

ユーザーガイド

AWS Cloud9



AWS Cloud9: ユーザーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

.....	xxi
とは AWS Cloud9	1
の AWS Cloud9 仕組み	1
AWS Cloud9 環境	1
環境とコンピューティングリソース	2
で何ができますか AWS Cloud9?	2
使用を開始するには	3
その他のトピック	3
機能	3
追加情報	5
関連動画	5
AWS サイトの関連トピック	6
料金	6
その他の質問やヘルプの問い合わせ	7
セットアップ AWS Cloud9	8
個人ユーザーのセットアップ	8
前提条件	9
その他の認証方法	11
次の手順	12
チーム設定	13
前提条件	9
ステップ 1: IAM グループとユーザーを作成し、ユーザーをグループに追加する	17
ステップ 2: AWS Cloud9 グループにアクセス許可を追加する	22
ステップ 3: AWS Cloud9 コンソールにサインインする	27
次の手順	27
エンタープライズセットアップ	28
ステップ 1: 組織の管理アカウントを作成する	31
ステップ 2: 管理アカウントの組織を作成する	32
ステップ 3: 組織にメンバーアカウントを追加する	33
ステップ 4: 組織全体で IAM Identity Center を有効にする	33
ステップ 5: 組織内のグループとユーザーをセットアップする	34
ステップ 6: 組織内のグループとユーザーが を使用できるようにする AWS Cloud9	35
ステップ 7: の使用を開始する AWS Cloud9	37
次の手順	38

追加のセットアップオプション (チームとエンタープライズ)	39
ステップ 1: カスタマー管理ポリシーを作成する	40
ステップ 2: カスタマーマネージドポリシーをグループに追加する	41
次の手順	43
を使用するチームのカスタマー管理ポリシーの例 AWS Cloud9	43
開始方法: ベーシックチュートリアル	50
前提条件	50
ステップ 1: 環境を作成する	50
ステップ 2: ベーシック演習	57
環境の準備をする	58
コードを記述する	58
コードを実行する	59
コードをデバッグする	60
ステップ 3: クリーンアップする	63
関連情報	65
環境を使用する	67
環境を作成する	67
EC2 環境を作成する	68
SSH 環境を作成する	87
Systems Manager を使って no-ingress EC2 インスタンスをアクセスする	92
EC2 環境で Systems Manager を使用する利点	94
Systems Manager 許可の管理	96
Session Manager によって管理されているインスタンスへのアクセスをユーザーに与える	98
CloudFormation を使用して非進入 EC2 環境を作成する	101
依存関係をダウンロードするため、Amazon S3 の VPC エンドポイントを設定する	104
プライベート接続用 VPC エンドポイントの設定	107
環境を開く	108
環境 AWS のサービス からの呼び出し	110
インスタンスプロファイルを作成・使用しマネージド一時認証情報を管理する	112
環境に永続的アクセス認証情報を作成して保存する	118
環境設定の変更	123
環境優先順位を変更する	123
コンソールで環境設定を変更する	124
コードで環境設定を変更する	126
共有環境を使用する	126

共有環境のユースケース	127
環境メンバーのアクセスロールについて	127
環境と同じアカウントのユーザーを招待する	130
環境と同じアカウントの AWS Cloud9 管理者に自分や他のユーザーを招待させる	132
共有環境を開く	134
環境メンバーのリストを参照する	135
環境メンバーのアクティブなファイルを開く	136
環境メンバーが開いているファイルを開く	136
環境メンバーのアクティブなカーソルに移動する	137
共有環境でのチャットの管理	137
環境メンバーのアクセスロールを変更する	139
共有環境から自分のユーザーを削除する	140
別の環境メンバーを削除する	142
環境共有のベストプラクティス	143
環境の移動 : Amazon EBS ボリューム	144
環境の移動	145
AWS Cloud9 EC2 環境を別の Amazon マシンイメージ (AMI) に移動する	147
Amazon EBS ボリュームのサイズ変更	152
Amazon EBS ボリュームの暗号化	155
環境を削除する	158
コンソールで環境を削除する	159
コードで環境を削除する	161
IDE を操作する	163
IDE のツアー	164
前提条件	165
ステップ 1: メニューバー	165
ステップ 2: ダッシュボード	167
ステップ 3: [Environment (環境)] ウィンドウ	168
ステップ 4: エディタ、タブ、ペイン	169
ステップ 5: コンソール	170
ステップ 6: [Open Files (開いているファイル)] セクション	171
ステップ 7: ガーター	172
ステップ 8: ステータスバー	172
ステップ 9: [Outline (アウトライン)] ウィンドウ	174
ステップ 10: [Go (移動)] ウィンドウ	175
ステップ 11: [Immediate (即時)] タブ	177

ステップ 12: プロセスリスト	178
ステップ 13: 設定	179
ステップ 14: ターミナル	180
ステップ 15: [Debugger (デバッガー)] ウィンドウ	181
結論	188
言語サポート	189
IDE AWS Cloud9 でサポートされているプログラミング言語バージョン	191
言語の拡張サポート	191
Java の拡張サポート	192
TypeScript サポートの強化	200
メニューコマンドリファレンス	204
AWS Cloud9 メニュー	205
[File] (ファイル) メニュー	206
[Edit] (編集) メニュー	208
[Find] (検索) メニュー	211
[View] (表示) メニュー	212
[Go] (移動) メニュー	214
[Run] (実行) メニュー	215
[Tools] (ツール) メニュー	216
[Window] (ウィンドウ) メニュー	217
[Support] (サポート) メニュー	221
[Preview] (プレビュー) メニュー	221
他のメニューバーコマンド	222
テキストの検索と置き換え	222
1 つのファイルでテキストを検索する	222
1 つのファイルでテキストを置き換える	223
複数のファイルでテキストを検索する	223
複数のファイルでテキストを置き換える	225
検索と置換のオプション	226
ファイルのプレビュー	227
プレビューするファイルを開く	227
ファイルプレビューの再ロード	229
ファイルプレビュータイプの変更	229
ウェブブラウザの別のタブでファイルプレビューを開く	229
別のファイルプレビューに切り替える	229
実行中のアプリケーションのプレビュー	229

アプリケーションの実行	230
実行中のアプリケーションのプレビュー	232
アプリケーションのプレビューの再ロード	234
アプリケーションプレビュータイプの変更	234
アプリケーションプレビューを開く	234
別のプレビュー URL に切り替える	235
実行中のアプリケーションの共有	235
ファイルリビジョンの操作	241
イメージファイルの操作	243
イメージの表示または編集	243
イメージのサイズ変更	244
イメージのクロップ	244
イメージの回転	245
イメージの反転	245
イメージのズーム	245
イメージのスムーズ化	245
ビルダー、ランナー、デバッガーの操作	246
組み込みの構築、実行、およびデバッグのサポート	246
プロジェクトのファイルを構築する	247
コードを実行する	247
コードをデバッグする	248
組み込みランナーを変更する	249
実行設定を作成する	249
ビルダーまたはランナーを作成する	250
ビルダーまたはランナーを定義する	251
カスタム環境変数の操作	255
コマンドレベルのカスタム環境変数を設定する	255
~/.bash_profile のカスタムユーザー環境変数を設定する	256
ローカルのカスタム環境変数を設定する	256
~/.bashrc のカスタムユーザー環境変数を設定する	256
ENV リストのカスタム環境変数を設定する	257
プロジェクト設定の操作	257
プロジェクト設定の表示または変更	258
環境の現在のプロジェクト設定を別の環境に適用	258
プロジェクト設定のカスタマイズ	259
環境のEC2 インスタンスを手動で停止する	266

ユーザー設定の操作	267
ユーザー設定の表示または変更	268
ユーザー設定の他のユーザーとの共有	268
ユーザー設定のカスタマイズ	269
AWS プロジェクトとユーザー設定の操作	278
プロジェクトレベル設定	279
ユーザーレベル設定	279
キー割り当てを使用する	279
キー割り当ての表示または変更	280
別のユーザーとキー割り当てを共有する	280
キーボードモードを変更する	281
オペレーティングシステムのキー割り当てを変更する	281
特定のキー割り当てを変更する	282
すべてのカスタムキー割り当てを削除する	283
テーマの操作	283
テーマの表示または変更	284
変更可能なテーマの全体的な設定	284
Theme overrides (テーマの上書き)	284
初期化スクリプトの管理	285
初期化スクリプトを再実行する	285
MacOS デフォルトキー割り当てリファレンス	286
General	286
タブ	290
パネル	292
コードエディタ	293
emmet	301
Terminal	302
実行およびデバッグ	302
MacOS Vim キー割り当てリファレンス	303
General	303
タブ	307
パネル	309
コードエディタ	310
emmet	318
Terminal	319
実行およびデバッグ	319

MacOS Emacs キー割り当てリファレンス	320
General	320
タブ	324
パネル	326
コードエディタ	327
emmet	335
Terminal	336
実行およびデバッグ	336
MacOS Sublime キー割り当てリファレンス	337
General	337
タブ	342
パネル	344
コードエディタ	345
emmet	353
Terminal	354
実行およびデバッグ	354
Windows/Linux デフォルトキー割り当てリファレンス	355
General	356
タブ	359
パネル	361
コードエディタ	362
emmet	370
Terminal	371
実行およびデバッグ	371
Windows/Linux Vim キー割り当てリファレンス	372
General	373
タブ	376
パネル	378
コードエディタ	379
emmet	387
Terminal	387
実行およびデバッグ	388
Windows/Linux Emacs キー割り当てリファレンス	388
General	389
タブ	393
パネル	395

コードエディタ	396
emmet	403
Terminal	404
実行およびデバッグ	404
Windows/Linux Sublime キー割り当てリファレンス	405
General	406
タブ	410
パネル	412
コードエディタ	413
emmet	421
Terminal	421
実行およびデバッグ	422
コマンドリファレンス	422
他の AWS サービスの使用	424
Amazon Lightsail インスタンスの使用	424
ステップ 1: Linux ベースの Lightsail インスタンスを作成する	425
ステップ 2: インスタンスを使用するようにインスタンスを設定する AWS Cloud9	428
ステップ 3: SSH AWS Cloud9 開発環境を作成して接続する	430
ステップ 4: AWS Cloud9 IDE を使用してインスタンスのコードを変更する	433
の使用 AWS CodePipeline	434
ステップ 1: ソースコードリポジトリを作成または識別する	435
ステップ 2: AWS Cloud9 開発環境を作成し、コードリポジトリに接続してコードをアップ ロードする	436
ステップ 3: を使用する準備をする AWS CodePipeline	437
ステップ 4: でパイプラインを作成する AWS CodePipeline	437
CodeCatalyst の使用	438
CodeCatalyst AWS Cloud9 で を使用し、CodeCatalyst で AWS Cloud9 環境をレプリケー トする方法について説明します。	438
から AWS Cloud9 Amazon CodeCatalyst への移行	439
レプリケーションツールを使用する	453
レプリケーションプロセスに関するよくある質問	457
CodeCatalyst の開発環境	460
の使用 AWS CDK	465
AWS CDK アプリケーション	465
Git パネルを使用したビジュアルソース管理	468
Git パネルを使用したソース管理の管理	469

Git リポジトリを初期化またはクローン化するには	472
ファイルのステージングとコミット	474
異なるファイルバージョンの表示	477
ブランチの操作	477
リモートリポジトリを操作する	481
ファイルの stash と取得	483
リファレンス: Git パネルで利用可能な Git コマンド	484
Git パネルメニューから利用可能な Git コマンドのリファレンス	486
Git パネルの検索フィールドから利用可能な Git コマンド	488
Toolkit の使用 AWS	491
Toolkit を使用する理由 AWS	491
AWS ツールキットの有効化	493
AWS Toolkit のアクセス認証情報の管理	494
IAMロールを使用して、EC2 インスタンスのアプリケーションに許可を付与する	495
Toolkit AWS コンポーネントの特定	496
Toolkit の無効化 AWS	496
AWS ツールキットのトピック	497
ナビゲーションおよび設定	497
AWS Explorer を使用して複数のリージョンのサービスとリソースを操作する	498
Toolkit AWS メニューへのアクセスと使用	499
AWS 設定ペインを使用した AWS Toolkit 設定の変更	501
API Gateway	504
REST API の呼び出し	505
AWS App Runner	506
前提条件	507
料金	510
App Runner サービスの作成	510
App Runner サービスの管理	513
CloudFormation スタック	516
CloudFormation スタックの削除	516
Amazon CloudWatch Logs	516
CloudWatch ロググループとログストリームの表示	517
CloudWatch ログイベントの操作	518
AWS Lambda 関数	519
リモートの Lambda 関数を呼び出す	520
Lambda 関数のダウンロード、アップロード、削除	521

リソース	527
リソースにアクセスするための IAM アクセス許可	527
既存のリソースの操作	528
Amazon S3	529
Amazon S3 バケットの操作	529
Amazon S3 オブジェクトの操作	531
AWS サーバーレスアプリケーション	534
サーバーレスアプリケーションの作成	534
サーバーレスアプリケーションの実行とデバッグ	536
サーバーレスアプリケーションの同期	544
サーバーレスアプリケーションの削除	545
AWS Toolkit コードレンズの有効化	546
サーバーレスアプリケーションのデバッグ用設定オプション	546
AWS Step Functions	550
前提条件	551
ステートマシンの作成と発行	551
AWS Toolkit でステートマシンを実行する	553
ステートマシン定義ファイルをダウンロードし、そのワークフローを視覚化する	554
AWS Systems Manager	555
仮定と前提条件	556
Systems Manager Automation ドキュメントの IAM アクセス許可	556
Systems Manager オートメーションドキュメントを新規作成する	557
Systems Manager オートメーションドキュメントの発行	557
既存の Systems Manager オートメーションドキュメントの編集	558
バージョンの使用	559
Systems Manager オートメーションドキュメントの削除	560
Systems Manager オートメーションドキュメントの実行	560
トラブルシューティング	561
Amazon ECR	562
前提条件	562
IDE での Amazon ECR AWS Cloud9 の使用	563
AWS IoT	572
AWS IoT 前提条件	572
AWS IoT モノ	572
AWS IoT 証明書	574
AWS IoT ポリシー	577

