 디렉터리에 있습니다. 다음은 내장 파일의 관련 부분을 보여 줍니다.

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all
...
```

이 설정을 다음으로 변경합니다.

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all -SSLv3 -SSLv2
...
```

사용자 지정 구성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 추가 구성 설정을 `opsworks_java` 네임스페이스에 있는 기본 제공 속성으로 노출합니다. 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 내장 속성을 재정의하고 사용자 지정 값을 지정할 수 있습니다. 예를 들어 JVM 및 Tomcat 버전은 내장 `jvm_version` 및 `java_app_server_version` 속성으로 표시되며, 둘 다 7로 설정됩니다. 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 둘 중 하나 또는 둘 모두를 6으로 설정할 수 있습니다. 다음 예제에서는 사용자 지정 JSON을 사용하여 두 속성 모두를 6으로 설정합니다.

```
{
  "opsworks_java": {
    "jvm_version": 6,
    "java_app_server_version" : 6
  }
}
```

자세한 내용은 사용자 지정 JSON 사용 섹션을 참조하세요.

사용자 지정 구성의 또 다른 예제는 `use_custom_pkg_location`, `custom_pkg_location_url_debian`, `custom_pkg_location_url_rhel` 속성을 재정의하여 사용자 지정 JDK를 설치하는 것입니다.

> **Note**
>
> 내장 쿡북을 재정의하는 경우, 이러한 구성 요소를 직접 업데이트해야 합니다.

속성과 속성 재정의 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요. 내장 속성의 목록은 opsworks_java 속성를 참조하세요.

Java 앱 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

다음 주제에서는 Java 앱 서버 계층의 인스턴스에 앱을 배포하는 방법을 설명합니다. 예제는 JSP 앱용이지만 다른 유형의 Java 앱 설치에도 기본적으로 동일한 절차를 사용할 수 있습니다.

지원되는 모든 리포지토리에서 JSP 페이지를 배포할 수 있습니다. WAR 파일을 배포하려는 경우 AWS OpsWorks Stacks는 Git 또는 Subversion 리포지토리가 아닌 Amazon S3 또는 HTTP 아카이브에서 배포된 WAR 파일을 자동으로 추출합니다. WAR 파일에 Git 또는 하위 버전을 사용하려면 다음 중 하나를 수행할 수 있습니다.

- 추출된 아카이브를 리포지토리에 저장합니다.
- 리포지토리에 WAR 파일을 저장하고 다음 예제의 설명처럼 Chef 배포 후크를 사용하여 아카이브를 추출합니다.

Chef 배포 후크를 사용하여 배포 4단계 중 어느 단계에서나 사용자 제공 Ruby 애플리케이션을 인스턴스에서 실행할 수 있습니다. 애플리케이션 이름이 단계를 결정합니다. 다음은 Git 또는 하위 버전 리포지토리에서 배포된 WAR 파일을 추출하는 `before_migrate.rb`라는 Ruby 애플리케이션의 예제입니다. 이 이름은 애플리케이션을 Checkout 배포 후크에 연결하므로 코드가 검사 후 마이그레이션 전에 배포 작업이 시작될 때 애플리케이션이 실행됩니다. 이 예제를 사용하는 방법에 대한 자세한 정보는 Chef 배포 후크 사용를 참조하세요.

```
::Dir.glob(::File.join(release_path, '*.war')) do |archive_file|
  execute "unzip_#{archive_file}" do
    command "unzip #{archive_file}"
    cwd release_path
  end
end
```

> **Note**
>
> JSP 앱에 업데이트를 배포하면 Tomcat이 업데이트를 인식하지 못하고 기존 앱 버전을 계속 실행할 수 있습니다. 예를 들어 JSP 페이지만 포함된 .zip 파일로 앱을 배포하는 경우, 이런 현상이 생길 수 있습니다. Tomcat이 가장 최근에 배포된 버전을 실행하도록 하려면 `web.xml` 파일이 포함된 WEB-INF 디렉터리가 프로젝트의 루트 디렉터리에 포함되어야 합니다. `web.xml` 파일에는 다양한 콘텐츠가 포함될 수 있지만 Tomcat이 업데이트를 인식하고 현재 배포된 앱 버전을 실행하도록 하는 데는 다음으로 충분합니다. 각 업데이트의 버전을 변경할 필요는 없습니다. Tomcat은 버전이 변경되지 않은 경우에도 업데이트를 인식합니다.
>
> ```
> <context-param>
>   <param-name>appVersion</param-name>
>   <param-value>0.1</param-value>
> </context-param>
> ```

주제

- JSP 앱 배포
- 백엔드 데이터베이스를 사용하여 JSP 앱 배포

JSP 앱 배포

JSP 앱을 배포하려면 이름 및 리포지토리 정보를 지정합니다. 필요한 경우, 도메인 및 SSL 설정도 지정할 수 있습니다. 앱을 생성하는 방법에 대한 자세한 정보는 앱 추가 단원을 참조하세요. 다음 절차에서는 퍼블릭 Amazon S3 아카이브에서 간단한 JSP 페이지를 생성하고 배포하는 방법을 살펴봅니다. 프라이빗 Amazon S3 아카이브 등 다른 리포지토리 유형을 사용하는 방법에 대해서는 애플리케이션 소스 단원을 참조하세요.

다음 예제는 단순히 일부 시스템 정보를 표시하는 JSP 페이지를 보여 줍니다.

```
<%@ page import="java.net.InetAddress" %>
<html>
<body>
<%
    java.util.Date date = new java.util.Date();
    InetAddress inetAddress = InetAddress.getLocalHost();
%>
```

```
The time is
<%
    out.println( date );
    out.println("<br>Your server's hostname is "+inetAddress.getHostName());
%>
<br>
</body>
</html>
```

> **Note**
>
> 다음 절차에서는 스택 생성, 계층에 인스턴스 추가 등등의 기본적 내용에 이미 익숙하다고 가정합니다. AWS OpsWorks Stacks를 처음 사용하는 경우 먼저 단원을 참조하십시오Chef 11 Linux 스택 시작하기.

Amazon S3 아카이브에서 JSP 페이지를 배포하려면

1. Java 앱 서버 계층으로 스택을 생성하고, 해당 계층에 연중무휴 인스턴스를 추가한 다음 시작하세요.
2. 코드를 `simplejsp.jsp` 파일에 복사하고 이 파일을 `simplejsp` 폴더에 저장한 다음 이 폴더의 `.zip` 아카이브를 생성합니다. 이름은 임의적이므로, 원하는 파일 또는 폴더 이름은 어떤 것이든 사용할 수 있습니다. 또한 gzip, bzip2, tarball 또는 Java WAR 파일을 비롯하여 다른 아카이브 유형을 사용할 수도 있습니다. 참고로 AWS OpsWorks Stacks는 압축되지 않은 tarball을 지원하지 않습니다. 여러 JSP 페이지를 배포하려면 배포하려는 페이지를 동일한 아카이브에 저장합니다.
3. 이 아카이브를 Amazon S3 버킷에 업로드하고 이 파일을 퍼블릭으로 설정합니다. 나중에 사용하기 위해 파일의 URL를 복사해 둡니다. 버킷 생성 및 파일 업로드 방법에 대한 자세한 내용은 Amazon Simple Storage Service 시작하기를 참조하세요.
4. 스택에 앱을 추가하고 다음 설정을 지정합니다.

   - 명칭 – `SimpleJSP`
   - 앱 유형 – `Java`
   - 리포지토리 유형 – `Http Archive`
   - 리포지토리 URL - 아카이브 파일의 Amazon S3 URL.

   나머지 설정에 대해서는 기본값을 사용하고 [앱 추가]를 클릭하여 앱을 생성합니다.
5. 앱을 Java 앱 서버 인스턴스에 배포합니다.

이제 앱의 URL로 이동하여 앱을 볼 수 있습니다. 도메인을 지정하지 않았다면 인스턴스의 퍼블릭 IP 주소나 퍼블릭 DNS 이름을 사용하여 URL을 생성할 수 있습니다. 인스턴스의 퍼블릭 IP 주소 또는 퍼블릭 DNS 이름을 가져오려면 AWS OpsWorks Stacks 콘솔로 이동하여 인스턴스 페이지에서 인스턴스 이름을 클릭하여 세부 정보 페이지를 엽니다.

나머지 URL은 앱의 짧은 이름, 즉 AWS OpsWorks Stacks가 앱을 생성할 때 지정한 앱 이름에서 생성하는 소문자 이름에 따라 달라집니다. 예를 들어 SimpleJSP의 짧은 이름은 simplejsp입니다. 앱의 짧은 이름은 앱의 세부 정보 페이지에서 가져올 수 있습니다.

- 짧은 이름이 `root`인 경우, `http://`*`public_DNS`*`/`*`appname`*`.jsp` 또는 `http://`*`public_IP`*`/`*`appname`*`.jsp`을 사용할 수 있습니다.
- 그렇지 않으면 `http://`*`public_DNS`*`/`*`app_shortname`*`/`*`appname`*`.jsp` 또는 `http://`*`public_IP`*`/`*`app_shortname`*`/`*`appname`*`.jsp`을 사용할 수 있습니다.

앱에 도메인을 지정한 경우, URL은 `http://`*`domain`*`/`*`appname`*`.jsp`입니다.

예제의 URL은 `http://192.0.2.0/simplejsp/simplejsp.jsp`와 비슷합니다.

같은 인스턴스에 여러 앱을 배포하려면 `root`를 짧은 이름으로 사용해선 안 됩니다. 이 경우, URL 충돌로 앱이 정상 작동하지 않습니다. 대신 앱마다 각기 다른 도메인 이름을 할당하세요.

백엔드 데이터베이스를 사용하여 JSP 앱 배포

JSP 페이지는 JDBC `DataSource` 객체를 사용하여 백엔드 데이터베이스에 연결할 수 있습니다. 이러한 앱은 이전 섹션의 절차와 연결을 설정하는 추가 단계를 사용하여 생성 및 배포할 수 있습니다.

다음 JSP 페이지는 `DataSource` 객체에 연결하는 방법을 보여 줍니다.

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
    <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
    <%
      StringBuffer output = new StringBuffer();
      DataSource ds = null;
      Connection con = null;
      Statement stmt = null;
      ResultSet rs = null;
```

```
      try {
        Context initCtx = new InitialContext();
        ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
        con = ds.getConnection();
        output.append("Databases found:<br>");
        stmt = con.createStatement();
        rs = stmt.executeQuery("show databases");
        while (rs.next()) {
          output.append(rs.getString(1));
          output.append("<br>");
        }
      }
      catch (Exception e) {
        output.append("Exception: ");
        output.append(e.getMessage());
        output.append("<br>");
      }
      finally {
        try {
          if (rs != null) {
            rs.close();
          }
          if (stmt != null) {
            stmt.close();
          }
          if (con != null) {
            con.close();
          }
        }
        catch (Exception e) {
          output.append("Exception (during close of connection): ");
          output.append(e.getMessage());
          output.append("<br>");
        }
      }
    %>
    <%= output.toString() %>
  </body>
</html>
```

AWS OpsWorks Stacks는 `DataSource` 객체를 생성 및 초기화하고 논리적 이름에 바인딩한 다음 Java 이름 지정 및 디렉터리 인터페이스(JNDI) 이름 지정 서비스에 이름을 등록합니다. 완전한 논

리적 이름은 `java:comp/env/`*`user-assigned-name`*입니다. 다음에서 설명하는 것처럼 사용자 지정 JSON 속성을 스택 구성 및 배포 속성에 추가함으로써 이름의 사용자 할당 부분을 지정하여 `['opsworks_java']['datasources']` 속성을 정의해야 합니다.

MySQL 데이터베이스에 연결하는 JSP 페이지를 배포하려면

1. Java 앱 서버 계층으로 스택을 만들고 각 계층에 연중무휴 인스턴스를 추가한 다음 시작합니다.
2. 데이터베이스 계층을 스택에 추가합니다. 세부적인 사항은 사용하는 데이터베이스에 따라 달라집니다.

   예를 들어 MySQL 인스턴스를 사용하려면 스택에 MySQL 계층을 추가하고, 이 계층에 24/7 인스턴스를 추가한 다음 해당 인스턴스를 시작합니다.

   예를 들어 Amazon RDS(MySQL) 인스턴스를 시작하려면,

   - 인스턴스의 MySQL 데이터베이스 엔진을 지정합니다.
   - AWS-OpsWorks-DB-Master-Server(***`security_group_id`***) 및 AWS-OpsWorks-Java-App-Server(***`security_group_id`***) 보안 그룹을 인스턴스에 할당합니다. AWS OpsWorks Stacks는 리전에서 첫 번째 스택을 생성할 때 이러한 보안 그룹을 생성합니다.
   - `simplejspdb`라는 데이터베이스를 생성합니다.
   - 마스터 사용자 이름 및 암호에는 Tomcat 오류를 일으킬 수 있는 & 또는 기타 문자가 포함되면 안 됩니다.

     특히, 시작 중에 Tomcat은 마스터 암호 및 사용자 이름이 포함된 XML 파일인 웹 앱 컨텍스트 파일을 구문 분석해야 합니다. 문자열 중 하나에 & 문자가 포함되어 있으면 XML 구문 분석기는 이 문자를 형식이 잘못된 XML 엔터티로 취급하여 구문 분석 예외가 발생해 Tomcat이 시작되지 않습니다. 웹 앱 컨텍스트 파일에 대한 자세한 정보는 tomcat::context 단원을 참조하세요.
   - MySQL 드라이버를 Java 앱 서버 계층에 추가합니다.
   - 스택에 RDS 인스턴스를 등록합니다.

   AWS OpsWorks Stacks에서 Amazon RDS 인스턴스를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요Amazon RDS 서비스 계층.
3. 예제 코드를 `simplejspdb.jsp` 파일에 복사하고 이 파일을 `simplejspdb` 폴더에 저장한 다음 이 폴더의 `.zip` 아카이브를 생성합니다. 이름은 임의적이므로, 원하는 파일 또는 폴더 이름은 어떤 것이든 사용할 수 있습니다. 또한 gzip, bzip2 또는 tarball을 비롯하여 다른 아카이브 유형을 사용할 수도 있습니다. 여러 JSP 페이지를 배포하려면 배포하려는 페이지를 동일한 아카이브에 저

장합니다. 다른 리포지토리 유형에서 앱을 배포하는 방법은 애플리케이션 소스 단원을 참조하세요.

4. 이 아카이브를 Amazon S3 버킷에 업로드하고 이 파일을 퍼블릭으로 설정합니다. 나중에 사용하기 위해 파일의 URL를 복사해 둡니다. 버킷 생성 및 파일 업로드 방법에 대한 자세한 내용은 Amazon Simple Storage Service 시작하기를 참조하세요.
5. 스택에 앱을 추가하고 다음 설정을 지정합니다.

   - 명칭 – `SimpleJSPDB`
   - 앱 유형 – `Java`
   - 데이터 소스 유형 - OpsWorks(MySQL 인스턴스의 경우) 또는 RDS(Amazon RDS 인스턴스의 경우).
   - 데이터베이스 인스턴스 - 앞서 생성한 MySQL 인스턴스(이름: db-master1(mysql)) 또는 Amazon RDS 인스턴스(이름: ***`DB_instance_name`*** (mysql)).
   - 데이터베이스 이름 – `simplejspdb`.
   - 리포지토리 유형 – `Http Archive`
   - 리포지토리 URL - 아카이브 파일의 Amazon S3 URL.

   나머지 설정에 대해서는 기본값을 사용하고 [앱 추가]를 클릭하여 앱을 생성합니다.
6. 스택 구성 속성에 다음 사용자 지정 JSON 속성을 추가합니다. 여기서 simplejspdb는 앱의 짧은 이름입니다.

```
{
  "opsworks_java": {
    "datasources": {
      "simplejspdb": "jdbc/mydb"
    }
  }
}
```

   AWS OpsWorks Stacks는이 매핑을 사용하여 필요한 데이터베이스 정보가 포함된 컨텍스트 파일을 생성합니다.

   스택 구성 속성에 사용자 지정 JSON 속성을 추가하는 방법은 사용자 지정 JSON 사용 단원을 참조하세요.
7. 앱을 Java 앱 서버 인스턴스에 배포합니다.

이제 앱의 URL을 사용하여 앱을 볼 수 있습니다. URL 생성 방법에 대한 설명은 JSP 앱 배포를 참조하세요.

예제의 URL은 `http://192.0.2.0/simplejspdb/simplejspdb.jsp`와 비슷합니다.

> **Note**
>
> `datasources` 속성에는 여러 속성이 포함될 수 있습니다. 각 속성은 앱의 짧은 이름으로 명명되며, 논리적 이름의 적절한 사용자 할당 부분으로 설정됩니다. 앱이 여러 개인 경우에는 별도의 논리적 이름을 사용할 수 있으며, 다음과 비슷한 사용자 지정 JSON이 필요합니다.
>
> ```
> {
>   "opsworks_java": {
>     "datasources": {
>       "myjavaapp": "jdbc/myappdb",
>       "simplejsp": "jdbc/myjspdb",
>       ...
>     }
>   }
> }
> ```

Node.js 앱 서버 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Node.js 앱 서버 계층은 Node.js 애플리케이션 서버로 작동하는 인스턴스에 대한 청사진을 제공하는 AWS OpsWorks Stacks 계층입니다. AWS OpsWorks Stacks는 Express도 설치하므로 계층의 인스턴스는 표준 애플리케이션과 Express 애플리케이션을 모두 지원합니다.

설치: Node.js가 `/usr/local/bin/node`에 설치됩니다.

[계층 추가] 페이지는 다음 구성 옵션을 제공합니다.

Node.js 버전

현재 지원되는 버전의 목록은 AWS OpsWorks 스택 운영 체제 단원을 참조하세요.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다.

> **Important**
>
> Node.js 애플리케이션이 SSL을 사용하는 경우, 가능하다면 SSLv3를 비활성화하여 CVE-2015-8027에 설명된 취약성을 해결하는 것이 좋습니다. 그러려면 [Node.js 버전]을 `0.12.9`로 설정해야 합니다.

Node.js 앱 배포

AWS OpsWorks Stacks를 위한 간단한 Node.js 애플리케이션을 구현하여 스택에 배포하는 방법의 자세한 안내는 첫 번째 Node.js 스택 생성 단원을 참조하세요. 일반적으로 AWS OpsWorks Stacks용 Node.js 애플리케이션이 충족해야 하는 조건은 다음과 같습니다.

- 메인 파일의 이름은 `server.js`여야 하며, 배포된 애플리케이션이 루트 디렉터리에 상주해야 합니다.
- Express 앱은 애플리케이션의 루트 디렉터리에 `package.json` 파일을 포함해야 합니다.
- 기본적으로 이 애플리케이션은 포트 80(HTTP) 또는 포트 443(HTTPS)에서 수신해야 합니다.

다른 포트에서도 수신이 가능하지만 Node.js 앱 서버 계층의 내장 보안 그룹인 AWS-OpsWorks-nodejs-App-Server는 포트 80, 443, 22(SSH)로의 인바운드 사용자 트래픽을 허용합니다. 다른 포트로의 인바운드 사용자 트래픽을 허용하려면 적절한 인바운드 규칙을 사용하여 보안 그룹을 생성하고 Node.js 앱 서버 계층에 할당합니다. 내장 보안 그룹을 편집하여 인바운드 규칙을 수정하지 마십시오. 스택을 생성할 때마다 AWS OpsWorks Stacks는 기본 제공 보안 그룹을 표준 설정으로 덮어쓰므로 변경 사항이 손실됩니다.

> **Note**
>
> AWS OpsWorks Stacks는 PORT 환경 변수를 80(기본값) 또는 443(SSL을 활성화한 경우)으로 설정하므로 다음 코드를 사용하여 요청을 수신할 수 있습니다.
>
> ```
> app.listen(process.env.PORT);
> ```

SSL을 지원하도록 Node.js 앱을 구성하는 경우 키와 인증서를 지정해야 합니다. AWS OpsWorks Stacks는 다음과 같이 각 애플리케이션 서버 인스턴스의 데이터를 `/srv/www/app_shortname/shared/config` 디렉터리에 별도의 파일로 저장합니다.

- `ssl.crt` – SSL 인증서입니다.
- `ssl.key` – SSL 키입니다.
- `ssl.ca` – 체인 인증서(지정한 경우).

애플리케이션은 이러한 파일에서 SSL 키와 인증서를 가져올 수 있습니다.

PHP 앱 서버 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

PHP 앱 서버 계층은 PHP 애플리케이션 서버로 작동하는 인스턴스에 대한 청사진을 제공하는 AWS OpsWorks Stacks 계층입니다. PHP 앱 서버 계층은 mod_php가 있는 Apache2를 기반으로 하며 표준 구성 옵션은 없습니다. PHP 및 Apache 버전은 계층의 인스턴스에 대해 지정하는 운영 체제에 따라 달라집니다.

운영 체제	PHP 버전	Apache 버전
Amazon Linux 2018.03	5.3	2.2
Amazon Linux 2017.09	5.3	2.2
Amazon Linux 2017.03	5.3	2.2
Amazon Linux 2016.09	5.3	2.2
Amazon Linux 2016.03	5.3	2.2
Amazon Linux 2015.09	5.3	2.2
Amazon Linux 2015.03	5.3	2.2
Amazon Linux 2014.09	5.3	2.2
Ubuntu 14.04 LTS	5.5	2.4

Installation: AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 기본 위치에 Apache2 및 mod_php를 설치합니다. 설치에 대한 자세한 정보는 Apache를 참조하세요.

[계층 추가] 페이지는 다음 구성 옵션을 제공합니다.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다.

사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 일부 Apache 구성 설정을 수정할 수 있습니다. 자세한 내용은 속성 재정의 섹션을 참조하세요. 재정의할 수 있는 Apache 속성의 목록은 apache2 속성 단원을 참조하세요.

PHP 앱을 백엔드 데이터베이스에 연결하는 방법 등 PHP 앱을 배포하는 방법의 예제는 Chef 11 Linux 스택 시작하기를 참조하세요.

> **Important**
>
> PHP 애플리케이션이 SSL을 사용하는 경우, 가능하다면 SSLv3를 비활성화하여 CVE-2014-3566에 설명된 취약성을 해결하는 것이 좋습니다. 그러려면 Apache 서버의 `SSLProtocol` 파일에서 `ssl.conf`설정을 수정해야 합니다. 이 설정을 수정하는 방법에 대한 자세한 정보는 Apache 서버에서 SSLv3 비활성화를 참조하세요.

Rails 앱 서버 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Rails 앱 서버 계층은 Rails 애플리케이션 서버로 작동하는 인스턴스에 대한 블루프린트를 제공하는 AWS OpsWorks Stacks 계층입니다.

Installation: AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 기본 위치에 서버 패키지를 설치합니다. Apache/Passenger 설치에 대한 자세한 정보는 Phusion Passenger를 참조하세

요. 로깅에 대한 자세한 정보는 로그 파일 단원을 참조하세요. Nginx/Unicorn 설치에 대한 자세한 정보는 Unicorn 단원을 참조하세요.

[계층 추가] 페이지는 모두가 선택 사항인 다음 구성 옵션을 제공합니다.

Ruby 버전

애플리케이션이 사용할 Ruby 버전. 기본값은 2.3입니다.

`[:opsworks][:ruby_version]` 속성을 재정의하면 선호하는 Ruby 버전을 지정할 수도 있습니다.

> **Note**
>
> AWS OpsWorks Stacks는 레시피와 인스턴스 에이전트에서 사용할 별도의 Ruby 패키지를 설치합니다. 자세한 내용은 Ruby 버전 단원을 참조하십시오.

Rails 스택

기본 Rails 스택은 Phusion Passenger가 있는 Apache2입니다. Nginx를 유니콘과 함께 사용할 수도 있습니다.

> **Note**
>
> Nginx와 Unicorn을 사용하는 경우, 다음 예제에서처럼 앱의 Gemfile에 unicorn gem을 추가해야 합니다.
>
> ```
> source 'https://rubygems.org'
> gem 'rails', '3.2.15'
> ...
> # Use unicorn as the app server
> gem 'unicorn'
> ...
> ```

Passenger 버전

Apache2/Passenger를 지정한 경우, Passenger 버전을 지정해야 합니다. 기본값은 5.0.28입니다.

Rubygems 버전

기본 Rubygems 버전은 2.5.1입니다.

Bundler 설치 및 관리

Bundler 설치 및 관리 여부를 선택할 수 있도록 해 줍니다. 기본 값은 [예]입니다.

Bundler 버전

기본 Bundler 버전은 1.12.5입니다.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다.

사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 일부 구성 설정을 수정할 수 있습니다. 자세한 내용은 속성 재정의 섹션을 참조하세요. 재정의할 수 있는 Apache, Nginx, Phusion Passenger 및 Unicorn 속성 목록은 내장 쿡북 속성를 참조하세요.

> **Important**
>
> Ruby on Rails 애플리케이션이 SSL을 사용하는 경우, 가능하다면 SSLv3를 비활성화하여 CVE-2014-3566에 설명된 취약성을 해결하는 것이 좋습니다. 자세한 내용은 Rails 서버에 대해 SSLv3 비활성화 섹션을 참조하세요.

주제

- Rails 서버에 대해 SSLv3 비활성화
- 데이터베이스에 연결
- Ruby on Rails 앱 배포

Rails 서버에 대해 SSLv3 비활성화

Rails 서버에 대해 SSLv3를 비활성화하려면 계층의 Ruby Version(Ruby 버전) 설정을 2.1로 업데이트합니다. 그러면 애플리케이션이 사용하는 버전으로 Ruby 2.1.4 이상이 설치됩니다.

- 계층의 Ruby 버전 설정을 2.1 이상으로 업데이트합니다.
- Rails 스택의 구성 파일을 다음과 같이 업데이트합니다.

Phusion Passenger 포함 Apache

Apache 서버 `SSLProtocol` 파일의 `ssl.conf` 설정을 Apache 서버에서 SSLv3 비활성화의 설명에 따라 업데이트합니다.

Unicorn 포함 Nginx

Nginx 서버의 `ssl_protocols` 파일에 명시적인 `nginx.conf` 명령을 추가합니다. SSLv3를 비활성화하려면 Rails 앱 서버 계층의 설정 레시피가 `nginx.conf`을 생성하는 데 사용하는 내장 nginx 쿡북의 `nginx.conf.erb` 템플릿 파일을 재정의하고 다음 지시문을 추가합니다.

```
ssl_protocols TLSv1.2;
```

`nginx.conf`를 구성하는 방법에 대한 자세한 정보는 HTTPS 서버 구성 단원을 참조하세요. 내장 템플릿을 재정의하는 방법에 대한 자세한 정보는 사용자 지정 템플릿 사용 를 참조하세요.

데이터베이스에 연결

앱을 배포하면 AWS OpsWorks Stacks는 앱 `deploy` 속성의 정보를 사용하여 새 `database.yml` 파일을 생성합니다. MySQL 또는 Amazon RDS 인스턴스를 앱에 연결하면 AWS OpsWorks Stacks는 속성에 연결 정보를 추가하여에 올바른 연결 데이터가 `deploy` `database.yml` 자동으로 포함됩니다.

앱에 연결된 데이터베이스가 없는 경우 기본적으로 AWS OpsWorks Stacks는 `deploy` 속성에 연결 정보를 추가하지 않으며를 생성하지 않습니다`database.yml`. 다른 데이터베이스를 사용하려는 경우, 사용자 지정 JSON을 사용하여 앱의 `deploy` 속성에 연결 정보와 함께 데이터베이스 속성을 추가할 수 있습니다. 속성은 모두 아래에 있습니다`["deploy"]["`*`appshortname`*`"]["database"]`. 여기서 *`appshortname`*은 AWS OpsWorks Stacks가 앱 이름에서 생성하는 앱의 짧은 이름입니다. 사용자 지정 JSON에서 지정하는 값은 기본 설정을 재정의합니다. 자세한 내용은 앱 추가 단원을 참조하십시오.

AWS OpsWorks Stacks는 다음 `[:...][:database]` 속성 값을에 통합합니다`database.yml`. 필수 속성은 특정 데이터베이스에 따라 다르지만 `host` 속성이 있어야 합니다. 그렇지 않으면 AWS OpsWorks Stacks가를 생성하지 않습니다`database.yml`.

- `[:adapter]` (String) - `mysql`과 같은 데이터베이스 어댑터.
- `[:database]`(문자열) - 데이터베이스 이름.

- [:encoding](문자열) - 일반적으로 utf8로 설정되는 인코딩.
- [:host](문자열) - railsexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com과 같은 호스트 URL.
- [:reconnect](Boolean) - 연결이 더 이상 존재하지 않는 경우, 애플리케이션이 다시 연결해야 하는지 여부.
- [:password](문자열) - 데이터베이스 암호.
- [:port] (번호). - 데이터베이스 포트 숫자. 이 속성을 사용하여 기본 포트 번호를 재정의합니다(기본 포트 번호는 어댑터에 의해 설정).
- [:username](문자열) - 데이터베이스 사용자 이름.

다음 예제는 짧은 이름이 myapp인 앱의 사용자 지정 JSON을 보여 줍니다.

```
{
  "deploy" : {
    "myapp" : {
      "database" : {
        "adapter" : "adapter",
        "database" : "databasename",
        "host" : "host",
        "password" : "password",
        "port" : portnumber
        "reconnect" : true/false,
        "username" : "username"
      }
    }
  }
}
```

사용자 지정 JSON을 지정하는 방법에 대해서는 사용자 지정 JSON 사용 단원을 참조하세요. database.yml(database.yml.erb) 생성에 사용되는 템플릿을 보려면 내장 쿡북 리포지토리로 이동하세요.

Ruby on Rails 앱 배포

지원되는 모든 리포지토리에서 Ruby on Rails 앱을 배포할 수 있습니다. 다음은 Apache/Passenger Rails 스택을 실행하는 서버에 예제 Ruby on Rails 앱을 배포하는 방법을 보여 줍니다. 예제 코드는 퍼블릭 GitHub 리포지토리에 저장되지만 기본 절차는 지원되는 다른 리포지토리와 동일합니다. 앱을 생

성하고 배포하는 방법에 대한 자세한 정보는 앱 단원을 참조하세요. 방대한 주석이 포함된 예제 코드를 보려면 https://github.com/awslabs/opsworks-demo-rails-photo-share-app으로 이동합니다.

GitHub 리포지토리에서 Ruby on Rails 앱을 배포하려면

1. Apache/Passenger를 Rails 스택으로 하는 Rails 앱 서버 계층을 사용하여 스택을 생성하고, 해당 계층에 연중무휴 인스턴스를 추가한 다음 시작합니다.
2. 인스턴스가 온라인 상태가 되면 스택에 앱을 추가하고 다음 설정을 지정합니다.

   - 이름 - 선호하는 모든 이름을 지정할 수 있습니다. 이 예제에서는 `PhotoPoll`을 사용합니다.

     AWS OpsWorks Stacks는 표시 목적으로이 이름을 사용하고 내부용으로 짧은 이름을 생성하고 스택 구성 및 배포 속성에서 앱을 식별합니다. 예를 들어 PhotoPoll의 짧은 이름은 photopoll입니다.

   - 앱 유형 – Ruby on Rails.
   - Rails 환경 - 사용 가능한 환경은 애플리케이션에서 결정합니다.

     예제에서 사용되는 앱에는 **`development`**, **`test`** 및 **`production`**, 이렇게 세 가지 환경이 있습니다. 이 예제에서는 환경을 **`development`**로 설정합니다. 각 환경에 대한 설명은 예제 코드를 참조하세요.
   - 리포지토리 유형 - 지원되는 모든 리포지토리 유형. 이 예제에서는 `Git`를 지정합니다.
   - 리포지토리 URL - 배포할 코드를 가져와야 하는 리포지토리.

     이 예제에서는 URL을 **`git://github.com/awslabs/opsworks-demo-rails-photo-share-app`**으로 설정합니다.

   나머지 설정에 대해서는 기본값을 사용하고 [앱 추가]를 클릭하여 앱을 생성합니다.
3. 앱을 Rails 앱 서버 인스턴스에 배포합니다.
4. 배포를 마치면 인스턴스 페이지로 이동해 Rails 앱 서버 인스턴스의 퍼블릭 IP 주소를 클릭합니다. 다음과 같은 모양이어야 합니다.

정적 웹 서버 AWS OpsWorks 스택 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

정적 웹 서버 계층은 인스턴스가 클라이언트 측 AWS OpsWorks 스크립팅을 포함할 수 있는 정적 HTML 페이지를 제공할 수 있는 템플릿을 제공하는 Stacks 계층입니다. 이 계층은 Nginx를 기반으로 합니다.

설치: Nginx가 `/usr/sbin/nginx`에 설치됩니다.

[계층 추가] 페이지는 다음 구성 옵션을 제공합니다.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다.

사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 일부 Nginx 구성 설정을 수정할 수 있습니다. 자세한 내용은 속성 재정의 섹션을 참조하세요. 재정의할 수 있는 Apache 속성의 목록은 nginx 속성 단원을 참조하세요.

Important

웹 애플리케이션이 SSL을 사용하는 경우, 가능하다면 SSLv3를 비활성화하여 CVE-2014-3566에 설명된 취약성을 해결하는 것이 좋습니다.
SSLv3를 비활성화하려면 Nginx 서버의 `nginx.conf` 파일을 수정해야 합니다. 이렇게 하려면 Rails 앱 서버 계층의 설정 레시피가 `nginx.conf`을 생성하는 데 사용하는 내장 nginx 쿡북의 `nginx.conf.erb` 템플릿 파일을 재정의하고 다음 지시문을 추가하세요.

```
ssl_protocols TLSv1.2;
```

`nginx.conf`를 구성하는 방법에 대한 자세한 정보는 HTTPS 서버 구성 단원을 참조하세요. 내장 템플릿을 재정의하는 방법에 대한 자세한 정보는 사용자 지정 템플릿 사용 를 참조하세요.

ECS 클러스터 계층 참조

Note

이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

ECS 클러스터 계층은 Amazon Elastic Container Service (Amazon ECS) 클러스터를 나타내며 클러스터 관리를 단순화합니다.

짧은 이름: ecs-cluster

호환성: Amazon ECS 서비스 계층은 사용자 지정 계층과만 호환됩니다.

개방 포트: ECS 클러스터는 포트 22(SSH)에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-ECS-App-Cluster

구성: ECS 클러스터 계층을 구성하려면 다음을 지정해야 합니다.

- 컨테이너 인스턴스에 퍼블릭 IP 주소 또는 탄력적 IP 주소를 할당할지 여부
- 컨테이너 인스턴스의 인스턴스 프로파일

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ecs::설정

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version
- opsworks_ecs::configure

Deploy 레시피:

- deploy::default

- opsworks_ecs::deploy

Undeploy 레시피:

- opsworks_ecs::undeploy

Shutdown 레시피:

- opsworks_shutdown::default
- opsworks_ecs::shutdown

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 Docker를 기본 위치에 설치합니다.
- 설정 이벤트용 Chef 로그는 Amazon ECS 에이전트가 성공적으로 설치되었는지 여부를 기록합니다. 그렇지 않으면 AWS OpsWorks Stacks에서 제공하는 로그에 Amazon ECS 오류 로그 정보가 포함되지 않습니다. Amazon ECS 오류 처리 방법에 대한 자세한 내용은 Amazon ECS 문제 해결을 참조하세요.

사용자 지정 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

표준 계층이 사용자의 요구 사항에 적합하지 않은 경우 사용자 지정 계층을 생성할 수 있습니다. 스택은 여러 개의 사용자 지정 계층을 포함할 수 있습니다. 기본적으로 사용자 지정 계층은 기본 기능을 지원하는 제한된 수의 표준 레시피를 실행합니다. 사용자는 각 수명 주기 이벤트에 대해 계층의 소프트웨어를 설정 및 구성하는 등의 사용자 지정 Chef 레시피 세트를 구현함으로써 계층의 주 기능을 구현할 수 있습니다. 사용자 지정 레시피는 각 이벤트에 대한 표준 AWS OpsWorks Stacks 레시피 이후에 실행됩니다.

짧은 이름: 사용자 정의됨. 스택 내 각 사용자 지정 계층은 서로 다른 짧은 이름을 가져야 합니다.

개방 포트: 기본적으로 사용자 지정 계층은 포트 22(SSH), 80(HTTP), 443(HTTPS), 그리고 스택의 Rails 및 PHP 애플리케이션 서버 계층의 모든 포트에 대해 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-Custom-Server

호환성: 사용자 지정 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, lb, memcached, monitoring-master, nodejs-app, php-app, rails-app, web.

구성: 사용자 지정 계층을 구성하려면 다음을 지정해야 합니다.

- 계층의 이름
- 계층의 짧은 이름(Chef 레시피에서 계층을 식별하며 a-z 및 숫자만 사용)

Linux 스택의 경우 사용자 지정 계층은 다음 레시피를 사용합니다.

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 레시피:

- deploy::default

Shutdown 레시피:

- opsworks_shutdown::default

기타 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 다음 계층도 지원합니다.

주제

- Ganglia 계층 참조
- Memcached 계층 참조

Ganglia 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Ganglia 계층은 인스턴스 측정치의 저장 및 시각화를 관리하는 분산형 모니터링 시스템인 Ganglia를 지원합니다. 이 계층은 계층적 인스턴스 토폴로지에서 사용하도록 설계되어 인스턴스 그룹에 특히 유용합니다. Ganglia에는 두 개의 기본 구성 요소가 있습니다.

- 저 오버헤드 클라이언트 - 스택 내 각 인스턴스에 설치되고 측정치를 마스터로 전송합니다.
- 마스터 - 클라이언트로부터 측정치를 수집하여 Amazon EBS 볼륨에 저장합니다. 또한 마스터는 측정치를 웹 페이지에 표시합니다.

AWS OpsWorks Stacks에는 관리하는 각 인스턴스에 Ganglia 모니터링 에이전트가 있습니다. Ganglia 계층을 스택에 추가하여 시작하면 각 인스턴스의 Ganglia 에이전트가 Ganglia 인스턴스로 측정치를 보고합니다. Ganglia를 사용하려면 스택에 하나의 인스턴스를 포함하는 Ganglia 계층을 추가합니다. 데이터는 마스터의 IP 주소에서 Ganglia 백엔드에 로그인하여 액세스합니다. Chef 레시피를 작성하면 추가 측정치 정의를 제공할 수 있습니다.

짧은 이름: monitoring-master

호환성: Ganglia 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, memcached, php-app, rails-app.

개방 포트: 로드 밸런서는 포트 22(SSH), 80(HTTP) 및 443(HTTPS)에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 예. 위치: `/vol/ganglia`

기본 보안 그룹: AWS-OpsWorks-Monitoring-Master-Server

구성: Ganglia 계층을 구성하려면 다음을 지정해야 합니다.

- 모니터링 그래프에 대한 액세스를 제공하는 URI. 기본값은 http://*DNSName*/ganglia입니다. 여기서 *DNSName*은 Ganglia 인스턴스의 DNS 이름입니다.
- 모니터링 통계에 대한 액세스를 제어하는 사용자 이름 및 암호.

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys

- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ganglia::server

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_ganglia::configure-server

Deploy 레시피:

- deploy::default
- opsworks_ganglia::configure-server
- opsworks_ganglia::deploy

Shutdown 레시피:

- opsworks_shutdown::default
- apache2::stop

설치:

- Ganglia 클라이언트는 `/etc/ganglia` 아래에 설치됩니다.
- Ganglia 웹 프런트 엔드는 `/usr/share/ganglia-webfrontend` 아래에 설치됩니다.
- Ganglia logtailer는 `/usr/share/ganglia-logtailer` 아래에 설치됩니다.

Memcached 계층 참조

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Linux 기반 스택에서만 사용할 수 있습니다.

Memcached는 임의 데이터용 분산형 메모리 캐싱 시스템입니다. 이 시스템은 문자열 및 객체를 RAM에서 키와 값으로 캐싱하여 외부 데이터 원본을 읽어야 하는 횟수를 줄이는 방식으로 웹 사이트 속도를 높여줍니다.

스택에서 Memcached를 사용하려면 Memcached 계층을 생성하고 Memcached 서버로 기능하는 인스턴스를 하나 이상 추가합니다. 인스턴스는 자동으로 Memcached를 설치하며, 스택의 다른 인스턴스는 Memcached 서버에 액세스하고 사용할 수 있습니다. Rails 앱 서버 계층을 사용하는 경우 AWS OpsWorks Stacks는 계층에 있는 각 인스턴스의 구성 디렉터리에 `memcached.yml` 구성 파일을 자동으로 배치합니다. 이 파일에서 Memcached 서버 및 포트 번호를 가져올 수 있습니다.

짧은 이름: memcached

호환성: Memcached 계층은 다음 계층과 호환됩니다. 사용자 지정, db-master, lb, monitoring-master, nodejs-app, php-app, rails-app, web.

개방 포트: Memcached 계층은 포트 22(SSH) 그리고 스택의 웹 서버, 사용자 지정 서버 및 Rails, PHP 및 Node.js 애플리케이션 서버의 모든 포트에 대한 퍼블릭 액세스를 허용합니다.

탄력적 IP 주소 자동 할당: 기본적으로 Off

기본 EBS 볼륨: 없음

기본 보안 그룹: AWS-OpsWorks-Memcached-Server

Memcached 계층을 구성하려면 캐시 크기를 MB 단위로 지정해야 합니다.

설정 레시피:

- opsworks_initial_설정
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- memcached

Configure 레시피:

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 레시피:

- deploy::default

Shutdown 레시피:

- opsworks_shutdown::default
- memcached::stop

설치:

- AWS OpsWorks Stacks는 인스턴스의 패키지 설치 관리자를 사용하여 Memcached와 해당 로그 파일을 기본 위치에 설치합니다.

기타 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이들 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

AWS OpsWorks Stacks는 Ganglia 및 Memcached 계층도 지원합니다.

주제

- Ganglia 계층
- Memcached

Ganglia 계층

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

AWS OpsWorks Stacks는 모든 인스턴스 및 볼륨 지표를 Amazon CloudWatch로 전송하므로 쉽게 그래프를 보고 경보를 설정하여 문제를 해결하고 리소스 상태에 따라 자동화된 조치를 취할 수 있습니다.

선택한 지표 저장과 같은 추가 애플리케이션 모니터링 옵션에 Ganglia AWS OpsWorks Stacks 계층을 사용할 수도 있습니다.

Ganglia 계층은 Ganglia 분산형 모니터링을 사용하여 스택을 모니터링하는 인스턴스의 청사진입니다. 스택은 Ganglia 인스턴스를 하나만 포함하는 것이 일반적입니다. Ganglia 계층에는 다음의 선택적 구성 설정이 포함됩니다.

Ganglia URL

통계의 URL 경로. 전체 URL은 http://*DNSNameURLPath*이며, 여기서 *DNSName*은 연결된 인스턴스의 DNS 이름입니다. 기본 *URLPath* 값은 "/ganglia"이며, 다음과 같은 값에 해당합니다. http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia.

Ganglia 사용자 이름

통계 웹 페이지의 사용자 이름. 페이지를 보려면 사용자 이름을 제공해야 합니다. 기본값은 “OpsWorks”입니다.

Ganglia 암호

통계 웹 페이지에 대한 액세스를 제어하는 암호. 페이지를 보려면 암호를 제공해야 합니다. 기본값은 무작위로 생성된 문자열입니다.

> **Note**
>
> 나중에 사용할 수 있도록 암호를 기록합니다. AWS OpsWorks Stacks에서는 계층을 생성한 후 암호를 볼 수 없습니다. 하지만 계층의 [편집] 페이지로 이동하고 [암호 업데이트]를 클릭하면 암호를 업데이트할 수 있습니다.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

Elastic Load Balancer

Elastic Load Balancing 로드 밸런서를 어떤 계층의 인스턴스에도 연결할 수 있습니다.

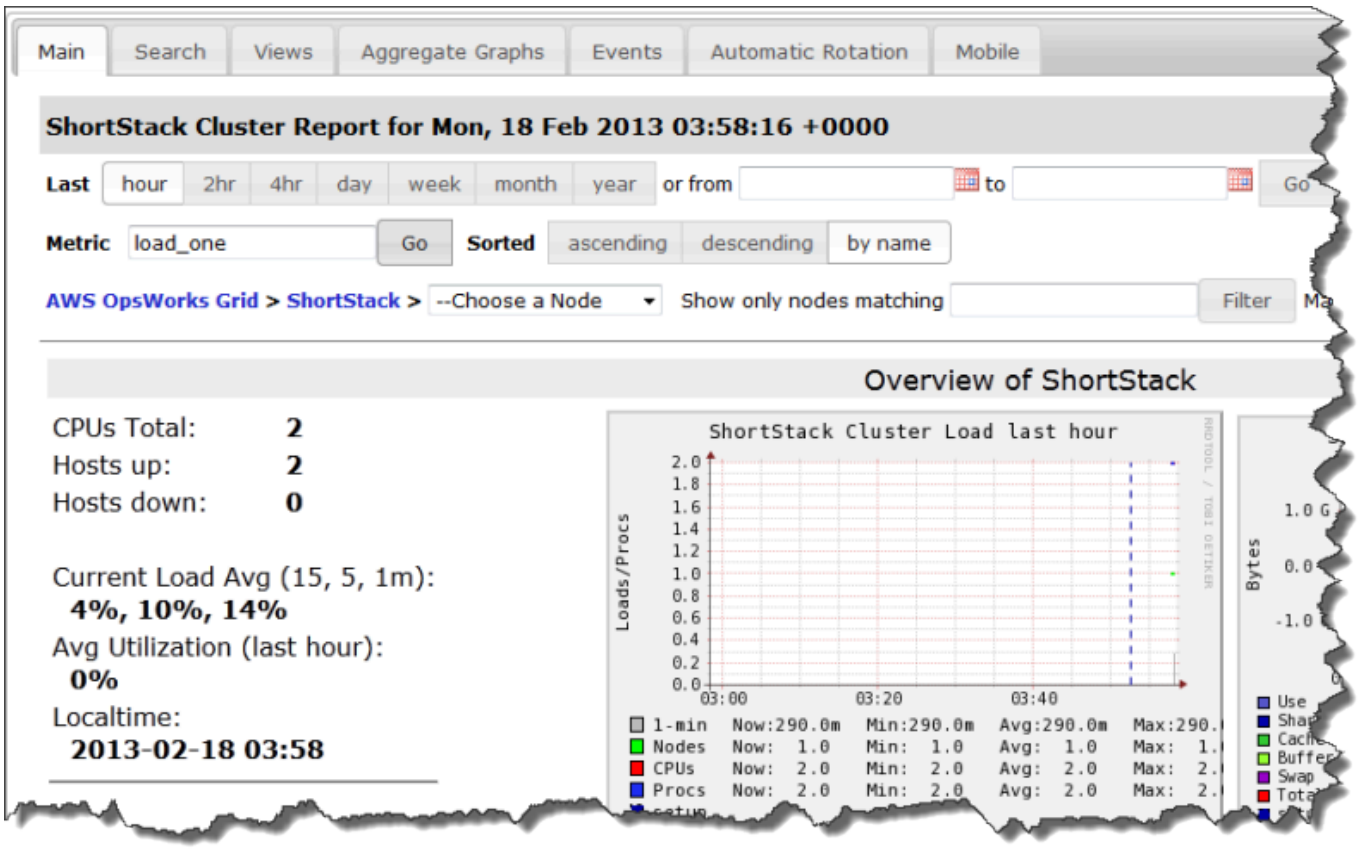
> **Important**
>
> 스택에 Ganglia 계층이 포함된 경우 이 계층이 CVE-2014-3566에 설명된 취약성을 해결하도록 가능하면 SSLv3를 비활성화하는 것이 좋습니다. 그러려면 Apache 서버의 `ssl.conf.erb` 템플릿을 재정의하여 `SSLProtocol` 설정을 수정해야 합니다. 자세한 내용은 Apache 서버에서 SSLv3 비활성화 섹션을 참조하세요.

Ganglia 통계 보기

AWS OpsWorks Stacks 레시피는 모든 인스턴스에 로우 오버헤드 Ganglia 클라이언트를 설치합니다. 스택에 Ganglia 계층이 포함된 경우 Ganglia 클라이언트는 인스턴스가 온라인 상태로 전환되는 즉시 자동으로 Ganglia에 보고를 시작합니다. Ganglia는 클라이언트 데이터를 사용하여 다양한 통계를 계산하고 통계 웹 페이지에 그래프로 결과를 표시합니다.

Ganglia 통계를 보려면

1. 계층 페이지에서 Ganglia를 클릭하여 계층의 세부정보 페이지를 엽니다.
2. 탐색 창에서 인스턴스를 클릭합니다. Ganglia에서 인스턴스 이름을 클릭합니다.
3. 인스턴스의 퍼블릭 DNS 이름을 복사합니다.
4. DNS 이름을 사용하여 통계 URL을 구성합니다. 이 URL은 http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia와 유사합니다.
5. 브라우저에 전체 URL을 붙여 넣고 이 페이지로 이동한 다음 Ganglia 사용자 이름 및 암호를 입력하여 페이지를 표시합니다. 예를 들면 다음과 같습니다.



Memcached

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이 계층은 Chef 11 및 이전 Linux 기반 스택에서만 사용할 수 있습니다.

Memcached 계층은 임의의 데이터를 위한 분산 메모리 캐싱 시스템인 Memcached 서버로 작동하는 인스턴스에 대한 청사진을 제공하는 AWS OpsWorks Stacks 계층입니다. Memcached 계층에는 다음 구성 설정이 포함됩니다.

허용된 메모리(MB)

(선택 사항) 계층 내 각 인스턴스의 캐시 메모리 크기(MB). 기본값은 512MB입니다.

사용자 지정 보안 그룹

이 설정은 내장 AWS OpsWorks Stacks 보안 그룹을 계층과 자동으로 연결하지 않도록 선택한 경우에 나타납니다. 계층에 어떤 보안 그룹을 연결할지 지정해야 합니다. 자세한 내용은 새 스택 생성 섹션을 참조하세요.

쿡북 구성 요소

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

기본적으로 쿡북은 다음의 기본 구성 요소를 포함합니다.

- 속성 파일은 레시피 및 템플릿에서 사용할 값을 나타내는 속성 세트를 포함합니다.
- 템플릿 파일은 레시피가 구성 파일과 같은 다른 파일을 생성하는 데 사용하는 템플릿입니다.

  일반적으로 템플릿 파일을 사용하면 구성 파일을 다시 작성하는 대신 쿡북을 건드리지 않고도 속성을 재정의하여 구성 파일을 수정할 수 있습니다. 표준적인 방법은 인스턴스에서 사소하더라도 구성 파일 변경이 예상될 때마다 템플릿 파일을 사용하는 것입니다.
- 레시피 파일은 폴더 생성 및 구성, 패키지 설치 및 구성, 서비스 시작 등 시스템을 구성하는 데 필요한 모든 것을 정의하는 Ruby 애플리케이션입니다.

쿡북이 세 구성 요소를 모두 포함할 필요는 없습니다. 보다 간단한 사용자 지정 방법에서는 속성 또는 템플릿 파일만 필요합니다. 또한 쿡북은 선택적으로 정의 또는 사양과 같은 다른 파일 유형을 포함할 수 있습니다.

이 섹션에서는 표준 쿡북 구성 요소 3개를 설명합니다. 자세한 내용, 특히 레시피를 구현하는 방법은 Opscode를 참조하세요.

주제

- 속성
- 템플릿
- 레시피

속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피 및 템플릿은 구성 설정과 같은 다양한 값을 기반으로 합니다. 이러한 값을 레시피 또는 템플릿에서 직접 하드코딩하지 않고 속성이 각 값을 나타내는 속성 파일을 생성할 수 있습니다. 그런 다음 레시피 또는 템플릿에 명시적 값 대신 속성을 사용합니다. 속성을 사용하는 장점은 쿡북을 손대지 않고 해당 값을 재정의할 수 있다는 점입니다. 이러한 이유 때문에 다음 유형의 값은 항상 속성을 사용하여 정의해야 합니다.

- 사용자 이름 같이 스택마다 또는 시간 경과에 따라 바뀔 수 있는 값.

  이러한 값을 하드코딩할 경우 값을 바꿔야 할 때마다 레시피 또는 템플릿을 변경해야 합니다. 속성을 사용하여 이러한 값을 정의하면 동일한 쿡북을 모든 스택에서 사용하고 해당 속성만 재정의할 수 있습니다.
- 암호 또는 보안 키 같은 중요한 값.

  쿡북에 중요한 값을 명시적으로 삽입한다면 노출 위험이 증가할 수 있습니다. 그 대신 더미 값으로 속성을 정의한 후 재정의하여 실제 값을 설정하세요. 이러한 속성을 재정의하는 최상의 방법은 사용자 지정 JSON입니다. 자세한 내용은 사용자 지정 JSON 사용 섹션을 참조하세요.

속성과 속성 재정의 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

다음은 예제 속성 파일의 한 부분입니다.

```
...
default["apache"]["listen_ports"] = [ '80','443' ]
default["apache"]["contact"] = 'ops@example.com'
default["apache"]["timeout"] = 120
default["apache"]["keepalive"] = 'Off'
default["apache"]["keepaliverequests"] = 100
default["apache"]["keepalivetimeout"] = 3
default["apache"]["prefork"]["startservers"] = 16
default["apache"]["prefork"]["minspareservers"] = 16
default["apache"]["prefork"]["maxspareservers"] = 32
default["apache"]["prefork"]["serverlimit"] = 400
default["apache"]["prefork"]["maxclients"] = 400
default["apache"]["prefork"]["maxrequestsperchild"] = 10000
...
```

AWS OpsWorks Stacks는 다음 구문을 사용하여 속성을 정의합니다.

```
node.type["attribute"]["subattribute"]["..."]=value
```

다음과 같이 콜론(:)을 사용할 수도 있습니다.

```
node.type[:attribute][:subattribute][:...]=value
```

속성 정의에는 다음의 구성 요소가 있습니다.

node.

node. 접두사는 예제에 나와 있는 것과 같이 선택 항목이며 일반적으로 생략됩니다.

type

유형은 속성을 재정의할 수 있는지 여부를 제어합니다. AWS OpsWorks Stacks 속성은 일반적으로 다음 유형 중 하나를 사용합니다.

- default는 속성 재정의를 허용하기 때문에 가장 자주 사용되는 유형입니다.
- normal는 표준 AWS OpsWorks Stacks 속성 값 중 하나를 재정의하는 속성을 정의합니다.

> **Note**
>
> Chef는 AWS OpsWorks Stacks에는 필요하지 않지만 프로젝트에 유용할 수 있는 추가 유형을 지원합니다. 자세한 정보는 속성 정보를 참조하세요.

attribute name

속성 이름은 표준 Chef 노드 구문 `[:attribute][:subattribute][...]`을 사용합니다. 속성에 원하는 이름을 사용할 수 있습니다. 그러나 속성 재정의 단원에서 설명한 것처럼 사용자 지정 쿡북 속성은 스택 구성 및 배포 속성과 Chef의 Ohai 도구의 속성과 함께 인스턴스의 노드 객체로 병합됩니다. port 또는 user와 같이 흔히 사용되는 구성 이름은 여러 쿡북에서 나타날 수 있습니다.

이름 충돌을 방지하기 위한 규칙은 예제에 나와 있듯이 2개 이상의 요소를 포함하여 한정된 속성 이름을 생성하는 것입니다. 첫 번째 요소는 고유해야 하며 일반적으로 Apache 같은 제품 이름을 기반으로 합니다. 그 뒤에는 특정 값을 식별하는 하나 이상의 하위 속성이 옵니다. 예: `[:user]` 또는 `[:port]`. 프로젝트에 따라 원하는 만큼 여러 개의 하위 속성을 사용할 수 있습니다.

value

속성은 다음 유형의 값으로 설정될 수 있습니다.

- 문자열, 예: `default[:apache][:keepalive] = 'Off'`.
- 숫자(따옴표 없음), 예: `default[:apache][:timeout] = 120`.
- 부울 값 `true` 또는 `false`(따옴표 없음).
- 값 목록, 예: `default[:apache][:listen_ports] = [ '80','443' ]`

속성 파일은 Ruby 애플리케이션이므로 노드 구문 및 논리 연산자도 사용할 수 있고 다른 속성을 기반으로 값을 할당할 수도 있습니다. 속성을 정의하는 방법에 대한 자세한 정보는 속성 정보를 참조하세요. 작동 속성 파일의 예는 AWS OpsWorks https://github.com/aws/opsworks-cookbooks Stacks 내장 쿡북을 참조하세요.

템플릿

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

구성 파일을 변경하고 적절한 디렉터리에 배치하여 여러 패키지를 구성합니다. 쿡북에 구성 파일을 포함하고 적절한 디렉터리에 복사할 수 있습니다. 하지만 보다 유연한 접근 방법은 레시피가 템플릿에서 구성 파일을 생성하게 하는 것입니다. 템플릿의 장점 한 가지는 속성을 사용하여 템플릿의 값을 정의할 수 있다는 것입니다. 그러면 예를 들어 사용자 정의 JSON을 사용하여 해당 속성 값을 재정의하면 쿡북을 손대지 않고 구성 파일을 수정할 수 있습니다.

템플릿은 연결된 파일과 콘텐츠 및 구조가 기본적으로 동일합니다. 다음은 예제 파일 `httpd.conf`입니다.

```
ServerRoot "<%= node[:apache][:dir] %>"
<% if node[:platform] == "debian" || node[:platform] == "ubuntu" -%>
  LockFile /var/lock/apache2/accept.lock
<% else -%>
   LockFile logs/accept.lock
<% end -%>
PidFile <%= node[:apache][:pid_file] %>
Timeout <%= node[:apache][:timeout] %>
KeepAlive <%= node[:apache][:keepalive] %>
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
<IfModule mpm_prefork_module>
    StartServers          <%= node[:apache][:prefork][:startservers] %>
    MinSpareServers       <%= node[:apache][:prefork][:minspareservers] %>
    MaxSpareServers       <%= node[:apache][:prefork][:maxspareservers] %>
    ServerLimit           <%= node[:apache][:prefork][:serverlimit] %>
    MaxClients            <%= node[:apache][:prefork][:maxclients] %>
    MaxRequestsPerChild   <%= node[:apache][:prefork][:maxrequestsperchild] %>
</IfModule>
...
```

다음 예제는 Ubuntu 인스턴스용으로 생성된 `httpd.conf` 파일입니다.

```
ServerRoot "/etc/httpd"
LockFile logs/accept.lock
PidFile /var/run/httpd/httpd.pid
```

```
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 3
<IfModule mpm_prefork_module>
    StartServers          16
    MinSpareServers       16
    MaxSpareServers       32
    ServerLimit           400
    MaxClients            400
    MaxRequestsPerChild   10000
</IfModule>
...
```

템플릿의 텍스트는 대부분 템플릿에서 `httpd.conf` 파일로 간단히 복사됩니다. 하지만 `<%= ... %>` 콘텐츠는 다음과 같이 처리됩니다.

- Chef가 `<%= node[:attribute][:sub_attribute][:...]%>`를 속성 값으로 대체합니다.

  예를 들어 `StartServers <%= node[:apache][:prefork][:startservers] %>`는 `httpd.conf`에서 `StartServers 16`이 됩니다.
- `<%if-%>, <%else-%>, and <%end-%>`를 사용하여 조건부로 값을 선택할 수 있습니다.

  예제는 `accept.lock`의 파일 경로를 플랫폼에 따라 다르게 설정합니다.

> **Note**
>
> 쿡북의 속성 파일 내 속성만 사용할 수 있는 것은 아닙니다. 인스턴스의 노드 객체에 포함된 속성은 모두 사용할 수 있습니다. 예를 들어, Ohai라는 Chef 도구로 생성하여 노드 객체에 통합하기도 합니다. 속성에 대한 자세한 정보는 속성 재정의를 참조하세요.

Ruby 코드를 통합하는 방법을 포함해 템플릿에 대한 자세한 정보는 템플릿 정보를 참조하세요.

레시피

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그

레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피는 시스템 구성을 정의하는 Ruby 애플리케이션입니다. 패키지를 설치하고, 템플릿에서 구성 파일을 생성하고, shell 명령을 실행하고, 파일 및 디렉터리를 생성하는 등의 작업을 수행합니다. 일반적으로 인스턴스에서 수명 주기 이벤트가 발생할 때 AWS OpsWorks Stacks가 레시피를 자동으로 실행하도록 하지만 Execute Recipes 스택 명령을 사용하여 언제든지 명시적으로 실행할 수도 있습니다. 자세한 정보는 About Recipes를 참조하세요.

대개 레시피는 일련의 리소스로 구성되는데, 각 리소스는 시스템 특정 측면에 대해 원하는 값을 나타냅니다. 각 리소스는 원하는 상태를 정의하는 속성 세트를 포함하며 수행될 작업을 지정합니다. Chef는 각 리소스를 작업을 수행하는 적절한 공급자와 연결합니다. 자세한 정보는 Resources and Providers Reference를 참조하세요.

`package` 리소스는 Linux 인스턴스에서 소프트웨어 패키지를 관리하는 데 유용합니다. 다음 예제에서는 Apache 패키지를 설치합니다.

```
...
package 'apache2' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    package_name 'httpd'
  when 'debian','ubuntu'
    package_name 'apache2'
  end
  action :install
end
...
```

Chef는 플랫폼에 적절한 패키지 공급자를 사용합니다. 리소스 속성은 단순히 할당된 값일 경우가 많지만, Ruby 논리 연산을 사용하여 조건 할당도 가능합니다. 예제에서는 `case` 연산자가 사용됩니다. 이 연산자는 `node[:platform]`를 사용하여 인스턴스의 운영 체제를 식별하고 그 결과에 따라 `package_name` 속성을 설정합니다. 표준 Chef 노드 구문을 사용하여 레시피에 속성을 삽입할 수 있으며, Chef가 해당 속성을 연결된 값으로 대체합니다. 쿡북 속성뿐 아니라 노드 객체의 모든 속성을 사용할 수 있습니다.

적절한 패키지 이름을 결정하면 코드 세그먼트가 패키지를 설치하는 `install` 작업으로 끝납니다. 이 리소스에 대한 다른 작업은 `upgrade` 및 `remove`를 포함합니다. 자세한 정보는 package를 참조하세요.

복잡한 설치 및 구성 작업을 하나 이상의 하위 작업으로 분할하고 각각 별도의 레시피로 구현한 후 주 레시피가 적절한 시간에 각 레시피를 실행하게 하는 것이 유용할 경우가 많습니다. 다음 예제는 이전 예제에 이어지는 코드입니다.

```
include_recipe 'apache2::service'
```

레시피가 하위 레시피를 실행하도록 하려면 `include_recipe` 키워드와 그 뒤에 레시피 이름을 사용합니다. 레시피는 표준 Chef *`CookbookName`*`::`*`RecipeName`* 구문을 사용하여 식별됩니다. 여기서 *`RecipeName`*에는 `.rb` 확장명을 생략합니다.

> **Note**
>
> `include_recipe` 문은 주 레시피에서 해당 지점의 레시피를 효과적으로 실행합니다. 하지만 실제로는 Chef가 각 `include_recipe` 문을 지정된 레시피의 코드로 대체한 후 주 레시피를 실행합니다.

`directory` 리소스는 디렉터리를 나타냅니다(예: 패키지 파일이 저장된 디렉터리). 다음 `default.rb` 리소스는 Linux 로그 디렉터리를 생성합니다.

```
directory node[:apache][:log_dir] do
    mode 0755
    action :create
end
```

로그 디렉터리는 쿡북의 속성 파일 중 하나에서 정의됩니다. 이 리소스는 디렉터리 모드를 0755로 지정하며, `create` 작업을 사용하여 디렉터리를 생성합니다. 자세한 정보는 directory를 참조하세요. 이 리소스를 Windows 인스턴스에서도 사용할 수 있습니다.

`execute` 리소스는 명령을 나타냅니다(예: shell 명령 또는 스크립트). 다음 예제는 module.load 파일을 생성합니다.

```
execute 'generate-module-list' do
  if node[:kernel][:machine] == 'x86_64'
    libdir = 'lib64'
  else
    libdir = 'lib'
  end
  command "/usr/local/bin/apache2_module_conf_generate.pl /usr/#{libdir}/httpd/
modules /etc/httpd/mods-available"
  action :run
end
```

이 리소스는 먼저 CPU 유형을 식별합니다. `[:kernel][:machine]`은 Chef가 다양한 시스템 속성(이 경우에는 CPU 유형)을 나타내기 위해 생성하는 또 하나의 자동 속성입니다. 그런 다음 명령 Perl 스크립트를 지정하고 `run` 작업을 사용하여 스크립트를 실행합니다. 그러면 module.load 파일이 생성됩니다. 자세한 정보는 실행을 참조하세요.

`template` 리소스는 쿡북의 템플릿 파일 중 하나에서 생성되는 파일(일반적으로 구성 파일)을 나타냅니다. 다음 예제는 템플릿 단원에 설명된 `apache2.conf.erb` 템플릿으로부터 `httpd.conf` 구성 파일을 생성합니다.

```
template 'apache2.conf' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    path "#{node[:apache][:dir]}/conf/httpd.conf"
  when 'debian','ubuntu'
    path "#{node[:apache][:dir]}/apache2.conf"
  end
  source 'apache2.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, resources(:service => 'apache2')
end
```

이 리소스는 인스턴스의 운영 체제에 따라 생성된 파일의 이름 및 위치를 결정합니다. 그런 다음 `apache2.conf.erb`를 파일을 생성하는 데 사용할 템플릿으로 지정하고 파일의 소유자, 그룹 및 모드를 설정합니다. 이 리소스는 `notify` 작업을 실행하여 Apache 서버를 나타내는 `service` 리소스에 서버를 재시작하라고 알립니다. 자세한 정보는 template 단원을 참조하세요.

스택 구성 및 배포 속성: Linux

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 주제에는 가장 일반적으로 사용되는 스택 구성 및 배포 속성과 연결된 노드 구문이 포함됩니다. 이 주제는 Linux 스택이 사용하는 스택 구성 네임스페이스 구조를 중심으로 짜여져 있습니다. 경우에 따라 동일한 속성 이름이 다른 목적으로 사용되며, 다른 네임스페이스에 존재한다는 점에 유의하세요. 예를 들어 `id`는 스택 ID, 계층 ID, 앱 ID 등등을 지칭할 수 있으므로 속성 값을 사용하려면 정규화된 이름이 필요합니다. 이 데이터를 시각화하는 편리한 방법은 JSON 객체로 시각화하는 것입니다. 예제는 스택 구성 및 배포 속성 단원을 참조하세요.

> **Note**
>
> Linux 인스턴스에서 AWS OpsWorks Stacks는 노드 객체에 데이터를 추가하는 것 외에도 각 인스턴스에이 JSON 객체를 설치합니다. 에이전트 CLI의 `get_json` 명령을 사용하면 검색할 수 있습니다.

주제

- opsworks 속성
- opsworks_custom_cookbooks 속성
- dependencies 속성
- ganglia 속성
- mysql 속성
- passenger 속성
- opsworks_bundler 속성
- deploy 속성
- 기타 최상위 속성

opsworks 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`opsworks` 네임스페이스라고도 하는 `opsworks` 요소에는 기본 스택 구성을 정의하는 속성 집합이 포함되어 있습니다.

> **Important**
>
> `opsworks` 네임스페이스에서 속성 값을 재정의하는 것은 권장하지 않습니다. 그럴 경우, 내장 레시피가 실패할 수 있습니다.

주제

- 애플리케이션
- instance 속성
- layers 속성
- rails_stack 속성
- stack 속성
- 그 밖의 최상위 opsworks 속성

애플리케이션

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택에 존재하는 앱마다 하나씩 포함 객체의 목록을 포함합니다. 각각의 포함 객체에는 애플리케이션 구성을 설명하는 다음 속성이 포함됩니다.

> **Note**
>
> 이러한 속성의 일반적 노드 구문은 다음과 같습니다. 여기서 *i*는 인스턴스의 0부터 시작하는 목록 인덱스를 지정합니다.
>
> ```
> node["opsworks"]["applications"]["i"]["attribute_name"]
> ```

application_type

애플리케이션의 유형(문자열). 가능한 값은 다음과 같습니다.

- php: PHP 앱
- `rails`: Ruby on Rails 앱
- `java`: Java 앱
- `nodejs`: Node.js 앱
- `web`: 정적 HTML 페이지
- `other`: 그 밖의 모든 애플리케이션 유형

```
node["opsworks"]["applications"]["i"]["application_type"]
```

name

`"SimplePHP"`와 같은 사용자 정의 표시 이름(문자열).

