aws

개발자 가이드

AWS Blockchain Templates

AWS Blockchain Templates: 개발자 가이드

Table of Contents

..... iv
AWS Blockchain 템플릿이란 무엇입니까? 1
시작하는 방법 2
AWS 및 블록체인에 능숙합니다. 2
에 능숙 AWS 하고 블록체인을 처음 사용하는 경우 3
의 초보자 AWS 이며 블록체인에 능숙합니다. 3
AWS 및 블록체인을 처음 사용하는 경우 3
관련 서비스 3
설정 5
AWS에 등록 5
IAM 사용자 생성 6
키 페어 생성 7
시작하기 9
사전 조건 설정 10
VPC 및 서브넷 생성 10
보안 그룹 생성 13
Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할 생성 15
Bastion Host 생성 20
Ethereum 네트워크 생성 22
Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결 24
리소스 정리 27
AWS Blockchain 템플릿 및 기능 29
Ethereum용 AWS Blockchain 템플릿 29
시작 링크 29
Ethereum 옵션 29
사전 조건 33
Ethereum 리소스에 연결하기 40
Hyperledger Fabric용 AWS Blockchain 템플릿 42
시작 링크 42
Hyperledger Fabric용 AWS Blockchain 템플릿 구성 요소 42
사전 조건 43
Hyperledger Fabric 리소스에 연결하기 45
문서 기록 47
AWS 용어집 48

AWS Blockchain 템플릿은 2019년 4월 30일에 중단되었습니다. 이 서비스나 이 지원 문서에 대한 추가 업데이트는 없을 것입니다. 에서 최상의 관리형 블록체인 경험을 위해 Amazon Managed Blockchain(AMB)을 사용하는 AWS것이 좋습니다. Amazon Managed Blockchain을 시작하는 방법에 대해 자세히 알아보려면 하이퍼레저 패브릭 워크숍 또는 이더리움 노드 배포에 관한 블로그를 참조하십시오. AMB에 대한 질문이 있거나 추가 지원이 필요한 경우 또는 AWS 계정 팀에 문의 지원하세요.

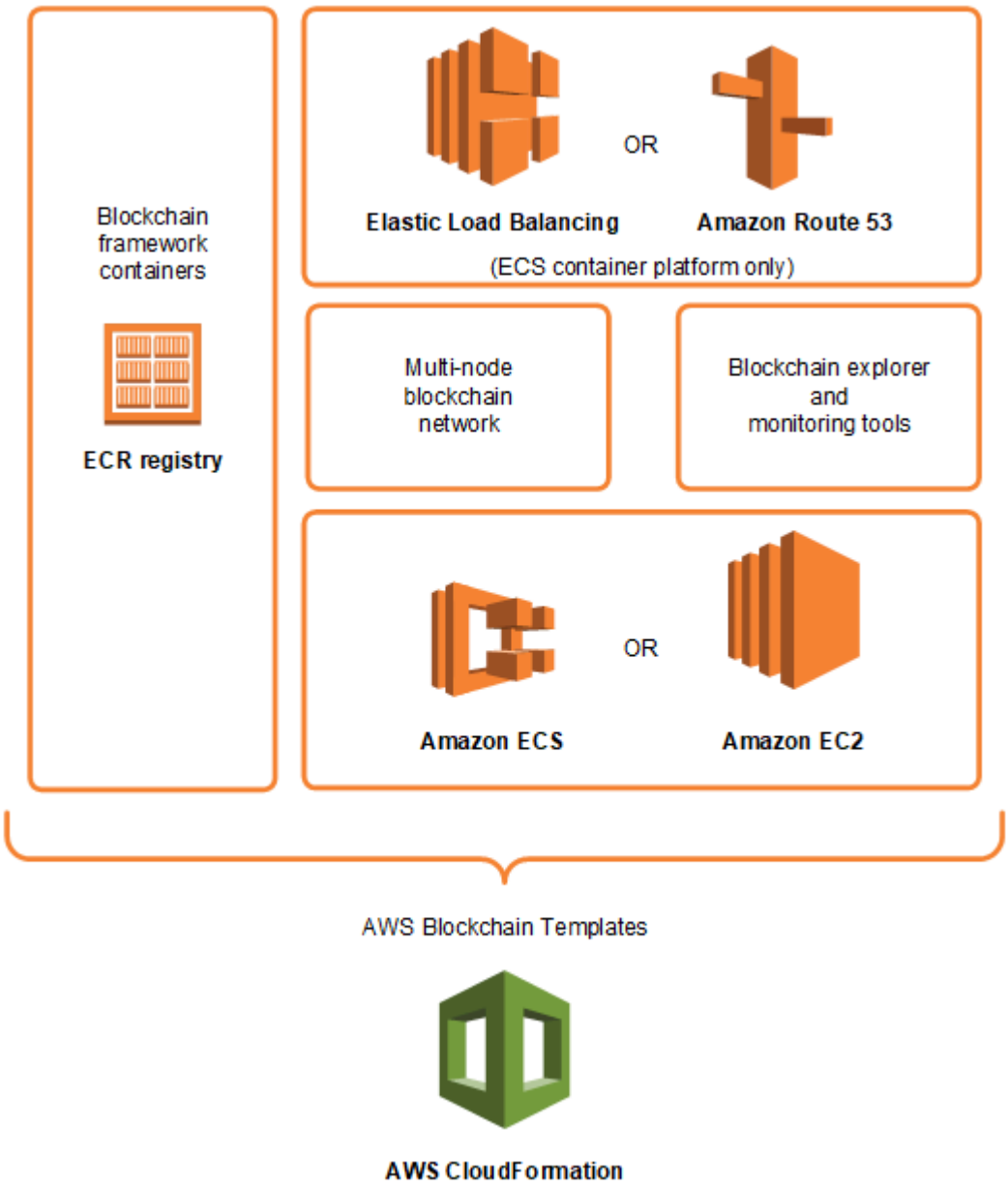
기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

AWS Blockchain 템플릿이란 무엇입니까?

AWS 블록체인 템플릿을 사용하면 다양한 블록체인 프레임워크를 AWS 사용하여에서 블록체인 네트워크를 빠르게 생성하고 배포할 수 있습니다. 블록 체인은 암호화 기법을 사용하여 변조 및 수정을 방지할 수 있도록 강화된 스마트 계약 및 지속적으로 증가하는 일련의 트랜잭션을 유지하는 분산 데이터베이스 기술입니다.

블록 체인 네트워크는 피어 투 피어 네트워크로, 국제 결제, 공급망 관리, 토지 등록, 크라우드펀딩, 거버넌스, 금융 거래 등의 비즈니스 프로세스를 위해 트랜잭션의 효율성과 불변성을 향상시킵니다. 이를 통해 서로를 알 수 없는 사람들과 조직은 트랜잭션 레코드를 신뢰하고 독립적으로 확인할 수 있습니다.

AWS Blockchain 템플릿을 사용하여 CloudFormation 스택을 구성하고 시작하여 블록체인 네트워크를 생성합니다. 사용하는 AWS 리소스 및 서비스는 선택한 WS Blockchain 템플릿 및 지정한 옵션에 따라 다릅니다. 사용 가능한 템플릿 및 그 특징에 대한 자세한 내용은 AWS Blockchain 템플릿 및 기능 단원을 참조하십시오. AWS Blockchain 템플릿을 사용하여 AWS 생성된의 블록체인 네트워크의 기본 구성 요소는 다음 다이어그램에 나와 있습니다.



시작하는 방법

가장 좋은 시작 장소는 블록체인에 대한 전문 지식 수준 AWS, 특히 AWS 블록체인 템플릿과 관련된 서비스에 따라 달라집니다.

AWS 및 블록체인에 능숙합니다.

사용하고자 하는 프레임워크에 관해 AWS Blockchain 템플릿 및 기능 주제에서 시작하십시오. 링크를 사용하여 WS Blockchain 템플릿을 시작하고 블록체인 네트워크를 구성하거나, 템플릿을 다운로드하여 직접 확인합니다.

에 능숙 AWS 하고 블록체인을 처음 사용하는 경우

AWS Blockchain 템플릿 시작하기 자습서로 시작합니다. 기본 설정과 함께 간단한 이더리움 블록 체인 네트워크를 생성하는 방법을 안내합니다. 완료되면 AWS Blockchain 템플릿 및 기능 단원에서 블록 체인 프레임워크의 개요와 구성 선택 및 특징에 대한 자세한 내용이 있는 링크를 참조하십시오.

의 초보자 AWS 이며 블록체인에 능숙합니다.

AWS Blockchain 템플릿 설정 단원에서 시작합니다. 이렇게 하면 계정 및 사용자 프로필과 AWS같은 기본 사항을 설정할 수 있습니다. 다음으로 AWS Blockchain 템플릿 시작하기 자습서를 진행하십시오. 이 자습서는 간단한 이더리움 블록 체인 네트워크를 생성하는 방법을 안내합니다. 궁극적으로 이더리움을 사용하지 않는다 해도 관련 서비스의 직접 설정을 경험합니다. 이 경험은 모든 블록 체인 프레임워크에 유용합니다. 마지막으로 AWS Blockchain 템플릿 및 기능 단원에서 프레임워크에 대한 주제를 참조하십시오.

AWS 및 블록체인을 처음 사용하는 경우

AWS Blockchain 템플릿 설정 단원에서 시작합니다. 이렇게 하면 계정 및 사용자 프로필과 AWS같은 기본 사항을 설정할 수 있습니다. 다음으로 AWS Blockchain 템플릿 시작하기 자습서를 진행하십시오. 이 자습서는 간단한 이더리움 블록 체인 네트워크를 생성하는 방법을 안내합니다. 시간을 내어 링크를 탐색하여 AWS 서비스 및 이더리움에 대해 자세히 알아보세요.

관련 서비스

선택한 옵션에 따라 AWS Blockchain 템플릿은 다음 AWS 서비스를 사용하여 블록체인을 배포할 수 있습니다.

- Amazon EC2—블록체인 네트워크의 컴퓨팅 용량을 제공합니다. 자세한 내용은 Amazon EC2 사용 설명서를 참조하세요.
- Amazon ECS—사용하기로 선택한 경우 블록체인 네트워크에 대해 클러스터의 EC2 인스턴스 간에 컨테이너 배포를 오케스트레이션합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서를 참조하세요.
- Amazon VPC—생성한 Ethereum 리소스에 대한 네트워크 액세스를 제공합니다. 액세스 가능성 및 보안에 대한 구성을 사용자 지정할 수 있습니다. 자세한 내용은 Amazon VPC 개발자 안내서를 참조하세요.
- Application Load Balancing—Amazon ECS를 컨테이너 플랫폼으로 사용할 때 사용 가능한 사용자 인터페이스 및 내부 서비스 검색에 대한 액세스의 단일 접점 역할을 합니다. 자세한 내용을 알아보려

면 Application Load Balancer 사용 설명서의 Application Load Balancer란 무엇입니까을 참조하세요.

AWS Blockchain 템플릿 설정

AWS Blockchain 템플릿을 시작하기 전에, 먼저 다음 작업을 완료합니다:

- AWS에 등록
- IAM 사용자 생성
- 키 페어 생성

모든 블록 체인 구성의 기본 사전 조건입니다. 추가로 사용자가 선택한 블록 체인 네트워크에는 사전 조건이 존재할 수 있으며, 이는 원하는 환경과 구성 선택에 따라 다릅니다. 자세한 내용은 AWS Blockchain 템플릿 및 기능의 블록 체인 템플릿 관련 단원을 참조하십시오.

Amazon ECS 클러스터를 사용하여 프라이빗 Ethereum 네트워크에 대한 사전 조건을 설정하기 위한 단계별 지침은 AWS Blockchain 템플릿 시작하기을 참조하십시오.

AWS에 등록

가입하면 AWS AWS 계정이 모든 서비스에 자동으로 등록됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다.

AWS 계정이 이미 있는 경우 다음 작업으로 건너뜁니다. AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정을 생성하려면

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자가 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 계정 번호를 기록해 둡니다. 다음 작업에서 IAM 사용자를 생성할 때 이 번호가 필요합니다.

IAM 사용자 생성

의 서비스는 리소스에 액세스할 수 있는 권한이 있는지 여부를 확인할 수 있도록 액세스 시 자격 증명을 AWS 제공해야 합니다. 콘솔은 암호를 요구합니다. AWS 계정에 대한 액세스 키를 생성하여 명령줄 인터페이스 또는 API에 액세스할 수 있습니다. 그러나 AWS 계정의 자격 증명을 AWS 사용하여에 액세스하지 않는 것이 좋습니다. 대신 AWS Identity and Access Management (IAM)을 사용하는 것이 좋습니다. IAM 사용자를 생성하여 관리자 권한과 함께 IAM 그룹에 추가하거나, 이 사용자에게 관리자 권한을 부여하십시오. 그런 다음 IAM 사용자의 특수 URL과 자격 증명을 AWS 사용하여에 액세스할 수 있습니다.

에 가입 AWS 했지만 IAM 사용자를 직접 생성하지 않은 경우 IAM 콘솔을 사용하여 생성할 수 있습니다. IAM 사용자가 이미 있는 경우 이 단계를 건너뛸 수 있습니다.

다음 옵션 중 하나를 선택하여 관리 사용자를 생성합니다.

관리자를 관리하는 방법 한 가지 선택	목적	By	다른 방법
IAM Identity Center에서 (권장)	단기 보안 인증 정보를 사용하여 AWS에 액세스합니다. 이는 보안 모범 사례와 일치합니다. 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.	AWS IAM Identity Center 사용 설명서의 시작하기 지침을 따릅니다.	AWS Command Line Interface 사용 설명서에서 사용하도록 AWS CLI 를 구성 AWS IAM Identity Center하여 프로그래밍 방식 액세스를 구성합니다.
IAM에서 (권장되지 않음)	장기 보안 인증 정보를 사용하여 AWS에 액세스합니다.	IAM 사용 설명서의 비상 액세스를 위한 IAM 사용자 생성에 나와 있는 지침을 따르세요.	IAM 사용 설명서에 나온 IAM 사용자의 액세스 키 관리를 수행하여 프로그래밍 방식의 액세스를 구성합니다.

이 새 IAM 사용자로 로그인하려면에서 로그아웃 AWS Management Console한 다음 다음 다음 URL을 사용합니다. 여기서 your_aws_account_id는 하이픈이 없는 AWS 계정 번호입니다(예: AWS 계정 번호가 인 경우 1234-5678-9012 AWS 계정 ID는). 123456789012

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

방금 생성한 IAM 사용자 이름과 암호를 입력합니다. 로그인하면 네비게이션 바에 "your_user_name @ your_aws_account_id"가 표시됩니다.

로그인 페이지의 URL에 AWS 계정 ID를 포함하지 않으려면 계정 별칭을 생성할 수 있습니다. IAM 대시보드에서 계정 별칭 생성을 선택하고 기업명 등의 별칭을 입력합니다. 계정 별칭 생성 후에는 다음 URL에서 로그인하세요.

```
https://your_account_alias.signin.aws.amazon.com/console/
```

사용자 계정의 IAM 사용자 로그인 링크를 확인하려면 IAM 콘솔을 열고 대시보드의 IAM 사용자 로그인 링크에서 확인합니다.

자세한 내용은 AWS 자격 증명 및 액세스 관리 사용 설명서를 참조하십시오.

키 페어 생성

AWS 는 퍼블릭 키 암호화를 사용하여 블록체인 네트워크의 인스턴스에 대한 로그인 정보를 보호합니다. 각 AWS Blockchain 템플릿을 사용할 때 키 페어의 이름을 지정합니다. 키 페어를 사용하여 인스턴스에 직접 액세스할 수 있습니다. SSH를 사용한 로그인이 그 예입니다.

오른쪽 리전에 키 페어가 이미 있는 경우 이 단계를 건너뛸 수 있습니다. 키 페어를 아직 생성하지 않은 경우 Amazon EC2 콘솔을 사용하여 생성할 수 있습니다. 이더리움 네트워크를 시작하는 데 사용하는 리전과 동일한 리전에서 키 페어를 생성합니다. 자세한 내용은 Amazon EC2 사용 설명서에서 리전 및 가용 영역을 참조하세요.