Amazon ECS	580
Amazon ECS Exec	581
Amazon EventBridge	583
Amazon EventBridge スキーマの使用	583
のチュートリアル AWS Cloud9	586
AWS CLI および aws-shell チュートリアル	586
前提条件	587
ステップ 1: 環境に AWS CLI、aws-shell、またはその両方をインストールする	588
ステップ 2: 環境で認証情報管理を設定する	589
ステップ 3: 環境内の AWS CLI または aws-shell を使用して基本的なコマンドを実行する ..	590
ステップ 4: クリーンアップする	591
AWS CodeCommit チュートリアル	591
前提条件	592
ステップ 1: 必要なアクセス許可を持つ IAM グループを設定する	592
ステップ 2: CodeCommit でリポジトリを作成する	594
ステップ 3: 環境をリモートリポジトリに接続する	595
ステップ 4: リモートリポジトリのクローンを環境に作成する	597
ステップ 5: リポジトリにファイルを追加する	597
ステップ 6: クリーンアップする	600
Amazon DynamoDB チュートリアル	600
前提条件	601
ステップ 1: 環境に AWS CLI、AWS CloudShell、またはその両方をインストールして設定 する	601
ステップ 2: テーブルを作成する	602
ステップ 3: テーブルに 1 つの項目を追加する	604
ステップ 4: テーブルに複数の項目を追加する	605
ステップ 5: グローバルセカンダリインデックスを作成する	609
ステップ 6: テーブルから項目を取得する	612
ステップ 7: クリーンアップする	616
AWS CDK チュートリアル	617
前提条件	618
ステップ 1: 必要なツールをインストールする	618
ステップ 2: コードを追加する	622
ステップ 3: コードを実行する	624
ステップ 4: クリーンアップする	627
LAMP チュートリアル	627

前提条件	628
ステップ 1: ツールのインストール	628
ステップ 2: MySQL を設定する	631
ステップ 3: ウェブサイトの設定	632
ステップ 4: クリーンアップする	636
WordPress チュートリアル	638
前提条件	638
インストールの概要	639
ステップ 1: MariaDB サーバーのインストールと設定	639
ステップ 2: WordPress のインストールと設定	640
ステップ 3: Apache HTTP サーバーの設定	642
ステップ 4: WordPress ウェブコンテンツのプレビュー	642
混在コンテンツのエラーを管理する	643
Java チュートリアル	644
前提条件	644
ステップ 1: 必要なツールをインストールする	645
ステップ 2: コードを追加する	647
ステップ 3: コードを構築して実行する	647
ステップ 4: を使用するように を設定する AWS SDK for Java	648
ステップ 5: 環境で AWS 認証情報管理を設定する	655
ステップ 6: AWS SDK コードを追加する	655
ステップ 7: AWS SDK コードを構築して実行する	657
ステップ 8: クリーンアップする	658
C++ チュートリアル	658
前提条件	659
ステップ1: g ++と必要な dev パッケージをインストールする	659
ステップ2: CMakeをインストールする	660
ステップ 3: SDK for C++ の取得と構築	660
ステップ4: C ++および CMakeLists ファイルを作成する	662
ステップ 5: C++ コードを構築および実行する	666
ステップ 6: クリーンアップする	667
Python チュートリアル	667
前提条件	668
ステップ 1: Python をインストールする	668
ステップ 2: コードを追加する	669
ステップ 3: コードを実行する	670

ステップ 4: をインストールして設定する AWS SDK for Python (Boto3)	670
ステップ 5: AWS SDK コードを追加する	671
ステップ 6: AWS SDK コードを実行する	673
ステップ 7: クリーンアップする	674
.NET チュートリアル	674
前提条件	675
ステップ 1: 必要なツールをインストールする	675
ステップ 2: (オプション): Lambda 関数用の .NET CLI 拡張をインストールする	677
ステップ 3: .NET コンソールアプリケーションプロジェクトを作成する	678
ステップ 4: コードを追加する	679
ステップ 5: コードを構築および実行する	679
ステップ 6: を使用する .NET コンソールアプリケーションプロジェクトを作成してセットアップする AWS SDK for .NET	681
ステップ 7: AWS SDK コードを追加する	683
ステップ 8: AWS SDK コードを構築して実行する	685
ステップ 9: クリーンアップする。	686
Node.js チュートリアル	686
前提条件	686
ステップ 1: 必要なツールをインストールする	687
ステップ 2: コードを追加する	688
ステップ 3: コードを実行する	689
ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する	690
ステップ 5: AWS SDK コードを追加する	692
ステップ 6: AWS SDK コードを実行する	695
ステップ 7: クリーンアップする	696
PHP チュートリアル	696
前提条件	697
ステップ 1: 必要なツールをインストールする	697
ステップ 2: コードを追加する	699
ステップ 3: コードを実行する	699
ステップ 4: をインストールして設定する AWS SDK for PHP	700
ステップ 5: AWS SDK コードを追加する	702
ステップ 6: AWS SDK コードを実行する	704
ステップ 7: クリーンアップする	705
Ruby	705
Go チュートリアル	705

前提条件	706
ステップ 1: 必要なツールをインストールする	706
ステップ 2: コードを追加する	707
ステップ 3: コードを実行する	708
ステップ 4: をインストールして設定する AWS SDK for Go	710
ステップ 5: AWS SDK コードを追加する	711
ステップ 6: AWS SDK コードを実行する	713
ステップ 7: クリーンアップする	714
TypeScript チュートリアル	714
前提条件	715
ステップ 1: 必要なツールをインストールする	715
ステップ 2: コードを追加する	717
ステップ 3: コードを実行する	718
ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する	719
ステップ 5: AWS SDK コードを追加する	720
ステップ 6: AWS SDK コードを実行する	722
ステップ 7: クリーンアップする	723
Docker チュートリアル	724
前提条件	724
ステップ 1: Docker のインストールと実行	725
ステップ 2: イメージの構築	726
ステップ 3: コンテナの実行	729
ステップ 4: 環境の作成	731
ステップ 5: コードの実行	737
ステップ 6: クリーンアップする	737
関連チュートリアル	739
の高度なトピック AWS Cloud9	740
EC2 環境と SSH 環境の比較	740
Amazon VPC の設定	742
の Amazon VPC 要件 AWS Cloud9	742
VPC と他の VPC リソースを作成する	758
VPC のみを作成する	759
のサブネットを作成する AWS Cloud9	761
サブネットをパブリックまたはプライベートとして設定する	762
SSH 環境ホスト要件	764
SSH 環境を作成するタイミングと方法	765

SSH ホスト要件	767
AWS Cloud9 インストーラ	769
AWS Cloud9 インストーラをダウンロードして実行する	770
AWS Cloud9 インストーラのトラブルシューティング	770
インバウンド SSH IP アドレスの範囲	772
ip-ranges.json がない IP アドレス	774
AMI のコンテンツ	775
Amazon Linux 2023/Amazon Linux 2	775
Ubuntu Server	777
サービスリンクロール	778
のサービスにリンクされたロールのアクセス許可 AWS Cloud9	779
のサービスにリンクされたロールの作成 AWS Cloud9	782
のサービスにリンクされたロールの編集 AWS Cloud9	783
のサービスにリンクされたロールの削除 AWS Cloud9	783
AWS Cloud9 サービスにリンクされたロールでサポートされているリージョン	783
CloudTrail による API コールのログ記録	783
AWS Cloud9 CloudTrail の情報	784
AWS Cloud9 ログファイルエントリについて	785
タグ	801
基礎となるリソースへのタグ更新の伝播	802
のセキュリティ AWS Cloud9	805
データ保護	806
データ暗号化	807
Identity and Access Management	809
オーディエンス	810
アイデンティティを使用した認証	810
ポリシーを使用したアクセスの管理	811
が IAM と AWS Cloud9 連携する方法	813
アイデンティティベースのポリシーの例	819
トラブルシューティング	822
AWS Cloud9 が IAM リソースとオペレーションと連携する方法	824
AWS マネージドポリシー	828
のカスタマー管理ポリシーの作成 AWS Cloud9	833
AWS Cloud9 アクセス許可リファレンス	849
AWS マネージド一時認証情報	856
ログ記録とモニタリング	862

CloudTrail によるアクティビティのモニタリング	862
EC2 環境のパフォーマンスのモニタリング	862
コンプライアンス検証	863
耐障害性	867
インフラストラクチャセキュリティ	867
ソフトウェアの更新プログラムと修正プログラム	868
セキュリティのベストプラクティス	869
トラブルシューティング AWS Cloud9	870
Installer (インストーラ)	870
AWS Cloud9 インストーラがハングまたは失敗する	870
AWS Cloud9 「Package Cloud9 IDE 1」と表示してもインストーラが終了しない	870
依存関係をインストールできませんでした	871
SSH 環境エラー: 「pty.js をインストールするには Python バージョン 3 が必要です」	872
AWS Cloud9 環境	872
環境の作成エラー: 「EC2 インスタンスを作成することができません ...」	872
環境作成エラー: 「sts:AssumeRole を実行する権限がありません」	873
フェデレーティッドアイデンティティで環境を作成できない	873
コンソールエラー: 「ユーザーにはリソースでアクションを実行する権限がありません」 ..	874
環境に接続できない	875
環境を開くことができない	875
AWS Cloud9 環境を開くことができません: 「この環境には現在共同作業者がアクセスでき ません。マネージド一時認証情報の削除が完了するまでお待ちください。または、この環境 の所有者にお問い合わせしてください。」	877
環境の削除エラー: 「1 つ以上の環境を削除できませんでした」	878
IDE AWS Cloud9 での環境のタイムアウト時間の変更	879
AWS Cloud9 環境に十分なディスク容量がないため、AWS Toolkit で SAM アプリケーショ ンをローカルで実行中にエラーが発生しました	879
古いバージョンの Microsoft Edge ブラウザを使用して IDE をロードすることはできませ ん	880
IDE File Explorer でサブフォルダ構造 /home/ec2-user/environment/home/ec2-user/ environment AWS Cloud9 を作成することはできません。	880
CodeCatalyst の IDE の File Explorer 内にサブフォルダ構造 /projects/projects AWS Cloud9 を作成することはできません。	881
tmux セッションエラー AWS Cloud9 のため、でターミナルウィンドウとやり取りできな い	881
Amazon EC2	883

Amazon EC2 インスタンスが自動的に更新されない	883
AWS CLI または AWS シェルエラー: EC2 環境の「リクエストに含まれるセキュリティトークンが無効です」	883
VPC の IP アドレスを Docker が使用しているため、EC2 環境に接続できません	884
IDE File Explorer でサブフォルダ構造 /home/ec2-user/environment/home/ec2-user/environment AWS Cloud9 を作成することはできません。	880
AWS License Manager ライセンス設定が Amazon EC2 インスタンスに関連付けられている場合、コンソール AWS Cloud9 から起動できない	885
EC2 環境で一部のコマンドやスクリプトを実行できない	886
を使用して EC2 環境を作成するときに「インスタンスプロファイル AWSCloud9SSMInstanceProfile がアカウントに存在しません」と報告するエラーメッセージ CloudFormation	886
を使用して EC2 環境を作成するときに「リソース perform: ssm:StartSession で」と報告されるエラーメッセージ CloudFormation	887
を使用して EC2 環境を作成するときに「実行する: iam:GetInstanceProfile on resource: instance profile AWSCloud9SSMInstanceProfile」という承認がないと報告するエラーメッセージ AWS CLI	887
デフォルトの暗号化が Amazon EBS ボリュームに適用されている場合に、環境の作成障害がでる	888
EC2-Classic アカウントの VPC エラー:「環境にアクセスできません」	888
その他の AWS サービス	889
CodeCatalyst の IDE の File Explorer 内にサブフォルダ構造 /projects/projects AWS Cloud9 を作成することはできません。	881
IDE の外部で実行中のアプリケーションを表示できない	890
AWS Toolkit の実行中にエラーが発生しました:「環境が inodes を使い果たしています。「fs.inotify.max_user_watches」の制限を増やしてください。」	892
Lambda ローカル関数実行エラー: SAM Local をインストールできない	893
AWS Control Tower AWS Cloud9を使用して Amazon EC2 環境を作成しよう とすると、「環境の作成がエラーで失敗しました: 次のフック (複数可) が失敗しました: [ControlTower::Guard::Hook]」	893
デフォルトの暗号化が Amazon EBS ボリュームに適用されている場合に、環境の作成障害がでる	888
AWS License Manager ライセンス設定が Amazon EC2 インスタンスに関連付けられている場合、コンソール AWS Cloud9 から起動できない	885
アプリケーションプレビュー	895

環境を再ロードした後、アプリケーションプレビューを最新の情報に更新する必要がある	895
アプリケーションプレビューまたはファイルプレビュー通知: 「サードパーティーの Cookie が無効になっています」	895
アプリケーションプレビュータブにエラーが表示される、または空白になる	900
サイトへの接続が安全でないため、IDE でウェブコンテンツをプレビューできません	901
ファイルをプレビューすると 499 エラーが返される	902
パフォーマンス	902
AWS Cloud9 IDE の長時間のフリーズ	902
コンソールの警告: 「最小限のコード補完エンジンに切り替えます...」	903
IDE による警告: 「この環境のメモリが不足しています」または「この環境の CPU ロードが高くなっています」	903
IDE AWS Cloud9 にファイルをアップロードできない	904
IDE AWS Cloud9 のダウンロード速度が遅い	905
サイトへの接続が安全でないため、IDE でウェブコンテンツをプレビューできません	901
サードパーティーのアプリケーションとサービス	905
tmux セッションエラー AWS Cloud9 のため、でターミナルウィンドウとやり取りできない	881
古いバージョンの Microsoft Edge ブラウザを使用して IDE をロードすることはできません	880
C++ プロジェクトのデバッグ時に、gdb を伴うエラー	908
での PHP ランナーの問題 AWS Cloud9	909
Node.js に関連する GLIBC エラー	909
サポートされるブラウザ	910
Service Quotas	912
AWS Cloud9 クォータ	912
AWS Cloud9 IDE ダウンロードクォータ	913
関連する AWS サービスクォータ	913
ドキュメント履歴	915