```
node["opsworks"]["applications"]["i"]["name"]
```

slug_name

앱의 이름으로부터 OpsWorks가 생성하는, `"simplephp"`와 같이 전체가 소문자인 짧은 이름(문자열).

```
node["opsworks"]["applications"]["i"]["slug_name"]
```

instance 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`instance` 속성에는 이 인스턴스의 구성을 지정하는 속성 세트가 포함되어 있습니다.

architecture	availability_zone	backends
aws_instance_id	hostname	id
instance_type	ip	계층
private_dns_name	private_ip	public_dns_name
리전		

architecture

`"i386"`과 같은 인스턴스의 아키텍처(문자열).

```
node["opsworks"]["instance"]["architecture"]
```

availability_zone

`"us-west-2a"`와 같은 인스턴스의 가용 영역(문자열).

```
node["opsworks"]["instance"]["availability_zone"]
```

backends

백엔드 웹 프로세스의 수(문자열). 예컨대 HAProxy가 Rails 백엔드에 전달할 동시 연결의 수를 결정합니다. 기본값은 인스턴스의 메모리 및 코어 개수에 따라 달라집니다.

```
node["opsworks"]["instance"]["backends"]
```

aws_instance_id

EC2 인스턴스 ID(문자열).

```
node["opsworks"]["instance"]["aws_instance_id"]
```

hostname

호스트 이름(문자열), 예: "php-app1".

```
node["opsworks"]["instance"]["hostname"]
```

id

인스턴스를 고유하게 식별하는 AWS OpsWorks Stacks 생성 GUID인 인스턴스 ID(문자열).

```
node["opsworks"]["instance"]["id"]
```

instance_type

"c1.medium"과 같은 인스턴스 유형(문자열).

```
node["opsworks"]["instance"]["instance_type"]
```

ip

퍼블릭 IP 주소(문자열).

```
node["opsworks"]["instance"]["ip"]
```

계층

"lb" 또는 "db-master" 같은 짧은 이름으로 식별되는 인스턴스 계층의 목록(문자열 목록).

```
node["opsworks"]["instance"]["layers"]
```

private_dns_name

프라이빗 DNS 이름(문자열).

```
node["opsworks"]["instance"]["private_dns_name"]
```

private_ip

프라이빗 IP 주소(문자열).

```
node["opsworks"]["instance"]["private_ip"]
```

public_dns_name

퍼블릭 DNS 이름(문자열).

```
node["opsworks"]["instance"]["public_dns_name"]
```

리전

"us-west-2"와 같은 AWS 리전(문자열).

```
node["opsworks"]["instance"]["region"]
```

layers 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`layers` 속성에는 php-app 등 계층의 짧은 이름으로 명명되는 계층 속성 세트가 스택의 각 계층마다 하나씩 포함됩니다. 스택에는 짧은 이름이 다음과 같은 각각의 내장 계층이 최대 하나 있을 수 있습니다.

- `db-master`: MySQL 계층
- `java-app`: Java 앱 서버 계층
- `lb`: HAProxy 계층

- `monitoring-master`: Ganglia 계층
- `memcached`: Memcached 계층
- `nodejs-app`: Node.js 앱 서버 계층
- `php-app`: PHP 앱 서버 계층
- `rails-app`: Rails 앱 서버 계층
- `web`: Static Web Server 계층

스택에는 사용자가 정의한 짧은 이름을 갖는 사용자 지정 계층이 개수 제한 없이 포함될 수 있습니다.

각 계층 속성은 다음 속성을 포함합니다.

- id
- instances
- 이름

id

OpsWorks에 의해 생성되고 계층을 고유하게 식별하는 GUID인 계층 ID(문자열).

```
node["opsworks"]["layers"]["layershortname"]["id"]
```

instances

`instances` 요소에는 계층의 온라인 인스턴스당 하나씩 인스턴스 속성 세트가 포함됩니다. 이들은 `php-app1`과 같이 인스턴스의 호스트 이름으로 명명됩니다.

> **Note**
>
> `instances` 요소에는 특정 스택 구성 및 배포 속성이 생성될 때 온라인 상태인 인스턴스만이 포함됩니다.

각 속성 요소에는 다음 속성이 포함됩니다.

availability_zone	aws_instance_id	backends

booted_at	created_at	elastic_ip
instance_type	ip	private_ip
public_dns_name	private_dns_name	리전
status		

availability_zone

"us-west-2a"와 같은 가용 영역(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["availability_zone"]
```

aws_instance_id

EC2 인스턴스 ID(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["aws_instance_id"]
```

backends

백엔드 웹 프로세스의 수(숫자). 예컨대 HAProxy가 Rails 백엔드에 전달할 동시 연결의 수를 결정합니다. 기본값은 인스턴스의 메모리 및 코어 개수에 따라 달라집니다.

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["backends"]
```

booted_at

EC2 인스턴스가 부팅된 시간으로 UTC yyyy-mm-dddThh:mm:ss+hh:mm 형식을 사용합니다(문자열). 예를 들어 "2013-10-01T08:35:22+00:00"은 2013년 10월 10일 8:35:22에 해당합니다(시간대 오프셋 없음). 자세한 정보는 ISO 8601를 참조하세요.

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["booted_at"]
```

created_at

EC2 인스턴스가 생성된 시간으로 UTC yyyy-mm-dddThh:mm:ss+hh:mm 형식을 사용합니다(문자열). 예를 들어 "2013-10-01T08:35:22+00:00"은 2013년 10월 10일 8:35:22에 해당합니다(시간대 오프셋 없음). 자세한 정보는 ISO 8601를 참조하세요.

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["created_at"]
```

elastic_ip

탄력적 IP 주소로서 인스턴스에 탄력적 IP 주소가 없으면 null로 설정됩니다(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["elastic_ip"]
```

instance_type

"c1.medium"과 같은 인스턴스 유형(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["instance_type"]
```

ip

퍼블릭 IP 주소(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["ip"]
```

private_ip

프라이빗 IP 주소(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_ip"]
```

public_dns_name

퍼블릭 DNS 이름(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["public_dns_name"]
```

private_dns_name

프라이빗 DNS 이름(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_dns_name"]
```

리전

"us-west-2"와 같은 AWS 리전(문자열).

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["region"]
```

status

상태(문자열). 가능한 값은 다음과 같습니다.

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["status"]
```

name

콘솔에서 계층을 나타내는 데 사용되는 계층의 이름(문자열). 사용자 정의 이름일 수 있으며 반드시 고유한 것은 아닙니다.

```
node["opsworks"]["layers"]["layershortname"]["name"]
```

rails_stack 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

name

rails 스택을 지정하며, "apache_passenger" 또는 "nginx_unicorn"으로 설정됩니다(문자열).

```
node["opsworks"]["rails_stack"]["name"]
```

레시피

연결된 레시피로서 Passenger를 사용 중인지 Unicorn을 사용 중인지에 따라 달라집니다(문자열).

- Unicorn: "unicorn::rails"
- Passenger: "passenger_apache2::rails"

```
node["opsworks"]["rails_stack"]["recipe"]
```

restart_command

재시작 명령으로서 Passenger를 사용 중인지 Unicorn을 사용 중인지에 따라 달라집니다(문자열).

- Unicorn: "../../shared/scripts/unicorn clean-restart"
- Passenger: "touch tmp/restart.txt"

서비스

서비스 이름으로서 Passenger를 사용 중인지 Unicorn을 사용 중인지에 따라 달라집니다(문자열).

- Unicorn: "unicorn"
- Passenger: "apache2"

```
node["opsworks"]["rails_stack"]["service"]
```

stack 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`stack` 속성은 서비스 계층 구성 등 스택 구성의 일부 측면을 지정합니다.

- elb-load-balancers
- id
- name
- rds_instances
- vpc_id

elb-load-balancers

스택에 있는 각각의 Elastic Load Balancing 로드 밸런서당 하나씩 포함 객체의 목록을 포함합니다. 각각의 포함 객체에는 로드 밸런서 구성을 설명하는 다음 속성이 포함됩니다.

> **Note**
>
> 이러한 속성의 일반적 노드 구문은 다음과 같습니다. 여기서 *i*는 인스턴스의 0부터 시작하는 목록 인덱스를 지정합니다.
>
> ```
> node["opsworks"]["stack"]["elb-load-balancers"]["i"]["attribute_name"]
> ```

dns_name

로드 밸런서의 DNS 이름(문자열).

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["dns_name"]
```

name

로드 밸런서의 이름(문자열).

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["name"]
```

layer_id

로드 밸런서가 연결된 계층의 ID(문자열).

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["layer_id"]
```

id

스택 ID(문자열).

```
node["opsworks"]["stack"]["id"]
```

name

스택 이름(문자열).

```
node["opsworks"]["stack"]["name"]
```

rds_instances

스택에 등록된 각각의 Amazon RDS 인스턴스에 하나씩 포함 객체의 목록을 포함합니다. 각각의 포함 객체에는 인스턴스의 구성을 설명하는 속성 세트가 포함됩니다. Amazon RDS 콘솔 또는 API를 사용하여 인스턴스를 생성할 때 이 값들을 지정합니다. 인스턴스가 생성된 후 Amazon RDS 콘솔 또는 API를 사용하여 일부 설정을 편집할 수도 있습니다. 자세한 내용은 Amazon RDS 설명서를 참조하세요.

> **Note**
>
> 이러한 속성의 일반적 노드 구문은 다음과 같습니다. 여기서 *i*는 인스턴스의 0부터 시작하는 목록 인덱스를 지정합니다.
>
> ```
> node["opsworks"]["stack"]["rds_instances"]["i"]["attribute_name"]
> ```

스택에 여러 개의 Amazon RDS 인스턴스가 있는 경우, 레시피에서 특정 인스턴스를 사용하는 방법의 예는 다음과 같습니다.

```
if my_rds = node["opsworks"]["stack"]["rds_instances"].select{|rds_instance|
 rds_instance["db_instance_identifier"] == 'db_id' }.first
  template "/etc/rds.conf" do
    source "rds.conf.erb"
    variables :address => my_rds["address"]
  end
end
```

address	allocated_storage	arn
auto_minor_version_upgrade	availability_zone	backup_retention_period
db_instance_class	db_instance_identifier	db_instance_status
db_name	db_parameter_groups	db_security_groups
db_user	engine	instance_create_time
license_model	multi_az	option_group_memberships
포트	preferred_backup_window	preferred_maintenance_window
publicly_accessible	read_replica_db_instance_identifiers	리전
status_infos	vpc_security_groups	

address

`opsinstance.ccdvt3hwog1a.us-west-2.rds.amazonaws.com`와 같은 인스턴스 URL(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["address"]
```

allocated_storage

할당된 스토리지, GB 단위(숫자).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["allocated_storage"]
```

arn

인스턴스의 ARN(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["arn"]
```

auto_minor_version_upgrade

마이너 버전 업그레이드를 자동으로 적용할지 여부(부울).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["auto_minor_version_upgrade"]
```

availability_zone

`us-west-2a`와 같은 인스턴스의 가용 영역(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["availability_zone"]
```

backup_retention_period

백업 보존 기간, 일 단위(숫자).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["backup_retention_period"]
```

db_instance_class

`db.m1.small`과 같은 DB 인스턴스 클래스(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_class"]
```

db_instance_identifier

사용자 정의 DB 인스턴스 식별자(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_identifier"]
```

db_instance_status

인스턴스의 상태(문자열). 자세한 내용은 DB 인스턴스를 참조하세요.

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_status"]
```

db_name

사용자 정의 DB 이름(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_name"]
```

db_parameter_groups

각 파라미터 그룹에 하나씩 포함 객체의 목록이 포함된 인스턴스의 DB 파라미터 그룹. 자세한 내용은 DB 파라미터 그룹 작업을 참조하세요. 각 객체에는 다음 속성이 포함됩니다.

db_parameter_group_name

그룹 이름(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["db_parameter_group_name"]
```

parameter_apply_status

적용 상태(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["parameter_apply_status"]
```

db_security_groups

각 보안 그룹에 하나씩 포함 객체의 목록이 포함된 인스턴스의 데이터베이스 보안 그룹. 자세한 내용은 DB 보안 그룹 작업을 참조하세요. 각 객체에는 다음 속성이 포함됩니다.

db_security_group_name

보안 그룹 이름(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["db_security_group_name"]
```

status

상태(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["status"]
```

db_user

사용자 정의 마스터 사용자 이름(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_user"]
```

engine

`mysql(5.6.13)`과 같은 데이터베이스 엔진(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["engine"]
```

instance_create_time

`2014-04-15T16:13:34Z`와 같은 인스턴스 생성 시간(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["instance_create_time"]
```

license_model

`general-public-license`와 같은 인스턴스의 라이선스 모델(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["license_model"]
```

multi_az

다중 AZ 배포가 활성화되었는지 여부(부울).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["multi_az"]
```

option_group_memberships

각 옵션 그룹에 하나씩 포함 객체의 목록이 포함된 인스턴스의 옵션 그룹 멤버십. 자세한 내용은 옵션 그룹 작업을 참조하세요. 각 객체에는 다음 속성이 포함됩니다.

option_group_name

그룹의 이름(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["option_group_name"]
```

status

그룹의 상태(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["status"]
```

포트

데이터베이스 서버의 포트(숫자).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["port"]
```

preferred_backup_window

`06:26-06:56`과 같은 기본 일일 백업 기간(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_backup_window"]
```

preferred_maintenance_window

`thu:07:13-thu:07:43`과 같은 기본 주별 유지 관리 기간(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_maintenance_window"]
```

publicly_accessible

데이터베이스에 공개적으로 액세스할 수 있는지 여부(부울).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["publicly_accessible"]
```

read_replica_db_instance_identifiers

읽기 전용 복제본 인스턴스 식별자의 목록(문자열의 목록). 자세한 내용은 읽기 전용 복제본 작업을 참조하세요.

```
node["opsworks"]["stack"]["rds_instances"]["i"]
["read_replica_db_instance_identifiers"]
```

리전

us-west-2와 같은 AWS 리전(문자열).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["region"]
```

status_infos

상태 정보의 목록(문자열의 목록).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["status_infos"]
```

vpc_security_groups

VPC 보안 그룹의 목록(문자열의 목록).

```
node["opsworks"]["stack"]["rds_instances"]["i"]["vpc_security_groups"]
```

vpc_id

VPC ID(문자열). 인스턴스가 VPC에 없는 경우, 이 값은 null입니다.

```
node["opsworks"]["stack"]["vpc_id"]
```

그 밖의 최상위 opsworks 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에는 하위 속성이 없는 opsworks 속성이 포함되어 있습니다.

activity

deploy와 같이 속성에 연결된 활동(문자열).

```
node["opsworks"]["activity"]
```

agent_version

인스턴스의 OpsWorks 에이전트 버전(문자열).

```
node["opsworks"]["agent_version"]
```

deploy_chef_provider

배포된 앱의 디렉터리 구조에 영향을 미치는 Chef 배포 공급자(문자열). 이 속성을 다음 중 하나로 설정할 수 있습니다.

- Branch
- Revision
- Timestamped(기본값)

```
node["opsworks"]["deploy_chef_provider"]
```

ruby_stack

Ruby 스택(문자열). 기본 설정은 엔터프라이즈 버전입니다(ruby_enterprise). MRI 버전의 경우, 이 속성을 ruby로 설정합니다.

```
node["opsworks"]["ruby_stack"]
```

ruby_version

애플리케이션이 사용할 Ruby 버전(문자열). 이 속성을 사용하면 메이저 및 마이너 버전만 지정할 수 있습니다. 패치 버전을 지정하려면 적절한 ["ruby"] 속성을 사용해야 합니다. 예제를 포함한 버전

지정 방법에 대한 자세한 정보는 Ruby 버전를 참조하세요. AWS OpsWorks Stacks가 Ruby 버전을 결정하는 방식에 대한 자세한 정보는 내장 속성 파일인 ruby.rb를 참조하세요.

```
node["opsworks"]["ruby_version"]
```

run_cookbook_tests

Chef 11.4 쿡북에서 minitest-chef-handler 테스트를 실행할지 여부(부울).

```
node["opsworks"]["run_cookbook_tests"]
```

sent_at

이 명령이 인스턴스에 전송된 때(숫자).

```
node["opsworks"]["sent_at"]
```

배포

이 속성이 배포 활동에 연결된 경우, `deployment`는 배포를 고유하게 식별하는 AWS OpsWorks Stacks 생성 GUID인 배포 ID로 설정됩니다(문자열). 그렇지 않으면 이 속성은 null로 설정됩니다.

```
node["opsworks"]["deployment"]
```

opsworks_custom_cookbooks 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 사용자 지정 쿡북을 지정하는 속성을 포함합니다.

enabled

사용자 지정 쿡북이 활성화되었는지 여부(부울).

```
node["opsworks_custom_cookbooks"]["enabled"]
```

recipes

사용자 지정 레시피를 포함하여 *cookbookname*::*recipename* 형식을 사용해 이 명령에 실행할 레시피의 목록(문자열의 목록).

```
node["opsworks_custom_cookbooks"]["recipes"]
```

dependencies 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`update_dependencies` 스택 명령에 연결된 몇몇 속성을 포함합니다.

gem_binary

Gems 이진수의 위치(문자열).

upgrade_debs

Debs 패키지를 업그레이드할지 여부(부울).

update_debs

Debs 패키지를 업데이트할지 여부(부울).

ganglia 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Ganglia 통계 웹 페이지에 액세스하는 방법을 지정하는 몇 가지 속성이 포함된 web 속성을 포함합니다.

비밀번호

통계 페이지에 액세스하는 데 필요한 암호(문자열).

```
node["ganglia"]["web"]["password"]
```

url

"/ganglia"와 같은 통계 페이지의 URL 경로(문자열). 완전한 URL은 `http://`*DNSNameURLPath*이며, 여기서 *DNSName*은 연결된 인스턴스의 DNS 이름입니다.

```
node["ganglia"]["web"]["url"]
```

사용자

통계 페이지에 액세스하는 데 필요한 사용자 이름(문자열).

```
node["ganglia"]["web"]["user"]
```

mysql 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

MySQL 데이터베이스 서버 구성을 지정하는 속성 세트를 포함합니다.

clients

클라이언트 IP 주소의 목록(문자열의 목록).

```
node["mysql"]["clients"]
```

server_root_password

루트 암호(문자열).

```
node["mysql"]["server_root_password"]
```

passenger 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Phusion Passenger 구성을 지정하는 속성 세트를 포함합니다.

gem_bin

RubyGems 바이너리의 위치(예: "/usr/local/bin/gem"(문자열))

```
node["passenger"]["gem_bin"]
```

max_pool_size

최대 풀 크기(숫자).

```
node["passenger"]["max_pool_size"]
```

ruby_bin

Ruby 바이너리의 위치(예: "/usr/local/bin/ruby")

```
node["passenger"]["ruby_bin"]
```

version

Passenger 버전(문자열).

```
node["passenger"]["version"]
```

opsworks_bundler 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Bundler 지원을 지정하는 요소를 포함합니다.

manage_package

Bundler를 설치 및 관리할지 여부(부울).

```
node["opsworks_bundler"]["manage_package"]
```

version

bundler 버전(문자열).

```
node["opsworks_bundler"]["version"]
```

deploy 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

속성이 Deploy 이벤트 또는 레시피 실행 스택 명령에 연결된 경우, `deploy` 속성에는 앱의 짧은 이름으로 명명된, 배포된 각 앱의 속성이 포함됩니다. 각 앱 속성은 다음 속성을 포함합니다.

애플리케이션	application_type	auto_bundle_on_deploy
데이터베이스	deploy_to	domains
document_root	environment_variables	그룹
keep_releases	memcached	마이그레이션
mounted_at	purge_before_symlink	rails_env
restart_command	scm	ssl_certificate
ssl_certificate_ca	ssl_certificate_key	ssl_support
스택	symlink_before_migrate	symlinks
사용자		

애플리케이션

"simplephp"와 같은 앱의 slug 이름(문자열).

```
node["deploy"]["appshortname"]["application"]
```

application_type

앱 유형(문자열). 가능한 값은 다음과 같습니다.

- `java`: Java 앱
- `nodejs`: Node.js 앱
- php: PHP 앱
- `rails`: Ruby on Rails 앱

- `web`: 정적 HTML 페이지
- `other`: 그 밖의 모든 애플리케이션 유형

```
node["deploy"]["appshortname"]["application_type"]
```

auto_bundle_on_deploy

Rails 애플리케이션의 경우, 배포 도중 bundler를 실행할지 여부(부울).

```
node["deploy"]["appshortname"]["auto_bundle_on_deploy"]
```

데이터베이스

앱의 데이터베이스를 연결하는 데 필요한 정보를 포함합니다. 앱에 데이터베이스 계층이 연결된 경우 AWS OpsWorks Stacks는 이러한 속성에 적절한 값을 자동으로 할당합니다.

adapter

`mysql`과 같은 데이터베이스 어댑터(문자열).

```
node["deploy"]["appshortname"]["database"]["adapter"]
```

데이터베이스

`"simplephp"` 등 일반적으로 앱의 slug 이름인 데이터베이스 이름(문자열).

```
node["deploy"]["appshortname"]["database"]["database"]
```

data_source_provider

데이터 원본: `mysql` 또는 `rds`(문자열).

```
node["deploy"]["appshortname"]["database"]["data_source_provider"]
```

host

데이터베이스 호스트의 IP 주소(문자열).

```
node["deploy"]["appshortname"]["database"]["host"]
```

비밀번호

데이터베이스 암호(문자열).

```
node["deploy"]["appshortname"]["database"]["password"]
```

포트

데이터베이스 포트(숫자).

```
node["deploy"]["appshortname"]["database"]["port"]
```

reconnect

Rails 애플리케이션의 경우, 연결이 더 이상 존재하지 않을 때 애플리케이션이 다시 연결해야 하는지 여부(부울).

```
node["deploy"]["appshortname"]["database"]["reconnect"]
```

사용자 이름

사용자 이름(문자열).

```
node["deploy"]["appshortname"]["database"]["username"]
```

deploy_to

앱을 배포할 위치(예: "/srv/www/simplephp"(문자열))

```
node["deploy"]["appshortname"]["deploy_to"]
```

domains

앱의 도메인 목록(문자열의 목록).

```
node["deploy"]["appshortname"]["domains"]
```

document_root

기본이 아닌 루트를 지정하는 경우에는 문서 루트, 기본 루트를 사용하는 경우에는 null(문자열).

```
node["deploy"]["appshortname"]["document_root"]
```

environment_variables

앱에 대해 정의된 사용자 지정 환경 변수를 나타내는 최대 20개의 속성 모음. 앱의 환경 변수를 정의하는 방법에 대한 자세한 정보는 앱 추가 단원을 참조하세요. 각 속성 이름은 환경 변수 이름으로 설정되고 해당 값은 변수의 값으로 설정되므로 다음 구문을 사용하여 특정 값 단원을 참조할 수 있습니다.

```
node["deploy"]["appshortname"]["environment_variables"]["variable_name"]
```

그룹

앱의 그룹(문자열).

```
node["deploy"]["appshortname"]["group"]
```

keep_releases

AWS OpsWorks Stacks가 저장할 앱 배포 수(숫자). 이 속성은 앱을 롤백할 수 있는 횟수를 제어합니다. 기본적으로 이 속성은 전역 값인 deploy_keep_releases 로 설정되며, 기본값은 5입니다. keep_releases를 재정의하여 특정 애플리케이션의 저장된 배포 수를 지정할 수 있습니다.

```
node["deploy"]["appshortname"]["keep_releases"]
```

memcached

memcached 구성을 정의하는 두 가지 속성이 포함됩니다.

host

Memcached 서버 인스턴스의 IP 주소(문자열).

```
node["deploy"]["appshortname"]["memcached"]["host"]
```

포트

memcached 서버가 수신 대기하는 포트(숫자).

```
node["deploy"]["appshortname"]["memcached"]["port"]
```

마이그레이션

Rails 애플리케이션의 경우, 마이그레이션을 실행할지 여부(부울).

```
node["deploy"]["appshortname"]["migrate"]
```

mounted_at

기본이 아닌 마운트 포인트를 지정하는 경우에는 앱의 마운트 루트, 기본 마운트 포인트를 사용하는 경우에는 null(문자열).

```
node["deploy"]["appshortname"]["mounted_at"]
```

purge_before_symlink

Rails 앱의 경우, symlink 생성 전에 삭제할 경로 어레이(문자열의 목록).

```
node["deploy"]["appshortname"]["purge_before_symlink"]
```

rails_env

Rails 앱 서버 인스턴스의 경우, "production"과 같은 rails 환경(문자열).

```
node["deploy"]["appshortname"]["rails_env"]
```

restart_command

"echo 'restarting app'" 등 앱이 다시 시작될 때 실행할 명령.

```
node["deploy"]["appshortname"]["restart_command"]
```

scm

OpsWorks가 소스 제어 리포지토리에서 앱을 배포하는 데 사용하는 정보를 지정하는 속성 세트를 포함합니다. 속성은 리포지토리 유형에 따라 다릅니다.

비밀번호

프라이빗 리포지토리의 경우에는 암호, 퍼블릭 리포지토리의 경우 null(문자열). 프라이빗 Amazon S3 버킷의 경우, 이 속성은 보안 키로 설정됩니다.

```
node["deploy"]["appshortname"]["scm"]["password"]
```

리포지토리

"git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git"와 같은 리포지토리 URL(문자열).

```
node["deploy"]["appshortname"]["scm"]["repository"]
```

개정

리포지토리에 여러 브랜치가 있는 경우, 이 속성은 "version1"과 같이 앱의 브랜치 또는 버전을 지정합니다(문자열). 그렇지 않으면 null로 설정됩니다.

```
node["deploy"]["appshortname"]["scm"]["revision"]
```

scm_type

리포지토리 유형(문자열). 가능한 값은 다음과 같습니다.

- "git": Git 리포지토리
- "svn": 하위 버전 리포지토리
- "s3": Amazon S3 버킷
- "archive": HTTP 아카이브
- "other": 기타 리포지토리 유형

```
node["deploy"]["appshortname"]["scm"]["scm_type"]
```

ssh_key

프라이빗 Git 리포지토리에 액세스하는 경우에는 배포 SSH 키, 퍼블릭 리포지토리의 경우에는 null(문자열).

```
node["deploy"]["appshortname"]["scm"]["ssh_key"]
```

사용자

프라이빗 리포지토리의 경우에는 사용자 이름, 퍼블릭 리포지토리의 경우 null(문자열). 프라이빗 Amazon S3 버킷의 경우, 이 속성은 액세스 키로 설정됩니다.

```
node["deploy"]["appshortname"]["scm"]["user"]
```

ssl_certificate

SSL 지원을 활성화한 경우에는 앱의 SSL 인증서, 그렇지 않은 경우에는 null(문자열).

```
node["deploy"]["appshortname"]["ssl_certificate"]
```

ssl_certificate_ca

SSL이 활성화된 경우, 중간 인증서 발급 기관 공개 키 또는 클라이언트 인증을 지정하기 위한 속성(문자열).

```
node["deploy"]["appshortname"]["ssl_certificate_ca"]
```

ssl_certificate_key

SSL 지원을 활성화한 경우에는 앱의 SSL 프라이빗 키, 그렇지 않은 경우에는 null(문자열).

```
node["deploy"]["appshortname"]["ssl_certificate_key"]
```

ssl_support

SSL이 지원되는지 여부(부울).

```
node["deploy"]["appshortname"]["ssl_support"]
```

스택

배포 중에 앱 서버를 다시 로드할지 지정하는 하나의 부울 속성 `needs_reload`를 포함합니다.

```
node["deploy"]["appshortname"]["stack"]["needs_reload"]
```

symlink_before_migrate

Rails 앱의 경우, "*link*":"*target*" 쌍으로 마이그레이션을 실행하기 전에 생성할 symlink를 포함합니다.

```
node["deploy"]["appshortname"]["symlink_before_migrate"]
```

symlinks

"*link*":"*target*" 쌍으로 배포의 symlink를 포함합니다.

```
node["deploy"]["appshortname"]["symlinks"]
```

사용자

앱의 사용자(문자열).

```
node["deploy"]["appshortname"]["user"]
```

기타 최상위 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에는 하위 속성이 없는 최상위 스택 구성 속성이 포함되어 있습니다.

rails 속성

서버의 최대 풀 크기를 지정하는 max_pool_size 속성을 포함합니다(숫자). 속성 값은 AWS OpsWorks Stacks에서 설정하며 인스턴스 유형에 따라 다르지만 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 재정의할 수 있습니다.

```
node["rails"]["max_pool_size"]
```

recipes 속성

"*cookbookname*::*recipename*" 형식을 사용해 이 활동이 실행한 내장 레시피의 목록(문자열의 목록).

```
node["recipes"]
```

opsworks_rubygems 속성

RubyGems 버전을 지정하는 버전 요소를 포함합니다(문자열).

```
node["opsworks_rubygems"]["version"]
```

languages 속성

ruby와 같이 언어에 명명된, 설치된 각 언어의 속성을 포함합니다. 속성은 "/usr/bin/ruby"(문자열) 등의 설치 폴더를 지정하는 ruby_bin과 같은 속성을 포함하는 객체입니다.

ssh_users 속성

SSH 권한을 부여받은 사용자 한 명을 각각 설명하는 속성 세트를 포함합니다. 각 속성의 이름은 사용자의 Unix ID로 지정됩니다. AWS OpsWorks Stacks는 "2001"와 같이 2000-4000 범위의 각 사용자에 대해 고유한 ID를 생성하고 모든 인스턴스에 해당 ID를 가진 사용자를 생성합니다. 는 2000-4000 범위를 AWS OpsWorks 예약하므로 (쿡북 레시피 AWS OpsWorks 를 사용하거나 IAM AWS OpsWorks 에서 사용자를 가져와서) 외부에서 생성하는 사용자는 AWS OpsWorks Stacks에서 다른 사용자에 대해 덮어쓰는 UIDs를 가질 수 있습니다. AWS OpsWorks Stacks 콘솔에서 사용자를 생성하고 액세스를 관리하는 것이 가장 좋습니다. AWS OpsWorks Stacks 외부에서 사용자를 생성하는 경우 4000보다 큰 *UnixID* 값을 사용합니다.

각 속성은 다음 속성을 포함합니다.

이메일

사용자의 이메일 주소(문자열).

```
node["ssh_users"]["UnixID"]["email"]
```

public_key

사용자의 퍼블릭 SSH 키(문자열).

```
node["ssh_users"]["UnixID"]["public_key"]
```

sudoer

사용자에게 sudo 권한이 있는지 여부(부울).

```
node["ssh_users"]["UnixID"]["sudoer"]
```

name

사용자 이름(문자열).

```
node["ssh_users"]["UnixID"]["name"]
```

내장 쿡북 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성 대부분은 Linux 스택에서만 사용할 수 있습니다.

대부분의 내장 레시피에는 다양한 설정을 정의하는 하나 이상의 속성 파일이 있습니다. 사용자 지정 레시피의 이러한 설정에 액세스하고 사용자 지정 JSON을 사용해 재정의할 수 있습니다. 일반적으로 AWS OpsWorks Stacks에서 지원하는 다양한 서버 기술의 구성을 제어하는 속성에 액세스하거나 재정의해야 합니다. 이 섹션에서는 이러한 속성을 요약합니다. 완전한 속성 파일과 연결된 레시피 및 템플릿은 https://github.com/aws/opsworks-cookbooks.git에 나와 있습니다.

> **Note**
>
> 모든 내장 레시피 속성은 `default` 유형입니다.

주제

- apache2 속성
- deploy 속성
- haproxy 속성
- memcached 속성
- mysql 속성
- nginx 속성

- opsworks_berkshelf 속성
- opsworks_java 속성
- passenger_apache2 Attributes
- ruby 속성
- unicorn 속성

apache2 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

apache2 속성은 Apache HTTP 서버 구성을 지정합니다. 자세한 정보는 Apache 핵심 기능 단원을 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

이진수	contact	deflate_types
dir	document_root	그룹
hide_info_headers	icondir	init_script
keepalive	keepaliverequests	keepalivetimeout
lib_dir	libexecdir	listen_ports
log_dir	logrotate 속성	pid_file

prefork 속성	serversignature	servertokens
제한 시간	traceenable	사용자
version	worker 속성	

이진수

Apache 이진수의 위치(문자열). 기본값은 `'/usr/sbin/httpd'`입니다.

```
node[:apache][:binary]
```

contact

이메일 연락처(문자열). 기본값은 더미 주소인 `'ops@example.com'`입니다.

```
node[:apache][:contact]
```

deflate_types

`mod_deflate`에게 브라우저가 지원하는 지정된 Mime 유형에 대해 압축을 활성화할 것을 지시합니다(문자열의 목록). 기본값은 다음과 같습니다.

```
['application/javascript',
 'application/json',
 'application/x-javascript',
 'application/xhtml+xml',
 'application/xml',
 'application/xml+rss',
 'text/css',
 'text/html',
 'text/javascript',
 'text/plain',
 'text/xml']
```

> **Warning**
>
> 압축은 보안 위험을 초래할 수 있습니다. 압축을 완전히 비활성화하려면 이 속성을 다음과 같이 설정하세요.

```
node[:apache][:deflate_types] = []
```

```
node[:apache][:deflate_types]
```

dir

서버의 루트 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 Red Hat Enterprise Linux(RHEL): `'/etc/httpd'`
- Ubuntu: `'/etc/apache2'`

```
node[:apache][:dir]
```

document_root

문서 루트(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'/var/www/html'`
- Ubuntu: `'/var/www'`

```
node[:apache][:document_root]
```

그룹

그룹 이름(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'apache'`
- Ubuntu: `'www-data'`

```
node[:apache][:group]
```

hide_info_headers

HTTP 헤더에서 버전 및 모듈 정보를 생략할지 여부(`'true'`/`'false'`)(문자열). 기본값은 `'true'`입니다.

```
node[:apache][:hide_info_headers]
```

icondir

아이콘 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'/var/www/icons/'`
- Ubuntu: `'/usr/share/apache2/icons'`

```
node[:apache][:icondir]
```

init_script

초기화 스크립트(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'/etc/init.d/httpd'`
- Ubuntu: `'/etc/init.d/apache2'`

```
node[:apache][:init_script]
```

keepalive

연결 유지를 활성화할지 여부(문자열). 가능한 값은 `'On'`과 `'Off'`입니다(문자열). 기본값은 `'Off'`입니다.

```
node[:apache][:keepalive]
```

keepaliverequests

Apache가 동시에 처리할 연결 유지 요청의 최대 개수(숫자). 기본값은 `100`입니다.

```
node[:apache][:keepaliverequests]
```

keepalivetimeout

연결을 닫기 전에 Apache가 요청을 기다리는 시간(숫자). 기본값은 `3`입니다.

```
node[:apache][:keepalivetimeout]
```

lib_dir

객체 코드 라이브러리가 포함된 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux(x86): `'/usr/lib/httpd'`
- Amazon Linux(x64) 및 RHEL: `'/usr/lib64/httpd'`

- Ubuntu: '/usr/lib/apache2'

```
node[:apache][:lib_dir]
```

libexecdir

프로그램 실행 파일이 포함된 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux(x86): '/usr/lib/httpd/modules'
- Amazon Linux(x64) 및 RHEL: '/usr/lib64/httpd/modules'
- Ubuntu: '/usr/lib/apache2/modules'

```
node[:apache][:libexecdir]
```

listen_ports

서버가 수신하는 포트의 목록(문자열의 목록). 기본값은 ['80','443']입니다.

```
node[:apache][:listen_ports]
```

log_dir

로그 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: '/var/log/httpd'
- Ubuntu: '/var/log/apache2'

```
node[:apache][:log_dir]
```

logrotate 속성

이러한 속성은 로그 파일을 교체하는 방법을 지정합니다.

delaycompress

다음 교체 주기가 시작될 때까지 닫힌 로그 파일 압축을 지연할지 여부('true'/'false')(문자열). 기본값은 'true'입니다.

```
node[:apache][:logrotate][:delaycompress]
```

그룹

로그 파일의 그룹(문자열). 기본값은 'adm'입니다.

```
node[:apache][:logrotate][:group]
```

mode

로그 파일의 모드(문자열). 기본값은 '640'입니다.

```
node[:apache][:logrotate][:mode]
```

owner

로그 파일의 소유자(문자열). 기본값은 'root'입니다.

```
node[:apache][:logrotate][:owner]
```

rotate

닫힌 로그 파일이 제거되기 전 교체 주기의 수(문자열). 기본값은 '30'입니다.

```
node[:apache][:logrotate][:rotate]
```

schedule

교체 일정(문자열). 가능한 값은 다음과 같습니다.

- 'daily'
- 'weekly'
- 'monthly'

기본값은 'daily'입니다.

```
node[:apache][:logrotate][:schedule]
```

pid_file

데몬의 프로세스 ID가 포함된 파일(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: '/var/run/httpd/httpd.pid'
- Ubuntu: '/var/run/apache2.pid'

```
node[:apache][:pid_file]
```

prefork 속성

이러한 속성은 프리포킹(pre-forking) 구성을 지정합니다.

maxclients

처리할 동시 요청의 최대 수(숫자). 기본값은 `400`입니다.

> **Note**
>
> 이 속성은 Amazon Linux 또는 RHEL을 실행하는 인스턴스에만 사용하세요. 인스턴스가 Ubuntu 14.04 LTS를 실행 중인 경우, maxrequestworkers를 사용하세요.

```
node[:apache][:prefork][:maxclients]
```

maxrequestsperchild

하위 서버 프로세스가 처리할 요청의 최대 수(숫자). 기본값은 `10000`입니다.

```
node[:apache][:prefork][:maxrequestsperchild]
```

maxrequestworkers

처리할 동시 요청의 최대 수(숫자). 기본값은 `400`입니다.

> **Note**
>
> 이 속성은 Ubuntu 14.04 LTS를 실행하는 인스턴스에만 사용하세요. 인스턴스가 Amazon Linux 또는 RHEL을 실행 중인 경우, maxclients 를 사용하세요.

```
node[:apache][:prefork][:maxrequestworkers]
```

maxspareservers

유휴 하위 서버 프로세스의 최대 수(숫자). 기본값은 `32`입니다.

```
node[:apache][:prefork][:maxspareservers]
```

minspareservers

유휴 하위 서버 프로세스의 최소 수(숫자). 기본값은 16입니다.

```
node[:apache][:prefork][:minspareservers]
```

serverlimit

구성할 수 있는 프로세스의 최대 수(숫자). 기본값은 400입니다.

```
node[:apache][:prefork][:serverlimit]
```

startservers

시작 시 생성될 하위 서버 프로세스의 수(숫자). 기본값은 16입니다.

```
node[:apache][:prefork][:startservers]
```

serversignature

서버 생성 문서의 꼬리말을 구성할지 여부 및 구성 방법을 지정합니다(문자열). 가능한 값은 'On', 'Off' 및 'Email'입니다. 기본값은 'Off'입니다.

```
node[:apache][:serversignature]
```

servertokens

응답 헤더에 어떤 유형의 서버 버전 정보가 포함되는지 지정합니다(문자열).

- 'Full': 전체 정보. 예를 들어 Server: Apache/2.4.2 (Unix) PHP/4.2.2 MyMod/1.2
- 'Prod': 제품 이름. 예를 들어 Server: Apache
- 'Major': 메이저 버전. 예를 들어 Server: Apache/2
- 'Minor': 메이저 및 마이너 버전. 예를 들어 Server: Apache/2.4
- 'Min': 최소 버전. 예를 들어 Server: Apache/2.4.2
- 'OS': 운영 체제 포함 버전. 예를 들어 Server: Apache/2.4.2 (Unix)

기본값은 'Prod'입니다.

```
node[:apache][:servertokens]
```

제한 시간

Apache가 I/O를 기다리는 시간(숫자). 기본값은 120입니다.

```
node[:apache][:timeout]
```

traceenable

TRACE 요청을 활성화할지 여부(문자열). 가능한 값은 'On'와 'Off'입니다. 기본값은 'Off'입니다.

```
node[:apache][:traceenable]
```

사용자

사용자 이름(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: 'apache'
- Ubuntu: 'www-data'

```
node[:apache][:user]
```

version

Apache 버전(문자열). 기본값은 다음과 같습니다.

- Amazon Linux: 2.2
- Ubuntu 14.04 LTS: 2.4
- RHEL: 2.4

```
node[:apache][:version]
```

worker 속성

이러한 속성은 worker 프로세스 구성을 지정합니다.

startservers

시작 시 생성될 하위 서버 프로세스의 수(숫자). 기본값은 4입니다.

```
node[:apache][:worker][:startservers]
```

maxclients

처리할 동시 요청의 최대 수(숫자). 기본값은 1024입니다.

```
node[:apache][:worker][:maxclients]
```

maxsparethreads

유휴 스레드의 최대 수(숫자). 기본값은 192입니다.

```
node[:apache][:worker][:maxsparethreads]
```

minsparethreads

유휴 스레드의 최소 수(숫자). 기본값은 64입니다.

```
node[:apache][:worker][:minsparethreads]
```

threadsperchild

하위 프로세스당 스레드 수(숫자). 기본값은 64입니다.

```
node[:apache][:worker][:threadsperchild]
```

maxrequestsperchild

하위 서버 프로세스가 처리할 요청의 최대 수(숫자). 기본값은 10000입니다.

```
node[:apache][:worker][:maxrequestsperchild]
```

deploy 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

내장 배포 쿡북의 `deploy.rb` 속성 파일은 `opsworks` 네임스페이스에서 다음 속성을 정의합니다. 배포 디렉터리에 대한 자세한 정보는 Deploy 레시피 단원을 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

deploy_keep_releases

AWS OpsWorks Stacks가 저장할 앱 배포 수(숫자)에 대한 전역 설정입니다. 기본값은 5입니다. 이 값은 앱을 롤백할 수 있는 횟수를 제어합니다.

```
node[:opsworks][:deploy_keep_releases]
```

그룹

(Linux 한정) 앱의 배포 디렉터리에 대한 `group` 설정(문자열). 기본값은 인스턴스의 운영 체제에 따라 다릅니다.

- Ubuntu 인스턴스의 경우, 기본값은 `www-data`입니다.
- Nginx와 Unicorn을 사용하는 Rails 앱 서버 계층에 속하는 Amazon Linux 또는 RHEL 인스턴스의 경우, 기본값은 `nginx`입니다.
- 다른 모든 Amazon Linux 또는 RHEL 인스턴스의 경우, 기본값은 `apache`입니다.

```
node[:opsworks][:deploy_user][:group]
```

사용자

(Linux 한정) 앱의 배포 디렉터리에 대한 `user` 설정(문자열). 기본값은 `deploy`입니다.

```
node[:opsworks][:deploy_user][:user]
```

haproxy 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

`haproxy` 속성은 HAProxy 서버 구성을 지정합니다. 자세한 정보는 HAProxy Docs를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

balance	check_interval	client_timeout
connect_timeout	default_max_connections	global_max_connections
health_check_method	health_check_url	queue_timeout
http_request_timeout	maxcon_factor_nodejs_app	maxcon_factor_nodejs_app_ssl
maxcon_factor_php_app	maxcon_factor_php_app_ssl	maxcon_factor_rails_app
maxcon_factor_rails_app_ssl	maxcon_factor_static	maxcon_factor_static_ssl
retries	server_timeout	stats_url
stats_user		

balance

로드 밸런서가 서버를 선택하는 데 사용하는 알고리즘(문자열). 기본값은 `'roundrobin'`입니다. 그 밖의 옵션은 다음과 같습니다.

- 'static-rr'
- 'leastconn'
- 'source'
- 'uri'
- 'url_param'
- 'hdr(name)'
- 'rdp-cookie'

- 'rdp-cookie(name)'

이러한 인수에 대한 자세한 정보는 밸런스를 참조하세요.

```
node[:haproxy][:balance]
```

check_interval

상태 확인 시간 간격(문자열). 기본값은 '10s'입니다.

```
node[:haproxy][:check_interval]
```

client_timeout

클라이언트가 비활성 상태로 있을 수 있는 최대 시간(문자열). 기본값은 '60s'입니다.

```
node[:haproxy][:client_timeout]
```

connect_timeout

HAProxy가 서버 연결 시도 성공을 기다리는 최대 시간(문자열). 기본값은 '10s'입니다.

```
node[:haproxy][:connect_timeout]
```

default_max_connections

연결의 기본 최대 수(문자열). 기본값은 '80000'입니다.

```
node[:haproxy][:default_max_connections]
```

global_max_connections

연결의 최대 수(문자열). 기본값은 '80000'입니다.

```
node[:haproxy][:global_max_connections]
```

health_check_method

상태 확인 메서드(문자열). 기본값은 'OPTIONS'입니다.

```
node[:haproxy][:health_check_method]
```

health_check_url

서버 상태 확인에 사용되는 URL 경로(문자열). 기본값은 '/'입니다.

```
node[:haproxy][:health_check_url ]
```

queue_timeout

무료 연결의 최대 대기 시간(문자열). 기본값은 '120s'입니다.

```
node[:haproxy][:queue_timeout]
```

http_request_timeout

HAProxy가 완전한 HTTP 요청을 기다리는 최대 시간(문자열). 기본값은 '30s'입니다.

```
node[:haproxy][:http_request_timeout]
```

retries

서버 연결 실패 후 재시도 횟수(문자열). 기본값은 '3'입니다.

```
node[:haproxy][:retries]
```

server_timeout

클라이언트가 비활성 상태로 있을 수 있는 최대 시간(문자열). 기본값은 '60s'입니다.

```
node[:haproxy][:server_timeout]
```

stats_url

통계 페이지의 URL 경로(문자열). 기본값은 '/haproxy?stats'입니다.

```
node[:haproxy][:stats_url]
```

stats_user

통계 페이지 사용자 이름(문자열). 기본값은 'opsworks'입니다.

```
node[:haproxy][:stats_user]
```

maxcon 속성은 HAProxy가 백엔드에 허용하는 최대 연결 수를 계산하는 데 사용되는 로드 비율 승수를 나타냅니다. 예를 들어 backend 값이 4인 작은 인스턴스에 Rails 앱 서버가 있다고 가정해 보겠습니다. 즉, AWS OpsWorks Stacks는 해당 인스턴스에 대해 4개의 Rails 프로세스를 구성합니다. 기본 maxcon_factor_rails_app 값인 7을 사용하는 경우, HAProxy는 Rails 서버와의 연결 28(4*7)개를 처리합니다.

maxcon_factor_nodejs_app

Node.js 앱 서버의 maxcon 비율(숫자). 기본값은 10입니다.

```
node[:haproxy][:maxcon_factor_nodejs_app]
```

maxcon_factor_nodejs_app_ssl

SSL 포함 Node.js 앱 서버의 maxcon 비율(숫자). 기본값은 10입니다.

```
node[:haproxy][:maxcon_factor_nodejs_app_ssl]
```

maxcon_factor_php_app

PHP 앱 서버의 maxcon 비율(숫자). 기본값은 10입니다.

```
node[:haproxy][:maxcon_factor_php_app]
```

maxcon_factor_php_app_ssl

SSL 포함 PHP 앱 서버의 maxcon 비율(숫자). 기본값은 10입니다.

```
node[:haproxy][:maxcon_factor_php_app_ssl]
```

maxcon_factor_rails_app

Rails 앱 서버의 maxcon 비율(숫자). 기본값은 7입니다.

```
node[:haproxy][:maxcon_factor_rails_app]
```

maxcon_factor_rails_app_ssl

SSL 포함 Rails 앱 서버의 maxcon 비율(숫자). 기본값은 7입니다.

```
node[:haproxy][:maxcon_factor_rails_app_ssl]
```

maxcon_factor_static

정적 웹 서버의 maxcon 비율(숫자). 기본값은 15입니다.

```
node[:haproxy][:maxcon_factor_static]
```

maxcon_factor_static_ssl

SSL 포함 정적 웹 서버의 maxcon 비율(숫자). 기본값은 15입니다.

```
node[:haproxy][:maxcon_factor_static_ssl]
```

memcached 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

memcached 속성은 Memcached 서버 구성을 지정합니다. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

메모리	max_connections	pid_file
포트	start_command	stop_command
사용자		

메모리

사용할 최대 메모리 양(MB)(숫자). 기본값은 512입니다.

```
node[:memcached][:memory]
```

max_connections

연결의 최대 수(문자열). 기본값은 '4096'입니다.

```
node[:memcached][:max_connections]
```

pid_file

데몬의 프로세스 ID가 포함된 파일(문자열). 기본값은 'var/run/memcached.pid'입니다.

```
node[:memcached][:pid_file]
```

포트

수신 대기할 포트(숫자). 기본값은 11211입니다.

```
node[:memcached][:port]
```

start_command

시작 명령(문자열). 기본값은 '/etc/init.d/memcached start'입니다.

```
node[:memcached][:start_command]
```

stop_command

중지 명령(문자열). 기본값은 '/etc/init.d/memcached stop'입니다.

```
node[:memcached][:stop_command]
```

사용자

사용자(문자열). 기본값은 'nobody'입니다.

```
node[:memcached][:user]
```

mysql 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

`mysql` 속성은 MySQL 마스터 구성을 지정합니다. 자세한 정보는 서버 시스템 변수를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

basedir	bind_address	clients
conf_dir	confd_dir	datadir
grants_path	mysql_bin	mysqladmin_bin
pid_file	포트	root_group
server_root_password	소켓	tunable 속성

basedir

기본 디렉터리(문자열). 기본값은 `'/usr'`입니다.

```
node[:mysql][:basedir]
```

bind_address

MySQL이 수신 대기하는 주소(문자열). 기본값은 `'0.0.0.0'`입니다.

```
node[:mysql][:bind_address]
```

clients

클라이언트 목록(문자열의 목록).

```
node[:mysql][:clients]
```

conf_dir

구성 파일이 포함된 디렉터리(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'/etc'`
- Ubuntu: `'/etc/mysql'`

```
node[:mysql][:conf_dir]
```

confd_dir

추가 구성 파일이 포함된 디렉터리(문자열). 기본값은 `'/etc/mysql/conf.d'`입니다.

```
node[:mysql][:confd_dir]
```

datadir

데이터 디렉터리(문자열). 기본값은 `'/var/lib/mysql'`입니다.

```
node[:mysql][:datadir]
```

grants_path

그랜트 테이블 위치(문자열). 기본값은 `'/etc/mysql_grants.sql'`입니다.

```
node[:mysql][:grants_path]
```

mysql_bin

mysql 이진수 위치(문자열). 기본값은 `'/usr/bin/mysql'`입니다.

```
node[:mysql][:mysql_bin]
```

mysqladmin_bin

mysqladmin 위치(문자열). 기본값은 '/usr/bin/mysqladmin'입니다.

```
node[:mysql][:mysqladmin_bin]
```

pid_file

데몬의 프로세스 ID가 포함된 파일(문자열). 기본값은 '/var/run/mysqld/mysqld.pid'입니다.

```
node[:mysql][:pid_file]
```

포트

서버가 수신 대기하는 포트(숫자). 기본값은 3306입니다.

```
node[:mysql][:port]
```

root_group

루트 그룹(문자열). 기본값은 'root'입니다.

```
node[:mysql][:root_group]
```

server_root_password

서버의 루트 암호(문자열). 기본값은 무작위로 생성됩니다.

```
node[:mysql][:server_root_password]
```

소켓

소켓 파일의 위치(문자열). 기본값은 '/var/lib/mysql/mysql.sock'입니다. 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: '/var/lib/mysql/mysql.sock'
- Ubuntu: '/var/run/mysqld/mysqld.sock'

```
node[:mysql][:socket]
```

tunable 속성

tunable 속성은 성능 튜닝에 사용됩니다.

back_log	innodb_additional_mem_pool_size	innodb_buffer_pool_size
innodb_flush_log_at_trx_commit	innodb_lock_wait_timeout	key_buffer
log_slow_queries	long_query_time	max_allowed_packet
max_connections	max_heap_table_size	net_read_timeout
net_write_timeout	query_cache_limit	query_cache_size
query_cache_type	thread_cache_size	thread_stack
wait_timeout	table_cache	

back_log

대기 중인 요청의 최대 수(문자열). 기본값은 `'128'`입니다.

```
node[:mysql][:tunable][:back_log]
```

innodb_additional_mem_pool_size

Innodb가 내부 데이터 구조를 저장하는 데 사용하는 풀의 크기(문자열). 기본값은 `'20M'`입니다.

```
node[:mysql][:tunable][:innodb_additional_mem_pool_size]
```

innodb_buffer_pool_size

Innodb 버퍼 풀 크기(문자열). 속성 값은 AWS OpsWorks Stacks에서 설정하며 인스턴스 유형에 따라 다르지만 사용자 지정 JSON 또는 사용자 지정 속성 파일을 사용하여 재정의할 수 있습니다.

```
node[:mysql][:tunable][:innodb_buffer_pool_size]
```

innodb_flush_log_at_trx_commit

Innodb가 로그 버퍼를 플러시하는 빈도(문자열). 기본값은 '2'입니다. 자세한 정보는 innodb_flush_log_at_trx_commit 단원을 참조하세요.

```
node[:mysql][:tunable][:innodb_flush_log_at_trx_commit]
```

innodb_lock_wait_timeout

Innodb 트랜잭션이 행 잠금을 기다리는 최대 시간(초)(문자열). 기본값은 '50'입니다.

```
node[:mysql][:tunable][:innodb_lock_wait_timeout]
```

key_buffer

인덱스 버퍼 크기(문자열). 기본값은 '250M'입니다.

```
node[:mysql][:tunable][:key_buffer]
```

log_slow_queries

느린 쿼리 로그 파일의 위치(문자열). 기본값은 '/var/log/mysql/mysql-slow.log'입니다.

```
node[:mysql][:tunable][:log_slow_queries]
```

long_query_time

쿼리를 장기 쿼리로 지정하는 데 필요한 시간(초)(문자열). 기본값은 '1'입니다.

```
node[:mysql][:tunable][:long_query_time]
```

max_allowed_packet

허용되는 최대 패킷 크기(문자열). 기본값은 '32M'입니다.

```
node[:mysql][:tunable][:max_allowed_packet]
```

max_connections

동시 클라이언트 연결 최대 수(문자열). 기본값은 '2048'입니다.

```
node[:mysql][:tunable][:max_connections]
```

max_heap_table_size

사용자가 만든 MEMORY 테이블의 최대 크기(문자열). 기본값은 '32M'입니다.

```
node[:mysql][:tunable][:max_heap_table_size]
```

net_read_timeout

연결로부터 더 많은 데이터를 기다릴 시간(초)(문자열). 기본값은 '30'입니다.

```
node[:mysql][:tunable][:net_read_timeout]
```

net_write_timeout

블록이 연결에 작성되기를 기다리는 시간(초)(문자열). 기본값은 '30'입니다.

```
node[:mysql][:tunable][:net_write_timeout]
```

query_cache_limit

캐시된 개별 쿼리의 최대 크기(문자열). 기본값은 '2M'입니다.

```
node[:mysql][:tunable][:query_cache_limit]
```

query_cache_size

쿼리 캐시 크기(문자열). 기본값은 '128M'입니다.

```
node[:mysql][:tunable][:query_cache_size]
```

query_cache_type

쿼리 캐시 유형(문자열). 가능한 값은 다음과 같습니다.

- '0': 캐시 또는 캐시된 데이터 검색 없음.
- '1': SELECT SQL_NO_CACHE로 시작하지 않는 Cache 문.
- '2': SELECT SQL_CACHE로 시작하는 Cache 문.

기본값은 '1'입니다.

```
node[:mysql][:tunable][:query_cache_type]
```

thread_cache_size

재사용을 위해 캐시되는 클라이언트 스레드 수(문자열). 기본값은 '8'입니다.

```
node[:mysql][:tunable][:thread_cache_size]
```

thread_stack

각 스레드의 스택 크기(문자열). 기본값은 '192K'입니다.

```
node[:mysql][:tunable][:thread_stack]
```

wait_timeout

상호 작용하지 않는 연결을 기다릴 시간(초). 기본값은 '180'입니다(문자열).

```
node[:mysql][:tunable][:wait_timeout]
```

table_cache

열려 있는 테이블의 수(문자열). 기본값은 '2048'입니다.

```
node[:mysql][:tunable][:table_cache]
```

nginx 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

nginx 속성은 Nginx 구성을 지정합니다. 자세한 정보는 명령 인덱스를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

이진수	dir	gzip
gzip_comp_level	gzip_disable	gzip_http_version
gzip_proxied	gzip_static	gzip_types
gzip_vary	keepalive	keepalive_timeout
log_dir	사용자	server_names_hash_bucket_size
worker_processes	worker_connections	

이진수

Nginx 이진수의 위치(문자열). 기본값은 `'/usr/sbin/nginx'`입니다.

```
node[:nginx][:binary]
```

dir

구성 파일 등의 파일 위치(문자열). 기본값은 `'/etc/nginx'`입니다.

```
node[:nginx][:dir]
```

gzip

gzip 압축이 활성화되어 있는지 여부(문자열). 가능한 값은 `'on'`와 `'off'`입니다. 기본값은 `'on'`입니다.

> **Warning**
>
> 압축은 보안 위험을 초래할 수 있습니다. 압축을 완전히 비활성화하려면 이 속성을 다음과 같이 설정하세요.
>
> ```
> node[:nginx][:gzip] = 'off'
> ```

```
node[:nginx][:gzip]
```

gzip_comp_level

1-9까지의 압축 수준(1이 최소 압축)(문자열). 기본값은 '2'입니다.

```
node[:nginx][:gzip_comp_level]
```

gzip_disable

지정된 사용자 에이전트에 대해 gzip 압축을 비활성화합니다(문자열). 값은 정규 표현식이며 기본값은 'MSIE [1-6].(?!.*SV1)'입니다.

```
node[:nginx][:gzip_disable]
```

gzip_http_version

지정된 HTTP 버전에 대해 gzip 압축을 활성화합니다(문자열). 기본값은 '1.0'입니다.

```
node[:nginx][:gzip_http_version]
```

gzip_proxied

프록시 요청에 대한 응답을 압축할지 여부 및 압축 방법이며, 다음 값 중 하나를 취할 수 있습니다(문자열).

- 'off': 프록시된 요청을 압축하지 않습니다
- 'expired': Expire 헤더가 캐싱을 금지하는 경우, 압축합니다
- 'no-cache': Cache-Control 헤더가 "no-cache"로 설정된 경우, 압축합니다
- 'no-store': Cache-Control 헤더가 "no-store"로 설정된 경우, 압축합니다
- 'private': Cache-Control 헤더가 "private"으로 설정된 경우, 압축합니다
- 'no_last_modified': Last-Modified가 설정되지 않은 경우, 압축합니다
- 'no_etag': 요청에 ETag 헤더가 없는 경우, 압축합니다
- 'auth': 요청에 Authorization 헤더가 포함된 경우, 압축합니다
- 'any': 모든 프록시된 요청을 압축합니다

기본값은 'any'입니다.

```
node[:nginx][:gzip_proxied]
```

gzip_static

gzip 정적 모듈이 활성화되어 있는지 여부(문자열). 가능한 값은 'on'와 'off'입니다. 기본값은 'on'입니다.

```
node[:nginx][:gzip_static]
```

gzip_types

압축할 MIME 형식의 목록(문자열의 목록). 기본값은 ['text/plain', 'text/html', 'text/css', 'application/x-javascript', 'text/xml', 'application/xml', 'application/xml+rss', 'text/javascript']입니다.

```
node[:nginx][:gzip_types]
```

gzip_vary

Vary:Accept-Encoding 응답 헤더를 활성화할지 여부(문자열). 가능한 값은 'on'와 'off'입니다. 기본값은 'on'입니다.

```
node[:nginx][:gzip_vary]
```

keepalive

연결 유지 연결을 활성화할지 여부(문자열). 가능한 값은 'on'와 'off'입니다. 기본값은 'on'입니다.

```
node[:nginx][:keepalive]
```

keepalive_timeout

연결 유지 연결이 계속 열려 있는 최대 시간(초)(숫자). 기본값은 65입니다.

```
node[:nginx][:keepalive_timeout]
```

log_dir

로그 파일의 위치(문자열). 기본값은 '/var/log/nginx'입니다.

```
node[:nginx][:log_dir]
```

사용자

사용자(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `'www-data'`
- Ubuntu: `'nginx'`

```
node[:nginx][:user]
```

server_names_hash_bucket_size

서버 이름의 해시 테이블 버킷 크기로서 32, 64 또는 128로 설정할 수 있습니다(숫자). 기본값은 64입니다.

```
node[:nginx][:server_names_hash_bucket_size]
```

worker_processes

worker 프로세스의 수(숫자). 기본값은 10입니다.

```
node[:nginx][:worker_processes]
```

worker_connections

worker 연결의 최대 수(숫자). 기본값은 1024입니다. 클라이언트의 최대 수는 `worker_processes * worker_connections`로 설정됩니다.

```
node[:nginx][:worker_connections]
```

opsworks_berkshelf 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

opsworks_berkshelf 속성은 Berkshelf 구성을 지정합니다. 자세한 정보는 Berkshelf를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

debug

Chef 로그에 Berkshelf 디버깅 정보를 포함할지 여부(부울). 기본값은 `false`입니다.

```
node['opsworks_berkshelf]['debug']
```

opsworks_java 속성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

opsworks_java 속성은 Tomcat 서버 구성을 지정합니다. 자세한 정보는 Apache Tomcat 구성 참조를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

datasources	java_app_server_version	java_shared_lib_dir
jvm_pkg Attributes	custom_pkg_location_url_debian	java_home_basedir
custom_pkg_location_url_rhel	use_custom_pkg_location	jvm_options
jvm_version	tomcat 속성	

datasources

JNDI 리소스 이름을 정의하는 속성 집합(문자열). 이 속성을 사용하는 방법에 대한 자세한 정보는 백엔드 데이터베이스를 사용하여 JSP 앱 배포 단원을 참조하세요. 기본값인 빈 해시는 앱 짧은 이름과 JNDI 이름 사이의 사용자 지정 매핑으로 채울 수 있습니다. 자세한 내용은 백엔드 데이터베이스를 사용하여 JSP 앱 배포 섹션을 참조하세요.

```
node['opsworks_java']['datasources']
```

java_app_server_version

Java 앱 서버 버전(숫자). 기본값은 7입니다. 이 속성을 재정의하여 버전 6을 지정할 수 있습니다. 기본이 아닌 JDK를 설치하는 경우, 이 속성은 무시됩니다.

```
node['opsworks_java']['java_app_server_version']
```

java_shared_lib_dir

Java 공유 라이브러리의 디렉터리(문자열). 기본값은 `/usr/share/java`입니다.

```
node['opsworks_java']['java_shared_lib_dir']
```

jvm_pkg Attributes

재정의하여 기본이 아닌 JDK를 설치할 수 있는 속성 세트.

use_custom_pkg_location

OpenJDK 대신 사용자 지정 JDK를 설치할지 여부(부울). 기본값은 `false`입니다.

```
node['opsworks_java']['jvm_pkg']['use_custom_pkg_location']
```

custom_pkg_location_url_debian

Ubuntu 인스턴스에 설치할 JDK 패키지의 위치(문자열). 기본값은 'http://aws.amazon.com/'이며, 별 의미가 없는 단순한 초기화 값입니다. 기본이 아닌 JDK를 설치하려면 이 속성을 재정의해 적절한 URL로 설정해야 합니다.

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_debian']
```

custom_pkg_location_url_rhel

Amazon Linux 및 RHEL 인스턴스에 설치할 JDK 패키지의 위치(문자열). 기본값은 'http://aws.amazon.com/'이며, 별 의미가 없는 단순한 초기화 값입니다. 기본이 아닌 JDK를 설치하려면 이 속성을 재정의해 적절한 URL로 설정해야 합니다.

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_rhel']
```

java_home_basedir

JDK 패키지가 추출될 디렉터리(문자열). 기본값은 /usr/local입니다. RPM 패키지의 경우, 이 설정을 지정할 필요가 없습니다. 완전한 디렉터리 구조가 포함되어 있기 때문입니다.

```
node['opsworks_java']['jvm_pkg']['java_home_basedir']
```

jvm_options

JVM 명령줄 옵션으로서 힙 크기 같은 설정을 지정할 수 있습니다(문자열). 공통 옵션 세트는 -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC입니다. 기본값은 no options입니다.

```
node['opsworks_java']['jvm_options']
```

jvm_version

OpenJDK 버전(숫자). 기본값은 7입니다. 이 속성을 재정의하여 OpenJDK 버전 6을 지정할 수 있습니다. 기본이 아닌 JDK를 설치하는 경우, 이 속성은 무시됩니다.

```
node['opsworks_java']['jvm_version']
```

tomcat 속성

재정의하여 기본 Tomcat 구성을 설치할 수 있는 속성 세트.

ajp_port	apache_tomcat_bind_mod	apache_tomcat_bind_path
auto_deploy	connection_timeout	mysql_connector_jar
포트	secure_port	shutdown_port
threadpool_max_threads	threadpool_min_spare_threads	unpack_wars
uri_encoding	use_ssl_connector	use_threadpool
userdatabase_pathname		

ajp_port

AJP 포트(숫자). 기본값은 8009입니다.

```
node['opsworks_java']['tomcat]['ajp_port']
```

apache_tomcat_bind_mod

프록시 모듈(문자열). 기본값은 proxy_http입니다. 이 속성을 재정의하여 AJP 프록시 모듈 proxy_ajp를 지정할 수 있습니다.