키 페어 생성

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 모음에서 키 페어를 만들 리전을 선택합니다. 위치에 상관없이 사용 가능한 어떤 리전도 선택할 수 있지만 키 페어는 리전에 고유합니다. 예를 들어 미국 동부(오하이오) 리전에서 인스턴스를 시작하려면 동일 리전에서 인스턴스에 대한 키 페어를 생성해야 합니다.
3. 탐색 창에서 Key Pairs(키 페어), Create Key Pair(키 페어 생성)를 선택합니다.

4. 키 페이 이름에 새 키 페어의 이름을 입력합니다. 기억하기 쉬운 이름을 선택합니다(예: IAM 사용자 이름, `-key-pair` 및 리전 이름의 조합). 예를 들어, me-key-pair-useast2로 지정할 수 있습니다. 생성(Create)을 선택합니다.
5. 브라우저에서 프라이빗 키 파일이 자동으로 다운로드됩니다. 기본 파일 이름은 키 페어의 이름으로 지정된 이름이며, 파일 이름 확장명은 `.pem`입니다. 안전한 장소에 프라이빗 키 파일을 저장합니다.

> **Important**
>
> 이때가 사용자가 프라이빗 키 파일을 저장할 수 있는 유일한 기회입니다. 이더리움 네트워크를 시작할 때 키 페어의 이름을 제공합니다.

자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요. 키 페어를 사용하여 EC2 인스턴스에 연결하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Linux 인스턴스에 연결을 참조하세요.

AWS Blockchain 템플릿 시작하기

이 자습서에서는 Ethereum용 AWS Blockchain 템플릿을 사용하여에서 AWS 까지 프라이빗 블록체인 네트워크를 생성하는 방법을 보여줍니다 CloudFormation. 생성한 네트워크에는 Amazon ECS 클러스터의 Amazon EC2 인스턴스에서 실행되는 Ethereum 클라이언트 2개와 마이너 1개가 있습니다. Amazon ECS는 Amazon ECR에서 가져온 Docker 컨테이너에서 이러한 서비스를 실행합니다. 이 자습서를 시작하기 전에 블록체인 네트워크 및 관련 AWS 서비스에 대해 아는 것이 유용하지만 필수는 아닙니다.

이 자습서는 AWS Blockchain 템플릿 설정에서 다루는 일반적인 사전 조건이 설정된 것으로 가정합니다. 또한 템플릿을 사용하기 전에 Amazon VPC 네트워크 및 IAM 역할에 대한 특정 권한과 같은 일부 AWS 리소스를 설정해야 합니다.

이 자습서는 이러한 사전 조건을 설정하는 방법을 설명합니다. 여기서는 설정 옵션을 미리 선택했지만, 이러한 선택을 반드시 따라야 하는 것은 아닙니다. 사전 조건을 충족하는 한, 애플리케이션과 환경의 필요를 기반으로 다른 구성을 선택할 수 있습니다. 각 템플릿의 특징 및 일반 사전 조건에 대한 자세한 내용을 알아보고 템플릿을 다운로드하거나 CloudFormation에서 직접 시작하려면 AWS Blockchain 템플릿 및 기능 단원을 참조하십시오.

이 자습서의 모든 예제에서는 미국 서부(오레곤) 리전(us-west-2)을 사용합니다. 하지만 AWS Blockchain 템플릿을 지원하는 모든 리전을 사용할 수 있습니다.

- 미국 서부(오레곤) 리전(us-west-2)
- 미국 동부(버지니아 북부) 리전(us-east-1)
- 미국 동부(오하이오) 리전(us-east-2)

> **Note**
>
> 위에 열거되지 않은 리전에서 템플릿을 실행하면 미국 동부(버지니아 북부) 리전(us-east-1)의 리소스를 시작합니다.

이 자습서를 사용하여 구성하는 Ethereum용 AWS 블록체인 템플릿에는 다음 리소스가 생성됩니다:

- 지정한 유형 및 개수의 온디맨드 EC2 인스턴스. 이 자습서에서는 기본 t2.medium 인스턴스 유형을 사용합니다.
- 내부 Application Load Balancer입니다.

이후 자습서에서 생성한 리소스를 제거하는 단계가 제공됩니다.

주제

- 사전 조건 설정
- Ethereum 네트워크 생성
- Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결
- 리소스 정리

사전 조건 설정

이 자습서에서 지정하는 Ethereum용 AWS 블록체인 템플릿 구성에서는 다음을 수행해야 합니다:

- VPC 및 서브넷 생성
- 보안 그룹 생성
- Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할 생성
- Bastion Host 생성

VPC 및 서브넷 생성

Ethereum용 AWS 블록체인 템플릿은 Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 사용자가 정의한 가상 네트워크로 리소스를 시작합니다. 이 자습서에서 지정하는 구성에는 Application Load Balancer가 생성되며, 서로 다른 가용 영역에 있는 2개의 퍼블릭 서브넷이 필요합니다. 또한 컨테이너 인스턴스에는 프라이빗 서브넷이 필요하며, 서브넷은 Application Load Balancer와 동일한 가용 영역에 있어야 합니다. 먼저 VPC 마법사를 사용하여 동일한 가용 영역에서 퍼블릭 서브넷 하나와 프라이빗 서브넷 하나를 생성합니다. 그런 다음 다른 가용 영역에 있는 이 VPC 내에서 두 번째 퍼블릭 서브넷을 생성합니다.

자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.

Amazon VPC 콘솔 (https://console.aws.amazon.com/vpc/)을 사용하여 엘라스틱 IP 주소, VPC 및 아래에 설명한 서브넷을 생성합니다.

탄력적 IP 주소를 만들려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탄력적 IP, 새 주소 할당, 할당을 선택합니다.

3. 생성한 탄력적 IP 주소를 기록해 두고 닫기를 선택합니다.
4. 탄력적 IP 주소 목록에서 이전에 생성한 탄력적 IP 주소의 할당 ID를 찾습니다. VPC를 생성할 때 이를 사용합니다.

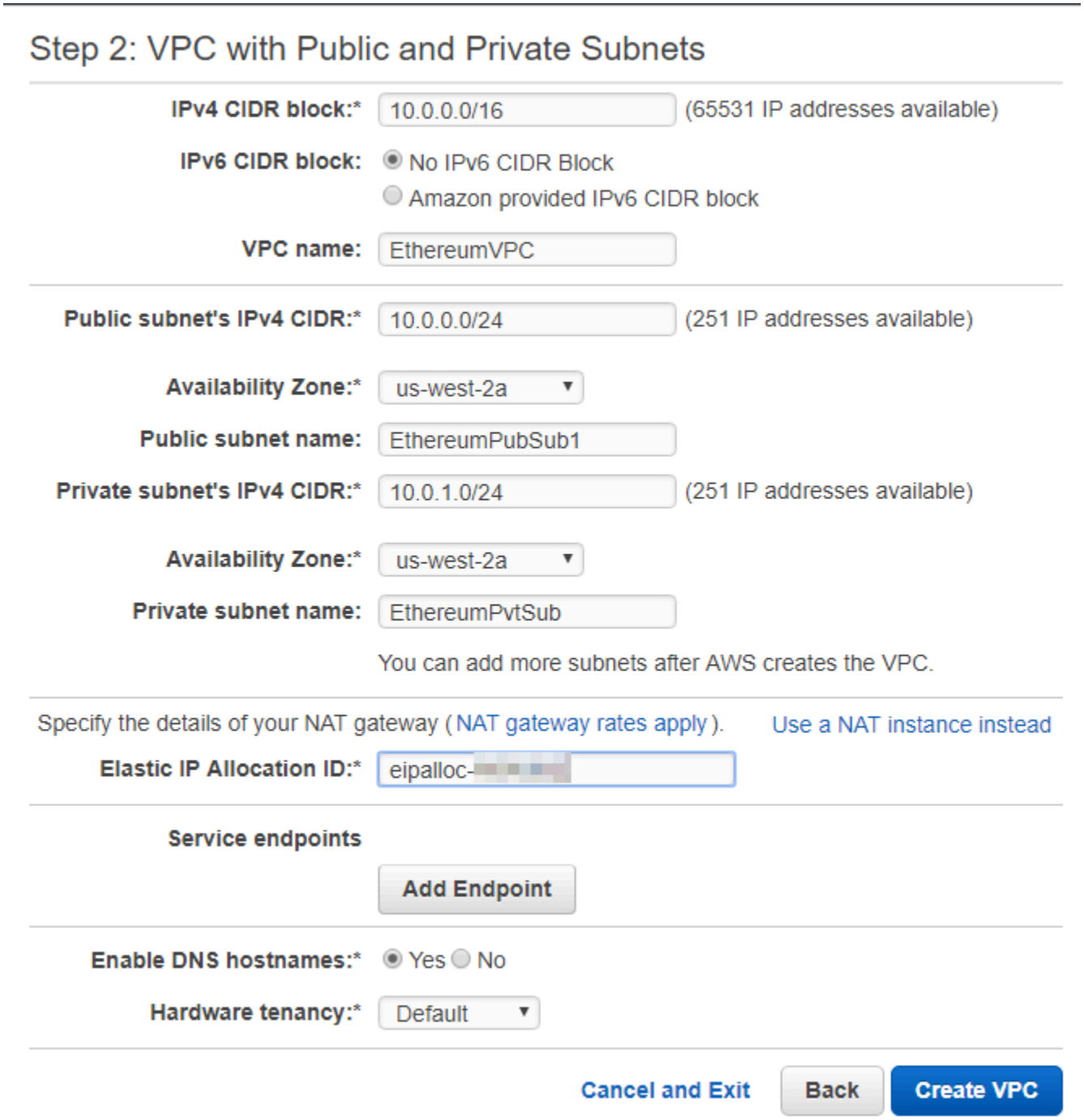
VPC를 생성하려면

1. 탐색 모음에서 VPC를 생성할 리전을 선택합니다. VPC는 리전마다 고유하므로 키 페어를 생성하고 Ethereum 스택을 시작한 리전과 동일한 리전을 선택합니다. 자세한 내용은 키 페어 생성 단원을 참조하십시오.
2. VPC 대시보드에서 [Start VPC Wizard]를 선택합니다.
3. 1단계: VPC 구성 선택 페이지에서 퍼블릭 및 프라이빗 서브넷이 있는 VPC, 선택을 선택합니다.
4. 2단계: 퍼블릭 및 프라이빗 서브넷이 있는 VPC 페이지에서 IPv4 CIDR 블록 및 IPv6 CIDR 블록을 기본값으로 그대로 둡니다. VPC 이름에 기억하기 쉬운 이름을 입력합니다.
5. 퍼블릿 서브넷의 IPv4 CIDR에서 기본값을 그대로 둡니다. 가용 영역에서 영역을 선택합니다. 퍼블릭 서브넷 이름에 기억하기 쉬운 이름을 입력합니다.

   템플릿을 사용할 때 이 서브넷을 Application Load Balancer에 대한 첫 두 개의 서브넷 중 하나로 지정합니다.

   프라이빗 서브넷에 대해 동일한 가용 영역을 선택하고 다른 퍼블릭 서브넷에 대해 다른 가용 영역을 선택하기 때문에 이 서브넷의 가용 영역에 주의해야 합니다.
6. 프라이빗 서브넷의 IPv4 CIDR에서 기본값을 그대로 둡니다. 가용 영역에서 이전 단계와 동일한 가용 영역을 선택합니다. 프라이빗 서브넷 이름에 기억하기 쉬운 이름을 입력합니다.
7. 탄력적 IP 할당 ID에서 이전에 생성한 탄력적 IP 주소를 선택합니다.
8. 기타 설정의 기본값을 유지합니다.
9. VPC 생성을 선택합니다.

   아래 예는 퍼블릭 서브넷 EthereumPubSub1과 프라이빗 서브넷 EthereumPvtSub1이 있는 VPC EthereumNetworkVPC를 보여줍니다. 퍼블릭 서브넷은 가용 영역 us-west-2a를 사용합니다.

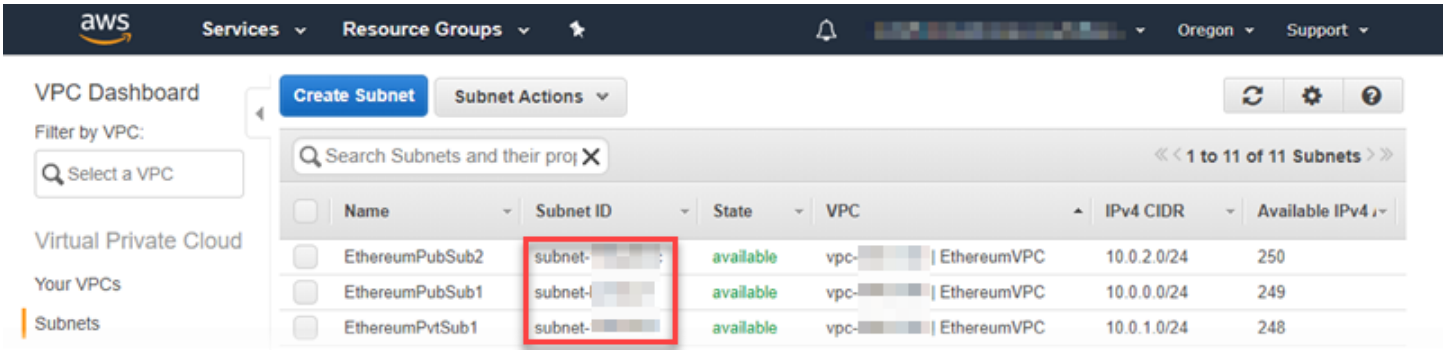


다른 가용 영역에서 두 번째 퍼블릭 서브넷을 생성하려면

1. 서브넷을 선택한 다음 목록에서 이전에 생성한 퍼블릭 서브넷을 선택합니다. 라우팅 테이블 탭을 선택하고 라우팅 테이블 ID를 기록해 둡니다. 아래의 두 번째 퍼블릭 서브넷에 대해 동일한 라우팅 테이블을 지정합니다.
2. 서브넷 생성을 선택합니다.

3. Name tag(이름 태그)에 서브넷의 이름을 입력합니다. 나중에 이 네트워크에서 Bastion Host를 생성할 때 이 이름을 사용합니다.
4. VPC에서 이전에 생성한 VPC를 선택합니다.
5. 가용 영역에서 첫 번째 퍼블릭 서브넷에 대해 선택한 영역과 다른 영역을 선택합니다.
6. IPv4 CIDR 블록에 10.0.2.0/24를 입력합니다.
7. 예, 생성을 선택합니다. 서브넷이 서브넷 목록에 추가됩니다.
8. 목록에서 서브넷을 선택한 상태에서 서브넷 작업, IP 자동 할당 설정 수정을 선택합니다. IP 자동 할당, 저장, 닫기를 선택합니다. 이렇게 하면 이 서브넷에서 Bastion Host를 생성할 때 접속 호스트가 퍼블릭 IP 주소를 얻을 수 있습니다.
9. 라우팅 테이블 탭에서 편집을 선택합니다. 다음으로 변경에서 이전에 기록해 둔 라우팅 테이블 ID를 선택하고 저장을 선택합니다.

이제 이전에 생성한 VPC에 대해 3개의 서브넷이 표시됩니다. 서브넷 이름과 ID를 기록해 두면 템플릿을 사용하여 해당 항목을 지정할 수 있습니다.



보안 그룹 생성

보안 그룹은 방화벽 역할을 하면서 리소스에 대한 인바운드 및 아웃바운드 트래픽을 제어합니다. 템플릿을 사용하여 Amazon ECS 클러스터에서 Ethereum 네트워크를 생성할 때 보안 그룹 두 개를 지정합니다.

- 클러스터의 EC2 인스턴스에 대한 트래픽을 제어하는 EC2 인스턴스용 보안 그룹
- Application Load Balancer, EC2 인스턴스 및 Bastion Host 간의 트래픽을 제어하는 Application Load Balancer에 대한 보안 그룹입니다. 이 보안 그룹을 Bastion Host와도 연결합니다.

각 보안 그룹은 기타 최소 규칙은 물론 Application Load Balancer와 EC2 인스턴스 사이의 통신을 허용하는 규칙을 보유합니다. 이로 인해 다른 보안 그룹 참조가 필요합니다. 이러한 이유로 보안 그룹을 먼저 생성한 다음 적절한 규칙을 업데이트합니다.