AWS Cloud9 は新規顧客には利用できなくなりました。AWS Cloud9 の既存のお客様は、通常どおりサービスを引き続き使用できます。[詳細はこちら](#)

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。

とは AWS Cloud9

AWS Cloud9 は統合開発環境、つまり IDE です。

AWS Cloud9 IDE は、複数のプログラミング言語とランタイムデバッガー、および組み込みターミナルをサポートする豊富なコード編集エクスペリエンスを提供します。また、ソフトウェアのコード作成、ビルド、実行、テスト、デバッグに使用するツールのコレクションが含まれ、クラウドへのソフトウェアのリリースに役立ちます。

ウェブブラウザから AWS Cloud9 IDE にアクセスします。IDE を好みに合わせて設定できます。カラーテーマを切り替えたり、ショートカットキーを割り当てたり、プログラミング言語固有の構文の色やコードのフォーマットを使用したりできます。

(ゴット! 試す準備ができました AWS Cloud9。 [使用を開始するには](#))

の AWS Cloud9 仕組み

次の図は、AWS Cloud9 の仕組みの概要を示しています。

図表 (下から開始) から、AWS Cloud9 IDE を使用して、ローカルコンピュータのウェブブラウザで実行し、AWS Cloud9 環境を操作します。コンピューティングリソース (Amazon EC2 インスタンスまたは独自のサーバーなど) はその環境に接続します。最後に、作業は AWS CodeCommit リポジトリまたは別のタイプのリモートリポジトリに保存されます。



AWS Cloud9 環境

AWS Cloud9 環境は、プロジェクトのファイルを保存する場所であり、アプリを開発するツールを実行する場所です。

AWS Cloud9 IDE を使用すると、次のことができます。

- プロジェクトのファイルをインスタンスまたはサーバーにローカルに保存する。
- のリポジトリなどのリモートコードリポジトリを環境にクローン AWS CodeCommitします。
- ローカルファイルと環境にクローンされたファイルを組み合わせて操作します。

それぞれの環境が特定の開発環境用にセットアップされた複数の環境を作成して切り替えることができます。環境をクラウドに保存することで、プロジェクトが単一のコンピュータまたはサーバーのセットアップに縛られる必要がなくなります。これにより、コンピュータを簡単に切り替えたり、開発者を迅速にチームに加えたりといった作業を行うことができます。

環境とコンピューティングリソース

バックグラウンドで、環境をコンピューティングリソースに接続する方法がいくつかあります。

- に Amazon EC2 インスタンスを作成し、新しく作成した EC2 インスタンスに環境を接続する AWS Cloud9 ように指示できます。このタイプのセットアップを EC2 環境と呼びます。
- 環境を既存のクラウドコンピューティングインスタンスまたは独自のサーバーに接続する AWS Cloud9 ように に指示できます。このタイプのセットアップを SSH 環境と呼びます。

EC2 環境と SSH 環境には、類似点と相違点があります。を初めて使用する場合は AWS Cloud9、設定の多くを が AWS Cloud9 処理するため、EC2 環境を使用することをお勧めします。の詳細を確認し AWS Cloud9、これらの類似点と相違点をよりよく理解するには、「」を参照してくださいの [SSH 環境と比較した EC2 環境 AWS Cloud9](#)。

の AWS Cloud9 仕組みの詳細については、これらの関連[動画](#)と[ウェブページ](#)を参照してください。

で何ができますか AWS Cloud9?

を使用すると AWS Cloud9、多くのエキサイティングなシナリオやバリエーションでソフトウェアをコーディング、構築、実行、テスト、デバッグ、リリースできます。以下が含まれます (ただし、これらに限定されません)。

- 複数のプログラミング言語と AWS Cloud Development Kit (AWS CDK) でコードを使用する。
- 実行中の Docker コンテナでコードを使用する。
- オンラインコードリポジトリを使用する。
- リアルタイムで他のユーザーとコラボレーションする。
- さまざまなデータベースやウェブサイト技術とやり取りする。
- ターゲット設定 AWS Lambda、Amazon API Gateway、AWS サーバーレスアプリケーション。
- Amazon Lightsail、AWS CodeStar、などの他の AWS 製品を活用します AWS CodePipeline。

より詳細なリストについては、「[で何ができますか AWS Cloud9 ?](#)」を参照してください。

使用を開始するには

の使用を開始するには AWS Cloud9、「」の手順に従ってから[セットアップ AWS Cloud9](#)、[基本的なチュートリアル](#)を実行します。

その他のトピック

- [で何ができますか AWS Cloud9 ?](#)
- [に関する追加情報 AWS Cloud9](#)

で何ができますか AWS Cloud9 ?

一般的なシナリオ AWS Cloud9 での の使用については、以下のリソースを参照してください。

主なシナリオ

シナリオ	リソース
AWS Toolkit を使用して、AWS Lambda 関数とサーバーレスアプリケーションでコードを作成、実行、デバッグします。	Toolkit を使用した AWS Lambda 関数の操作 AWS
一般的なアプリケーションおよびフレームワーク (たとえば WordPress、LAMP (Linux、Apache、MySQL、PHP)、Node.js、Nginx、Drupal、Joomla、Linux ディストリビューション (Amazon Linux、Ubuntu、Debian、FreeBSD、openSUSE など)) で事前設定された Amazon Lightsail インスタンスを操作する	IDE AWS Cloud9 での Amazon Lightsail インスタンスの操作
AWS CodePipelineでの継続的デリバリーソリューションのコードを操作します。	IDE AWS CodePipeline での AWS Cloud9 の使用
AWS CLI とを使用して AWS サービスを自動化します AWS CloudShell。	AWS CLI および の aws-shell チュートリアル AWS Cloud9

シナリオ	リソース
でソースコードリポジトリを使用します AWS CodeCommit。	AWS CodeCommit の チュートリアル AWS Cloud9
Git パネルインターフェイスを使用して GitHub でソースコードのリポジトリを操作します。	Git パネルを使用したビジュアルソース管理
Amazon DynamoDB で NoSQL データベースを操作します。	の Amazon DynamoDB チュートリアル AWS Cloud9
LAMP (Linux、Apache HTTP Server、My SQL、PHP) スタックを操作します。	の LAMP チュートリアル AWS Cloud9
WordPress ウェブサイトを使用します。	の WordPress チュートリアル AWS Cloud9
Java および AWS SDK for Javaのコードを操作します。	の Java チュートリアル AWS Cloud9
C++ および AWS SDK for C++のコードを操作します。	の C++ チュートリアル AWS Cloud9
Python および AWS SDK for Python (Boto)のコードを操作します。	の Python チュートリアル AWS Cloud9
.NET Core のコードと AWS SDK for .NETを操作します。	の .NET チュートリアル AWS Cloud9
Node.js のコードと AWS SDK for JavaScriptを操作します。	の Node.js チュートリアル AWS Cloud9
PHP および AWS SDK for PHPのコードを操作します。	の PHP チュートリアル AWS Cloud9
Ruby および AWS SDK for Rubyのコードを操作します。	AWS の SDK for Ruby AWS Cloud9
Go および AWS SDK for Goのコードを操作します。	の Go チュートリアル AWS Cloud9

シナリオ	リソース
TypeScript のコードと AWS SDK for JavaScript を操作します。	の TypeScript チュートリアル AWS Cloud9
AWS Cloud Development Kit (AWS CDK) のコードを操作します。	AWS CDK の チュートリアル AWS Cloud9
Docker コンテナを実行しているコードを使用します。	の Docker チュートリアル AWS Cloud9
リアルタイムのチャットサポートにより、他のユーザーを招待して環境を共有します。	での共有環境の使用 AWS Cloud9
AWS RoboMaker でインテリジェントロボットアプリケーションのコードを使用します。	AWS RoboMaker デベロッパーガイド内にある AWS Cloud9 を使って開発

に関する追加情報 AWS Cloud9

このトピックでは、 の詳細について説明します AWS Cloud9。

トピック

- [関連動画](#)
- [AWS サイトの関連トピック](#)
- [料金](#)
- [その他の質問やヘルプの問い合わせ](#)

関連動画

- [AWS re:Invent 2017 - 紹介 AWS Cloud9: Werner Vogels Keynote](#) (9 分、YouTube ウェブサイト)
- [AWS re:Invent Launchpad 2017 - AWS Cloud9](#), (15 分、YouTube ウェブサイト)
- [紹介 AWS Cloud9 - AWS オンライン Tech Talks](#) (33 分、YouTube ウェブサイト)
- [AWS シドニーサミット 2018: AWS Cloud9 および AWS CodeStar](#) (25 分、YouTube ウェブサイト)

AWS サイトの関連トピック

- [の紹介 AWS Cloud9](#)
- [AWS Cloud9 – クラウド開発者環境](#)
- [AWS Cloud9 概要](#)
- [AWS Cloud9 特徴](#)
- [AWS Cloud9 よくある質問](#)

料金

追加料金はかかりません AWS Cloud9。AWS Cloud9 開発環境に Amazon EC2 インスタンスを使用する場合、コードの実行と保存に使用されるコンピューティングリソースとストレージリソース (Amazon EC2 インスタンス、Amazon EBS ボリュームなど) に対してのみ料金が発生します。また、環境をSSH 経由で既存の Linux サーバー (オンプレミスサーバーなど) に接続できます。追加料金はかかりません。

お支払いはお客様が実際に使用した分だけです。最低料金や初期費用は不要です。環境内で作成または使用する AWS リソース (AWS Lambda 関数など) には、通常の AWS 料金が課金されます。

AWS 無料利用枠の対象となる新規の AWS お客様は、AWS Cloud9 を無料で使用できます。環境が AWS 無料利用枠を超えてリソースを使用する場合、それらのリソースの通常の AWS 料金が課金されます。

詳細については、以下を参照してください。

- AWS Cloud9 料金: [AWS Cloud9 「料金表」](#) を参照してください。
- AWS サービス料金: [Amazon EC2 料金表](#)、[「Amazon EBS 料金表」](#)、[AWS Lambda 「料金表」](#)、[AWS 「料金表」](#) を参照してください。
- AWS 無料利用枠: AWS Billing and Cost Management ユーザーガイドの [AWS 「無料利用枠の使用」](#) と [「無料利用枠の使用の追跡」](#) を参照してください。
- 教育機関向け料金: [「AWS Educate プログラム」](#) を参照してください。

その他の質問やヘルプの問い合わせ

AWS Cloud9 コミュニティに質問したり、サポートを求めたりするには、[AWS Cloud9 ディスカッションフォーラム](#)を参照してください。(このフォーラムに参加すると、サインインが必要になるAWS 場合があります)。

また、[よくある質問](#) (FAQ) を参照したり、[直接お問い合わせ](#)いただくこともできます。

セットアップ AWS Cloud9

の使用を開始するには AWS Cloud9、使用する計画に応じて、これらの手順のいずれかに従います AWS Cloud9。

使用パターン	以下の手順に従います
AWS アカウントを使用するのは私だけであり、学生ではありません。	個々のユーザーセットアップ
私は、1 つの AWS アカウント内に複数のユーザーを持つチームに属しています。	チームセットアップ
私は、1 つの組織内に 1 つ以上の AWS アカウントを持つ エンタープライズに属しています。	エンタープライズセットアップ

の一般的な情報については AWS Cloud9、[「とは AWS Cloud9」](#) を参照してください。

トピック

- [の個々のユーザー設定 AWS Cloud9](#)
- [のチームのセットアップ AWS Cloud9](#)
- [のエンタープライズセットアップ AWS Cloud9](#)
- [の追加のセットアップオプション AWS Cloud9](#)

の個々のユーザー設定 AWS Cloud9

このトピックでは、学生でないときに で AWS アカウント 唯一のユーザー AWS Cloud9 として を セットアップして使用する 方法について説明します。他の使用パターン AWS Cloud9 用に を設定できます。詳細については、「[セットアップ AWS Cloud9](#)」を参照してください。

で唯一のユーザー AWS Cloud9 として を使用するには AWS アカウント、まだ がない場合は にサインアップ AWS アカウント します。次に、AWS Cloud9 コンソールにサインインします。

トピック

- [前提条件](#)

- [その他の認証方法](#)
- [次の手順](#)

前提条件

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

にサインアップすると AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、[ルートユーザーアクセスが必要なタスク](#)の実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。<https://aws.amazon.com/> の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM アイデンティティセンター、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、AWS アカウント E メールアドレスを入力して、アカウント所有者[AWS マネジメントコンソール](#)として にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの[ルートユーザーとしてサインインする](#)を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAM [ユーザーガイドの AWS アカウント 「ルートユーザー \(コンソール\) の仮想 MFA デバイス](#)を有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[AWS IAM アイデンティティセンターの有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

を ID ソース IAM アイデンティティセンターディレクトリとして使用する方法のチュートリアルについては、「AWS IAM アイデンティティセンター ユーザーガイド」の「[デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ](#)」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「[ユーザーガイド](#)」の AWS「[アクセスポータルにサインイン](#)する」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[権限設定を作成する](#)」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[Add groups](#)」を参照してください。

その他の認証方法

Warning

セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM アイデンティティセンター](#)などの ID プロバイダーとのフェデレーションを使用してください。

間のアクセスを管理する AWS アカウント

セキュリティのベストプラクティスとして、IAM Identity Center AWS Organizations で を使用して、すべての へのアクセスを管理することをお勧めします AWS アカウント。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティベストプラクティス](#)」を参照してください。

IAM Identity Center でユーザーを作成するか、Microsoft Active Directory を使用するか、SAML 2.0 ID プロバイダー (IdP) を使用するか、IdP を個別にフェデレーションできます AWS アカウント。これらのアプローチのいずれかを使用して、ユーザーにシングルサインオンのエクスペリエンスを提供できます。多要素認証 (MFA) を適用し、一時的な認証情報を使用して AWS アカウント アクセスすることもできます。これは IAM ユーザーとは異なります。IAM ユーザーは、共有できる長期的な認証情報であり、AWS リソースに対するセキュリティリスクが高まる可能性があります。