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_mod']
```

apache_tomcat_bind_path

Apache-Tomcat 바인딩 경로(문자열). 기본값은 /입니다. 이 속성은 재정의하면 안 됩니다. 바인딩 경로를 변경하면 애플리케이션 작동이 중지될 수 있습니다.

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_path']
```

auto_deploy

autodeploy할지 여부(부울). 기본값은 true입니다.

```
node['opsworks_java']['tomcat]['auto_deploy']
```

connection_timeout

연결 제한 시간(밀리초)(숫자). 기본값은 20000(20초)입니다.

```
node['opsworks_java']['tomcat]['connection_timeout']
```

mysql_connector_jar

MySQL 커넥터 라이브러리의 JAR 파일(문자열). 기본값은 mysql-connector-java.jar입니다.

```
node['opsworks_java']['tomcat]['mysql_connector_jar']
```

포트

표준 포트(숫자). 기본값은 8080입니다.

```
node['opsworks_java']['tomcat]['port']
```

secure_port

보안 포트(숫자). 기본값은 8443입니다.

```
node['opsworks_java']['tomcat]['secure_port']
```

shutdown_port

종료 포트(숫자). 기본값은 8005입니다.

```
node['opsworks_java']['tomcat]['shutdown_port']
```

threadpool_max_threads

스레드 풀의 스레드 최대 수(숫자). 기본값은 150입니다.

```
node['opsworks_java']['tomcat]['threadpool_max_threads']
```

threadpool_min_spare_threads

스레드 풀의 예비 스레드 최소 수(숫자). 기본값은 4입니다.

```
node['opsworks_java']['tomcat]['threadpool_min_spare_threads']
```

unpack_wars

WAR 파일의 압축을 풀지 여부(부울). 기본값은 `true`입니다.

```
node['opsworks_java']['tomcat]['unpack_wars']
```

uri_encoding

URI 인코딩(문자열). 기본값은 `UTF-8`입니다.

```
node['opsworks_java']['tomcat]['uri_encoding']
```

use_ssl_connector

SSL 커넥터를 사용할지 여부(부울). 기본값은 `false`입니다.

```
node['opsworks_java']['tomcat]['use_ssl_connector']
```

use_threadpool

스레드 풀을 사용할지 여부(부울). 기본값은 `false`입니다.

```
node['opsworks_java']['tomcat]['use_threadpool']
```

userdatabase_pathname

사용자 데이터베이스 경로 이름(문자열). 기본값은 `conf/tomcat-users.xml`입니다.

```
node['opsworks_java']['tomcat]['userdatabase_pathname']
```

passenger_apache2 Attributes

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

`passenger_apache2` 속성은 Phusion Passenger 구성을 지정합니다. 자세한 정보는 Phusion Passenger 사용 설명서, Apache 버전 단원을 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

friendly_error_pages	gem_bin	gems_path
high_performance_mode	root_path	max_instances_per_app
max_pool_size	max_requests	module_path
pool_idle_time	rails_app_spawner_idle_time	rails_framework_spawner_idle_time
rails_spawn_method	ruby_bin	ruby_wrapper_bin
stat_throttle_rate	version	

friendly_error_pages

애플리케이션 시작이 실패하는 경우, 익숙한 오류 페이지를 표시할지 여부(문자열). 이 속성은 'on' 또는 'off'로 설정할 수 있으며, 기본 값은 'off'입니다.

```
node[:passenger][:friendly_error_pages]
```

gem_bin

Gem 이진수의 위치(문자열). 기본값은 `'/usr/local/bin/gem'`입니다.

```
node[:passenger][:gem_bin]
```

gems_path

gems 경로(문자열). 기본값은 인스턴스의 Ruby 버전에 따라 다릅니다. 예시:

- Ruby 버전 1.8: `'/usr/local/lib/ruby/gems/1.8/gems'`
- Ruby 버전 1.9: `'/usr/local/lib/ruby/gems/1.9.1/gems'`

```
node[:passenger][:gems_path]
```

high_performance_mode

Passenger의 고성능 모드를 사용할지 여부(문자열). 가능한 값은 `'on'`와 `'off'`입니다. 기본값은 `'off'`입니다.

```
node[:passenger][:high_performance_mode ]
```

root_path

Passenger 루트 디렉터리(문자열). 기본값은 Ruby 및 Passenger 버전에 따라 다릅니다. Chef 구문에서 값은 `"#{node[:passenger][:gems_path]}/passenger-#{passenger[:version]}"`입니다.

```
node[:passenger][:root_path]
```

max_instances_per_app

앱당 애플리케이션 프로세스 최대 수(숫자). 기본값은 `0`입니다. 자세한 정보는 PassengerMaxInstancesPerApp 단원을 참조하세요.

```
node[:passenger][:max_instances_per_app]
```

max_pool_size

애플리케이션 프로세서의 최대 수(숫자). 기본값은 `8`입니다. 자세한 정보는 PassengerMaxPoolSize를 참조하세요.

```
node[:passenger][:max_pool_size]
```

max_requests

요청의 최대 수(숫자). 기본값은 `0`입니다.

```
node[:passenger][:max_requests]
```

module_path

모듈 경로(문자열). 기본값은 다음과 같습니다.

- Amazon Linux 및 RHEL: `"#{node['apache']['libexecdir']}/mod_passenger.so"`
- Ubuntu: `"#{passenger[:root_path]}/ext/apache2/mod_passenger.so"`

```
node[:passenger][:module_path]
```

pool_idle_time

애플리케이션 프로세스가 유휴 상태로 있을 수 있는 최대 시간(초)(숫자). 기본값은 `14400`(4시간)입니다. 자세한 정보는 PassengerPoolIdleTime 단원을 참조하세요.

```
node[:passenger][:pool_idle_time]
```

rails_app_spawner_idle_time

Rails 앱 스포너의 최대 유휴 시간(숫자). 이 속성이 `0`으로 설정되면 앱 스포너는 제한 시간이 없습니다. 기본값은 `0`입니다. 자세한 정보는 Spawning Methods Explained를 참조하세요.

```
node[:passenger][:rails_app_spawner_idle_time]
```

rails_framework_spawner_idle_time

Rails 프레임워크 스포너의 최대 유휴 시간(숫자). 이 속성이 `0`으로 설정되면 프레임워크 스포너는 제한 시간이 없습니다. 기본값은 `0`입니다. 자세한 정보는 Spawning Methods Explained를 참조하세요.

```
node[:passenger][:rails_framework_spawner_idle_time]
```

rails_spawn_method

Rails 복제 메서드(문자열). 기본값은 `'smart-lv2'`입니다. 자세한 정보는 Spawning Methods Explained를 참조하세요.

```
node[:passenger][:rails_spawn_method]
```

ruby_bin

Ruby 이진수의 위치(문자열). 기본값은 '/usr/local/bin/ruby'입니다.

```
node[:passenger][:ruby_bin]
```

ruby_wrapper_bin

Ruby 래퍼 스크립트의 위치(문자열). 기본값은 '/usr/local/bin/ruby_gc_wrapper.sh'입니다.

```
node[:passenger][:ruby_wrapper_bin]
```

stat_throttle_rate

Passenger가 파일 시스템 검사를 수행하는 속도(숫자). 기본값은 5이며, 이는 검사가 최대 5초마다 수행됨을 뜻합니다. 자세한 정보는 PassengerStatThrottleRate를 참조하세요.

```
node[:passenger][:stat_throttle_rate]
```

version

버전(문자열). 기본값은 '3.0.9'입니다.

```
node[:passenger][:version]
```

ruby 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

ruby 속성은 애플리케이션이 사용하는 Ruby 버전을 지정합니다. Ruby 2.1에서 의미 체계 버전 관리가 도입되면서 속성 사용이 바뀌었습니다. 예제를 포함한 버전 지정 방법에 대한 자세한 정보는 Ruby 버전를 참조하세요. AWS OpsWorks Stacks가 Ruby 버전을 결정하는 방법에 대한 자세한 내용은 기본 제공 속성 파일인 ruby.rb를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하십시오.

full_version

정품 버전 번호(문자열). 이 속성은 재정의해서는 안 됩니다. 그 대신 [:opsworks][:ruby_version]과 적절한 패치 버전 속성을 사용하여 버전을 지정하세요.

```
[:ruby][:full_version]
```

major_version

메이저 버전 번호(문자열). 이 속성은 재정의해서는 안 됩니다. 그 대신 [:opsworks][:ruby_version]을 사용하여 메이저 버전을 지정하세요.

```
[:ruby][:major_version]
```

minor_version

마이너 버전 번호(문자열). 이 속성은 재정의해서는 안 됩니다. 그 대신 [:opsworks][:ruby_version]을 사용하여 마이너 버전을 지정하세요.

```
[:ruby][:minor_version]
```

패치

패치 수준(문자열). 이 속성은 Ruby 버전 2.0.0 및 이전 버전에 유효합니다. 이후의 Ruby 버전은 `patch_version` 속성을 사용합니다.

```
[:ruby][:patch]
```

패치 번호는 p로 시작해야 합니다. 예를 들어 다음 사용자 지정 JSON을 사용하여 패치 수준 484를 지정합니다.

```
{
  "ruby":{"patch":"p484"}
}
```

patch_version

패치 번호(문자열). 이 속성은 Ruby 버전 2.1 및 이후 버전에 유효합니다. 그 이전의 Ruby 버전은 `patch` 속성을 사용합니다.

```
[:ruby][:patch_version]
```

pkgrelease

패키지 릴리스 번호(문자열).

```
[:ruby][:pkgrelease]
```

unicorn 속성

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

이러한 속성은 Linux 스택에서만 사용할 수 있습니다.

`unicorn` 속성은 Unicorn 구성을 지정합니다. 자세한 정보는 Unicorn::Configurator를 참조하세요. 내장 속성을 재정의해 사용자 지정 값을 지정하는 방법에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

accept_filter	backlog	delay

tcp_nodelay	tcp_nopush	preload_app
제한 시간	tries	version
worker_processes		

accept_filter

accept 필터, `'httpready'` 또는 `'dataready'`(문자열). 기본값은 `'httpready'`입니다.

```
node[:unicorn][:accept_filter]
```

backlog

대기열이 보관할 수 있는 요청의 최대 수(숫자). 기본값은 `1024`입니다.

```
node[:unicorn][:backlog]
```

delay

소켓 바인딩을 재시도하기 위해 기다릴 시간(초)(숫자). 기본값은 `0.5`입니다.

```
node[:unicorn][:delay]
```

preload_app

worker 프로세스 포킹 전에 앱을 사전 로드할지 여부(부울). 기본값은 `true`입니다.

```
node[:unicorn][:preload_app]
```

tcp_nodelay

TCP 소켓에 대한 Nagle의 알고리즘을 비활성화할지 여부(부울). 기본값은 `true`입니다.

```
node[:unicorn][:tcp_nodelay]
```

tcp_nopush

TCP_CORK를 활성화할지 여부(부울). 기본값은 `false`입니다.

```
node[:unicorn][:tcp_nopush]
```

제한 시간

각 요청마다 worker 사용이 허용되는 최대 시간(초)(숫자). 제한 시간 값을 초과하는 worker는 종료됩니다. 기본값은 60입니다.

```
node[:unicorn][:timeout]
```

tries

소켓에 대한 바인딩을 재시도할 최대 횟수(숫자). 기본값은 5입니다.

```
node[:unicorn][:tries]
```

version

Unicorn 버전(문자열). 기본값은 '4.7.0'입니다.

```
node[:unicorn][:version]
```

worker_processes

worker 프로세스의 수(숫자). 기본값은 존재하는 경우, max_pool_size이고 그렇지 않은 경우에는 4입니다.

```
node[:unicorn][:worker_processes]
```

Linux용 Chef 11.10 및 이전 버전 문제 해결

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 추가 문제 해결 정보는 디버깅 및 문제 해결 안내서 단원을 참조하세요.

Linux용 Chef 11.10 및 이전 버전의 Chef 로그

AWS OpsWorks Stacks는 각 인스턴스의 Chef 로그를 `/var/lib/aws/opsworks/chef` 디렉터리에 저장합니다. 이 디렉터리에 액세스하려면 sudo 권한이 필요합니다. 각 실행에 대한 로그 파일은 *YYYY*-*MM*-*DD*-*HH*-*MM*-*SS*-*NN*.log 파일에 저장됩니다.

자세한 내용은 다음을 참조하세요.

- 콘솔을 사용하여 Chef 로그 보기
- CLI 또는 API를 사용하여 Chef 로그 보기
- Chef 로그 해석
- 일반적인 Chef 로그 오류

다른 AWS 서비스와 함께 AWS OpsWorks Stacks 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Stacks 스택에서 실행되는 애플리케이션 서버가 AWS OpsWorks Stacks와 직접 통합되지 않은 다양한 AWS 서비스를 사용하도록 AWS OpsWorks 할 수 있습니다. 예를 들어 애플리케이션 서버가 Amazon RDS를 백엔드 데이터베이스로 사용하게 할 수 있습니다. 다음의 일반 패턴을 사용하여 그러한 서비스에 액세스할 수 있습니다.

1. AWS console, API 또는 CLI를 사용하여 AWS 서비스를 생성하고 구성한 다음, 애플리케이션이 해당 서비스에 액세스하는 데 필요한 필수 구성 데이터(예: 호스트 이름 또는 포트)를 기록합니다.
2. 애플리케이션을 서비스에 액세스할 수 있도록 구성하기 위한 하나 이상의 사용자 지정 레시피를 생성합니다.

레시피는 사용자가 레시피를 실행하기 전에 사용자 지정 JSON을 사용하여 정의하는 스택 구성 및 배포 JSON 속성으로부터 구성 데이터를 가져옵니다.

3. 애플리케이션 서버 계층의 Deploy 수명 주기 이벤트에 사용자 지정 레시피를 할당합니다.
4. 구성 데이터 속성에 적절한 값을 할당하는 사용자 지정 JSON 객체를 생성하여 스택 구성 및 배포 JSON에 추가합니다.
5. 애플리케이션을 스택에 배포합니다.

   배포는 사용자 지정 JSON에서 정의된 구성 데이터 값을 사용하여 애플리케이션을 서비스에 액세스할 수 있도록 구성하는 사용자 지정 레시피를 실행합니다.

이 섹션에서는 AWS OpsWorks Stacks 애플리케이션 서버가 다양한 AWS 서비스에 액세스하도록 하는 방법을 설명합니다. 여기서는 사용자가 이미 Chef 쿡북에 대해, 또한 어떻게 레시피가 스택 및 구성 JSON 속성을 사용하여 애플리케이션을 구성할 수 있는지(일반적으로 구성 파일을 생성)에 대해 잘 알고 있는 것으로 가정합니다. 그렇지 않다면 먼저 쿡북과 레시피 및 AWS OpsWorks 스택 사용자 지정 섹션을 읽으십시오.

주제

- 백엔드 데이터 스토어 사용
- ElastiCache Redis를 인 메모리 키-값 저장소로 사용
- Amazon S3 버킷 사용하기
- AWS OpsWorks Stacks AWS CodePipeline 에서 사용

백엔드 데이터 스토어 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

애플리케이션 서버 스택에는 일반적으로 백엔드 데이터 스토어를 제공하는 데이터베이스 서버가 포함됩니다. AWS OpsWorks Stacks는 MySQL 계층을 통해 MySQL 서버에 대한 통합 지원을 제공하고

Amazon Relational Database Service(Amazon RDS) 계층을 통해 여러 유형의 데이터베이스 서버에 대한 통합 지원을 제공합니다. 하지만 애플리케이션 서버가 Amazon DynamoDB 또는 MongoDB 등 다른 데이터베이스 서버를 사용하도록 스택을 간편하게 사용자 지정할 수 있습니다. 이 주제에서는 애플리케이션 서버를 AWS 데이터베이스 서버에 연결하는 기본 절차를 설명합니다. Chef 11 Linux 스택 시작하기의 스택과 애플리케이션을 사용하여 PHP 애플리케이션 서버를 RDS 데이터베이스에 수동으로 연결하는 방법을 살펴봅니다. 예제는 Linux 스택에 기반하고 있지만 기본 원칙은 Windows 스택에도 적용됩니다. MongoDB 데이터베이스 서버를 스택에 통합하는 방법의 예제는 OpsWorks를 사용하여 MongoDB 배포를 참조하세요.

> **Note**
>
> 이 주제에서 편리한 예시로 Amazon RDS를 사용합니다. 하지만 스택에 Amazon RDS 데이터베이스를 사용하려면 Amazon RDS 계층을 사용하는 것이 훨씬 쉽습니다.

주제

- 데이터베이스 연결 설정 방법
- 애플리케이션 서버 인스턴스를 Amazon RDS에 연결하는 방법

데이터베이스 연결 설정 방법

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 레시피를 사용하여 애플리케이션 서버와 백엔드 데이터베이스 간 연결을 설정합니다. 레시피는 일반적으로 구성 파일을 생성하여 필요에 따라 애플리케이션을 구성합니다. 레시피는 스택 구성의 속성 집합과 Stacks가 모든 인스턴스에 설치하는 배포 속성에서 호스트 및 데이터베이스 이름과 같은 연결 데이터를 가져옵니다. AWS OpsWorks

예를 들어 Chef 11 Linux 스택 시작하기의 2단계는 각각 인스턴스가 하나씩 있는 PHP 앱 서버 및 MySQL이라는 2개의 계층을 가진 MyStack이라는 스택에 기반합니다. MySQL 인스턴스에서 데이터베이스를 백엔드 데이터스토어로 사용하는 PHP 앱 서버 인스턴스에 SimplePHPApp이라는 앱을 배포합

니다. 애플리케이션을 배포하면 AWS OpsWorks Stacks는 데이터베이스 연결 정보가 포함된 스택 구성 및 배포 속성을 설치합니다. 다음 예제는 JSON으로 표시되는 데이터베이스 연결 속성을 보여 줍니다.

```
{
  ...
  "deploy": {
    "simplephpapp": {
      ...
      "database": {
        "reconnect": true,
        "password": null,
        "username": "root",
        "host": null,
        "database": "simplephpapp"
        ...
      },
      ...
    }
  }
}
```

속성 값은 AWS OpsWorks Stacks에서 제공하며 사용자가 제공한 정보를 기반으로 생성되거나 생성됩니다.

SimplePHPApp이 데이터 스토어에 액세스하도록 허용하려면 `appsetup.rb`(이)라는 이름의 사용자 지정 레시피를 PHP 앱 서버 계층의 Deploy 수명 주기 이벤트에 할당하여 PHP 애플리케이션 서버와 MySQL 데이터베이스 간 연결을 설정해야 합니다. SimplePHPApp을 배포하면 AWS OpsWorks Stacks는 다음 발췌문과 같이 연결을 `db-connect.php` 설정하는 라는 구성 파일을 `appsetup.rb`생성하는를 실행합니다.

```
node[:deploy].each do |app_name, deploy|
  ...
  template "#{deploy[:deploy_to]}/current/db-connect.php" do
    source "db-connect.php.erb"
    mode 0660
    group deploy[:group]

    if platform?("ubuntu")
```

```
      owner "www-data"
    elsif platform?("amazon")
      owner "apache"
    end

    variables(
      :host =>     (deploy[:database][:host] rescue nil),
      :user =>     (deploy[:database][:username] rescue nil),
      :password => (deploy[:database][:password] rescue nil),
      :db =>       (deploy[:database][:database] rescue nil),
      :table =>    (node[:phpapp][:dbtable] rescue nil)
    )
    ...
  end
end
```

연결을 특징짓는 변수(host, user 등)는 배포 JSON의 [:deploy][:app_name][:database] 속성에서 해당 값을 설정합니다. 간단히 하기 위해 예제에서는 이미 urler라는 이름의 테이블을 생성했다고 가정합니다. 따라서 테이블 이름은 쿡북의 속성 파일에서 [:phpapp][:dbtable]로 나타납니다.

이 레시피는 실제로 PHP 애플리케이션 서버를, MySQ 계층의 멤버만이 아니라 어떤 MySQL 데이터베이스 서버에도 연결할 수 있습니다. 다른 MySQL 서버를 사용하려면 [:database] 속성을 서버에 적합한 값으로 설정하기만 하면 됩니다.이 값은 사용자 지정 JSON을 사용하여 설정할 수 있습니다. AWS OpsWorks Stacks는 이러한 속성과 값을 스택 구성 및 배포 속성에 통합하고 이를 appsetup.rb 사용하여 연결을 설정하는 템플릿을 생성합니다. 스택 구성 및 배포 JSON 재정의에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

애플리케이션 서버 인스턴스를 Amazon RDS에 연결하는 방법

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 PHP 애플리케이션 서버가 RDS 인스턴스에 연결되도록 Chef 11 Linux 스택 시작하기의 MyStack을 사용자 지정하는 방법을 살펴봅니다.

주제

- Amazon RDS MySQL 데이터베이스를 생성합니다.
- RDS 데이터베이스에 연결하도록 스택을 사용자 지정

Amazon RDS MySQL 데이터베이스를 생성합니다.

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이제 Amazon RDS 콘솔의 DB 인스턴스 시작 마법사를 사용하여 예제용 RDS 데이터베이스를 생성할 준비가 되었습니다. 다음 절차는 필수적 세부 사항의 간략한 요약입니다. 데이터베이스 생성 방법에 대한 자세한 설명은 Amazon RDS 시작하기를 참조하세요.

Amazon RDS 데이터베이스를 생성하려면

1. RDS 데이터베이스를 처음 사용하는 경우 [지금 시작하기]를 클릭합니다. 생성해 본 적이 있으면 탐색 창에서 [RDS 대시보드]를 클릭한 다음, [DB 인스턴스 시작]을 클릭합니다.
2. [MySQL 커뮤니티 에디션]을 DB 인스턴스로 선택합니다.
3. 프로덕션 목적으로 이 데이터베이스를 사용할 계획입니까?에서 No, this instance...(아니요, 이 인스턴스...)를 선택합니다. 이 예에서는 이 옵션이면 충분합니다. 프로덕션용으로 사용할 경우 [예, 다중 AZ 배포 사용...]를 선택합니다. [다음 단계]를 클릭합니다.
4. [DB 세부 정보 지정] 페이지에서 다음 설정을 지정합니다.
   - [DB 인스턴스 클래스]: [db.t2.micro]
   - [다중 AZ 배포]: [No]
   - 할당된 스토리지: **5** GB
   - DB 인스턴스 식별자: **`rdsexample`**
   - 마스터 사용자 이름: **`opsworksuser`**.
   - [마스터 암호]: 적절한 암호를 지정하고 나중에 사용하기 위해 적어 둡니다.

다른 옵션에 대해서는 기본 설정을 수락하고 [다음 단계]를 클릭합니다.

5. [고급 설정 구성] 페이지에서 다음 설정을 지정합니다.

   - [네트워크 및 보안] 섹션에서 [VPC 보안 그룹]에 대해 [phpsecgroup (VPC)]을 선택합니다.
   - 데이터베이스 옵션 섹션에서 데이터베이스 이름으로 **`rdsexampledb`**를 입력합니다.
   - 이 연습을 위해 [백업] 섹션에서 [백업 보관 기간]을 [0]으로 설정합니다.

   다른 옵션에 대해서는 기본 설정을 수락하고 [DB 인스턴스 시작]을 클릭합니다.

6. [DB 인스턴스 보기]를 선택하여 DB 인스턴스 목록을 확인합니다.

7. 이 목록에서 rdsexample 인스턴스를 선택하고 화살표를 클릭하여 인스턴스 엔드포인트 및 기타 세부 정보를 표시합니다. 나중에 사용하기 위해 이 엔드포인트를 적어 둡니다. 이 엔드포인트는 `rdsexample.c6c8mntzhgv0.us-west-2.rds.amazonaws.com:3306`과 유사합니다. DNS 이름만 적어둡니다. 포트 번호는 필요 없습니다.

8. MySQL Workbench와 같은 도구에서 다음 SQL 명령을 사용하여 `urler` 데이터베이스에 `rdsexampledb`라는 테이블을 만듭니다.

```
CREATE TABLE urler(id INT UNSIGNED NOT NULL AUTO_INCREMENT,author VARCHAR(63) NOT
 NULL,message TEXT,PRIMARY KEY (id))
```

RDS 데이터베이스에 연결하도록 스택을 사용자 지정

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

PHP 애플리케이션 서버의 백엔드 데이터베이스로 사용할 RDS 인스턴스를 생성한 다음 Chef 11 Linux 스택 시작하기의 MyStack을 사용자 지정할 수 있습니다.

RDS 데이터베이스에 PHP 앱 서버를 연결하려면

1. 에 설명된 대로 AWS OpsWorks Stacks 콘솔을 열고 인스턴스 하나가 포함된 PHP 앱 서버 계층으로 스택을 생성하고 SimplePHPApp을 배포합니다Chef 11 Linux 스택 시작하기. 이 스택은 데이터베이스 연결을 사용하지 않는 SimplePHPApp의 version1을 사용합니다.
2. `appsetup.rb` 레시피가 포함된 사용자 지정 쿡북과 관련 템플릿 및 속성 파일을 사용하도록 스택 구성을 업데이트하세요.

   1. [사용자 지정 Chef 쿡북 사용]을 [예]로 설정합니다.
   2. [리포지토리 유형]을 [Git]로, [리포지토리 URL]을 `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`로 설정합니다.
3. 스택의 [사용자 지정 Chef JSON] 상자에 아래 내용을 추가하여 `appsetup.rb`를 사용하여 구성 파일을 생성하는 `[:database]` 속성에 RDS 연결 데이터를 할당합니다.

```
{
  "deploy": {
    "simplephpapp": {
      "database": {
        "username": "opsworksuser",
        "password": "your_password",
        "database": "rdsexampledb",
        "host": "rds_endpoint",
        "adapter": "mysql"
      }
    }
  }
}
```

   다음 속성 값을 사용합니다.

   - [사용자 이름]: RDS 인스턴스를 생성할 때 지정한 마스터 사용자 이름.

     이 예제에서는 `opsworksuser`를 사용합니다.
   - [암호]: RDS 인스턴스를 생성할 때 지정한 마스터 암호.

     지정한 암호를 입력합니다.
   - [데이터베이스]: RDS 인스턴스를 생성할 때 생성한 데이터베이스.

     이 예제에서는 `rdsexampledb`를 사용합니다.

- [호스트]: RDS 인스턴스의 엔드포인트로, 이전 단원에서 인스턴스를 생성할 때 RDS 콘솔에서 확인했습니다. 포트 번호를 포함해서는 안 됩니다.
- [어댑터]: 어댑터.

  이 예제의 RDS 인스턴스는 MySQL을 사용하므로 [어댑터]를 [`mysql`]로 설정합니다. 다른 속성과 달리 [어댑터]는 `appsetup.rb`에서 사용하지 않습니다. 대신 PHP 앱 서버 계층의 내장 Configure 레시피에서 다른 구성 파일을 생성하는 데 사용합니다.

4. SimplePHPApp 구성을 편집하여 다음과 같이 백엔드 데이터베이스를 사용하는 SimplePHPApp의 버전을 지정합니다.

   - [문서 루트]: 이 옵션은 `web`으로 설정합니다.
   - [분기/개정]: 이 옵션은 [`version2`]로 설정합니다.

   나머지 옵션은 변경하지 않고 그대로 둡니다.

5. 계층의 Deploy 레시피에 `phpapp::appsetup`을(를) 추가하여 데이터베이스 연결을 설정하려면 PHP 앱 서버 계층을 편집합니다.
6. 새 SimplePHPApp 버전을 배포합니다.
7. SimplePHPApp이 배포되면 [인스턴스] 페이지로 이동한 다음 php-app1 인스턴스의 퍼블릭 IP 주소를 클릭하여 애플리케이션을 실행합니다. 브라우저에 다음 페이지가 표시되어야 하고, 여기서 텍스트를 입력하면 데이터베이스에 저장됩니다.

> **Note**
>
> 스택에 MySQL 계층이 있는 경우 AWS OpsWorks Stacks는 `[:database]` 속성에 해당 연결 데이터를 자동으로 할당합니다. 하지만 다른 `[:database]` 값을 정의하는 사용자 지정 JSON을 스택에 할당하는 경우, 기본값이 재정의됩니다. `[:deploy]` 속성은 모든 인스턴스에 설치되기 때문에 `[:database]` 속성에 의존하는 모든 레시피는 MySQL 계층 데이터가 아니라 사용자 지정 연결 데이터를 사용합니다. 특정 애플리케이션 서버 계층이 사용자 지정 연결 데이터를 사용하게 하려면 사용자 지정 JSON을 계층의 Deploy 이벤트에 할당하고 배포를 해당 계층으로 제한하세요. 배포 속성을 사용하는 방법에 대한 자세한 정보는 앱 배포를 참조하세요. AWS OpsWorks Stacks 내장 속성 재정의에 대한 자세한 정보는 속성 재정의 단원을 참조하세요.

ElastiCache Redis를 인 메모리 키-값 저장소로 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 주제는 Linux 스택을 기반으로 하지만 Windows 스택도 Amazon ElastiCache(ElastiCache)를 사용할 수 있습니다. Windows 인스턴스에서 ElastiCache를 사용하는 방법의 예는 ElastiCache as an ASP.NET Session Store를 참조하세요.

캐싱 서버를 사용하여 문자열과 같은 작은 데이터 항목을 위한 인메모리 키-값 저장소를 제공함으로써 애플리케이션 서버 성능을 향상시킬 수 있는 경우가 많습니다. Amazon ElastiCache는 Memcached 또는 Redis 캐싱 엔진을 사용하여 애플리케이션 서버에 대한 캐싱 지원을 쉽게 제공할 수 있는 AWS 서비스입니다. AWS OpsWorks Stacks는 Memcached에 대한 기본 지원을 제공합니다. 하지만 Redis가 요구 사항에 더 적합하다면 애플리케이션 서버가 ElastiCache Redis를 사용하도록 스택을 사용자 지정할 수 있습니다.

이 주제에서는 Rails 애플리케이션 서버를 예제로 사용하여 Linux 스택에 ElastiCache Redis 캐싱 지원을 제공하는 기본적 프로세스를 단계별로 살펴봅니다. 여기서는 이미 적절한 Ruby on Rails 애플리케이션이 있는 것으로 가정합니다. ElastiCache에 대한 자세한 내용은 Amazon ElastiCache란 무엇입니까?를 참조하세요.

주제

- 1단계: ElastiCache Redis 클러스터 생성
- 2단계: Rails 스택 설정
- 3단계: 사용자 지정 쿡북 생성 및 배포
- 4단계: 수명 주기 이벤트에 레시피 할당
- 5단계: 스택 구성 JSON에 액세스 정보 추가
- 6단계: 앱 배포 및 실행

1단계: ElastiCache Redis 클러스터 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

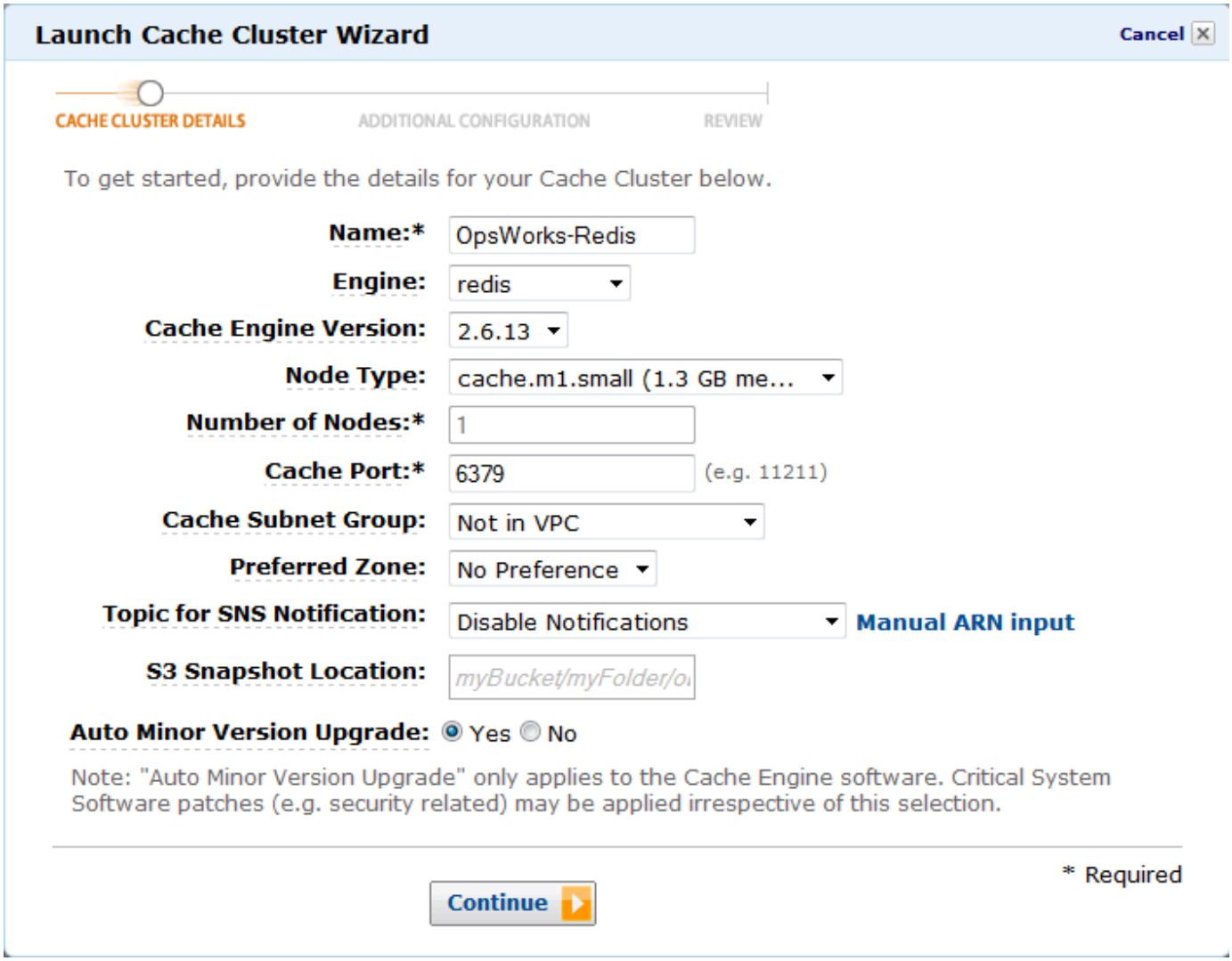
먼저 ElastiCache 콘솔, API 또는 CLI를 사용하여 Amazon ElastiCache Redis 클러스터를 생성해야 합니다. 이제부터 콘솔을 사용하여 클러스터를 생성하는 방법을 설명합니다.

ElastiCache Redis 클러스터를 생성합니다.

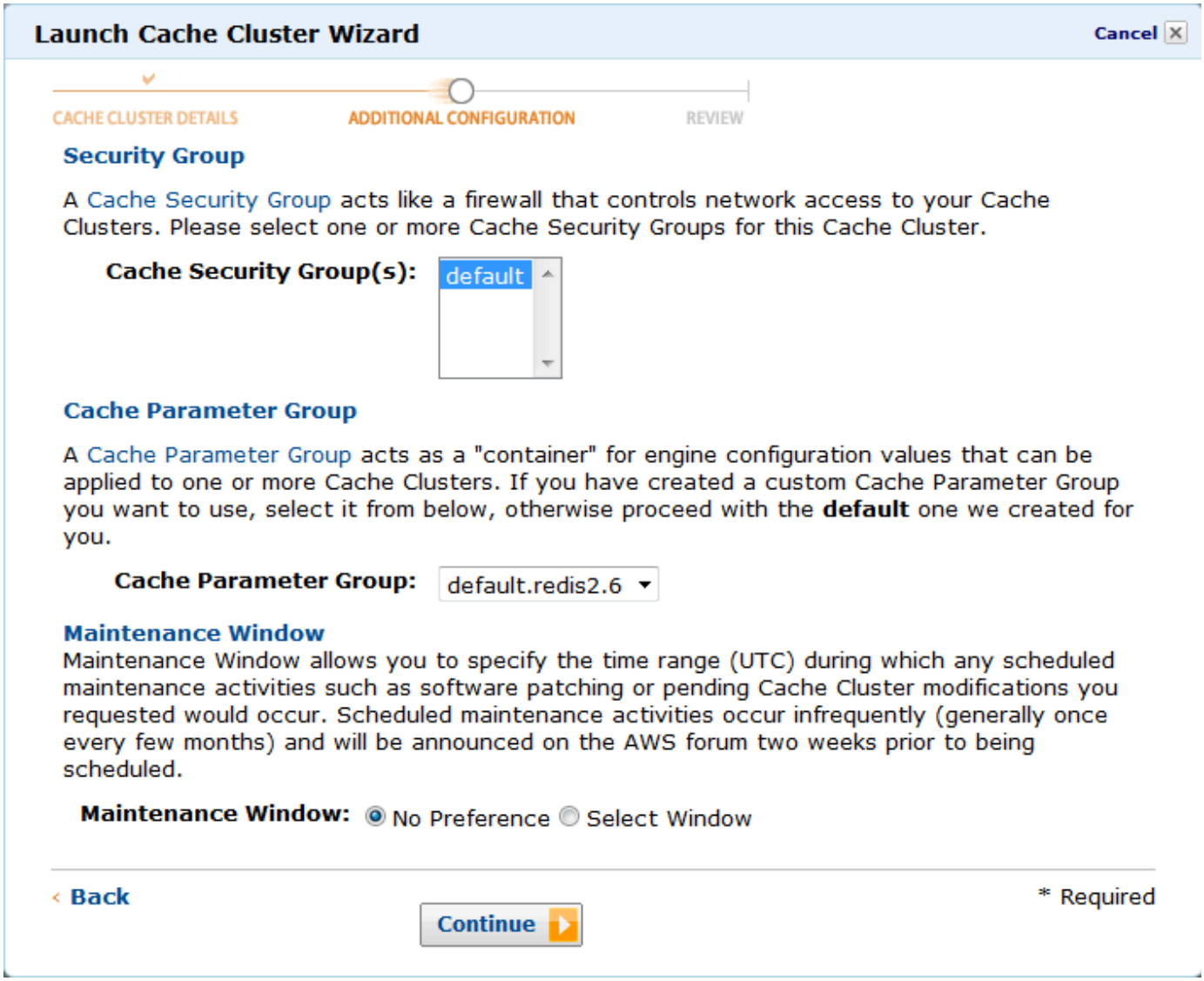
1. ElastiCache 콘솔로 이동하여 캐시 클러스터 시작를 클릭해 캐시 클러스터 마법사를 시작합니다.
2. [캐시 클러스터 세부 정보] 페이지에서 다음 작업을 수행합니다.
   - [이름]을 캐시 서버 이름으로 설정합니다.

     이 예제에서는 OpsWorks-Redis를 사용합니다.
   - [엔진]을 [redis]로 설정합니다.
   - [SNS 알림 항목]을 [알림 비활성화]로 설정합니다.

- 다른 설정에 대해 기본값을 수락하고 [계속]을 클릭합니다.



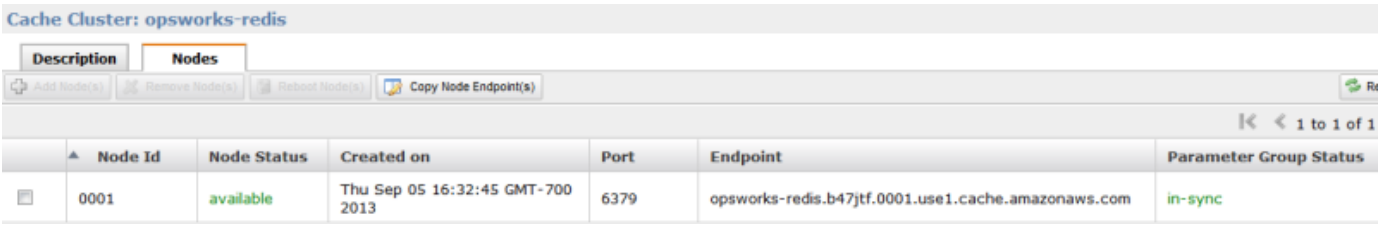
3. [추가 구성] 페이지에서 기본값을 수락하고 [계속]을 클릭합니다.



4. [캐시 클러스터 시작]을 클릭하여 클러스터를 생성합니다.

> **Important**
>
> 이 예제에서는 기본 캐시 보안 그룹이면 충분하지만 프로덕션 환경에서 사용하는 경우 환경에 맞는 보안 그룹을 하나 생성해야 합니다. 자세한 내용은 캐시 보안 그룹 관리를 참조하세요.

5. 클러스터가 시작되면 이름을 클릭하여 세부 정보 페이지를 열고 [노드] 탭을 클릭합니다. 나중에 사용하기 위해 클러스터의 [포트] 및 [엔드포인트] 값을 적어 둡니다.



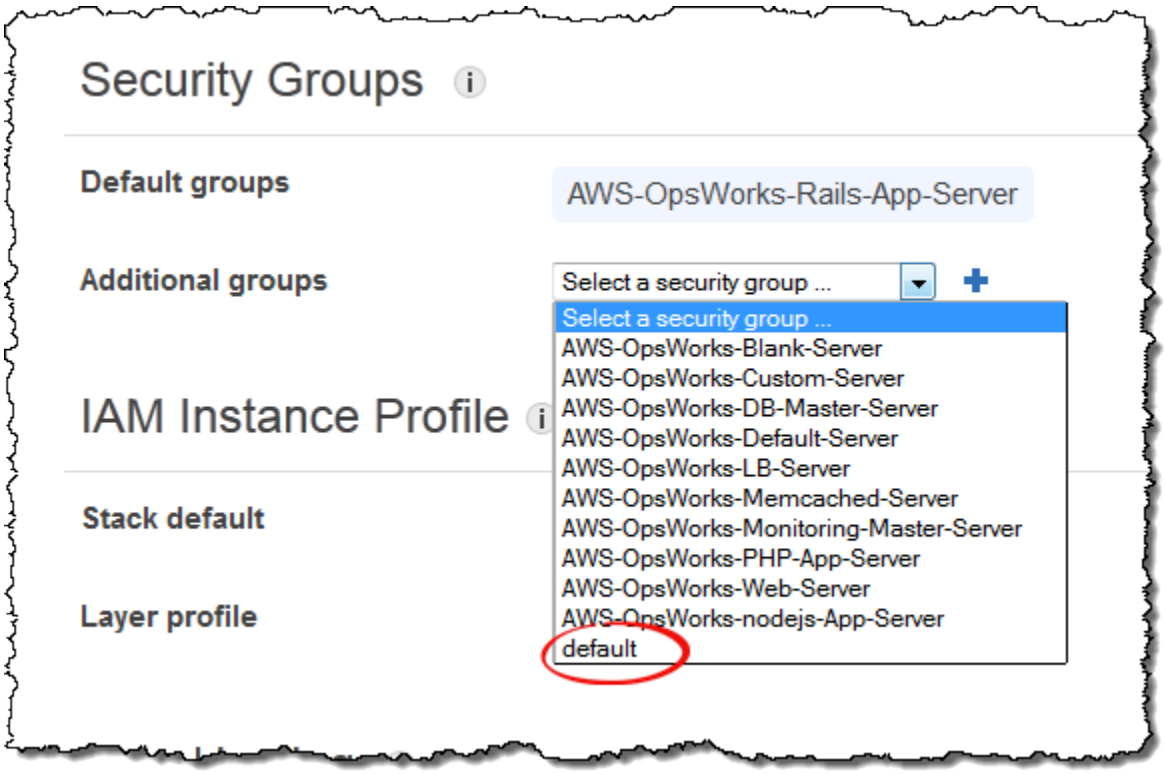
2단계: Rails 스택 설정

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Rails 앱 서버 계층을 지원하는 스택을 생성하는 이외에 Rails 서버가 Redis 서버와 올바로 통신할 수 있도록 계층의 보안 그룹도 구성해야 합니다.

스택을 설정하려면

1. 이 예제에서 이름을 따온 새 **`RedisStack`** 스택을 만들고 Rails 앱 서버 계층을 추가합니다. 새 스택과 계층 둘 다에 기본 설정을 사용할 수 있습니다. 자세한 내용은 새 스택 생성 및 OpsWorks 계층 생성 단원을 참조하세요.
2. 계층 페이지에서 Rails 앱 서버에 대해 보안을 클릭하고 편집을 클릭합니다.
3. 보안 그룹 섹션으로 이동하여 ElastiCache 클러스터의 보안 그룹을 추가 그룹에 추가합니다. 이 예제에서는 [기본] 보안 그룹을 선택하고 [+]를 클릭하여 계층에 이 보안 그룹을 추가한 다음 [저장]을 클릭하여 새 구성을 저장합니다.



4. 인스턴스를 Rails 앱 서버 계층에 추가하고 시작합니다. 인스턴스를 추가 및 시작하는 방법에 대한 자세한 정보는 계층에 인스턴스 추가 단원을 참조하세요.

3단계: 사용자 지정 쿡북 생성 및 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

현재 상태로는 스택이 아직 제 기능을 발휘할 수 없습니다. 애플리케이션이 Redis 서버에 액세스할 수 있도록 설정해야 합니다. 가장 유연한 방법은 액세스 정보를 포함하는 YAML 파일을 애플리케이션의 `config` 하위 폴더에 배치하는 것입니다. 그러면 애플리케이션이 파일에서 정보를 가져올 수 있습니다. 이 방법을 사용하면 애플리케이션을 재작성 및 재배포하지 않아도 연결 정보를 변경할 수 있습니다. 이 예제에서는 파일의 이름을 `redis.yml`이라고 지정하고, 다음과 같이 ElastiCache 클러스터의 호스트 이름 및 포트를 포함해야 합니다.

```
host: cache-cluster-hostname
port: cache-cluster-port
```

이 파일을 서버에 수동으로 복사할 수 있지만 더 좋은 방법은 Chef 레시피를 구현하여 파일을 생성하고 AWS OpsWorks Stacks가 모든 서버에서 레시피를 실행하도록 하는 것입니다. Chef 레시피는 AWS OpsWorks Stacks가 패키지 설치 또는 구성 파일 생성과 같은 인스턴스 작업을 수행하는 데 사용하는 특수 Ruby 애플리케이션입니다. 레시피는 여러 레시피와 구성 파일 템플릿 같은 관련 파일을 포함할 수 있는 쿡북에 패키징되어 있습니다. 쿡북은 GitHub 같은 리포지토리에 배치되며, 표준 디렉터리 구조를 가져야 합니다. 사용자 지정 쿡북 리포지토리가 없는 경우, 쿡북 리포지토리에서 설정 방법 단원을 참조하세요.

이 예제에서는 다음 콘텐츠를 포함하는 `redis-config`라는 쿡북을 쿡북 리포지토리에 추가합니다.

```
my_cookbook_repository
  redis-config
    recipes
      generate.rb
    templates
      default
        redis.yml.erb
```

`recipes` 폴더에는 다음과 같이 `generate.rb`로부터 애플리케이션의 구성 파일을 생성하는 레시피 `redis.yml.erb`가 들어 있습니다.

```
node[:deploy].each do |app_name, deploy_config|
  # determine root folder of new app deployment
  app_root = "#{deploy_config[:deploy_to]}/current"

  # use template 'redis.yml.erb' to generate 'config/redis.yml'
  template "#{app_root}/config/redis.yml" do
    source "redis.yml.erb"
    cookbook "redis-config"

    # set mode, group and owner of generated file
    mode "0660"
    group deploy_config[:group]
    owner deploy_config[:user]
```

```
    # define variable “@redis” to be used in the ERB template
    variables(
      :redis => deploy_config[:redis] || {}
    )

    # only generate a file if there is Redis configuration
    not_if do
      deploy_config[:redis].blank?
    end
  end
end
```

레시피는 각 인스턴스에 설치되고 스택 및 배포된 앱에 대한 자세한 정보를 포함하는 AWS OpsWorks Stacks 스택 구성 및 배포 JSON 객체의 데이터에 따라 달라집니다. 스택 구성 및 배포 속성 이 객체의 deploy 노드는 다음과 같은 구조입니다.

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

배포 노드에는 배포된 각 앱마다 앱의 짧은 이름으로 명명된 포함된 JSON 객체 세트가 포함됩니다. 각각의 앱 객체에는 문서 루트와 애플리케이션 유형 같은 앱의 구성을 정의하는 속성 세트가 포함됩니다. 배포 속성의 목록은 deploy 속성 단원을 참조하세요. 레시피는 Chef 속성 구문을 사용하여 스택 구성 및 배포 JSON 값을 나타낼 수 있습니다. 예를 들어 [:deploy][:app1][:application]은 app1 애플리케이션의 짧은 이름을 나타냅니다.

[:deploy] 내 각 앱에 대해 레시피가 연결된 코드 블록을 실행합니다. 여기서 deploy_config는 앱 속성을 나타냅니다. 먼저 레시피는 app_root를 앱의 루트 디렉터리 [:deploy][:*app_name*]

[:deploy_to]/current로 설정합니다. 그런 다음 Chef 템플릿 리소스를 사용하여 redis.yml.erb로부터 구성 파일을 생성하고 app_root/config에 저장합니다.

구성 파일은 일반적으로 템플릿으로부터 생성되며, 이 경우 설정이 대부분 Chef 속성에 의해 정의됩니다. 속성의 경우, 나중에 설명하듯이 템플릿 파일을 재작성하지 않고 사용자 지정 JSON을 사용하여 설정을 변경할 수 있습니다. redis.yml.erb 템플릿에는 다음 항목이 포함됩니다.

```
host: <%= @redis[:host] %>
port: <%= @redis[:port] || 6379 %>
```

<%... %> 요소는 속성 값을 나타내는 자리 표시자입니다.

- <%= @redis[:host] %>는 캐시 클러스터의 호스트 이름을 나타내는 redis[:host]의 값입니다.
- <%= @redis[:port] || 6379 %>는 redis[:port]의 값을 나타내며, 해당 속성이 정의되지 않은 경우에는 기본 포트 값 6379입니다.

template 리소스는 다음과 같이 작동합니다.

- source 및 cookbook은 각각 템플릿 이름과 쿡북 이름을 지정합니다.
- mode, group 및 owner는 구성 파일에 애플리케이션과 동일한 액세스 권한을 부여합니다.
- variables 섹션은 템플릿에 사용되는 @redis 변수를 애플리케이션의 [:redis] 속성 값으로 설정합니다.

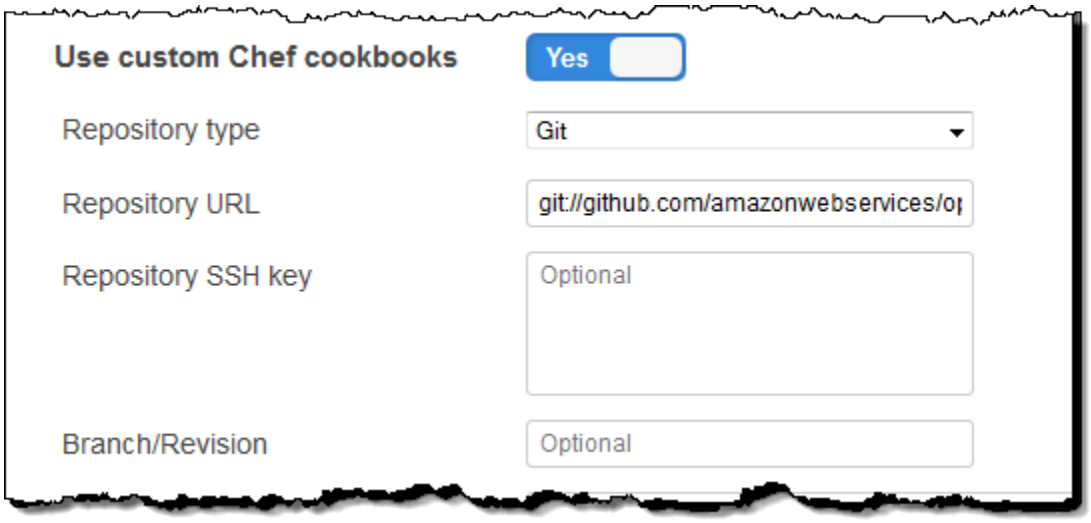
  [:redis] 속성의 값은 나중에 설명하듯이 사용자 지정 JSON을 사용하여 설정됩니다. 이 속성은 표준 앱 속성이 아닙니다.
- not_if 명령은 레시피가 이미 존재하는 구성 파일은 생성하지 않게 해줍니다.

쿡북을 작성한 후에는 각 인스턴스의 쿡북 캐시로 배포해야 합니다. 이 작업은 레시피는 실행하지 않고 새 쿡북을 스택의 인스턴스에 설치하기만 합니다. 일반적으로 레시피는 나중에 설명하듯이 계층의 수명 주기 이벤트에 할당하여 실행합니다.

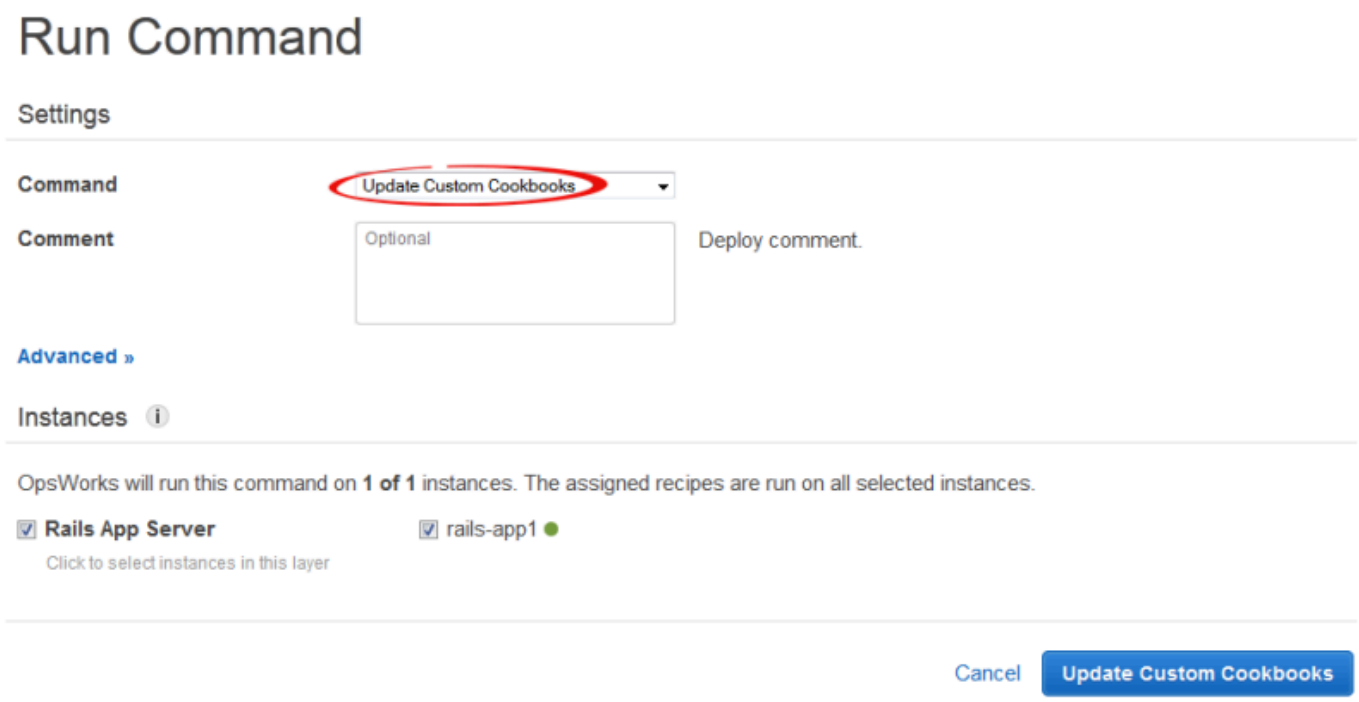
사용자 지정 쿡북을 배포하려면

1. AWS OpsWorks 스택 스택 페이지에서 스택 설정을 클릭한 다음 편집을 클릭합니다.

2. [구성 관리] 섹션에서 [사용자 지정 Chef 쿡북 사용]을 [예]로 설정하고, 쿡북 리포지토리 정보를 입력한 다음 [저장]을 클릭하여 스택 구성을 업데이트합니다.



3. [스택] 페이지에서 [명령 실행]을 클릭하고 [사용자 지정 쿡북 업데이트] 스택 명령을 선택한 다음 [사용자 지정 쿡북 업데이트]를 클릭하여 인스턴스의 쿡북 캐시에 새 쿡북을 설치합니다.



쿡북을 수정한 경우 [사용자 지정 쿡북 업데이트]를 다시 실행하면 업데이트된 버전이 설치됩니다. 이 절차에 대한 자세한 정보는 사용자 지정 쿡북 설치 단원을 참조하세요.

4단계: 수명 주기 이벤트에 레시피 할당

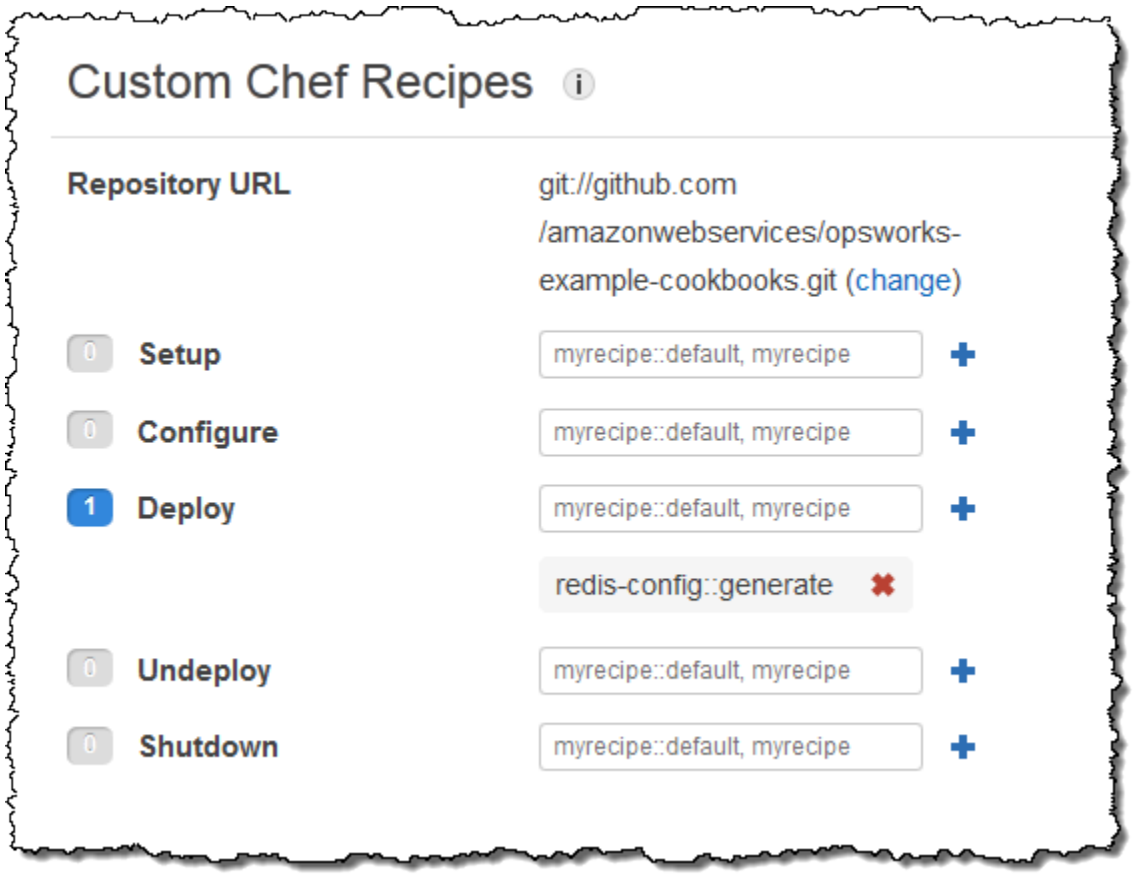
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자 지정 레시피를 수동으로 실행할 수 있지만 가장 좋은 방법은 일반적으로 AWS OpsWorks Stacks가 자동으로 실행하도록 하는 것입니다. 모든 계층에는 설정, 구성, 배포, 배포 취소, 종료의 5가지 수명 주기 이벤트 각각이 할당된 내장 레시피 세트가 있습니다. 인스턴스에서 이벤트가 발생할 때마다 AWS OpsWorks Stacks는 인스턴스의 각 계층마다 연결된 레시피를 실행하여 해당 작업을 처리합니다. 예를 들어 인스턴스 부팅이 완료되면 AWS OpsWorks Stacks는 설정 이벤트를 트리거합니다. 이 이벤트는 연결된 계층의 설정 레시피를 실행하는데, 이러한 레시피는 일반적으로 패키지 설치 및 구성과 같은 작업을 처리합니다.

적절한 수명 주기 이벤트에 레시피를 할당하여 AWS OpsWorks Stacks가 계층의 인스턴스에서 사용자 지정 레시피를 실행하도록 할 수 있습니다. 이 예제에서는 Rails 앱 서버 계층의 배포 이벤트에 `generate.rb` 레시피를 할당해야 합니다. 그러면 AWS OpsWorks 스택은 시작 중, 설정 레시피가 완료된 후, 앱을 배포할 때마다 계층의 인스턴스에서 레시피를 실행합니다. 자세한 내용은 자동으로 레시피 실행 단원을 참조하십시오.

Rails 앱 서버 layer's 계층의 Deploy 이벤트에 레시피를 할당하려면

1. AWS OpsWorks 스택 계층 페이지의 Rails 앱 서버에서 레시피를 클릭한 다음 편집을 클릭합니다.
2. [사용자 지정 Chef 레시피]에서 배포 이벤트에 정규화된 레시피 이름을 추가하고 [+]를 클릭합니다. 정규화된 레시피 이름은 *`cookbookname`*`::`*`recipename`* 형식을 사용합니다. 여기서 *`recipename`*에는 `.rb` 확장명이 포함되지 않습니다. 이 예에서 정규화된 이름은 `redis-config::generate`입니다. [저장]을 클릭하여 계층 구성을 업데이트합니다.



5단계: 스택 구성 JSON에 액세스 정보 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`generate.rb` 레시피는 Redis 서버의 호스트 이름 및 포트를 나타내는 한 쌍의 스택 구성 및 배포 JSON 속성을 사용합니다. 이러한 속성은 표준 `[:deploy]` 네임스페이스의 일부이지만 AWS OpsWorks Stacks에서 자동으로 정의하지는 않습니다. 사용자 지정 JSON 객체를 스택에 추가하여 속성 및 해당 값을 정의해 합니다. 다음 예는 이 예제의 사용자 지정 JSON을 보여줍니다.

스택 구성 및 배포 JSON에 액세스 정보를 추가하려면

1. AWS OpsWorks 스택 스택 페이지에서 스택 설정을 클릭한 다음 편집을 클릭합니다.
2. [구성 관리] 섹션에서 [사용자 지정 Chef JSON] 상자에 액세스 정보를 추가합니다. 액세스 정보는 다음과 유사해야 하며 아래와 같이 수정합니다.

   - `elasticache_redis_example`은 앱의 짧은 이름으로 대체합니다.
   - `host` 및 `port` 값은 1단계: ElastiCache Redis 클러스터 생성에서 생성한 ElastiCache Redis 서버 인스턴스의 값으로 대체합니다.

```
{
  "deploy": {
     "elasticache_redis_example": {
       "redis": {
         "host": "mycluster.XXXXXXXXX.amazonaws.com",
         "port": "6379"
       }
     }
  }
}
```

이 접근 방식의 장점은 사용자 지정 쿡북을 건드리지 않고 언제든지 포트 또는 호스트 값을 변경할 수 있다는 것입니다. AWS OpsWorks Stacks는 사용자 지정 JSON을 기본 제공 JSON으로 병합하고 이후의 모든 수명 주기 이벤트에 대해 스택의 인스턴스에 설치합니다. 그러면 3단계: 사용자 지정 쿡북 생

성 및 배포에 설명된 대로 앱이 Chef 노드 구문을 사용하여 속성 값에 액세스할 수 있습니다. 다음 번에 앱을 배포할 때 AWS OpsWorks Stacks가 새 정의를 포함하는 스택 구성 및 배포 JSON을 설치하고 `generate.rb`가 업데이트된 호스트 및 포트 값을 사용하여 구성 파일을 생성합니다.

> **Note**
>
> `[:deploy]`는 자동으로 모든 배포된 앱의 속성을 포함하며, 따라서 `[:deploy][elasticache_redis_example]`은 이미 스택 및 구성 JSON에 들어 있습니다. 그러나 `[:deploy][elasticache_redis_example]`에는 `[:redis]` 속성이 포함되지 않습니다. 사용자 지정 JSON으로 속성을 정의하면 AWS OpsWorks Stacks가 해당 속성을에 추가하도록 지시합니다`[:deploy][elasticache_redis_example]`. 사용자 지정 JSON을 사용하여 기존 속성을 재정의할 수도 있습니다. 자세한 내용은 속성 재정의 단원을 참조하십시오.

6단계: 앱 배포 및 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 예제에서는 이미 Redis를 사용하는 적절한 Ruby on Rails 애플리케이션이 있는 것으로 가정합니다. 구성 파일에 액세스하려면 다음과 같이 `redis` 젬을 Gemfile에 추가하고 `config/initializers/redis.rb`에서 Rails 이니셜라이저를 생성할 수 있습니다.

```
REDIS_CONFIG = YAML::load_file(Rails.root.join('config', 'redis.yml'))
$redis = Redis.new(:host => REDIS_CONFIG['host'], :port => REDIS_CONFIG['port'])
```

그런 다음 애플리케이션을 나타내는 앱을 만들고 이를 Rails 앱 서버 계층의 인스턴스에 배포합니다. 그러면 Rails 앱 서버 계층의 인스턴스가 애플리케이션 코드를 업데이트하고 `generate.rb`을 실행하여 구성 파일을 생성합니다. 사용자가 애플리케이션을 실행하면 애플리케이션이 ElastiCache Redis 인스턴스를 인 메모리 키-값 저장소로 사용합니다.

Amazon S3 버킷 사용하기

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

애플리케이션은 종종 Amazon Simple Storage Service(S3) 버킷을 사용하여 이미지 또는 기타 미디어 파일 같은 큰 항목을 저장합니다. AWS OpsWorks Stacks는 Amazon S3에 대한 통합 지원을 제공하지 않지만 애플리케이션이 Amazon S3 스토리지를 사용할 수 있도록 스택을 쉽게 사용자 지정할 수 있습니다. 이 주제에서는 PHP 애플리케이션 서버가 있는 Linux 스택을 예제로 사용하여 애플리케이션에 Amazon S3 액세스 권한을 제공하는 기본 프로세스를 단계별로 살펴봅니다. 이 기본 원칙은 Windows 스택에도 적용됩니다.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까?를 참조하세요.