2개의 보안 그룹을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹(Security Groups)과 보안 그룹 생성(Create Security Group)을 차례로 선택합니다.
3. 보안 그룹 이름에 식별하기 쉽고 다른 것과 쉽게 구분되는 보안 그룹의 이름을 입력합니다(예: EthereumEC2-SG 또는 EthereumALB-SG). 나중에 이러한 이름을 사용합니다. 설명에 간략한 설명을 입력합니다.
4. VPC에서 이전에 생성한 VPC를 선택합니다.
5. 생성(Create)을 선택합니다.
6. 위 단계를 반복하여 또 다른 보안 그룹을 생성합니다.

EC2 인스턴스용 보안 그룹에 인바운드 규칙 추가

1. 이전에 생성한 EC2 인스턴스용 보안 그룹을 선택합니다.
2. 인바운드 탭에서 편집을 선택합니다.
3. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 소스에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 현재 편집 중인 보안 그룹(예: EthereumEC2-SG)을 선택합니다. 이를 통해 보안 그룹의 EC2 인스턴스가 다른 인스턴스와 통신할 수 있습니다.
4. 규칙 추가(Add Rule)를 선택합니다.
5. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 소스에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 Application Load Balancer에 대한 보안 그룹(예: EthereumALB-SG)을 선택합니다. 이를 통해 보안 그룹의 EC2 인스턴스가 Application Load Balancer와 통신할 수 있습니다.
6. 저장을 선택합니다.

Application Load Balancer용 보안 그룹에 대한 인바운드 규칙 추가 및 아웃바운드 규칙 편집

1. 이전에 생성한 Application Load Balancer용 보안 그룹을 선택합니다.
2. 인바운드 탭에서 편집을 선택하고 다음 인바운드 규칙을 추가합니다.

    a. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 소스에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 현재 편집 중인 보안 그룹(예: EthereumALB-SG)을 선택합니다. 이렇게 하면 Application Load Balancer가 자체 및 Bastion Host와 통신할 수 있습니다.

    b. 규칙 추가(Add Rule)를 선택합니다.

c. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 소스에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 EC2 인스턴스에 대한 보안 그룹(예: EthereumEC2-SG)을 선택합니다. 이렇게 하면 보안 그룹의 EC2 인스턴스가 Application Load Balancer 및 Bastion Host와 통신할 수 있습니다.

d. 규칙 추가(Add Rule)를 선택합니다.

e. Type(유형)에서 SSH를 선택합니다. 소스에서 내 IP를 선택합니다. 그러면 컴퓨터의 IP CIDR이 감지되어 입력됩니다.

> **Important**
>
> 이 규칙을 사용하면 Bastion Host가 컴퓨터에서 오는 SSH 트래픽을 수락할 수 있으므로, 컴퓨터가 Bastion Host를 사용하여 웹 인터페이스를 보고 Ethereum 네트워크의 EC2 인스턴스에 연결할 수 있습니다. 다른 사람들이 Ethereum 네트워크에 연결할 수 있도록 허용하려면 해당 컴퓨터를 이 규칙에 소스로 추가하십시오. 신뢰할 수 있는 소스로 가는 인바운드 트래픽만 허용하십시오.

f. 저장을 선택합니다.

3. 아웃바운드 탭에서 편집을 선택하고 자동으로 생성된 규칙을 삭제하여 모든 IP 주소로 가는 아웃바운드 트래픽을 허용합니다.
4. 규칙 추가(Add Rule)를 선택합니다.
5. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 대상에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 EC2 인스턴스에 대한 보안 그룹을 선택합니다. 이렇게 하면 Application Load Balancer 및 Bastion Host에서 Ethereum 네트워크의 EC2 인스턴스로 가는 아웃바운드 연결이 허용됩니다.
6. 규칙 추가(Add Rule)를 선택합니다.
7. 유형(Type)에서 모든 트래픽(All traffic)을 선택합니다. 대상에서 사용자 지정을 선택된 상태로 그대로 두고 목록에서 현재 편집 중인 보안 그룹(예: EthereumALB-SG)을 선택합니다. 이렇게 하면 Application Load Balancer가 자체 및 Bastion Host와 통신할 수 있습니다.
8. 저장을 선택합니다.

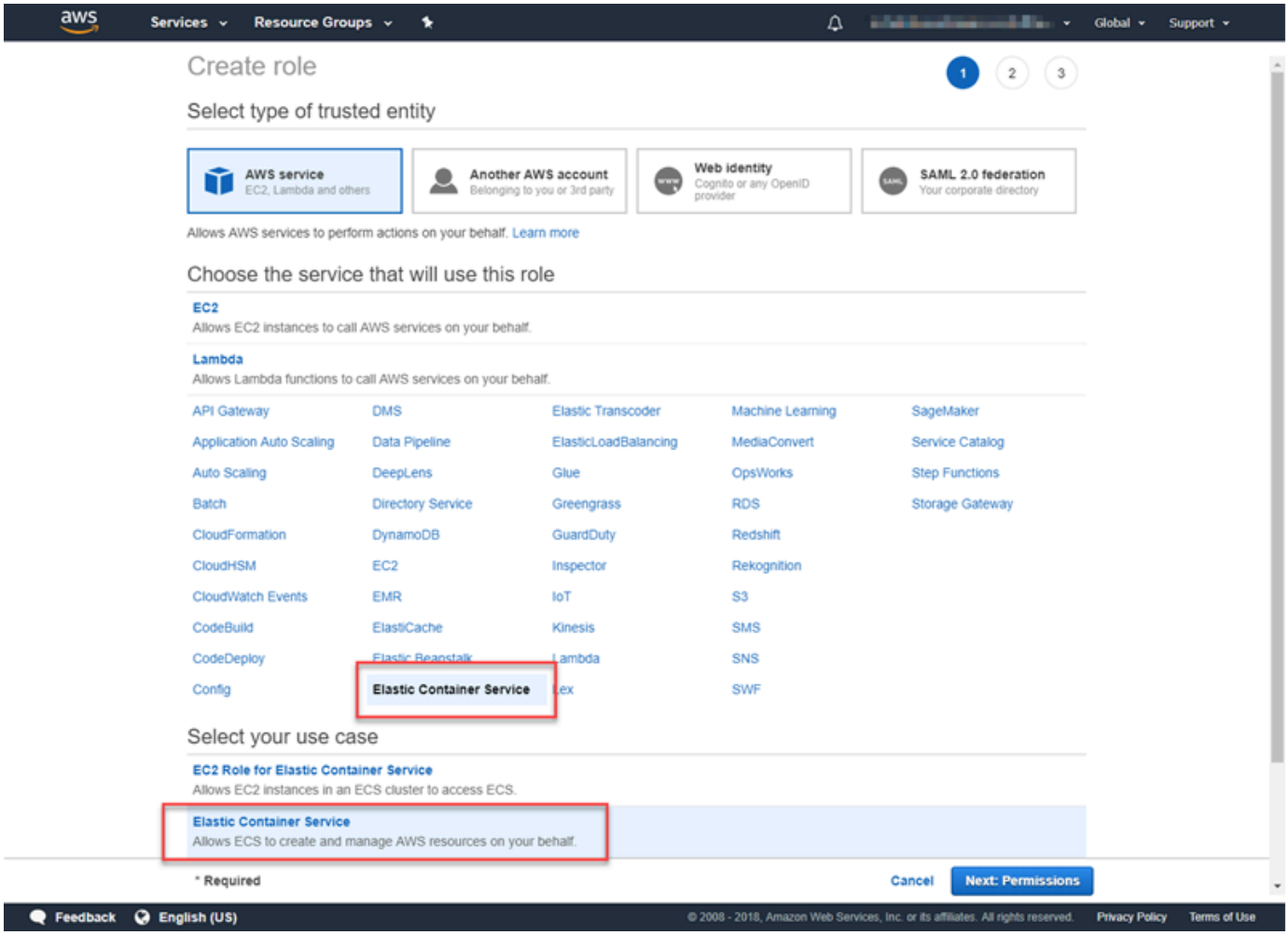
Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할 생성

이 템플릿을 사용할 때 Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할을 지정합니다. 이러한 역할에 연결된 권한 정책을 사용하면 클러스터의 AWS 리소스 및 인스턴스가 다른 AWS 리소스와 상호 작용할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요. IAM 콘솔

(https://console.aws.amazon.com/iam/)을 사용하여 Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할을 설정합니다.

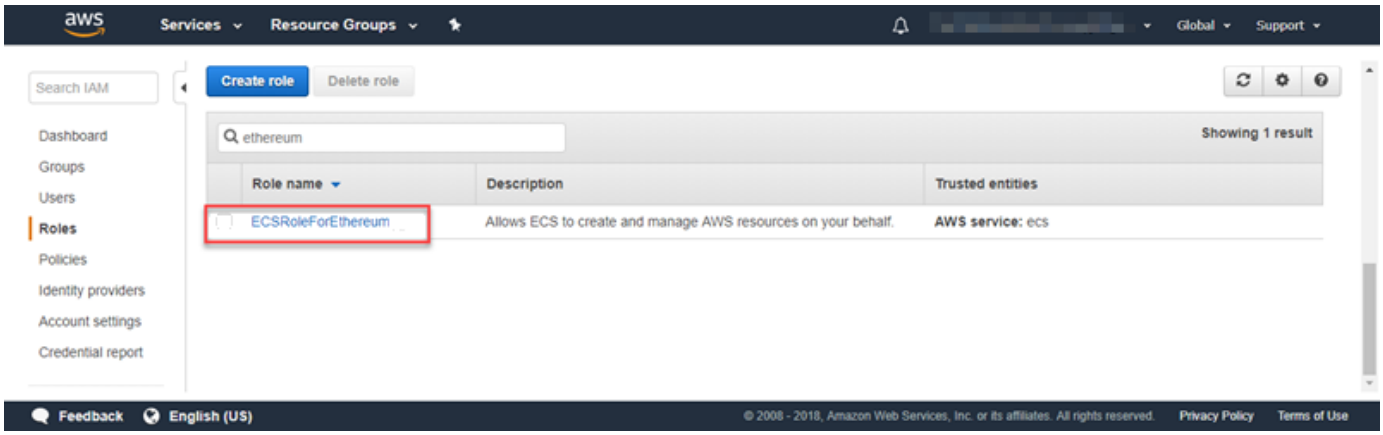
Amazon ECS에 대한 IAM 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할, 역할 생성을 선택합니다.
3. Select type of trusted entity(신뢰할 수 있는 유형의 엔터티 선택) 아래에서 AWS 서비스를 선택합니다.
4. 이 역할을 사용할 서비스 선)에서 Elastic Container Service를 선택합니다.
5. [Select your use case]에서 [Elastic Container Service], [Next:Permissions]를 선택합니다.

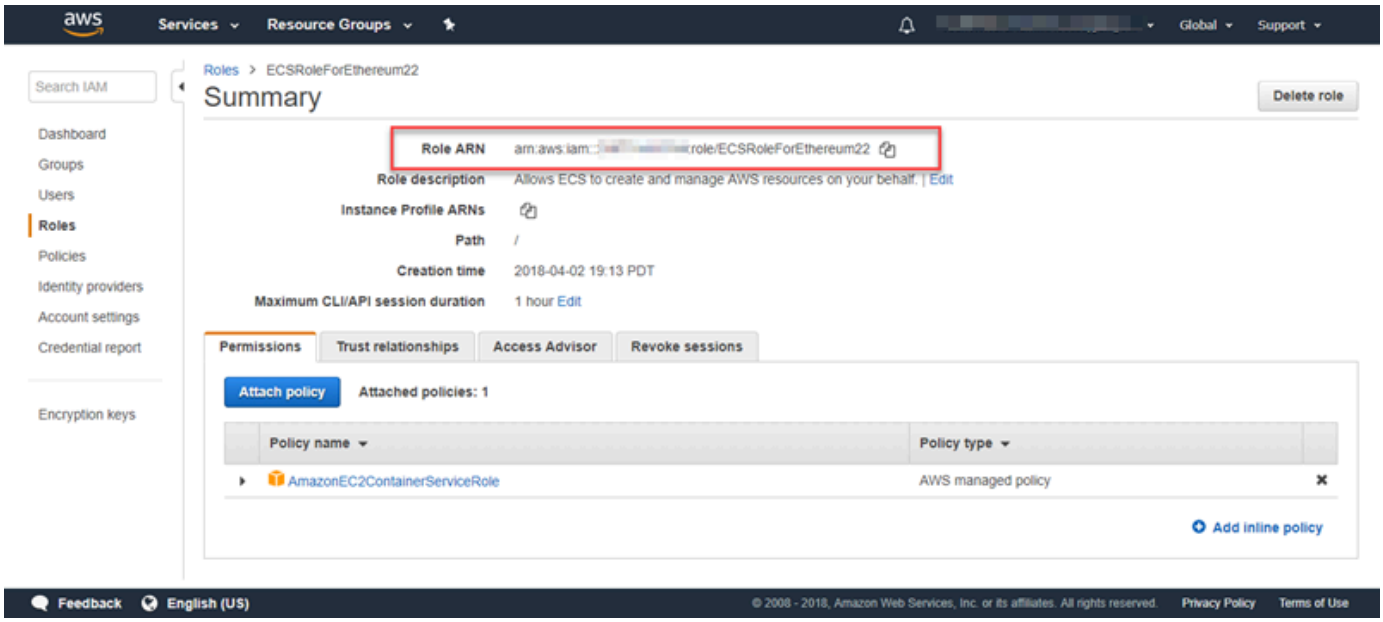


6. [Permissions policy]에서 기본 권한 정책([AmazonEC2ContainerServiceRole])을 선택한 상태로 유지하고 [Next:Review]를 선택합니다.

7. [Role name]에서 이 역할을 식별하는 데 도움이 되는 값을 입력합니다(예: ECSRoleForEthereum). [Role Description]에 간략한 설명을 입력합니다. 나중에 사용할 수 있도록 역할 이름을 기록해 두십시오.
8. 역할 생성을 선택합니다.
9. 목록에서 방금 생성한 역할을 선택합니다. 계정에 여러 역할이 있는 경우 역할 이름을 검색할 수 있습니다.



10. [Role ARN] 값을 복사하고 다시 복사할 수 있도록 저장합니다. Ethereum 네트워크를 생성할 때 이 ARN이 필요합니다.



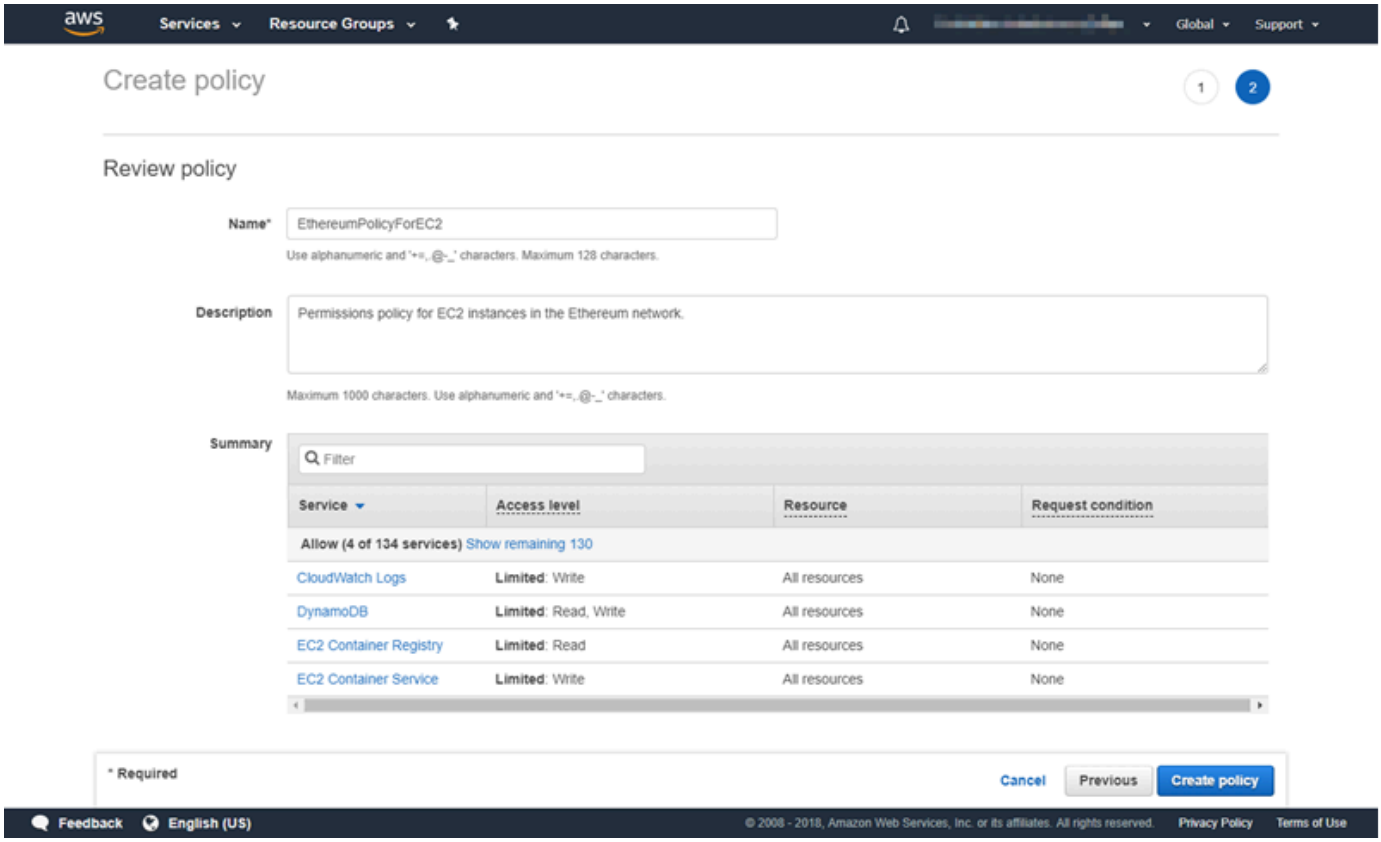
템플릿에서 지정하는 EC2 인스턴스 프로파일은 Ethereum 네트워크의 EC2 인스턴스가 다른 AWS 서비스와 상호 작용한다고 가정합니다. 역할에 대한 권한 정책을 생성하고, 역할을 생성(동일한 이름의 인스턴스 프로파일을 자동으로 생성)한 다음 역할에 권한 정책을 연결합니다.