サンドボックス環境専用の IAM ユーザーを作成する

を初めて使用する場合は AWS、テスト IAM ユーザーを作成し、それを使用してチュートリアルを実行し、 が提供する AWS ものを調べることができます。学習中はこの種の資格情報を使用しても問題ありませんが、サンドボックス環境以外では使用しないことをお勧めします。

以下のユースケースでは、 で IAM ユーザーの使用を開始するのが理にかなっている場合があります AWS。

- AWS SDK またはツールの使用を開始し、AWS のサービス サンドボックス環境で探索する。
- 学習の一環として、人間によるサインインプロセスをサポートしない、スケジュールされたスク립ト、ジョブ、その他の自動プロセスを実行する。

これらのユースケース以外で IAM ユーザーを使用している場合は、IAM Identity Center に移行するか、ID プロバイダーを AWS アカウント できるだけ早く にフェデレーションします。詳細については、「[AWSでの ID フェデレーション](#)」を参照してください。

IAM ユーザーのアクセスキーを保護する

IAM ユーザーのアクセスキーは定期的に更新する必要があります。「IAM ユーザーガイド」の「[アクセスキーの更新](#)」のガイダンスに従ってください。IAM ユーザーのアクセスキーを誤って共有したと思われる場合は、アクセスキーを更新してください。

IAM ユーザーアクセスキーは、ローカルマシンの共有 AWS credentialsファイルに保存する必要があります。IAM ユーザーのアクセスキーをコードに保存しないでください。IAM ユーザーのアクセスキーを含む設定ファイルは、いずれのソースコード管理ソフトウェアにも含めないでください。オープンソースプロジェクトの [git-secrets](#) などの外部ツールを使用すると、機密情報を誤って Git リポジトリにコミットすることを防ぐことができます。詳細については、「IAM ユーザーガイド」の「[IAM アイデンティティ \(ユーザー、ユーザーグループ、ロール\)](#)」を参照してください。

次の手順

学習のためのタスク	トピック
IDE AWS Cloud9 の使用方法を説明します。	開始方法: ベーシックチュートリアル および IDE を操作する
より高度なタスク	トピック
AWS Cloud9 開発環境を作成し、IDE AWS Cloud9 を使用して新しい環境でコードを操作します。	環境を作成する
リアルタイムでチャットサポートを使用し、他のユーザーを招待して一緒に新しい環境を使用します。	共有環境を使用する

のチームのセットアップ AWS Cloud9

このトピックでは、を使用して[AWS IAM アイデンティティセンター](#)、単一の内の複数のユーザーが AWS アカウント を使用できるようにする方法について説明します AWS Cloud9。他の使用パターン AWS Cloud9 に使用するようにを設定するには、正しい手順[セットアップ AWS Cloud9](#)については、「」を参照してください。

これらの手順では、単一の AWS アカウントへの管理アクセスを持っているか、またはこれから取得することを前提としています。詳細については、「IAM [ユーザーガイド](#)」の [AWS アカウント「ルートユーザー」](#) および [「最初の管理者とグループの作成」](#) を参照してください。が既にある AWS アカウント が、アカウントへの管理アクセス権がない場合は、AWS アカウント 管理者にお問い合わせください。

Warning

セキュリティリスクを避けるため、専用ソフトウェアを開発するときや実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM アイデンティティセンター](#) などの ID プロバイダーとのフェデレーションを使用してください。

Note

IAM の代わりに IAM [Identity Center](#) を使用して、単一の内の複数のユーザーが AWS アカウント を使用できるようにします AWS Cloud9。この使用パターンでは、単一の AWS アカウント が組織の管理アカウントとして機能します AWS Organizations。さらに、その組織にはメンバーアカウントがありません。IAM Identity Center を使用するには、このトピックをスキップして、代わりに「[エンタープライズセットアップ](#)」の指示に従ってください。関連情報については、以下のリソースを参照してください。

- AWS Organizations ユーザーガイドの [AWS Organizations とは](#) (IAM Identity Center では使用する必要があります AWS Organizations)
- AWS IAM アイデンティティセンター ユーザーガイドの「[AWS IAM アイデンティティセンターとは](#)」
- YouTube の 4 分間の動画「[AWS ナレッジセンター動画: AWS Organizationsを使用開始する方法](#)」
- YouTube の [IAM Identity Center を使用して複数の AWS アカウントへのユーザーアクセスを管理する](#) 7 分間の動画

- YouTube の 9 分間の動画「[オンプレミスのActive Directoryユーザーに IAM Identity Center をセットアップする方法](#)」

1 つの で複数のユーザーが の使用 AWS アカウント を開始できるようにするには AWS Cloud9、使用している AWS リソースのステップを開始します。

AWS アカウントをお持ちですか？	そのアカウントには、少なくとも 1 つ以上の IAM グループおよびユーザーがありますか。	このステップから開始します
いいえ	—	ステップ 1: にサインアップする AWS アカウント
はい	なし	ステップ 2: IAM グループとユーザーを作成し、ユーザーをグループに追加する
はい	はい	ステップ 3: AWS Cloud9 グループにアクセス許可を追加する

トピック

- [前提条件](#)
- [ステップ 1: IAM グループとユーザーを作成し、ユーザーをグループに追加する](#)
- [ステップ 2: AWS Cloud9 グループにアクセス許可を追加する](#)
- [ステップ 3: AWS Cloud9 コンソールにサインインする](#)
- [次の手順](#)

前提条件

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

にサインアップすると AWS アカウント、AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、[ルートユーザーアクセスが必要なタスク](#)の実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。<https://aws.amazon.com/> の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 のセキュリティを確保し AWS IAM アイデンティティセンター、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、AWS アカウント E メールアドレスを入力して、アカウント所有者[AWS マネジメントコンソール](#)として にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「[ルートユーザーとしてサインインする](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、IAM [ユーザーガイドの AWS アカウント 「ルートユーザー \(コンソール\) の仮想 MFA デバイス](#)を有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[AWS IAM アイデンティティセンターの有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

を ID ソース IAM アイデンティティセンターディレクトリ として使用方法のチュートリアルについては、「AWS IAM アイデンティティセンター ユーザーガイド」の「[デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ](#)」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「ユーザーガイド」の [AWS 「アクセスポータルにサインイン」](#) を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[アクセス許可セットを作成する](#)」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[グループを追加する](#)」を参照してください。

ステップ 1: IAM グループとユーザーを作成し、ユーザーをグループに追加する

このステップでは、AWS Identity and Access Management (IAM) でグループとユーザーを作成し、そのユーザーをグループに追加してから、そのユーザーを使用して にアクセスします AWS Cloud9。これは AWS セキュリティのベストプラクティスです。詳細については、「IAM ユーザーガイド」の「[IAM のベストプラクティス](#)」を参照してください。

必要なすべての IAM グループとユーザーが既にある場合は、[「ステップ 3: AWS Cloud9 グループにアクセス許可を追加する」](#)に進みます。

Note

組織で IAM グループとユーザーを設定済みである場合があります。組織に AWS アカウント管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。

これらのタスクは、[AWS マネジメントコンソール](#) または [AWS コマンドラインインターフェイス \(AWS CLI\)](#) を使って完了できます。

以下のコンソール手順に関連する 9 分間の動画を視聴するには、[「IAM ユーザーをセットアップし、YouTube で IAM 認証情報 AWS マネジメントコンソール を使用して にサインインする方法」](#)を参照してください。YouTube

ステップ 1.1: コンソールで IAM グループを作成する

1. まだサインインしていない場合は AWS マネジメントコンソール、<https://console.aws.amazon.com/codecommit> で にサインインします。

Note

AWS アカウント の作成時に提供された E メールアドレスとパスワード AWS マネジメントコンソール を使用して にサインインできます。これはルートユーザーとしての署名と呼ばれます。ただし、これは AWS セキュリティのベストプラクティスではありません。今後は、AWS アカウントの管理者ユーザーの認証情報を使用してサインインすることをお勧めします。管理者ユーザーは、AWS アカウント ルートユーザーと同様の AWS アクセス許可を持ち、関連するセキュリティリスクの一部を回避します。管理者ユーザーとしてサインインできない場合は、AWS アカウント 管理者に確認してください。詳細について

ては、IAM ユーザーガイドの「[最初の IAM ユーザーおよびグループの作成](#)」を参照してください。

2. [IAM コンソール] を開きます。これを行うには、AWS ナビゲーションバーでサービスを選択します。次に、[IAM] を選択します。
3. IAM コンソールのナビゲーションペインで、[グループ] を選択します。
4. [Create New Group (新しいグループの作成)] を選択します。
5. [グループ名の設定] ページで、[グループ名] に新しいグループの名前を入力します。
6. [Next Step] (次のステップ) をクリックします。
7. [ポリシーのアタッチ] ページで、ポリシーをアタッチせずに [次のステップ] を選択します。
ステップ [3: AWS Cloud9 グループにアクセス許可を追加するでポリシーをアタッチします](#)。
8. グループを作成 を選択します。

 Note

この手順を繰り返して少なくとも 2 つのグループを作成することをお勧めします。1 つは AWS Cloud9 ユーザー用、もう 1 つは AWS Cloud9 管理者用です。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

「[ステップ 2.2: IAM ユーザーを作成して、コンソールでグループにユーザーを追加する](#)」に進みます。

ステップ 1.2: を使用して IAM グループを作成する AWS CLI

 Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できます AWS CLI。

1. まだインストールしていない場合は、コンピュータ AWS CLI に をインストールして設定します。これを行うには、AWS Command Line Interface ユーザーガイドで以下の項目を参照してください。

- [AWS コマンドラインインターフェイスのインストール](#)
- [クイック設定](#)

 Note

AWS アカウント の作成時に提供された E メールアドレスとパスワードに関連付けられた認証情報 AWS CLI を使用して、 を設定できます。これはルートユーザーとしての署名と呼ばれます。ただし、これは AWS セキュリティのベストプラクティスではありません。代わりに、アカウントの IAM 管理者ユーザーの AWS 認証情報 AWS CLI を使用して を設定することをお勧めします。IAM 管理者ユーザーは、AWS アカウント ルートユーザーと同様の AWS アクセス許可を持ち、関連するセキュリティリスクの一部を回避します。を IAM 管理者ユーザー AWS CLI として設定できない場合は、AWS アカウント 管理者に確認してください。詳細については、IAM ユーザーガイドの「[最初の IAM 管理者ユーザー およびグループの作成](#)」を参照してください。

2. IAM create-group コマンドを実行して新しいグループの名前 (例: MyCloud9Group) を指定します。

```
aws iam create-group --group-name MyCloud9Group
```

 Note

この手順を繰り返して少なくとも 2 つのグループを作成することをお勧めします。1 つは AWS Cloud9 ユーザー用、もう 1 つは AWS Cloud9 管理者用です。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

ステップ [2.2: AWS CLI を使用して IAM ユーザーを作成し、そのユーザーをグループに追加します](#)。

ステップ 1.3: IAM ユーザーを作成して、コンソールでグループにユーザーを追加する

1. 前の手順で開いた IAM コンソールで、ナビゲーションペインの [ユーザー] を選択します。
2. [ユーザーを追加] を選択します。

3. [ユーザー名] に新しいユーザーの名前を入力します。

Note

[別のユーザーの追加] を選択することで、複数のユーザーを同時に作成できます。この手順のその他の設定は、これらの新しいユーザーのそれぞれに適用されます。

4. [プログラムによるアクセス] と [AWS マネジメントコンソール アクセス] チェックボックスを選択します。これにより、新しいユーザーがさまざまな AWS デベロッパー用ツールとサービスコンソールを使用できるようになります。
5. [Autogenerated password (自動生成パスワード)] のデフォルトの選択を維持します。これにより、新しいユーザーがコンソールにサインインするためのパスワードがランダムに作成されます。または、[Custom password] (カスタムパスワード) を選択して、新しいユーザー用のパスワードを入力します。
6. [パスワードのリセットが必要] のデフォルトの選択を維持します。このメッセージは、コンソールに初回サインインした後にパスワードを変更することを、新しいユーザーに促すものです。
7. [Next: Permissions (次へ: アクセス許可)] を選択します。
8. [グループにユーザーを追加] (複数のユーザーの場合も [グループにユーザーを追加]) をデフォルトの選択のままにしておきます。
9. グループのリストで、ユーザーを追加するグループの横にあるチェックボックス (名前ではなく) を選択します。
10. [Next: Review] を選択します。
11. [Create user] を選択します。または、複数のユーザーに [Create users] (ユーザーを作成) します。
12. ウィザードの最後のページで、次のいずれかの操作を行います。
 - 新しいユーザーの横にある [Eメールの送信] を選択し、画面の指示に従って新しいユーザーにコンソールのサインイン URL とユーザー名を E メールで送信します。次に、コンソールのサインインパスワード、AWS アクセスキー ID、シー AWS クレジットアクセスキーを新しい各ユーザーと個別に通信します。
 - [.csv のダウンロード] を選択します。次に、ダウンロードしたファイルにあるコンソールサインイン URL、コンソールサインインパスワード、AWS アクセスキー ID、シー AWS クレジットアクセスキーを新しい各ユーザーに伝えます。
 - それぞれの新しいユーザーの横で、[シークレットアクセスキー] と [パスワード] の両方で [表示] を選択します。次に、コンソールのサインイン URL、コンソールのサインインパス

ワード、AWS アクセスキー ID、シー AWS クレットアクセスキーを新しい各ユーザーに伝えます。

 Note

.csv のダウンロードを選択しない場合、新しいユーザーの AWS シークレットアクセスキーとコンソールのサインインパスワードを表示できるのは今回のみです。新しいユーザーの新しい AWS シークレットアクセスキーまたはコンソールサインインパスワードを生成するには、IAM ユーザーガイドの以下を参照してください。

- [アクセスキーの作成、変更、表示 \(コンソール\)](#)
- [IAM ユーザーパスワードの作成、変更、削除 \(コンソール\)](#)

13.作成したい追加の各 IAM ユーザーにこの手順を繰り返したら、「[ステップ 3: グループに AWS Cloud9 アクセス許可を追加する](#)」に進みます。

ステップ 1.4: IAM ユーザーを作成し、を使用してそのユーザーをグループに追加する AWS CLI

 Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できます AWS CLI。

1. IAM create-user コマンドを実行してユーザーを作成し、新しいユーザーの名前 (例: MyCloud9User) を指定します。

```
aws iam create-user --user-name MyCloud9User
```