주제

- 1단계: Amazon S3 버킷 생성
- 2단계: PHP 앱 서버 스택 생성
- 3단계: 사용자 지정 쿡북 생성 및 배포
- 4단계: 수명 주기 이벤트에 레시피 할당
- 5단계: 스택 구성 및 배포 속성에 액세스 정보 추가
- 6단계: PhotoApp 배포 및 실행

1단계: Amazon S3 버킷 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

먼저 Amazon S3 버킷을 만들어야 합니다. Amazon S3 콘솔, API 또는 CLI를 사용하여 직접 만들 수도 있지만 리소스를 생성하는 더 간단한 방법은 AWS CloudFormation 템플릿을 사용하는 것입니다. 다음 템플릿은 이 예제를 위한 Amazon S3 버킷을 생성하고, 버킷에 대한 무제한 액세스를 허용하는 IAM 역할을 사용하여 인스턴스 프로파일을 설정합니다. 그런 다음 계층 설정을 사용하여 스택의 애플리케이션 서버 인스턴스에 인스턴스 프로파일을 연결하면 뒤에 설명하는 것처럼 애플리케이션이 버킷에 액세스할 수 있습니다. 인스턴스 프로파일의 유용성은 Amazon S3로 국한되지 않습니다. 인스턴스 프로파일은 다양한 AWS 서비스를 통합하는 데에도 매우 중요합니다.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "AppServerRootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "AppServerRolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "AppServerS3Perms",
            "PolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "s3:*",
                  "Resource": { "Fn::Join" : ["", [ "arn:aws:s3:::", { "Ref" :
 "AppBucket" } , "/*" ]
                  ] }
```

```
               } ]
            },
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
      "AppServerInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
     "AppBucket" : {
      "Type" : "AWS::S3::Bucket"
      }
   },
   "Outputs" : {
       "BucketName" : {
           "Value" : { "Ref" : "AppBucket" }
       },
       "InstanceProfileName" : {
           "Value" : { "Ref" : "AppServerInstanceProfile" }
       }
   }
}
```

템플릿을 시작하면 다음과 같은 몇 가지 작업이 이루어집니다.

- `AWS::S3::Bucket` 리소스가 Amazon S3 버킷을 생성합니다.
- `AWS::IAM::InstanceProfile` 리소스가 애플리케이션 서버 인스턴스에 할당될 인스턴스 프로파일을 생성합니다.
- `AWS::IAM::Role` 리소스가 인스턴스 프로파일의 역할을 생성합니다.
- Amazon S3 버킷에 대한 무제한 액세스가 가능하도록 `AWS::IAM::Policy` 리소스가 역할의 권한을 설정합니다.
- 템플릿을 시작한 후 `Outputs` 섹션에는 AWS CloudFormation 콘솔에 버킷 및 인스턴스 프로파일 이름이 표시됩니다.

  스택과 앱을 설정하려면 이 값들이 필요합니다.

AWS CloudFormation 템플릿을 생성하는 방법에 대한 자세한 내용은 템플릿 기본 정보 학습을 참조하세요.

Amazon S3 버킷을 생성하려면

1. 예제 템플릿을 시스템의 텍스트 파일에 복사합니다.

   이 예제에서는 파일의 이름을 `appserver.template`(이)라고 가정합니다.
2. AWS CloudFormation 콘솔을 열고 스택 생성을 선택합니다.
3. [스택 이름] 상자에 스택 이름을 입력합니다.

   이 예제에서는 이름을 **`AppServer`**(이)라고 가정합니다.
4. 템플릿 파일 업로드, 찾아보기를 차례로 선택하고 1단계에서 만든 `appserver.template` 파일을 선택한 후 다음 단계를 선택합니다.
5. 파라미터 지정 페이지에서 이 템플릿이 IAM 리소스를 생성할 수 있음을 확인합니다.를 선택한 다음 마지막 페이지에 도달할 때까지 마법사의 각 페이지에서 다음 단계를 선택합니다. 생성(Create)을 선택합니다.
6. AppServer 스택이 CREATE_COMPLETE 상태에 도달하면 이 스택을 선택한 후 출력 탭을 선택합니다.

   상태를 업데이트하려면 여러 번 새로 고침해야 할 수 있습니다.
7. [출력] 탭에서 나중에 사용하기 위해 [BucketName] 및 [InstanceProfileName] 값을 기록합니다.

> **Note**
>
> AWS CloudFormation 는 스택이라는 용어를 사용하여 템플릿에서 생성된 리소스 모음을 참조합니다. 이는 AWS OpsWorks Stacks 스택과 동일하지 않습니다.

2단계: PHP 앱 서버 스택 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

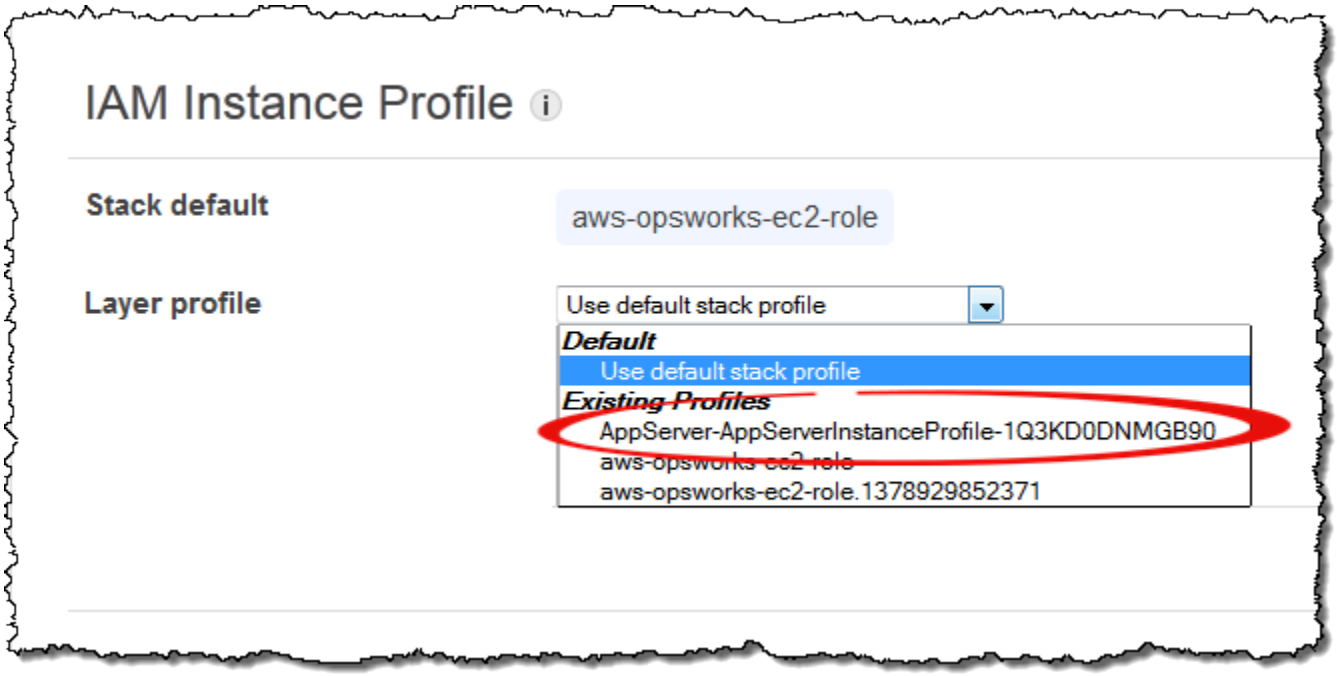
하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 스택은 각각 하나의 인스턴스를 포함하는 PHP 앱 서버 및 MySQL이라는 두 계층으로 구성됩니다. 애플리케이션은 Amazon S3 버킷에 사진을 저장하지만 MySQL 인스턴스를 백엔드 데이터 스토어로 사용하여 각 사진의 메타데이터를 보관합니다.

Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

스택을 생성하는 방법

1. 이 예제에서 명명된 새 **PhotoSite** 스택을 만들고 PHP 앱 서버 계층을 추가합니다. 새 스택과 계층 둘 다에 기본 설정을 사용할 수 있습니다. 자세한 정보는 새 스택 생성 및 OpsWorks 계층 생성 섹션을 참조하세요.
2. 계층 페이지에서 PHP 앱 서버에 대한 보안을 선택하고 편집을 선택합니다.
3. 계층 프로파일 섹션에서 AppServer AWS CloudFormation 스택을 시작한 후 이전에 기록한 인스턴스 프로파일 이름을 선택합니다. 이는와 같습니다`AppServer-AppServerInstanceProfile-1Q3KD0DNMGB90`. AWS OpsWorks Stacks는 계층의 모든 Amazon EC2 인스턴스에이 프로파일을 할당하여 계층의 인스턴스에서 실행되는 애플리케이션에 Amazon S3 버킷에 액세스할 수 있는 권한을 부여합니다.



4. 인스턴스를 PHP 앱 서버 계층에 추가하고 시작합니다. 인스턴스를 추가 및 시작하는 방법에 대한 자세한 정보는 계층에 인스턴스 추가 단원을 참조하세요.
5. 스택에 MySQL 계층을 추가하고 인스턴스를 추가한 다음 시작합니다. 계층과 인스턴스 둘 다에 기본 설정을 사용할 수 있습니다. 특히 MySQL 인스턴스는 Amazon S3 버킷에 액세스할 필요가 없으므로 기본적으로 선택되는 표준 AWS OpsWorks Stacks 인스턴스 프로파일을 사용할 수 있습니다.

3단계: 사용자 지정 쿡북 생성 및 배포

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택은 아직 준비되지 않았습니다.

- 애플리케이션이 MySQL 데이터베이스 서버와 Amazon S3 버킷에 액세스하려면 데이터베이스 호스트 이름과 Amazon S3 버킷 이름 같은 몇 가지 정보가 필요합니다.
- MySQL 데이터베이스 서버에서 데이터베이스를 설정하고 사진의 메타데이터를 보관할 테이블을 생성해야 합니다.

이러한 작업을 수동으로 처리할 수 있지만 더 좋은 방법은 Chef 레시피를 구현하고 AWS OpsWorks Stacks가 적절한 인스턴스에서 레시피를 자동으로 실행하도록 하는 것입니다. Chef 레시피는 AWS OpsWorks Stacks가 패키지 설치 또는 구성 파일 생성과 같은 인스턴스 작업을 수행하는 데 사용하는 특수 Ruby 애플리케이션입니다. 이 작업들은 여러 레시피와 구성 파일 템플릿 같은 관련 파일을 포함할 수 있는 쿡북에 패키징되어 있습니다. 쿡북은 GitHub 같은 레포지토리에 배치되며, 표준 디렉터리 구조를 가져야 합니다. 사용자 지정 쿡북 리포지토리가 없는 경우, 쿡북 리포지토리에서 설정 방법 단원을 참조하세요.

이 예제에서는 쿡북이 구현되어 퍼블릭 GitHub 리포지토리에 저장되어 있습니다. 이 쿡북에는 `appsetup.rb`와 `dbsetup.rb`라는 2개의 레시피와 `db-connect.php.erb`라는 1개의 템플릿 파일이 포함되어 있습니다.

appsetup.rb 레시피는 애플리케이션이 데이터베이스와 Amazon S3 버킷에 액세스하는 데 필요한 정보가 포함된 구성 파일을 생성합니다. 이것은 기본적으로 데이터베이스에 애플리케이션 연결에 설명된 appsetup.rb 레시피를 약간 수정한 버전입니다. 주된 차이점은 템플릿에 전달되는 변수이며, 액세스 정보를 나타냅니다.

처음 4개의 속성은 데이터베이스 연결 설정을 정의하고 MySQL 인스턴스를 생성할 때 AWS OpsWorks Stacks에서 자동으로 정의합니다.

이 변수들과 오리지널 레시피의 변수들 사이에는 다음 두 가지 차이점이 있습니다.

- table오리지널 레시피와 마찬가지로 변수는 dbsetup.rb에 의해 생성되는 데이터베이스 테이블의 이름을 나타내며, 쿡북의 속성 파일에서 정의되는 속성의 값으로 설정됩니다.

  다만 속성의 이름은 [:photoapp][:dbtable]로 다릅니다.
- s3bucket 변수는 이 예제에 고유한 것으로서 Amazon S3 버킷 이름을 나타내는 속성의 값인 [:photobucket](으)로 설정됩니다.

  [:photobucket]은(는) 뒤에 설명하는 것처럼 사용자 지정 JSON을 사용하여 정의됩니다. 속성에 대한 자세한 정보는 속성을(를) 참조하세요.

속성에 대한 자세한 정보는 속성을(를) 참조하세요.

dbsetup.rb 레시피는 각 사진의 메타데이터를 보관하는 데이터베이스 테이블을 설정합니다. 이것은 기본적으로 dbsetup.rb에 설명된 데이터베이스 설정 레시피를 약간 수정한 버전입니다. 자세한 설명은 해당 주제를 참조하세요.

이 예제와 오리지널 레시피의 유일한 차이점인 데이터베이스 스키마에는 Amazon S3 버킷에 저장된 각 사진의 ID, URL, 캡션이 포함된 3개의 열이 있습니다.

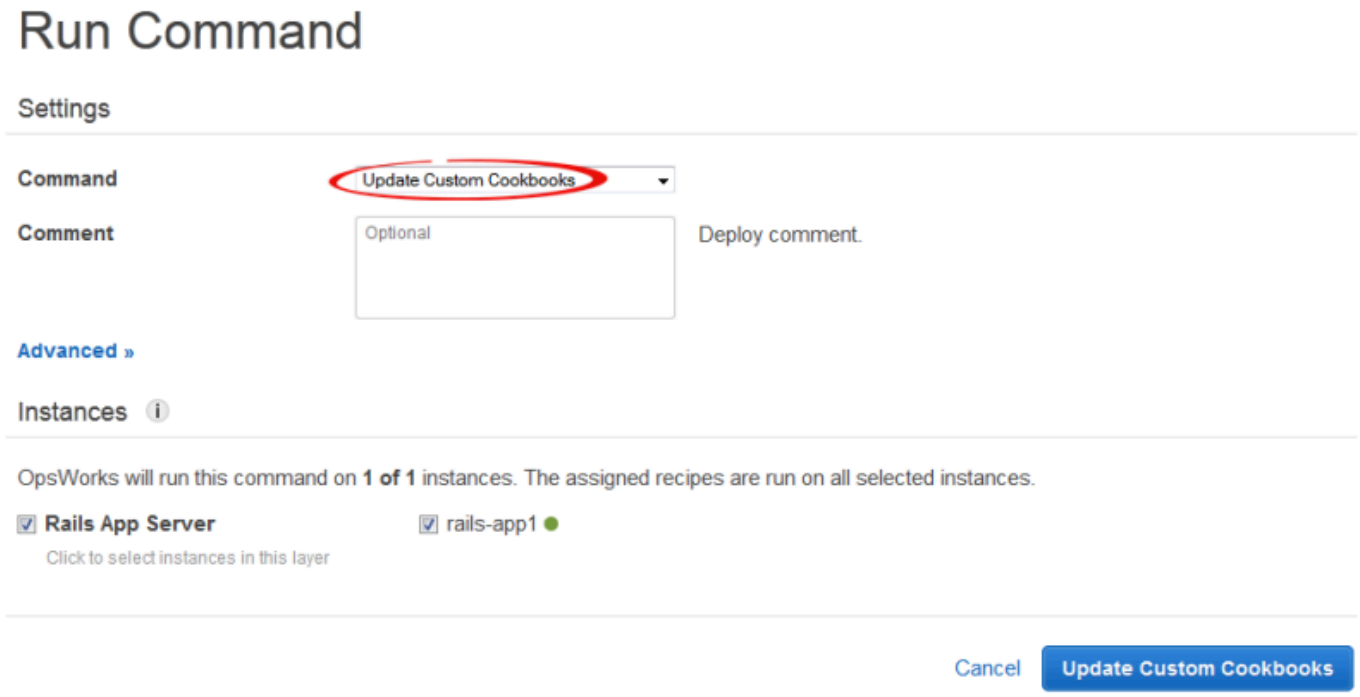
레시피가 이미 구현되어 있으므로 나중에 설명하듯이 각 인스턴스의 쿡북 캐시에 포토앱 쿡북을 배포하기만 하면 됩니다. 그러면 AWS OpsWorks Stacks는 적절한 수명 주기 이벤트가 발생할 때 캐시된 레시피를 실행합니다.

photoapp 쿡북을 배포하려면

1. AWS OpsWorks 스택 스택 페이지에서 스택 설정을 선택한 다음 편집을 선택합니다.
2. [구성 관리] 섹션에서:
   - [사용자 지정 Chef 쿡북 사용]을 [예]로 설정합니다.
   - [리포지토리 유형]을 Git로 설정합니다.

- 리포지토리 URL을 **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**로 설정합니다.

3. 스택 페이지에서 명령 실행을 선택하고 사용자 지정 쿡북 업데이트(사용자 지정 쿡북 업데이트) 스택 명령을 선택한 다음 사용자 지정 쿡북 업데이트(사용자 지정 쿡북 업데이트)를 선택하여 인스턴스 쿡북 캐시에 새 쿡북을 설치합니다.



4단계: 수명 주기 이벤트에 레시피 할당

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

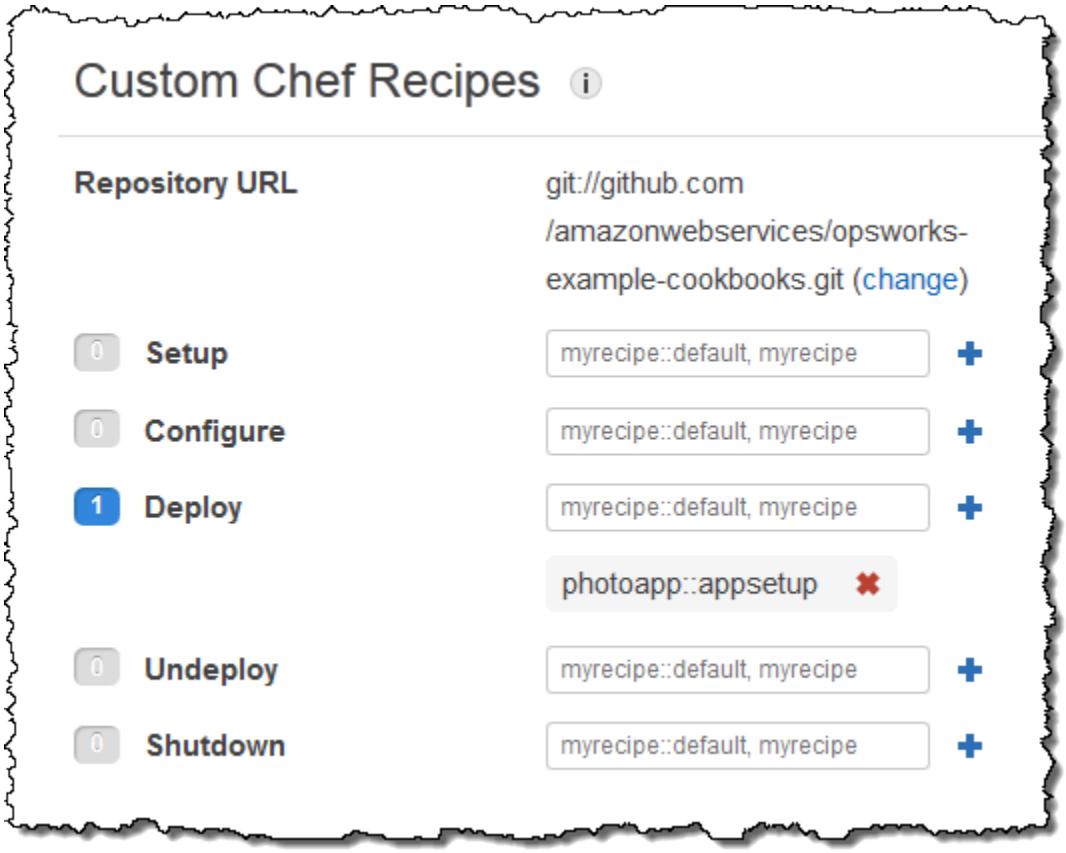
사용자 지정 레시피를 수동으로 실행할 수 있지만 가장 좋은 방법은 일반적으로 AWS OpsWorks Stacks가 자동으로 실행하도록 하는 것입니다. 모든 계층에는 5가지 라이프사이클 이벤트(설정, 구성, 배포, 배포 취소, 종료) 각각에 할당된 내장 레시피 세트가 있습니다. 인스턴스에서 이벤트가 발생할 때마다 AWS OpsWorks Stacks는 인스턴스의 각 계층에 연결된 레시피를 실행하여 필요한 작업을 처리합니다. 예를 들어 인스턴스 부팅이 완료되면 AWS OpsWorks Stacks는 Setup 이벤트를 트리거하여

Setup 레시피를 실행합니다. Setup 레시피는 일반적으로 패키지 설치 및 구성과 같은 작업을 처리합니다.

각 레시피를 적절한 수명 주기 이벤트에 할당하여 AWS OpsWorks Stacks가 계층의 인스턴스에서 사용자 지정 레시피를 실행하도록 할 수 있습니다. AWS OpsWorks Stacks는 계층의 내장 레시피가 완료된 후 모든 사용자 지정 레시피를 실행합니다. 이 예제에서는 PHP 앱 서버 계층의 배포 이벤트와 MySQL 계층의 배포 이벤트에 `dbsetup.rb`를 할당`appsetup.rb`합니다. 그러면 AWS OpsWorks 스택은 시작 중, 기본 제공 Setup 레시피가 완료된 후, 앱을 배포할 때마다, 빌드된 Deploy 레시피가 완료된 후 연결된 계층의 인스턴스에서 레시피를 실행합니다. 자세한 내용은 자동으로 레시피 실행 단원을 참조하십시오.

계층의 Deploy 이벤트에 사용자 지정 레시피를 할당하려면

1. AWS OpsWorks 스택 계층 페이지의 PHP 앱 서버에서 레시피를 선택한 다음 편집을 선택합니다.
2. 사용자 지정 Chef 레시피에서 Deploy 이벤트에 레시피 이름을 추가하고 +를 선택합니다. 이 레시피 이름은 Chef *`cookbookname`*`::`*`recipename`* 형식이어야 하며, 여기서 *`recipename`*에는 `.rb` 확장명이 포함되지 않습니다. 이 예제에서는 `photoapp::appsetup`을(를) 입력합니다. 저장을 선택하여 계층 구성을 업데이트합니다.



3. 계층 페이지에서, MySQL 계층의 작업 열의 편집을 선택합니다.

4. `photoapp::dbsetup`을(를) 계층의 Deploy 이벤트에 추가하고 새 구성을 저장합니다.

5단계: 스택 구성 및 배포 속성에 액세스 정보 추가

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

`appsetup.rb` 레시피는 각 인스턴스에 설치되고 스택 및 배포된 앱에 대한 자세한 정보를 포함하는 AWS OpsWorks Stacks 스택 구성 및 배포 속성의 데이터에 따라 달라집니다. 스택 구성 및 배포 속성 객체의 `deploy` 속성에는 편의상 JSON으로 표시되는 다음과 같은 구조가 있습니다.

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

배포 노드에는 앱의 짧은 이름으로 명명된, 배포된 각 앱의 속성이 포함됩니다. 각각의 앱 속성에는 문서 루트와 앱 유형 같은 앱의 구성을 정의하는 속성 세트가 포함됩니다. `deploy` 속성의 목록은 deploy 속성 단원을 참조하세요. Chef 속성 구문을 사용하여 레시피에서 스택 구성 및 배포 속성 값을 나타낼 수 있습니다. 예를 들어 `[:deploy][:app1][:application]`은(는) app1 앱의 짧은 이름을 나타냅니다.

사용자 지정 레시피는 데이터베이스 및 Amazon S3 액세스 정보를 나타내는 몇몇 스택 구성 및 배포 속성에 기반합니다.

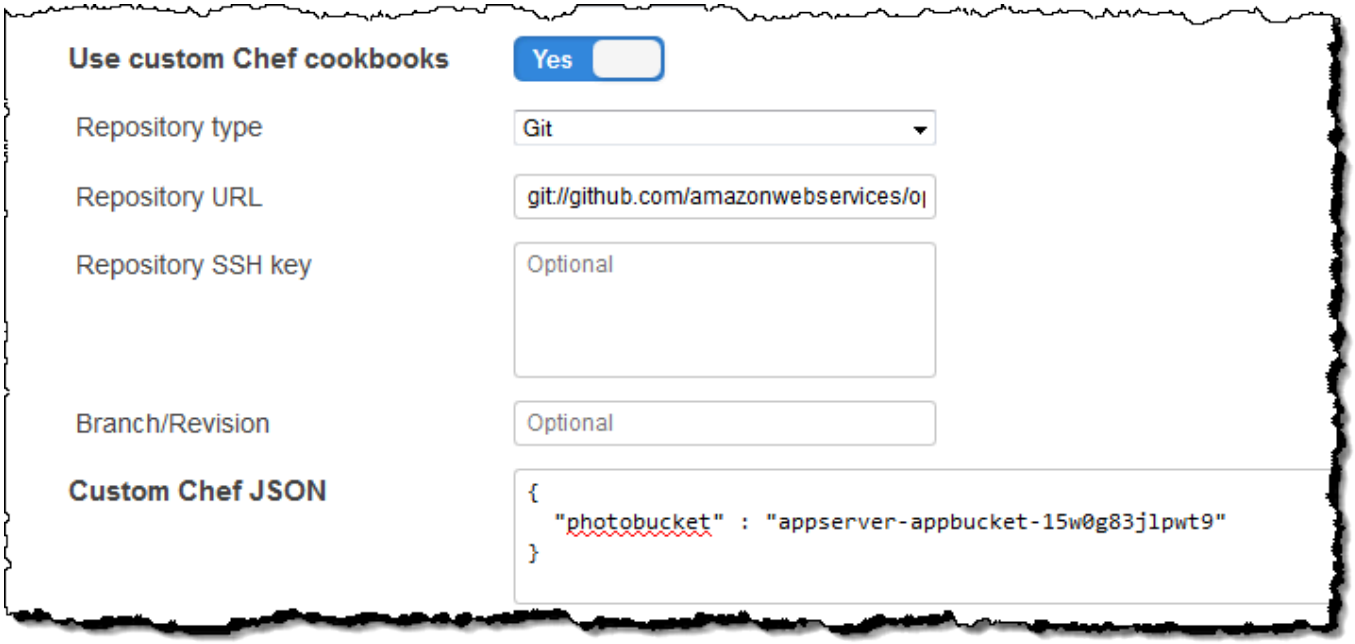
- 와 같은 데이터베이스 연결 속성[:deploy][:database][:host]은 MySQL 계층을 생성할 때 AWS OpsWorks Stacks에서 정의합니다.
- [:photoapp][:dbtable]와(과) 같은 테이블 이름 속성은 사용자 지정 쿡북의 속성 파일에서 정의되며, foto(으)로 설정됩니다.
- 사용자 지정 JSON을 사용하여 버킷 이름 속성 [:photobucket]을(를) 스택 구성 및 배포 속성에 추가하여 이 속성을 정의해야 합니다.

Amazon S3 버킷 이름 속성을 정의하려면

1. AWS OpsWorks 스택 스택 페이지에서 스택 설정을 선택한 다음 편집을 선택합니다.
2. [구성 관리] 섹션에서 [사용자 지정 Chef JSON] 상자에 액세스 정보를 추가합니다. 다음과 같이 보여야 합니다.

```
{
  "photobucket" : "yourbucketname"
}
```

*yourbucketname*을 1단계: Amazon S3 버킷 생성에서 기록해 둔 버킷 이름으로 대체합니다.



AWS OpsWorks Stacks는 사용자 지정 JSON을 스택의 인스턴스에 설치하기 전에 스택 구성 및 배포 속성에 병합합니다. 그러면가 [:photobucket] 속성에서 버킷 이름을 가져올 appsetup.rb 수 있

습니다. 레시피를 건드리지 않고 버킷을 변경할 수 있습니다. 속성을 재정의하여 새 버킷 이름을 제공하면 됩니다.

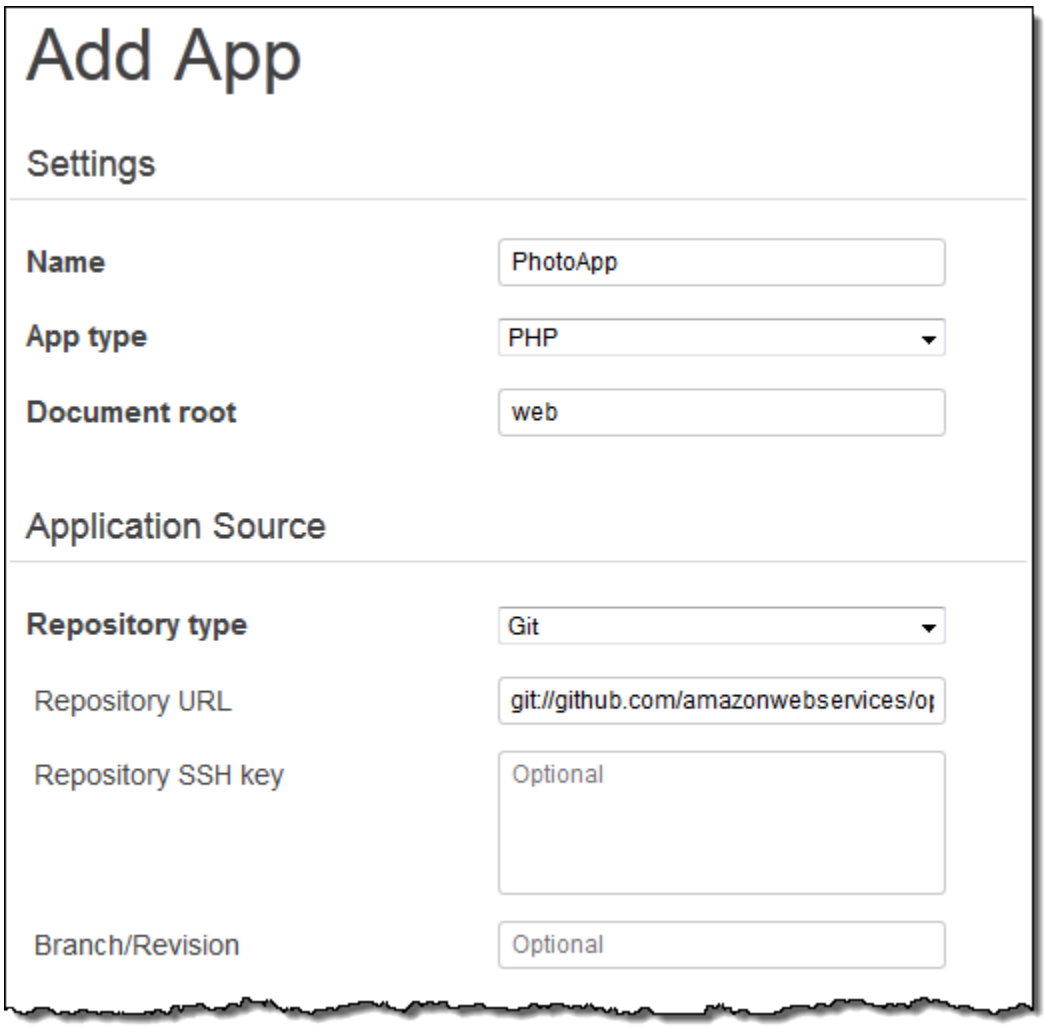
6단계: PhotoApp 배포 및 실행

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

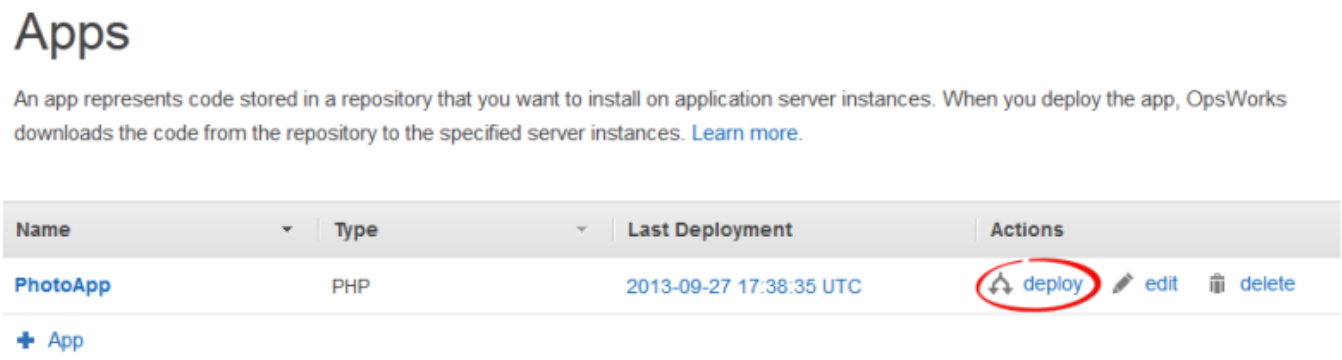
이 예제에서는 애플리케이션도 구현되어 퍼블릭 GitHub 리포지토리에 저장되어 있습니다. 앱을 스택에 추가하고 애플리케이션 서버에 배포한 다음 실행하기만 하면 됩니다.

앱을 스택에 추가하고 애플리케이션 서버에 배포하려면

1. 앱 페이지를 열고 앱 추가를 선택합니다.
2. [앱 추가] 페이지에서 다음을 수행합니다.
   - 이름을 **PhotoApp**으로 설정합니다.
   - [앱 유형]을 [PHP]로 설정합니다.
   - 문서 루트를 **web**으로 설정합니다.
   - [리포지토리 유형]을 [Git]로 설정합니다.
   - 리포지토리 URL을 **git://github.com/awslabs/opsworks-demo-php-photo-share-app.git**로 설정합니다.
   - 앱 추가를 선택하여 다른 설정에 대해 기본값을 수락합니다.

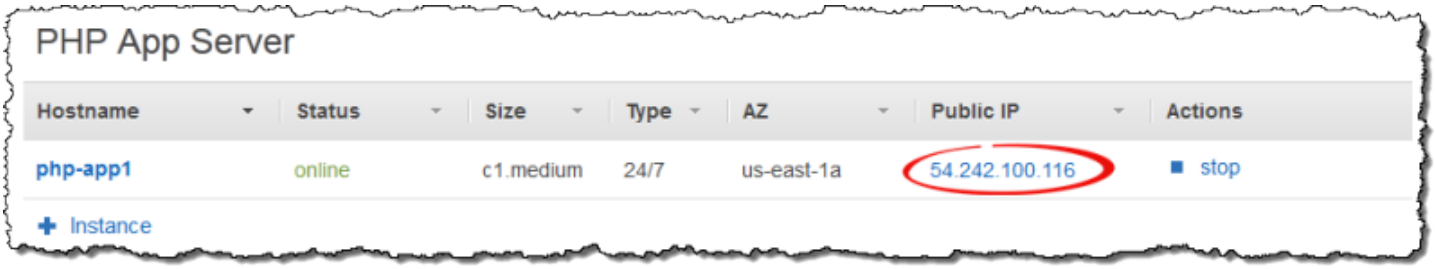


3. 앱 페이지에서 PhotoApp 앱의 작업 열에서 배포를 선택합니다.



4. 기본값을 수락하고 배포를 선택하여 서버에 앱을 배포합니다.

PhotoApp을 실행하려면 인스턴스 페이지로 이동해 PHP 앱 서버 인스턴스의 퍼블릭 IP 주소를 선택합니다.



다음과 같은 사용자 인터페이스가 표시됩니다. 사진을 Amazon S3 버킷에 저장하고 메타데이터를 백엔드 데이터 스토어에 저장하려면 사진 추가를 선택합니다.

AWS OpsWorks Stacks AWS CodePipeline 에서 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS CodePipeline을(를) 이용해 CodeCommit, Amazon Simple Storage Service(S3) 또는 GitHub와 같은 소스에서 코드 변경을 추적하는 지속적 전송 파이프라인을 생성할 수 있습니다. CodePipeline을 사용하여 Chef AWS OpsWorks 11.10, Chef 12 및 Chef 12.2 스택에서 Stacks에 대한 Chef 쿡북 및 애플리케이션 코드 릴리스를 자동화할 수 있습니다. 이 섹션의 예제에서는 CodePipeline에서 간단한 파이프라인을 생성하여 AWS OpsWorks Stacks 계층에서 실행하는 코드의 배포 도구로 사용하는 방법을 설명합니다.

> **Note**
>
> CodePipeline 및 AWS OpsWorks Stacks 통합은 Chef 11.4 이상 스택에 배포하는 데 지원되지 않습니다.

주제

- AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 12 스택
- AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 11 스택

AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 12 스택

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS CodePipeline을(를) 이용해 CodeCommit, Amazon Simple Storage Service(S3) 또는 GitHub와 같은 소스에서 코드 변경을 추적하는 지속적 전송 파이프라인을 생성할 수 있습니다. 이 항목의 예제는 CodePipeline에서 간단한 파이프라인을 생성하여 AWS OpsWorks Stacks 계층에서 실행할 코드의 배포 도구로 사용하는 방법을 설명합니다. 이 예제에서는 간단한 Node.js 앱에 대한 파이프라인을 생성한 다음 Chef AWS OpsWorks 12 스택의 계층에 있는 모든 인스턴스(이 경우 단일 인스턴스)에서 앱을 실행하도록 Stacks에 지시합니다.

> **Note**
>
> 이 항목은 파이프라인을 사용하여 Chef 12 스택에서 앱을 실행하고 업데이트하는 방법을 설명합니다. 파이프라인을 사용하여 Chef 11.10 스택에서 앱을 실행하고 업데이트하는 방법에 대한 자세한 내용은 AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 11 스택 섹션을 참조하세요. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

주제

- 사전 조건
- 지원되는 다른 시나리오
- 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성
- 2단계: 사용자 지정 쿡북을 사용하도록 스택 및 계층을 구성
- 3단계: Amazon S3 버킷에 앱 코드 업로드
- 4단계: AWS OpsWorks Stacks에 앱 추가
- 5단계: CodePipeline에서 파이프라인 생성
- 6단계: AWS OpsWorks Stacks에서 앱 배포 확인
- 7단계(선택 사항): 앱 코드를 업데이트하여 CodePipeline이 자동으로 앱을 재배포하는지 확인
- 8단계(선택 사항): 리소스 정리

사전 조건

이 연습을 시작하기 전에 다음 작업을 모두 수행할 수 있는 관리자 권한이 있는지 확인하세요. AdministratorAccess 정책이 적용되는 그룹의 멤버일 수도 있고, 다음 표의 권한과 정책이 있는 그룹의 멤버일 수도 있습니다. 보안 모범 사례로서, 개별 사용자에 필요한 권한을 할당하는 대신 사용자가 다음 작업을 수행할 수 있는 권한을 보유한 그룹에 속해야 합니다.

IAM에서 보안 그룹을 생성하고 해당 그룹에 권한을 할당하는 자세한 정보는 IAM 사용자 그룹 만들기를 참조하세요. AWS OpsWorks Stacks 권한 관리에 대한 자세한 내용은 모범 사례: 권한 관리를 참조하세요.

권한	그룹에 연결할 권장 정책
AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성하고 편집합니다.	AWSOpsWorks_FullAccess
AWS CloudFormation에서 템플릿을 생성하고, 편집하고, 실행합니다.	AmazonCloudFormationFullAccess
Amazon S3 버킷을 생성하고 편집하고 액세스합니다.	AmazonS3FullAccess

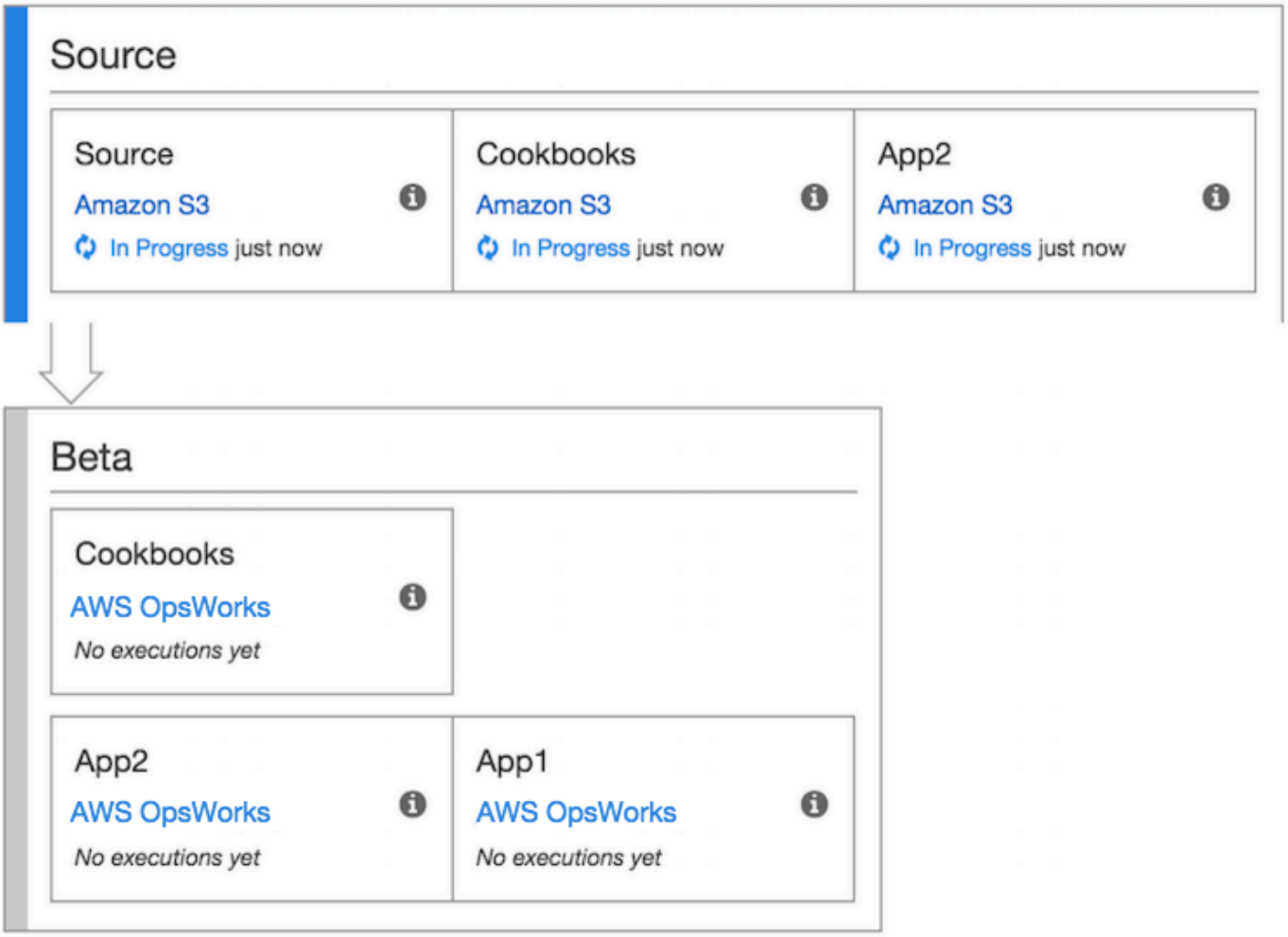
권한	그룹에 연결할 권장 정책
CodePipeline에서 파이프라인, 특히 AWS OpsWorks Stacks를 공급자로 사용하는 파이프라인을 생성, 편집 및 실행합니다.	AWSCodePipeline_FullAccess

Amazon EC2 키 페어도 있어야 합니다. 이 연습에서 샘플 스택, 계층 및 인스턴스를 생성하는 AWS CloudFormation 템플릿을 실행할 때이 키 페어의 이름을 입력하라는 메시지가 표시됩니다. Amazon EC2 콘솔에서 키 페어를 얻는 방법에 대한 자세한 내용은 Amazon EC2 문서의 키 페어 생성을 참조하세요. 키 페어는 미국 동부(버지니아 북부) 리전에 있어야 합니다. 해당 리전에 이미 키 페어가 있는 경우 기존 키 페어를 사용할 수 있습니다.

지원되는 다른 시나리오

이 연습에서는 Source 단계와 Deploy 단계를 하나씩 포함하는 간단한 파이프라인을 생성합니다. 그러나 AWS OpsWorks Stacks를 공급자로 사용하는 더 복잡한 파이프라인을 생성할 수 있습니다. 다음은 지원되는 파이프라인 및 시나리오의 예입니다.

- 파이프라인을 편집하여 Chef 쿡북을 Source 단계에서 추가하고 업데이트된 쿡북의 연결된 대상을 Deploy 단계에 추가할 수 있습니다. 이 경우, 사용자가 소스를 변경할 때 쿡북 업데이트를 트리거하는 Deploy 작업을 추가합니다. 업데이트된 쿡북은 앱보다 먼저 배포됩니다.
- 사용자 지정 쿡북과 여러 앱을 사용하여 복잡한 파이프라인을 생성하고 AWS OpsWorks Stacks 스택에 배포할 수 있습니다. 파이프라인은 애플리케이션 및 쿡북 소스에 대한 변경 사항을 추적하고 사용자가 변경할 경우 재배포합니다. 다음은 복잡한 파이프라인의 예입니다.



CodePipeline 작업에 대한 자세한 내용은 CodePipeline 사용 설명서를 참조하세요.

1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks를 파이프라인의 배포 공급자로 사용하려면 먼저 스택, 계층 및 계층에 하나 이상의 인스턴스가 있어야 합니다. Linux 스택 시작하기 또는 Windows 스택 시작하기의 지침에

AWS OpsWorks 따라 스택에서 스택을 생성할 수 있지만, 시간을 절약하기 위해이 예제에서는 AWS CloudFormation 템플릿을 사용하여 Linux 기반 Chef 12 스택, 계층 및 인스턴스를 생성합니다. 이 템플릿에 의해 생성된 인스턴스는 Amazon Linux 2016.03을 실행하며 인스턴스 유형은 `c3.large`입니다. 템플릿에서는 사용자 지정 쿡북을 사용하도록 스택을 구성하지 않지만, 이 연습에서 나중에 이렇게 구성합니다.

> **Important**
>
> AWS CloudFormation 템플릿은 나중에 앱을 업로드하는 Amazon S3 버킷과 동일한 리전과 나중에 CodePipeline에서 파이프라인을 생성하는 리전에 저장하고 실행해야 합니다. 현재 CodePipeline은 미국 동부(버지니아 북부) 리전(us-east-1)에서만 AWS OpsWorks Stacks 공급자를 지원합니다. 이 연습에서는 모든 리소스를 미국 동부(버지니아 북부) 리전에서 생성해야 합니다.
>
> 스택 생성이 실패할 경우 계정에 허용되는 최대 수의 IAM 역할에 접근한 것일 수 있습니다. 계정에서 `c3.large` 인스턴스 유형의 인스턴스를 시작할 수 없는 경우에도 스택 생성이 실패할 수 있습니다. 예를 들어 AWS 프리 티어를 사용 중이라면 `Root device type: must be included in EBS` 같은 오류가 반환될 수 있습니다. 계정에 AWS 프리 티어에서 부과하는 제한과 같이 생성할 수 있는 인스턴스 유형에 대한 제한이 있는 경우 템플릿의 인스턴스 블록에 있는 `InstanceType` 파라미터의 값을 계정에서 사용할 수 있는 인스턴스 유형으로 변경해 보세요.

를 사용하여 스택, 계층 및 인스턴스를 생성하려면 AWS CloudFormation

1. 다음 AWS CloudFormation 템플릿을 새 일반 텍스트 문서에 복사합니다. 로컬 컴퓨터의 편리한 위치에 파일을 저장하고 NewOpsWorksStack.template 또는 원하는 다른 이름을 지정합니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
```

```
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "eu-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
  },
  "Parameters": {
    "EC2KeyPairName": {
   "Type": "String",
   "Description": "The name of an existing EC2 key pair that lets you use SSH to
connect to the OpsWorks instance."
  }
  },
```

```
 "Resources": {
"CPOpsDeploySecGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription" : "Lets you manage OpsWorks instances to which you deploy
apps with CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
     }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
  }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
```

```
            "OpsWorksServiceRole",
            "Arn"
          ]
        },
"ConfigurationManager" : { "Name": "Chef","Version": "12" },
"DefaultOs": "Amazon Linux 2016.03",
        "DefaultInstanceProfileArn": {
          "Fn::GetAtt": [
            "OpsWorksInstanceProfile",
            "Arn"
          ]
        },
"UseCustomCookbooks": "false"
    }
  },
  "MyLayer": {
    "Type": "AWS::OpsWorks::Layer",
    "Properties": {
      "StackId": {
        "Ref": "MyStack"
      },
      "Name": "Node.js App Server",
"Type": "custom",
      "Shortname": "app1",
"EnableAutoHealing": "true",
      "AutoAssignElasticIps": "false",
      "AutoAssignPublicIps": "true",
"CustomSecurityGroupIds": [
  {
    "Fn::GetAtt": [
            "CPOpsDeploySecGroup", "GroupId"
    ]
  }
 ]
    },
    "DependsOn": [
      "MyStack",
      "CPOpsDeploySecGroup"
    ]
  },
  "OpsWorksServiceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
```

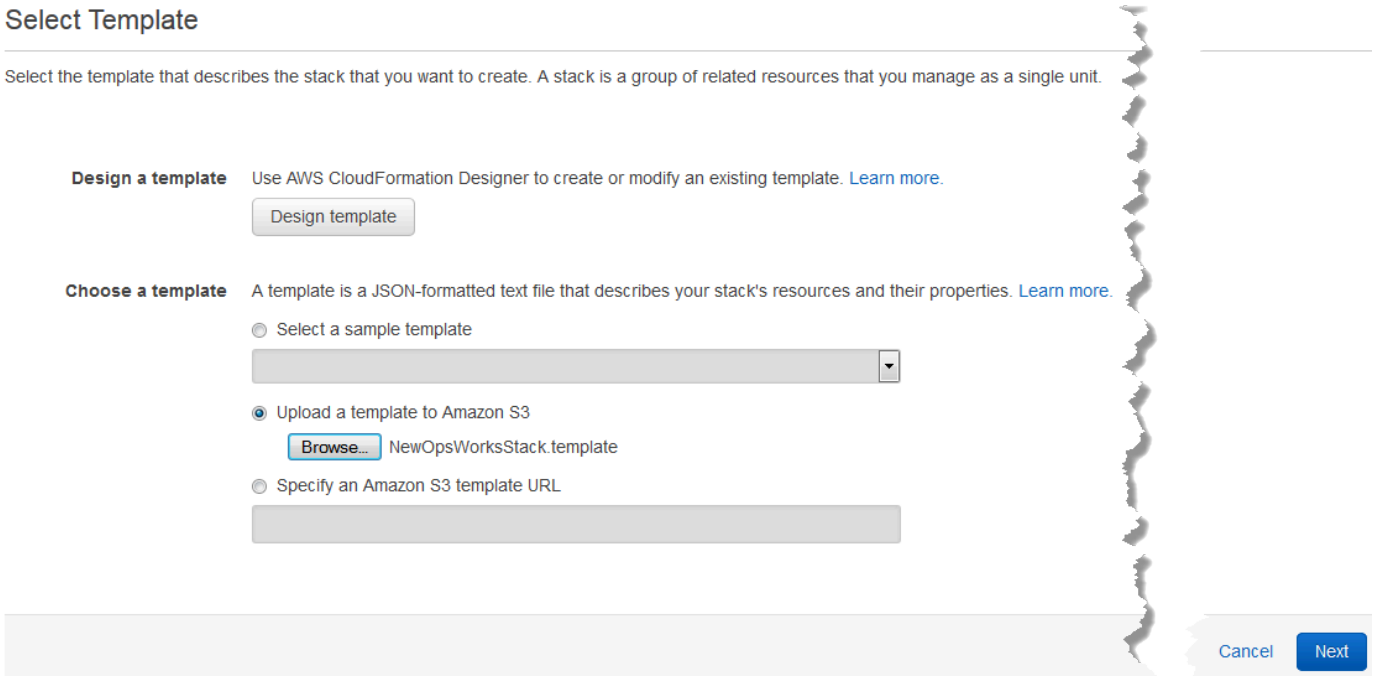
```
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "OpsWorksPrincipal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "opsworks-service",
            "PolicyDocument": {
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": [
                    "ec2:*",
                    "iam:PassRole",
                    "cloudwatch:GetMetricStatistics",
                    "elasticloadbalancing:*"
                  ],
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
```

```
    "OpsWorksInstanceProfile": {
      "Type": "AWS::IAM::InstanceProfile",
      "Properties": {
        "Path": "/",
        "Roles": [
          {
            "Ref": "OpsWorksInstanceRole"
          }
        ]
      }
    },
    "OpsWorksInstanceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "EC2Principal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
  "Policies": [
          {
            "PolicyName": "s3-get",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
```

```
                {
                  "Effect": "Allow",
                  "Action": [
                    "s3:GetObject"
                  ],
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "myinstance": {
      "Type": "AWS::OpsWorks::Instance",
      "Properties": {
        "LayerIds": [
          {
            "Ref": "MyLayer"
          }
        ],
        "StackId": {
          "Ref": "MyStack"
        },
        "InstanceType": "c3.large",
        "SshKeyName": {
    "Ref": "EC2KeyPairName"
  }
      }
    }
  },
  "Outputs": {
    "StackId": {
      "Description": "Stack ID for the newly created AWS OpsWorks stack",
      "Value": {
        "Ref": "MyStack"
      }
    }
  }
}
```

2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation AWS CloudFormation 콘솔을 엽니다.

3. AWS CloudFormation 홈 페이지에서 스택 생성을 선택합니다.
4. [템플릿 선택] 페이지의 [템플릿 선택] 영역에서 [Amazon S3에 템플릿 업로드]를 선택한 다음 [찾아보기]를 선택합니다.
5. 1단계에서 저장한 AWS CloudFormation 템플릿을 찾아 열기를 선택합니다. 템플릿 선택 페이지에서 다음을 선택합니다.

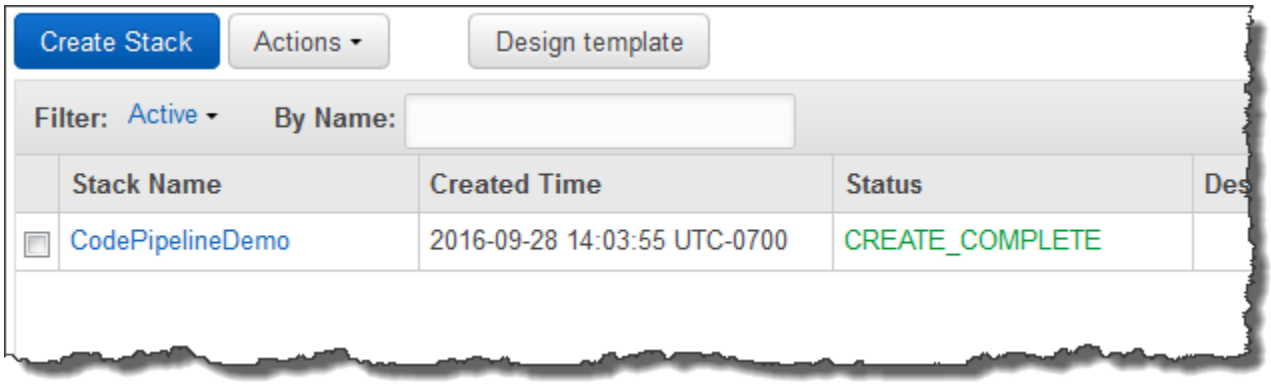


6. [세부 정보 지정] 페이지에서 스택의 이름을 [CodePipelineDemo] 또는 계정에 고유한 다른 스택 이름으로 지정합니다. 스택에 다른 이름을 선택하면 이 연습 전체에서 해당 스택 이름을 변경해야 합니다.
7. 파라미터 영역에서 AWS OpsWorks Stacks 인스턴스가 생성된 후 인스턴스에 액세스하는 데 사용할 EC2 키 페어의 이름을 입력합니다. 다음을 선택합니다.
8. 옵션 페이지에서 다음을 선택합니다. 이 페이지의 설정은 이번 연습에는 필요하지 않습니다.
9. 이 연습에서 사용하는 AWS CloudFormation 템플릿은 IAM 역할, 인스턴스 프로파일 및 인스턴스를 생성합니다.

> **Important**
>
> 생성을 선택하기 전에 비용을 선택하여이 템플릿으로 리소스를 생성하기 AWS 위해 발생할 수 있는 요금을 추정합니다.

IAM 리소스 생성이 허용되는 경우 이 템플릿으로 인해 AWS CloudFormation에서 IAM 리소스를 생성할 수 있음을 승인합니다 확인란을 선택한 다음 생성을 선택합니다. IAM 리소스를 생성할 수 없는 경우 이 절차를 계속 진행할 수 없습니다.

10. AWS CloudFormation 대시보드에서 스택 생성 진행 상황을 볼 수 있습니다. 다음 단계를 계속하기 전에 상태 열에 CREATE_COMPLETE가 표시될 때까지 기다리세요.



AWS OpsWorks Stacks에서 스택 생성을 확인하려면

1. https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks Stacks 대시보드에서 생성한 스택을 확인합니다.

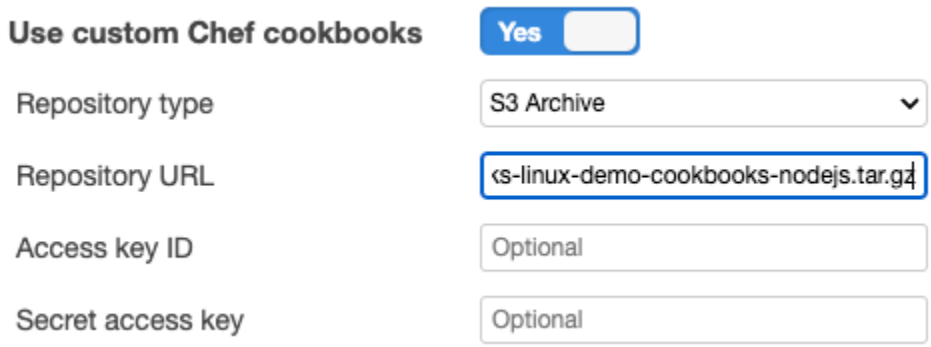
3. 스택을 열고 계층 및 인스턴스를 확인합니다. AWS CloudFormation 템플릿에 제공된 이름 및 기타 메타데이터를 사용하여 계층과 인스턴스가 생성되었는지 확인합니다. 사용자 지정 Chef 쿡북 및 레시피를 사용하도록 스택 및 계층을 구성할 준비가 되었습니다.

2단계: 사용자 지정 쿡북을 사용하도록 스택 및 계층을 구성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Stacks의 Chef 12 스택은 사용자 지정 애플리케이션 계층을 빌드하기 위해 자체 또는 커뮤니티 생성 AWS OpsWorks 쿡북이 필요합니다. 이 연습을 위해 여러 개의 Chef 쿡북 및 Chef 레시피가 들어 있는 리포지토리를 가리킬 수 있습니다. 이들 레시피는 사용자의 인스턴스에 Node.js 패키지와 해당 종속성을 설치합니다. 4단계: AWS OpsWorks Stacks에 앱 추가에서 준비할 Node.js 앱을 배포하기 위해서는 다른 Chef 레시피를 사용합니다. CodePipeline이 애플리케이션의 새 버전을 배포할 때마다 이 단계에서 지정한 Chef 레시피가 실행됩니다.

1. Stacks 콘솔에서 AWS OpsWorks 에서 생성한 스택을 엽니다1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성. [스택 설정]을 선택한 다음 [편집]을 선택합니다.
2. [사용자 지정 Chef 쿡북 사용]을 [예]로 설정합니다. 이렇게 하면 관련된 사용자 지정 쿡북 설정이 표시됩니다.
3. [리포지토리 유형] 드롭다운 목록에서 [S3 아카이브]를 선택합니다. CodePipeline과를 모두 사용하려면 AWS OpsWorks쿡북 소스가 S3여야 합니다.
4. [리포지토리 URL]에서 **`https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`**을 지정합니다. 설정은 다음과 유사합니다.



5. 저장을 선택합니다.
6. 탐색 창에서 계층을 선택합니다.
7. 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성에서 만든 계층의 설정을 선택합니다.
8. [일반 설정] 탭에서 계층 이름이 Node.js 앱 서버이고, 계층의 짧은 이름이 app1인지 확인합니다. [레시피]를 선택합니다.
9. 레시피 탭에서 **`nodejs_demo`**을(를) 배포 수명 주기 이벤트 중에 실행할 레시피로 지정합니다. 저장을 선택합니다.
10. 보안 탭의 보안 그룹 드롭다운 목록에서 AWS-OpsWorks-Webapp 보안 그룹을 선택합니다.
11. 저장을 선택합니다.

3단계: Amazon S3 버킷에 앱 코드 업로드

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

파이프라인 설정의 일환으로 코드 리포지토리에 대한 링크를 제공해야 하므로 파이프라인을 설정하기 전에 코드 리포지토리를 준비하세요. 이 연습에서는 Node.js 앱을 Amazon S3 버킷에 업로드합니다.

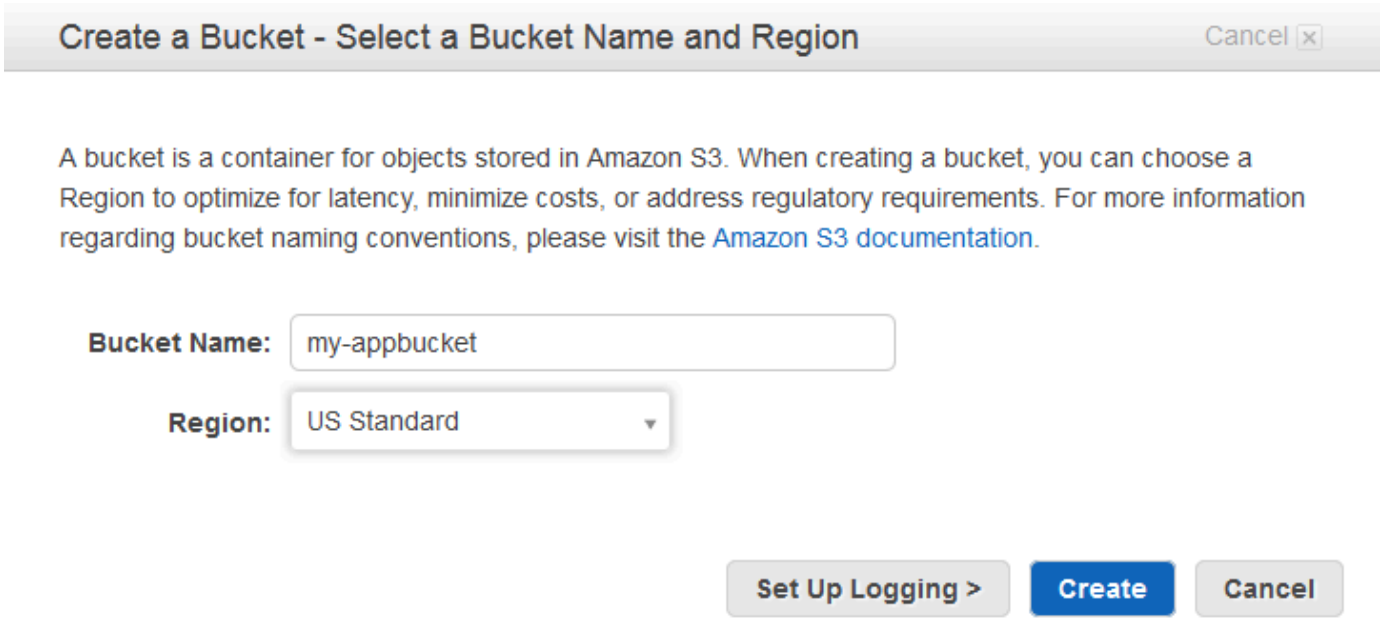
CodePipelin은 소스로 GitHub 또는 CodeCommit로부터 바로 코드를 사용할 수 있지만, 이 연습에서는 Amazon S3 버킷을 사용하는 방법을 예시합니다. 이 연습에서는 샘플 Node.js 앱을 자체 Amazon S3 버킷에 업로드하여 앱을 변경할 수 있습니다. 이 단계에서 생성한 Amazon S3 버킷을 통해 CodePipeline가 앱 코드 변경을 감지하고 변경된 앱을 자동으로 배포할 수 있습니다. 원한다면 기존 버킷을 사용해도 됩니다. 버킷이 CodePipeline 설명서의 간단한 파이프라인 연습(Amazon S3 버킷)에 설명된 기준을 충족하는지 확인하세요.

> **Important**
>
> Amazon S3 버킷은 나중에 파이프라인을 생성할 리전과 동일한 리전에 위치해야 합니다. 현재 CodePipeline은 미국 동부(버지니아 북부) 리전(us-east-1)에서만 AWS OpsWorks Stacks 공급자를 지원합니다. 이 연습에서는 모든 리소스를 미국 동부(버지니아 북부) 리전에서 생성해야 합니다. 또한 CodePipeline은 버전 관리된 소스를 요구하므로 버킷의 버전을 관리해야 합니다. 자세한 내용은 버전 관리 사용을 참조하세요.

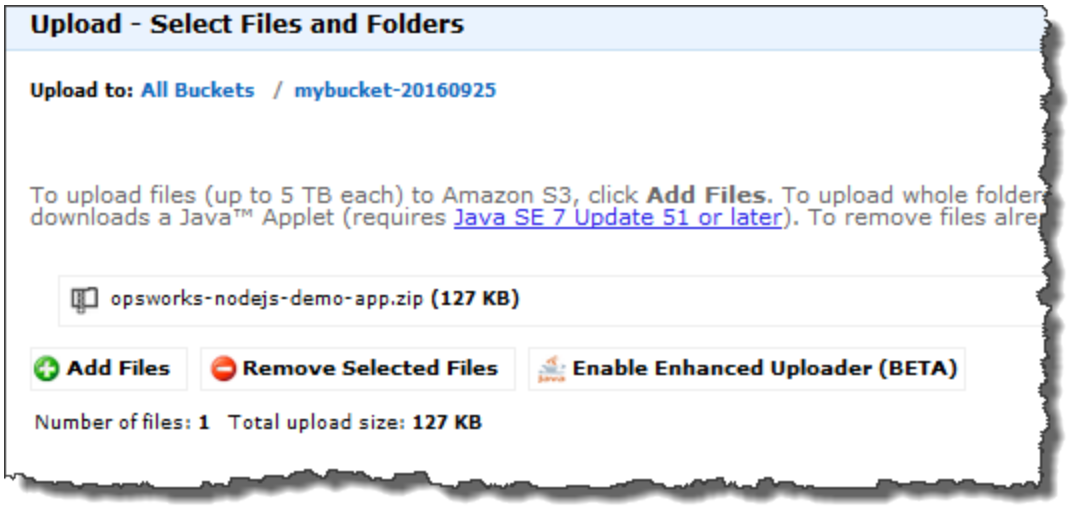
Amazon S3 버킷에 앱을 업로드하려면

1. AWS OpsWorks Stacks 샘플의 ZIP 파일인 Node.js 앱을 다운로드하여 로컬 컴퓨터의 편리한 위치에 저장합니다.
2. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
3. 버킷 생성을 선택합니다.
4. [버킷 생성 - 버킷 이름 및 리전 선택] 페이지에서 [버킷 이름]으로 버킷의 고유한 이름을 입력합니다. 버킷 이름은 자신의 AWS 계정뿐만 아니라 모든 계정에서 고유해야 합니다. 이 연습에서는

my-appbucket이라는 이름을 사용하지만 `my-appbucket-`*`yearmonthday`*를 사용하여 버킷 이름을 고유하게 만들 수 있습니다. [리전] 드롭다운 목록에서 [미국 표준]을 선택한 다음 [만들기]를 선택합니다. [미국 표준]는 [us-east-1]에 해당합니다.