EC2 인스턴스 프로파일을 생성하려면

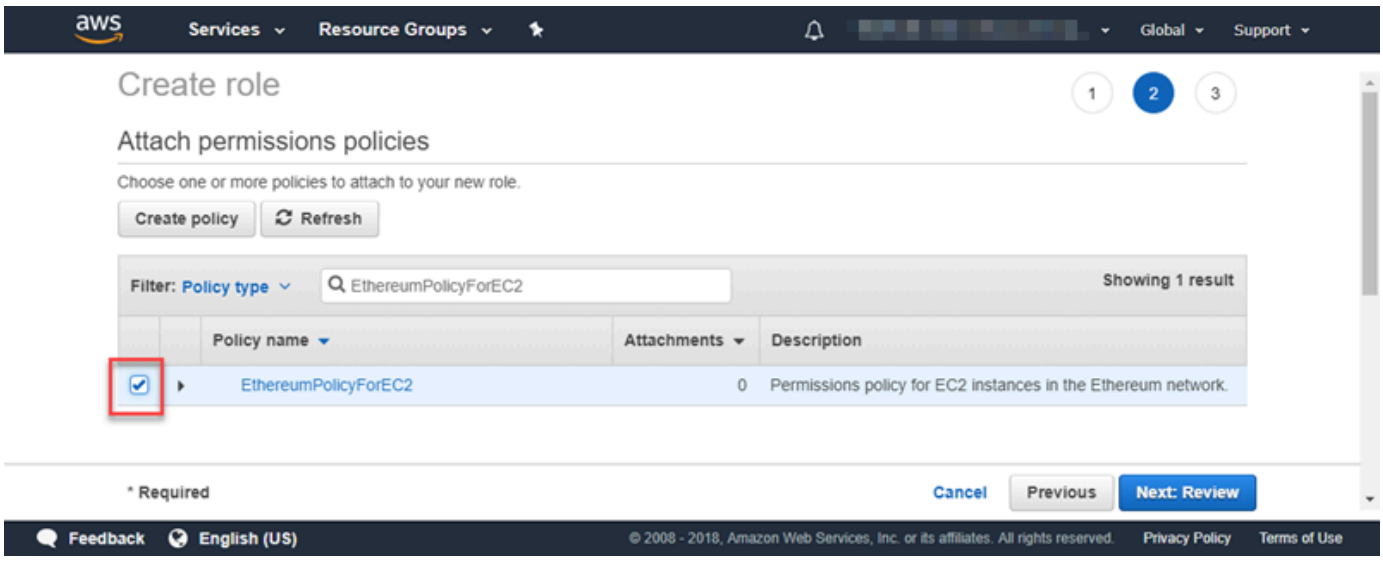
1. 탐색 창에서 정책(Policies)을 선택한 후 정책 생성(Create policy)을 선택합니다.
2. JSON을 선택하고 기본 정책 설명을 다음 JSON 정책으로 대체합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

3. 정책 검토를 선택합니다.
4. 이름에 이 권한 정책을 식별하는 데 도움이 되는 값을 입력합니다(예: EthereumPolicyForEC2). 설명에 간략한 설명을 입력합니다. 정책 생성을 선택합니다.

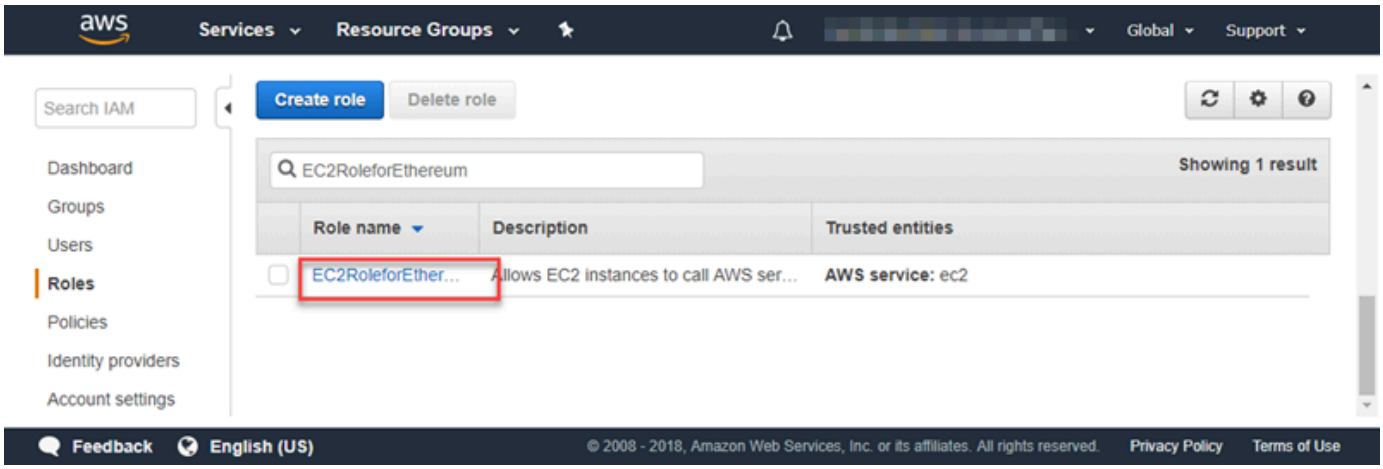


5. 역할(Roles)과 역할 생성(Create role)을 차례로 선택합니다.
6. EC2, 다음: 권한을 선택합니다.
7. 검색 필드에 이전에 생성한 권한 정책의 이름을 입력합니다(예: EthereumPolicyForEC2).
8. 이전에 생성한 정책에 대한 확인란을 선택하고 다음: 검토를 선택합니다.

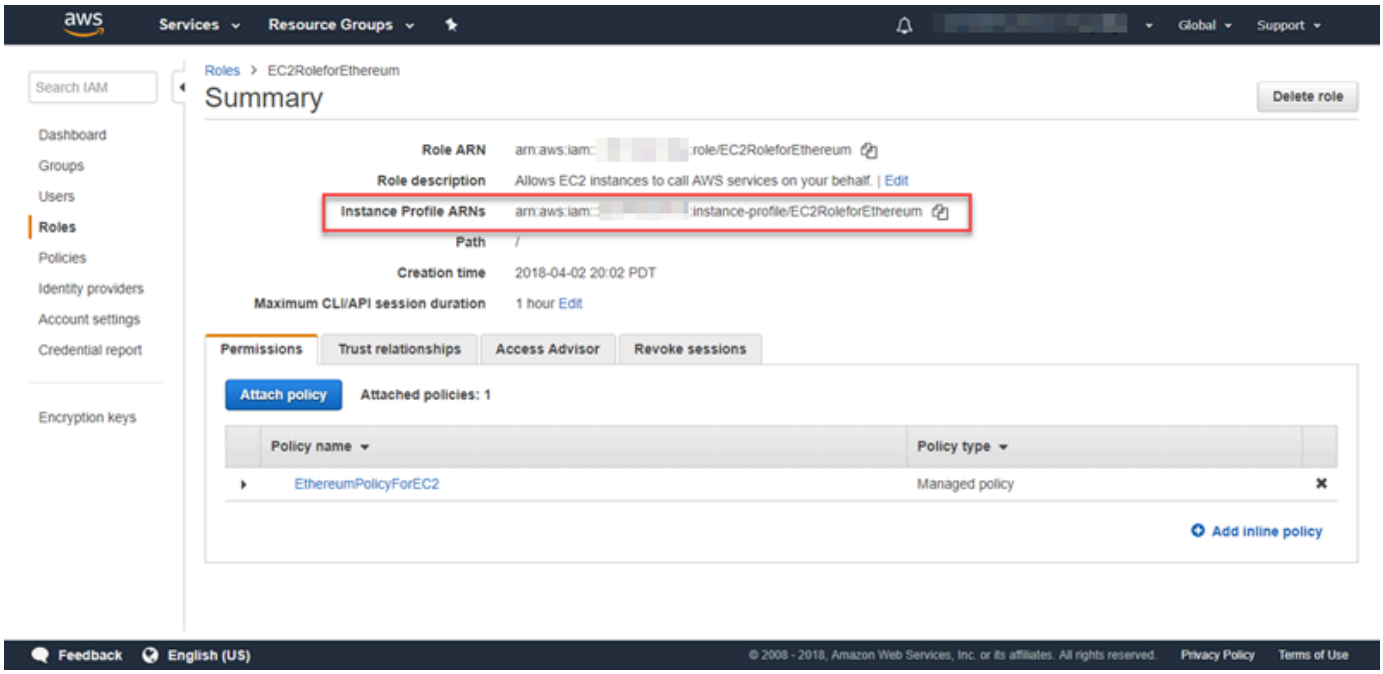


9. 역할 이름에 이 역할을 식별하는 데 도움이 되는 값을 입력합니다(예: EC2RoleForEthereum). 역할 설명에 간략한 설명을 입력하고 역할 생성을 선택합니다.

10. 목록에서 방금 생성한 역할을 선택합니다. 계정에 여러 역할이 있는 경우 검색 필드에 역할 이름을 입력할 수 있습니다.



11. 인스턴스 프로파일 ARN 값을 복사하고 다시 복사할 수 있도록 저장합니다. Ethereum 네트워크를 생성할 때 이 ARN이 필요합니다.



Bastion Host 생성

이 자습서에서는 Bastion Host를 생성합니다. 이 인스턴스는 Ethereum 네트워크에서 웹 인터페이스 및 인스턴스에 연결하는 데 사용하는 EC2 인스턴스입니다. 유일한 목적은 VPC 외부의 신뢰할 수 있는 클라이언트에서 오는 SSH 트래픽을 전달하여 Ethereum 네트워크 리소스에 액세스할 수 있도록 하는 것입니다.

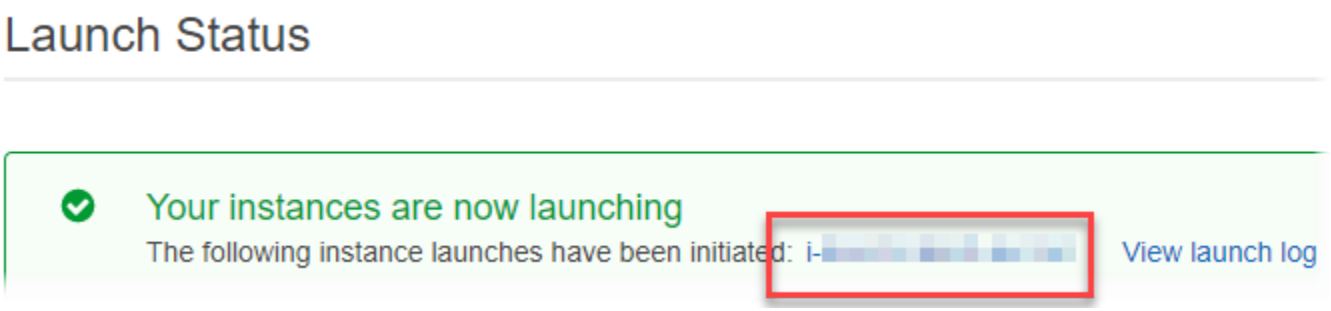
템플릿에서 생성하는 Application Load Balancer가 내부적이기 때문에, 즉 내부 IP 주소만 라우팅하기 때문에 Bastion Host를 설정합니다. Bastion Host:

- 이전에 생성한 두 번째 퍼블릭 서브넷에서 시작하므로 Application Load Balancer가 인식하는 내부 IP 주소가 있습니다.
- 서브넷이 할당하는 퍼블릭 IP 주소가 있으며, VPC 외부의 신뢰할 수 있는 소스에서 액세스할 수 있습니다.
- 이전에 생성한 Application Load Balancer에 대한 보안 그룹과 연결되어 있으며, 신뢰할 수 있는 클라이언트에서 오는 SSH 트래픽(포트 22)을 허용하는 인바운드 규칙이 있습니다.

Ethereum 네트워크에 액세스 할 수 있으려면 신뢰할 수 있는 클라이언트가 Bastion Host를 통해 연결하도록 설정해야 합니다. 자세한 내용은 Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결 단원을 참조하십시오. Bastion Host는 하나의 접근 방식입니다. 신뢰할 수 있는 클라이언트에서 VPC 내의 프라이빗 리소스에 대한 액세스를 제공하는 모든 접근 방식을 사용할 수 있습니다.

Bastion Host를 생성하려면

1. Amazon EC2 사용 설명서의 처음 5단계에 따라 인스턴스를 시작합니다.
2. [Edit Instance Details]를 선택합니다. 네트워크에서 이전에 생성한 VPC를 선택하고, 서브넷에서 이전에 생성한 두 번째 퍼블릭 서브넷을 선택합니다. 다른 모든 설정은 기본값으로 그대로 둡니다.
3. 메시지가 표시되면 변경 내용을 확인한 다음 검토 후 시작을 선택합니다.
4. 보안 그룹 편집을 선택합니다. 보안 그룹 할당에서 Select an existing security group(기존 보안 그룹 선택)을 선택합니다.
5. 보안 그룹 목록에서 이전에 생성한 Application Load Balancer에 대한 보안 그룹을 선택한 다음 검토 후 시작을 선택합니다.
6. 시작을 선택합니다.
7. 인스턴스 ID를 기록해 둡니다. 나중에 Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결할 때 이 항목이 필요합니다.



Ethereum 네트워크 생성

이 주제의 템플릿을 사용하여 지정하는 Ethereum 네트워크는 Ethereum 네트워크에 대한 EC2 인스턴스의 Amazon ECS 클러스터를 생성하는 CloudFormation 스택을 시작합니다. 이 템플릿은 사전 조건 설정에서 이전에 생성한 리소스를 이용합니다.

템플릿을 사용하여 CloudFormation 스택을 시작하면 일부 작업에 대해 중첩 스택이 생성됩니다. 완료한 후 Bastion Host를 통해 네트워크의 Application Load Balancer에서 서비스되는 리소스에 연결하여 Ethereum 네트워크가 실행 중이고 액세스 가능한지 확인할 수 있습니다.

Ethereum용 AWS 블록체인 템플릿을 사용하여 Ethereum 네트워크를 만들려면

1. AWS Blockchain 템플릿 시작하기를 참조하고 AWS 리전의 빠른 링크를 사용하여 CloudFormation 콘솔에서 Ethereum용 최신 AWS Blockchain 템플릿을 엽니다.
2. 다음 지침을 따라 값을 입력합니다.