2. IAM create-login-profile コマンドを実行して新しいコンソールにサインインするユーザーのパスワードを作成し、ユーザー名と最初のサインインパスワードを指定します (例: MyC10ud9Us3r!). ユーザーがサインインしたら、AWS は サインインパスワードの変更をユーザーに依頼します。

```
aws iam create-login-profile --user-name MyCloud9User --password MyC10ud9Us3r! --password-reset-required
```

後でユーザーのために交換用コンソールのサインインパスワードを生成する必要がある場合は、IAM ユーザーガイドの「[IAM ユーザーパスワードの作成、変更、削除 \(API、CLI、PowerShell\)](#)」を参照してください。

3. IAM create-access-key コマンドを実行して、ユーザーの新しい AWS アクセスキーと対応する AWS シークレットアクセスキーを作成します。

```
aws iam create-access-key --user-name MyCloud9User
```

表示されている [AccessKeyId] と [SecretAccessKey] の値をメモします。IAM create-access-key コマンドを実行した後、ユーザーの AWS シークレットアクセスキーを表示できるのは今回だけです。後でユーザーの新しい AWS シークレットアクセスキーを生成する必要がある場合は、IAM ユーザーガイドの「[アクセスキーの作成、変更、表示 \(API、CLI、PowerShell\)](#)」を参照してください。

4. IAM add-user-to-group コマンドを実行してユーザーをグループに追加し、グループおよびユーザーの名前を指定します。

```
aws iam add-user-to-group --group-name MyCloud9Group --user-name MyCloud9User
```

5. コンソールのサインイン URL、初期コンソールのサインインパスワード、AWS アクセスキー ID、シー AWS クレットアクセスキーをユーザーに伝えます。
6. 作成したい追加 IAM ユーザーそれぞれにこの手順を繰り返します。

ステップ 2: AWS Cloud9 グループにアクセス許可を追加する

デフォルトでは、ほとんどの IAM グループとユーザーは AWS のサービス、AWS Cloud9を含むにアクセスできません (例外は、AWS アカウント デフォルトで AWS のサービスのすべてのにアクセスできる IAM 管理者グループと IAM 管理者ユーザーです)。このステップでは、IAM を使用して、1 人以上のユーザーが属する AWS Cloud9 IAM グループにアクセス許可を直接追加します。これにより、それらのユーザーが AWS Cloud9にアクセスできるようになります。

Note

組織では、適切なアクセス許可を持つグループを設定済みである場合があります。組織に AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。

[AWS マネジメントコンソール](#)または[AWS CLI](#)を使ってこのタスクを完成できます。

ステップ 2.1: コンソールを使用して AWS Cloud9 グループにアクセス許可を追加する

1. まだサインインしていない場合は AWS マネジメントコンソール、<https://console.aws.amazon.com/codecommit> でサインインします。

Note

AWS アカウント の作成時に提供された E メールアドレスとパスワード AWS マネジメントコンソール を使用してサインインできます。これはルートユーザーとしての署名と呼ばれます。ただし、これは AWS セキュリティのベストプラクティスではありません。今後は、AWS アカウントの IAM 管理者ユーザーの認証情報を使用してサインインすることをお勧めします。管理者ユーザーは、AWS アカウント ルートユーザーと同様の AWS アクセス許可を持ち、関連するセキュリティリスクの一部を回避します。管理者ユーザーとしてサインインできない場合は、AWS アカウント 管理者に確認してください。詳細については、IAM ユーザーガイドの「[最初の IAM 管理者ユーザーおよびグループの作成](#)」を参照してください。

2. IAM コンソールを開きます。これを行うには、AWS ナビゲーションバーでサービスを選択します。次に、[IAM] を選択します。
3. [グループ] を選択します。
4. グループの名前を選択します。
5. AWS Cloud9 ユーザーまたは AWS Cloud9 管理者のアクセス許可をグループに追加するかどうかを決定します。これらのアクセス許可は、グループ内の各ユーザーに適用されます。

AWS Cloud9 ユーザーアクセス許可により、グループ内の各ユーザーは自分の 内で次の操作を実行できます AWS アカウント。

- 独自の AWS Cloud9 開発環境を作成します。
- 独自の環境に関する情報を取得する。

- 自分の環境設定を変更する。

AWS Cloud9 管理者アクセス許可により、グループ内の各ユーザーは自分の 内で追加の操作を実行できます AWS アカウント。

- 自分自身または他のユーザーの環境を作成する。
- 自分自身または他のユーザーの環境に関する情報を取得する。
- 自分自身または他のユーザーの環境を削除する。
- 自分自身または他のユーザーの環境の設定を変更する。

 Note

AWS Cloud9 管理者グループには、限定した数のユーザーのみ追加するようお勧めします。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

6. [許可] タブの [マネージドポリシー] で、[ポリシーのアタッチ] を選択します。
7. ポリシー名のリストで、AWS Cloud9 ユーザーアクセス許可の場合は AWSCloud9User の横にあるボックスを選択し、AWS Cloud9 管理者アクセス許可の場合は AWSCloud9Administrator を選択します。どちらのポリシー名もリストに表示されない場合は、[Filter] (フィルター) ボックスにポリシー名を入力して表示させます。
8. [ポリシーをアタッチ] を選択します。

 Note

AWS Cloud9 アクセス許可を追加するグループが複数ある場合は、それらのグループごとにこの手順を繰り返します。

これらの AWS 管理ポリシーがグループに付与するアクセス許可のリストを確認するには、「[AWS 管理 \(事前定義\) ポリシー](#)」を参照してください。

に必要な AWS アクセス許可に加えてグループに追加できるアクセス許可については AWS Cloud9、IAM ユーザーガイドの「[管理ポリシーとインラインポリシー](#)」および「[ポリシーによって付与されるアクセス許可について](#)」を参照してください。

[ステップ 4: AWS Cloud9 コンソールにサインインする](#)」に進みます。

ステップ 2.2: を使用して AWS Cloud9 グループにアクセス許可を追加する AWS CLI

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できますAWS CLI。

1. まだインストールしていない場合は、コンピュータ AWS CLI に をインストールして設定します。これを行うには、AWS Command Line Interface ユーザーガイドで以下の項目を参照してください。

- [AWS コマンドラインインターフェイスのインストール](#)
- [クイック設定](#)

Note

AWS アカウント の作成時に提供された E メールアドレスとパスワードに関連付けられた認証情報 AWS CLI を使用して、 を設定できます。これはルートユーザーとしての署名と呼ばれます。ただし、これは AWS セキュリティのベストプラクティスではありません。代わりに、 で IAM 管理者ユーザーの認証情報 AWS CLI を使用して を設定することをお勧めします AWS アカウント。IAM 管理者ユーザーは、AWS アカウント ルートユーザーと同様の AWS アクセス許可を持ち、関連するセキュリティリスクの一部を回避します。を管理者ユーザー AWS CLI として設定できない場合は、AWS アカウント 管理者に確認してください。詳細については、IAM ユーザーガイドの「[最初の IAM 管理者ユーザーおよびグループの作成](#)」を参照してください。

2. AWS Cloud9 ユーザーまたは AWS Cloud9 管理者のアクセス許可をグループに追加するかどうかを決定します。これらのアクセス許可は、グループ内の各ユーザーに適用されます。

AWS Cloud9 ユーザーアクセス許可により、グループ内の各ユーザーは自分の 内で次のことを実行できます AWS アカウント。

- 独自の AWS Cloud9 開発環境を作成します。
- 独自の環境に関する情報を取得する。
- 自分の環境設定を変更する。

AWS Cloud9 管理者アクセス許可により、グループ内の各ユーザーは自分の 内で追加の操作を実行できます AWS アカウント。

- 自分自身または他のユーザーの環境を作成する。
- 自分自身または他のユーザーの環境に関する情報を取得する。
- 自分自身または他のユーザーの環境を削除する。
- 自分自身または他のユーザーの環境の設定を変更する。

 Note

AWS Cloud9 管理者グループには、限定した数のユーザーのみ追加するようお勧めします。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

3. IAM attach-group-policy コマンドを実行し、グループの名前と追加する AWS Cloud9 アクセス許可ポリシーの Amazon リソースネーム (ARN) を指定します。

AWS Cloud9 ユーザーアクセス許可には、次の ARN を指定します。

```
aws iam attach-group-policy --group-name MyCloud9Group --policy-arn
arn:aws:iam::aws:policy/AWSCloud9User
```

AWS Cloud9 管理者アクセス許可には、次の ARN を指定します。

```
aws iam attach-group-policy --group-name MyCloud9Group --policy-arn
arn:aws:iam::aws:policy/AWSCloud9Administrator
```

 Note

AWS Cloud9 アクセス許可を追加するグループが複数ある場合は、それらのグループごとにこの手順を繰り返します。

これらの AWS 管理ポリシーがグループに付与するアクセス許可のリストを確認するには、[AWS 「管理 \(事前定義\) ポリシー」](#) を参照してください。

に必要な AWS アクセス許可に加えてグループに追加できるアクセス許可については AWS Cloud9、IAM ユーザーガイドの「[管理ポリシーとインラインポリシー](#)」および「[ポリシーによって付与されるアクセス許可の理解](#)」を参照してください。

ステップ 3: AWS Cloud9 コンソールにサインインする

このトピックの前のステップを完了すると、ユーザーとユーザーは AWS Cloud9 コンソールにサインインする準備が整います。

1. AWS アカウント ルートユーザー AWS マネジメントコンソール として に既にサインインしている場合は、コンソールからサインアウトします。
2. <https://console.aws.amazon.com/cloud9/> で AWS Cloud9 コンソールを開きます。
3. 先ほど作成または識別した IAM ユーザーの AWS アカウント 番号を入力し、次へを選択します。

Note

AWS アカウント番号を入力するオプションが表示されない場合は、別のアカウントにサインインを選択します。次のページで AWS アカウント を入力し、[Next] (次へ) を選択します。

4. 先ほど作成または識別した IAM ユーザーのサインイン認証情報を入力し、[Sign In] (サインイン) を選択します。
5. プロンプトが表示されたら、画面の指示に従って、ユーザーの最初のサインインパスワードを変更します。新しいサインインパスワードを安全な場所に保存します。

AWS Cloud9 コンソールが表示され、使用を開始できます AWS Cloud9。

次の手順

タスク	次のトピックを参照
コストを制御するために AWS アカウント、内の他のユーザー AWS Cloud9 の使用を制限します。	追加のセットアップオプション

タスク	次のトピックを参照
AWS Cloud9 開発環境を作成し、IDE AWS Cloud9 を使用して新しい環境でコードを操作します。	環境を作成する
IDE AWS Cloud9 の使用方法を説明します。	開始方法: ベーシックチュートリアル 、および IDE を操作する
リアルタイムでチャットサポートを使用し、他のユーザーを招待して一緒に新しい環境を使用します。	共有環境を使用する

のエンタープライズセットアップ AWS Cloud9

このトピックでは、[AWS IAM アイデンティティセンター](#)を使用して 1 つ以上の AWS アカウントをエンタープライズ AWS Cloud9 内で使用できるようにする方法について説明します。他の使用パターン AWS Cloud9 に使用するように をセットアップするには、正しい手順[セットアップ AWS Cloud9](#)については、「」を参照してください。

Warning

セキュリティリスクを避けるため、専用ソフトウェアを開発するときや実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、[AWS IAM アイデンティティセンター](#)などの ID プロバイダーとのフェデレーションを使用してください。

これらの手順では、AWS Organizationsで組織への管理アクセス権を持っているか、または持つであろうことを前提としています。で組織への管理アクセスがまだない場合は AWS Organizations、AWS アカウント 管理者にお問い合わせください。詳細については、以下のリソースを参照してください。

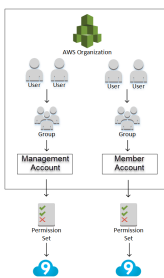
- [AWS Organizations ユーザーガイドの Organization のアクセス許可の管理](#) (IAM Identity Center では を使用する必要があります AWS Organizations) AWS
- AWS IAM アイデンティティセンター ユーザーガイドの「[IAM Identity Center リソースへのアクセス許可の管理の概要](#)」

- [マルチ](#) AWS Control Tower AWS アカウント環境のセットアップと管理に使用できるサービスである [AWS IAM アイデンティティセンター](#) を使用します。は AWS のサービス、ランディングゾーンを 1 時間未満で構築 AWS Service Catalog AWS IAM アイデンティティセンターするなど AWS Organizations、他の の機能 AWS Control Tower を使用します。

このトピックに関連する基本情報については、以下のリソースを参照してください。

- AWS Organizations ユーザーガイドの [AWS Organizations とは](#) (IAM Identity Center では [を使用する](#) 必要があります AWS Organizations)
- AWS IAM アイデンティティセンター ユーザーガイドの「[AWS IAM アイデンティティセンターとは](#)」
- [AWS Control Tower ユーザーガイドの](#) AWS 「Control Tower の開始方法」
- 4 分間の動画 [AWS ナレッジセンターの動画: YouTube で AWS Organizations の使用を開始する方法](#) YouTube
- YouTube で [を使用して複数の AWS アカウントへのユーザーアクセスを管理する AWS IAM アイデンティティセンター](#) 7 分間の動画
- YouTube で [オンプレミス Active Directory ユーザーに AWS シングルサインオンを設定する方法に関する](#) 9 分間の動画 YouTube

次の概念図は、最終的なセットアップ結果を示しています。



1 つ以上の が エンタープライズ AWS Cloud9 内で の使用 AWS アカウント を開始できるようにするには、既に持っている AWS リソースに従ってステップに従います。