5. 모든 버킷 목록에서 생성한 버킷을 선택합니다.
6. 버킷 페이지에서 업로드를 선택합니다.
7. [업로드 - 파일 및 폴더 선택] 페이지에서 [파일 추가]를 선택합니다. 1단계에서 저장한 ZIP 파일을 찾아 열기를 선택하고 업로드 시작을 선택합니다.



8. 업로드가 완료되면 버킷의 파일 목록에서 ZIP 파일을 선택한 다음 [속성]를 선택합니다.
9. [속성] 창에서 ZIP 파일에 대한 링크를 복사해 적어 둡니다. 파이프라인을 생성하려면 이 링크의 버킷 이름 및 ZIP 파일 이름 부분이 필요합니다.

4단계: AWS OpsWorks Stacks에 앱 추가

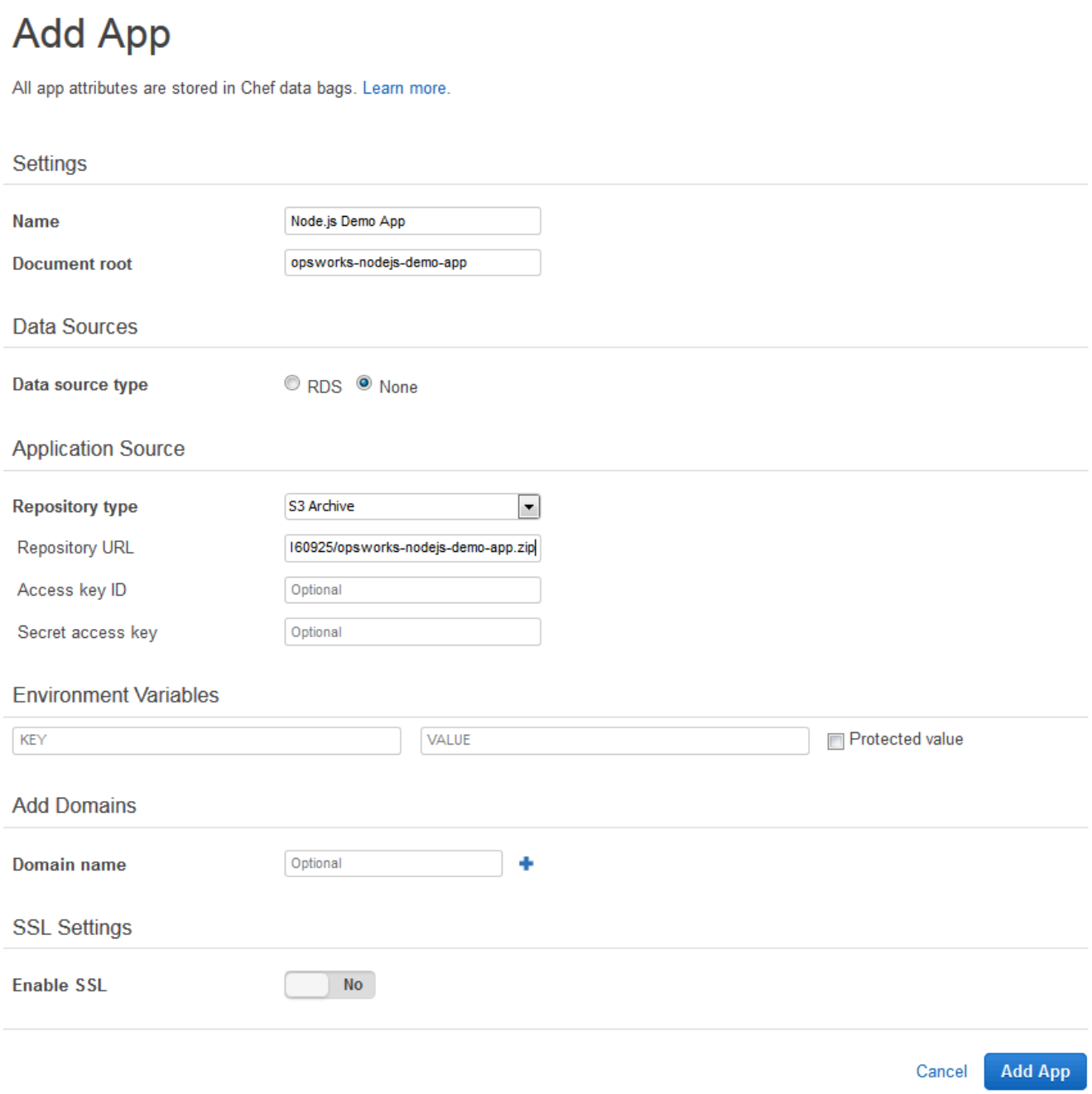
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

CodePipeline에서 파이프라인을 생성하기 전에 Node.js 테스트 앱을 AWS OpsWorks Stacks에 추가합니다. 파이프라인을 생성할 때 AWS OpsWorks Stacks에 추가한 앱을 선택해야 합니다.

이전 절차의 9단계에서 생성한 Amazon S3 버킷 링크를 준비합니다. 이 절차를 완료하려면 테스트 앱을 저장한 버킷에 대한 링크가 필요합니다.

AWS OpsWorks Stacks에 앱을 추가하려면

1. AWS OpsWorks Stacks 콘솔에서 CodePipelineDemo를 열고 탐색 창에서 앱을 선택합니다.
2. 앱 추가를 선택합니다.
3. 앱 추가 페이지에서 다음 정보를 입력합니다.
   a. 앱의 이름을 지정합니다. 이 연습에서는 `Node.js Demo App`이라는 이름을 사용합니다.
   b. 데이터 원본 유형에 대해 없음을 선택합니다. 이 앱에는 외부 데이터베이스 또는 데이터 원본이 필요하지 않습니다.
   c. [리포지토리 유형] 드롭다운 목록에서 [S3 아카이브]를 선택합니다.
   d. 리포지토리 URL 문자열 상자에 3단계: Amazon S3 버킷에 앱 코드 업로드의 9단계에서 복사한 URL을 붙여 넣습니다. 형식이 다음과 유사해야 합니다.



4. 이 형식의 다른 설정은 변경할 필요가 없습니다. [앱 추가]를 선택합니다.
5. Node.js 데모 앱 앱이 앱 페이지의 목록에 나타나면 다음 절차인 5단계: CodePipeline에서 파이프라인 생성을(를) 계속 진행하세요.

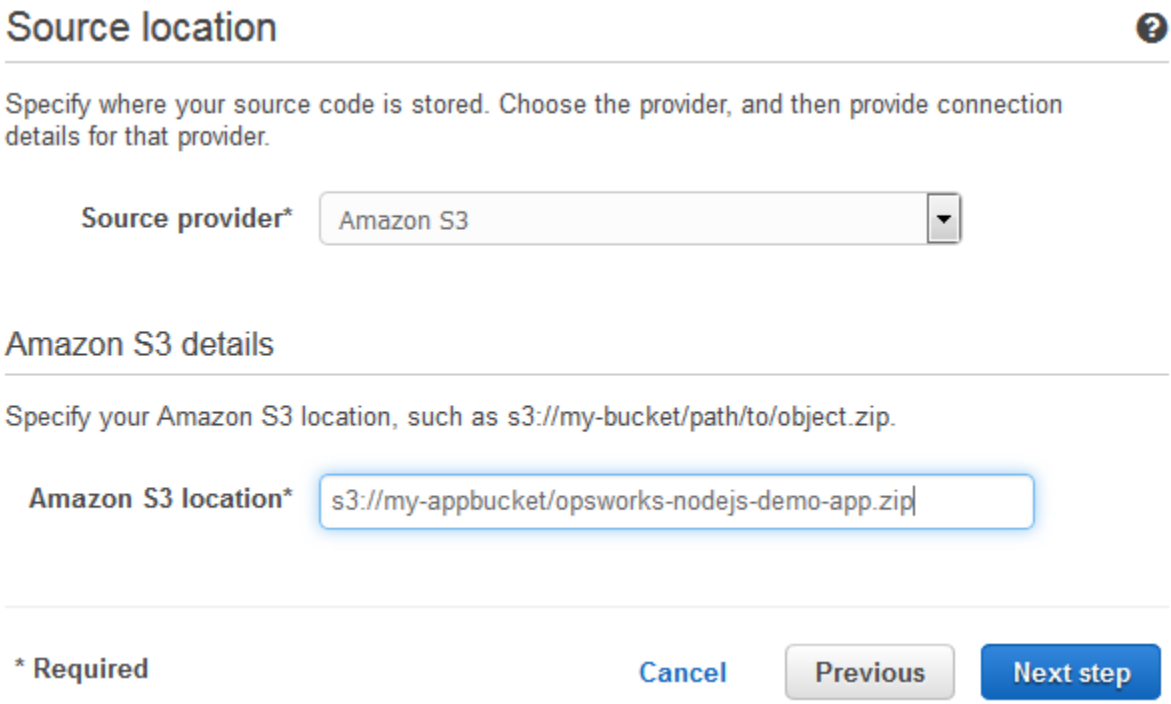
5단계: CodePipeline에서 파이프라인 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

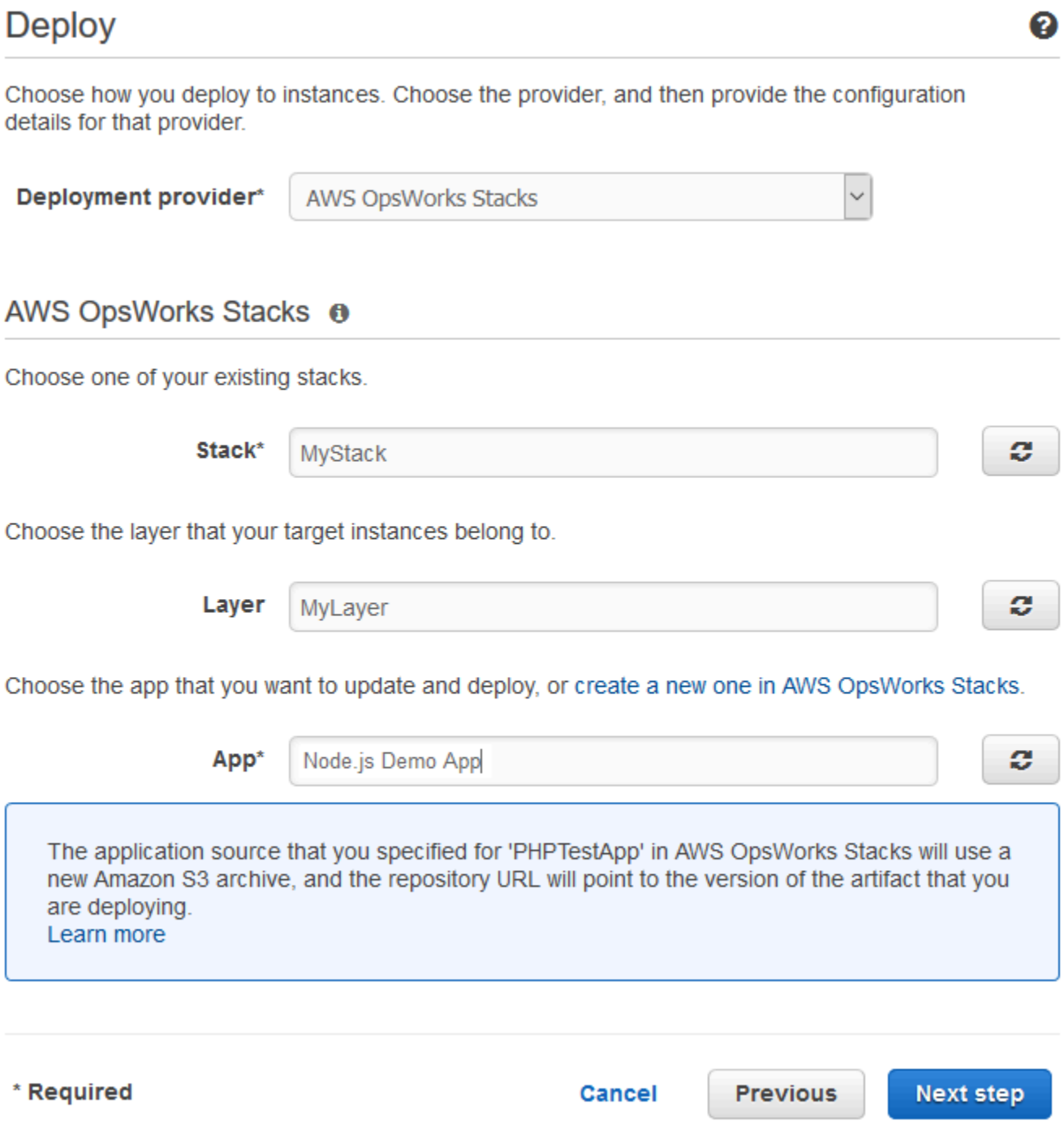
계층이 있는 스택과 AWS OpsWorks Stacks에 구성된 인스턴스가 하나 이상 있으면 CodePipeline에서 AWS OpsWorks Stacks를 공급자로 사용하여 파이프라인을 생성하여 Stacks 리소스에 앱 또는 Chef AWS OpsWorks 쿡북을 배포합니다.

파이프라인을 만들려면

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
2. [파이프라인 생성]을 선택합니다.
3. CodePipeline 시작하기 페이지에서 **MyOpsWorksPipeline** 또는 사용자 계정에 고유한 다른 파이프라인 이름을 입력하고 다음 단계를 선택합니다.
4. 소스 위치 페이지의 소스 공급자 드롭다운 목록에서 Amazon S3를 선택합니다.
5. Amazon S3 세부 정보 영역에서 Amazon S3 버킷 경로를 **s3://*bucket-name*/*file name*** 형식으로 입력합니다. 3단계: Amazon S3 버킷에 앱 코드 업로드 단원의 9단계에서 적어 둔 링크를 참조하세요. 이 연습에서는 경로가 `s3://my-appbucket/opsworks-nodejs-demo-app.zip`입니다. 다음 단계를 선택합니다.



6. [빌드] 페이지의 드롭다운 목록에서 [빌드 없음]를 선택한 다음 [다음 단계]를 선택합니다.
7. 배포 페이지에서 배포 공급자로 AWS OpsWorks Stacks를 선택합니다.



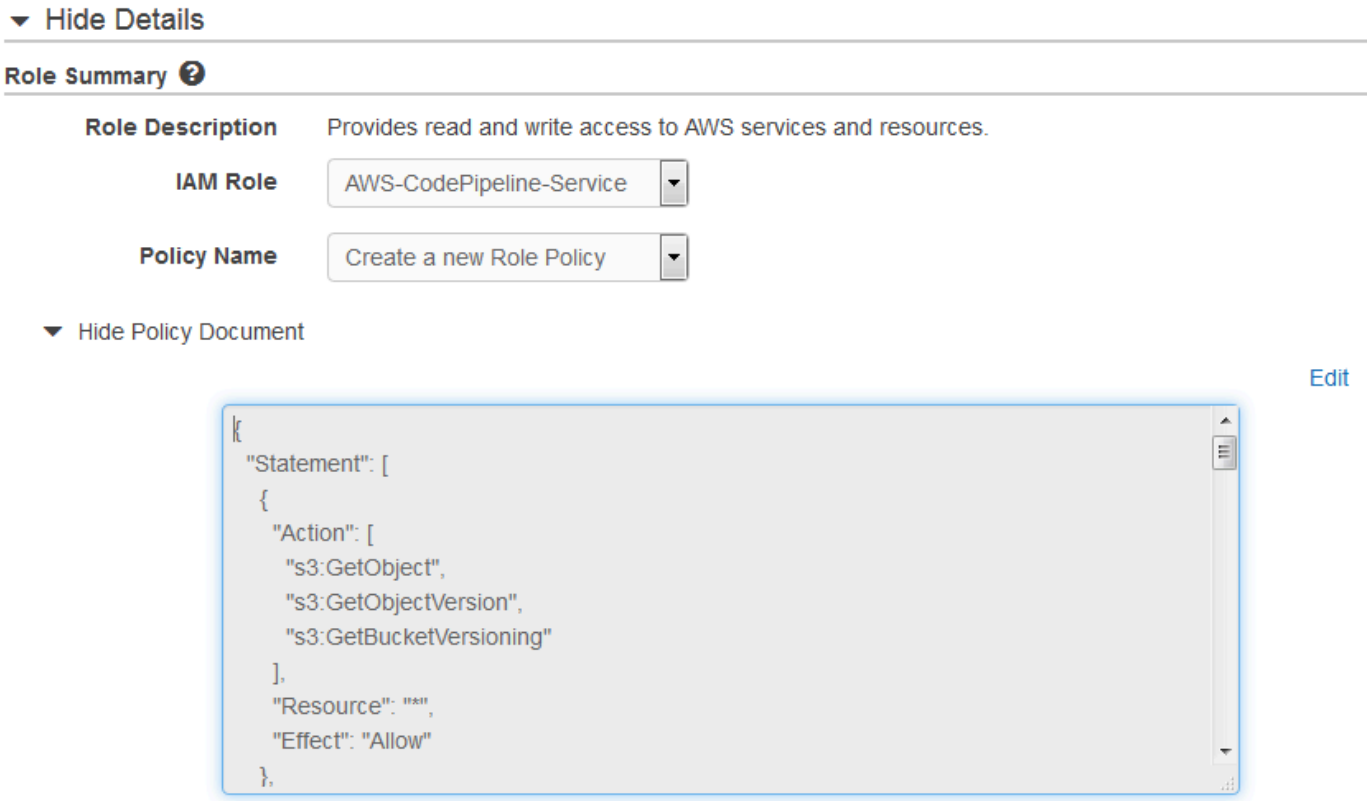
8. [스택] 필드에 `CodePipelineDemo` 또는 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성 단원에서 생성한 스택의 이름을 입력합니다.
9. [계층] 필드에 `Node.js App Server` 또는 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성 단원에서 생성한 계층의 이름을 입력합니다.

10. 앱 필드에서 3단계: Amazon S3 버킷에 앱 코드 업로드의 Amazon S3에 업로드한 앱을 선택한 후 다음 단계를 선택합니다.
11. AWS 서비스 역할 페이지에서 역할 생성을 선택합니다.

    자동으로 생성될 역할인 AWS-CodePipeline-Service을(를) 설명하는 IAM 콘솔 페이지가 새 창에서 열립니다. [정책 이름] 드롭다운 목록에서 [새 정책 만들기]를 선택합니다. 정책 문서에 다음 내용이 포함되어 있는지 확인합니다. 필요한 경우 [편집]을 선택하고 정책 문서를 변경합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    정책 문서의 변경을 마친 뒤 허용을 선택합니다. IAM 콘솔에 변경 내용이 표시됩니다.



> **Note**
>
> 역할 생성에 실패하는 경우 AWS-CodePipeline-Service라는 IAM 역할이 이미 있기 때문일 수 있습니다. 2016년 5월 이전에 AWS-CodePipeline-Service 역할을 사용한 경우 역할에 AWS OpsWorks Stacks를 배포 공급자로 사용할 권한이 없을 수 있습니다. 그 경우 이 단계에서 보여주는 것처럼 정책 설명을 업데이트해야 합니다. 오류 메시지가 표시되면 이 단계의 시작 부분으로 돌아가서 역할 생성 대신 기존 역할 사용을 선택하세요. 기존 역할을 사용하는 경우에는 해당 역할에 이 단계에 표시된 권한이 포함된 정책이 연결되어 있어야 합니다. 서비스 역할 및 해당 정책 설명에 대한 자세한 정보는 IAM 서비스 역할에 대한 정책 편집 단원을 참조하세요.

12. 역할 생성 프로세스에 성공하면 IAM 페이지가 닫히고 AWS 서비스 역할 페이지로 돌아갑니다. 다음 단계를 선택합니다.
13. [검토 your pipeline] 페이지에서 표시된 선택 사항을 확인한 다음 [파이프라인 생성]을 선택합니다.
14. 파이프라인이 준비되면 파이프라인은 자동으로 소스 코드를 찾고 스택에 앱을 배포하기 시작할 것입니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.

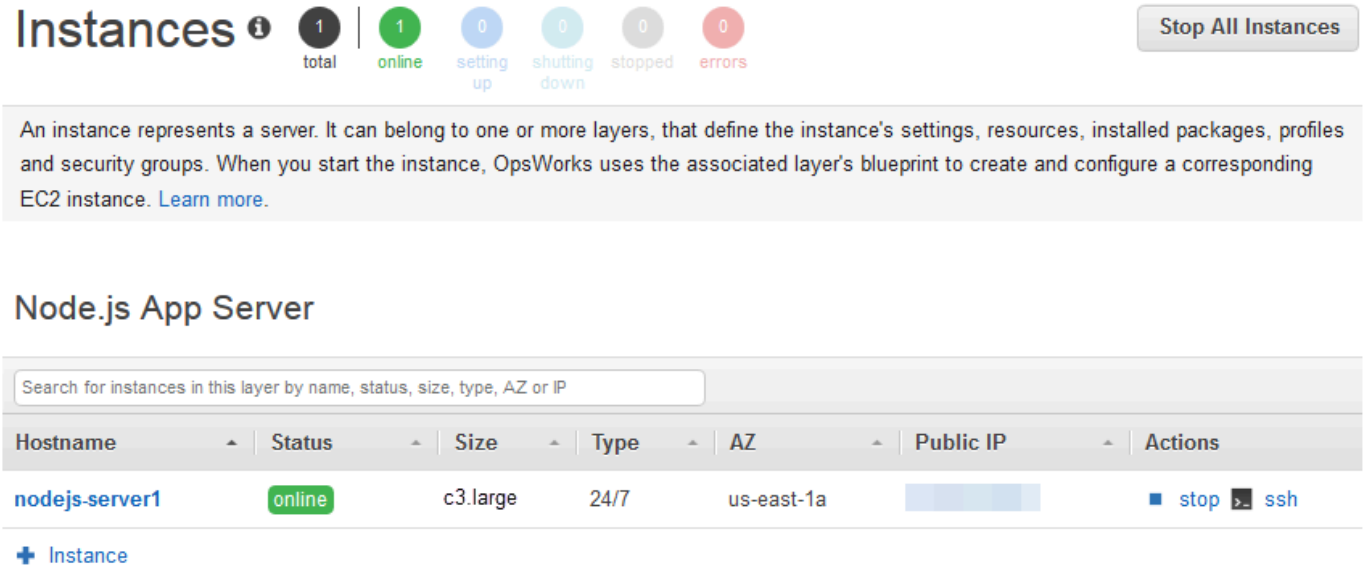
6단계: AWS OpsWorks Stacks에서 앱 배포 확인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

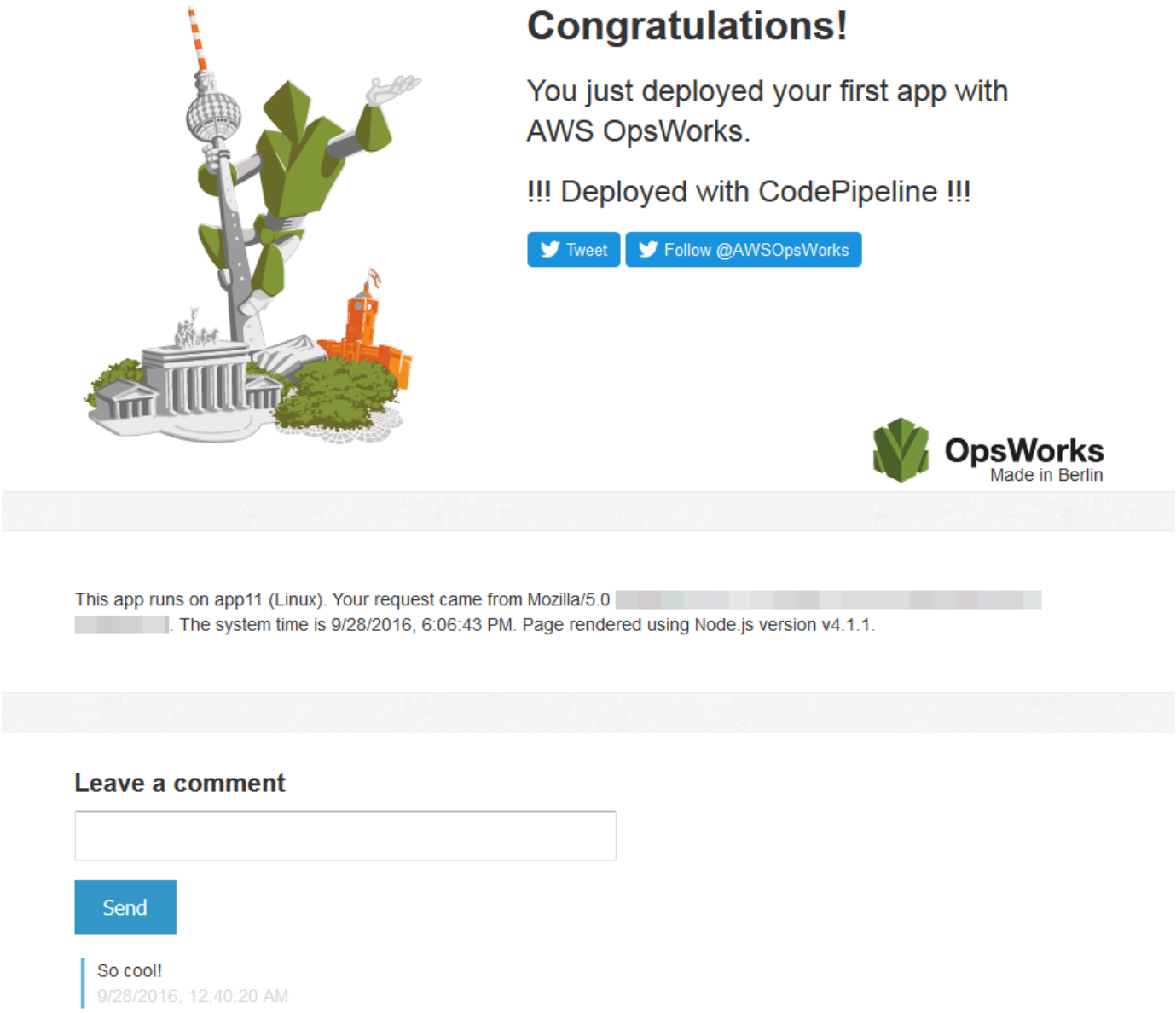
CodePipeline이 스택에 Node.js 앱을 배포했는지 확인하려면 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스 생성 단원에서 생성한 인스턴스에 로그인합니다. Node.js 웹 앱을 보고 사용할 수 있어야 합니다.

AWS OpsWorks Stacks 인스턴스에서 앱 배포를 확인하려면

1. https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks Stacks 대시보드에서 CodePipelineDemo를 선택한 다음 Node.js 앱 서버를 선택합니다.
3. 탐색 창에서 [인스턴스]를 선택한 다음 웹 앱을 확인하기 위해 생성한 인스턴스의 퍼블릭 IP 주소를 선택합니다.



새 웹 브라우저 탭에 앱이 표시됩니다.



7단계(선택 사항): 앱 코드를 업데이트하여 CodePipeline이 자동으로 앱을 재배포하는지 확인

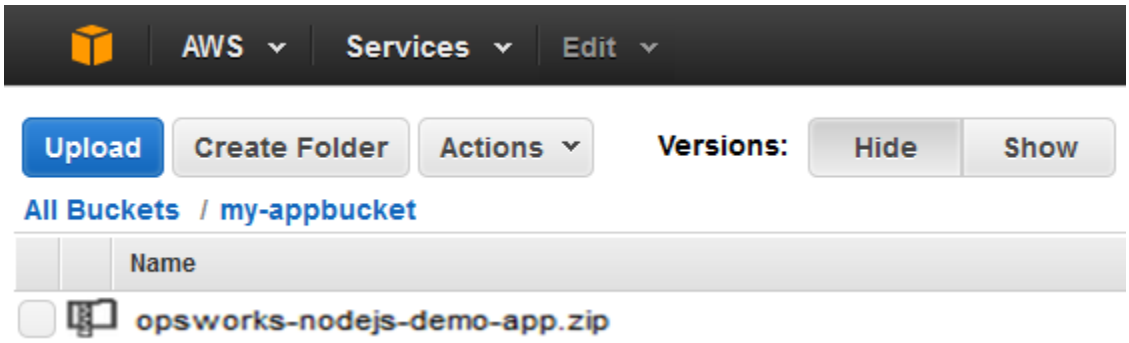
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

CodePipeline을 사용하여 배포한 앱 또는 쿡북에서 코드를 변경하면 업데이트된 아티팩트가 CodePipeline에서 대상 인스턴스(이 경우 대상 AWS OpsWorks Stacks 스택)에 자동으로 배포됩니다. 이 섹션에서는 샘플 Node.js 앱에서 코드를 업데이트할 때의 자동 배포를 보여줍니다. 이 연습을 위한 앱 코드가 여전히 로컬에 저장되어 있고 이 연습을 시작한 이후 아무도 코드를 변경하지 않았다면 이 절차의 1~4단계를 건너뛸 수 있습니다.

샘플 앱에서 코드를 편집하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 샘플 Node.js 앱을 저장하는 버킷을 엽니다.

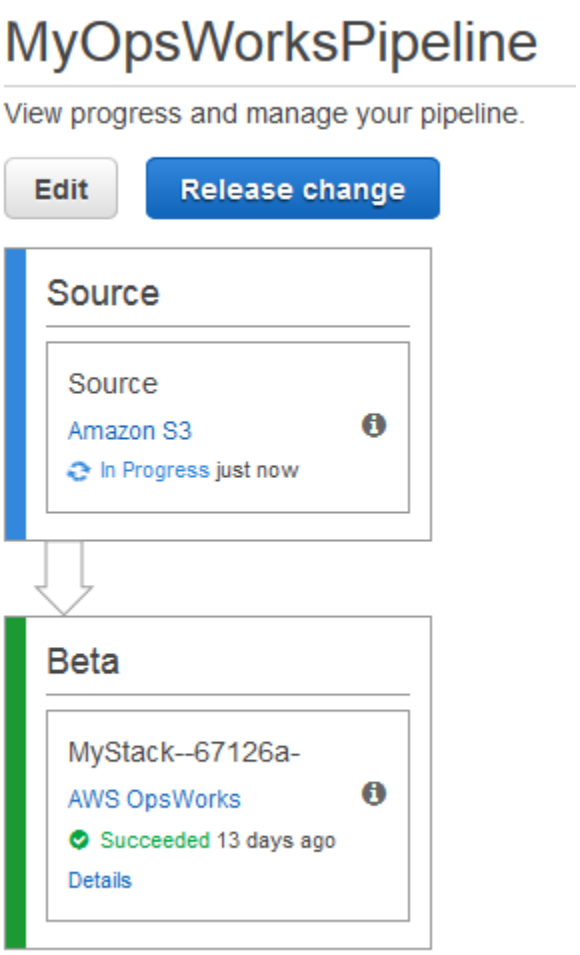


3. 앱이 포함된 ZIP 파일을 선택합니다. [작업] 메뉴에서 [다운로드]를 선택합니다.
4. 대화 상자에서 마우스 오른쪽 버튼을 클릭해 컨텍스트 메뉴를 연 다음 [다운로드]를 선택하고 ZIP 파일을 원하는 위치에 저장합니다. 확인을 선택합니다.
5. ZIP 파일의 내용을 원하는 위치로 추출합니다. 편집을 허용하려면 추출한 폴더, 하위 폴더 및 내용에 대한 권한을 변경해야 할 수 있습니다. `opsworks-nodejs-demo-app\views` 폴더에서 편집할 `header.html` 파일을 엽니다.
6. `You just deployed your first app with`이라는 문구를 검색합니다. 단어 `deployed`를 `updated`로 바꿉니다. 다음 행에서 `AWS OpsWorks.`를 `AWS OpsWorks and AWS CodePipeline.`으로 바꾸되, 이 텍스트를 제외한 어떤 부분도 편집하지 마십시오.

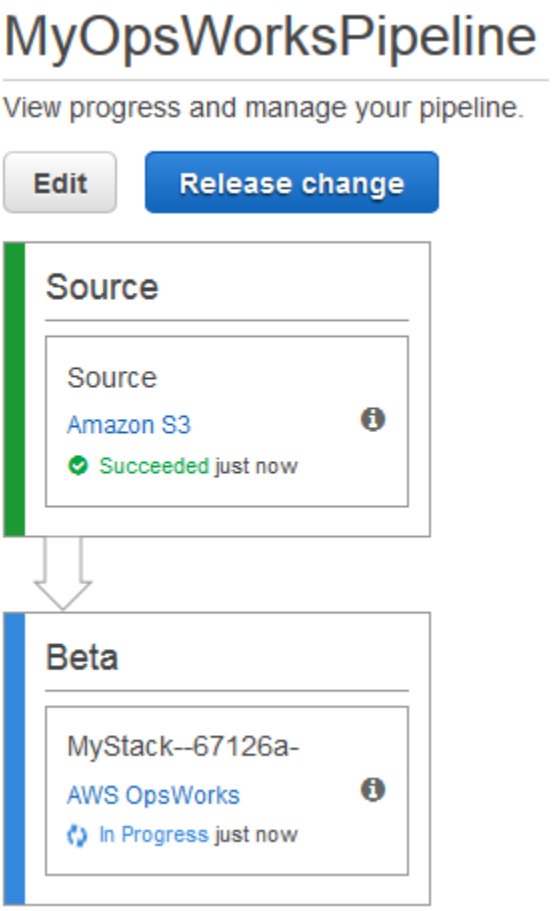
```
<div id="main" role="main">LF
  <div class="container">LF
    <div class="hero-unit">LF
      <div class="robot">LF
        <h1>Congratulations!</h1>LF
        <h2>LF
          You just updated your first app with<br/>LF
          AWS OpsWorks and AWS CodePipeline.LF
        </h2>LF
```

7. header.html 파일을 저장하고 닫습니다.
8. opsworks-nodejs-demo-app 폴더를 압축하여 ZIP 파일을 원하는 위치에 저장합니다. ZIP 파일의 이름은 변경하지 마십시오.
9. Amazon S3 버킷에 새로운 ZIP 파일을 업로드합니다. 이 연습에서 버킷의 이름은 my-appbucket입니다.
10. CodePipeline 콘솔을 열고 AWS OpsWorks Stacks 파이프라인(MyOpsWorksPipeline)을 엽니다. [릴리스 변경]을 선택합니다.

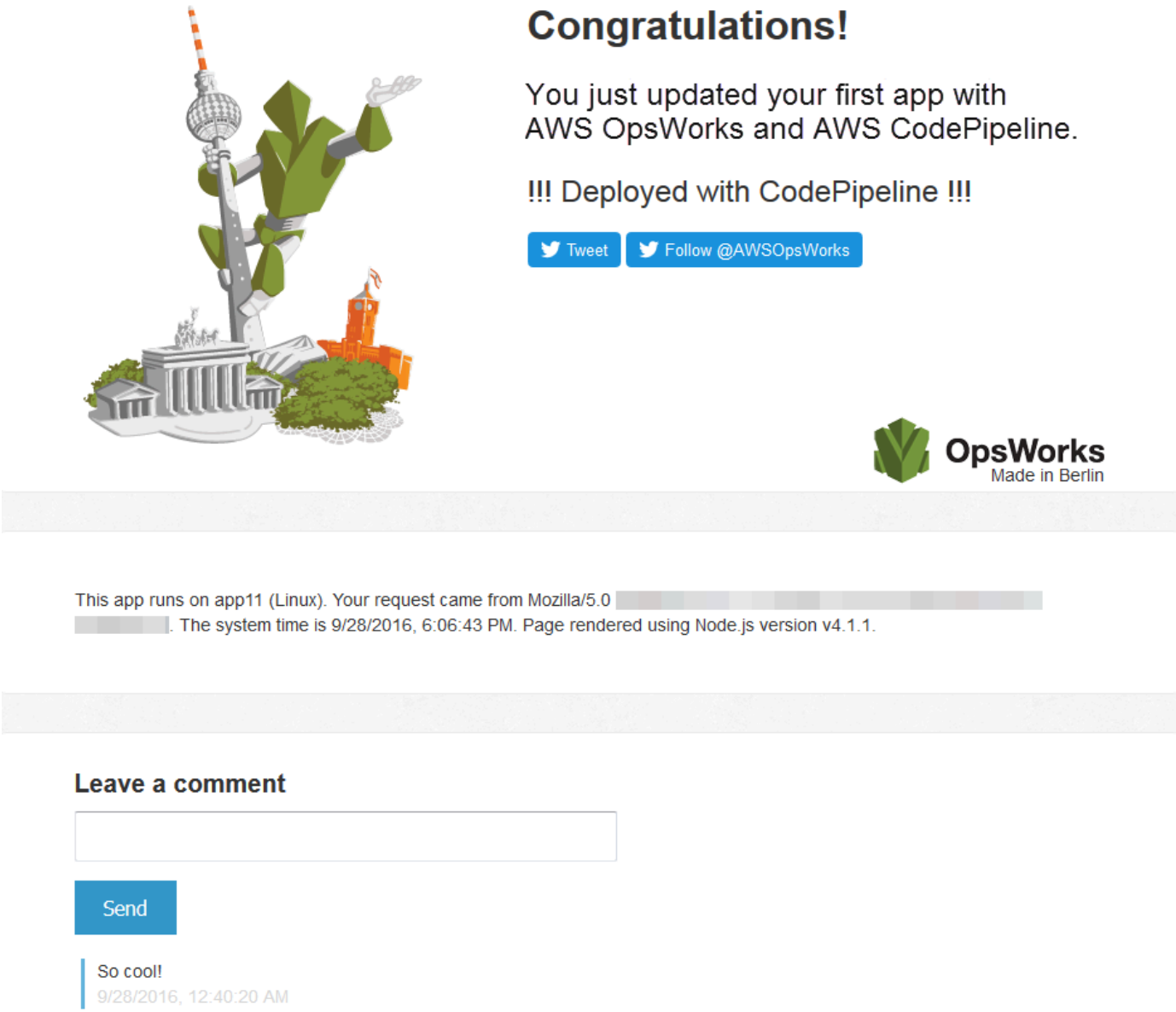
    (CodePipeline이 Amazon S3 버킷의 업데이트된 앱 버전에서 코드 변경을 감지할 때까지 기다릴 수 있습니다. 시간을 절약하기 위해 이 연습에서는 릴리스 변경을 선택하기만 하면 됩니다.)
11. 파이프라인의 단계를 따라 CodePipeline이 실행되는 것을 지켜봅니다. 먼저, CodePipeline이 소스 아티팩트에 대한 변경 사항을 찾습니다.



CodePipeline은 업데이트된 코드를 AWS OpsWorks Stacks의 스택으로 푸시합니다.



12. 파이프라인의 두 단계 모두 완료되면 AWS OpsWorks Stacks에서 스택을 엽니다.
13. 스택 속성 페이지에서 인스턴스를 선택합니다.
14. 퍼블릭 IP 열에서 인스턴스의 퍼블릭 IP 주소를 선택하여 업데이트된 앱의 텍스트를 확인합니다.



8단계(선택 사항): 리소스 정리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션

하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 계정에 원치 않는 요금이 부과되는 것을 방지하기 위해이 연습에 사용한 AWS 리소스를 삭제할 수 있습니다. 이러한 AWS 리소스에는 AWS OpsWorks Stacks 스택, IAM 역할 및 인스턴스 프로파일, CodePipeline에서 생성한 파이프라인이 포함됩니다. 그러나 AWS OpsWorks Stacks 및 CodePipeline에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용하고 싶을 수 있습니다. 이러한 리소스를 유지하려면 이 연습을 마쳐야 합니다.

스택에서 앱을 삭제하려면

AWS CloudFormation 템플릿의 일부로 앱을 생성하거나 적용하지 않았으므로 스택을 삭제하기 전에 Node.js 테스트 앱을 삭제합니다 AWS CloudFormation.

1. AWS OpsWorks Stacks 콘솔의 서비스 탐색 창에서 앱을 선택합니다.
2. [앱] 페이지에서 [Node.js 데모 앱]을 선택한 다음 [작업]에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 앱을 삭제합니다.

스택을 삭제하려면

AWS CloudFormation 템플릿을 실행하여 스택을 생성했으므로 AWS CloudFormation 콘솔에서 템플릿이 생성한 계층, 인스턴스, 인스턴스 프로파일 및 보안 그룹을 포함하여 스택을 삭제할 수 있습니다.

1. AWS CloudFormation 콘솔을 엽니다.
2. AWS CloudFormation 콘솔 대시보드에서 생성한 스택을 선택합니다. 작업 메뉴에서 스택 삭제를 선택합니다. 확인 메시지가 표시되면 예, 삭제를 선택합니다.
3. 스택의 상태 열에 DELETE_COMPLETE가 표시될 때까지 기다리세요.

파이프라인을 삭제하려면

1. CodePipeline 콘솔을 엽니다.
2. CodePipeline 대시보드에서 이 연습을 위해 생성한 파이프라인을 선택합니다.
3. 파이프라인 페이지에서 편집을 선택합니다.
4. [편집] 페이지에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다.

AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 11 스택

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS CodePipeline을(를) 이용해 CodeCommit, Amazon Simple Storage Service(S3) 또는 GitHub와 같은 소스에서 코드 변경을 추적하는 지속적 전송 파이프라인을 생성할 수 있습니다. 이 항목의 예제는 CodePipeline에서 간단한 파이프라인을 생성하여 AWS OpsWorks Stacks 계층에서 실행할 코드의 배포 도구로 사용하는 방법을 설명합니다. 이 예제에서는 간단한 PHP 앱에 대한 파이프라인을 생성한 다음 Chef AWS OpsWorks 11.10 스택(이 경우 단일 인스턴스)의 계층에 있는 모든 인스턴스에서 앱을 실행하도록 Stacks에 지시합니다.

> **Note**
>
> 이 항목은 파이프라인을 사용하여 Chef 11.10 스택에서 앱을 실행하고 업데이트하는 방법을 설명합니다. 파이프라인을 사용하여 Chef 12 스택에서 앱을 실행하고 업데이트하는 방법에 대한 자세한 정보는 AWS CodePipelineAWS OpsWorks 스택 포함 - Chef 12 스택 섹션을 참조하세요. Amazon S3 버킷에 전달한 콘텐츠에는 고객 콘텐츠가 포함될 수 있습니다. 중요 데이터 제거에 관한 자세한 내용은 S3 버킷을 비우려면 어떻게 해야 합니까? 단원 또는 S3 버킷을 삭제하려면 어떻게 해야 합니까? 단원을 참조하세요.

주제

- 사전 조건
- 지원되는 다른 시나리오
- 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성합니다.
- 2단계: Amazon S3 버킷에 앱 코드 업로드
- 3단계: AWS OpsWorks Stacks에 앱 추가
- 4단계: CodePipeline에서 파이프라인 생성
- 5단계: AWS OpsWorks Stacks에서 앱 배포 확인

- 6단계(선택 사항): 앱 코드를 업데이트하여 CodePipeline이 자동으로 앱을 재배포하는지 확인
- 7단계(선택 사항): 리소스 정리

사전 조건

이 연습을 시작하기 전에 다음 작업을 모두 수행할 수 있는 관리자 권한이 있는지 확인하세요. AdministratorAccess 정책이 적용되는 그룹의 멤버일 수도 있고, 다음 표의 권한과 정책이 있는 그룹의 멤버일 수도 있습니다. 보안 모범 사례로서, 개별 사용자에 필요한 권한을 할당하는 대신 사용자가 다음 작업을 수행할 수 있는 권한을 보유한 그룹에 속해야 합니다.

IAM에서 보안 그룹을 생성하고 해당 그룹에 권한을 할당하는 자세한 정보는 IAM 사용자 그룹 만들기를 참조하세요. AWS OpsWorks Stacks 권한 관리에 대한 자세한 내용은 모범 사례: 권한 관리를 참조하세요.

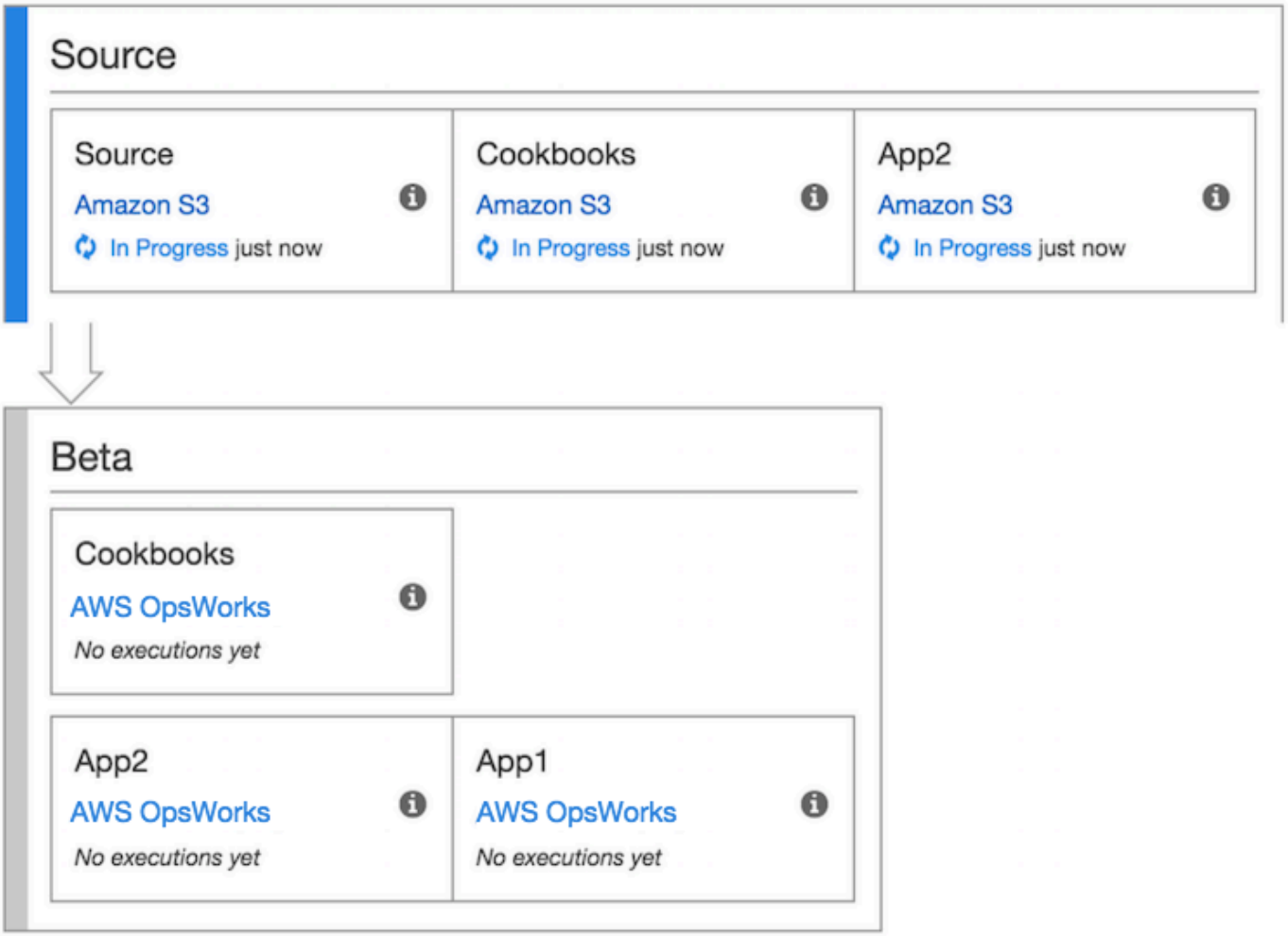
권한	그룹에 연결할 권장 정책
AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성하고 편집합니다.	AWSOpsWorks_FullAccess
AWS CloudFormation에서 템플릿을 생성하고, 편집하고, 실행합니다.	AmazonCloudFormationFullAccess
Amazon S3 버킷을 생성하고 편집하고 액세스합니다.	AmazonS3FullAccess
CodePipeline에서 파이프라인, 특히 AWS OpsWorks Stacks를 공급자로 사용하는 파이프라인을 생성, 편집 및 실행합니다.	AWSCodePipeline_FullAccess

Amazon EC2 키 페어도 있어야 합니다. 이 연습에서 샘플 스택, 계층 및 인스턴스를 생성하는 AWS CloudFormation 템플릿을 실행할 때이 키 페어의 이름을 입력하라는 메시지가 표시됩니다. Amazon EC2 콘솔에서 키 페어를 얻는 방법에 대한 자세한 내용은 Amazon EC2 문서의 키 페어 생성을 참조하세요. 키 페어는 미국 동부(버지니아 북부) 리전에 있어야 합니다. 해당 리전에 이미 키 페어가 있는 경우 기존 키 페어를 사용할 수 있습니다.

지원되는 다른 시나리오

이 연습에서는 Source 단계와 Deploy 단계를 하나씩 포함하는 간단한 파이프라인을 생성합니다. 그러나 AWS OpsWorks Stacks를 공급자로 사용하는 더 복잡한 파이프라인을 생성할 수 있습니다. 다음은 지원되는 파이프라인 및 시나리오의 예입니다.

- 파이프라인을 편집하여 Chef 쿡북을 Source 단계에서 추가하고 업데이트된 쿡북의 연결된 대상을 Deploy 단계에 추가할 수 있습니다. 이 경우, 사용자가 소스를 변경할 때 쿡북 업데이트를 트리거하는 Deploy 작업을 추가합니다. 업데이트된 쿡북은 앱보다 먼저 배포됩니다.
- 사용자 지정 쿡북과 여러 앱을 사용하여 복잡한 파이프라인을 생성하고 AWS OpsWorks Stacks 스택에 배포할 수 있습니다. 파이프라인은 애플리케이션 및 쿡북 소스에 대한 변경 사항을 추적하고 사용자가 변경할 경우 재배포합니다. 다음은 복잡한 파이프라인의 예입니다.



CodePipeline을 사용하는 방법에 대한 자세한 내용을 알아보려면 CodePipeline 문서를 참조하세요.

1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성합니다.

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks를 파이프라인의 배포 공급자로 사용하려면 먼저 스택, 계층 및 계층에 하나 이상의 인스턴스가 있어야 합니다. Linux 스택 시작하기 또는 Windows 스택 시작하기의 지침에 AWS OpsWorks 따라 스택에서 스택을 생성할 수 있지만, 시간을 절약하기 위해이 예제에서는 AWS CloudFormation 템플릿을 사용하여 Linux 기반 Chef 11.10 스택, 계층 및 인스턴스를 생성합니다. 이 템플릿에 의해 생성된 인스턴스는 Amazon Linux 2016.03을 실행하며 인스턴스 유형은 `c3.large`입니다.

> **Important**
>
> AWS CloudFormation 템플릿은 나중에 앱을 업로드하는 Amazon S3 버킷과 동일한 리전과 나중에 CodePipeline에서 파이프라인을 생성하는 리전에 저장하고 실행해야 합니다. 현재 CodePipeline은 미국 동부(버지니아 북부) 리전(us-east-1)에서만 AWS OpsWorks Stacks 공급자를 지원합니다. 이 연습에서는 모든 리소스를 미국 동부(버지니아 북부) 리전에서 생성해야 합니다.
>
> 스택 생성이 실패할 경우 계정에 허용되는 최대 수의 IAM 역할에 접근한 것일 수 있습니다. 계정에서 `c3.large` 인스턴스 유형의 인스턴스를 시작할 수 없는 경우에도 스택 생성이 실패할 수 있습니다. 예를 들어 AWS 프리 티어를 사용 중이라면 `Root device type: must be included in EBS` 같은 오류가 반환될 수 있습니다. 계정에 AWS 프리 티어에서 부과하는 제한과 같이 생성할 수 있는 인스턴스 유형에 대한 제한이 있는 경우 템플릿의 인스턴스 블록에 있는 `InstanceType` 파라미터의 값을 계정에서 사용할 수 있는 인스턴스 유형으로 변경해 보세요.

를 사용하여 스택, 계층 및 인스턴스를 생성하려면 AWS CloudFormation

1. 다음 AWS CloudFormation 템플릿을 새 일반 텍스트 문서에 복사합니다. 로컬 컴퓨터의 편리한 위치에 파일을 저장하고 NewOpsWorksStack.template 또는 원하는 다른 이름을 지정합니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "eu-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
```

```
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
 },
 "Parameters": {
   "EC2KeyPairName": {
  "Type": "String",
  "Description": "The name of an existing EC2 key pair that allows you to use SSH
to connect to the OpsWorks instance."
 }
 },
 "Resources": {
"CPOpsDeploySecGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription" : "Lets you manage OpsWorks instances deployed to by
CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
     }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
```

```
    "Fn::GetAtt": [
      "CPOpsDeploySecGroup", "GroupId"
    ]
  }
   }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
           "OpsWorksServiceRole",
           "Arn"
         ]
       },
 "ConfigurationManager" : { "Name": "Chef","Version": "11.10" },
 "DefaultOs": "Amazon Linux 2016.03",
       "DefaultInstanceProfileArn": {
         "Fn::GetAtt": [
           "OpsWorksInstanceProfile",
           "Arn"
         ]
       }
     }
   },
   "MyLayer": {
     "Type": "AWS::OpsWorks::Layer",
     "Properties": {
       "StackId": {
         "Ref": "MyStack"
       },
       "Name": "MyLayer",
       "Type": "php-app",
 "Shortname": "mylayer",
       "EnableAutoHealing": "true",
       "AutoAssignElasticIps": "false",
       "AutoAssignPublicIps": "true",
 "CustomSecurityGroupIds": [
   {
     "Fn::GetAtt": [
             "CPOpsDeploySecGroup", "GroupId"
```

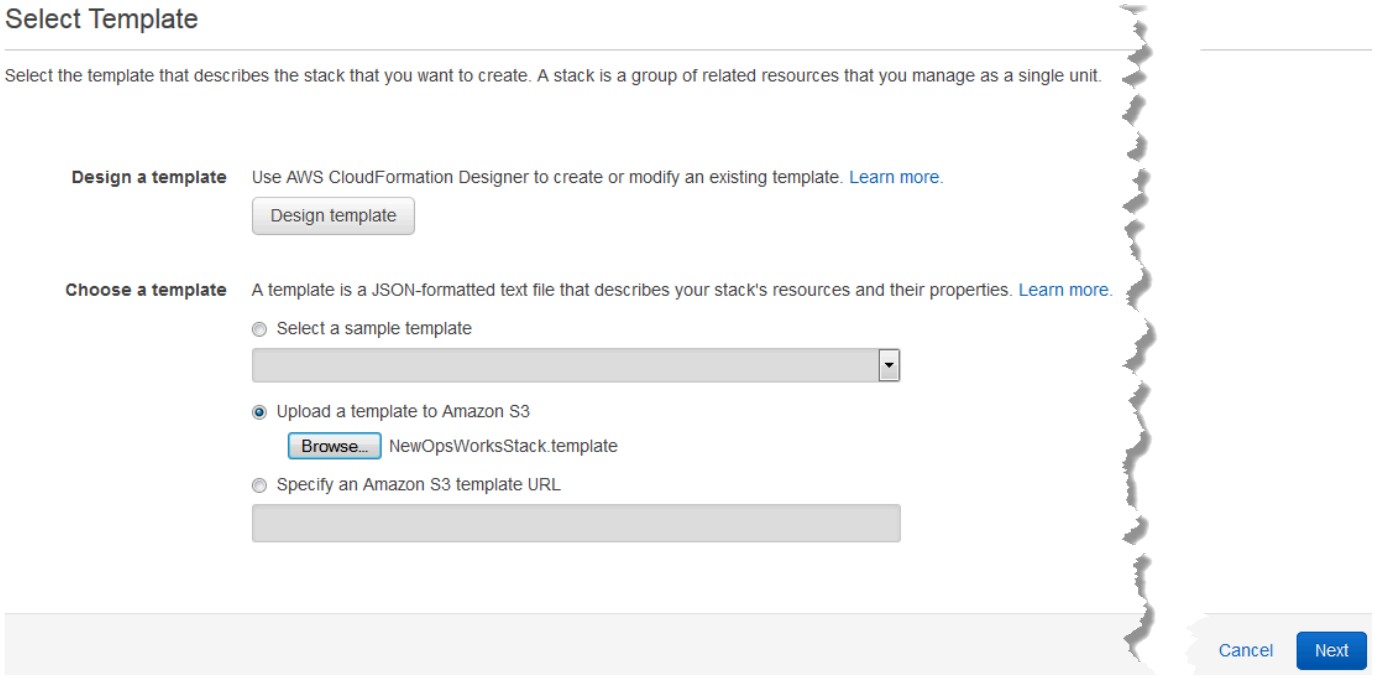
```
      ]
          }
        ]
      },
      "DependsOn": [
        "MyStack",
        "CPOpsDeploySecGroup"
      ]
    },
    "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "OpsWorksPrincipal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "opsworks-service",
            "PolicyDocument": {
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": [
```

```
                  "ec2:*",
                  "iam:PassRole",
                  "cloudwatch:GetMetricStatistics",
                  "elasticloadbalancing:*"
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "OpsWorksInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
      "Path": "/",
      "Roles": [
        {
          "Ref": "OpsWorksInstanceRole"
        }
      ]
    }
  },
  "OpsWorksInstanceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": [
                {
                  "Fn::FindInMap": [
                    "Region2Principal",
                    {
                      "Ref": "AWS::Region"
                    },
                    "EC2Principal"
                  ]
                }
              ]
            },
```

```
            "Action": [
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "Path": "/",
"Policies": [
        {
          "PolicyName": "s3-get",
          "PolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": [
                  "s3:GetObject"
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "myinstance": {
    "Type": "AWS::OpsWorks::Instance",
    "Properties": {
      "LayerIds": [
        {
          "Ref": "MyLayer"
        }
      ],
      "StackId": {
        "Ref": "MyStack"
      },
      "InstanceType": "c3.large",
      "SshKeyName": {
  "Ref": "EC2KeyPairName"
}
    }
  }
},
```