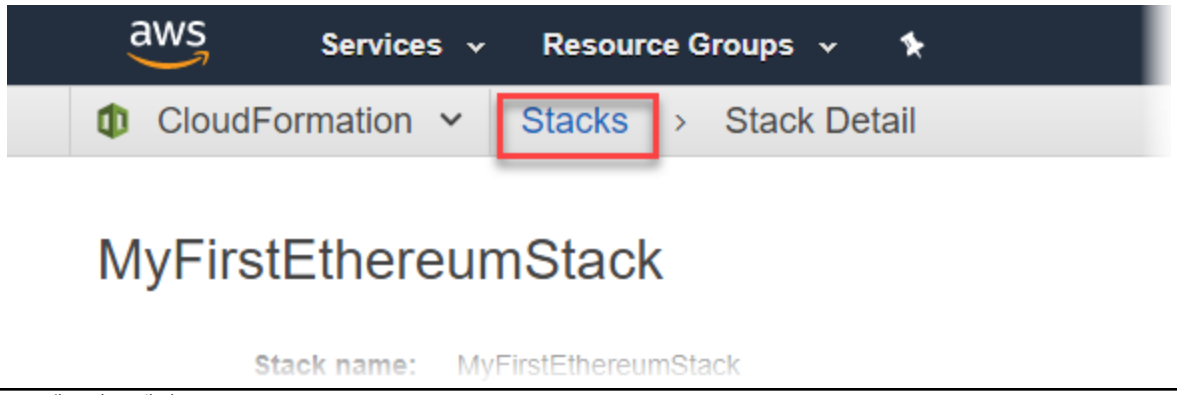
   - 스택 이름에 쉽게 식별할 수 있는 이름을 입력합니다. 이 이름은 스택에서 생성한 리소스의 이름 내에서 사용됩니다.
   - Ethereum 네트워크 파라미터 및 프라이빗 Ethereum 네트워크 파라미터에서 기본 설정을 그대로 둡니다.

     > **Warning**
     >
     > 기본 계정 및 관련 니모닉 구문은 테스트 용도로만 사용하십시오. 기본 계정 집합을 사용하여 실제 Ether를 전송하지 마십시오. 니모닉 구문의 액세스 권한이 있는 사람은 계정의 Ether에 액세스하거나 Ether를 도용할 수 있습니다. 대신 프로덕션 용도로 사용자 지정 계정을 지정하십시오. 기본 계정과 연결된 니모닉 구문은 `outdoor father modify clever trophy abandon vital feel portion grit evolve twist`입니다.

   - 플랫폼 구성에서 기본 설정을 그대로 둡니다. 그러면 EC2 인스턴스의 Amazon ECS 클러스터가 생성됩니다. 그 대신 docker-local을 선택하면 단일 EC2 인스턴스를 사용하여 Ethereum 네트워크가 생성됩니다.
   - EC2 구성에서 다음 지침에 따라 옵션을 선택합니다.
     - EC2 키 페어에서 키 페어를 선택합니다. 키 페어 생성에 대한 자세한 내용은 키 페어 생성 단원을 참조하십시오.
     - EC2 보안 그룹에서 보안 그룹 생성에서 이전에 생성한 보안 그룹을 선택합니다.

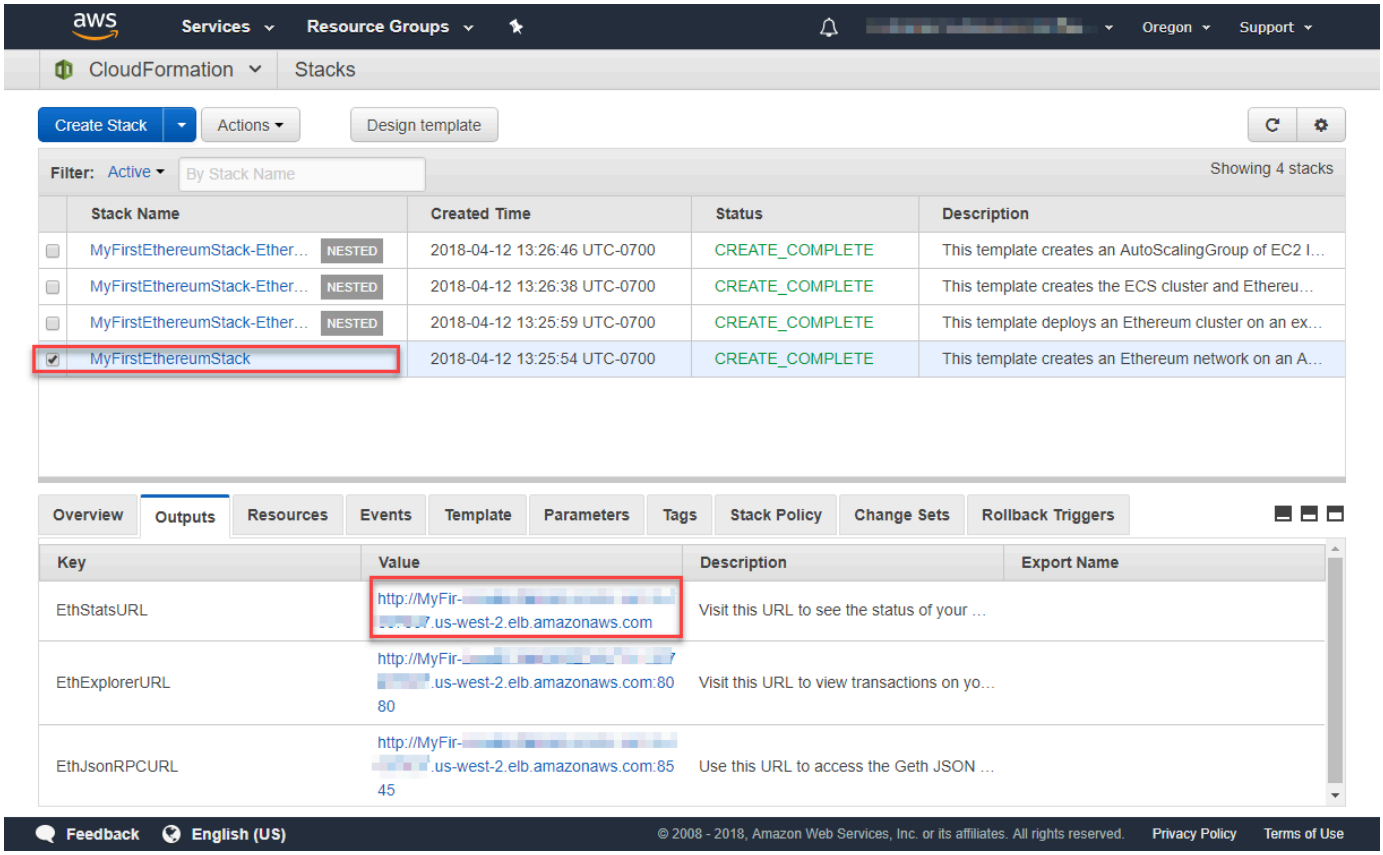
    - EC2 인스턴스 프로파일 ARN에서 Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할 생성에서 이전에 생성한 인스턴스 프로파일의 ARN을 입력합니다.
  - VPC 네트워크 구성에서 다음 지침에 따라 옵션을 선택합니다.
    - VPC ID에서 VPC 및 서브넷 생성에서 이전에 생성한 VPC를 선택합니다.
    - Ethereum 네트워크 서브넷 ID에서 To create the VPC 절차에서 이전에 생성한 단일 프라이빗 서브넷을 선택합니다.
  - ECS 클러스터 구성에서 기본값을 그대로 둡니다. 이렇게 하면 EC2 인스턴스 3개로 구성된 ECS 클러스터가 생성됩니다.
  - Application Load Balancer 구성(ECS만 해당)에서 다음 지침에 따라 옵션을 선택합니다.
    - Application Load Balancer 서브넷 ID에서 이전에 기록해 둔 list of subnets의 퍼블릭 서브넷 두 개를 선택합니다.
    - Application Load Balancer 보안 그룹에서 보안 그룹 생성에서 이전에 생성한 Application Load Balancer에 대한 보안 그룹을 선택합니다.
    - IAM 역할에 Amazon ECS 및 EC2 인스턴스 프로파일에 대한 IAM 역할 생성에서 이전에 생성한 ECS 역할의 ARN을 입력합니다.
  - EthStats에서 다음 지침에 따라 옵션을 선택합니다.
    - EthStats 배포에서 기본 설정인 true를 그대로 둡니다.
    - EthStats 연결 암호에 최소 6자 이상의 임의값을 입력합니다.
  - EthExplorer에서 EthExplorer 배포의 기본 설정인 true를 그대로 둡니다.
  - 기타 파라미터에서 중첩 템플릿 S3 URL 접두사의 기본값을 그대로 두고 기록해 둡니다. 여기서 중첩 템플릿을 찾을 수 있습니다.
3. 다른 모든 설정을 기본값으로 그대로 두고 승인 확인란을 선택한 다음 생성을 선택합니다.

   가 CloudFormation 시작하는 루트 스택의 스택 세부 정보 페이지가 나타납니다.
4. 루트 스택 및 중첩 스택의 진행 상황을 모니터링하려면 스택을 선택합니다.



5. 스테이터스에 모든 스택이 CREATE_COMPLETE로 표시되면 Ethereum 사용자 인터페이스에 연결하여 네트워크가 실행 중이고 액세스 가능한지 확인할 수 있습니다. ECS 컨테이너 플랫폼을 사용할 때 Application Load Balancer를 통해 EthStats, EthExplorer 및 EthJsonRPC에 연결하는 URL을 루트 스택의 출력 탭에서 사용할 수 있습니다.

> **Important**
>
> 클라이언트 컴퓨터에서 Bastion Host를 통해 프록시 연결을 설정할 때까지 이러한 URL 또는 SSH에 직접 연결할 수 없습니다. 자세한 내용은 Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결 단원을 참조하십시오.



Bastion Host를 사용하여 EthStats 및 EthExplorer에 연결

이 자습서에서 Ethereum 리소스에 연결하려면 Bastion Host를 통해 SSH 포트 전달(SSH 터널링)을 설정합니다. 다음 지침에서는 브라우저를 사용하여 EthStats 및 EthExplorer URL에 연결할 수 있도록 이 작업을 수행하는 방법을 보여 줍니다. 아래 지침에서는 먼저 로컬 포트에서 SOCKS 프록시를 설정합

니다. 그런 다음 브라우저 확장인 FoxyProxy를 사용하여 Ethereum 네트워크 URL에 대해 이 전달된 포트를 사용합니다.

Mac OS 또는 Linux를 사용하는 경우 SSH 클라이언트를 사용하여 Bastion Host에 대한 SOCKS 프록시 연결을 설정합니다. Windows 사용자인 경우 PuTTY를 사용합니다. 연결하기 전에 사용 중인 클라이언트 컴퓨터가 이전에 설정한 Application Load Balancer의 보안 그룹에서 인바운드 SSH 트래픽의 허용된 소스로 지정되었는지 확인합니다.

SSH를 사용하여 SSH 포트 전달을 통해 Bastion Host에 연결하려면

- Amazon EC2 사용 설명서의 SSH를 사용하여 Linux 인스턴스에 연결의 절차를 따릅니다. Linux 인스턴스에 연결하기 절차의 4단계에서 `-D 9001`를 SSH 명령에 추가하고, Ethereum용 AWS 블록체인 템플릿 구성에서 지정한 것과 동일한 키 페어를 지정한 다음, Bastion Host의 DNS 이름을 지정합니다.

```
ssh -i /path/my-template-key-pair.pem ec2-user@bastion-host-dns -D 9001
```

PuTTY(Windows)를 사용하여 SSH 포트 전달을 통해 Bastion Host에 연결하려면

1. Ethereum용 AWS Blockchain 템플릿 구성에서 지정한 것과 동일한 키 페어를 사용하여 PuTTY 세션 시작 절차의 7단계까지 Amazon EC2 사용 설명서의 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결의 절차를 따릅니다. Amazon EC2 PuTTY
2. PuTTY의 범주에서 연결, SSH, 터널을 선택합니다.
3. 포트 전달에서 로컬 포트가 다른 호스트의 연결 수락을 선택합니다.
4. 전달된 새 포트 추가에서 다음을 수행합니다.

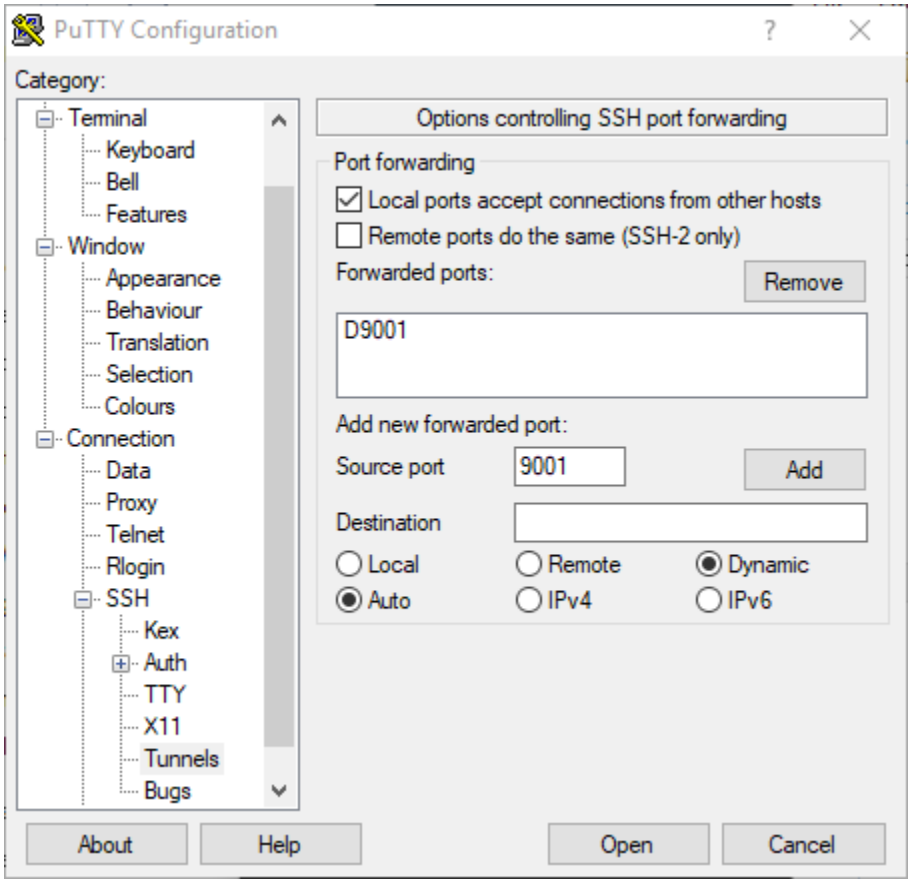
   a. 소스 포트에 9001을 입력합니다. 이 포트는 여기서 선택한 임의의 미사용 포트이므로 필요한 경우 다른 포트를 선택할 수 있습니다.

   b. 대상을 비워 둡니다.

   c. 동적을 선택합니다.

   d. 추가를 선택합니다.

   전달된 포트에서 D9001이 아래와 같이 나타나야 합니다.



5. 열기를 선택한 다음 키 구성에 따라 필요한 대로 Bastion Host에 대해 인증합니다. 연결을 열어 둡니다.

PuTTY 연결을 열어 두고, 이제 Ethereum 네트워크 URL의 전달된 포트를 사용하도록 시스템 또는 브라우저 확장을 구성합니다. 다음 지침에서는 이전에 전달된 포트로 설정한 EthStats 및 EthExplorer와 포트 9001의 URL 패턴을 기반으로 FoxyProxy Standard를 사용하여 연결을 전달하지만, 각자 선호하는 방법을 사용할 수 있습니다.

Ethereum 네트워크 URL에 SSH 터널을 사용하도록 FoxyProxy를 구성하려면

이 절차는 Chrome을 기반으로 작성되었습니다. 다른 브라우저를 사용하는 경우 설정 및 시퀀스를 해당 브라우저의 FoxyProxy 버전으로 변환합니다.