組織の管理 アカウント AWS アカ ウントと して機能す るはありま すか AWS Organizat ions?	その管理 アカウン トの AWS Organizations に組織はあり ますか?	その組織に必 要な AWS ア カウント メ ンバーは全員 ですか?	その組織は IAM Identity Center を使 用するように セットアップ されています か。	その組織は、 AWS Cloud9 を使用するす べての必要な グループおよ びユーザーで 構成されてい ますか。	このステップ から開始しま す
いいえ	—	—	—	—	ステップ 1: 組織の管理ア カウントを作 成する
はい	なし	—	—	—	ステップ 2: 管理アカウン トの組織を作 成する
はい	あり	なし	—	—	ステップ 3: 組織にメン バーアカウン トを追加する
はい	はい	あり	なし	—	ステップ 4: 組織全体で IAM Identity Center を有 効にする
はい	はい	はい	あり	なし	ステップ 5: 組織内のグ ループとユー ザーをセット アップする

組織の管理アカウント AWS アカウントとして機能する はありますか AWS Organizations?	その管理アカウントの AWS Organizations に組織はありますか?	その組織に必要な AWS アカウント メンバーは全員ですか?	その組織は IAM Identity Center を使用するように セットアップ されていますか。	その組織は、AWS Cloud9 を使用するすべての必要なグループおよびユーザーで構成されていますか。	このステップから開始します
はい	はい	はい	はい	はい	ステップ 6: 組織内のグループとユーザーが使用できるようにする AWS Cloud9

ステップ 1: 組織の管理アカウントを作成する

Note

エンタープライズで管理アカウントが既にお客様に対して設定されている場合があります。エンタープライズに AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。管理アカウントが既にある場合は、「[ステップ 2: 管理アカウントの組織を作成する](#)」までスキップします。

AWS IAM アイデンティティセンター (IAM Identity Center) を使用するには、が必要です AWS アカウント。AWS アカウント は、 の組織の管理アカウントとして機能します AWS Organizations。詳細については、AWS Organizations ユーザーガイドの「[AWS Organizations の用語と概念](#)」で管理アカウントに関する説明を参照してください。

次の手順に関連した 4 分間の動画をご覧になるには、YouTube の「[Amazon Web Services アカウントの作成](#)」を参照してください。

管理アカウントの作成:

1. <https://aws.amazon.com/> に移動します。
2. [コンソールにサインインする] を選択します。
3. [AWS アカウントの作成] を選択します。
4. 画面上の指示に従ってプロセスを完了します。この中で、AWS に E メールアドレスとクレジットカード情報を提供します。また、電話を使用して、AWS が提供するコードを入力する必要があります。

アカウントの作成が完了すると、AWS から確認メールが送信されます。この確認が完了するまで、次のステップに進まないでください。

ステップ 2: 管理アカウントの組織を作成する

Note

管理アカウントを使用するようにエンタープライズが既に AWS Organizations をセットアップしている可能性があります。エンタープライズに AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。管理アカウントを使用するようにすでに AWS Organizations 設定している場合は、[ステップ 3: メンバーアカウントを組織に追加するに進んでください](#)。

IAM Identity Center を使用するには、に管理アカウント AWS Organizations を使用する組織が必要です。詳細については、AWS Organizations ユーザーガイドの「[AWS Organizations の用語と概念](#)」で、組織に関する説明を参照してください。

管理 AWS Organizations 用の組織を で作成するには AWS アカウント、AWS Organizations ユーザーガイドの以下の手順に従ってください。

1. [組織の作成](#)
2. [組織内のすべての機能の有効化](#)

これらの手順に関連する 4 分間の動画を見るには、[AWS 「ナレッジセンターの動画: YouTube で AWS Organizations の使用を開始する方法」](#)を参照してください。YouTube

ステップ 3: 組織にメンバーアカウントを追加する

Note

エンタープライズは、必要なメンバーアカウントで既に AWS Organizations をセットアップしている可能性があります。エンタープライズに AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。必要なメンバーアカウントを既に AWS Organizations セットアップしている場合は、[「ステップ 4: 組織全体で IAM アイデンティティセンターを有効にする」に進んでください](#)。

このステップでは、組織のメンバーアカウント AWS アカウント として機能する を追加します AWS Organizations。詳細については、AWS Organizations ユーザーガイドの「[AWS Organizations の用語と概念](#)」で、メンバーアカウントに関する説明を参照してください。

Note

組織にメンバーアカウントを追加する必要はありません。組織の単一の管理アカウントだけで IAM Identity Center を使用できます。必要に応じて、後でメンバーアカウントを組織に追加できます。今はメンバーアカウントを追加しない場合は、[「ステップ 4: 組織全体で IAM Identity Center を有効にする」](#)までスキップします。

でメンバーアカウントを組織に追加するには AWS Organizations、AWS Organizations 「ユーザーガイド」の以下の一連の指示のいずれかまたは両方に従います。組織のメンバーとして必要な がすべて揃うまで AWS アカウント 、必要な回数だけこれらの手順を繰り返します。

- [組織 AWS アカウント での の作成](#)
- [を組織に招待 AWS アカウント する](#)

ステップ 4: 組織全体で IAM Identity Center を有効にする

Note

IAM Identity Center を使用するようにエンタープライズが既に AWS Organizations をセットアップしている可能性があります。エンタープライズに AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。IAM Identity Center を使用

するように、すでに AWS Organizations を設定している場合は、[ステップ 5 に進みます。組織内のグループとユーザーをセットアップする](#)」までスキップします。

このステップでは、の組織が IAM Identity Center AWS Organizations を使用できるようにします。これを行うには、AWS IAM アイデンティティセンター ユーザーガイドの以下の手順に従います。

1. [IAM Identity Center の前提条件](#)
2. [IAM Identity Center を有効にする](#)

ステップ 5: 組織内のグループとユーザーをセットアップする

Note

エンタープライズでは、IAM Identity Center ディレクトリ、またはで管理されている AWS Managed Microsoft AD または AD Connector ディレクトリのグループとユーザーがすでに AWS Organizations セットアップされている可能性があります AWS Directory Service。エンタープライズに AWS アカウント 管理者がある場合は、次の手順を開始する前に、その管理者に確認してください。IAM Identity Center ディレクトリまたはからグループとユーザーを既に AWS Organizations セットアップしている場合は AWS Directory Service、[ステップ 6 に進みます。組織内のグループとユーザーが AWS Cloud9を使用できるようにする](#)」にスキップします。

このステップでは、組織用に IAM Identity Center ディレクトリにグループとユーザーを作成します。または、AWS Directory Service 組織の で管理されている AWS Managed Microsoft AD または AD Connector ディレクトリに接続します。後のステップで、AWS Cloud9を使用するために必要なアクセス許可をグループに付与します。

- 組織の IAM Identity Center ディレクトリを使用する場合は、AWS IAM アイデンティティセンター ユーザーガイドの以下の手順に従います。必要なグループとユーザーがすべて揃うまで、これらのステップを必要な回数だけ繰り返します。

1. [グループを追加します](#)。組織全体の AWS Cloud9 管理者には、少なくとも 1 つのグループを作成することをお勧めします。次に、このステップを繰り返して、組織全体のすべての AWS Cloud9 ユーザーに別のグループを作成します。オプションで、このステップを繰り返して、既存の AWS Cloud9 開発環境を共有する組織全体のすべてのユーザーの 3 番目のグループを作成することもできます。ただし、ユーザーが自分自身で環境を作成する

ことは許可しないようにします。わかりやすいように、これらのグループにはそれぞれ `AWSCloud9Administrators`、`AWSCloud9Users`、`AWSCloud9EnvironmentMembers` という名前を付けることをお勧めします。詳細については、「[AWS Cloud9用AWS マネージドポリシー](#)」を参照してください。

2. [ユーザーを追加します](#)。

3. [グループにユーザーを追加します](#)。 `AWSCloud9Administrators` グループに AWS Cloud9 管理者を追加し、このステップを繰り返してユーザーを `AWSCloud9Users` グループに追加 AWS Cloud9 します。必要に応じて、このステップを繰り返して残りのユーザーを `AWSCloud9EnvironmentMembers` グループに追加します。グループへのユーザーの追加は、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立つ AWS セキュリティのベストプラクティスです。

- 組織の で管理している AWS Managed Microsoft AD または AD Connector ディレクトリを使用している場合は、「AWS IAM アイデンティティセンター ユーザーガイド AWS Directory Service」の「[Microsoft AD ディレクトリに接続する](#)」を参照してください。

ステップ 6. 組織内のグループとユーザーが 使用できるようにする AWS Cloud9

デフォルトでは、 の組織内のほとんどのユーザーとグループは AWS のサービス、 を含む にアクセス AWS Organizations できません AWS Cloud9。このステップでは、IAM Identity Center を使用して、 の組織全体のグループとユーザーが AWS Organizations 、参加アカウントの任意の組み合わせ AWS Cloud9 内で 使用できるようにします。

1. [\[IAM Identity Center console\]](#) (IAM Identity Center コンソール) で、サービスナビゲーションページの [AWS アカウント] を選択します。
2. [Permission sets] (アクセス許可セット) タブを選択します。
3. [Create permission set] (アクセス許可セットを作成) を選択します。
4. [Create a custom permission set] (カスタムアクセス許可セットを作成) を選択します。
5. このアクセス許可セットの名前を入力します。組織全体の AWS Cloud9 管理者には、少なくとも 1 つのアクセス許可セットを作成することをお勧めします。次に、この手順のステップ 3 ~ 10 を繰り返して、組織全体のすべての AWS Cloud9 ユーザーに別のアクセス許可セットを作成します。必要に応じて、この手順のステップ 3 ~ 10 を繰り返して、既存の AWS Cloud9 開発環境を共有する組織全体のすべてのユーザーに 3 番目のアクセス許可セットを作成することもできます。ただし、ユーザーが独自に環境を作成することは許可しないようにします。わかりやすいように、これらのアクセス許可セットにそれぞれ

AWSCloud9AdministratorsPerms、AWSCloud9UsersPerms、AWSCloud9EnvironmentMembersPerms という名前を付けることをお勧めします。詳細については、「[AWS Cloud9用AWS マネージドポリシー](#)」を参照してください。

6. 必要に応じて、アクセス許可セットの説明を入力します。
7. アクセス許可セットの [Session duration] (セッション期間) を選択するか、デフォルトのセッション期間 (1 時間) のままにします。
8. AWS 管理ポリシーのアタッチを選択します。
9. ポリシーのリストで、正しい [ポリシー名] エントリの横にある以下のボックスのいずれかを選択します。(ポリシー名そのものを選択しないでください。リストにポリシー名が表示されていない場合は、[検索] ボックスにポリシー名を入力して表示します)。
 - AWSCloud9AdministratorsPerms アクセス許可セットとして、[AWSCloud9Administrator] を選択します。
 - AWSCloud9UsersPerms アクセス許可セットとして、[AWSCloud9User] を選択します。
 - オプションで、AWSCloud9EnvironmentMembersPerms アクセス許可セットとして、[AWSCloud9EnvironmentMember] を選択します。

 Note

で必要なポリシーに加えて追加できるポリシーについては AWS Cloud9、IAM ユーザーガイドの「[管理ポリシーとインラインポリシー](#)」および「[ポリシーによって付与されるアクセス許可について](#)」を参照してください。

- 10[作成] を選択します。
- 11必要なすべてのアクセス許可セットの作成が完了したら、AWS 組織タブで、AWS Cloud9 アクセス許可 AWS アカウント を割り当てる を選択します。[AWS 組織] タブが表示されない場合は、サービスナビゲーションペインで、[AWS アカウント] を選択します。[AWS 組織] タブが表示されます。
- 12[ユーザーの割り当て] を選択します。
- 13.Groups タブで、AWS Cloud9 アクセス許可を割り当てるグループの名前の横にあるボックスを選択します。グループ名自体を選択しないでください。
 - 組織の IAM Identity Center ディレクトリを使用する場合、AWS Cloud9 管理者用に AWSCloud9Administrators という名前のグループが既に作成されている場合があります。
 - AWS Directory Service 組織の で管理している AWS Managed Microsoft AD または AD Connector ディレクトリを使用している場合は、ディレクトリの ID を選択します。次に、グループ名の一部または全部を入力して、[Search connected directory] (接続されたディレクトリ

を検索) を選択します。最後に、AWS Cloud9 アクセス許可を割り当てるグループの名前の横にあるボックスを選択します。

 Note

個々のユーザーではなく グループにアクセス AWS Cloud9 許可を割り当てることをお勧めします。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

14[Next: Permissions sets] (次へ: アクセス許可セット) を選択します。

15.このグループに割り当てるアクセス許可セットの名前の横にあるボックスを選択します (たとえば、AWS Cloud9 管理者グループの AWSCloud9AdministratorsPerms)。アクセス許可セット自体を選択しないでください。

16[Finish] を選択してください。

17 続行 AWS アカウントを選択します。

18 組織全体で に割り当てる追加の AWS Cloud9 アクセス許可については、この手順のステップ 11 ~ 17 AWS アカウント を繰り返します。

ステップ 7: の使用を開始する AWS Cloud9

このトピックの前のステップを完了すると、ユーザーとユーザーは IAM Identity Center にサインインして の使用を開始する準備が整います AWS Cloud9。

1. AWS アカウントまたは IAM Identity Center に既にサインインしている場合は、サインアウトします。これを行うには、AWS サポートウェブサイトの[AWS 「アカウントからサインアウトする方法」](#)または「[ユーザーガイド](#)」の「[ユーザーポータルからサインアウトする方法](#)」を参照してください。AWS IAM アイデンティティセンター
2. IAM Identity Center にサインインするには、AWS IAM アイデンティティセンター ユーザーガイドの「[IAM アイデンティティセンターへの参加招待を受け入れる方法](#)」の手順に従ってください。これには、固有のサインイン URL にアクセスして固有の認証情報でサインインすることが含まれます。AWS アカウント 管理者はこの情報を E メールで送信するか、お客様に提供します。

 Note

提供された固有のサインイン URL を必ずブックマークしてください。こうすることで、後で簡単に戻ることができます。また、この URL の固有のサインイン認証情報は安全な場所に保管してください。

この URL、ユーザー名、パスワードの組み合わせは、AWS アカウント 管理者が付与する AWS Cloud9 アクセス許可のレベルによって異なる場合があります。例えば、1 つのアカウントへの AWS Cloud9 管理者アクセスを許可するには、1 つの URL、ユーザー名、およびパスワードを使用できます。別のアカウントへの AWS Cloud9 ユーザーアクセスのみを許可する別の URL、ユーザー名、パスワードを使用できます。