```
  "Outputs": {
    "StackId": {
      "Description": "Stack ID for the newly created AWS OpsWorks stack",
      "Value": {
        "Ref": "MyStack"
      }
    }
  }
}
```

2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation AWS CloudFormation 콘솔을 엽니다.
3. AWS CloudFormation 홈 페이지에서 스택 생성을 선택합니다.
4. [템플릿 선택] 페이지의 [템플릿 선택] 영역에서 [Amazon S3에 템플릿 업로드]를 선택한 다음 [찾아보기]를 선택합니다.
5. 1단계에서 저장한 AWS CloudFormation 템플릿을 찾아 열기를 선택합니다. 템플릿 선택 페이지에서 다음을 선택합니다.



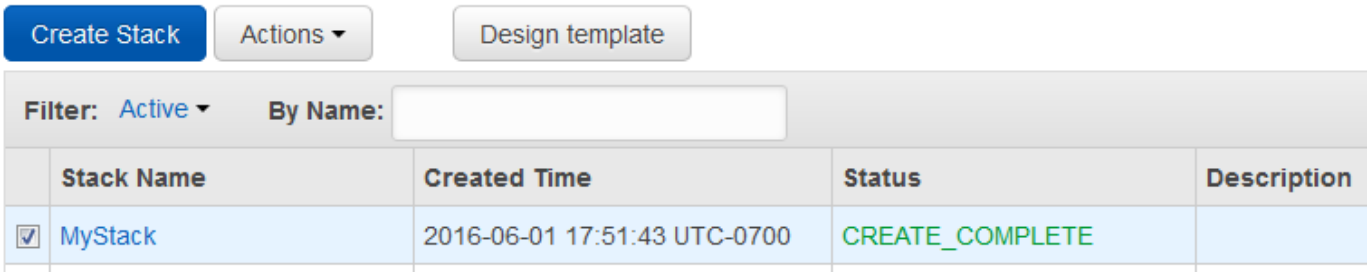
6. [세부 정보 지정] 페이지에서 스택의 이름을 [MyStack] 또는 계정에 고유한 다른 스택 이름으로 지정합니다. 스택에 다른 이름을 선택하면 이 연습 전체에서 해당 스택 이름을 변경해야 합니다.
7. 파라미터 영역에서 AWS OpsWorks Stacks 인스턴스가 생성된 후 인스턴스에 액세스하는 데 사용할 EC2 키 페어의 이름을 입력합니다. 다음을 선택합니다.
8. 옵션 페이지에서 다음을 선택합니다. 이 페이지의 설정은 이번 연습에는 필요하지 않습니다.

9. 이 연습에서 사용하는 AWS CloudFormation 템플릿은 IAM 역할, 인스턴스 프로파일 및 인스턴스를 생성합니다.

> **Important**
>
> 생성을 선택하기 전에 비용을 선택하여이 템플릿으로 리소스를 생성하기 AWS 위해 발생할 수 있는 요금을 추정합니다.

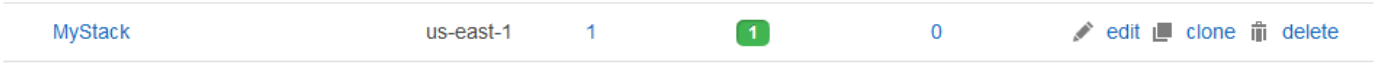
   IAM 리소스 생성이 가능한 경우 이 템플릿으로 인해 AWS CloudFormation에서 IAM 리소스를 생성할 수 있음을 인정 확인란을 선택한 다음 생성을 선택합니다. IAM 리소스를 생성할 수 없는 경우 이 절차를 계속 진행할 수 없습니다.

10. AWS CloudFormation 대시보드에서 스택 생성 진행 상황을 볼 수 있습니다. 다음 단계를 계속하기 전에 상태 열에 CREATE_COMPLETE가 표시될 때까지 기다리세요.



AWS OpsWorks Stacks에서 스택 생성을 확인하려면

1. https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks Stacks 대시보드에서 생성한 스택을 확인합니다.



3. 스택을 열고 계층 및 인스턴스를 확인합니다. AWS CloudFormation 템플릿에 제공된 이름 및 기타 메타데이터를 사용하여 계층과 인스턴스가 생성되었는지 확인합니다. Amazon S3 버킷으로 앱을 업로드할 준비가 되었습니다.

2단계: Amazon S3 버킷에 앱 코드 업로드

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

파이프라인 설정의 일환으로 코드 리포지토리에 대한 링크를 제공해야 하므로 파이프라인을 설정하기 전에 코드 리포지토리를 준비하세요. 이 연습에서는 PHP 앱을 Amazon S3 버킷에 업로드합니다.

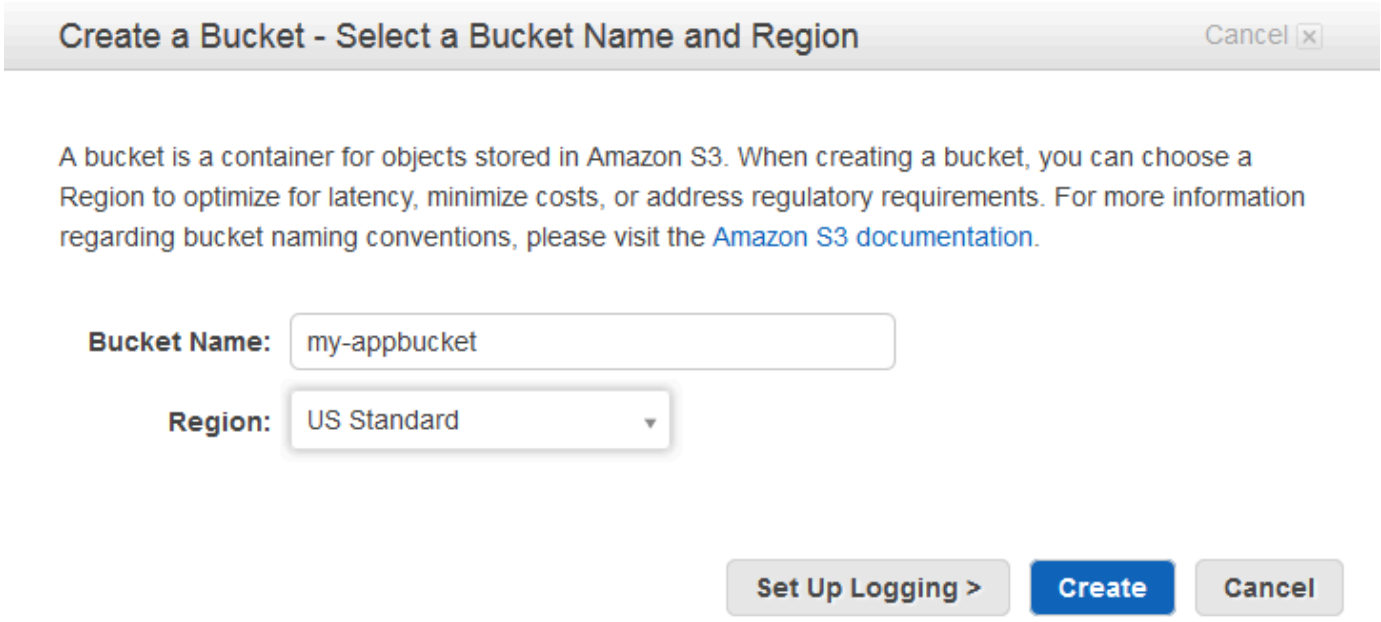
CodePipeline은 소스로 GitHub 또는 CodeCommit로부터 바로 코드를 사용할 수 있지만, 이 연습에서는 Amazon S3 버킷을 사용하는 방법을 예시합니다. Amazon S3 버킷은 CodePipeline이 앱 코드 변경을 감지하고 변경된 앱을 자동으로 배포할 수 있게 합니다. 원한다면 기존 버킷을 사용해도 됩니다. 버킷이 CodePipeline 문서의 간단한 파이프라인 연습(Amazon S3 버킷)에 설명된 기준을 충족하는지 확인하세요.

> **Important**
>
> Amazon S3 버킷은 나중에 파이프라인을 생성할 리전과 동일한 리전에 위치해야 합니다. 현재 CodePipeline은 미국 동부(버지니아 북부) 리전(us-east-1)에서만 AWS OpsWorks Stacks 공급자를 지원합니다. 이 연습에서는 모든 리소스를 미국 동부(버지니아 북부) 리전에서 생성해야 합니다. 또한 CodePipeline은 버전 관리된 소스를 요구하므로 버킷의 버전을 관리해야 합니다. 자세한 내용은 버전 관리 사용을 참조하세요.

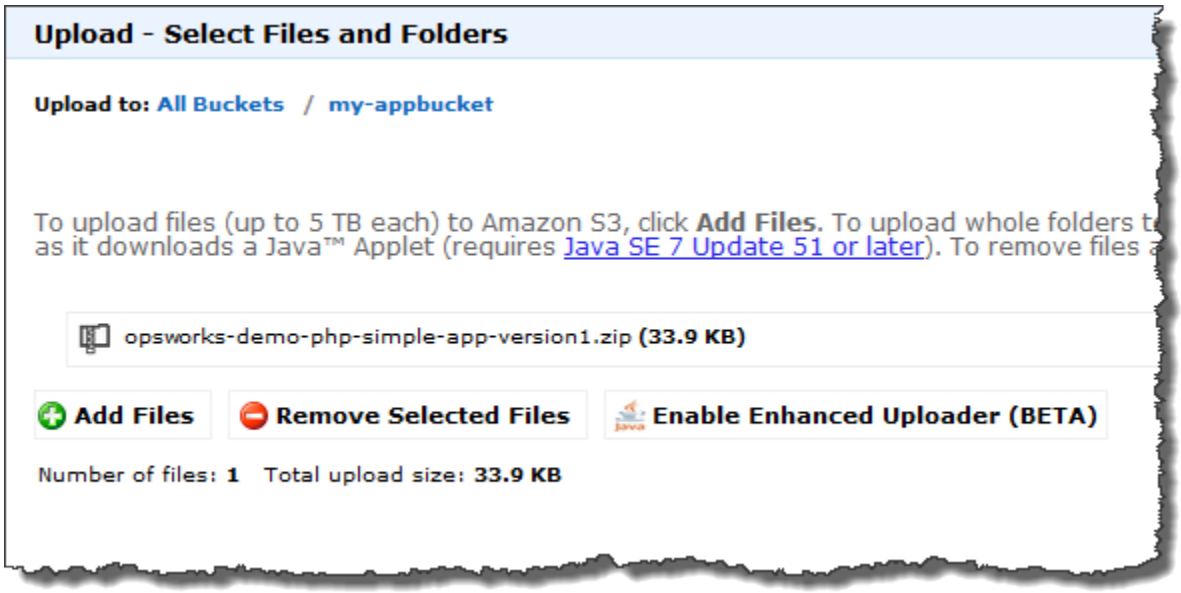
Amazon S3 버킷에 앱을 업로드하려면

1. GitHub 웹 사이트에서 AWS OpsWorks Stacks 샘플 PHP 앱의 ZIP 파일을 다운로드하여 로컬 컴퓨터의 편리한 위치에 저장합니다.
2. `index.php` 및 `ASSETS` 폴더가 다운로드한 ZIP 파일의 루트 레벨에 있는지 확인합니다. 루트 레벨에 없으면 이 파일의 압축을 푼 다음 압축이 풀린 모든 파일이 포함된 ZIP 파일을 루트 레벨에서 새로 만듭니다.
3. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
4. 버킷 생성을 선택합니다.

5. [버킷 생성 - 버킷 이름 및 리전 선택] 페이지에서 [버킷 이름]으로 버킷의 고유한 이름을 입력합니다. 버킷 이름은 자신의 AWS 계정뿐만 아니라 모든 계정에서 고유해야 합니다. 이 연습에서는 **my-appbucket**이라는 이름을 사용하지만 `my-appbucket-`*`yearmonthday`*를 사용하여 버킷 이름을 고유하게 만들 수 있습니다. [리전] 드롭다운 목록에서 [미국 표준]을 선택한 다음 [만들기]를 선택합니다. [미국 표준]는 `[us-east-1]`에 해당합니다.



6. 생성한 버킷을 [모든 버킷] 목록에서 선택합니다.
7. 버킷 페이지에서 업로드를 선택합니다.
8. [업로드 - 파일 및 폴더 선택] 페이지에서 [파일 추가]를 선택합니다. 1단계에서 저장한 ZIP 파일을 찾아 열기를 선택하고 업로드 시작을 선택합니다.



9. 업로드가 완료되면 버킷의 파일 목록에서 ZIP 파일을 선택한 다음 [속성]를 선택합니다.
10. [속성] 창에서 ZIP 파일에 대한 링크를 복사해 적어 둡니다. 파이프라인을 생성하려면 이 링크의 버킷 이름 및 ZIP 파일 이름 부분이 필요합니다.

3단계: AWS OpsWorks Stacks에 앱 추가

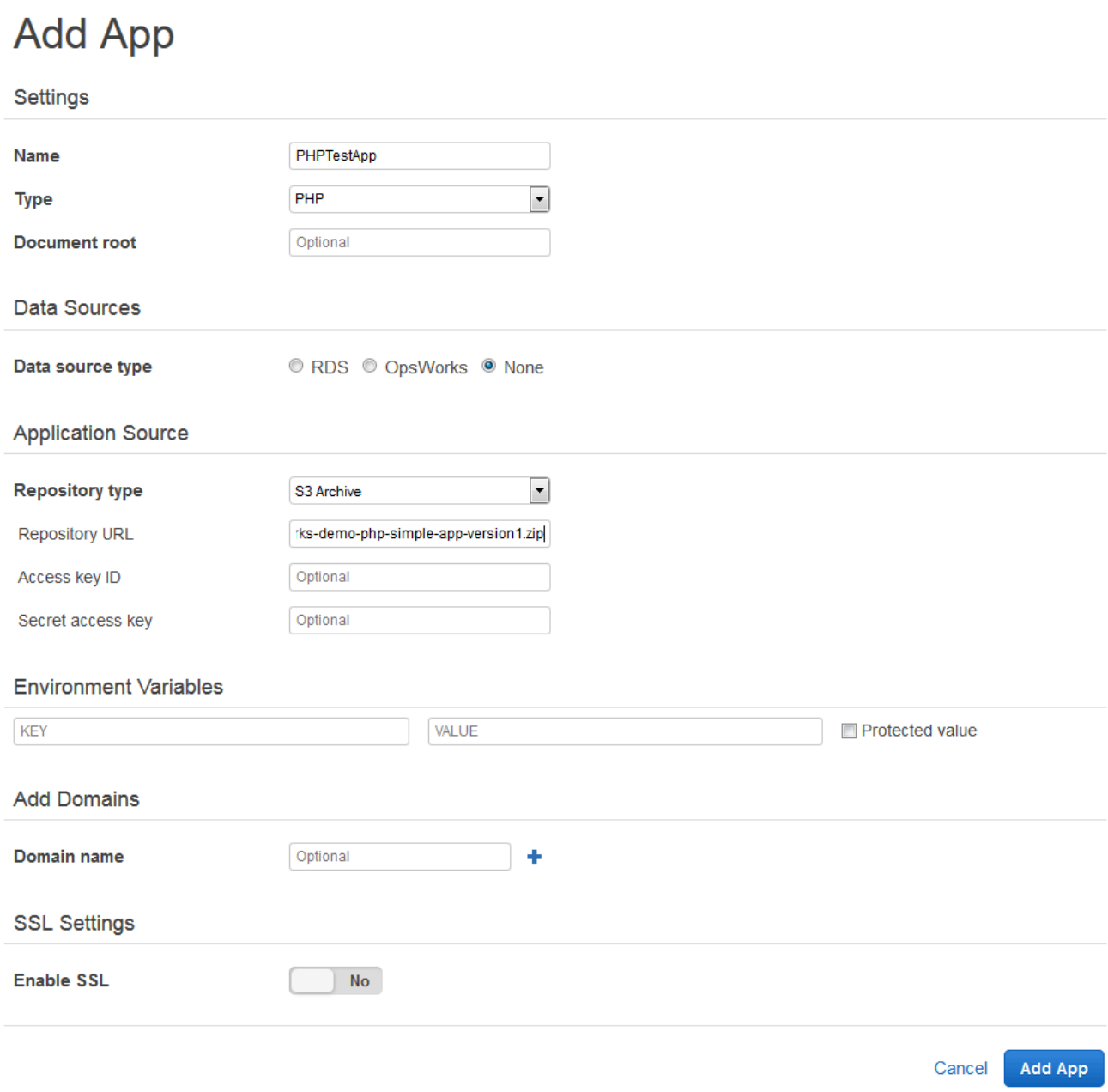
> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

CodePipeline에서 파이프라인을 생성하기 전에 PHP 테스트 앱을 AWS OpsWorks Stacks에 추가합니다. 파이프라인을 생성할 때 AWS OpsWorks Stacks에 추가한 앱을 선택해야 합니다.

이전 절차의 10단계에서 생성한 Amazon S3 버킷 링크를 준비합니다. 이 절차를 완료하려면 테스트 앱을 저장한 버킷에 대한 링크가 필요합니다.

AWS OpsWorks Stacks에 앱을 추가하려면

1. AWS OpsWorks Stacks 콘솔에서 MyStack을 열고 탐색 창에서 앱을 선택합니다.
2. 앱 추가를 선택합니다.
3. 앱 추가 페이지에서 다음 정보를 입력합니다.
   a. 앱의 이름을 지정합니다. 이 연습에서는 `PHPTestApp`이라는 이름을 사용합니다.
   b. [유형] 드롭다운 목록에서 [PHP]를 선택합니다.
   c. 데이터 원본 유형에 대해 없음을 선택합니다. 이 앱에는 외부 데이터베이스 또는 데이터 원본이 필요하지 않습니다.
   d. [리포지토리 유형] 드롭다운 목록에서 [S3 아카이브]를 선택합니다.
   e. [리포지토리 URL] 문자열 상자에 2단계: Amazon S3 버킷에 앱 코드 업로드의 10단계에서 복사한 URL을 붙여 넣습니다. 형식이 다음과 유사해야 합니다.



4. 이 형식의 다른 설정은 변경할 필요가 없습니다. [앱 추가]를 선택합니다.
5. PhpTestApp 앱이 앱 페이지의 목록에 나타나면 다음 절차 4단계: CodePipeline에서 파이프라인 생성을(를) 계속 진행하세요.

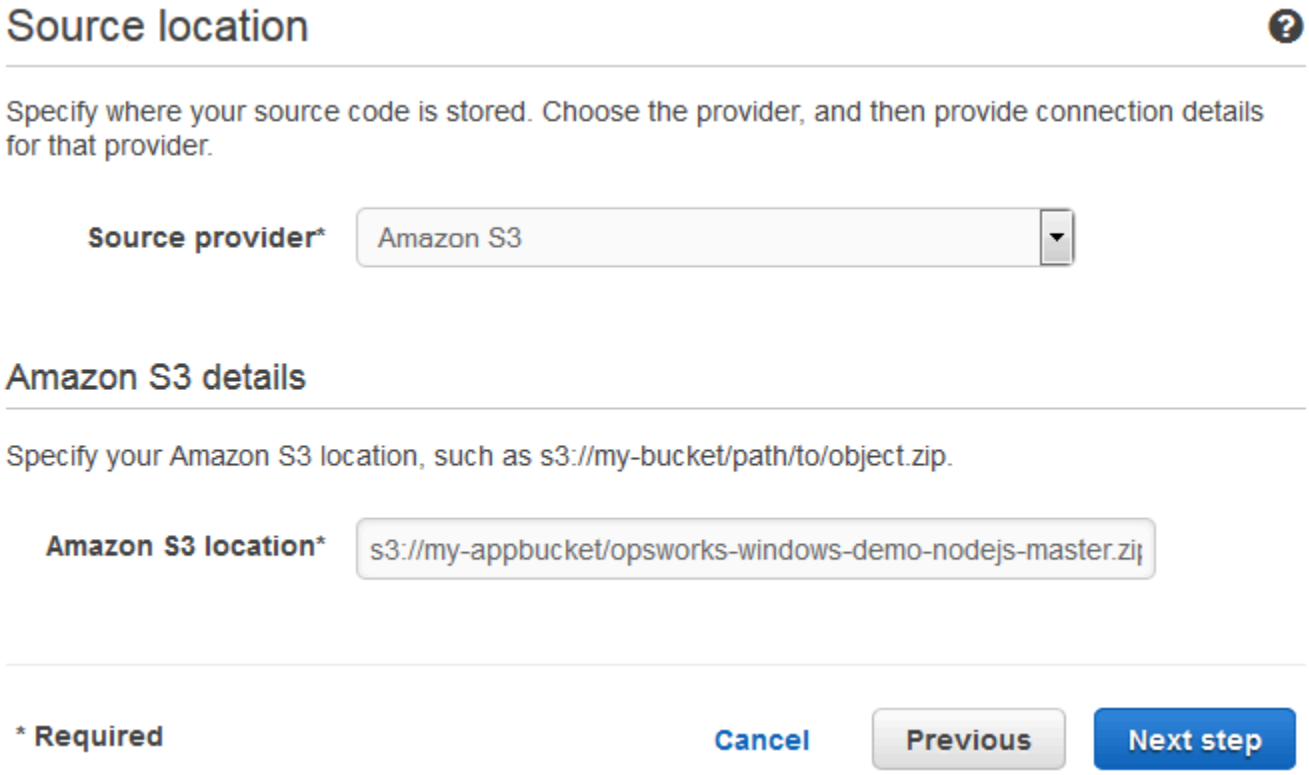
4단계: CodePipeline에서 파이프라인 생성

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

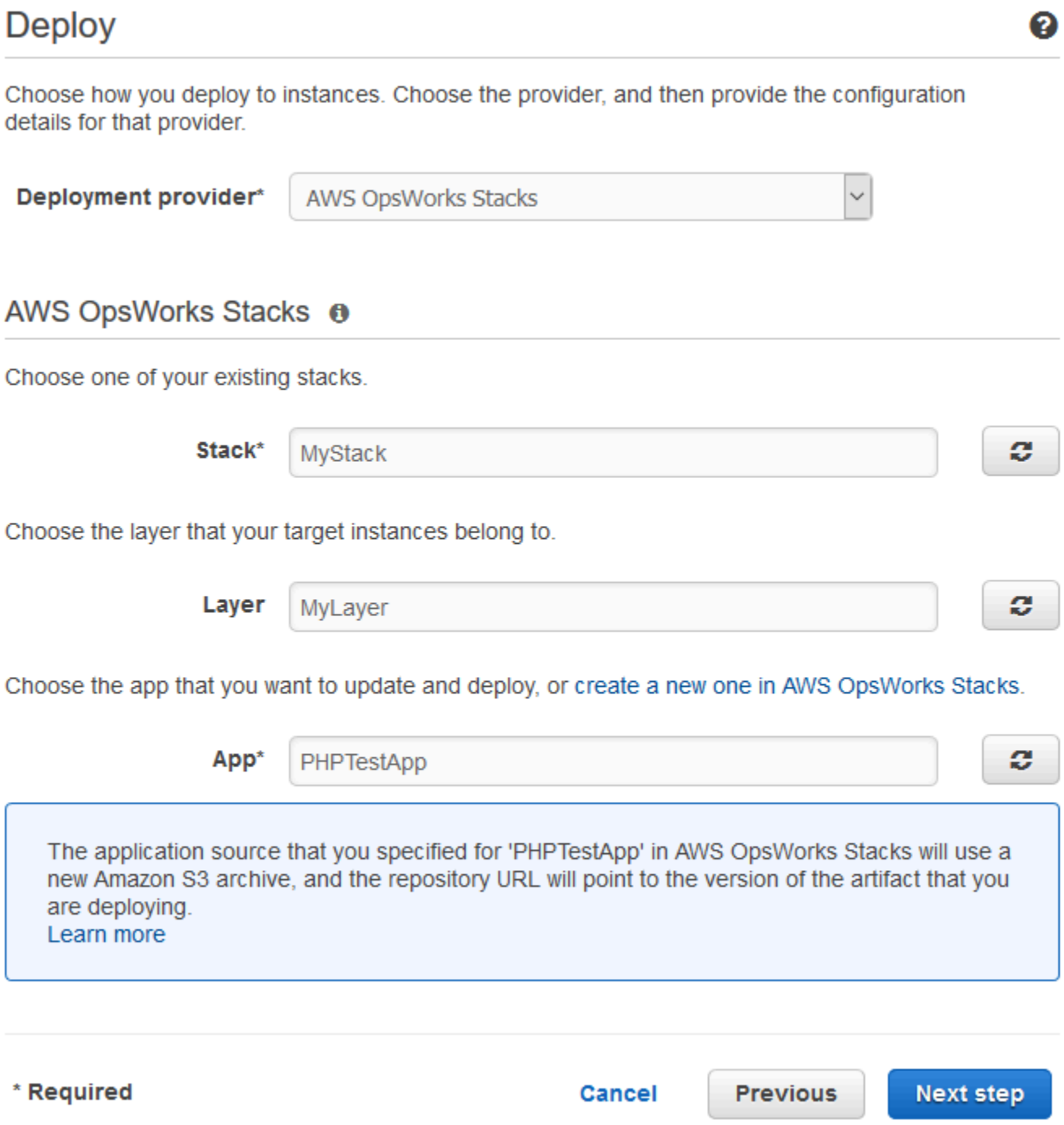
계층이 있는 스택과 AWS OpsWorks Stacks에 구성된 인스턴스가 하나 이상 있으면 CodePipeline에서 AWS OpsWorks Stacks를 공급자로 사용하여 파이프라인을 생성하여 Stacks 리소스에 앱 또는 Chef AWS OpsWorks 쿡북을 배포합니다.

파이프라인을 만들려면

1. https://console.aws.amazon.com/codepipeline/에서 CodePipeline 콘솔을 엽니다.
2. [파이프라인 생성]을 선택합니다.
3. CodePipeline 시작하기 페이지에서 **`MyOpsWorksPipeline`** 또는 사용자 계정에 고유한 다른 파이프라인 이름을 입력하고 다음 단계를 선택합니다.
4. 소스 위치 페이지의 소스 공급자 드롭다운 목록에서 Amazon S3를 선택합니다.
5. Amazon S3 세부 정보 영역에서 Amazon S3 버킷 경로를 **`s3://`*`bucket-name`*`/`*`file name`** 형식으로 입력합니다. 2단계: Amazon S3 버킷에 앱 코드 업로드 단원의 10단계에서 적어 둔 링크를 참조하세요. 이 연습에서는 경로가 `s3://my-appbucket/opsworks-demo-php-simple-app-version1.zip`입니다. 다음 단계를 선택합니다.



6. [빌드] 페이지의 드롭다운 목록에서 [빌드 없음]를 선택한 다음 [다음 단계]를 선택합니다.
7. 배포 페이지에서 배포 공급자로 AWS OpsWorks Stacks를 선택합니다.



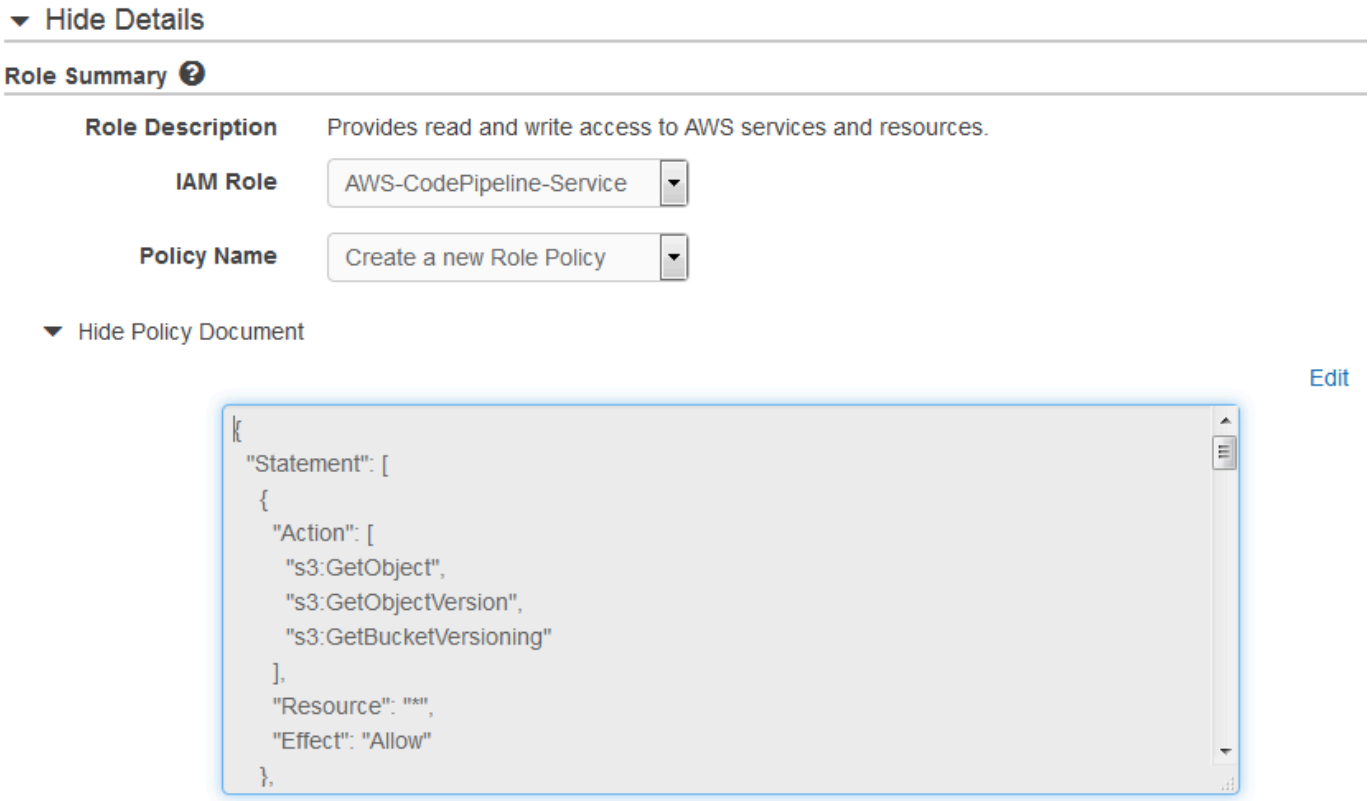
8. [스택] 필드에 `MyStack` 또는 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성합니다. 단원에서 생성한 스택의 이름을 입력합니다.
9. [계층] 필드에 `MyLayer` 또는 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성합니다. 단원에서 생성한 계층의 이름을 입력합니다.

10. 앱 필드에서 2단계: Amazon S3 버킷에 앱 코드 업로드의 Amazon S3에 업로드한 앱을 선택한 후 다음 단계를 선택합니다.
11. [AWS 서비스 역할] 페이지에서 [역할 생성]을 선택합니다.

    자동으로 생성될 역할인 AWS-CodePipeline-Service을(를) 설명하는 IAM 콘솔 페이지가 새 창에서 열립니다. [정책 이름] 드롭다운 목록에서 [새 정책 만들기]를 선택합니다. 정책 문서에 다음 내용이 포함되어 있는지 확인합니다. 필요한 경우 [편집]을 선택하고 정책 문서를 변경합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    정책 문서의 변경을 마친 뒤 허용을 선택합니다. IAM 콘솔에 변경 내용이 표시됩니다.



> **Note**
>
> 역할 생성에 실패하는 경우 AWS-CodePipeline-Service라는 IAM 역할이 이미 있기 때문일 수 있습니다. 2016년 5월 이전에 AWS-CodePipeline-Service 역할을 사용한 경우 역할에 AWS OpsWorks Stacks를 배포 공급자로 사용할 권한이 없을 수 있습니다.이 경우이 단계에 표시된 대로 정책 설명을 업데이트해야 합니다. 오류 메시지가 표시되면 이 단계의 시작 부분으로 돌아가서 역할 생성 대신 기존 역할 사용을 선택하세요. 기존 역할을 사용하는 경우에는 해당 역할에 이 단계에 표시된 권한이 포함된 정책이 연결되어 있어야 합니다. 서비스 역할 및 해당 정책 설명에 대한 자세한 정보는 IAM 서비스 역할에 대한 정책 편집 단원을 참조하세요.

12. 역할 생성 프로세스에 성공하면 IAM 페이지가 닫히고 AWS 서비스 역할 페이지로 돌아갑니다. 다음 단계를 선택합니다.
13. [검토 your pipeline] 페이지에서 표시된 선택 사항을 확인한 다음 [파이프라인 생성]을 선택합니다.

We will create your pipeline with the following resources.

Source Stage

Source provider Amazon S3

Amazon S3 location s3://my-appbucket0/opsworks-demo-php-simple-app-version1.zip

Build Stage

Build provider No Build

Beta Stage

Deployment provider AWS OpsWorks

Stack MyStack

App PHPTestApp

Layer MyLayer

Pipeline settings

Pipeline name MyOpsWorksPipeline

Artifact location s3://codepipeline-us-east-
AWS CodePipeline will use this existing S3 bucket to store artifacts for this pipeline. Depending on the size of your artifacts, you might be charged for storage costs. For more information, see Amazon S3 storage pricing.

Role name AWS-CodePipeline-Service

To save this configuration with these resources, choose Create pipeline.

Would you like to create this pipeline?

Cancel Previous Create pipeline

14. 파이프라인이 준비되면 파이프라인은 자동으로 소스 코드를 찾고 스택에 앱을 배포하기 시작할 것입니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.

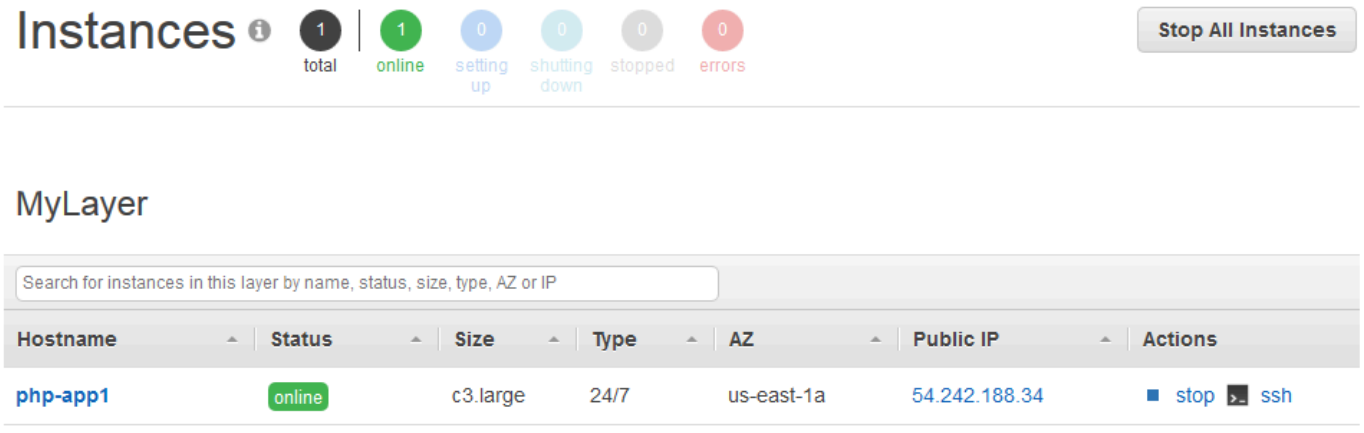
5단계: AWS OpsWorks Stacks에서 앱 배포 확인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

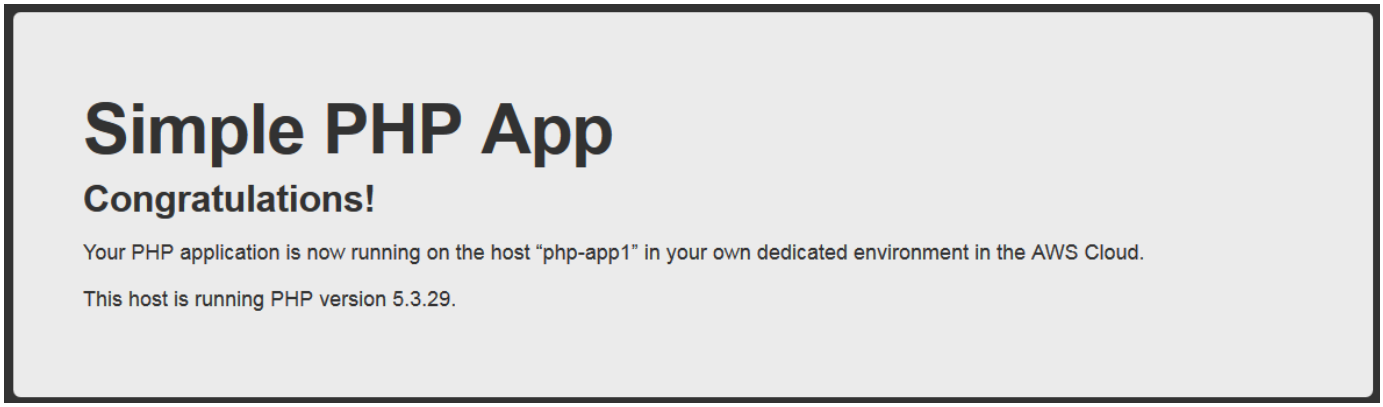
CodePipeline이 스택에 PHP 앱을 배포했는지 확인하려면 1단계: AWS OpsWorks Stacks에서 스택, 계층 및 인스턴스를 생성합니다. 섹션에서 생성한 인스턴스에 로그인합니다. PHP 웹 앱을 보고 사용할 수 있어야 합니다.

AWS OpsWorks Stacks 인스턴스에서 앱 배포를 확인하려면

1. https://console.aws.amazon.com/opsworks/ AWS OpsWorks 콘솔을 엽니다.
2. AWS OpsWorks Stacks 대시보드에서 MyStack을 선택한 다음 MyLayer를 선택합니다.
3. 탐색 창에서 [인스턴스]를 선택한 다음 웹 앱을 확인하기 위해 생성한 인스턴스의 퍼블릭 IP 주소를 선택합니다.



새 웹 브라우저 탭에 앱이 표시됩니다.



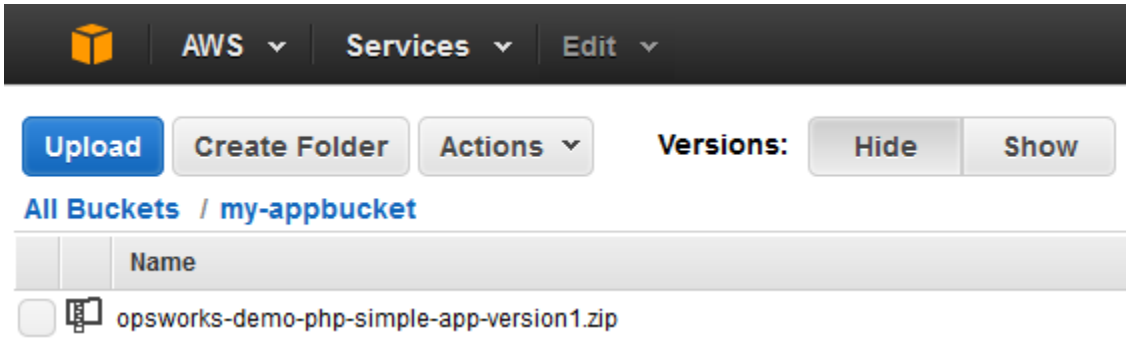
6단계(선택 사항): 앱 코드를 업데이트하여 CodePipeline이 자동으로 앱을 재배포하는지 확인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

CodePipeline을 사용하여 배포한 앱 또는 쿡북에서 코드를 변경하면 업데이트된 아티팩트가 CodePipeline에서 대상 인스턴스(이 경우 대상 AWS OpsWorks Stacks 스택)에 자동으로 배포됩니다. 이 섹션에서는 샘플 PHP 앱에서 코드를 업데이트할 때의 자동 배포를 보여줍니다.

샘플 앱에서 코드를 편집하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 샘플 PHP 앱을 저장하는 버킷을 엽니다.



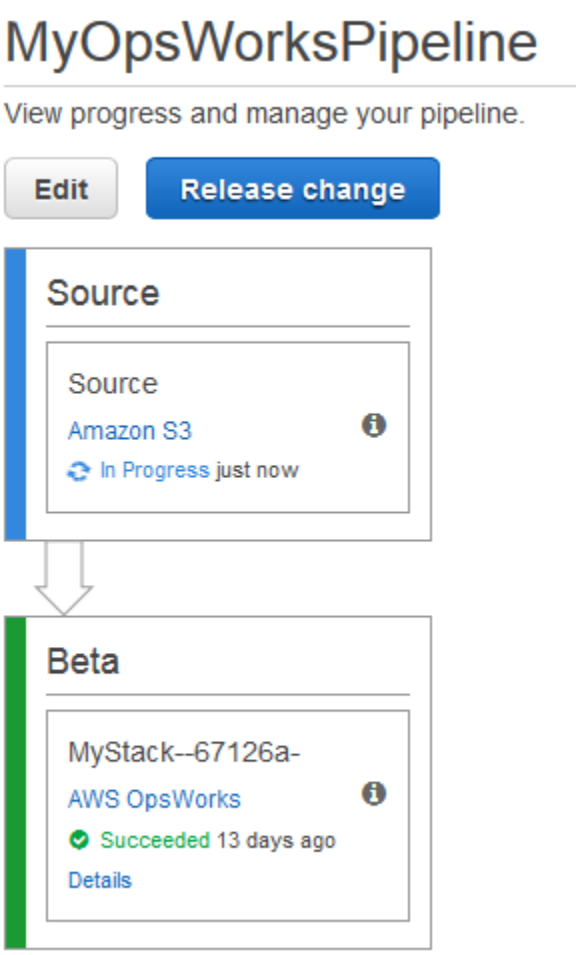
3. 앱이 포함된 ZIP 파일을 선택합니다. [작업] 메뉴에서 [다운로드]를 선택합니다.

4. 대화 상자에서 마우스 오른쪽 버튼을 클릭해 컨텍스트 메뉴를 연 다음 [다운로드]를 선택하고 ZIP 파일을 원하는 위치에 저장합니다. 확인을 선택합니다.
5. ZIP 파일의 내용을 원하는 위치로 추출합니다. 편집을 허용하려면 추출한 폴더, 하위 폴더 및 내용에 대한 권한을 변경해야 할 수 있습니다. `opsworks-demo-php-simple-app-version1` 폴더에서 편집할 `index.php` 파일을 엽니다.
6. `Your PHP application is now running`이라는 문구를 검색합니다. 텍스트 `Your PHP application is now running`을 `You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,`으로 바꿉니다. 변수는 편집하지 마십시오.

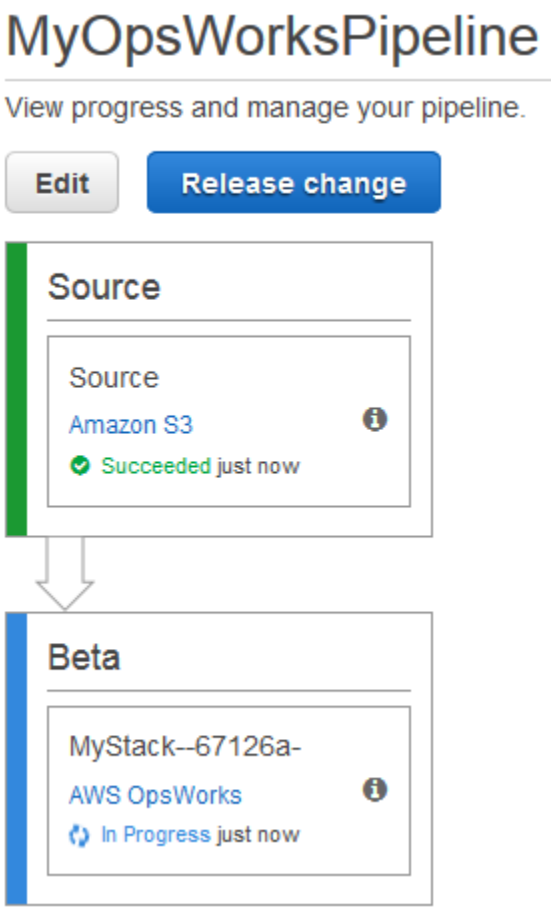
```
<body>
    <div class="container">
        <div class="hero-unit">
            <h1>Simple PHP App</h1>
            <h2>Congratulations!</h2>
            <p>You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,</p>
            <p>on the host &ldquo;<?php echo gethostname(); ?>&rdquo; </p>
            <p>in your own dedicated environment in the AWS Cloud.</p>
            <p>This host is running PHP version <?php echo phpversion(); ?>.</p>
        </div>
    </div>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>
</body>
```

7. `index.php` 파일을 저장하고 닫습니다.
8. `opsworks-demo-php-simple-app-version1` 폴더를 압축하여 ZIP 파일을 원하는 위치에 저장합니다. ZIP 파일의 이름은 변경하지 마십시오.
9. Amazon S3 버킷에 새로운 ZIP 파일을 업로드합니다. 이 연습에서 버킷의 이름은 `my-appbucket`입니다.
10. CodePipeline 콘솔을 열고 AWS OpsWorks Stacks 파이프라인(MyOpsWorksPipeline)을 엽니다. [릴리스 변경]을 선택합니다.

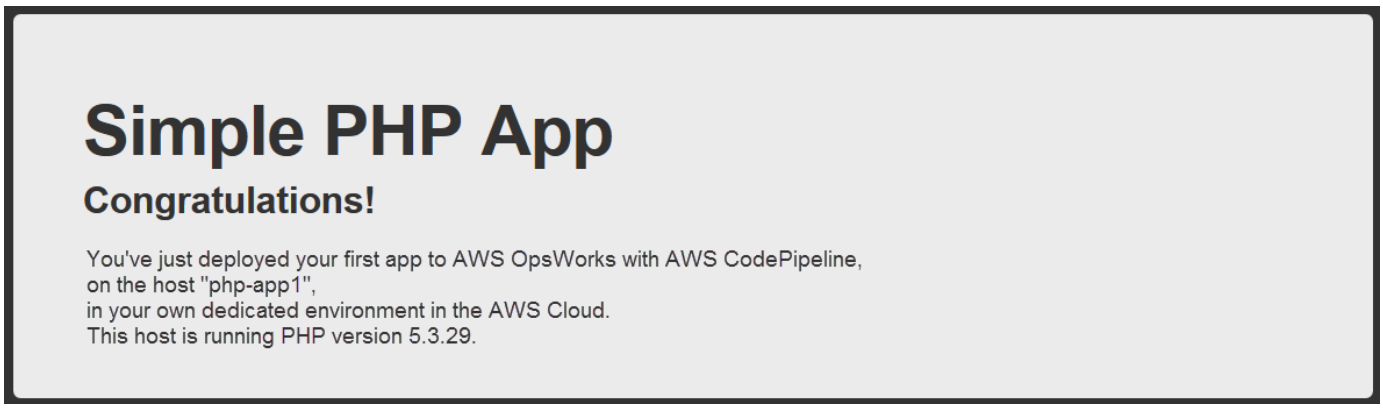
    (CodePipeline이 Amazon S3 버킷의 업데이트된 앱 버전에서 코드 변경을 감지할 때까지 기다릴 수 있습니다. 시간을 절약하기 위해 이 연습에서는 릴리스 변경을 선택하기만 하면 됩니다.)
11. 파이프라인의 단계를 따라 CodePipeline이 실행되는 것을 지켜봅니다. 먼저, CodePipeline이 소스 아티팩트에 대한 변경 사항을 찾습니다.



CodePipeline은 업데이트된 코드를 AWS OpsWorks Stacks의 스택으로 푸시합니다.



12. 파이프라인의 두 단계가 성공적으로 완료되면 AWS OpsWorks Stacks에서 스택(MyStack)을 엽니다.
13. [MyStack] 속성 페이지에서 [인스턴스]를 선택합니다.
14. 퍼블릭 IP 열에서 인스턴스의 퍼블릭 IP 주소를 선택하여 업데이트된 앱의 텍스트를 확인합니다.



7단계(선택 사항): 리소스 정리

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 수명이 종료되었으며 신규 및 기존 고객 모두에서 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS 계정에 원치 않는 요금이 부과되지 않도록이 연습에 사용한 AWS 리소스를 삭제할 수 있습니다. 이러한 AWS 리소스에는 AWS OpsWorks Stacks 스택, IAM 역할 및 인스턴스 프로파일, CodePipeline에서 생성한 파이프라인이 포함됩니다. 그러나 AWS OpsWorks Stacks 및 CodePipeline에 대해 계속 자세히 알아볼 때 이러한 AWS 리소스를 계속 사용하고 싶을 수 있습니다. 이러한 리소스를 유지하려면 이 연습을 마쳐야 합니다.

스택에서 앱을 삭제하려면

AWS CloudFormation 템플릿의 일부로 앱을 생성하거나 적용하지 않았으므로 스택을 삭제하기 전에 PHP 테스트 앱을 삭제합니다 AWS CloudFormation.

1. AWS OpsWorks Stacks 콘솔의 서비스 탐색 창에서 앱을 선택합니다.
2. [앱] 페이지에서 [PHPTestApp]을 선택한 다음 [작업]에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 삭제를 선택합니다. AWS OpsWorks 스택이 앱을 삭제합니다.

스택을 삭제하려면

AWS CloudFormation 템플릿을 실행하여 스택을 생성했으므로 AWS CloudFormation 콘솔에서 템플릿이 생성한 계층, 인스턴스, 인스턴스 프로파일 및 보안 그룹을 포함하여 스택을 삭제할 수 있습니다.

1. AWS CloudFormation 콘솔을 엽니다.
2. AWS CloudFormation 콘솔 대시보드에서 생성한 스택(MyStack)을 선택합니다. 작업 메뉴에서 스택 삭제를 선택합니다. 확인 메시지가 표시되면 예, 삭제를 선택합니다.
3. 스택의 상태 열에 DELETE_COMPLETE가 표시될 때까지 기다리세요.

파이프라인을 삭제하려면

1. CodePipeline 콘솔을 엽니다.

2. CodePipeline 대시보드에서 이 연습을 위해 생성한 파이프라인을 선택합니다.
3. 파이프라인 페이지에서 편집을 선택합니다.
4. [편집] 페이지에서 [삭제]를 선택합니다. 확인 메시지가 표시되면 [삭제]를 선택합니다.

AWS OpsWorks Stacks CLI 사용

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 명령줄 인터페이스(CLI)는 콘솔과 동일한 기능을 제공하며 다양한 작업에 사용할 수 있습니다. AWS OpsWorks Stacks CLI는의 일부입니다 AWS CLI. 설치 및 구성 방법을 비롯한 자세한 내용은 AWS 명령줄 인터페이스란 무엇입니까?를 AWS CLI참조하세요. 각 명령에 대한 전체 설명은 AWS OpsWorks Stacks reference를 참조하세요.

> **Note**
>
> Windows 기반 워크스테이션을 사용하는 경우 AWS Tools for Windows PowerShell을 실행하여 명령줄에서 AWS OpsWorks Stacks 작업을 수행할 수도 있습니다. 자세한 내용은 AWS Tools for Windows PowerShell을 참조하세요.

AWS OpsWorks Stacks 명령의 일반적인 형식은 다음과 같습니다.

```
aws opsworks --region us-west-1 opsworks command-name [--argument1 value] [...]
```

인수 값이 JSON 객체일 경우 " 문자를 이스케이프해야 하며, 그렇지 않을 경우 명령이 잘못된 JSON이라는 오류를 반환할 수 있습니다. 예를 들어 JSON 객체가 `"{"somekey":"somevalue"}"`일 경우 `"{\"somekey\":\"somevalue\"}"` 형식을 사용해야 합니다. 대안적 방법은 JSON 객체를 파일에 배치하고 `file://`을 사용하여 명령줄에 포함시키는 것입니다. 다음 예제는 appsource.json에 저장된 애플리케이션 소스 객체를 사용하여 앱을 생성합니다.

```
aws opsworks --region us-west-1 create-app --stack-id 8c428b08-a1a1-46ce-a5f8-
feddc43771b8 --name SimpleJSP --type java --app-source file://appsource.json
```

대부분의 명령은 하나 이상의 값을 하나의 JSON 객체로 패키징하여 반환합니다. 다음 섹션에는 몇 가지 예제가 포함되어 있습니다. 각 명령에서 반환되는 값에 대한 자세한 설명은 AWS OpsWorks Stacks reference를 참조하세요.

> **Note**
>
> AWS CLI 명령은 예제와 같이 리전을 지정해야 합니다. 다음 테이블에 --region 파라미터에 유효한 값이 나와 있습니다. AWS OpsWorks Stacks 명령 문자열을 단순화하려면 `--region` 파라미터를 생략할 수 있도록 기본 리전을 지정하도록 CLI를 구성합니다. 일반적으로 여러 리전 엔드포인트에서 작업하는 경우 기본 리전 엔드포인트를 사용하도록 AWS CLI 를 구성하지 마십시오. 캐나다(중부) 리전 엔드포인트는 API에서만 사용할 수 있으며 AWS CLI 에서 생성한 스택에는 사용할 수 없습니다 AWS Management Console. 자세한 내용은 AWS 리전 구성을 참조하세요.

리전 이름	명령 코드
US East (Ohio) Region	us-east-2
미국 동부(버지니아 북부) 리전	us-east-1
US West (N. California) Region	us-west-1
미국 서부(오레곤) 리전	us-west-2
캐나다(중부) 리전	ca-central-1
Europe (Ireland) Region	eu-west-1
Europe (London) Region	eu-west-2
Europe (Paris) Region	eu-west-3
Europe (Frankfurt) Region	eu-central-1
아시아 태평양(도쿄) 리전	ap-northeast-1

리전 이름	명령 코드
아시아 태평양(서울) 리전	ap-northeast-2
아시아 태평양(뭄바이) 리전	ap-south-1
아시아 태평양(싱가포르) 리전	ap-southeast-1
아시아 태평양(시드니) 리전	ap-southeast-2
South America (São Paulo) Region	sa-east-1

CLI 명령을 사용하려면 해당 권한이 있어야 합니다. AWS OpsWorks Stacks 권한에 대한 자세한 정보는 사용자 권한 관리 단원을 참조하세요. 특정 명령에 필요한 권한을 결정하려면 AWS OpsWorks Stacks reference에서 명령의 참조 페이지를 참조하세요.

다음 섹션에서는 AWS OpsWorks Stacks CLI를 사용하여 다양한 공통 작업을 수행하는 방법을 설명합니다.

인스턴스 생성(create-instance)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

지정된 스택에서 인스턴스를 생성하려면 create-instance 명령을 사용합니다.

주제

- 기본 호스트 이름을 사용하여 인스턴스 생성
- 테마 호스트 이름을 사용하여 인스턴스 생성
- 사용자 지정 AMI를 사용하여 인스턴스 생성

기본 호스트 이름을 사용하여 인스턴스 생성

```
C:\>aws opsworks --region us-west-1 create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux"
```

인수는 다음과 같습니다:

- `stack-id` – 콘솔에서 스택의 설정 페이지로부터(OpsWorks ID를 찾음) 또는 describe-stacks를 호출하여 스택 ID를 가져올 수 있습니다.
- `layer-ids` – 콘솔에서 계층 세부 정보 페이지로부터(OpsWorks ID를 찾음) 또는 describe-layers를 호출하여 계층 ID를 가져올 수 있습니다. 이 예제에서는 인스턴스가 한 계층에만 속합니다.
- `instance-type` - 메모리, CPU, 스토리지 용량 및 인스턴스 시간당 비용을 정의하는 사양으로, 이 예제에서는 `m1.large`입니다.
- `os` - 인스턴스의 운영 체제로, 이 예제에서는 Amazon Linux입니다.

이 명령은 다음과 같이 인스턴스 ID를 포함하는 JSON 객체를 반환합니다.

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

이 예제는 기본 호스트 이름(정수)을 사용하여 인스턴스를 생성합니다. 다음 섹션에서는 테마에서 생성된 호스트 이름을 사용하여 인스턴스를 생성하는 방법을 설명합니다.

테마 호스트 이름을 사용하여 인스턴스 생성

테마 호스트 이름을 사용하여 인스턴스를 생성할 수도 있습니다. 스택을 생성할 때 테마를 지정합니다. 자세한 내용은 새 스택 생성 단원을 참조하세요. 인스턴스를 생성하려면 먼저 get-hostname-suggestion을 호출하여 이름을 생성합니다. 예시:

```
C:\>aws opsworks get-hostname-suggestion --region us-west-1 --layer-id 5c8c272a-
f2d5-42e3-8245-5bf3927cb65b
```

기본 Layer Dependent 테마를 지정하면 get-hostname-suggestion이 계층의 짧은 이름에 숫자 하나만 추가합니다. 자세한 내용은 새 스택 생성 단원을 참조하십시오.

이 명령은 생성된 호스트 이름을 반환합니다.

```
{
    "Hostname": "php-app2",
    "LayerId": "5c8c272a-f2d5-42e3-8245-5bf3927cb65b"
}
```

다음과 같이 hostname 인수를 사용하여 생성된 이름을 create-instance에 전달할 수 있습니다.

```
c:\>aws --region us-west-1 opsworks create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux" --hostname "php-app2"
```

사용자 지정 AMI를 사용하여 인스턴스 생성

다음 create-instance 명령은 사용자 지정 AMI를 사용하여 인스턴스를 생성합니다(스택과 동일한 리전이어야 함). AWS OpsWorks Stacks용 사용자 지정 AMI를 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요사용자 지정 AMI 사용.

```
C:\>aws opsworks create-instance --region us-west-1 --stack-id c5ef46ce-3ccd-472c-
a3de-9bec94c6028e
   --layer-ids 6ff8a2ac-c9cc-49cf-9c67-fc852539ade4 --instance-type c3.large --os
 Custom
   --ami-id ami-6c61f104
```

인수는 다음과 같습니다:

- stack-id – 콘솔에서 스택의 설정 페이지로부터(OpsWorks ID를 찾음) 또는 describe-stacks를 호출하여 스택 ID를 가져올 수 있습니다.
- layer-ids – 콘솔에서 계층 세부 정보 페이지로부터(OpsWorks ID를 찾음) 또는 describe-layers를 호출하여 계층 ID를 가져올 수 있습니다. 이 예제에서는 인스턴스가 한 계층에만 속합니다.
- instance-type - 이 값은 인스턴스의 메모리, CPU, 스토리지 용량 및 시간당 비용을 정의하며 AMI와 호환되어야 합니다(이 예제에서는 c3.large).
- os - 인스턴스의 운영 체제. 사용자 지정 AMI에서는 Custom으로 설정되어야 합니다.

- `ami-id` - AMI ID로, `ami-6c61f104`와 비슷해야 합니다.

> **Note**
>
> 사용자 지정 AMI를 사용하는 경우 블록 디바이스 매핑은 지원되지 않으며 `--block-device-mappings` 옵션에 지정하는 값이 무시됩니다.

이 명령은 다음과 같이 인스턴스 ID를 포함하는 JSON 객체를 반환합니다.

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

앱 배포(create-deployment)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

지정된 스택에 앱을 배포하려면 create-deployment 명령을 사용합니다.

주제

- 앱 배포

앱 배포

```
aws opsworks --region us-west-1 create-deployment --stack-id cfb7e082-ad1d-4599-8e81-
de1c39ab45bf
   --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb --command "{\"Name\":\"deploy\"}"
```

인수는 다음과 같습니다:

- `stack-id` - 콘솔에서 스택의 설정 페이지로부터(OpsWorks ID를 찾음) 또는 `describe-stacks`를 호출하여 스택 ID를 가져올 수 있습니다.
- `app-id` – 콘솔에서 앱 세부 정보 페이지로부터(OpsWorks ID를 찾음) 또는 describe-apps를 호출하여 앱 ID를 가져올 수 있습니다.
- `command` - 인수는 명령 이름을 지정된 앱을 스택에 배포하는 `deploy`로 설정하는 JSON 객체를 취합니다.

  JSON 객체의 " 문자는 모두 이스케이프됩니다. 그렇지 않으면 명령이 잘못된 JSON이라는 오류를 반환할 수 있습니다.

이 명령은 다음과 같이 배포 ID를 포함하는 JSON 객체를 반환합니다.

```
{
    "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"
}
```

> **Note**
>
> 이전 예제는 스택 내 모든 인스턴스로 배포합니다. 지정된 인스턴스 하위 집합으로 배포하려면 `instance-ids` 인수를 추가하고 인스턴스 ID를 나열합니다.

스택의 앱 나열(describe-apps)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 앱을 나열하거나 지정된 앱에 대한 세부 정보를 가져오려면 describe-apps 명령을 사용합니다.

```
aws opsworks --region us-west-1 describe-apps --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

이전 예제는 각 앱에 대한 정보가 포함된 JSON 객체를 반환합니다. 이 예제는 앱이 하나뿐입니다. 각 파라미터에 대한 설명은 describe-apps를 참조하세요.


```
{
  "Apps": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "AppSource": {
        "Url": "url",
        "Type": "archive"
      },
      "Name": "SimpleJSP",
      "EnableSsl": false,
      "SslConfiguration": {},
      "AppId": "da1decc1-0dff-43ea-ad7c-bb667cd87c8b",
      "Attributes": {
        "RailsEnv": null,
        "AutoBundleOnDeploy": "true",
        "DocumentRoot": "ROOT"
      },
      "Shortname": "simplejsp",
      "Type": "other",
      "CreatedAt": "2013-08-01T21:46:54+00:00"
    }
  ]
}
```


스택의 명령 나열(describe-commands)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 명령을 나열하거나 지정된 명령에 대한 세부 정보를 가져오려면 describe-commands 명령을 사용합니다. 다음 예제는 지정된 인스턴스에서 실행된 명령에 대한 정보를 가져옵니다.

```
aws opsworks --region us-west-1 describe-commands --instance-id
 8c2673b9-3fe5-420d-9cfa-78d875ee7687
```

이 명령은 각 명령에 대한 세부 정보가 포함된 JSON 객체를 반환합니다. Type 파라미터는 명령 이름을 식별합니다(이 예제에서는 deploy 또는 undeploy). 다른 파라미터에 대한 설명은 describe-commands를 참조하세요.

```
{
  "Commands": [
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:47+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "AcknowledgedAt": "2013-07-25T18:57:41+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/008c1a91-
ec59-4d51-971d-3adff54b00cc?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=HkXil6UuNfxTCC37EPQAa462E1E%3D&response-cache-
control=private&response-content-encoding=gzip&response-content- type=text%2Fplain",
      "Type": "undeploy",
      "CommandId": "008c1a91-ec59-4d51-971d-3adff54b00cc",
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "ExitCode": 0
    },
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:55:40+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
      "AcknowledgedAt": "2013-07-25T18:55:32+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/
logs/899d3d64-0384-47b6-a586-33433aad117c?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=xMsJvtLuUqWmsr8s%2FAjVru0BtRs%3D&response-cache-
control=private&response-content-encoding=gzip&response-conten t-type=text%2Fplain",
      "Type": "deploy",
      "CommandId": "899d3d64-0384-47b6-a586-33433aad117c",
      "CreatedAt": "2013-07-25T18:55:29+00:00",
      "ExitCode": 0
    }
  ]
}
```

스택의 배포 나열(describe-deployments)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 배포를 나열하거나 지정된 배포에 대한 세부 정보를 가져오려면 describe-deployments 명령을 사용합니다.

```
aws opsworks --region us-west-1 describe-deployments --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

이전 명령은 지정된 스택의 각 배포에 대한 세부 정보가 포함된 JSON 객체를 반환합니다. 각 파라미터에 대한 설명은 describe-deployments를 참조하세요.

```
{
  "Deployments": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Status": "successful",
          "CompletedAt": "2013-07-25T18:57:49+00:00",
          "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
          "Command": {
              "Args": {},
              "Name": "undeploy"
          },
          "CreatedAt": "2013-07-25T18:57:34+00:00",
          "Duration": 15,
          "InstanceIds": [
              "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
              "9e588a25-35b2-4804-bd43-488f85ebe5b7"
          ]
      },
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Status": "successful",
          "CompletedAt": "2013-07-25T18:56:41+00:00",
```

```
            "IamUserArn": "arn:aws:iam::444455556666:user/example-user",
            "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
            "Command": {
                "Args": {},
                "Name": "deploy"
            },
            "InstanceIds": [
                "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
                "9e588a25-35b2-4804-bd43-488f85ebe5b7"
            ],
            "Duration": 72,
            "CreatedAt": "2013-07-25T18:55:29+00:00"
        }
    ]
}
```


스택의 탄력적 IP 주소 나열(describe-elastic-ips)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택에 등록된 탄력적 IP 주소를 나열하거나 지정된 탄력적 IP 주소에 대한 세부 정보를 가져오려면 describe-elastic-ips 명령을 사용합니다.

```
aws opsworks --region us-west-2 describe-elastic-ips --instance-id b62f3e04-e9eb-436c-a91f-d9e9a396b7b0
```

이전 명령은 지정된 인스턴스의 각 탄력적 IP 주소(이 예제에서는 1개)에 대한 세부 정보가 포함된 JSON객체를 반환합니다. 각 파라미터에 대한 설명은 describe-elastic-ips를 참조하세요.


```
{
  "ElasticIps": [
      {
          "Ip": "192.0.2.0",
          "Domain": "standard",
          "Region": "us-west-2"
```

```
      }
  ]
}
```

스택의 인스턴스 나열(describe-instances)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 인스턴스를 나열하거나 지정된 인스턴스에 대한 세부 정보를 가져오려면 describe-instances 명령을 사용합니다.

```
C:\>aws opsworks --region us-west-2 describe-instances --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a
```

이전 명령은 지정된 스택의 모든 인스턴스에 대한 세부 정보가 포함된 JSON 객체를 반환합니다. 각 파라미터에 대한 설명은 describe-instances를 참조하세요.

```
{
  "Instances": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "SshHostRsaKeyFingerprint":
 "f4:3b:8e:27:1b:73:98:80:5d:d7:33:e2:b8:c8:8f:de",
          "Status": "stopped",
          "AvailabilityZone": "us-west-2a",
          "SshHostDsaKeyFingerprint":
 "e8:9b:c7:02:18:2a:bd:ab:45:89:21:4e:af:0b:07:ac",
          "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
          "Os": "Amazon Linux",
          "Hostname": "db-master1",
          "SecurityGroupIds": [],
          "Architecture": "x86_64",
          "RootDeviceType": "instance-store",
          "LayerIds": [
              "41a20847-d594-4325-8447-171821916b73"
```

```
            ],
            "InstanceType": "c1.medium",
            "CreatedAt": "2013-07-25T18:11:27+00:00"
        },
        {
            "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
            "SshHostRsaKeyFingerprint":
 "ae:3a:85:54:66:f3:ce:98:d9:83:39:1e:10:a9:38:12",
            "Status": "stopped",
            "AvailabilityZone": "us-west-2a",
            "SshHostDsaKeyFingerprint":
 "5b:b9:6f:5b:1c:ec:55:85:f3:45:f1:28:25:1f:de:e4",
            "InstanceId": "9e588a25-35b2-4804-bd43-488f85ebe5b7",
            "Os": "Amazon Linux",
            "Hostname": "tomcustom1",
            "SecurityGroupIds": [],
            "Architecture": "x86_64",
            "RootDeviceType": "instance-store",
            "LayerIds": [
                "e6cbcd29-d223-40fc-8243-2eb213377440"
            ],
            "InstanceType": "c1.medium",
            "CreatedAt": "2013-07-25T18:15:52+00:00"
        }
    ]
}
```


계정의 스택 나열(describe-스택)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계정의 스택을 나열하거나 지정된 스택에 대한 세부 정보를 가져오려면 describe-stacks 명령을 사용합니다.

```
aws opsworks --region us-west-2 describe-stacks
```

이전 명령은 계정의 각 스택(이 예제에서는 2개)에 대한 세부 정보가 포함된 JSON 객체를 반환합니다. 각 파라미터에 대한 설명은 describe-stacks를 참조하세요.

```
{
    "Stacks": [
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "aeb7523e-7c8b-49d4-b866-03aae9d4fbcb",
            "DefaultRootDeviceType": "instance-store",
            "Name": "TomStack-sd",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
            },
            "UseCustomCookbooks": true,
            "CustomJson": "{\n  \"tomcat\": {\n    \"base_version\": 7,\n
 \"java_opts\": \"-Djava.awt.headless=true -Xmx256m\"\n  },\n  \
"datasources\": {\n    \"ROOT\": \"jdbc/mydb\"\n  }\n}",
            "Region": "us-west-2",
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {
                "Url": "git://github.com/example-repo/tomcustom.git",
                "Type": "git"
            },
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-08-01T22:53:42+00:00"
        },
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "40738975-da59-4c5b-9789-3e422f2cf099",
            "DefaultRootDeviceType": "instance-store",
            "Name": "MyStack",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
```

```
            },
            "UseCustomCookbooks": false,
            "Region": "us-west-2",
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {},
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-10-25T19:24:30+00:00"
        }
    ]
}
```

스택의 계층 나열(describe-layers)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 계층을 나열하거나 지정된 계층에 대한 세부 정보를 가져오려면 describe-layers 명령을 사용합니다.

```
aws opsworks --region us-west-2 describe-layers --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

위 명령은 지정된 스택의 각 계층에 대한 세부 정보가 포함된 JSON 객체(이 예시에서는 MySQL 계층 및 사용자 지정 계층)를 반환합니다. 각 파라미터에 대한 설명은 describe-layers를 참조하세요.

```
{
  "Layers": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Type": "db-master",
```

```
            "DefaultSecurityGroupNames": [
                "AWS-OpsWorks-DB-Master-Server"
            ],
            "Name": "MySQL",
            "Packages": [],
            "DefaultRecipes": {
                "Undeploy": [],
                "Setup": [
                    "opsworks_initial_setup",
                    "ssh_host_keys",
                    "ssh_users",
                    "mysql::client",
                    "dependencies",
                    "ebs",
                    "opsworks_ganglia::client",
                    "mysql::server",
                    "dependencies",
                    "deploy::mysql"
                ],
                "Configure": [
                    "opsworks_ganglia::configure-client",
                    "ssh_users",
                    "agent_version",
                    "deploy::mysql"
                ],
                "Shutdown": [
                    "opsworks_shutdown::default",
                    "mysql::stop"
                ],
                "Deploy": [
                    "deploy::default",
                    "deploy::mysql"
                ]
            },
            "CustomRecipes": {
                "Undeploy": [],
                "Setup": [],
                "Configure": [],
                "Shutdown": [],
                "Deploy": []
            },
            "EnableAutoHealing": false,
            "LayerId": "41a20847-d594-4325-8447-171821916b73",
            "Attributes": {
```

```
            "MysqlRootPasswordUbiquitous": "true",
            "RubygemsVersion": null,
            "RailsStack": null,
            "HaproxyHealthCheckMethod": null,
            "RubyVersion": null,
            "BundlerVersion": null,
            "HaproxyStatsPassword": null,
            "PassengerVersion": null,
            "MemcachedMemory": null,
            "EnableHaproxyStats": null,
            "ManageBundler": null,
            "NodejsVersion": null,
            "HaproxyHealthCheckUrl": null,
            "MysqlRootPassword": "*****FILTERED*****",
            "GangliaPassword": null,
            "GangliaUser": null,
            "HaproxyStatsUrl": null,
            "GangliaUrl": null,
            "HaproxyStatsUser": null
        },
        "Shortname": "db-master",
        "AutoAssignElasticIps": false,
        "CustomSecurityGroupIds": [],
        "CreatedAt": "2013-07-25T18:11:19+00:00",
        "VolumeConfigurations": [
            {
                "MountPoint": "/vol/mysql",
                "Size": 10,
                "NumberOfDisks": 1
            }
        ]
    },
    {
        "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
        "Type": "custom",
        "DefaultSecurityGroupNames": [
            "AWS-OpsWorks-Custom-Server"
        ],
        "Name": "TomCustom",
        "Packages": [],
        "DefaultRecipes": {
            "Undeploy": [],
            "Setup": [
                "opsworks_initial_setup",
```

```
                "ssh_host_keys",
                "ssh_users",
                "mysql::client",
                "dependencies",
                "ebs",
                "opsworks_ganglia::client"
            ],
            "Configure": [
                "opsworks_ganglia::configure-client",
                "ssh_users",
                "agent_version"
            ],
            "Shutdown": [
                "opsworks_shutdown::default"
            ],
            "Deploy": [
                "deploy::default"
            ]
        },
        "CustomRecipes": {
            "Undeploy": [],
            "Setup": [
                "tomcat::setup"
            ],
            "Configure": [
                "tomcat::configure"
            ],
            "Shutdown": [],
            "Deploy": [
                "tomcat::deploy"
            ]
        },
        "EnableAutoHealing": true,
        "LayerId": "e6cbcd29-d223-40fc-8243-2eb213377440",
        "Attributes": {
            "MysqlRootPasswordUbiquitous": null,
            "RubygemsVersion": null,
            "RailsStack": null,
            "HaproxyHealthCheckMethod": null,
            "RubyVersion": null,
            "BundlerVersion": null,
            "HaproxyStatsPassword": null,
            "PassengerVersion": null,
            "MemcachedMemory": null,
```

```
            "EnableHaproxyStats": null,
            "ManageBundler": null,
            "NodejsVersion": null,
            "HaproxyHealthCheckUrl": null,
            "MysqlRootPassword": null,
            "GangliaPassword": null,
            "GangliaUser": null,
            "HaproxyStatsUrl": null,
            "GangliaUrl": null,
            "HaproxyStatsUser": null
        },
        "Shortname": "tomcustom",
        "AutoAssignElasticIps": false,
        "CustomSecurityGroupIds": [],
        "CreatedAt": "2013-07-25T18:12:53+00:00",
        "VolumeConfigurations": []
    }
  ]
}
```


레시피 실행(create-deployment)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

create-deployment 명령을 사용하여 스택 명령 및 배포 명령을 실행합니다. 다음 예제는 스택 명령을 실행하여 지정된 스택에서 사용자 지정 레시피를 실행합니다.

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --command "{\"Name\":\"execute_recipes\", \"Args\":{\"recipes\":[\"phpapp::appsetup
\"]}}"
```

command 인수는 다음과 같은 형식의 JSON 객체를 취합니다.

- Name - 명령 이름을 지정합니다. 이 예제에 사용된 execute_recipes 명령은 스택의 인스턴스에서 지정된 레시피를 실행합니다.
- Args - 인수 및 해당 값의 목록을 지정합니다. 이 예제에는 인수가 recipes 1개이며, 이 인수는 실행될 레시피 phpapp::appsetup으로 설정되어 있습니다.

JSON 객체의 " 문자는 모두 이스케이프됩니다. 그렇지 않으면 명령이 잘못된 JSON이라는 오류를 반환할 수 있습니다.

이 명령은 배포 ID를 반환합니다. 이 ID를 사용하여 다른 CLI 명령에 대한 명령(예: describe-commands)을 식별하는 데 사용할 수 있습니다.

```
{
    "DeploymentId": "5cbaa7b9-4e09-4e53-aa1b-314fbd106038"
}
```

종속성 설치(create-deployment)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

create-deployment 명령을 사용하여 스택 명령 및 배포 명령을 실행합니다. 다음 예제는 update_dependencies 스택 명령을 실행하여 스택의 인스턴스에서 종속성을 업데이트합니다.

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
--command "{\"Name\":\"install_dependencies\"}"
```

command 인수는 값이 명령 이름(이 예제에서는 Name)을 지정하는 install_dependencies 파라미터가 포함된 JSON 객체를 취합니다. JSON 객체의 " 문자는 모두 이스케이프됩니다. 그렇지 않으면 명령이 잘못된 JSON이라는 오류를 반환할 수 있습니다.