1. FoxyProxy Standard 브라우저 확장을 다운로드하여 설치한 다음, 사용 중인 브라우저의 지침에 따라 옵션을 엽니다.
2. 새 프록시 추가를 선택합니다.
3. 일반 탭에서 프록시가 활성인지 확인하고 이 프록시 구성을 식별하는 데 도움이 되는 프록시 이름 및 프록시 참고 사항을 입력합니다.

4. 프록시 세부 정보 탭에서 수동 프록시 구성을 선택합니다. 호스트 또는 IP 주소(또는 일부 버전의 경우 서버 또는 IP 주소)에 localhost를 입력합니다. 포트에 9001을 입력합니다. SOCKS 프록시?를 선택합니다.
5. URL 패턴 탭에서 새 패턴 추가를 선택합니다.
6. 패턴 이름에 식별하기 쉬운 이름을 입력하고 URL 패턴에 템플릿을 사용하여 생성한 모든 Ethereum 리소스 URL과 일치하는 패턴을 입력합니다(예: http://internal-MyUser-LoadB-*). URL 보기에 대한 자세한 내용은 Ethereum URLs 단원을 참조하십시오.
7. 다른 설정의 기본 선택 사항을 그대로 두고 저장을 선택합니다.

이제 Ethereum URL에 연결할 수 있습니다. 이 URL은 템플릿으로 생성한 루트 스택의 출력 탭을 사용하여 CloudFormation 콘솔에서 확인할 수 있습니다.

리소스 정리

CloudFormation 를 사용하면 스택이 생성한 리소스를 쉽게 정리할 수 있습니다. 스택을 삭제할 때 스택이 생성한 모든 리소스가 삭제됩니다.

템플릿이 생성한 리소스를 삭제하려면

- CloudFormation 콘솔을 열고 이전에 생성한 루트 스택을 선택한 다음 작업, 삭제를 선택합니다.

  이전에 생성한 루트 스택 및 연결된 중첩 스택의 상태가 DELETE_IN_PROGRESS로 업데이트됩니다.

Ethereum 네트워크에 대해 생성한 사전 조건을 삭제할 수 있습니다.

VPC 삭제

- Amazon VPC 콘솔을 열고 이전에 생성한 VPC를 선택한 다음 작업, VPC 삭제를 선택합니다. 이를 통해 VPC와 연결된 서브넷, 보안 그룹 및 NAT 게이트웨이가 삭제됩니다.

IAM 역할 및 EC2 인스턴스 프로파일 삭제

- IAM 콘솔을 열고 역할을 선택합니다. ECS에 대한 역할과 이전에 생성한 EC2에 대한 역할을 선택하고 삭제를 선택합니다.

Bastion Host에 대한 EC2 인스턴스 종료

- Amazon EC2 대시보드를 열고 실행 중인 인스턴스를 선택한 다음, Bastion Host에 대해 생성한 EC2 인스턴스를 선택하고 작업, 인스턴스 상태, 종료를 선택합니다.

AWS Blockchain 템플릿 및 기능

이 섹션에서는 블록체인 네트워크 생성을 즉시 시작할 수 있는 링크와 AWS에서 네트워크를 설정하기 위한 구성 옵션 및 사건 조건에 대한 정보를 제공합니다.

다음의 템플릿을 사용할 수 있습니다:

- Ethereum용 AWS Blockchain 템플릿
- Hyperledger Fabric용 AWS Blockchain 템플릿

AWS Blockchain 템플릿은 다음 리전에서 사용할 수 있습니다:

- 미국 서부(오레곤) 리전(us-west-2)
- 미국 동부(버지니아 북부) 리전(us-east-1)
- 미국 동부(오하이오) 리전(us-east-2)

> **Note**
>
> 위에 열거되지 않은 리전에서 템플릿을 실행하면 미국 동부(버지니아 북부) 리전(us-east-1)의 리소스를 시작합니다.

Ethereum용 AWS Blockchain 템플릿 사용

Ethereum은 블록 체인 프레임워크로, 이더리움 관련 언어인 Solidity를 사용하여 스마트 계약을 실행합니다. Homestead는 Ethereum의 최신 릴리스입니다. 자세한 내용은 Ethereum Homestead 설명서 및 Solidity 설명서를 참조하십시오.

시작 링크

Ethereum 템플릿을 사용하여 특정 리전에서 시작할 수 있는 링크는 AWS Blockchain 템플릿 시작하기를 참조하세요. CloudFormation

Ethereum 옵션

템플릿을 사용하여 Ethereum 네트워크를 구성할 때, 선택하는 옵션에 따라 후속 요구 사항이 결정됩니다.

- 컨테이너 플랫폼 선택
- 프라이빗 또는 퍼블릭 Ethereum 네트워크 선택
- 기본 계정 및 니모닉 문구 변경

컨테이너 플랫폼 선택

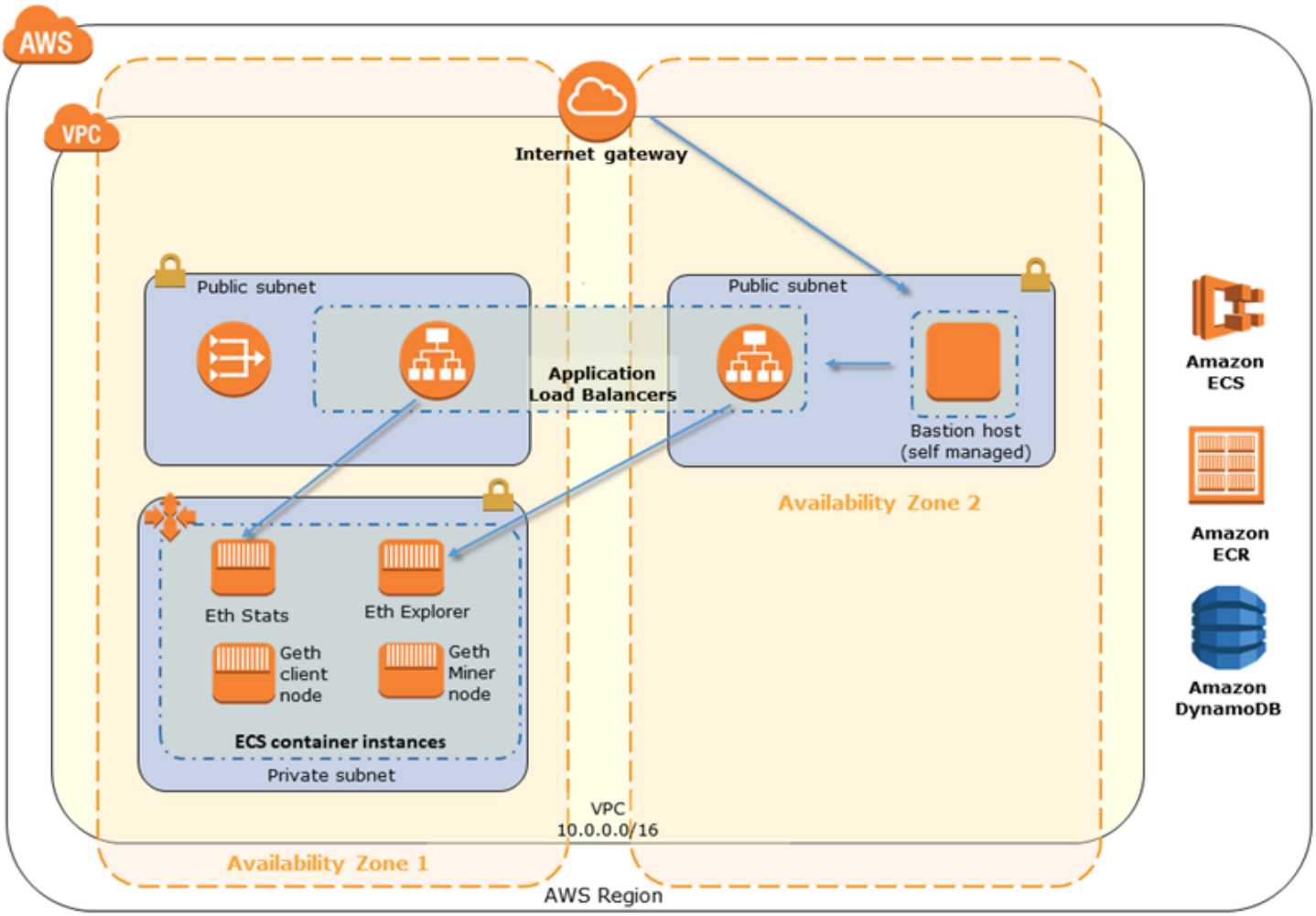
AWS Blockchain 템플릿은 Amazon ECR에 저장된 Docker 컨테이너를 사용하여 블록체인 소프트웨어를 배포합니다. Ethereum용 AWS Blockchain 템플릿은 컨테이너 플랫폼을 위한 두 가지 선택 사항을 제공합니다.

- ecs—Ethereum이 Amazon EC2 인스턴스의 Amazon ECS 클러스터에서 실행되도록 지정합니다.
- docker-local—Ethereum이 단일 EC2 인스턴스에서 실행하도록 지정합니다.

Amazon ECS 컨테이너 플랫폼 사용

Amazon ECS를 사용하여 Application Load Balancer 및 관련 리소스가 있고 여러 개의 EC2 인스턴스로 구성된 ECS 클러스터에서 Ethereum 네트워크를 생성합니다. Amazon ECS 구성 사용에 대한 자세한 내용은 AWS Blockchain 템플릿 시작하기 자습서를 참조하십시오.

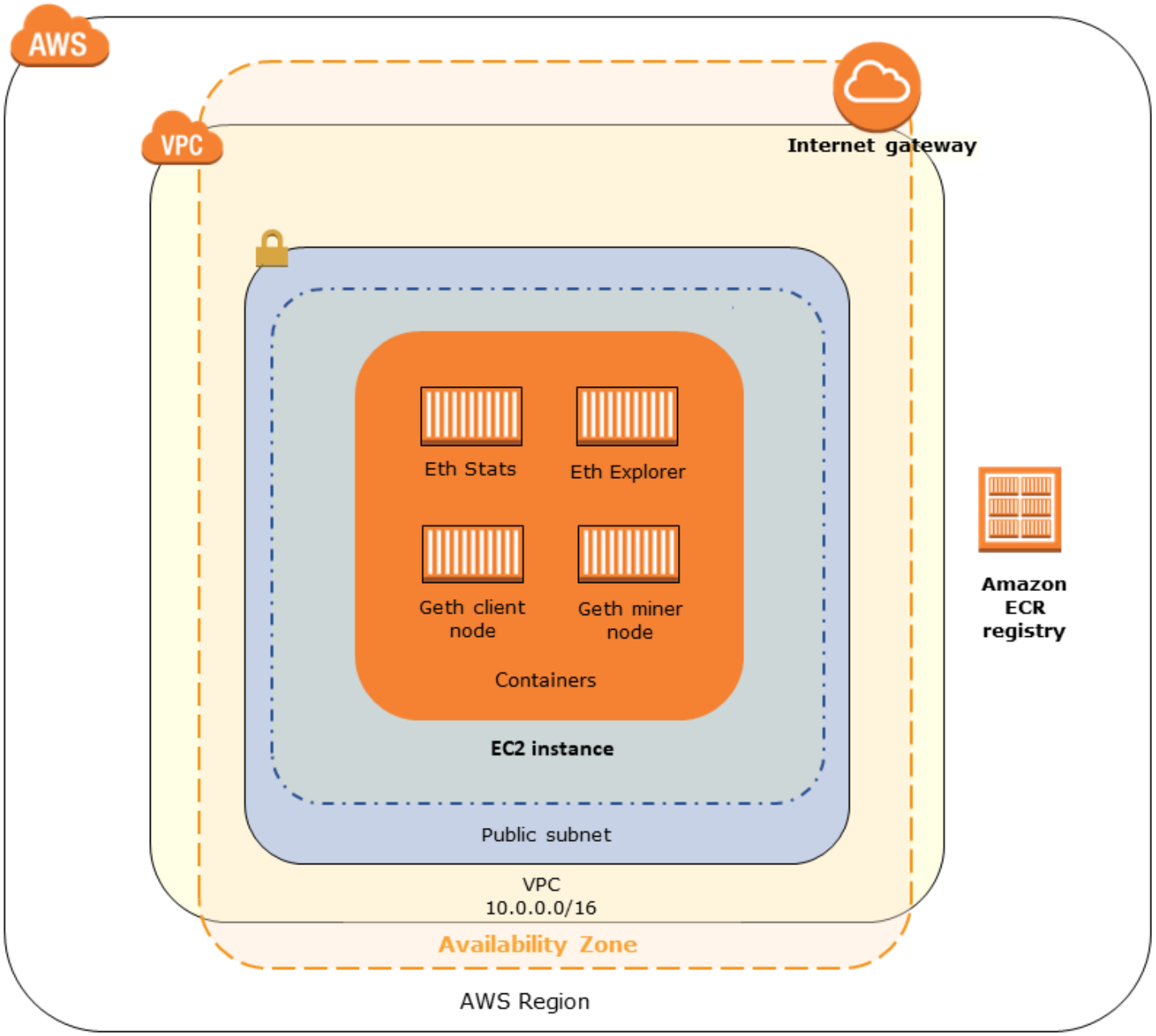
다음 다이어그램은 ECS 컨테이너 플랫폼 옵션과 함께 템플릿을 사용하여 생성된 Ethereum 네트워크를 보여줍니다:



docker-local 플랫폼 사용

또는 단일 Amazon EC2 인스턴스 내에 있는 Ethereum 컨테이너를 시작할 수 있습니다. 모든 컨테이너는 단일 EC2 인스턴스에서 실행됩니다. 이는 간소화된 설정입니다.

다음 다이어그램은 docker-local 컨테이너 플랫폼 옵션과 함께 템플릿을 사용하여 생성된 Ethereum 네트워크를 보여줍니다:



프라이빗 또는 퍼블릭 Ethereum 네트워크 선택

1—4 이외의 Ethereum 네트워크 ID 값을 선택하면, 지정한 프라이빗 네트워크 매개 변수를 사용하여, 사용자가 정의한 네트워크 내에서 실행되는 프라이빗 Ethereum 노드가 생성됩니다.

1~4 중에서 Ethereum 네트워크 ID를 선택하면 생성한 Ethereum 노드가 퍼블릭 Ethereum 네트워크에 연결됩니다. 프라이빗 네트워크 설정 및 그 기본값은 무시할 수 있습니다. Ethereum 노드를 퍼블릭 Ethereum 네트워크에 연결하고자 하는 경우 네트워크 내 해당 서비스가 인터넷에 액세스할 수 있어야 합니다.

기본 계정 및 니모닉 문구 변경

니모닉 문구는 어떤 네트워크에서든 연결된 계정의 Ethereum 지갑(즉, 프라이빗/퍼블릭 키 페어)을 생성하는 데 사용할 수 있는 임의의 단어 집합입니다. 니모닉 구문은 연결된 계정의 Ether에 액세스하는 데 사용할 수 있습니다. Ethereum 템플릿이 사용하는 기본 계정과 연결된 기본 니모닉을 생성했습니다.

> **Warning**
>
> 기본 계정 및 관련 니모닉 구문은 테스트 용도로만 사용하십시오. 기본 계정 집합을 사용하여 실제 Ether를 전송하지 마십시오. 니모닉 구문의 액세스 권한이 있는 사람은 계정의 Ether에 액세스하거나 Ether를 도용할 수 있습니다. 대신 프로덕션 용도로 사용자 지정 계정을 지정하십시오. 기본 계정과 연결된 니모닉 구문은 `outdoor father modify clever trophy abandon vital feel portion grit evolve twist`입니다.

사전 조건

Ethereum용 AWS Blockchain 템플릿을 사용하여 Ethereum 네트워크를 설정하는 경우 아래 열거된 최소 요구 사항을 충족해야 합니다. 템플릿에는 다음 각 범주에 나열된 AWS 구성 요소가 필요합니다.