3. IAM Identity Center にサインインしたら、[AWS アカウント] タイルを選択します。
4. 表示されたドロップダウンリストからユーザーの表示名を選択します。複数の名前が表示された場合は、使用を開始する名前を選択します AWS Cloud9。どちらの名前を選択するべきかわからない場合は、AWS アカウント 管理者にお問い合わせください。
5. ユーザーの表示名の横にある [管理コンソール] リンクを選択します。複数の [Management console] (管理コンソール) リンクが表示されている場合は、正しいアクセス許可セットの横にあるリンクを選択してください。これらのリンクのどれを選択するかわからない場合は、AWS アカウント 管理者にお問い合わせください。
6. から AWS マネジメントコンソール、次のいずれかを実行します。
 - 既に表示されている場合は、[Cloud9] を選択します。
 - [すべてのサービス] を展開し、[Cloud9] を選択します。
 - [Find services (サービスの検索)] ボックスに、「Cloud9」と入力し、Enter を押します。
 - AWS ナビゲーションバーで、サービスを選択し、Cloud9 を選択します。

AWS Cloud9 コンソールが表示され、使用を開始できます AWS Cloud9。

次の手順

タスク	次のトピックを参照
AWS Cloud9 開発環境を作成し、IDE AWS Cloud9 を使用して新しい環境でコードを操作します。	環境を作成する

タスク	次のトピックを参照
IDE AWS Cloud9 の使用方法を説明します。	開始方法: ベーシックチュートリアル および IDE を操作する
リアルタイムでチャットサポートを使用し、他のユーザーを招待して一緒に新しい環境を使用します。	共有環境を使用する

の追加のセットアップオプション AWS Cloud9

このトピックでは、[\[Team Setup\]](#) (チームセットアップ) または [\[Enterprise Setup\]](#) (エンタープライズセットアップ) のセットアップステップを完了していることを前提としています。

[チームセットアップ](#) または [エンタープライズセットアップ](#) で、グループを作成し、それらのグループにアクセス AWS Cloud9 許可を直接追加しました。これは、それらのグループのユーザーが AWS Cloud9 にアクセスできるようにするためです。このトピックでは、さらにアクセス許可を追加して、それらのグループのユーザーが作成できる環境の種類を制限します。これにより、AWS Cloud9 AWS アカウントや組織のに関連するコストを制御できます。

これらのアクセス許可を追加するには、強制したい AWS アクセス許可を定義する独自の一連のポリシーを作成します。このような各ポリシーをカスタマー管理ポリシーと呼びます。次に、これらのカスタマー管理ポリシーを、ユーザーが含まれるグループにアタッチします。シナリオによっては、それらのグループに既にアタッチされている既存の AWS 管理ポリシーをデタッチする必要もあります。これを設定するには、このトピックの手順に従います。

Note

次の手順では、AWS Cloud9 ユーザーのみのポリシーのアタッチとデタッチについて説明します。これらの手順は、別の AWS Cloud9 ユーザーグループと AWS Cloud9 管理者グループが既にあることを前提としています。AWS Cloud9 管理者グループのユーザー数は限定されていることを前提としています。この AWS セキュリティのベストプラクティスは、AWS リソースアクセスの問題をより適切に制御、追跡、トラブルシューティングするのに役立ちます。

ステップ 1: カスタマー管理ポリシーを作成する

カスタマー管理ポリシーは、[AWS マネジメントコンソール](#) または [AWS コマンドラインインターフェイス \(AWS CLI\)](#) を使って作成できます。

Note

このステップでは、IAM グループ用のカスタマー管理ポリシーを作成する方法のみを示します。グループのカスタムアクセス許可セットを作成するには AWS IAM アイデンティティセンター、このステップをスキップし、AWS IAM アイデンティティセンター「ユーザーガイド」の「[アクセス許可セットの作成](#)」の手順に従います。このトピックでは、カスタムアクセス許可セットを作成する手順を示します。関連するカスタムアクセス許可ポリシーは、このトピックの後半にある「[AWS Cloud9を使用したチームのカスタマー管理ポリシーの例](#)」を参照してください。

ステップ 1.1: コンソールを使用してカスタマー管理ポリシーを作成する

1. まだサインインしていない場合は AWS マネジメントコンソール、にサインインします。

AWS アカウントの管理者ユーザーの認証情報を使用してサインインすることをお勧めします。これができない場合は、AWS アカウント 管理者に確認してください。

2. [IAM コンソール] を開きます。これを行うには、コンソールのナビゲーションバーで、[サービス] を選択します。次に、[IAM] を選択します。
3. サービスのナビゲーションペインで、[ポリシー] を選択します。
4. [Create policy] (ポリシーを作成) を選択します。
5. [JSON] タブに、提案された[カスタマー管理ポリシーの例](#)のいずれかを貼り付けます。

Note

独自のカスタマーマネージドポリシーを作成することもできます。詳細については、「IAM ユーザーガイド」と AWS のサービスの[ドキュメント](#)の「[IAM JSON ポリシーリファレンス](#)」を参照してください。

6. [ポリシーの確認] を選択します。
7. [ポリシーの確認] ページで、ポリシーの [名前] と [説明] (省略可能) を入力して、[ポリシーの作成] を選択します。

作成したい追加のカスタマー管理ポリシーそれぞれに対してこのステップを繰り返します。次に、「[コンソールを使用してカスタマー管理ポリシーをグループに追加する](#)」までスキップします。

ステップ 1.2: を使用してカスタマー管理ポリシーを作成する AWS CLI

1. を実行するコンピュータで AWS CLI、ポリシーを記述するファイルを作成します (例: `policy.json`)。

別のファイル名を使用してファイルを作成する場合は、この手順全体でそれを置き換えてください。

2. `policy.json` ファイルに、提案された[カスタマー管理ポリシーの例](#)のいずれかを貼り付けます。

Note

独自のカスターマネージドポリシーを作成することもできます。詳細については、IAM ユーザーガイドの[IAM JSON ポリシーリファレンス](#)と AWS サービスの「[ドキュメント](#)」を参照してください。

3. ターミナルまたはコマンドプロンプトで、`policy.json` ファイルが格納されているディレクトリに移動します。
4. ポリシーの名前および `create-policy` ファイルを指定して、IAM `policy.json` コマンドを実行します。

```
aws iam create-policy --policy-document file://policy.json --policy-name MyPolicy
```

前述のコマンドで、`MyPolicy` をポリシーの名前に置き換えます。

「」をスキップして、[を使用してカスタマー管理ポリシーをグループに追加します AWS CLI](#)。

ステップ 2: カスターマネージドポリシーをグループに追加する

カスタマー管理ポリシーをグループに追加するには、「[AWS マネジメントコンソール](#)」または「[AWS コマンドラインインターフェイス \(AWS CLI\)](#)」を使用します。詳細については、「[を使用するチームのカスタマー管理ポリシーの例 AWS Cloud9](#)」を参照してください。

Note

このステップでは、IAM グループにカスタマー管理ポリシーを追加する方法のみを示します。のグループにカスタムアクセス許可セットを追加するには AWS IAM アイデンティティセンター、このステップをスキップし、代わりにAWS IAM アイデンティティセンター「ユーザーガイド」の「[ユーザーアクセスの割り当て](#)」の手順に従います。

ステップ 2.1: コンソールを使用してカスタマーマネージドポリシーをグループに追加する

1. 前の手順で開いた IAM コンソールを使って、サービスのナビゲーションペイン内で、[グループ] を選択します。
2. グループの名前を選択します。
3. [許可] タブの [マネージドポリシー] には、[ポリシーのアタッチ] を選択します。
4. ポリシー名のリストで、グループにアタッチする各カスタマー管理ポリシーの横にあるボックスを選択します。特定のポリシー名がリストに表示されない場合は、[Filter] (フィルター) ボックスにポリシー名を入力して表示させます。
5. [ポリシーをアタッチ] を選択します。

ステップ 2.2: を使用してカスタマー管理ポリシーをグループに追加する AWS CLI

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できますAWS CLI。

ポリシーのグループの名前と Amazon リソースネーム (ARN) を指定して、IAM attach-group-policy コマンドを実行します。

```
aws iam attach-group-policy --group-name MyGroup --policy-arn
arn:aws:iam::123456789012:policy/MyPolicy
```

前述のコマンドで、MyGroup をグループの名前に置き換えます。を AWS アカウント ID 123456789012に置き換えます。MyPolicy はカスタマー管理ポリシーの名前に置き換えます。

次の手順

タスク	次のトピックを参照
AWS Cloud9 開発環境を作成し、IDE AWS Cloud9 を使用して新しい環境でコードを操作します。	環境を作成する
IDE AWS Cloud9 の使用方法を説明します。	開始方法: ベーシックチュートリアル および IDE を操作する
リアルタイムでチャットサポートを使用し、他のユーザーを招待して一緒に新しい環境を使用します。	共有環境を使用する

を使用するチームのカスタマー管理ポリシーの例 AWS Cloud9

以下に、グループのユーザーが AWS アカウント内に作成できる環境を制限するために使用できるポリシーの例をいくつか示します。

- [グループ内のユーザーが環境を作成できないようにする](#)
- [グループ内のユーザーが EC2 環境を作成できないようにする](#)
- [グループ内のユーザーに特定の Amazon EC2 インスタンスタイプでのみ EC2 環境を作成することを許可する](#)
- [グループ内のユーザーがリージョンごとに 1 つの EC2 環境のみを作成できるようにする AWS](#)

グループ内のユーザーが環境を作成できないようにする

次のカスタマー管理ポリシーは、AWS Cloud9 ユーザーグループにアタッチすると、それらのユーザーが環境を作成できないようにします AWS アカウント。これは、の管理者ユーザーが環境の作成 AWS アカウント を管理する場合に便利です。それ以外の場合、ユーザーグループの AWS Cloud9 ユーザーはこれを行います。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cloud9:CreateEnvironmentEC2",
        "cloud9:CreateEnvironmentSSH"
      ],
      "Resource": "*"
    }
  ]
}
```

前述のカスタマー管理ポリシー "Effect": "Allow" は、AWS Cloud9 ユーザーグループに既にアタッチされている AWS Cloud9 ユーザー管理ポリシー "Resource": "*" の "Action": "cloud9:CreateEnvironmentEC2" と "cloud9:CreateEnvironmentSSH" の を明示的に上書きします。

グループ内のユーザーが EC2 環境を作成できないようにする

次のカスタマー管理ポリシーは、AWS Cloud9 ユーザーグループにアタッチされると、それらのユーザーが EC2 環境を作成できないようにします AWS アカウント。これは、の管理者ユーザーが EC2 環境の作成 AWS アカウント を管理できるようにする場合に便利です。それ以外の場合、ユーザーグループの AWS Cloud9 ユーザーはこれを行います。これは、そのグループのユーザーが SSH 環境を作成できないようにするポリシーがアタッチ済みでないことを前提としています。アタッチ済みの場合、それらのユーザーは環境を作成できません。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "cloud9:CreateEnvironmentEC2",
      "Resource": "*"
    }
  ]
}
```

```
    }
  ]
}
```

前述のカスタマー管理ポリシーは、AWS Cloud9 ユーザーグループに既にアタッチされているAWSCloud9User管理ポリシー"Resource": "*"の "Action": "cloud9:CreateEnvironmentEC2" "Effect": "Allow"の を明示的に上書きします。

グループ内のユーザーに特定の Amazon EC2 インスタンスタイプでのみ EC2 環境を作成することを許可する

次のカスタマー管理ポリシーを AWS Cloud9 ユーザーグループにアタッチすると、ユーザーグループのユーザーは、 t2 から始まるインスタンスタイプのみを使用する EC2 環境を作成できます AWS アカウント。このポリシーは、そのグループのユーザーが EC2 環境を作成できないようにするポリシーをアタッチ済みでないことも前提としています。アタッチ済みの場合、それらのユーザーは EC2 環境を作成できません。

次のポリシーの "t2.*" は、別のインスタンスクラス (例: "m4.*") に置き換えることができます。または、複数のインスタンスクラスやインスタンスタイプ (例: ["t2.*", "m4.*"] または ["t2.micro", "m4.large"]) に制限できます。

AWS Cloud9 ユーザーグループの場合は、AWSCloud9User管理ポリシーをグループからデタッチします。次に、その場所に、次のカスタマー管理ポリシーを追加します。AWSCloud9User マネージドポリシーをデタッチしないと、次のカスタマー管理ポリシーは効果がありません。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloud9:CreateEnvironmentSSH",
        "cloud9:GetUserPublicKey",
        "cloud9:UpdateUserSettings",
        "cloud9:GetUserSettings",
        "iam:GetUser",
        "iam:ListUsers",
        "ec2:DescribeVpcs",
```

```

        "ec2:DescribeSubnets"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "cloud9:CreateEnvironmentEC2",
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "cloud9:InstanceType": "t2.*"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "cloud9:DescribeEnvironmentMemberships"
    ],
    "Resource": [
        "*"
    ],
    "Condition": {
        "Null": {
            "cloud9:UserArn": "true",
            "cloud9:EnvironmentId": "true"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "cloud9.amazonaws.com"
        }
    }
}
]
}

```

前述のカスタマー管理ポリシーによって、そのユーザーの環境の作成も許可されます。これらのユーザーが SSH 環境を作成できないようにするには、前述のカスタマー管理ポリシーから "cloud9:CreateEnvironmentSSH", を削除します。

グループ内のユーザーがそれぞれに 1 つの EC2 環境のみを作成することを許可する AWS リージョン

次のカスタマー管理ポリシーを AWS Cloud9 ユーザーグループにアタッチすると、各ユーザーは AWS リージョン AWS Cloud9 で使用できる各に最大 1 つの EC2 環境を作成できます。これは、環境の名前をその AWS リージョン内で 1 つの特定の名前に制限することで行われます。この例では、環境は my-demo-environment に制限されています。

Note

AWS Cloud9 では、環境の作成を特定の に制限することはできません AWS リージョン 。AWS Cloud9 また、 では、作成できる環境の総数の制限も有効になりません。唯一の例外は、公開されている [サービスの制限](#) です。

AWS Cloud9 ユーザーグループの場合、グループからAWSCloud9User管理ポリシーをデタッチし、代わりに次のカスタマー管理ポリシーを追加します。AWSCloud9User マネージドポリシーをデタッチしないと、次のカスタマー管理ポリシーは効果がありません。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloud9:CreateEnvironmentSSH",
        "cloud9:GetUserPublicKey",
        "cloud9:UpdateUserSettings",
        "cloud9:GetUserSettings",
        "iam:GetUser",
        "iam:ListUsers",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets"
      ]
    }
  ],
}
```



```
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "cloud9:CreateEnvironmentEC2"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "cloud9:EnvironmentName": "my-demo-environment"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "cloud9:DescribeEnvironmentMemberships"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "Null": {
        "cloud9:UserArn": "true",
        "cloud9:EnvironmentId": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": "cloud9.amazonaws.com"
      }
    }
  }
]
```