이 명령은 배포 ID를 반환합니다. 이 ID를 사용하여 다른 CLI 명령에 대한 명령(예: describe-commands)을 식별하는 데 사용할 수 있습니다.

```
{
    "DeploymentId": "aef5b255-8604-4928-81b3-9b0187f962ff"
}
```

스택 구성 업데이트(update-stack)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

지정된 스택의 구성을 업데이트하려면 update-stack 명령을 사용합니다. 다음 예제는 스택을 업데이트하여 사용자 지정 JSON을 스택 구성 속성에 추가합니다.

```
aws opsworks --region us-west-1 update-stack --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --custom-json "{\"somekey\":\"somevalue\"}" --service-role-arn
 arn:aws:iam::444455556666:role/aws-opsworks-service-role
```

JSON 객체의 " 문자는 모두 이스케이프됩니다. 그렇지 않으면 명령이 잘못된 JSON이라는 오류를 반환할 수 있습니다.

> **Note**
>
> 이 예제는 스택에 대한 서비스 역할도 지정합니다. `service-role-arn`을 유효한 서비스 역할 ARN으로 설정해야 하며, 그렇지 않으면 작업이 실패합니다. 기본값은 없습니다. 원할 경우 스택의 현재 서비스 역할 ARN을 지정할 수 있지만, 명시적으로 지정해야 합니다.

`update-stack` 명령은 값을 반환하지 않습니다.

디버깅 및 문제 해결 안내서

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피를 디버깅하거나 서비스 문제를 해결해야 하는 경우, 일반적으로 가장 좋은 방법은 다음 단계를 순서대로 따르는 것입니다.

1. 구체적 문제를 일반적인 디버깅 및 문제 해결에서 확인합니다.
2. AWS OpsWorks Stacks 포럼을 검색하여 문제가 논의된 적이 있는지 확인합니다.

   포럼에는 경험이 풍부한 많은 사용자가 포함되며 AWS OpsWorks Stacks 팀에서 모니터링합니다.
3. 레시피 관련 문제는 레시피 디버깅 단원을 참조하세요.
4. AWS OpsWorks Stacks Support에 문의하거나 AWS OpsWorks Stacks 포럼에 문제를 게시합니다.

다음 섹션에서는 레시피 디버깅에 대해 안내합니다. 마지막 섹션에서는 일반적인 디버깅 및 문제 해결과 해법을 설명합니다.

> **Note**
>
> 각각의 Chef 실행은 실행에 대한 상세한 설명을 제공하고 중요한 문제 해결 리소스인 로그를 생성합니다. 로그의 세부 정보의 양을 지정하려면 사용자 지정 레시피에 원하는 로그 수준을 지정하는 `Chef::Log.level` 문을 추가합니다. 기본값은 `:info`입니다. 다음 예제는 Chef 로그 수준을 실행에 대한 가장 상세한 설명을 제공하는 `:debug`로 설정하는 방법을 보여 줍니다.
>
> ```
> Chef::Log.level = :debug
> ```
>
> Chef 로그를 보고 해석하는 방법에 대한 자세한 정보는 Chef 로그 단원을 참조하세요.

주제

- 레시피 디버깅

- 일반적인 디버깅 및 문제 해결

레시피 디버깅

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

수명 주기 이벤트가 발생하거나 레시피 실행 스택 명령이 사용되면 AWS OpsWorks Stacks가 에이전트로 명령을 전송해 지정된 인스턴스에서 해당 레시피를 실행하기 위한 Chef Solo 실행을 시작합니다. 이 섹션에서는 실패한 레시피를 디버깅할 수 있는 몇 가지 방법을 설명합니다.

주제

- 실패한 인스턴스에 로그인
- Chef 로그
- AWS OpsWorks Stacks 에이전트 CLI 사용

실패한 인스턴스에 로그인

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

레시피가 실패하면 인스턴스는 온라인 상태가 아니라 `setup_failed` 상태로 끝납니다. 인스턴스가 AWS OpsWorks Stacks에 관한 한 온라인 상태가 아니더라도 EC2 인스턴스가 실행 중이므로 로그인해 문제를 해결하는 것이 유용한 경우가 많습니다. 예를 들어 애플리케이션 또는 사용자 지정 쿡북이 올바로 설치되었는지 확인할 수 있습니다. SSH 및 RDP 로그인에 대한 AWS OpsWorks Stacks 기본

제공 지원은 온라인 상태의 인스턴스에서만 사용할 수 있습니다. 하지만 SSH 키 페어를 인스턴스에 할당했다면 다음과 같이 로그인이 가능합니다.

- Linux 인스턴스 - SSH 키 페어의 프라이빗 키를 사용하여 OpenSSH 또는 PuTTY와 같은 타사 SSH 클라이언트로 로그인합니다.

  이 목적을 위해 EC2 키 페어 또는 개인 SSH 키 페어를 사용할 수 있습니다.
- Windows 인스턴스 - EC2 키 페어의 프라이빗 키를 사용하여 인스턴스의 관리자 암호를 검색합니다.

  이 암호를 사용하여 원하는 RDP 클라이언트로 로그인합니다. 자세한 내용은 관리자로 로그인 단원을 참조하십시오. 개인 SSH 키 페어를 사용하여 관리자 암호를 검색할 수는 없습니다.

Chef 로그

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 로그는 특히 레시피 디버깅을 위한 주요 문제 해결 리소스 중 하나입니다. AWS OpsWorks Stacks는 각 명령에 대한 Chef 로그를 캡처하고 인스턴스의 최신 명령 30개에 대한 로그를 유지합니다. 실행이 디버깅 모드이므로 로그에는 `stdout` 및 `stderror`로 전송되는 텍스트를 비롯해 Chef 실행에 대한 자세한 설명이 포함됩니다. 레시피가 실패할 경우 로그에 Chef 스택 트레이스가 포함됩니다.

AWS OpsWorks Stacks는 Chef 로그를 볼 수 있는 여러 가지 방법을 제공합니다. 로그 정보를 입수하면 이 정보를 사용하여 실패한 레시피를 디버깅할 수 있습니다.

> **Note**
>
> 또한 SSH를 사용하여 인스턴스에 연결하고 CLI `show_log` 명령을 실행하여 지정된 로그의 끝부분을 볼 수도 있습니다. 자세한 내용은 Chef 로그 표시 단원을 참조하십시오.

주제

- 콘솔을 사용하여 Chef 로그 보기

- CLI 또는 API를 사용하여 Chef 로그 보기
- 인스턴스에서 Chef 로그 보기
- Chef 로그 해석
- 일반적인 Chef 로그 오류

콘솔을 사용하여 Chef 로그 보기

Chef 로그를 보는 가장 간단한 방법은 인스턴스의 세부 정보 페이지로 이동하는 것입니다. [로그] 섹션에는 각 이벤트 및 레시피 실행 명령에 대한 항목이 포함되어 있습니다. 다음은 인스턴스의 [로그] 섹션입니다. 구성 및 설정 수명 주기 이벤트에 해당하는 [구성] 및 [설정] 명령이 표시되어 있습니다.

해당 명령의 로그 열에서 표시를 클릭하면 해당 Chef 로그를 볼 수 있습니다. 오류가 발생하면 AWS OpsWorks Stacks는 일반적으로 파일 끝에 있는 오류에 대한 로그를 자동으로 엽니다.

CLI 또는 API를 사용하여 Chef 로그 보기

AWS OpsWorks Stacks CLI `describe-commands` 명령 또는 `DescribeCommands` API 작업을 사용하여 Amazon S3 버킷에 저장된 로그를 볼 수 있습니다. 다음은 CLI를 사용하여 지정된 인스턴스에 대한 현재 로그 파일 세트를 보는 방법을 보여줍니다. `DescribeCommands`를 사용하기 위한 절차는 기본적으로 비슷합니다.

AWS OpsWorks Stacks를 사용하여 인스턴스의 Chef 로그를 보려면

1. 인스턴스의 세부 정보 페이지를 열고 인스턴스의 OpsWorks ID 값을 복사해 둡니다.
2. 이 ID 값을 사용하여 다음과 같이 `describe-commands` CLI 명령을 실행합니다.

```
aws opsworks describe-commands --instance-id 67bf0da2-29ed-4217-990c-d895d51812b9
```

이 명령은 AWS OpsWorks Stacks가 인스턴스에서 실행한 각 명령에 대해 포함된 객체와 함께 JSON 객체를 반환하며, 가장 최근 객체가 먼저 반환됩니다. 이 예제의 `Type` 파라미터에는 각 포함 객체의 명령 유형, `configure` 명령 및 `setup` 명령이 포함되어 있습니다.

```
{
    "Commands": [
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:38:36+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
            "AcknowledgedAt": "2013-10-25T19:38:24+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/
b6c402df-5c23-45b2-a707-ad20b9c5ae40?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=YkqS5IZN2P4wixjHwoC3aCMbn5s%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "configure",
            "CommandId": "b6c402df-5c23-45b2-a707-ad20b9c5ae40",
            "CreatedAt": "2013-10-25T19:38:11+00:00",
            "ExitCode": 0
        },
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:31:08+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
            "AcknowledgedAt": "2013-10-25T19:29:01+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/2a90e862-
f974-42a6-9342-9a4f03468358?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=cxKYHO8mCCd4MvOyFb6ywebeQtA%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "setup",
            "CommandId": "2a90e862-f974-42a6-9342-9a4f03468358",
            "CreatedAt": "2013-10-25T19:26:01+00:00",
            "ExitCode": 0
        }
    ]
}
```

3. `LogUrl` 값을 브라우저에 복사해 로그를 확인합니다.

인스턴스에 많은 명령이 있을 경우 `describe-commands`에 파라미터를 추가하여 응답 객체에 포함된 명령을 필터링할 수 있습니다. 자세한 내용은 describe-commands를 참조하세요.

인스턴스에서 Chef 로그 보기

> **Note**
>
> 이 섹션의 각 항목은 Chef 12에 적용됩니다. Chef 11.10 및 이전 릴리스에 대한 Chef 로그의 위치에 대한 자세한 정보는 Linux용 Chef 11.10 및 이전 버전 문제 해결 단원을 참조하세요.

Linux 인스턴스

AWS OpsWorks Stacks는 각 인스턴스의 Chef 로그를 `/var/chef/runs` 디렉터리에 저장합니다. (Linux 인스턴스의 경우 이 디렉터리에 JSON 형식 파일로 저장되는 연결된 데이터 백도 들어 있습니다.) 이 디렉터리에 액세스하려면 sudo 권한이 필요합니다. 각 실행에 대한 로그는 개별 실행의 하위 디렉터리 안에 위치하는 `chef.log`라는 파일에 기록됩니다.

AWS OpsWorks Stacks는 내부 로그를 인스턴스의 `/var/log/aws/opsworks` 폴더에 저장합니다. 일반적으로 이 정보는 문제 해결 목적으로는 그다지 유용하지 않습니다. 그러나 이러한 로그는 AWS OpsWorks Stacks 지원에 유용하며, 서비스에 문제가 발생할 경우 해당 로그를 제공하도록 요청 받을 수 있습니다. Linux 로그가 때로는 유용한 문제 해결 데이터를 제공할 수도 있습니다.

Windows 인스턴스

에이전트 로그

Windows 인스턴스에서는 OpsWorks 로그가 다음과 같은 `ProgramData` 경로에 저장됩니다. 번호에 타임스탬프가 포함됩니다.

```
C:\ProgramData\OpsWorksAgent\var\logs\number
```

> **Note**
>
> 기본적으로 `ProgramData`는 숨겨진 폴더입니다. 폴더 숨김을 해제하려면 [폴더 옵션]으로 이동합니다. [보기]에서 숨겨진 파일을 표시하는 옵션을 선택합니다.

다음 예제는 Windows 인스턴스에 대한 에이전트 로그입니다.

```
Mode                LastWriteTime     Length Name
----                -------------     ------ ----
```

```
-a---         5/24/2015  11:59 PM     127277 command.20150524.txt
-a---         5/25/2015  11:59 PM     546772 command.20150525.txt
-a---         5/26/2015  11:59 PM     551514 command.20150526.txt
-a---         5/27/2015   9:43 PM     495181 command.20150527.txt
-a---         5/24/2015  11:59 PM      24353 keepalive.20150524.txt
-a---         5/25/2015  11:59 PM     106232 keepalive.20150525.txt
-a---         5/26/2015  11:59 PM     106208 keepalive.20150526.txt
-a---         5/27/2015   8:54 PM      92593 keepalive.20150527.txt
-a---         5/24/2015   7:19 PM       3891 service.20150524.txt
-a---         5/27/2015   8:54 PM       1493 service.20150527.txt
-a---         5/24/2015  11:59 PM     112549 wire.20150524.txt
-a---         5/25/2015  11:59 PM     501501 wire.20150525.txt
-a---         5/26/2015  11:59 PM     499640 wire.20150526.txt
-a---         5/27/2015   8:54 PM     436870 wire.20150527.txt
```

Chef 로그

Windows 인스턴스에서는 Chef 로그가 다음과 같은 `ProgramData` 경로에 저장됩니다. 번호에 타임스탬프가 포함됩니다.

```
C:\ProgramData\OpsWorksAgent\var\commands\number
```

> **Note**
>
> 이 디렉터리에는 첫 번째 (OpsWorks 소유) Chef 실행의 출력만 포함됩니다.

다음 예제는 Windows 인스턴스에 대한 OpsWorks 소유 Chef 로그입니다.

```
 Mode                 LastWriteTime            Name
 ----                 -------------            ----
 d----         5/24/2015   7:23 PM
 configure-7ecb5f47-7626-439b-877f-5e7cb40ab8be
 d----         5/26/2015   8:30 PM            configure-8e74223b-d15d-4372-aeea-
a87b428ffc2b
 d----         5/24/2015   6:34 PM            configure-
c3980a1c-3d08-46eb-9bae-63514cee194b
 d----         5/26/2015   8:32 PM            grant_remote_access-70dbf834-1bfa-4fce-
b195-e50e85402f4c
 d----         5/26/2015  10:30 PM            revoke_remote_access-1111fce9-843a-4b27-
b93f-ecc7c5e9e05b
```

```
 d----         5/24/2015   7:21 PM            setup-754ec063-8b60-4cd4-
b6d7-0e89d7b7aa78
 d----         5/26/2015   8:27 PM            setup-af5bed36-5afd-4115-
af35-5766f88bc039
 d----         5/24/2015   6:32 PM            setup-d8abeffa-24d4-414b-
bfb1-4ad07319f358
 d----         5/24/2015   7:13 PM            shutdown-c7130435-9b5c-4a95-
be17-6b988fc6cf9a
 d----         5/26/2015   8:25 PM
 sync_remote_users-64c79bdc-1f6f-4517-865b-23d2def4180c
 d----         5/26/2015   8:48 PM
 update_custom_cookbooks-2cc59a94-315b-414d-85eb-2bdea6d76c6a
```

사용자 Chef 로그

사용자의 Chef 실행에 대한 로그는 다음 다이어그램과 같이 Chef 명령의 번호를 사용하여 명명되는 폴더 안의 `logfile.txt`라는 이름의 파일에서 확인할 수 있습니다.

C:/chef └── `runs` └── `command-12345` ├── `attribs.json` ├── `client.rb` └── `logfile.txt`

Chef 로그 해석

로그의 첫 부분은 대부분 내부 Chef 로깅을 포함합니다

```
# Logfile created on Thu Oct 17 17:25:12 +0000 2013 by logger.rb/1.2.6
[2013-10-17T17:25:12+00:00] INFO: *** Chef 11.4.4 ***
[2013-10-17T17:25:13+00:00] DEBUG: Building node object for php-app1.localdomain
[2013-10-17T17:25:13+00:00] DEBUG: Extracting run list from JSON attributes provided on
 command line
[2013-10-17T17:25:13+00:00] INFO: Setting the run_list to
 ["opsworks_custom_cookbooks::load", "opsworks_custom_cookbooks::execute"] from JSON
[2013-10-17T17:25:13+00:00] DEBUG: Applying attributes from json file
[2013-10-17T17:25:13+00:00] DEBUG: Platform is amazon version 2013.03
[2013-10-17T17:25:13+00:00] INFO: Run List is [recipe[opsworks_custom_cookbooks::load],
 recipe[opsworks_custom_cookbooks::execute]]
[2013-10-17T17:25:13+00:00] INFO: Run List expands to [opsworks_custom_cookbooks::load,
 opsworks_custom_cookbooks::execute]
[2013-10-17T17:25:13+00:00] INFO: Starting Chef Run for php-app1.localdomain
[2013-10-17T17:25:13+00:00] INFO: Running start handlers
[2013-10-17T17:25:13+00:00] INFO: Start handlers complete.
[2013-10-17T17:25:13+00:00] DEBUG: No chefignore file found at /opt/aws/opsworks/
releases/20131015111601_209/cookbooks/chefignore no files will be ignored
```

```
[2013-10-17T17:25:13+00:00] DEBUG: Cookbooks to compile: ["gem_support", "packages",
 "opsworks_bundler", "opsworks_rubygems", "ruby", "ruby_enterprise", "dependencies",
 "opsworks_commons", "scm_helper", :opsworks_custom_cookbooks]
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook gem_support's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/gem_support/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook packages's library file: /opt/aws/
opsworks/releases/20131015111601_209/cookbooks/packages/libraries/packages.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook dependencies's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/dependencies/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
activesupport_blank.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
monkey_patch_chefgem_resource.rb
...
```

파일의 이 부분은 주로 Chef 전문가에게 유용합니다. 대부분의 명령이 훨씬 더 많은 레시피와 관련되지만 실행 목록에는 두 개의 레시피만 나열됩니다. 이들 두 레시피는 다른 모든 내장 및 사용자 지정 레시피를 로드 및 실행하는 작업을 처리합니다.

이 파일에서 가장 중요한 부분은 일반적으로 끝부분입니다. 실행이 성공적으로 종료하면 다음과 같은 내용이 보일 것입니다.

```
...
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: STDERR:
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: ---- End output of /sbin/service mysqld
 restart ----
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Ran /sbin/service mysqld restart returned 0
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: service[mysql]: restarted successfully
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Chef Run complete in 84.07096 seconds
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: cleaning the checksum cache
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-8wef7e-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-1xpwyb6-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--etc-monit-conf
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Running report handlers
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Report handlers complete
```

```
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Exiting
```

> **Note**
>
> 에이전트 CLI를 사용하여 실행 도중 또는 이후에 로그의 끝부분을 표시할 수 있습니다. 자세한 내용은 Chef 로그 표시 단원을 참조하십시오.

레시피가 실패할 경우 ERROR 수준 출력을 찾아야 합니다. 이 출력에는 예외, 그 다음에 Chef 스택 트레이스가 기록됩니다. 다음은 그 예입니다.

```
...
Please report any problems with the /usr/scripts/mysqlbug script!

[  OK  ]
MySQL Daemon failed to start.
Starting mysqld:  [FAILED]STDERR: 130611 15:07:55 [Warning] The syntax '--log-slow-
queries' is deprecated and will be removed in a future release. Please use '--slow-
query-log'/'--slow-query-log-file' instead.
130611 15:07:56 [Warning] The syntax '--log-slow-queries' is deprecated and will be
 removed in a future release. Please use '--slow-query-log'/'--slow-query-log-file'
 instead.
---- End output of /sbin/service mysqld start ----

/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:184:in `handle_command_failures'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:131:in `run_command'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service/init.rb:37:in `start_service'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service.rb:60:in `action_start'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `send'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:53:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
```

```
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `each'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:94:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in `call'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in
 `call_iterator_block'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:85:in `step'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:104:in `iterate'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:55:in
 `each_with_index'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:92:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:84:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:268:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:158:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:190:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `loop'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application.rb:62:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/chef-solo:25
  /opt/aws/opsworks/current/bin/chef-solo:16:in `load'
  /opt/aws/opsworks/current/bin/chef-solo:16
```

파일의 끝부분이 Chef 스택 트레이스입니다. 예외 바로 앞의 출력도 검토해야 합니다. 이 출력에 실패 원인을 판단하는 데 유용할 수 있는 `package not available`과 같은 시스템 오류가 포함되는 경우가 종종 있습니다. 이 경우 MySQL 데몬이 시작하지 못한 것입니다.

일반적인 Chef 로그 오류

다음은 일반적인 Chef 로그 오류와 그 해결 방법입니다.

로그를 찾을 수 없음

Chef 실행 시작 후 Chef 실행이 끝났을 때 웹 페이지에서 로그를 볼 수 있게 해주는 미리 서명된 Amazon S3 URL이 인스턴스로 전송됩니다. 이 URL은 2시간 후에 만료되므로 Chef 실행이 2시간 이상 걸릴 경우 Chef 실행 도중 문제가 발생하지 않는다 해도 로그가 Amazon S3 사이트로 업로드되지 않습니다. 로그 생성 명령은 성공하지만 로그를 미리 서명된 URL에서가 아니라 인스턴스에서만 볼 수 있습니다.

로그가 갑자기 끝남

Chef 로그가 성공을 나타내거나 오류 정보를 표시하지 않고 갑자기 끝날 경우 아마도 메모리 부족 상태가 발생하여 Chef가 로그를 완료하지 못한 것일 수 있습니다. 최선의 옵션은 더 큰 용량의 인스턴스를 사용하여 다시 시도하는 것입니다.

쿡북 또는 레시피 누락

Chef가 쿡북 캐시에 없는 쿡북 또는 레시피를 마주칠 경우 다음과 같은 로그가 기록됩니다.

```
DEBUG: Loading Recipe mycookbook::myrecipe via include_recipe
ERROR: Caught exception during execution of custom recipe: mycookbook::myrecipe:
   Cannot find a cookbook named mycookbook; did you forget to add metadata to a
 cookbook?
```

이 항목은 `mycookbook` 쿡북이 쿡북 캐시에 없음을 나타냅니다. Chef 11.4에서는 `metadata.rb`에서 종속성을 올바로 선언하지 않을 경우에도 이 오류가 발생할 수 있습니다.

AWS OpsWorks Stacks는 인스턴스의 쿡북 캐시에서 레시피를 실행합니다. 인스턴스가 시작하면 쿡북이 리포지토리에서 이 캐시로 다운로드됩니다. 그러나 이후에 리포지토리의 쿡북을 수정하는 경우 AWS OpsWorks Stacks는 온라인 인스턴스의 캐시를 자동으로 업데이트하지 않습니다. 인스턴스 시작 후 쿡북을 수정했거나 새 쿡북을 추가한 경우 다음 단계를 수행하세요.

1. 리포지토리에서 변경을 커밋했는지 확인합니다.
2. Update Cookbooks 스택 명령을 실행하여 리포지토리의 최신 버전으로 쿡북 캐시를 업데이트합니다.

로컬 명령 실패

Chef `execute` 리소스가 지정된 명령을 실행하지 못할 경우 다음과 같은 로그가 기록됩니다.

```
DEBUG: ---- End output of ./configure --with-config-file-path=/ returned 2
ERROR: execute[PHP: ./configure] (/root/opsworks-agent/site-cookbooks/php-fpm/
recipes/install.rb line 48) had an error:
   ./configure --with-config-file-path=/
```

로그를 위로 스크롤하면 명령의 `stderr` 및 `stdout` 출력이 보일 것입니다. 이들은 명령이 실패한 이유를 판단하는 데 도움이 됩니다.

패키지 실패

패키지 설치가 실패할 경우 다음과 같은 로그가 기록됩니다.

```
ERROR: package[zend-server-ce-php-5.3] (/root/opsworks-agent/site-cookbooks/
zend_server/recipes/install.rb line 20)
   had an error: apt-get -q -y --force-yes install zend-server-ce-php-5.3=5.0.4+b17
 returned 100, expected 0
```

로그를 위로 스크롤하면 명령의 STDOUT 및 STDERROR 출력이 보일 것입니다. 이들은 패키지 설치가 실패한 이유를 판단하는 데 도움이 됩니다.

AWS OpsWorks Stacks 에이전트 CLI 사용

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Note

에이전트 CLI는 Linux 인스턴스에서만 사용할 수 있습니다.

모든 온라인 인스턴스에서 AWS OpsWorks Stacks는 서비스와 통신하는 에이전트를 설치합니다. AWS OpsWorks Stacks 서비스는 수명 주기 이벤트가 발생할 때 인스턴스에서 Chef 실행을 시작하는

등의 작업을 수행하도록 에이전트에게 명령을 전송합니다. Linux 인스턴스에서는 에이전트가 문제 해결에 매우 유용한 명령줄 인터페이스(CLI)를 표시합니다. 에이전트 CLI 명령을 실행하려면 SSH를 사용하여 인스턴스에 연결합니다. 그런 다음 에이전트 CLI 명령을 실행하여 다음을 포함하여 다양한 작업을 수행합니다.

- 레시피를 실행.
- Chef 로그를 표시.
- 스택 구성 및 배포 JSON를 표시

인스턴스와 SSH 연결을 설정하는 방법에 대한 자세한 정보는 SSH를 사용하여 로그인 단원을 참조하세요. 또한 스택에 대한 SSH 및 sudo 권한도 있어야 합니다.

이 섹션에서는 문제 해결을 위해 에이전트 CLI를 사용하는 방법을 설명합니다. 자세한 내용 및 전체 명령 참조는 AWS OpsWorks Stacks 에이전트 CLI 단원을 참조하세요.

주제

- 레시피 실행
- Chef 로그 표시
- 스택 구성 및 배포 JSON 표시

레시피 실행

에이전트 CLI `run_command` 명령은 에이전트에게 앞서 실행한 명령을 재실행하도록 지시합니다. 문제 해결에 가장 유용한 명령 `setup`, `configure`, `deploy` 및 `undeploy`은 각각 수명 주기 이벤트에 해당합니다. 이들 명령은 에이전트에게 Chef 실행을 시작하여 연결된 레시피를 실행하도록 지시합니다.

> **Note**
>
> `run_command` 명령은 지정된 명령과 연결된 레시피 그룹(일반적으로 특정 수명 주기 이벤트에 연결된 레시피)을 실행하는 것으로 제한됩니다. 이 명령을 사용하여 특정 레시피를 실행할 수는 없습니다. 하나 이상의 지정된 레시피를 실행하려면 레시피 실행 스택 명령 또는 이와 동등한 CLI 또는 API 작업(`create-deployment` 및 `CreateDeployment`)을 사용합니다.

`run_command` 명령은 사용자 지정 레시피, 특히 설정 및 Configure 수명 주기 이벤트에 할당된 것으로서 콘솔에서 직접 트리거할 수 없는 레시피를 디버깅하는 데 매우 유용합니다. `run_command`를 사용

하여 특정 이벤트의 레시피를 인스턴스를 시작 또는 중지하지 않고도 원하는 횟수 만큼 실행할 수 있습니다.

> **Note**
>
> AWS OpsWorks Stacks는 쿡북 리포지토리가 아닌 인스턴스의 쿡북 캐시에서 레시피를 실행합니다. AWS OpsWorks Stacks는 인스턴스가 시작될 때이 캐시에 쿡북을 다운로드하지만 나중에 쿡북을 수정할 경우 온라인 인스턴스에서 캐시를 자동으로 업데이트하지 않습니다. 인스턴스 시작 이후 쿡북을 수정하지 않았다면 Update Cookbooks 스택 명령을 실행하여 리포지토리의 최신 버전으로 쿡북 캐시를 업데이트하세요.

에이전트는 최근 명령만 캐시합니다. `list_commands`를 실행하여 최근 명령을 나열할 수 있습니다. 그러면 캐시된 명령과 명령 실행 시간의 목록이 반환됩니다.

```
sudo opsworks-agent-cli list_commands
2013-02-26T19:08:26        setup
2013-02-26T19:12:01        configure
2013-02-26T19:12:05        configure
2013-02-26T19:22:12        deploy
```

최근 명령을 재실행하려면 다음을 실행합니다.

```
sudo opsworks-agent-cli run_command
```

지정된 명령의 최근 인스턴스를 재실행하려면 다음을 실행합니다.

```
sudo opsworks-agent-cli run_command command
```

예를 들어 설정 레시피를 재실행하려면 다음 명령을 실행할 수 있습니다.

```
sudo opsworks-agent-cli run_command setup
```

각 명령에는 명령이 실행된 당시의 스택 및 배포 상태를 나타내는 스택 구성 및 배포 JSON이 연결되어 있습니다. 이 데이터는 명령마다 바뀔 수 있으므로 명령의 이전 인스턴스와 최근 인스턴스는 다른 데이터를 사용할 수 있습니다. 명령의 특정 인스턴스를 재실행하려면 `list_commands` 출력에서 시간을 복사하고 다음을 실행합니다.

```
sudo opsworks-agent-cli run_command time
```

이전 예제는 모두 해당 명령에 대해 설치된 JSON인 기본 JSON을 사용하여 명령을 재실행합니다. 다음과 같이 임의의 JSON 파일을 사용하여 명령을 재실행할 수 있습니다.

```
sudo opsworks-agent-cli run_command -f /path/to/valid/json.file
```

Chef 로그 표시

에이전트 CLI show_log 명령은 지정된 로그를 표시합니다. 명령이 완료되면 파일의 끝부분이 표시됩니다. 따라서 show_log 명령은 일반적으로 오류 정보를 찾을 수 있는 로그의 끝부분을 확인하는 편리한 방법을 제공합니다. 로그를 위로 스크롤하면 이전 부분을 볼 수 있습니다.

현재 명령의 로그를 표시하려면 다음을 실행합니다.

```
sudo opsworks-agent-cli show_log
```

특정 명령에 대한 로그를 표시할 수도 있지만, 에이전트는 마지막 30개의 명령에 대해서만 로그를 캐시한다는 점을 염두에 두십시오. list_commands를 실행하여 인스턴스의 명령을 나열할 수 있습니다. 그러면 캐시된 명령과 명령 실행 시간의 목록이 반환됩니다. 예시는 레시피 실행에서 확인하십시오.

특정 명령의 최근 실행에 대한 로그를 표시하려면 다음을 실행합니다.

```
sudo opsworks-agent-cli show_log command
```

명령 파라미터는 setup, configure, deploy, undeploy, start, stop 또는 restart로 설정할 수 있습니다. 이들 명령은 대부분 수명 주기 이벤트에 해당하며 에이전트에게 연결된 레시피를 실행하도록 지시합니다.

특정 명령 실행에 대한 로그를 표시하려면 list_commands 출력에서 날짜를 복사하고 다음을 실행합니다.

```
sudo opsworks-agent-cli show_log date
```

실행 중인 명령이 있을 경우 show_log가 로그의 현재 상태를 표시합니다.

> **Note**
>
> 오류 및 메모리 부족 문제를 해결하는 데 show_log를 사용하는 한 가지 방법은 다음과 같이 실행 도중 로그의 끝부분을 표시하는 것입니다.

1. run_command를 사용하여 해당 수명 주기 이벤트를 트리거합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.
2. show_log명령을 반복적으로 실행하여 로그가 기록되는 동안 로그의 끝부분을 확인합니다.

Chef에서 메모리 부족 또는 예기치 않은 종료가 발생하는 경우 로그가 갑자기 끝납니다. 레시피가 실패할 경우 로그가 예외 및 스택 트레이스로 끝납니다.

스택 구성 및 배포 JSON 표시

레시피가 사용하는 데이터는 많은 부분이 스택 구성 및 배포 JSON에서 나옵니다. 이 JSON은 스택 구성, 배포, 그리고 사용자가 선택적으로 추가할 수 있는 사용자 지정 속성에 대한 상세한 설명을 제공하는 Chef 속성 세트를 정의합니다. 각 명령에 대해 AWS OpsWorks Stacks는 명령의 시점에 스택 및 배포 상태를 나타내는 JSON을 설치합니다. 자세한 내용은 스택 구성 및 배포 속성 단원을 참조하십시오.

사용자 지정 레시피가 스택 구성 및 배포 JSON으로부터 데이터를 가져오는 경우 JSON을 검사하여 데이터를 확인할 수 있습니다. 스택 구성 및 배포 JSON을 표시하는 가장 간단한 방법은 에이전트 CLI get_json 명령을 실행하는 것입니다. 이 명령은 JSON 객체의 서식 지정된 버전을 표시합니다. 다음은 일반적인 출력의 처음 몇 줄입니다.


```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "10.112.235.192",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
```

```
            "public_dns_name": "ec2-10-112-235-192.us-west-2.compute.amazonaws.com"
...
```

다음과 같이 최근의 스택 구성 및 배포 JSON을 표시할 수 있습니다.

```
sudo opsworks-agent-cli get_json
```

다음을 실행하여 지정된 명령에 대해 최근의 스택 구성 및 배포 JSON을 표시할 수 있습니다.

```
sudo opsworks-agent-cli get_json command
```

명령 파라미터는 `setup`, `configure`, `deploy`, `undeploy`, `start`, `stop` 또는 `restart`로 설정할 수 있습니다. 이들 명령은 대부분 수명 주기 이벤트에 해당하며 에이전트에게 연결된 레시피를 실행하도록 지시합니다.

다음과 같이 명령의 날짜를 지정하여 특정 명령 실행에 대해 스택 구성 및 배포 JSON을 표시할 수 있습니다.

```
sudo opsworks-agent-cli get_json date
```

이 명령을 사용하는 가장 간단한 방법은 다음과 같습니다.

1. `list_commands`를 실행합니다. 그러면 인스턴스에서 실행된 명령과 각 명령이 실행된 날짜의 목록이 반환됩니다.
2. 해당 명령의 날짜를 복사하여 `get_json` *`date`* 인수로 사용합니다.

일반적인 디버깅 및 문제 해결

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에서는 일반적으로 발생하는 일부 디버깅 및 문제 해결과 해법을 설명합니다.

주제

- AWS OpsWorks 스택 문제 해결
- 인스턴스 등록 문제 해결

AWS OpsWorks 스택 문제 해결

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 섹션에는 일반적으로 발생하는 몇 가지 AWS OpsWorks Stacks 문제와 해결 방법이 포함되어 있습니다.

주제

- 인스턴스를 관리할 수 없습니다
- Chef 실행 후 인스턴스가 부팅되지 않습니다
- 계층의 인스턴스가 Elastic Load Balancing 상태 확인에 모두 실패
- Elastic Load Balancing 로드 밸런서와 통신할 수 없음
- 가져온 온프레미스 인스턴스가 재시작 후 볼륨 설정을 완료하는 데 실패합니다
- 재부팅 후 EBS 볼륨이 다시 연결되지 않습니다
- AWS OpsWorks Stacks 보안 그룹을 삭제할 수 없습니다
- AWS OpsWorks Stacks 보안 그룹을 실수로 삭제했습니다.
- Chef 로그가 갑자기 종료됩니다
- 쿡북이 업데이트되지 않습니다
- 인스턴스가 부팅 상태에 멈춰 있습니다
- 인스턴스가 예기치 않게 다시 시작됩니다
- opsworks-agent 프로세스가 인스턴스에서 실행 중입니다
- 예기치 않은 execute_recipes 명령

인스턴스를 관리할 수 없습니다

문제: 전에는 관리할 수 있었던 인스턴스를 더 이상 관리할 수 없습니다. 일부 경우, 로그에 다음과 비슷한 오류가 표시될 수 있습니다.

```
Aws::CharlieInstanceService::Errors::UnrecognizedClientException - The security token
 included in the request is invalid.
```

원인: 이 문제는 인스턴스가 의존하는 AWS OpsWorks 외부의 리소스가 편집 또는 삭제된 경우, 발생할 수 있습니다. 다음은 인스턴스와의 통신을 끊을 수 있는 리소스 변경의 예입니다.

- 인스턴스와 연결된 IAM 사용자 또는 역할이 AWS OpsWorks Stacks 외부에서 실수로 삭제되었습니다. 이로 인해 인스턴스에 설치된 AWS OpsWorks 에이전트와 Stacks 서비스 간에 통신이 실패합니다 AWS OpsWorks . 인스턴스에 연결된 사용자는 인스턴스의 수명이 끝날 때까지 필요합니다.
- 인스턴스가 오프라인일 때 볼륨 또는 스토리지 구성을 편집하면 인스턴스를 관리할 수 없게 될 수 있습니다.
- EC2 인스턴스를 ELB에 수동으로 추가합니다.는 인스턴스가 온라인 상태가 되거나 온라인 상태에서 나갈 때마다 할당된 Elastic Load Balancing 로드 밸런서를 AWS OpsWorks 재구성합니다.는 알고 있는 인스턴스 AWS OpsWorks 만 유효한 멤버로 간주하고, 외부 AWS OpsWorks또는 다른 프로세스에 의해 추가된 인스턴스는 제거됩니다. 인스턴스는 하나 걸러 하나씩 제거됩니다.

해결 방법: 인스턴스가 의존하는 IAM 사용자나 역할을 삭제하지 마세요. 가능하다면 종속 인스턴스가 실행 중일 때만 볼륨 또는 스토리지 구성을 편집하세요. AWS OpsWorks 를 사용하여 AWS OpsWorks 인스턴스의 로드 밸런서 또는 EIP 멤버십을 관리합니다. 인스턴스를 등록할 때 사용자가 우연히 삭제될 경우 발생하는 등록된 인스턴스 관리 문제를 예방하려면, `register` 명령에 `--use-instance-profile` 파라미터를 추가하여 인스턴스의 내장 인스턴스 프로파일을 대신 사용합니다.

Chef 실행 후 인스턴스가 부팅되지 않습니다

문제: 사용자 지정 쿡북을 사용하도록 구성된 Chef 11.10 또는 이전의 스택에서 커뮤니티 쿡북을 사용한 Chef 실행 후 인스턴스가 부팅되지 않습니다. 로그 메시지에는 레시피가 컴파일에 실패했다거나 ("Recipe Compile Error") 종속성을 찾을 수 없어서 로드할 수 없다고 표시될 수 있습니다.

원인: 가장 가능성 높은 원인은 사용자 지정 또는 커뮤니티 쿡북이 스택이 사용하는 Chef 버전을 지원하지 않는 것입니다. `apt` 및 `build-essential`과 같은 일부 인기 있는 커뮤니티 쿡북은 Chef 11.10과의 알려진 호환성 문제가 있습니다.

솔루션: 사용자 지정 Chef AWS OpsWorks 쿡북 사용 설정이 켜져 있는 스택 스택의 경우 사용자 지정 또는 커뮤니티 쿡북은 스택에서 사용하는 Chef 버전을 항상 지원해야 합니다. 스택 설정에서 구성된

Chef 버전과 호환되는 버전으로 커뮤니티 쿡북을 고정시키십시오(즉, 쿡북 버전을 특정 버전으로 설정). 지원되는 커뮤니티 쿡북 버전을 찾으려면 컴파일에 실패하는 쿡북의 변경 로그를 보고 스택이 지원할 쿡북의 가장 최근 버전만을 사용합니다. 쿡북 버전을 고정하려면 사용자 지정 쿡북 리포지토리의 Berksfile에서 정확한 버전 번호를 지정하세요. 예: `cookbook 'build-essential', '= 3.2.0'`.

계층의 인스턴스가 Elastic Load Balancing 상태 확인에 모두 실패

문제: Elastic Load Balancing 로드 밸런서를 앱 서버 계층에 연결하지만 모든 인스턴스가 상태 확인에 실패합니다.

원인: Elastic Load Balancing 로드 밸런서를 생성할 때 인스턴스 상태가 정상인지 확인하기 위해 로드 밸런서가 호출하는 ping 경로를 지정해야 합니다. 애플리케이션에 적합한 ping 경로를 지정해야 합니다. 기본값은 /index.html입니다. 애플리케이션에 `index.html`이 포함되지 않은 경우, 적절한 경로를 지정해야 상태 확인이 실패하지 않습니다. 예를 들어 Chef 11 Linux 스택 시작하기에서 사용되는 SimplePHPApp 애플리케이션은 index.html을 사용하지 않습니다. 이러한 서버에 적절한 ping 경로는 /입니다.

해결책: 로드 밸런서의 ping 경로를 편집합니다. 자세한 내용은 Elastic Load Balancing을 참조하세요.

Elastic Load Balancing 로드 밸런서와 통신할 수 없음

문제: Elastic Load Balancing 로드 밸런서를 생성해 앱 서버 계층에 연결했지만 애플리케이션을 실행하기 위해 로드 밸런서의 DNS 이름이나 IP 주소를 클릭하면 "원격 서버가 응답하지 않습니다"라는 오류가 표시됩니다.

원인: 스택이 기본 VPC에서 실행되는 경우, 리전에서 Elastic Load Balancing 로드 밸런서를 생성할 때 보안 그룹을 지정해야 합니다. 보안 그룹에는 IP 주소로부터의 인바운드 트래픽을 허용하는 수신 규칙이 있어야 합니다. [기본 VPC 보안 그룹]을 지정한 경우, 기본 수신 규칙은 인바운드 트래픽을 전혀 허용하지 않습니다.

해결책: 적절한 IP 주소로부터의 인바운드 트래픽을 허용하도록 보안 그룹 수신 규칙을 편집합니다.

1. Amazon EC2 콘솔의 탐색 창에서 보안 그룹을 클릭합니다.
2. 로드 밸런서의 보안 그룹을 선택합니다.
3. [인바운드] 탭에서 [편집]을 클릭합니다.
4. [소스]를 적절한 CIDR로 설정하고 수신 규칙을 추가합니다.

   예를 들어 [모든 곳]를 지정하면 CIDR이 0.0.0.0/0으로 설정되어 모든 IP 주소의 수신 트래픽을 허용하도록 로드 밸런서에게 명령합니다.

가져온 온프레미스 인스턴스가 재시작 후 볼륨 설정을 완료하는 데 실패합니다

문제: AWS OpsWorks Stacks로 가져온 EC2 인스턴스를 다시 시작하면 AWS OpsWorks Stacks 콘솔에 인스턴스 상태로 실패가 표시됩니다. 이 문제는 Chef 11 또는 Chef 12 인스턴스에서 발생할 수 있습니다.

원인: AWS OpsWorks Stacks가 설정 프로세스 도중 인스턴스에 볼륨을 연결하지 못할 수 있습니다. 가능한 한 가지 원인은 `setup` 명령을 실행할 때 AWS OpsWorks Stacks가 인스턴스에서 볼륨 구성을 덮어쓰는 것입니다.

해결책: 인스턴스의 [세부 정보] 페이지를 열고 [볼륨] 영역에서 볼륨 구성을 검사합니다. 볼륨 구성은 인스턴스가 [중지됨] 상태일 때만 변경할 수 있습니다. 모든 볼륨의 탑재 지점과 이름이 지정되어 있어야 합니다. 인스턴스를 다시 시작하기 전에 AWS OpsWorks Stacks에서 구성에 올바른 탑재 지점을 제공했는지 확인합니다.

재부팅 후 EBS 볼륨이 다시 연결되지 않습니다

문제: 콘솔을 사용하여 Amazon EBS 볼륨을 인스턴스에 연결했지만 인스턴스를 재부팅할 때 볼륨이 더 이상 연결되지 않습니다.

원인: AWS OpsWorks Stacks는 알고 있는 Amazon EBS 볼륨만 다시 연결할 수 있으며, 이는 다음으로 제한됩니다.

- AWS OpsWorks Stacks에서 생성한 볼륨입니다.
- [리소스] 페이지를 사용하여 스택에 명시적으로 등록한 계정의 볼륨.

솔루션: AWS OpsWorks Stacks 콘솔, API 또는 CLI를 사용해야만 Amazon EBS 볼륨을 관리할 수 있습니다. 스택에 계정의 Amazon EBS 볼륨 중 하나를 사용하려면 스택의 리소스 페이지를 사용하여 볼륨을 등록하고 인스턴스에 연결하세요. 자세한 내용은 리소스 관리 단원을 참조하십시오.

AWS OpsWorks Stacks 보안 그룹을 삭제할 수 없습니다

문제: 스택을 삭제한 후에는 삭제할 수 없는 여러 AWS OpsWorks Stacks 보안 그룹이 남아 있습니다.

원인: 보안 그룹은 특정 순서로 삭제해야 합니다.

해결책: 먼저 보안 그룹을 사용 중인 인스턴스가 없어야 합니다. 그 후 다음의 보안 그룹(존재하는 경우)을 다음 순서로 삭제하세요.

1. AWS-OpsWorks-Blank-Server

2. AWS-OpsWorks-Monitoring-Master-Server
3. AWS-OpsWorks-DB-Master-Server
4. AWS-OpsWorks-Memcached-Server
5. AWS-OpsWorks-Custom-Server
6. AWS-OpsWorks-nodejs-App-Server
7. AWS-OpsWorks-PHP-App-Server
8. AWS-OpsWorks-Rails-App-Server
9. AWS-OpsWorks-Web-Server
10. AWS-OpsWorks-Default-Server
11. AWS-OpsWorks-LB-Server

AWS OpsWorks Stacks 보안 그룹을 실수로 삭제했습니다.

문제: AWS OpsWorks Stacks 보안 그룹 중 하나를 삭제했으며 다시 생성해야 합니다.

원인: 이러한 보안 그룹은 보통 우연히 삭제됩니다.

해결책: 다시 생성되는 그룹은 그룹 이름의 동일한 대문자를 포함하여 원본의 정확한 복제본이어야 합니다. 수동으로 그룹을 다시 생성하는 것보다는 AWS OpsWorks Stacks가 대신 이 작업을 수행하도록 하는 것이 좋습니다. 동일한 AWS 리전과 VPC에 새 스택을 생성하면 AWS OpsWorks Stacks는 삭제한 보안 그룹을 포함하여 모든 기본 제공 보안 그룹을 자동으로 다시 생성합니다. 그런 다음 더 이상 사용할 일이 없으면 스택을 삭제할 수 있습니다. 보안 그룹은 그대로 남습니다.

Chef 로그가 갑자기 종료됩니다

문제: Chef 로그가 갑자기 종료됩니다. 로그의 끝에 성공적 실행이 표시되지 않거나 예외 및 스택 추적이 표시되지 않습니다.

원인: 이 동작의 원인은 일반적으로 메모리 부족입니다.

해결책: 더 큰 인스턴스를 생성해 에이전트 CLI `run_command` 명령을 사용하여 레시피를 다시 실행합니다. 자세한 내용은 레시피 실행 단원을 참조하십시오.

쿡북이 업데이트되지 않습니다

문제: 쿡북을 업데이트했으나 스택의 인스턴스가 계속해서 이전 레시피를 실행 중입니다.

원인: AWS OpsWorks 스택은 각 인스턴스에 쿡북을 캐시하고 리포지토리가 아닌 캐시에서 레시피를 실행합니다. 새 인스턴스를 시작하면 AWS OpsWorks Stacks는 리포지토리에서 인스턴스의 캐시로 쿡북을 다운로드합니다. 하지만 이후에 사용자 지정 쿡북을 수정하면 AWS OpsWorks Stacks는 온라인 인스턴스의 캐시를 자동으로 업데이트하지 않습니다.

해결 방법: 쿡북 스택 업데이트 명령을 실행하여 Stacks에 온라인 인스턴스의 AWS OpsWorks 쿡북 캐시를 업데이트하도록 명시적으로 지시합니다.

인스턴스가 부팅 상태에 멈춰 있습니다

문제: 인스턴스를 다시 시작하거나 자동 복구가 인스턴스를 자동으로 다시 시작할 때 시작 작업이 `booting` 상태에 멈춰 있습니다.

원인: 이 문제의 가능한 원인 한 가지는 기본 VPC를 포함한 VPC 구성입니다. 인스턴스는 항상 AWS OpsWorks Stacks 서비스, Amazon S3 및 패키지, 쿡북 및 앱 리포지토리와 통신할 수 있어야 합니다. 예를 들어 기본 VPC에서 기본 게이트웨이를 제거하면 인스턴스가 AWS OpsWorks Stacks 서비스에 대한 연결이 끊어집니다. AWS OpsWorks Stacks는 더 이상 인스턴스 에이전트와 통신할 수 없으므로 인스턴스를 실패로 취급하고 자동으로 복구합니다. 그러나 연결 없이는 AWS OpsWorks Stacks가 복구된 인스턴스에 인스턴스 에이전트를 설치할 수 없습니다. 에이전트가 없으면 AWS OpsWorks Stacks는 인스턴스에서 설정 레시피를 실행할 수 없으므로 시작 작업은 "부팅" 상태를 초과하여 진행될 수 없습니다.

해결책: 인스턴스가 필요한 연결을 갖추도록 VPC 구성을 수정합니다.

인스턴스가 예기치 않게 다시 시작됩니다

문제: 정지한 인스턴스가 예기치 않게 다시 시작합니다.

원인 1: 인스턴스에 대해 자동 복구를 활성화한 경우, AWS OpsWorks Stacks는 연결된 Amazon EC2 인스턴스에서 주기적으로 상태 확인을 수행하고 비정상 상태인 인스턴스를 다시 시작합니다. Amazon EC2 콘솔, API 또는 CLI를 AWS OpsWorks 사용하여 Stacks 관리형 인스턴스를 중지하거나 종료하면 AWS OpsWorks Stacks에 알림이 전송되지 않습니다. 그 대신 정지된 인스턴스를 비정상으로 인식하고 자동으로 해당 인스턴스를 다시 시작합니다.

해결 방법: AWS OpsWorks Stacks 콘솔, API 또는 CLI만을 사용하여 인스턴스를 관리합니다. AWS OpsWorks Stacks를 사용하여 인스턴스를 중지하거나 삭제하는 경우 인스턴스가 다시 시작되지 않습니다. 자세한 내용은 수동으로 24/7 인스턴스 시작, 중지 및 재부팅 및 AWS OpsWorks 스택 인스턴스 삭제 단원을 참조하세요.

원인 2: 인스턴스는 다양한 이유로 실패할 수 있습니다. 자동 복구를 활성화한 경우 AWS OpsWorks Stacks는 실패한 인스턴스를 자동으로 다시 시작합니다.

해결 방법: 이는 정상적인 작업입니다. Stacks가 실패한 인스턴스를 다시 시작하지 않도록 AWS OpsWorks 하지 않는 한 아무 작업도 할 필요가 없습니다. 이 경우, 자동 복구를 비활성화해야 합니다.

opsworks-agent 프로세스가 인스턴스에서 실행 중입니다

문제: 몇몇 `opsworks-agent` 프로세스가 인스턴스에서 실행 중입니다. 예시:

```
aws 24543 0.0 1.3 172360 53332 ? S Feb24 0:29 opsworks-agent: master 24543
aws 24545 0.1 2.0 208932 79224 ? S Feb24 22:02 opsworks-agent: keep_alive of master
 24543
aws 24557 0.0 2.0 209012 79412 ? S Feb24 8:04 opsworks-agent: statistics of master
 24543
aws 24559 0.0 2.2 216604 86992 ? S Feb24 4:14 opsworks-agent: process_command of master
 24
```

원인: 이들은 에이전트의 정상 작동에 필요한 올바른 프로세스입니다. 이들은 배포를 처리하고 서비스에 연결 유지 메시지를 다시 전송하는 등의 작업을 수행합니다.

해결책: 이것은 정상적 동작입니다. 이 프로세스들을 중지하지 마십시오. 중지할 경우, 에이전트 작동이 손상될 수 있습니다.

예기치 않은 execute_recipes 명령

문제: 인스턴스 세부 정보 페이지의 [로그] 섹션에 예기치 않은 `execute_recipes` 명령이 포함되어 있습니다. 예기치 않은 `execute_recipes` 명령은 [스택] 및 [배포] 페이지에도 표시될 수 있습니다.

원인: 이 문제는 흔히 권한 변경으로 인해 발생합니다. 사용자 또는 그룹의 SSH 또는 sudo 권한을 변경하면 AWS OpsWorks Stacks가 실행`execute_recipes`되어 스택의 인스턴스를 업데이트하고 구성 이벤트도 트리거합니다. `execute_recipes` 명령의 또 다른 출처는 인스턴스 에이전트를 업데이트하는 AWS OpsWorks Stacks입니다.

해결책: 이것은 정상 작동이며 아무 조치도 취할 필요가 없습니다.

`execute_recipes` 명령이 어떤 작업을 수행했는지 보려면 [배포] 페이지로 가서 명령의 타임스탬프를 클릭하세요. 그러면 실행된 주요 레시피가 나열된 명령의 세부 정보 페이지가 열립니다. 예를 들어 다음 세부 정보 페이지는 `ssh_users`를 실행하여 SSH 권한을 업데이트한 `execute_recipes` 명령의 세부 정보 페이지입니다.

모든 세부 정보를 보려면 명령의 로그 열에서 표시를 클릭하여 관련 Chef 로그를 표시하세요. 로그에서를 검색합니다**`Run List`**. AWS OpsWorks 스택 유지 관리 레시피는 아래에 있습니다`OpsWorks Custom Run List`. 예를 들어 다음은 이전 스크린샷에 나온 `execute_recipes` 명령의 실행 목록으로서 명령에 연결된 모든 레시피를 보여 줍니다.

```
[2014-02-21T17:16:30+00:00] INFO: OpsWorks Custom Run List:
 ["opsworks_stack_state_sync",
  "ssh_users", "test_suite", "opsworks_cleanup"]
```

인스턴스 등록 문제 해결

이 섹션에는 일반적으로 발생하는 일부 인스턴스 등록 문제와 해결 방법이 나와 있습니다.

> **Note**
>
> 등록 문제가 있는 경우, 추가 디버깅 정보를 제공하는 `--debug` 인수와 함께 `register`를 실행합니다.

주제

- EC2User에게 다음을 수행할 권한이 부여되지 않았습니다
- 자격 증명 범위가 유효한 리전으로 지정되어야 합니다

EC2User에게 다음을 수행할 권한이 부여되지 않았습니다

문제: `register` 명령이 다음과 비슷한 결과를 반환합니다.

```
A client error (AccessDenied) occurred when calling the CreateGroup operation:
User: arn:aws:iam::123456789012:user/ImportEC2User is not authorized to
perform: iam:CreateGroup on resource:
arn:aws:iam::123456789012:group/AWS/OpsWorks/OpsWorks-b583ce55-1d01-4695-b3e5-
ee19257d1911
```

원인: 필요한 권한을 부여하지 않는 자격 증명을 사용하여 `register` 명령이 실행 중입니다. 사용자의 정책이 무엇보다 `iam:CreateGroup` 작업을 허용해야 합니다.

해결책 필요한 권한이 있는 IAM 자격 증명을 `register`에 제공합니다. 자세한 내용은 AWS CLI설치 및 구성 단원을 참조하십시오.

자격 증명 범위가 유효한 리전으로 지정되어야 합니다

문제: `register` 명령이 다음을 반환합니다.

```
A client error (InvalidSignatureException) occurred when calling the
DescribeStacks operation: Credential should be scoped to a valid region, not 'cn-
north-1'.
```

원인: 명령의 리전은 유효한 AWS OpsWorks Stacks 리전이어야 합니다. 지원되는 리전 목록은 리전 지원 섹션을 참조하세요. 이 오류는 일반적으로 다음 이유 중 하나로 인해 발생할 수 있습니다.

- 스택이 다른 리전에 있고 스택의 리전을 명령의 `--region` 인수에 할당했습니다.

  스택 리전을 지정할 필요가 없습니다. AWS OpsWorks 스택은 스택 ID에서 자동으로 결정합니다.
- `--region` 인수를 생략하여 묵시적으로 기본 리전을 지정했지만 AWS OpsWorks Stacks가 지원하지 않는 기본 리전입니다.

해결 방법: 지원되는 AWS OpsWorks Stacks 리전`--region`으로 명시적으로 설정하거나 `config` 파일을 편집 AWS CLI 하여 기본 리전을 지원되는 AWS OpsWorks Stacks 리전으로 변경합니다. 자세한 내용은 AWS 명령줄 인터페이스 구성 단원을 참조하세요.

AWS OpsWorks Stacks 에이전트 CLI

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

> **Note**
>
> 이 기능은 Linux 인스턴스에서만 사용할 수 있습니다.

AWS OpsWorks Stacks가 모든 인스턴스에 설치하는 에이전트는 명령줄 인터페이스(CLI)를 노출합니다. 인스턴스에 SSH를 사용하여 로그인하는 경우 CLI를 사용하여 다음 작업을 수행할 수 있습니다.

- Chef 실행 로그 파일에 액세스
- Stacks 명령 AWS OpsWorks 에 액세스합니다.
- 수동으로 Chef 레시피 실행
- 인스턴스 보고서 보기
- 에이전트 보고서 보기
- 일부 스택 구성 및 배포 속성 보기

> **Important**
>
> 에이전트 CLI 명령은 루트 사용자이거나 sudo를 사용해야만 실행할 수 있습니다.

기본 명령 구문은 다음과 같습니다.

```
sudo opsworks-agent-cli [--help] [command [activity] [date]]
```

4개 인수는 다음과 같습니다.

help

(선택 사항) 단독으로 사용될 경우 사용 가능한 명령에 대해 간략한 개요를 표시합니다. 명령과 함께 사용될 경우 help는 명령에 대한 설명을 표시합니다.

명령

(선택 사항) 다음 중 하나로 설정되어야 하는 CLI 명령입니다.

- agent_report
- get_json
- instance_report
- list_commands
- run_command
- show_log
- stack_state

activity

(선택 사항) 일부 명령과 함께 인수로 사용되어 특정 AWS OpsWorks Stacks 활동을 지정합니다. `setup`, `configure`, `deploy`, `undeploy`, `start`, `stop` 또는 `restart`

date

(선택 사항) 일부 명령과 함께 인수로 사용되어 특정 AWS OpsWorks Stacks 명령 실행을 지정합니다. date를 명령이 실행된 타임스탬프(*`yyyy-mm-ddThh:mm:ss`* 형식, 작은 따옴표로 묶음)로 지정하여 명령 실행을 지정합니다. 예를 들어 2013년 2월 5일(화요일) 10:31:55는 `'2013-02-05T10:31:55'`를 사용합니다. 특정 AWS OpsWorks Stacks 명령이 실행된 시기를 확인하려면를 실행합니다 list_commands.

> **Note**
>
> 에이전트가 동일한 AWS OpsWorks Stacks 활동을 여러 번 실행한 경우 활동 및 실행 시간을 모두 지정하여 특정 실행을 선택할 수 있습니다. 활동을 지정하고 시간을 생략하면 에이전트 CLI 명령이 해당 활동의 가장 최근 실행에 적용됩니다. 두 인수를 모두 생략하면 에이전트 CLI 명령은 가장 최근 활동에 적용됩니다.

다음 섹션에서는 명령 및 관련 인수를 설명합니다. 간결한 설명을 위해 구문 섹션에서는 모든 명령에서 사용할 수 있는 `--help` 옵션(선택 사항)을 생략했습니다.

주제

- agent_report
- get_json
- instance_report
- list_commands

- run_command
- show_log
- stack_state

agent_report

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

에이전트 보고서를 반환합니다.

```
sudo opsworks-agent-cli agent_report
```

다음 출력 예제는 최근에 구성 활동을 실행한 인스턴스에서 가져온 것입니다.

```
$ sudo opsworks-agent-cli agent_report

AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
  Agent Version: 4004-20151201152533, up to date
```

get_json

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 실행에 대한 정보를 JSON 객체로 반환합니다.

```
sudo opsworks-agent-cli get_json [activity] [date] [-i | --internal | --no-i | --no-internal]
```

기본적으로 `get_json`은 가장 최근의 Chef 실행에 대한 고객 제공 정보를 표시합니다. 특정 정보 세트를 지정하려면 다음 옵션을 사용합니다.

activity

가장 최근에 지정된 활동과 연결된 Chef 실행에 대한 정보를 표시합니다. 유효한 활동의 목록을 가져오려면 list_commands를 실행합니다.

date

지정된 타임스탬프 동안 실행된 활동과 연결된 Chef 실행에 대한 정보를 표시합니다. 유효한 타임스탬프의 목록을 가져오려면 list_commands를 실행합니다.

-i, --internal

AWS OpsWorks Stacks가 Chef 실행에 내부적으로 사용하는 정보를 표시합니다.

--no-i, --no-internal

Chef 실행에 대한 고객 제공 정보를 명시적으로 표시합니다. 달리 지정하지 않은 경우 이 값이 기본값입니다.

> **Note**
>
> Chef 12 Linux 인스턴스의 경우 이 명령을 실행하면 인스턴스의 스택 구성 및 배포 속성과 같은 유효한 정보를 반환합니다. 그러나 자세한 정보를 얻으려면 AWS OpsWorks Stacks가 인스턴스에서 생성하는 Chef 데이터 백을 참조하세요. 자세한 정보는 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하십시오.

다음 출력 예제는 가장 최근의 Chef 실행에서 가장 최근의 구성 활동에 대한 고객 제공 정보를 보여줍니다.

```
$ sudo opsworks-agent-cli get_json configure

{
```

```
  "run_list": [
    "recipe[opsworks_cookbook_demo::configure]"
  ]
}
```

다음 출력 예제는 AWS OpsWorks Stacks가 지정된 타임스탬프에 대해 실행된 Chef 실행에 내부적으로 사용하는 정보를 보여줍니다.

```
$ sudo opsworks-agent-cli get_json 2015-12-01T18:20:24 -i

{
  "aws_opsworks_agent": {
    "version": "4004-20151201152533",
    "valid_client_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
      "sync_remote_users"
    ],
    "command": {
      "type": "configure",
      "args": {
        "app_ids": [

        ]
      },
      "sent_at": "2015-12-01T18:19:23+00:00",
      "command_id": "5c2113f3-c6d5-40eb-bcfa-77da2885eeEX",
      "iam_user_arn": null,
      "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX"
    },
    "resources": {
      "apps": [
```

```
      ],
      "layers": [
        {
          "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
          "name": "MyCookbooksDemoLayer",
          "packages": [

          ],
          "shortname": "cookbooks-demo",
          "type": "custom",
          "volume_configurations": [

          ]
        }
      ],
      "instances": [
        {
          "ami_id": "ami-d93622EX",
          "architecture": "x86_64",
          "auto_scaling_type": null,
          "availability_zone": "us-west-2a",
          "created_at": "2015-11-18T00:21:05+00:00",
          "ebs_optimized": false,
          "ec2_instance_id": "i-a480e960",
          "elastic_ip": null,
          "hostname": "cookbooks-demo1",
          "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
          "instance_type": "c3.large",
          "layer_ids": [
            "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
          ],
          "os": "Amazon Linux 2015.09",
          "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
          "private_ip": "10.122.69.33",
          "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
          "public_ip": "192.0.2.0",
          "root_device_type": "ebs",
          "root_device_volume_id": "vol-f6f7e8EX",
          "ssh_host_dsa_key_fingerprint": "f2:...:15",
          "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
          "ssh_host_rsa_key_fingerprint": "0a:...:96",
          "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
          "status": "online",
          "subnet_id": null,
```

```
          "virtualization_type": "paravirtual",
          "infrastructure_class": "ec2",
          "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----
\nMIIDVwIB...g5OtgQ==\n-----END DSA PRIVATE KEY-----\n",
          "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIB...78kprtIw\n-----END RSA PRIVATE KEY-----\n"
        }
      ],
      "users": [

      ],
      "elastic_load_balancers": [

      ],
      "rds_db_instances": [

      ],
      "stack": {
        "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
        "custom_cookbooks_source": {
          "type": "s3",
          "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
          "username": "AKIAJUQN...WG644EXA",
          "password": "O5v+4Zz+...rcKbFTJu",
          "ssh_key": null,
          "revision": null
        },
        "name": "MyCookbooksDemoStack",
        "region": "us-west-2",
        "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
        "use_custom_cookbooks": true,
        "vpc_id": null
      },
      "ecs_clusters": [

      ],
      "volumes": [

      ]
    },
    "chef": {
      "customer_recipes": [
```

```
        "opsworks_cookbook_demo::configure"
      ],
      "customer_json": "e30=\n",
      "customer_data_bags": "e30=\n"
    }
  }
}
```

instance_report

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

확장된 인스턴스 보고서를 반환합니다.

```
sudo opsworks-agent-cli instance_report
```

다음 출력 예제는 인스턴스에서 가져온 것입니다.

```
$ sudo opsworks-agent-cli instance_report

AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
  Agent Version: 4004-20151201152533, up to date
  OpsWorks Stack: MyCookbooksDemoStack
  OpsWorks Layers: MyCookbooksDemoLayer
  OpsWorks Instance: cookbooks-demo1
  EC2 Instance ID: i-a480e9EX
  EC2 Instance Type: c3.large
  Architecture: x86_64
  Total Memory: 3.84 Gb
  CPU: 2x  Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz

Location:
```

```
  EC2 Region: us-west-2
  EC2 Availability Zone: us-west-2a

Networking:

  Public IP: 192.0.2.0
  Private IP: 198.51.100.0
```

list_commands

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

이 인스턴스에서 실행된 각 활동의 시간을 나열합니다. 다른 에이전트 CLI 명령에 이러한 시간을 사용하여 특정 실행을 지정할 수 있습니다.

```
sudo opsworks-agent-cli list_commands  [activity] [date]
```

다음 출력 예제는 구성, 설정 및 업데이트 사용자 지정 쿡북 활동을 실행한 인스턴스에서 가져온 것입니다.

```
$ sudo opsworks-agent-cli list_commands

2015-11-24T21:00:28        update_custom_cookbooks
2015-12-01T18:19:09        setup
2015-12-01T18:20:24        configure
```

run_command

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks 활동(설정, 구성, 배포 등)을 실행하는 데 필요한 정보가 포함된 Chef 실행 목록이 포함된 JSON 파일인 AWS OpsWorks Stacks 명령을 실행합니다. `run_command` 명령은 `show_log` 를 실행하면 볼 수 있는 로그 항목을 생성합니다. 이 옵션은 개발 목적으로만 사용되므로 AWS OpsWorks Stacks는 변경 사항을 추적하지 않습니다.

```
sudo opsworks-agent-cli run_command [activity] [date] [/path/to/valid/json.file]
```

기본적으로는 최신 AWS OpsWorks Stacks 명령을 `run_command` 실행합니다. 특정 명령을 지정하려면 다음 옵션을 사용합니다.

activity

`setup`, , , `configure`, `deployundeploy`, `start` `stop`또는 지정된 AWS OpsWorks Stacks 명령을 실행합니다`restart`.

date

지정된 타임스탬프에 실행된 AWS OpsWorks 명령을 실행합니다. 유효한 타임스탬프의 목록을 가져오려면 `list_commands`를 실행합니다.

file

지정된 명령 JSON 파일을 실행합니다. 명령의 파일 경로를 가져오려면 `get_json`을 실행합니다.

다음 출력 예제는 인스턴스에서 가져온 것으로, 구성 명령을 실행합니다.

```
$ sudo opsworks-agent-cli run_command configure

[2015-12-02 16:52:53]  INFO [opsworks-agent(21970)]: About to re-run 'configure' from
 2015-12-01T18:20:24
...
[2015-12-02 16:53:02]  INFO [opsworks-agent(21970)]: Finished Chef run with exitcode 0
```

show_log

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

명령의 로그 파일을 반환합니다.

```
sudo opsworks-agent-cli show_log [activity] [date]
```

기본적으로 `show_log`는 가장 최근 로그 파일의 끝부분을 표시합니다. 특정 명령을 지정하려면 다음 옵션을 사용합니다.

activity

지정된 활동의 로그 파일을 표시합니다.

date

지정된 타임스탬프에 실행된 활동의 로그 파일을 표시합니다. 유효한 타임스탬프의 목록을 가져오려면 list_commands를 실행합니다.