주제

- Ethereum 리소스에 액세스하기 위한 사전 조건
- IAM 사전 조건
- 보안 그룹 사전 조건
- VPC 사전 조건
- EC2 인스턴스 프로파일 및 ECS 역할에 대한 IAM 권한 예제

Ethereum 리소스에 액세스하기 위한 사전 조건

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
EC2 인스턴스에 액세스하는 데 사용할 수 있는 Amazon EC2 키 페어입니다. 키는 ECS	✓	✓

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
클러스터 및 다른 리소스와 동일한 리전에 있어야 합니다.		
애플리케이션 로드 밸런서로 들어가는 트래픽이 허용되는 내부 주소가 있는 접속 호스트 또는 인터넷 연결 로드 밸런서와 같은 인터넷 연결 구성 요소입니다. ECS 플랫폼에서는 보안상의 이유로 템플릿에서 내부 로드 밸런서가 생성되기 때문에 이 구성 요소가 필요합니다. docker-local 플랫폼에서는 EC2 인스턴스가 프라이빗 서브넷(권장됨)에 있을 때 이 구성 요소가 필요합니다. 접속 호스트 구성에 대한 자세한 내용은 Bastion Host 생성 단원을 참조하십시오.	✓	✓ (프라이빗 서브넷 포함)

IAM 사전 조건

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
모든 관련 서비스에서 작업할 권한을 가지고 있는 IAM 보안 주체(사용자 또는 그룹)입니다.	✓	✓
EC2 인스턴스에서 다른 서비스와 상호 작용할 수 있는 적절한 권한이 있는 Amazon EC2 인스턴스 프로파일입니다. 자세한 내용은 To create an EC2	✓	✓

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
instance profile 단원을 참조하십시오.		
Amazon ECS에서 다른 서비스와 상호 작용할 수 있는 권한이 있는 IAM 역할입니다. 자세한 내용은 ECS 역할 및 권한 생성 단원을 참조하십시오.	✓	

보안 그룹 사전 조건

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
EC2 인스턴스에 대한 보안 그룹의 요건은 다음과 같습니다.	✓	✓
• 0.0.0.0/0으로의 트래픽을 허용하는 아웃바운드 규칙(기본값).	✓	✓
• 자체(동일한 보안 그룹)에서 오는 모든 트래픽을 허용하는 인바운드 규칙.	✓	✓
• Application Load Balancer에 대한 보안 그룹에서 오는 모든 트래픽을 허용하는 인바운드 규칙입니다.	✓	
• 클라이언트 컴퓨터의 IP CIDR과 같은 신뢰할 수 있는 외부 소스에서 오는 HTTP(포트 80), ETHStats(포트 8080에서 서비스됨), HTTP를 통한 JSON RPC(포트 8545) 및 SSH(포트 22)를		✓

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
허용하는 인바운드 규칙입니다.		
Application Load Balancer에 대한 보안 그룹의 요건은 다음과 같습니다. • 자체(동일한 보안 그룹)에서 오는 모든 트래픽을 허용하는 인바운드 규칙. • EC2 인스턴스에 대한 보안 그룹의 모든 트래픽을 허용하는 인바운드 규칙. • EC2 인스턴스에 대한 보안 그룹으로 가는 모든 트래픽만을 허용하는 아웃바운드 규칙. 자세한 내용은 보안 그룹 생성 단원을 참조하십시오. • 이 동일한 보안 그룹을 접속 호스트와 연결하는 경우 신뢰할 수 있는 소스에서 오는 SSH(포트 22) 트래픽을 허용하는 인바운드 규칙. • 접속 호스트 또는 기타 인터넷 연결 구성 요소가 다른 보안 그룹에 있는 경우 해당 구성 요소에서 오는 트래픽을 허용하는 인바운드 규칙.	✓	

VPC 사전 조건

사전 조건	ECS 플랫폼의 경우	docker-local의 경우
Ethereum 서비스에 액세스하는 데 사용되는 Elastic IP 주소입니다.	✓	✓
EC2 인스턴스를 실행할 서브넷. 프라이빗 서브넷을 권장합니다.	✓	✓
공개적으로 액세스할 수 있는 두 개의 서브넷입니다. 각 서브넷은 서로 다른 가용 영역에 있어야 하며, 하나는 EC2 인스턴스의 서브넷과 동일한 가용 영역에 있어야 합니다.	✓	

EC2 인스턴스 프로파일 및 ECS 역할에 대한 IAM 권한 예제

템플릿을 사용할 때 파라미터 중 하나로 EC2 인스턴스 프로파일 ARN을 지정합니다. ECS 컨테이너 플랫폼을 사용하는 경우 ECS 역할 ARN 또한 지정합니다. 이러한 역할에 연결된 권한 정책을 사용하면 클러스터의 AWS 리소스 및 인스턴스가 다른 AWS 리소스와 상호 작용할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요. 아래 정책 설명 및 절차를 권한 생성의 시작점으로 사용하십시오.

EC2 인스턴스 프로파일에 대한 권한 정책 예제

다음 권한 정책은 ECS 컨테이너 플랫폼 선택 시 EC2 인스턴스 프로필에 대해 허용되는 작업을 보여줍니다. `ecs` 컨텍스트 키를 제거하여 액세스를 제한하면 동일한 정책 설명을 docker-local 컨테이너 플랫폼에서 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

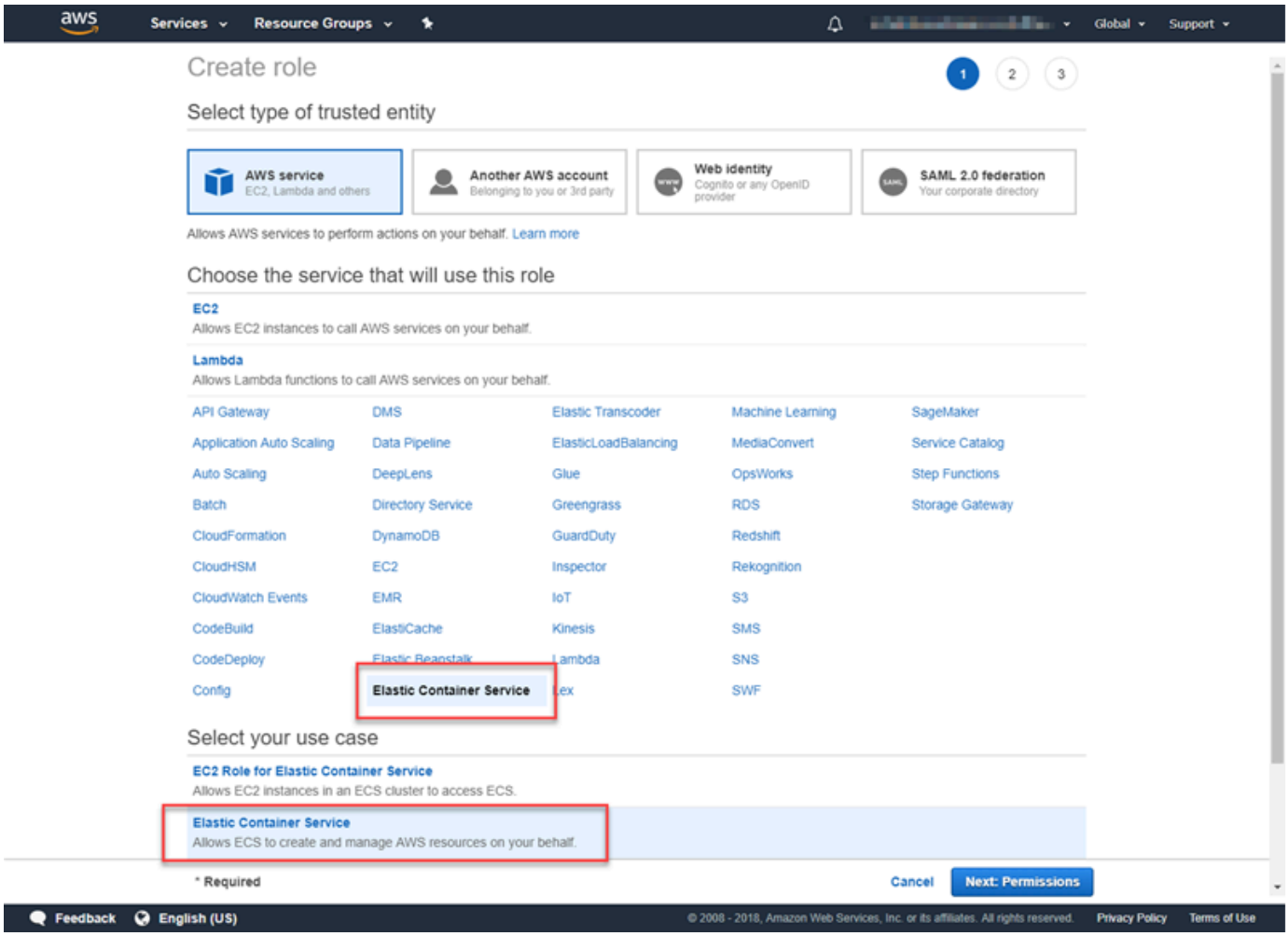
ECS 역할 및 권한 생성

ECS 역할에 연결된 권한의 경우 AmazonEC2ContainerServiceRole 권한 정책부터 시작하는 것이 좋습니다. 다음 절차를 사용하여 역할을 생성하고 이 권한 정책을 연결합니다. IAM 콘솔을 사용하여 이 정책의 가장 최신 권한을 봅니다.

Amazon ECS에 대한 IAM 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할, 역할 생성을 선택합니다.
3. Select type of trusted entity(신뢰할 수 있는 유형의 엔터티 선택) 아래에서 AWS 서비스를 선택합니다.
4. 이 역할을 사용할 서비스 선)에서 Elastic Container Service를 선택합니다.

5. [Select your use case]에서 [Elastic Container Service], [Next:Permissions]를 선택합니다.

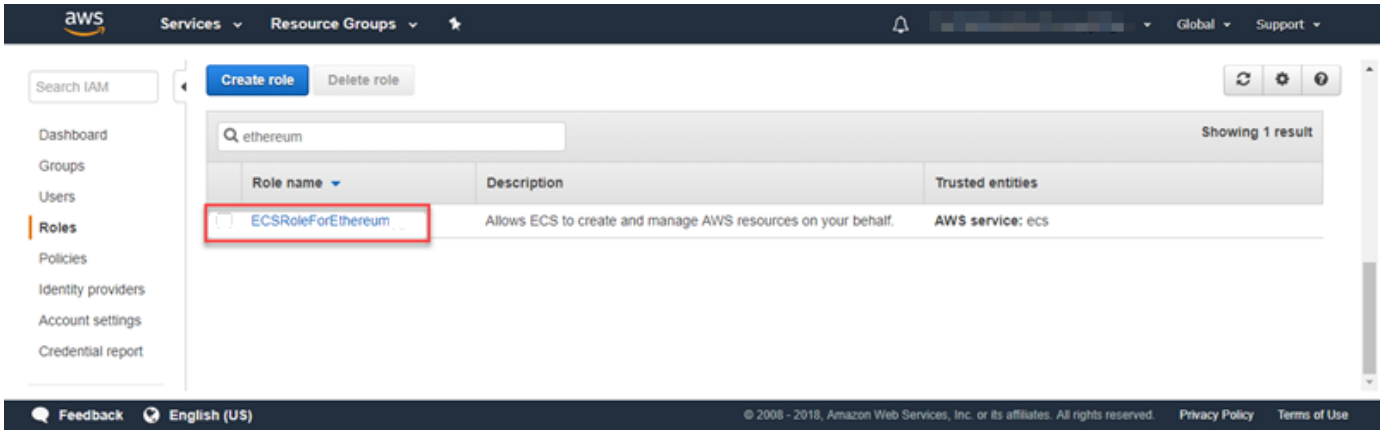


6. [Permissions policy]에서 기본 권한 정책([AmazonEC2ContainerServiceRole])을 선택한 상태로 유지하고 [Next:Review]를 선택합니다.

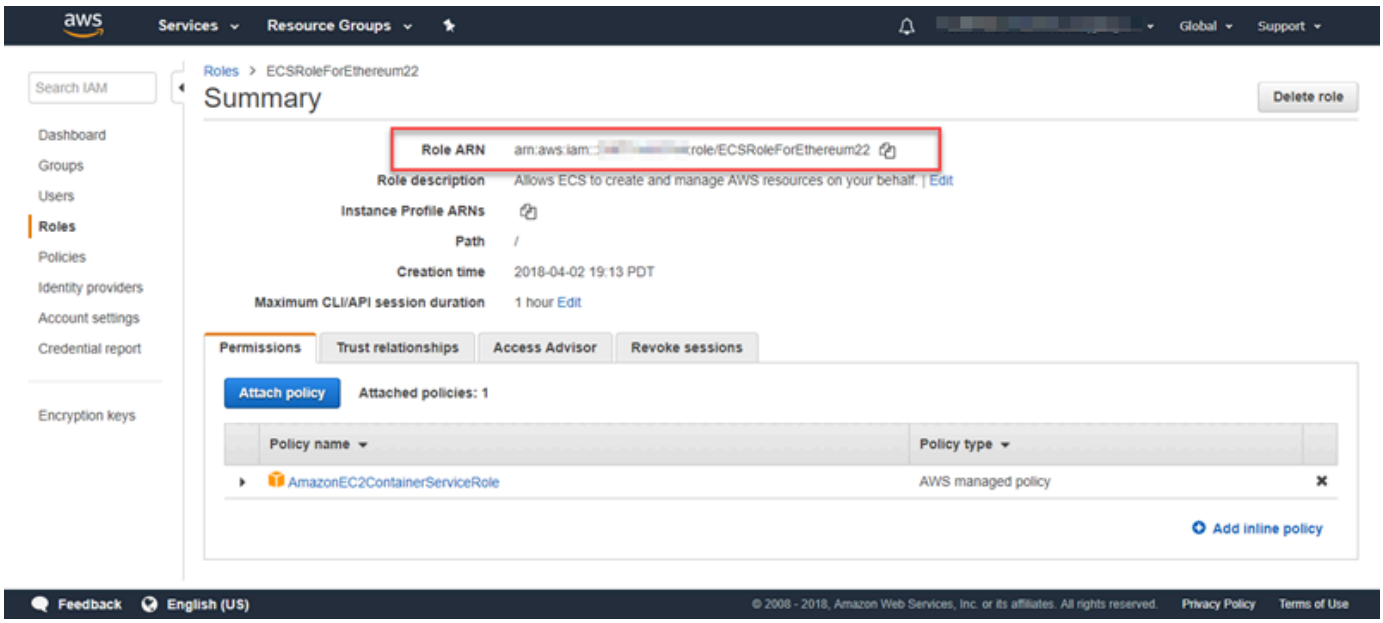
7. [Role name]에서 이 역할을 식별하는 데 도움이 되는 값을 입력합니다(예: ECSRoleForEthereum). [Role Description]에 간략한 설명을 입력합니다. 나중에 사용할 수 있도록 역할 이름을 기록해 두십시오.

8. 역할 생성을 선택합니다.

9. 목록에서 방금 생성한 역할을 선택합니다. 계정에 여러 역할이 있는 경우 역할 이름을 검색할 수 있습니다.



10. [Role ARN] 값을 복사하고 다시 복사할 수 있도록 저장합니다. Ethereum 네트워크를 생성할 때 이 ARN이 필요합니다.



Ethereum 리소스에 연결하기

템플릿으로 생성한 루트 스택에 CREATE_COMPLETE가 표시되면 CloudFormation 콘솔을 사용하여 Ethereum 리소스에 연결할 수 있습니다. 연결 방식은 선택한 컨테이너 플랫폼(ECS 또는 docker-local)에 따라 다릅니다.

- ECS—루트 스택의 출력 탭에는 Application Load Balancer에서 실행 중인 서비스의 링크가 제공됩니다. 이러한 URL은 보안 사유로 인해 직접 액세스할 수 없습니다. 연결하려면 접속 호스트를 설정하고 사용하여 이러한 URL에 대한 연결을 프록시할 수 있습니다. 자세한 내용은 아래 Bastion Host를 사용한 프록시 연결 섹션을 참조하세요.

- docker-local—아래 나열된 Ethereum 서비스를 호스팅하는 EC2 인스턴스의 IP 주소를 사용하여 연결합니다. EC2 콘솔을 사용하여 템플릿으로 생성된 인스턴스의 *ec2-IP-address*를 찾습니다.
  - EthStats—http://*ec2-IP-address* 사용
  - EthExplorer—http://*ec2-IP-address*:8080 사용
  - EthJsonRpc—http://*ec2-IP-address*:8545 사용

  Ethereum 네트워크 서브넷 ID(템플릿 내에서는 사용할 VPC 서브넷 목록)에 대해 퍼블릭 서브넷을 지정한 경우 직접 연결할 수 있습니다. 클라이언트는 나열된 포트는 물론 SSH(포트 22)에 대한 인바운드 트래픽의 신뢰할 수 있는 소스여야 합니다. 이는 Ethereum용 AWS 블록체인 템플릿을 사용하여 지정한 EC2 보안 그룹에 의해 결정됩니다.