다음은 출력 예제는 최신 로그를 보여줍니다.

```
$ sudo opsworks-agent-cli show_log

[2015-12-02T16:52:59+00:00] INFO: Storing updated cookbooks/opsworks_cookbook_demo/
opsworks-cookbook-demo.tar.gz in the cache.
...
[2015-12-02T16:52:59+00:00] INFO: Report handlers complete
```

stack_state

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 가장 최근 Chef 실행에 내부적으로 사용하는 정보를 표시합니다.

```
opsworks-agent-cli stack_state
```

> **Note**
>
> Chef 12 Linux 인스턴스의 경우 이 명령을 실행하면 인스턴스의 스택 구성 및 배포 속성과 같은 유효한 정보를 반환합니다. 그러나 자세한 정보를 얻으려면 AWS OpsWorks Stacks가 인스턴스에서 생성하는 Chef 데이터 백을 참조하세요. 자세한 정보는 AWS OpsWorks Stacks 데이터 백 참조 단원을 참조하십시오.

다음 출력 예제는 인스턴스에서 가져온 것입니다.

```
$ sudo opsworks-agent-cli stack_state

{
  "last_command": {
    "sent_at": "2015-12-01T18:19:23+00:00",
    "activity": "configure"
  },
  "instance": {
    "ami_id": "ami-d93622EX",
    "architecture": "x86_64",
    "auto_scaling_type": null,
    "availability_zone": "us-west-2a",
    "created_at": "2015-11-18T00:21:05+00:00",
    "ebs_optimized": false,
    "ec2_instance_id": "i-a480e9EX",
    "elastic_ip": null,
```

```
    "hostname": "cookbooks-demo1",
    "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
    "instance_type": "c3.large",
    "layer_ids": [
      "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
    ],
    "os": "Amazon Linux 2015.09",
    "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
    "private_ip": "10.122.69.33",
    "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
    "public_ip": "192.0.2.0",
    "root_device_type": "ebs",
    "root_device_volume_id": "vol-f6f7e8EX",
    "ssh_host_dsa_key_fingerprint": "f2:...:15",
    "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
    "ssh_host_rsa_key_fingerprint": "0a:...:96",
    "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
    "status": "online",
    "subnet_id": null,
    "virtualization_type": "paravirtual",
    "infrastructure_class": "ec2",
    "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----\nMIIDVwIB...g5OtgQ==
\n-----END DSA PRIVATE KEY-----\n",
    "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIB...78kprtIw
\n-----END RSA PRIVATE KEY-----\n"
  },
  "layers": [
    {
      "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
      "name": "MyCookbooksDemoLayer",
      "packages": [

      ],
      "shortname": "cookbooks-demo",
      "type": "custom",
      "volume_configurations": [

      ]
    }
  ],
  "applications": null,
  "stack": {
    "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
```

```
    "custom_cookbooks_source": {
      "type": "s3",
      "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
      "username": "AKIAJUQN...WG644EXA",
      "password": "O5v+4Zz+...rcKbFTJu",
      "ssh_key": null,
      "revision": null
    },
    "name": "MyCookbooksDemoStack",
    "region": "us-west-2",
    "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
    "use_custom_cookbooks": true,
    "vpc_id": null
  },
  "agent": {
    "valid_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
      "sync_remote_users"
    ]
  }
}
```

AWS OpsWorks Stacks 데이터 백 참조

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하

는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks는 레시피에 Chef 데이터 백 콘텐츠로 다양한 설정을 노출합니다. 이 참조에는 이 데이터 백 콘텐츠가 나열되어 있습니다.

데이터 백은 Chef 개념의 하나입니다. 데이터 백은 JSON 데이터로 저장되는 인스턴스에 대한 전역 변수로서, 이 JSON 데이터는 Chef에서 액세스할 수 있습니다. 예를 들어 데이터 백은 앱의 소스 URL, 인스턴스의 호스트 이름 및 연결된 스택의 VPC 식별자와 같은 전역 변수를 저장할 수 있습니다. AWS OpsWorks Stacks는 각 스택의 인스턴스에 데이터 백을 저장합니다. Linux 인스턴스에서 AWS OpsWorks Stacks는 `/var/chef/runs/run-ID/data_bags` 디렉터리에 데이터 백을 저장합니다. Windows 인스턴스에서는 데이터 백이 `drive:\chef\runs\run-id\data_bags` 디렉터리에 저장됩니다. 두 경우 모두 *run-ID*는 AWS OpsWorks Stacks가 인스턴스에서 실행되는 각 Chef에 할당하는 고유 ID입니다. 이들 디렉터리에는 데이터 백 세트가 포함됩니다(하위 디렉터리). 각 데이터 백에는 여러 개의 데이터 백 콘텐츠를 포함하는 JSON 형식 파일인 데이터 백 항목이 0개 이상 들어 있습니다.

> **Note**
>
> AWS OpsWorks Stacks는 암호화된 데이터 백을 지원하지 않습니다. 암호나 인증서 등 암호화된 형식의 민감한 데이터를 저장해야 하는 경우, 프라이빗 S3 버킷에 저장하는 것이 좋습니다. 그런 다음 Ruby용 Amazon SDK를 사용하는 사용자 지정 레시피를 생성해 데이터를 검색할 수 있습니다. 예시는 SDK for Ruby 사용에서 확인하십시오.

데이터 백 콘텐츠는 다음 콘텐츠를 포함할 수 있습니다.

- 문자열 콘텐츠 - 표준 Ruby 구문을 따르며 작은따옴표 또는 큰따옴표를 사용할 수 있습니다. 단, 일부 특수 문자를 포함하는 문자열은 반드시 큰따옴표가 있어야 합니다. 자세한 정보는 Ruby 설명서 사이트를 참조하세요.
- 부울 콘텐츠 - `true` 또는 `false`입니다(따옴표 없음).
- 숫자 콘텐츠 - 정수(예: `4`) 또는 소수(예: `2.5`)입니다(따옴표 없음).
- 목록 콘텐츠 - 대괄호로 묶인 쉼표로 구분된 값(예: `[ '80', '443' ]`)입니다(따옴표 없음).
- JSON 객체 - 추가 데이터 백 콘텐츠(예: `"my-app": {"elastic_ip": null,...}`)를 포함합니다.

Chef 레시피는 Chef 검색을 통해 또는 직접 데이터 백, 데이터 백 항목 및 데이터 백 콘텐츠에 액세스할 수 있습니다. 이제부터 두 액세스 접근 방식을 사용하는 방법을 설명합니다(Chef 검색이 선호됨).

Chef 검색을 통해 데이터 백에 액세스하려면 원하는 검색 인덱스를 지정하여 검색 방법을 사용합니다. AWS OpsWorks Stacks는 다음과 같은 검색 인덱스를 제공합니다.

- aws_opsworks_app - 특정 스택의 배포된 앱 세트를 나타냅니다.
- aws_opsworks_command - 특정 스택에서 실행된 명령 세트를 나타냅니다.
- aws_opsworks_ecs_cluster - 특정 스택의 Amazon Elastic Container Service(Amazon ECS) 클러스터 인스턴스 집합을 나타냅니다.
- aws_opsworks_elastic_load_balancer - 특정 스택의 Elastic Load Balancing 로드 밸런서 집합을 나타냅니다.
- aws_opsworks_instance - 특정 스택의 인스턴스 집합을 나타냅니다.
- aws_opsworks_layer - 특정 스택의 계층 집합을 나타냅니다.
- aws_opsworks_rds_db_instance - 특정 스택의 Amazon Relational Database Service(RDS) 인스턴스 집합을 나타냅니다.
- aws_opsworks_stack - 특정 스택을 나타냅니다.
- aws_opsworks_user - 특정 스택의 사용자 집합을 나타냅니다.

검색 인덱스 이름을 알면 해당 검색 인덱스의 데이터 백의 콘텐츠에 액세스할 수 있습니다. 예를 들어 다음 레시피 코드는 `aws_opsworks_app` 검색 인덱스를 사용하여 `aws_opsworks_app` 데이터 백(`aws_opsworks_app` 디렉터리)에서 첫 번째 데이터 백 항목(첫 번째 JSON 파일)의 콘텐츠를 가져옵니다. 그런 다음 코드는 Chef 로그에 메시지 2개를 기록합니다. 한 메시지는 앱의 짧은 이름 데이터 백 콘텐츠(JSON 파일 내 문자열)를 포함하고 다른 메시지는 앱의 소스 URL 데이터 백 콘텐츠(JSON 파일 내 다른 문자열)를 포함합니다.

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

여기서 `['shortname']` 및 `['app_source']['url']`은 해당 JSON 파일에서 다음 데이터 백 콘텐츠를 지정합니다.

```
{
  ...
```

```
  "shortname": "mylinuxdemoapp",
  ...
  "app_source": {
    ...
    "url": "https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-
nodejs.tar.gz",
  },
  ...
}
```

검색 가능한 데이터 백 콘텐츠의 목록은 이 섹션의 참조 항목 단원을 참조하세요.

또한 한 데이터 백 안의 여러 데이터 백 항목에 대해 반복할 수 있습니다. 예를 들어 다음 레시피 코드는 이전 예제와 비슷합니다. 이 코드는 데이터 백 항목이 여러 개일 경우 데이터 백의 각 데이터 백 항목을 반복합니다.

```
search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

특정 데이터 백 콘텐츠의 존재를 알고 있다면 다음 구문을 사용하여 해당 데이터 백 항목을 찾을 수 있습니다.

```
search("search_index", "key:value").first
```

예를 들어 다음 레시피 코드는 aws_opsworks_app 검색 인덱스를 사용하여 mylinuxdemoapp의 짧은 이름을 포함하는 데이터 백 항목을 찾습니다. 그런 다음 데이터 백 항목의 콘텐츠를 사용하여 해당 앱의 짧은 이름 및 소스 URL을 포함하는 메시지를 Chef 로그에 기록합니다.

```
app = search("aws_opsworks_app", "shortname:mylinuxdemoapp").first
Chef::Log.info("********** For the app with the short name '#{app['shortname']}', the
 app's URL is '#{app['app_source']['url']}' **********")
```

aws_opsworks_instance 검색 인덱스에 한해, self:true를 지정하여 현재 레시피가 실행되고 있는 인스턴스를 나타낼 수 있습니다. 다음 레시피 코드는 해당 데이터 백 항목의 콘텐츠를 사용하여 해당 인스턴스의 AWS OpsWorks Stacks 생성 ID 및 운영 체제와 함께 Chef 로그에 메시지를 작성합니다.

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 operating system is '#{instance['os']}' **********")
```

Chef 검색을 사용하여 데이터 백, 데이터 백 항목 및 데이터 백 콘텐츠에 액세스하는 대신, 직접 이들에 액세스할 수 있습니다. 이렇게 하려면 data_bag 및 data_bag_item 메서드를 사용하여 각각 데이터 백 및 데이터 백 항목에 액세스합니다. 예를 들어 다음 레시피 코드는 앞서의 예제와 동일한 작업을 수행하지만, 데이터 백 항목이 여러 개일 경우 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목에 직접 액세스하는 것이 다릅니다.

```
# Syntax: data_bag_item("the data bag name", "the file name in the data bag without the
 file extension")
app = data_bag_item("aws_opsworks_app", "mylinuxdemoapp")
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

data_bag("aws_opsworks_app").each do |data_bag_item|
  app = data_bag_item("aws_opsworks_app", data_bag_item)
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

두 접근 방식 중 Chef 검색을 사용할 것을 권장합니다. 이 설명서의 관련 예제에서는 모두 이 접근 방식을 예시합니다.

주제

- 앱 데이터 백(aws_opsworks_app)
- 명령 데이터 백(aws_opsworks_command)
- Amazon 클러스터 데이터 백(aws_opsworks_ecs_cluster)
- Elastic Load Balancing 데이터 백(aws_opsworks_elastic_load_balancer)
- 인스턴스 데이터 백(aws_opsworks_instance)
- 계층 데이터 백(aws_opsworks_layer)
- Amazon RDS 데이터 백(aws_opsworks_rds_db_instance)
- 스택 데이터 백(aws_opsworks_stack)
- 사용자 데이터 백(aws_opsworks_user)

앱 데이터 백(aws_opsworks_app)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Deploy 이벤트 또는 레시피 실행 스택 명령의 경우, 앱의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 앱의 짧은 이름과 소스 URL을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

app_id	app_source	data_sources
배포	attributes	domains
enable_ssl	환경	name
shortname	ssl_configuration	type

app_id

앱 ID(문자열). 앱을 식별하는 GUID입니다.

app_source

Stacks가 소스 제어 AWS OpsWorks 리포지토리에서 앱을 배포하는 데 사용하는 정보를 지정하는 콘텐츠 세트입니다. 콘텐츠는 리포지토리 유형에 따라 다릅니다.

비밀번호

프라이빗 리포지토리의 경우에는 암호, 퍼블릭 리포지토리의 경우 "null"(문자열). 프라이빗 S3 버킷의 경우, 이 콘텐츠가 암호 키로 설정됩니다.

개정

리포지토리에 여러 브랜치가 있는 경우, 이 콘텐츠는 "version1"과 같이 앱의 브랜치 또는 버전을 지정합니다(문자열). 그렇지 않으면 "null"로 설정됩니다.

ssh_key

프라이빗 Git 리포지토리에 액세스하는 경우에는 배포 SSH 키, 퍼블릭 리포지토리의 경우에는 "null"(문자열).

type

앱의 소스 위치(문자열). 유효한 값으로는 다음이 포함됩니다.

- "archive"
- "git"
- "other"
- "s3"

url

앱 소스의 위치(문자열).

사용자

프라이빗 리포지토리의 경우에는 사용자 이름, 퍼블릭 리포지토리의 경우 "null"(문자열). 프라이빗 S3 버킷의 경우, 이 콘텐츠는 액세스 키로 설정됩니다.

attributes

앱의 디렉터리 구조 및 콘텐츠를 설명하는 콘텐츠 세트.

document_root

문서 트리의 루트 디렉터리. 배포 디렉터리를 기준으로 문서 루트의 경로 또는 앱 홈 페이지의 위치(예: home_html)를 정의합니다. 이 속성이 지정되지 않으면 document_root는 기본적으로

public입니다. document_root의 값은 a-z, A-Z, 0-9, _(밑줄) 또는 -(하이픈) 문자로만 시작할 수 있습니다.

data_sources

앱의 데이터베이스에 연결하는 데 필요한 정보. 앱에 연결된 데이터베이스 계층이 있는 경우 AWS OpsWorks Stacks는이 콘텐츠에 적절한 값을 자동으로 할당합니다.

data_sources의 값은 어레이이고, 어레이는 키가 아니라 내장 오프셋에 의해 액세스됩니다. 예를 들어 앱의 첫 번째 데이터 원본에 액세스하려면 app[:data_sources][0][:type]을 사용합니다.

database_name

일반적으로 앱의 짧은 이름인 데이터베이스 이름(문자열).

type

일반적으로 "RdsDbInstance"인 데이터베이스 인스턴스의 유형(문자열).

arn

데이터베이스 인스턴스의 Amazon 리소스 이름(ARN)(문자열).

배포

앱을 배포할지 여부(부울). Deploy 수명 주기 이벤트에서 배포할 앱의 경우 true입니다. 설정 수명 주기 이벤트에서는 이 콘텐츠가 모든 앱에 대해 true입니다. 인스턴스에 배포될 앱을 결정하려면 인스턴스가 속한 계층에 확인 표시를 합니다.

domains

앱의 도메인 목록(문자열의 목록).

enable_ssl

SSL 지원이 활성화되는지 여부(부울).

환경

앱에 대해 정의된 사용자 지정 환경 변수의 모음. 앱의 환경 변수를 정의하는 방법에 대한 자세한 정보는 앱 추가 단원을 참조하세요. 각 콘텐츠 이름은 환경 변수 이름으로 설정되고 해당 값은 변수의 값으로 설정됩니다.

name

표시용으로 사용되는 앱 이름(문자열).

shortname

이름(문자열)에서 AWS OpsWorks Stacks에 의해 생성되는 앱의 짧은 이름입니다. 짧은 이름은 레시피 내부에서 사용되며, 앱 파일이 설치되는 디렉터리의 이름으로 사용됩니다.

ssl_configuration

인증서

SSL 지원을 활성화한 경우에는 앱의 SSL 인증서, 그렇지 않은 경우에는 "null"(문자열).

chain

SSL이 활성화된 경우, 중간 인증서 발급 기관 공개 키 또는 클라이언트 인증을 지정하기 위한 콘텐츠(문자열).

private_key

SSL 지원을 활성화한 경우에는 앱의 SSL 프라이빗 키, 그렇지 않은 경우에는 "null"(문자열).

type

앱의 유형(문자열). Chef 12 Linux 및 Chef 12.2 Windows 스택의 경우 항상 "other"로 설정됩니다.

명령 데이터 백(aws_opsworks_command)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

AWS OpsWorks Stacks가 하나 이상의 인스턴스에서 실행하는 명령의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 명령의 유형과 전송 시점을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
command = search("aws_opsworks_command").first
Chef::Log.info("********** The command's type is '#{command['type']}' **********")
Chef::Log.info("********** The command was sent at '#{command['sent_at']}' **********")
```

```
search("aws_opsworks_command").each do |command|
  Chef::Log.info("********** The command's type is '#{command['type']}' **********")
  Chef::Log.info("********** The command was sent at '#{command['sent_at']}'
 **********")
end
```

args	command_id	iam_user_arn
instance-id	sent_at	type

args

명령의 인수(문자열).

command_id

AWS OpsWorks Stacks(문자열)에서 할당한 명령의 임의 고유 식별자입니다.

iam_user_arn

명령이 고객에 의해 생성되는 경우, 명령을 생성한 사용자의 Amazon 리소스 이름(ARN)(문자열).

instance-id

명령이 실행되는 인스턴스의 식별자(문자열).

sent_at

AWS OpsWorks Stacks가 명령을 실행한 시점의 타임스탬프(문자열).

type

명령 유형(문자열). 유효한 값으로는 다음이 포함됩니다.

- `"configure"`
- `"deploy"`
- `"deregister"`
- `"execute_recipes"`
- `"grant_remote_access"`
- `"install_dependencies"`
- `"restart"`
- `"revoke_remote_access"`

- "rollback"
- "setup"
- "shutdown"
- "start"
- "stop"
- "sync_remote_users"
- "undeploy"
- "update_agent"
- "update_custom_cookbooks"
- "update_dependencies"

Amazon 클러스터 데이터 백(aws_opsworks_ecs_cluster)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Amazon ECS 클러스터 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 Amazon ECS 클러스터의 이름과 Amazon 리소스 이름(ARN)을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
ecs_cluster = search("aws_opsworks_ecs_cluster").first
Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
Chef::Log.info("********** The ECS cluster's ARN is '#{ecs_cluster['ecs_cluster_arn']}'
 **********")

search("aws_opsworks_ecs_cluster").each do |ecs_cluster|
  Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
```

```
  Chef::Log.info("********** The ECS cluster's ARN is
 '#{ecs_cluster['ecs_cluster_arn']}' **********")
end
```

ecs_cluster_arn	ecs_cluster_name

ecs_cluster_arn

클러스터의 Amazon 리소스 이름(ARN)(문자열).

ecs_cluster_name

클러스터의 이름(문자열).

Elastic Load Balancing 데이터 백(aws_opsworks_elastic_load_balancer)

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Elastic Load Balancing 로드 밸런서의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 Elastic Load Balancing 로드 밸런서의 이름과 DNS 이름을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
elastic_load_balancer = search("aws_opsworks_elastic_load_balancer").first
Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
Chef::Log.info("********** The ELB's DNS name is '#{elastic_load_balancer['dns_name']}'
 **********")

search("aws_opsworks_elastic_load_balancer").each do |elastic_load_balancer|
  Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
```

```
  Chef::Log.info("********** The ELB's DNS name is
 '#{elastic_load_balancer['dns_name']}' **********")
end
```

elastic_load_balancer_name	dns_name	layer_id

elastic_load_balancer_name

로드 밸런서의 이름(문자열).

dns_name

로드 밸런서의 DNS 이름(문자열).

layer_id

로드 AWS OpsWorks 밸런서가 할당된 계층의 스택 ID(문자열).

인스턴스 데이터 백(aws_opsworks_instance)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

인스턴스의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 인스턴스의 호스트 이름과 ID를 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
instance = search("aws_opsworks_instance").first
Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")

search("aws_opsworks_instance").each do |instance|
```

```
  Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
  Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")
end
```

다음 예제는 다중 데이터 백 항목을 검색하여 지정된 Amazon EC2 인스턴스 ID를 포함하는 데이터 백 항목을 찾기 위해 Chef 검색을 사용하는 다양한 방법을 보여줍니다. 그런 다음 이 예제는 데이터 백 항목의 콘텐츠를 사용하여 해당 인스턴스의 퍼블릭 IP 주소를 포함하는 메시지를 Chef 로그에 기록합니다.

```
instance = search("aws_opsworks_instance", "ec2_instance_id:i-12345678").first
Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")

search("aws_opsworks_instance").each do |instance|
  if instance['ec2_instance_id'] == 'i-12345678'
    Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")
  end
end
```

다음 예제는 `self:true`로 설정된 Chef 검색을 사용하여 레시피가 실행되는 인스턴스와 관련된 정보를 포함하는 데이터 항목을 찾는 방법을 보여줍니다. 그런 다음이 예제에서는 데이터 백 항목의 콘텐츠를 사용하여 해당 인스턴스의 AWS OpsWorks Stacks 생성 ID와 인스턴스의 퍼블릭 IP 주소를 사용하여 Chef 로그에 메시지를 작성합니다.

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 public IP address is '#{instance['public_ip']}' **********")
```

ami_id	architecture	auto_scaling_type
availability_zone	created_at	ebs_optimized
ec2_instance_id	elastic_ip	hostname
instance-id	instance_type	layer_ids
os	private_dns	private_ip

public_dns	public_ip	root_device_type
root_device_volume_id	self	ssh_host_dsa_key_fingerprint
ssh_host_dsa_key_private	ssh_host_dsa_key_public	ssh_host_rsa_key_fingerprint
ssh_host_rsa_key_private	ssh_host_rsa_key_public	status
subnet_id	virtualization_type	

ami_id

인스턴스의 AMI(Amazon 머신 이미지) ID(문자열).

architecture

인스턴스의 아키텍처(문자열). 항상 "x86_64"로 설정됩니다.

auto_scaling_type

인스턴스의 조정 유형(문자열): null, timer 또는 load.

availability_zone

인스턴스의 가용 영역(AZ)(문자열), 예: "us-west-2a".

created_at

인스턴스가 생성된 UTC "*yyyy*-*mm*-*ddd*T*hh*:*mm*:*ss*+*hh*:*mm*" 형식 시간(문자열). 예를 들어 "2013-10-01T08:35:22+00:00"은 2013년 10월 10일 8:35:22에 해당합니다(시간대 오프셋 없음). 자세한 정보는 ISO 8601를 참조하세요.

ebs_optimized

인스턴스가 EBS 최적화되었는지 여부(부울).

ec2_instance_id

EC2 인스턴스 ID(문자열).

elastic_ip

탄력적 IP 주소(문자열). 인스턴스에 탄력적 IP 주소가 없으면 "null"로 설정됩니다.

hostname

호스트 이름(문자열), 예: "demo1".

instance-id

인스턴스를 고유하게 식별하는 AWS OpsWorks Stacks 생성 GUID인 인스턴스 ID(문자열).

instance_type

인스턴스 유형(문자열), 예: "c1.medium".

layer_ids

고유 ID(예: 307ut64c-c7e4-40cc-52f0-67d5k1f9992c)로 식별되는 인스턴스 내 계층의 목록.

os

인스턴스의 운영 체제(문자열). 유효한 값으로는 다음이 포함됩니다.

- "Amazon Linux 2"
- "Amazon Linux 2018.03"
- "Amazon Linux 2017.09"
- "Amazon Linux 2017.03"
- "Amazon Linux 2016.09"
- "Custom"
- "Microsoft Windows Server 2022 Base"
- "Microsoft Windows Server 2022 with SQL Server Express"
- "Microsoft Windows Server 2022 with SQL Server Standard"
- "Microsoft Windows Server 2022 with SQL Server Web"
- "Microsoft Windows Server 2019 Base"
- "Microsoft Windows Server 2019 with SQL Server Express"
- "Microsoft Windows Server 2019 with SQL Server Standard"
- "Microsoft Windows Server 2019 with SQL Server Web"
- "CentOS 7"
- "Red Hat Enterprise Linux 7"
- "Ubuntu 20.04 LTS"
- "Ubuntu 18.04 LTS"
- "Ubuntu 16.04 LTS"
- "Ubuntu 14.04 LTS"

private_dns

프라이빗 DNS 이름(문자열).

private_ip

프라이빗 IP 주소(문자열).

public_dns

퍼블릭 DNS 이름(문자열).

public_ip

퍼블릭 IP 주소(문자열).

root_device_type

루트 디바이스 유형(문자열). 유효한 값으로는 다음이 포함됩니다.

- `"ebs`
- `"instance-store"`

root_device_volume_id

루트 디바이스의 볼륨 ID(문자열).

self

이 데이터 백 항목에 레시피가 실행되는 인스턴스에 대한 정보가 포함되면 `true`, 그렇지 않으면 `false`(부울). 이 값은 AWS OpsWorks Stacks API를 통하지 않고 레시피에서만 사용할 수 있습니다.

ssh_host_dsa_key_fingerprint

더 긴 DSA 퍼블릭 키를 식별하는 더 짧은 바이트 시퀀스(문자열).

ssh_host_dsa_key_private

인스턴스에 대한 SSH 인증용 DSA 생성 프라이빗 키(문자열).

ssh_host_dsa_key_public

인스턴스에 대한 SSH 인증용 DSA 생성 퍼블릭 키(문자열).

ssh_host_rsa_key_fingerprint

더 긴 RSA 퍼블릭 키를 식별하는 더 짧은 바이트 시퀀스(문자열).

ssh_host_rsa_key_private

인스턴스에 대한 SSH 인증용 RSA 생성 프라이빗 키(문자열).

ssh_host_rsa_key_public

인스턴스에 대한 SSH 인증용 RSA 생성 퍼블릭 키(문자열).

status

인스턴스의 상태(문자열). 유효한 값으로는 다음이 포함됩니다.

- `"requested"`
- `"booting"`
- `"running_setup"`
- `"online"`
- `"setup_failed"`
- `"start_failed"`
- `"terminating"`
- `"terminated"`
- `"stopped"`
- `"connection_lost"`

subnet_id

인스턴스의 서브넷 ID(문자열).

virtualization_type

인스턴스의 가상화 유형(문자열).

계층 데이터 백(aws_opsworks_layer)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

계층의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 계층의 이름과 짧은 이름을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
layer = search("aws_opsworks_layer").first
Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")

search("aws_opsworks_layer").each do |layer|
  Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
  Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")
end
```

ecs_cluster_arn	layer_id	name
packages	shortname	type
volume_configurations		

ecs_cluster_arn

계층에 Amazon ECS 클러스터가 할당된 경우 Amazon ECS 클러스터의 Amazon 리소스 이름(ARN)(문자열).

encrypted

EBS 볼륨이 암호화된 경우 `true`이고, 그렇지 않으면 `false`(부울)입니다.

layer_id

계층 ID는 AWS OpsWorks Stacks에서 생성하고 계층(문자열)을 고유하게 식별하는 GUID입니다.

name

콘솔에서 계층을 나타내는 데 사용되는 계층의 이름(문자열). 사용자 정의 이름일 수 있으며 반드시 고유할 필요는 없습니다.

packages

설치할 패키지의 목록(문자열의 목록).

shortname

계층의 사용자 정의 짧은 이름(문자열).

type

계층의 유형(문자열). Chef 12 Linux 및 Chef 12.2 Windows의 경우 항상 `"custom"`으로 설정됩니다.

volume_configurations

Amazon EBS 볼륨 구성 목록입니다.

iops

볼륨이 지원할 수 있는 초당 I/O 작업 수.

mount_point

볼륨의 탑재 지점 디렉터리.

number_of_disks

볼륨에 있는 디스크의 수입니다.

raid_level

볼륨의 RAID 구성 수준.

size

GiB 단위의 볼륨 크기.

volume_type

볼륨의 유형: 범용, 마그네틱, 프로비저닝된 IOPS, 처리량에 최적화된 HDD 또는 콜드 HDD입니다.

Amazon RDS 데이터 백(aws_opsworks_rds_db_instance)

Important

이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

다음과 같이 Amazon Relational Database Service(RDS) 인스턴스의 구성을 지정하는 데이터 백 콘텐츠 세트.

address	db_instance_identifier	db_password
db_user	engine	rds_db_instance_arn
리전		

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 Amazon RDS 인스턴스의 주소와 데이터베이스 엔진 유형을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
rds_db_instance = search("aws_opsworks_rds_db_instance").first
Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")

search("aws_opsworks_rds_db_instance").each do |rds_db_instance|
  Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
  Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")
end
```

address

인스턴스의 DNS 이름.

포트

인스턴스의 포트.

db_instance_identifier

인스턴스의 ID.

db_password

인스턴스의 마스터 암호.

db_user

인스턴스의 마스터 사용자 이름.

engine

인스턴스의 데이터베이스 엔진, 예: `mysql`.

rds_db_instance_arn

인스턴스의 Amazon 리소스 이름(ARN).

리전

인스턴스의 AWS 리전, 예: `us-west-2`.

스택 데이터 백(aws_opsworks_stack)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

스택의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 스택의 이름과 쿡북의 소스 URL을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** The stack's name is '#{stack['name']}' **********")
Chef::Log.info("********** The stack's cookbook URL is
 '#{stack['custom_cookbooks_source']['url']}' **********")
```

arn	custom_cookbooks_source	name
리전	stack_id	use_custom_cookbooks
vpc_id		

arn

스택의 Amazon 리소스 이름(ARN)(문자열).

custom_cookbooks_source

사용자 지정 쿡북의 소스 리포지토리를 지정하는 콘텐츠 세트.

type

리포지토리 유형(문자열). 유효한 값으로는 다음이 포함됩니다.

- `"archive"`
- `"git"`
- `"s3"`

url

`"git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git"`와 같은 리포지토리 URL(문자열).

사용자 이름

프라이빗 리포지토리의 경우에는 사용자 이름, 퍼블릭 리포지토리의 경우 `null`(문자열). 프라이빗 Amazon Simple Storage Service(Amazon S3) 버킷의 경우 콘텐츠가 액세스 키로 설정됩니다.

비밀번호

프라이빗 리포지토리의 경우에는 암호, 퍼블릭 리포지토리의 경우 `null`(문자열). 프라이빗 S3 버킷의 경우, 이 콘텐츠가 암호 키로 설정됩니다.

ssh_key

프라이빗 Git 리포지토리에 액세스하는 경우에는 배포 SSH 키, 퍼블릭 리포지토리의 경우에는 `null`(문자열).

개정

리포지토리에 여러 브랜치가 있는 경우, 이 콘텐츠는 `"version1"`과 같이 앱의 브랜치 또는 버전을 지정합니다(문자열). 그렇지 않으면 `null`로 설정됩니다.

name

스택의 이름(문자열).

리전

스택의 AWS 리전(문자열).

stack_id

스택을 식별하는 GUID(문자열).

use_custom_cookbooks

사용자 지정 쿡북이 활성화되었는지 여부(부울).

vpc_id

스택이 VPC에서 실행 중인 경우, VPC ID(문자열).

사용자 데이터 백(aws_opsworks_user)

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

사용자의 설정을 나타냅니다.

다음 예제는 Chef 검색을 사용하여 단일 데이터 백 항목, 그런 다음 다중 데이터 백 항목을 검색하여 사용자의 사용자 이름과 Amazon 리소스 이름(ARN)을 포함하는 메시지를 Chef 로그에 기록하는 방법을 보여줍니다.

```
user = search("aws_opsworks_user").first
Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")

# Or...

search("aws_opsworks_user").each do |user|
  Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
  Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")
end
```

administrator_privileges	iam_user_arn	remote_access
ssh_public_key	unix_user_id	사용자 이름

administrator_privileges

사용자에게 관리자 권한이 있는지 여부(부울).

iam_user_arn

사용자의 Amazon 리소스 이름(ARN)(문자열).

remote_access

사용자가 RDP를 사용하여 인스턴스에 로그인할 수 있는지 여부(부울).

ssh_public_key

AWS OpsWorks Stacks 콘솔 또는 API(문자열)를 통해 제공되는 사용자의 퍼블릭 키입니다.

unix_user_id

사용자의 Unix ID(숫자).

사용자 이름

사용자 이름(문자열).

OpsWorks 에이전트 변경 사항

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

Chef 12 에이전트 릴리스

다음 표에서는 AWS OpsWorks Stacks이 관리하는 인스턴스에 설치하는 Chef 12 에이전트의 중요 변경 사항을 설명합니다.

에이전트 버전	설명	릴리스 날짜
4042	• 이 에이전트 릴리스에는 새로운 기능 없이 사소한 변경 사항만 포함되어 있습니다.	2023년 2월 7일
4041	• 이 에이전트 릴리스에는 새로운 기능 없이 사소한 변경 사항만 포함되어 있습니다. • Amazon CA 인증서 업데이트	2023년 1월 27일
4040	• 이 에이전트 릴리스에는 새로운 기능 없이 사소한 변경 사항만 포함되어 있습니다.	2022년 7월 22일
4039	• Ubuntu AMI를 위한 ECS 통합 수정	2020년 4월 30일
4038	• DST 변경 중 인스턴스 통계 전송 시 버그 수정 • 에이전트 다운로드 및 설치 중 `no_proxy` 환경 변수 준수	2020년 3월 5일
4037	• SigV4를 사용하여 리전 없이 S3 URL 요청에 서명하는 지원 추가 • SigV2를 사용하여 S3 요청에 서명하는 지원 제거	2019년 6월 4일
4035	• ECS 설정 중 버그 수정 • 인스턴스 유형 변경 후 중복되는 `fstab` 항목 수정	2019년 5월 8일
4033	• Ubuntu 18.04에 대한 지원 추가 • Amazon Linux 2에서 에이전트 설치 버그 수정	2018년 11월 26일
4032	• Amazon Linux 2에 대한 지원 추가	2018년 10월 24일
4031	• Amazon Linux 2018.03에 대한 지원 추가 • 다른 계정에 호스트된 퍼블릭 S3 아카이브 지원	2018년 8월 15일
4030	• c5d 인스턴스 처리를 위한 볼륨 수정	2018년 5월 31일
4029	• Ubuntu 14.04에 `nvme-cli`를 설치합니다. • c5, m5 인스턴스 탑재를 위한 볼륨 수정 • 재시작 시 호스트 이름을 항상 보존합니다	2018년 5월 2일

에이전트 버전	설명	릴리스 날짜
4028	• CentOS용 `monit` 구성 수정	2018년 3월 20일
4027	• Ubuntu 14.04에 NVMe 볼륨 탑재 지원(`nvme-cli`는 수동으로 설치되어야 합니다) • 볼륨을 위한 `name` 속성이 필요하지 않습니다	2018년 2월 17일
4026	• EBS 볼륨 ID를 사용하여 NVMe 기반 EBS 볼륨을 탑재합니다 • i3 인스턴스 탑재를 위한 EBS 볼륨 수정 • c5, m5 인스턴스에 탑재된 EBS 볼륨 순서 수정	2018년 1월 31일
4025	• NVMe 디바이스의 처리 수정	2017년 12월 13일
4024	• Amazon Linux 2017.09에 대한 지원 추가	2017년 12월 5일
4023	• CloudWatch Logs 통합에 대한 지원 추가	2017년 2월 4일
4022	• Chef 클라이언트 버전을 12.18.31로 업데이트	2017년 2월 1일
4021	• 프록시 처리 개선	2016년 12월 16일
4020	• Chef 클라이언트 버전을 12.16.42로 업데이트	2016년 12월 8일
4019	• 에이전트 설치 도중 원본 프록시 변수 • Red Hat Enterprise Linux 7은 이제 `systemd` 대신 `monit`를 사용합니다 • Red Hat Enterprise Linux 7에서 EPEL을 설정하지 마십시오 • 프로세스 잠금에 `flock` 대신 `lockrun.c` 을 사용하세요 • `ps -p1`를 확인할 때 `systemd` 홀수 출력을 피하세요	2016년 10월 19일
4018	• Chef 클라이언트 버전을 12.13.37로 업데이트 • Amazon Linux 2016.09에 대한 지원 추가	2016년 8월 25일

에이전트 버전	설명	릴리스 날짜
4017	• Chef 클라이언트 버전을 12.12.15로 업데이트	2016년 8월 10일
4016	• `monit`가 사용되지 않는 시스템에서 에이전트 설치 제거 수정	2016년 6월 23일
4015	• Amazon Linux 2016.03의 ECS 설정 수정	2016년 6월 17일
4011	• Chef 클라이언트 버전을 12.10.24로 업데이트 • 로그 업로드 처리 개선	2016년 5월 19일
4008	• Amazon Linux 2016.03에 대한 지원 추가 • 번들 설치에 제한 시간 추가 • 존재하는 경우 /etc/filesystems에 xfs 추가	2016년 3월 16일
4007	• Chef 클라이언트 버전을 12.7.2로 업데이트 • 온프레미스 인스턴스(AWS 외부에서 호스팅되는 서버)의 오류 처리 개선 • 최신 chef-sugar와의 호환성 개선 • 배포용 아카이브 다운로드 재시도	2016년 3월 4일
4006	• Chef 클라이언트 버전을 12.6.0으로 업데이트 • 에이전트 설치 시 libxml2-devel/libxml2-dev 및 libxslt-devel/libxslt-dev 패키지를 설치하지 마십시오	2016년 1월 21일
4005	• ec2 인프라의 경우, ohai에서 항상 ec2 데이터를 활성화하여 ec2 가져오기 수정	2015년 12월 17일
4004	• AWS OpsWorks Chef 12 Linux- Chef Client 12.5.1에 대한 Stacks 지원	2015년 12월 3일

Chef 11.10 에이전트 릴리스

다음 표에서는 AWS OpsWorks Stacks이 관리하는 인스턴스에 설치하는 Chef 11.10 에이전트의 중요 변경 사항을 설명합니다.

에이전트 버전	설명	릴리스 날짜
3456	• 이 에이전트 릴리스에는 새로운 기능 없이 사소한 변경 사항만 포함되어 있습니다. • Amazon CA 인증서 업데이트	2023년 1월 27일
3455	• 이 에이전트 릴리스에는 새로운 기능 없이 사소한 변경 사항만 포함되어 있습니다.	2022년 11월 1일
3454	• Ubuntu AMI를 위한 ECS 통합 수정	2020년 4월 28일
3453	• DST 변경 중 인스턴스 통계 전송 시 버그 수정 • RHEL7 설정에서 누락된 패키지 버그 수정 • 에이전트 다운로드 및 설치 중 `no_proxy` 환경 변수 준수	2020년 3월 5일
3452	• `us-east-1` 인 경우 Amazon S3 가상 경로 URL에 리전을 포함하지 않음 • 내부 쿡북을 추출하여 스테이지 리전 특정 버킷에 업로드 • Chef 11.10에 대한 `fstab` 항목 수정 • S3에 대한 SigV2 사용을 제거하고 요청에서 버킷의 리전 가져오기	2019년 8월 13일
3451	• Ruby 2.6.1에 대한 지원 추가	2019년 3월 20일
3450	• 기본 EBS 속성 수정 • Amazon Linux 2에 대한 CloudWatchLogs 에이전트 설치 수정 • 2.6.14보다 최신인 rubygem 버전에 대한 번들러 설치 수정 • 퍼블릭 S3 아카이브 지원 수정	2018년 12월 3일
3449	• c5d 인스턴스 처리를 위한 볼륨 수정 • NVMe 디바이스 인스턴스에 RAID 배열 지원 수정	2018년 6월 5일
3448	• Ruby의 기본 2.3 버전을 2.3.7로 업그레이드합니다.	2018년 5월 8일

에이전트 버전	설명	릴리스 날짜
	• Ubuntu 14.04 인스턴스에 NVMe 기반 인스턴스 EBS 볼륨 탑재 수정 • 다른 계정에 호스트된 퍼블릭 Amazon S3 아카이브 지원 • Red Hat Enterprise Linux 인스턴스에 `opsworks-agent` 부트 문제 해결	
3447	• EBS 볼륨 ID를 사용하여 NVMe 기반 EBS 볼륨을 탑재합니다 • i3 인스턴스 탑재를 위한 EBS 볼륨 수정 • c5, m5에 탑재된 EBS 볼륨 순서 수정 • Ruby의 기본 2.3 버전을 2.3.6로 업데이트합니다.	2018년 1월 31일
3446	• NVMe 디바이스의 처리 수정 • Ruby의 기본 2.3 버전을 2.3.5로 업데이트합니다.	2017년 12월 14일
3445	• Amazon Linux 2017.09에 대한 지원 추가 • Ruby의 기본 2.2 버전을 2.2.8로 업데이트합니다.	2017년 10월 31일
3444	• CloudWatch Logs에 대한 지원 추가	2017년 4월 1일
3443	• 프록시 처리 개선	2016년 12월 15일
3442	• Ruby의 기본 2.3 버전을 2.3.3로 업데이트합니다. • Ruby의 기본 2.2 버전을 2.2.6로 업데이트합니다.	2016년 6월 12일
3441	• 에이전트 설치 도중 원본 프록시 변수	2016년 10월 21일
3440	• Amazon Linux 2016.09에 대한 지원 추가	2016년 9월 13일
3439	• 새로운 기능이 없는 작은 변화	2016년 7월 29일

에이전트 버전	설명	릴리스 날짜
3438	• Ruby 2.3.1 지원 추가 • IAM 인스턴스 프로파일에서 자격 증명으로 인스턴스 등록을 향상합니다 • `s3curl.pl` 나머지 제거 • Amazon Linux 2016.03의 ECS 설정 수정	2016년 6월 17일
3437	• Ruby의 기본 2.2 버전을 2.2.5로 업데이트합니다.	2016년 4월 5일
3436	• Red Hat Enterprise Linux용 EPEL URL 업데이트. 중요: 이 변경 없이는 Red Hat Enterprise Linux 인스턴스는 부트 실행에 실패합니다.	2016년 4월 18일
3435	• Ruby의 기본 2.1 버전을 2.1.9로 업데이트합니다. • Amazon S3 및 아카이브 배포 처리 향상	2016년 6월 4일
3434	• Amazon Linux 2016.03에 대한 지원 추가 • 패키지 설치 재시도	2016년 3월 16일
3433	• 온프레미스 인스턴스(외부에서 호스팅되는 서버 AWS)에 대한 몇 가지 개선 사항 • 최신 `chef-sugar` 와의 호환성 개선 • 배포용 아카이브 다운로드 재시도 • Ruby gems 설치 URL 수정	2016년 2월 27일
3432	• 버킷 이름 중 특수 문자 처리 향상 • `s3_file`을 버전 2.6.6으로 업데이트 • 지정된 탑재 지점 없이 볼륨 탑재 건너뛰기 • 중지 대신 항상 `unicorn`을 재시작하고 배포 중 가동 중지를 방지합니다 • `setup` 명령에 대해서는 항상 사용자 지정 쿡북을 업데이트합니다 • RAID 배열을 생성한 후 `initramfs` 를 업데이트하여 재부팅 시 디바이스 매핑 문제를 방지합니다	2016년 1월 20일

에이전트 버전	설명	릴리스 날짜
3431	• Rails 계층에서 해결된 `passenger` 및 `unicorn` gem 설치 문제 • Ruby 기본 2.0, 2.1 및 2.2 버전을 2.0.0p648, 2.1.8 및 2.2.4로 업데이트 중 • `postgres` 패키지 이름을 사용자 지정 JSON으로 설정할 수 있도록 허용 • Node.js 기본 버전을 0.12.9로 업데이트합니다.	2015년 12월 22일
3430	• 새로운 기능이 없는 작은 변화	2015년 11월 25일
3429	• OpsWorks 에이전트 데몬화 개선(stdout/stderr 닫기) • `s3_file` 리소스의 견고함 향상(재시도 후 예외 발견)	2015년 11월 18일
3428	• Gemfile 기반으로 `postgres` 어댑터 감지를 추가하면 https://github.com/aws/opsworks-cookbooks/issues/136이 해결됩니다	2016년 6월 17일
3427	• 에이전트의 자격 증명 검색 문제를 해결했습니다. • Ruby 기본 2.0, 2.1 및 2.2 버전을 2.0.0p647, 2.1.7 및 2.2.3로 업데이트 중	2015년 9월 11일
3426	• `aws-sdk`를 1.65.0로 업데이트됨 • `s3curl`을(를) `s3_file cookbook`(으)로 교체하여 Amazon S3에서 다운로드 개선 • Node.js 기본 버전을 0.12.7로 변경합니다. • Node.js 앱에서 로깅 추가 `shared/log` 디렉터리에서 STDOUT 및 STDERR 로깅 및 회전됨 • 사용자 지정 쿡북 하위 모듈이 명시적으로 업데이트를 체크아웃하도록 만듭니다 • `deploy` 디렉터리가 생성되기 전에 바인드 탑재가 확실한지 확인하는 https://github.com/aws/opsworks-cookbooks/issues/213용 추가된 차선책	2015년 8월 27일

에이전트 버전	설명	릴리스 날짜
3425	• Amazon Linux 및 Ubuntu에 대한 ECS 지원	2015년 7월 27일
3424	• 새로운 기능이 없는 작은 변화	2015년 7월 9일
3422	• Red Hat Enterprise Linux 7의 전폭적인 지원 • `/etc/hosts` 세대가 오류에 대하여 신속하게 복원할 수 있도록 만듭니다.	2015년 6월 29일
3421	• Red Hat Enterprise Linux 7의 데이터베이스 패키지 이름을 재정의할 수 있는 옵션 • `monit systemd` 구성을 업데이트하여 `systemd`가 `kill` 시그널을 `monit`가 모니터 하는 프로세스로 전송하지 않도록 방지합니다	2015년 6월 11일

AWS OpsWorks Stacks 리소스

> **Important**
>
> 이 AWS OpsWorks Stacks 서비스는 2024년 5월 26일에 만료되었으며 신규 및 기존 고객 모두에 대해 비활성화되었습니다. 가능한 한 빨리 워크로드를 다른 솔루션으로 마이그레이션하는 것이 좋습니다. 마이그레이션에 대한 질문이 있는 경우 AWS re:Post 또는 AWS Premium Support를 통해 AWS Support 팀에 문의하세요.

다음의 관련 리소스는 이 서비스 사용 시 도움이 될 수 있습니다.

참조 설명서, 도구, 지원 리소스

AWS OpsWorks Stacks 및 Amazon Web Services에서 몇 가지 유용한 가이드, 포럼, 연락처 정보 및 기타 리소스를 얻을 수 있습니다.

- AWS OpsWorks Stacks API 참조 - 공통 파라미터 및 오류 코드를 포함하여 AWS OpsWorks Stacks 작업 및 데이터 유형에 대한 설명, 구문 및 사용 예제입니다.
- AWS OpsWorks Stacks 기술 FAQ - 개발자가이 제품에 대해 질문한 주요 질문입니다.
- AWS OpsWorks Stacks 출시 정보 – 최신 릴리스에 대한 수준 높은 개요를 제공합니다. 이 문서는 특히 새로운 기능, 상관 관계 및 알려진 문제점에 대해 다룹니다.
- PowerShell용 AWS 도구 - PowerShell 환경에서의 기능을 노출하는 Windows AWS SDK for .NET PowerShell cmdlet 세트입니다.
- AWS 명령줄 인터페이스- AWS 서비스에 액세스하기 위한 균일한 명령줄 구문. AWS CLI는 단일 설정 프로세스를 통해 지원되는 모든 서비스에 대한 액세스를 활성화합니다.
- AWS OpsWorks Stacks 명령줄 참조 - 명령줄 프롬프트에서 사용할 AWS OpsWorks 스택별 명령입니다.

- 클래스 및 워크숍 - AWS 기술을 연마하고 실용적인 경험을 얻는 데 도움이 되는 자기 주도형 랩 외에도 역할 기반 및 특수 과정으로 연결되는 링크입니다.
- AWS 개발자 센터 - 자습서를 살펴보고, 도구를 다운로드하고, AWS 개발자 이벤트에 대해 알아봅니다.
- AWS 개발자 도구 - AWS 애플리케이션 개발 및 관리를 위한 개발자 도구, SDKs, IDE 도구 키트 및 명령줄 도구에 대한 링크입니다.

- 리소스 센터 시작 -를 설정하고 AWS 계정, AWS 커뮤니티에 가입하고, 첫 번째 애플리케이션을 시작하는 방법을 알아봅니다.
- 실습 튜토리얼 – 단계별 튜토리얼에 따라 AWS에서 첫 번째 애플리케이션을 시작합니다.
- AWS 백서 - Solutions Architects 또는 기타 기술 전문가가 작성한 AWS 아키텍처, 보안 및 경제와 같은 주제를 다루는 포괄적인 기술 AWS 백서 목록으로 연결되는 링크입니다.
- AWS Support 센터 - AWS Support 사례를 생성하고 관리하기 위한 허브입니다. 포럼, 기술 FAQs, 서비스 상태 및 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다 AWS Trusted Advisor.
- 지원 - 클라우드에서 애플리케이션을 구축하고 실행하는 데 도움이 지원되는 one-on-one 빠른 응답 지원 채널에 대한 정보를 제공하는 기본 웹 페이지입니다.
- Contact Us(문의처) - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대해 문의할 수 있는 중앙 연락 창구입니다.
- AWS 사이트 약관 - 저작권 및 상표, 계정, 라이선스 및 사이트 액세스, 기타 주제에 대한 자세한 정보입니다.

AWS 소프트웨어 개발 키트

Amazon Web Services는 여러 프로그래밍 언어에서 AWS OpsWorks Stacks에 액세스하기 위한 소프트웨어 개발 키트를 제공합니다. SDK 라이브러리는 서비스 요청에 암호화된 서명 적용, 요청 재시도 또는 오류 응답 처리를 포함한 여러 공통적인 작업을 자동화합니다.

- AWS SDK for Java – 설정 및 기타 설명서
- AWS SDK for .NET – 설정 및 기타 설명서
- AWS SDK for PHP – 설명서
- AWS SDK for Ruby – 설명서
- 기타 설명서
- AWS SDK for Python (Boto) – 설정 및 기타 설명서

오픈 소스 소프트웨어

AWS OpsWorks Stacks에는 해당 라이선스에 의해 관리되는 다양한 오픈 소스 소프트웨어 패키지가 포함되어 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- Chef 12 Linux 인스턴스의 경우 인스턴스의 `/opt/aws/opsworks/current` 디렉터리에서 `THIRD_PARTY_LICENSES` 파일을 엽니다.

- Linux용 Chef 11.10 및 이전 버전의 경우, OpsWorks Linux 에이전트 속성 문서(PDF)로 이동합니다.

AWS OpsWorks 문서 기록

변경 사항	설명	날짜
AWS OpsWorks Stacks에 대한 업데이트	이제 Detach in Place 도구를 사용하여 OpsWorks Stacks 서비스에서 OpsWorks 인스턴스를 분리할 수 있습니다.이 설명서의 AWS OpsWorks Stacks Detach in Place 도구 사용을 참조하세요.	2024년 4월 11일
AWS OpsWorks Stacks에 대한 업데이트	이제 마이그레이션 스크립트를 사용하여 AWS OpsWorks Stacks를 AWS Systems Manager Application Manager로 마이그레이션할 수 있습니다. 자세한 내용은이 설명서의 AWS Systems Manager Application Manager로 AWS OpsWorks Stacks 애플리케이션 마이그레이션을 참조하세요.	2022년 12월 22일
AWS OpsWorks for Chef Automate 및에 대한 업데이트 AWS OpsWorks for Puppet Enterprise	이제 AWS OpsWorks for Chef Automate 또는 OpsWorks for Puppet Enterprise 서버에 대한 시스템 유지 관리가 실패할 경우 수행할 수 있는 작업을 설명하는 문제 해결 절차를 사용할 수 있습니다. 자세한 내용은 이 안내서의 Chef Automate 서버의 시스템 유지 관리 실패 또는 Puppet Enterprise 서버의 시스템 유지 관리 실패를 참조하세요.	2022년 9월 29일

AWS OpsWorks for Chef Automate 및에 대한 업데이트 AWS OpsWorks for Puppet Enterprise	이제 AWS OpsWorks for Chef Automate 또는 OpsWorks for Puppet Enterprise 서버가 `Connection lost` 상태가 되면 문제 해결 절차를 사용할 수 있습니다. 자세한 내용은 이 안내서의 Chef Automate 서버가 `Connection lost` 상태에 있거나 Puppet Enterprise 서버가 `Connection lost` 상태에 있음을 참조하세요.	2022년 3월 23일
AWS OpsWorks Stacks에 대한 업데이트	보안 모범 사례로 이제 `aws:SourceArn` 또는 `aws:SourceAccount` 조건 키(또는 둘 다)를 추가하여 AWS OpsWorks Stacks가 다른 AWS 서비스에서 작업을 수행할 수 있도록 허용하는 신뢰 관계 정책을 신뢰할 수 있습니다. 자세한 내용은 이 안내서의 AWS OpsWorks 스택에서 서비스 간 혼동된 대리자 방지를 참조하세요.	2022년 3월 4일

AWS OpsWorks for Chef Automate 및에 대한 업데이트 AWS OpsWorks for Puppet Enterprise	보안 모범 사례로, 이제 AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise 액세스가 다른 AWS 서비스에서 작업을 수행하도록 허용하는 신뢰 관계 정책에 `aws:SourceArn` 또는 `aws:SourceAccount` 조건 키(또는 둘 다)를 추가할 수 있습니다. 자세한 내용은 이 안내서의 서비스 간 혼동된 대리자 방지를 참조하세요.	2022년 1월 10일
AWS OpsWorks for Chef Automate 및에 대한 업데이트 AWS OpsWorks for Puppet Enterprise	AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise는 관리형 정책 `AWSOpsWorksCMServiceRole` 및 를 업데이트`AWSOpsWorksCMInstanceProfileRole` 했으며 이제 에 보안 암호를 저장합니다 AWS Secrets Manager.	2021년 5월 3일
에 대한 업데이트 AWS OpsWorks for Puppet Enterprise	콘솔에서 생성하는 OpsWorks for Puppet Enterprise 서버의 엔진 버전은 이제 2019.8.5입니다. API를 사용하면 Puppet Enterprise 서버를 생성할 때 버전 `2019` 또는 `2017`을 지정할 수 있습니다. 이제 `DescribeServers` API는 결과에 `PUPPET_API_CRL` 이라는 속성을 반환합니다. 이 속성에는 내부용 인증서 해지 목록이 포함되어 있습니다.	2021년 4월 28일

AWS OpsWorks Stacks는 새로운 관리형 정책을 사용합니다.	AWS OpsWorks Stacks는 AWS OpsWorks Stacks에서 모든 작업을 수행할 수 있는 권한이 포함된 관리형 정책을 변경했습니다. 새 정책은 AWSOpsWorks_FullAccess입니다. 이 정책을 수동으로 생성하는 데 대한 자세한 내용은 예시 정책을 참조하세요.	2021년 2월 19일
EC2-Classic에서 VPC로 AWS OpsWorks Stacks 스택 마이그레이션	AWS OpsWorks Stacks 스택을 EC2-Classic에서 VPC로 마이그레이션하는 방법을 설명하는 설명서가 추가되었습니다.	2020년 9월 29일
AWS OpsWorks for Chef Automate 및에 대한 스타터 키트 재생성 AWS OpsWorks for Puppet Enterprise	AWS OpsWorks for Chef Automate 또는 AWS OpsWorks for Puppet Enterprise 서버의 스타터 키트를 재생성하는 방법을 설명하는 설명서가 추가되었습니다.	2020년 7월 29일
AWS OpsWorks for Puppet Enterprise 를 사용하면 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하는 서버를 생성할 수 있습니다.	이제 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하는 OpsWorks for Puppet Enterprise 서버를 생성할 수 있습니다. 기존 서버의 백업에서 서버를 만들어 사용자 지정 도메인을 사용하도록 기존 Puppet Enterprise 서버를 업데이트할 수 있습니다.	2020년 4월 17일

AWS OpsWorks for Chef AutomateAWS OpsWorks for Puppet Enterprise 이제 콘솔에서 태그 지정을 지원합니다.	이제 AWS Management Console 또는를 사용하여 AWS OpsWorks for Chef Automate 서버 또는 AWS OpsWorks for Puppet Enterprise 마스터 또는 서버 백업에 태그를 추가할 수 있습니다 AWS CLI. 자세한 내용은 태그 작업(Chef) 또는 태그 작업(Puppet)을 참조하세요.	2020년 2월 26일
AWS OpsWorks for Chef Automate 기존 Chef Automate 1 서버를 Chef Automate 2로 업그레이드 간소화	콘솔의 AWS OpsWorks for Chef Automate 서버 세부 정보 페이지에서 업그레이드 시작을 선택하거나 `StartMaintenance` API 작업을 실행하여 Chef Automate 1을 실행하는 적격 서버를 Chef Automate 2로 업그레이드할 수 있습니다. 자세한 내용은 AWS OpsWorks for Chef Automate 서버를 Chef Automate 2로 업그레이드를 참조하세요.	2020년 1월 24일
AWS OpsWorks for Chef Automate 그리고 AWS OpsWorks for Puppet Enterprise	AWS OpsWorks CM(AWS OpsWorks for Chef Automate 및 AWS OpsWorks for Puppet Enterprise)의 보안에 대한 새 장이 가이드에 추가되었습니다.	2019년 12월 23일

AWS OpsWorks for Chef Automate 및 AWS OpsWorks for Puppet Enterprise 지원 태그 지정	이제를 사용하여 AWS OpsWorks for Chef Automate 서버 또는 AWS OpsWorks for Puppet Enterprise 마스터에 태그를 추가하거나 서버 백업에 태그를 추가할 수 있습니다 AWS CLI. AWS OpsWorks CM은 이제 태그 기반 권한 부여를 지원합니다.	2019년 12월 18일
AWS OpsWorks for Chef Automate 를 사용하면 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하는 서버를 생성할 수 있습니다.	이제 사용자 지정 도메인, 인증서 및 프라이빗 키를 사용하는 AWS OpsWorks for Chef Automate 2.0 서버를 생성할 수 있습니다. 기존 서버의 백업에서 서버를 만들어 사용자 지정 도메인을 사용하도록 기존 Chef Automate 2.0 서버를 업데이트할 수 있습니다.	2019년 10월 22일
AWS OpsWorks 이제 Stacks에서 Ruby 2.6.1 지원	AWS OpsWorks Stacks는 Chef 11.10 스택의 Ruby 2.6.1 on Rails 앱 서버 계층을 지원합니다.	2019년 5월 2일
AWS OpsWorks for Chef Automate 이제에서 Chef Automate 2.0을 지원합니다.	새 AWS OpsWorks for Chef Automate 서버는 Chef Automate 2.0을 실행합니다. 여기에는 Chef InSpec에 대한 업데이트, 규정 준수 스캔 및 보고의 새로운 기능, Chef Infra가 포함됩니다.	2019년 4월 30일

AWS OpsWorks for Chef Automate 그리고 AWS OpsWorks for Puppet Enterprise	이제 AWS CloudFormation 를 사용하여 AWS OpsWorks for Chef Automate 서버 또는 AWS OpsWorks for Puppet Enterprise 마스터 서버를 생성할 수 있습니다.	2019년 1월 24일
AWS OpsWorks 스택	AWS OpsWorks 이제 Stacks 는 Chef 12 스택에서 Ubuntu 18.04 LTS를 실행하는 인스턴스를 지원합니다.	2018년 12월 18일
AWS OpsWorks for Puppet Enterprise	CodeCommit를 사용하는 제어 리포지토리에 대한 SSH 기반 연결을 설정하는 절차를 추가했습니다.	2018년 12월 3일
AWS OpsWorks 스택	AWS OpsWorks 이제 Stacks 는 Chef 12 스택에서 Amazon Linux 2를 실행하는 인스턴스를 지원합니다.	2018년 11월 15일
AWS OpsWorks 스택	AWS OpsWorks Stacks는 이제 Chef 11.10 스택에서 Amazon Linux 2018.03을 실행하는 인스턴스를 지원합니다.	2018년 10월 23일
AWS OpsWorks 스택	AWS OpsWorks Stacks는 이제 Chef 12 스택에서 Amazon Linux 2018.03을 실행하는 인스턴스를 지원합니다.	2018년 8월 23일
AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise	OpsWorks for Puppet Enterprise을 PE 2018.1.2.로 업그레이드하고, AWS OpsWorks for Chef Automate를 Chef Automate 1.8.68로 업그레이드하였습니다.	2018년 6월 29일

- AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise API 버전: 2016-11-01
- AWS OpsWorks Stacks API 버전: 2016-03-08
- 최신 설명서 업데이트: 2024-04-11

이전 업데이트

다음 표에서는 2018년 6월 이전 AWS OpsWorks 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

설명	날짜
AWS OpsWorks Windows 기반 스택용 Stacks Chef 버전이 12.22로 업그레이드되었습니다. Ruby 버전은 이제 2.3.6입니다.	2018년 4월 19일
를 사용하여 AWS OpsWorks for Chef Automate 서버 또는 OpsWorks for Puppet Enterprise 마스터를 생성하는 새로운 절차입니다 AWS CLI.	2018년 3월 23일
Chef Automate 버전이 1.8로 업데이트되었습니다. `opsworks-audit` 쿡북이 추가되어 Chef Compliance 설정이 간소화되었습니다.	2018년 3월 5일
Amazon CloudWatch Events의 AWS OpsWorks Stacks 이벤트에 대한 지원이 추가되었습니다.	2018년 2월 20일
AWS OpsWorks Stacks의 새로운 EBS 볼륨 유형과 새로운 API DescribeOperatingSystems에 대한 지원이 추가되었습니다.	2018년 1월 25일
OpsWorks for Puppet Enterprise 및는 AWS OpsWorks for Chef Automate 이제 서버를 생성할 때 여러 보안 그룹 선택을 지원합니다.	2018년 1월 18일
유럽(파리) 리전의 AWS OpsWorks Stacks에 대한 지원이 추가되었습니다.	2017년 12월 19일
6개 추가 리전에서 AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise에 대한 지원을 추가하고에서 AWS OpsWorks for Chef Automate 및 OpsWorks for Puppet Enterprise 서버의 백업을 생성하는 절차를 추가했습니다 AWS Management Console.	2017년 12월 18일

설명	날짜
OpsWorks for Puppet Enterprise 서비스 및 설명서가 새로 추가되었습니다.	2017년 11월 16일
Amazon Linux 2017.09에 대한 지원이 AWS OpsWorks Stacks에 추가되었습니다.	2017년 11월 7일
Chef Compliance에 대한 지원이에 추가되었습니다 AWS OpsWorks for Chef Automate.	2017년 10월 25일
Amazon Linux 2017.09에 대한 지원이에 추가되었습니다 AWS OpsWorks for Chef Automate.	2017년 10월 9일
AWS OpsWorks for Chef Automate 장에 시스템 유지 관리 주제가 추가되었습니다.	2017년 7월 28일
AWS OpsWorks Stacks의 태그에 대한 지원이 추가되었습니다.	2017년 6월 6일
CloudWatch Logs와 통합이 추가되었습니다.	2017년 10월 4일
새 AWS OpsWorks for Chef Automate 서비스 및 설명서가 추가되었습니다.	2016년 1월 12일
미국 동부(오하이오) 리전 엔드포인트에 대한 지원이 추가되었습니다.	2016년 10월 12일
Amazon Linux 2016.09 운영 체제를 실행하는 스택 및 인스턴스에 대한 지원 추가됨.	2016년 9월 30일
아시아 태평양(서울) 리전 및 9개 추가 리전 엔드포인트에 대한 지원 추가됨.	2016년 8월 15일
내장 계층에서 Node.js 0.12.15 및 Ruby 2.3에 대한 지원 추가됨.	2016년 7월 6일
아시아 태평양(뭄바이) 리전에 대한 지원이 추가되었습니다.	2016년 6월 28일
CentOS 7 운영 체제를 실행하는 스택 및 인스턴스에 대한 지원 추가됨.	2016년 6월 22일
CodePipeline 및 AWS OpsWorks Stacks 통합을 설명하는 안내가 추가되었습니다.	2016년 6월 2일

설명	날짜
Ubuntu 16.04 LTS 운영 체제를 실행하는 스택 및 인스턴스에 대한 지원 추가됨.	2016년 6월 1일
Chef 12 Linux 지원 및 관련 설명서 추가됨.	2015년 12월 3일
시작하기에 Node.js 연습 추가됨.	2015년 7월 14일
Cookbooks 101에 새로운 쿡북 예제 2개 추가됨.	2015년 7월 14일
에이전트 버전 관리에 대한 지원 추가됨.	2015년 6월 23일
에이전트 버전 관리에 대한 지원 추가됨.	2015년 6월 24일
사용자 지정 Windows AMI에 대한 지원 추가됨.	2015년 6월 22일
새로운 모범 사례 주제 3개 추가됨.	2015년 6월 11일
Windows 스택에 대한 지원 추가됨.	2015년 5월 18일
모범 사례 장 추가됨.	2014년 12월 15일
Elastic Load Balancing Connection Draining 및 사용자 지정 종료 제한 시간에 대한 지원 추가됨.	2014년 12월 15일
AWS OpsWorks Stacks 외부에서 생성된 인스턴스 등록에 대한 지원 추가됨.	2014년 12월 9일
Amazon SWF에 대한 지원 추가됨.	2014년 9월 4일
환경 변수와 앱 및 확장 Cookbooks 101 연결에 대한 지원 추가됨.	2014년 7월 16일
쿡북 구현 입문용 자습서 Cookbooks 101 추가됨.	2014년 7월 16일
CloudTrail에 대한 지원 추가됨.	2014년 6월 4일
Amazon RDS 지원 추가됨.	2014년 5월 14일
Chef 11.10 및 Berkshelf에 대한 지원 추가됨.	2014년 3월 27일
Amazon EBS PIOPS 볼륨에 대한 지원 추가됨.	2013년 12월 16일

설명	날짜
리소스 기반 권한 추가됨.	2013년 12월 5일
리소스 관리 추가됨.	2013년 10월 7일
VPC에 대한 지원 추가됨.	2013년 8월 29일
사용자 지정 AMI 및 Chef 11.4에 대한 지원 추가됨.	2013년 7월 24일
인스턴스당 다중 계층에 대한 콘솔 지원 추가됨.	2013년 7월 1일
Amazon EBS 지원 인스턴스, Elastic Load Balancing 및 Amazon CloudWatch 모니터링에 대한 지원 추가됨.	2013년 5월 14일
AWS OpsWorks Stacks 사용 설명서 최초 릴리스.	2013년 18월 2일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.