  프라이빗 서브넷을 지정한 경우 접속 호스트를 설정하고 사용하여 이러한 주소에 대한 연결을 프록시할 수 있습니다. 자세한 내용은 아래 Bastion Host를 사용한 프록시 연결 섹션을 참조하세요.

Bastion Host를 사용한 프록시 연결

어떤 구성은 공용 Ethereum 서비스가 제공되지 않습니다. 이 경우 Bastion Host를 통해 Ethereum 리소스에 연결할 수 있습니다. Bastion Host에 대한 자세한 내용은 Linux Bastion Host 퀵 스타트 가이드의 Linux Bastion Host 아키텍처를 참조하십시오.

Bastion Host는 EC2 인스턴스입니다. 다음 요구 사항을 충족하는지 확인합니다:

- Bastion Host의 EC2 인스턴스는 퍼블릭 IP 자동 할당이 활성화되어 있고 인터넷 게이트웨이가 있는 퍼블릭 서브넷 내에 있습니다.
- Bastion Host에는 ssh 연결을 허용하는 키 쌍이 있습니다.
- Bastion Host는 연결하는 클라이언트의 인바운드 SSH 트래픽을 허용하는 보안 그룹과 연결됩니다.
- Ethereum 호스트에 할당된 보안 그룹(예: ECS가 컨테이너 플랫폼인 경우 또는 Application Load Balancer 또는 docker-local이 컨테이너 플랫폼인 경우 호스트 EC2 인스턴스)은 VPC 내 소스의 모든 포트에 인바운드 트래픽을 허용합니다.

Bastion Host를 설정한 후 연결하는 클라이언트가 Bastion Host를 프록시로 사용하는지 확인하십시오. 다음은 Mac OS를 사용하여 프록시 연결을 설정한 예입니다. *BastionIP*를 Bastion 호스트 EC2 인스턴스의 IP 주소로 대체하고 *MySshKey.pem*을 Bastion Host로 복제한 키 페어 파일로 대체합니다.

명령줄에 다음을 입력합니다:

```
ssh -i mySshKey.pem  ec2-user@BastionIP -D 9001
```

이렇게 하면 로컬 시스템의 포트 9001에 대한 포트 포워딩이 Bastion Host로 설정됩니다.

다음으로 `localhost:9001`에 대한 SOCKS 프록시를 사용하도록 브라우저 또는 시스템을 구성하십시오. 예를 들어, Mac OS의 경우, [System Preferences]와 [Network], [Advanced]를 선택하고 [SOCKS proxy]를 선택하여 localhost:9001을 입력합니다.

FoxyProxy Standard와 Chrome을 사용하여 [More Tools]와 [Extensions]를 선택합니다. [FoxyProxy Standard]에서 [Details]와 [Extension options], [Add New Proxy]를 선택합니다. [Manual Proxy Configuration]을 선택합니다. [Host or IP Address]에 localhost를 입력하고 [Port]에는 9001을 입력합니다. [SOCKS proxy?], [Save]를 선택합니다.

이제 템플릿 출력에 나열된 Ethereum 호스트 주소에 연결할 수 있습니다.

Hyperledger Fabric용 AWS Blockchain 템플릿 사용

Hyperledger Fabric은 Go로 작성되는 chaincode라는 스마트 계약을 실행하는 블록체인 프레임워크입니다. Hyperledger Fabric으로 프라이빗 네트워크를 생성하여 네트워크에 연결하고 참여할 수 있는 피어를 제한할 수 있습니다. Hyperledger Fabric에 대한 자세한 내용은 Hyperledger Fabric 설명서를 참조하십시오. 체인코드에 대한 자세한 내용은 Hyperledger Fabric 설명서의 개발자를 위한 체인코드 항목을 참조하십시오.

Hyperledger Fabric용 AWS Blockchain 템플릿은 docker-local 컨테이너 플랫폼만을 지원합니다. 따라서 Hyperledger Fabric 컨테이너는 단일 EC2 인스턴스에 배포됩니다.

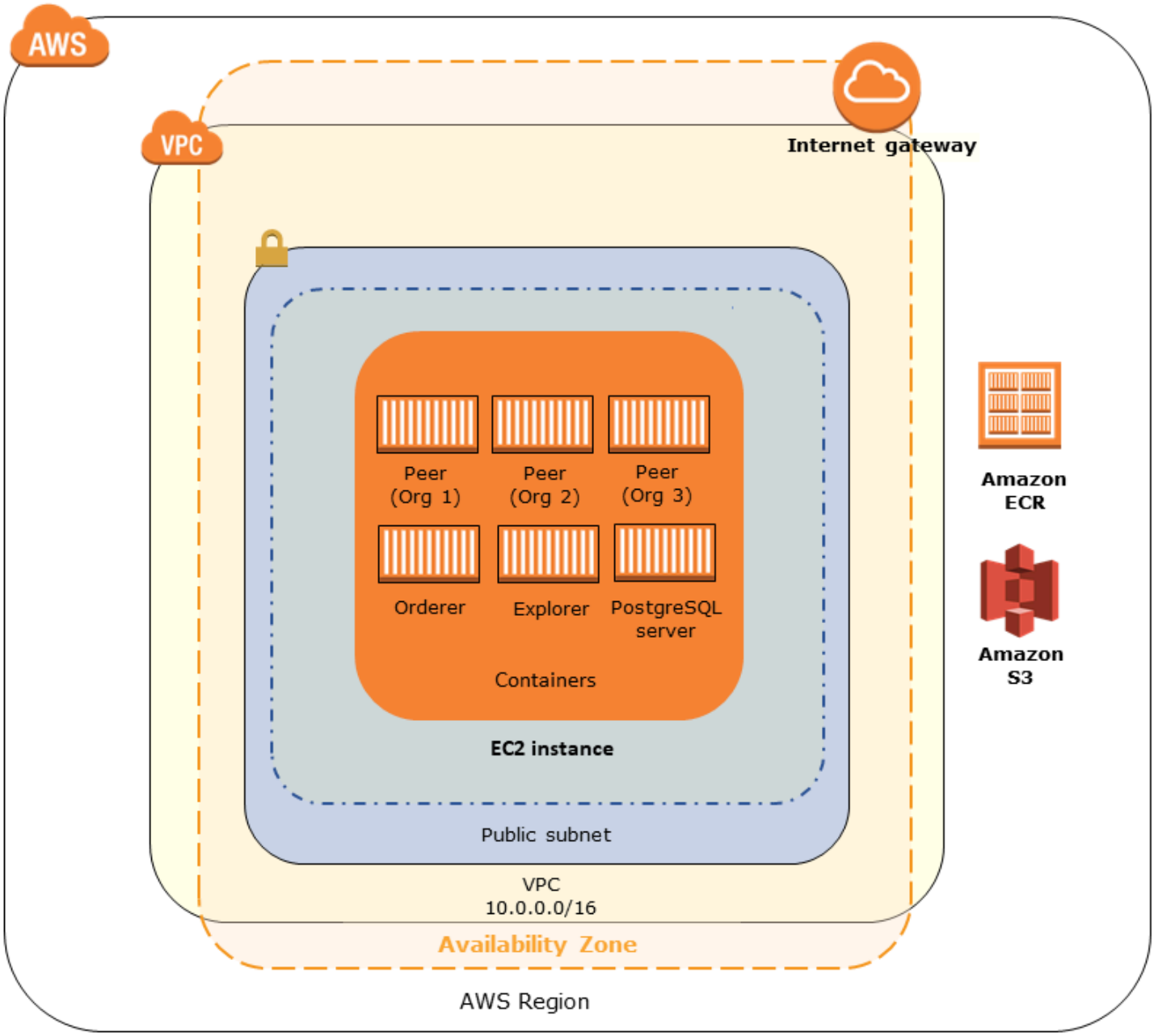
시작 링크

Hyperledger Fabric 템플릿을 사용하여 특정 리전에서 시작할 수 있는 링크는 AWS Blockchain 템플릿 시작하기를 참조하세요. CloudFormation

Hyperledger Fabric용 AWS Blockchain 템플릿 구성 요소

Hyperledger Fabric용 AWS Blockchain 템플릿은 Docker가 있는 EC2 인스턴스를 생성하고, 해당 인스턴스에서 컨테이너를 사용하여 Hyperledger Fabric 네트워크를 시작합니다. 네트워크에는 하나의 명령 서비스 및 3개의 조직이 포함되며, 각각에는 하나의 피어 서비스가 있습니다. 템플릿은 또한 Hyperledger Explorer 컨테이너를 시작하는데, 이를 통해 블록 체인 데이터를 찾을 수 있습니다. PostgreSQL 서버 컨테이너가 시작되어 Hyperledger Explorer를 지원합니다.

다음 다이어그램은 템플릿을 사용하여 생성된 Hyperledger Fabric 네트워크를 보여줍니다.



사전 조건

템플릿을 사용하여 Hyperledger Fabric 네트워크를 시작하기 전에 다음 요구 사항이 충족되는지 확인해야 합니다:

- 사용하는 IAM 보안 주체(사용자 또는 그룹)는 모든 관련 서비스에서 작업할 수 있는 권한을 가지고 있어야 합니다.
- EC2 인스턴스 액세스에 사용(예: SSH 사용)할 수 있는 키 페어에 액세스할 수 있어야 합니다. 키가 인스턴스와 같은 리전에 있어야 합니다.

- Amazon S3 및 Amazon Elastic Container Registry (Amazon ECR)에 액세스하여 컨테이너를 가져올 수 있도록 허용하는 권한 정책이 연결되어 있는 EC2 인스턴스 프로파일이 있어야 합니다. 권한 정책 예시는 EC2 인스턴스 프로파일에 대한 IAM 권한 예제 단원을 참조하십시오.
- Amazon S3 및 Amazon ECR에 액세스할 수 있으려면 퍼블릭 서브넷이 있는 Amazon VPC 네트워크 또는 NAT 게이트웨이 CloudFormation와 탄력적 IP 주소가 있는 프라이빗 서브넷이 있어야 합니다.
- SSH를 사용하여 인스턴스에 연결하는 데 필요한 IP 주소에서 오는 SSH 트래픽(포트 22)를 허용하는 인바운드 규칙과 함께 EC2 보안 그룹이 있어야 하며, Hyperledger Explorer(포트 8080)에 연결하는 데 필요한 클라이언트에 대해서도 동일한 보안 그룹이 있어야 합니다.

EC2 인스턴스 프로파일에 대한 IAM 권한 예제

Hyperledger Fabric용 AWS Blockchain 템플릿을 사용할 때 파라미터 중 하나로 EC2 인스턴스 프로파일 ARN을 지정합니다. EC2 역할 및 인스턴스 프로파일에 연결된 권한 정책에 대한 시작점으로 다음 정책 설명을 사용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

Hyperledger Fabric 리소스에 연결하기

템플릿을 사용하여 생성한 루트 스택에 CREATE_COMPLETE이 표시된 후 EC2 인스턴스에서 Hyperledger Fabric 리소스에 연결할 수 있습니다. 퍼블릭 서브넷을 지정한 경우 다른 EC2 인스턴스처럼 EC2 인스턴스에 연결할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 SSH를 사용하여 Linux 인스턴스에 연결을 참조하세요.

프라이빗 서브넷을 지정한 경우 Bastion Host를 설정하고 사용하여 Hyperledger Fabric 리소스에 대한 연결을 프록시해야 합니다. 자세한 내용은 아래 Bastion Host를 사용한 프록시 연결 섹션을 참조하세요.

> **Note**
>
> 템플릿이 Hyperledger Fabric 서비스를 호스팅하는 EC2 인스턴스에 퍼블릭 IP 주소를 할당하는 것을 알 수 있습니다. 하지만 지정한 프라이빗 서브넷의 라우팅 정책이 이 IP 주소와 퍼블릭 소스 사이의 트래픽을 허용하지 않기 때문에 이 IP 주소는 공개적으로 액세스할 수 없습니다.

Bastion Host를 사용한 프록시 연결

어떤 구성은 공용 Hyperledger Fabric 서비스가 제공되지 않습니다. 이러한 경우 Bastion Host를 통해 Hyperledger Fabric 리소스에 연결할 수 있습니다. Bastion Host에 대한 자세한 내용은 Linux Bastion Host 퀵 스타트 가이드의 Linux Bastion Host 아키텍처를 참조하십시오.

Bastion Host는 EC2 인스턴스입니다. 다음 요구 사항을 충족하는지 확인합니다:

- Bastion Host의 EC2 인스턴스는 퍼블릭 IP 자동 할당이 활성화되어 있고 인터넷 게이트웨이가 있는 퍼블릭 서브넷 내에 있습니다.
- Bastion Host에는 ssh 연결을 허용하는 키 쌍이 있습니다.
- Bastion Host는 연결하는 클라이언트의 인바운드 SSH 트래픽을 허용하는 보안 그룹과 연결됩니다.
- Hyperledger Fabric 호스트에 할당된 보안 그룹(예: ECS가 컨테이너 플랫폼인 경우 또는 Application Load Balancer 또는 docker-local이 컨테이너 플랫폼인 경우 호스트 EC2 인스턴스)은 VPC 내 소스의 모든 포트에 인바운드 트래픽을 허용합니다.

Bastion Host를 설정한 후 연결하는 클라이언트가 Bastion Host를 프록시로 사용하는지 확인하십시오. 다음은 Mac OS를 사용하여 프록시 연결을 설정한 예입니다. *BastionIP*를 Bastion 호스트 EC2 인스턴스의 IP 주소로 대체하고 *MySshKey.pem*을 Bastion Host로 복제한 키 페어 파일로 대체합니다.

명령줄에 다음을 입력합니다:

```
ssh -i mySshKey.pem  ec2-user@BastionIP -D 9001
```

이렇게 하면 로컬 시스템의 포트 9001에 대한 포트 포워딩이 Bastion Host로 설정됩니다.

다음으로 `localhost:9001`에 대한 SOCKS 프록시를 사용하도록 브라우저 또는 시스템을 구성하십시오. 예를 들어, Mac OS의 경우, [System Preferences]와 [Network], [Advanced]를 선택하고 [SOCKS proxy]를 선택하여 localhost:9001을 입력합니다.

FoxyProxy Standard와 Chrome을 사용하여 [More Tools]와 [Extensions]를 선택합니다. [FoxyProxy Standard]에서 [Details]와 [Extension options], [Add New Proxy]를 선택합니다. [Manual Proxy Configuration]을 선택합니다. [Host or IP Address]에 localhost를 입력하고 [Port]에는 9001을 입력합니다. [SOCKS proxy?], [Save]를 선택합니다.

이제 템플릿 출력에 나열된 Hyperledger Fabric 호스트 주소에 연결할 수 있습니다.

문서 기록

다음 표는 이 가이드에 대한 설명서 변경 사항을 설명합니다.

최종 설명서 업데이트: 2019년 5월 1일

변경	설명	Date
AWS Blockchain 템플릿 중단.	AWS Blockchain 템플릿은 2019년 4월 30일에 중단되었습니다. 이 서비스나 이 지원 문서에 대한 추가 업데이트는 없을 것입니다. 에서 최상의 관리형 블록체인 경험을 위해 Amazon Managed Blockchain(AMB)을 사용하는 AWS것이 좋습니다.	2019년 5월 1일
Bastion Host 업데이트입니다.	Bastion Host 추가에 대한 수정된 시작 자습서 및 Ethereum 사전 조건 요구 사항입니다. Bastion Host를 사용하면 ECS 플랫폼을 사용할 때 내부 로드 밸런서를 통해 서비스되고 docker-local을 사용할 때 EC2 인스턴스를 통해 서비스되는 웹 리소스에 액세스할 수 있습니다.	2018년 5월 3일
이 가이드를 생성했습니다.	AWS Blockchain 템플릿의 최초 릴리스를 지원하는 새 개발자 안내서입니다.	2018년 4월 19일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.