前述のカスタマー管理ポリシーによって、そのユーザーの SSH 環境の作成が許可されます。これらのユーザーが SSH 環境を作成できないようにするには、前述のカスタマー管理ポリシーから "cloud9:CreateEnvironmentSSH", を削除します。

その他の例については、「[お客様のマネージドポリシーの例](#)」を参照してください。

の開始方法 AWS Cloud9

の使用を開始するには、このチュートリアルを使用します AWS Cloud9。コンソールまたは [AWS Command Line Interface \(AWS CLI\)](#) を使用して IDE AWS Cloud9 を使用できます AWS Cloud9。このチュートリアルでは、AWS Cloud9 開発環境をセットアップし、IDE AWS Cloud9 を使用して最初のアプリケーションをコーディング、実行、デバッグする方法について説明します。詳細については AWS Cloud9、[「とは AWS Cloud9」](#) を参照してください。

AWS Cloud9 IDE の詳細については、「」を参照してください [IDE AWS Cloud9 のツアー](#)。

チュートリアルの所要時間は約 1 時間です。

Warning

このチュートリアルを完了すると、AWS アカウントに料金が発生する可能性があります。これには、Amazon EC2 の使用に伴う料金が含まれる場合があります。詳細については、[「Amazon EC2 の料金」](#) を参照してください。

前提条件

このチュートリアルを正常に完了するには、まず「[セットアップ AWS Cloud9](#)」のステップを完了する必要があります。

ステップ 1: 環境を作成する

このステップでは、AWS Cloud9 コンソールまたは AWS CLI を使用して AWS Cloud9 開発環境を作成できます。

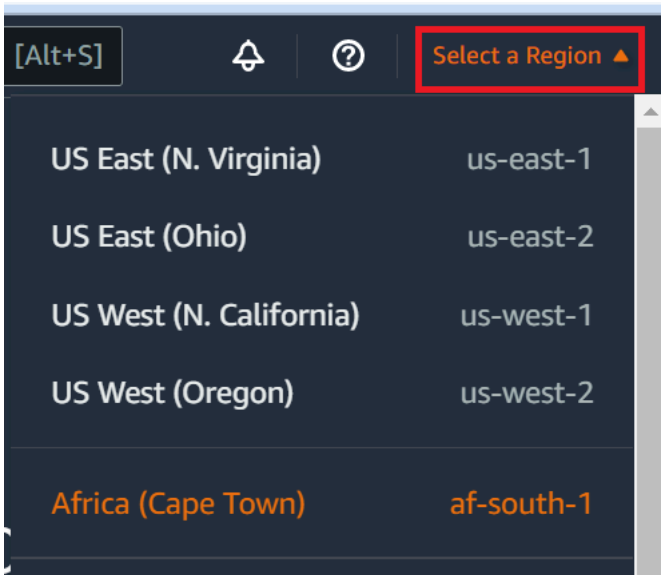
Note

このチュートリアルで使用する環境を作成済みである場合は、その環境を開き、[「ステップ 2: IDE のベーシック演習」](#)に進んでください。

では AWS Cloud9、開発環境または 環境は、開発プロジェクトのファイルを保存し、アプリケーションを開発するためのツールを実行する場所です。このチュートリアルでは、EC2 環境を作成し、この環境でファイルとツールを操作します。

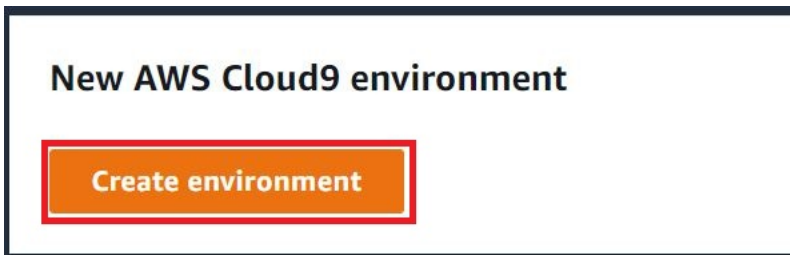
Create an EC2 Environment with the console

1. AWS Cloud9 コンソールにサインインします。
 - を使用するののお客様だけであるか、単一の IAM ユーザー AWS アカウント である場合は AWS アカウント、<https://console.aws.amazon.com/cloud9/> にアクセスしてください。
 - 組織が を使用している場合は AWS IAM アイデンティティセンター、AWS アカウント 管理者にサインイン手順を依頼してください。
 - 教室内の学生である場合は、インストラクターにサインインの手順をお問い合わせください。
2. AWS Cloud9 コンソールにサインインしたら、上部のナビゲーションバーで AWS リージョン を選択して環境を作成します。利用可能な のリストについては AWS リージョン、[AWS Cloud9](#) の「」を参照してくださいAWS 全般のリファレンス。



3. 表示されている場所の 1 つで、大きな [環境を作成する] ボタンを選択します。

まだ AWS Cloud9 環境がない場合は、ウェルカムページにボタンが表示されます。



環境がすでにある場合 AWS Cloud9、ボタンは次のように表示されます。



4. [Create environment] (環境の作成) ページで、[Name] (名前) に環境の名前を入力します。
5. [Description (説明)] に、環境に関する説明を入力します。このチュートリアルでは、This environment is for the AWS Cloud9 tutorial. を使用します。
6. [Environment type] (環境タイプ) で、[New EC2 instance] (新しい EC2 インスタンス) を選択して Amazon EC2 環境を作成します。
 - [New EC2 instance] (新しい EC2 インスタンス) — AWS Cloud9 から SSH 経由で直接接続できる新しい Amazon EC2 インスタンスを起動します。新しい Amazon EC2 インスタンスは、Systems Manager を使用して操作できます。詳細については、「[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)」を参照してください。
 - [Existing compute] (既存のコンピューティング) — SSH ログインの詳細を必要とする既存の Amazon EC2 インスタンスを起動します。Amazon EC2 インスタンスにはインバウンドセキュリティグループルールが必要です。
 - [Existing compute] (既存のコンピューティング) オプションを選択すると、サービスロールが自動的に作成されます。サービスロールの名前は、セットアップ画面の下部にある注記で確認できます。

 Note

自動シャットダウンは、既存のコンピューティングを使用する Amazon EC2 インスタンスを使用して作成された AWS Cloud9 環境では利用できません。

 Warning

環境の Amazon EC2 インスタンスを作成すると、Amazon EC2 の AWS アカウントに対する課金が発生する場合があります。Systems Manager を使用して EC2 インスタンスへの接続を管理する場合、追加コストはかかりません。

7. [Instance type] (インスタンスタイプ) の [New EC2 instance] (新しい EC2 インスタンス) パネルでは、デフォルトの選択のままにします。このオプションでは、RAM と vCPU が少ない可能性があります。ただし、このチュートリアルでは、このメモリ量で十分です。

 Warning

RAM と vCPUs の数が多いインスタンスタイプを選択すると、Amazon EC2 AWS アカウントの に追加料金が発生する可能性があります。

- プラットフォームで、必要な Amazon EC2 インスタンスのタイプを選択します。Amazon Linux 2023、Amazon Linux 2、または Ubuntu 22.04 LTS。はインスタンス AWS Cloud9 を作成し、環境をそのインスタンスに接続します。

 Important

EC2 環境には、[Amazon Linux 2023] オプションを選択することをお勧めします。Amazon Linux 2023 AMI は、安全で安定した、高パフォーマンスのランタイム環境を提供します。さらに、2024 年までの長期サポートも含まれています。詳細については、[AL2023 のページ](#)を参照してください。

- [Timeout] (タイムアウト) の期間を選択します。このオプションは、自動休止状態になるまでの AWS Cloud9 の非アクティブ時間を決定します。環境の IDE に接続されているすべてのウェブブラウザインスタンスが閉じられると、は指定された時間を AWS Cloud9 待ってから、環境の Amazon EC2 インスタンスをシャットダウンします。

 Warning

期間を長くするほど、AWS アカウントに対する課金が増える可能性があります。

- [Network settings] (ネットワーク設定) パネルで、環境にアクセスする方法を、次の 2 つのオプションから選択します。
 - AWS System Manager (SSM) – このメソッドは、インバウンドポートを開かずに SSM を使用して環境にアクセスします。
 - Secure Shell (SSH) — この方法では、SSH を使用して環境にアクセスします。受信ポートを開く必要があります。
- VPC 設定を選択すると、環境の Amazon Virtual Private Cloud とサブネットが表示されます。は Amazon Virtual Private Cloud (Amazon VPC) AWS Cloud9 を使用して、新しく作成された Amazon EC2 インスタンスと通信します。このチュートリアルでは、事前に選択されたデフォルト設定を変更しないことをお勧めします。デフォルト設定では、AWS Cloud9

は、新しい環境と同じ AWS アカウント およびリージョンにある単一のサブネットを持つデフォルトの VPC を自動的に使用しようとしています。

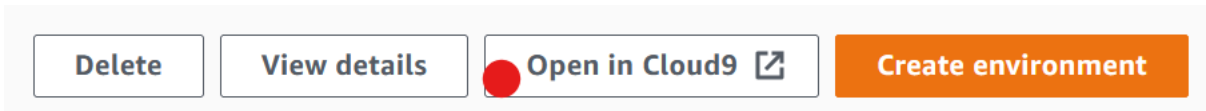
Amazon VPC 選択肢の詳細については、「[コンソールを使用した EC2 環境の作成](#)」および「[AWS Cloud9 開発環境の VPC 設定](#)」を参照してください。

12. 最大 50 個のタグを追加します。タグごとにキーと値を指定します。これを行うには、[Add new tag] (新しいタグを追加) を選択します。タグはリソースタグとして AWS Cloud9 環境にアタッチされ、スタック、Amazon EC2 インスタンス CloudFormation、Amazon EC2 セキュリティグループの基盤となるリソースに伝達されます。タグの詳細については、「[IAM ユーザーガイド](#)」の[AWS「リソースタグを使用したアクセスの制御」](#)およびこのガイドの「[詳細情報](#)」を参照してください。

 Warning

これらのタグを作成後に更新した場合、変更は基になるリソースには反映されません。詳細については、[タグ](#)に関する詳細情報の「[基礎となるリソースへのタグ更新の伝播](#)」を参照してください。

13. [Create] (作成) を選択して環境を作成すると、ホームページにリダイレクトされます。アカウントが正常に作成されると、AWS Cloud9 コンソールの上部に緑色のフラッシュバーが表示されます。新しい環境を選択し、[Open in Cloud9] (Cloud9 で開く) を選択して IDE を起動できます。



アカウントの作成に失敗すると、AWS Cloud9 コンソールの上部に赤いフラッシュバーが表示されます。アカウントの作成に失敗する原因としては、ウェブブラウザ、AWS アクセス許可、インスタンス、または関連するネットワークの問題が考えられます。可能な解決方法は、「[AWS Cloud9 のトラブルシューティング](#)」セクションで参照できます。

 Note

AWS Cloud9 はIMDSv1 と IMDSv2 の両方をサポートしています。IMDSv1 よりもセキュリティレベルが強化されているため、IMDSv2 の導入をお勧めします。IMDSv2 の利点の詳細については、「[AWS セキュリティブログ](#)」を参照してください。IMDSv1 から IMDSv2 への移行の詳細については、「Linux インスタンス用

Amazon EC2 ユーザーガイド」の「[インスタンスメタデータサービスバージョン 2 の使用への移行](#)」を参照してください。

Note

環境でプロキシを使用してインターネットにアクセスしている場合は、依存関係をインストール AWS Cloud9 できるようにプロキシの詳細を に提供する必要があります。詳細については、「[依存関係をインストールできませんでした](#)」を参照してください。

Create an EC2 environment with the AWS CLI

1. まだインストールしていない場合は AWS CLI、 をインストールして設定します。これを行うには、AWS Command Line Interface ユーザーガイドで以下の項目を参照してください。

- [AWS コマンドラインインターフェイスのインストール](#)
- [クイック設定](#)

次のいずれかの認証情報 AWS CLI を使用して を設定できます。

- [のチームのセットアップ AWS Cloud9](#) で作成した IAM ユーザー。
- AWS アカウント全体の複数のユーザーの AWS Cloud9 リソースを定期的に操作する場合、アカウントの IAM 管理者。を IAM 管理者 AWS CLI として設定できない場合は、アカウント管理者に AWS 確認してください。詳細については、IAM ユーザーガイドの「[最初の IAM 管理者ユーザーおよびグループの作成](#)」を参照してください。
- AWS アカウントのルートユーザー。ただし、常に自分の AWS アカウントを使用する唯一のユーザーであり、他のユーザーと環境を共有する必要はありません。このオプションは AWS セキュリティのベストプラクティスではないため、お勧めしません。詳細については、「Amazon Web Services 全般のリファレンス」の「[AWS アカウントのアクセスキーの作成、無効化、削除](#)」を参照してください。
- その他のオプションについては、AWS アカウント管理者またはクラスルームインストラクターにお問い合わせください。

2. 次の AWS Cloud9 コマンドで、`--region`と の値を指定します`--subnet-id`。次に、そのコマンドを実行し、「`environmentId`」値を記録します。この値は、後でクリーンアップに必要になります。

```
aws cloud9 create-environment-ec2 --name my-demo-environment --description "This environment is for the AWS Cloud9 tutorial." --instance-type t2.micro --image-id resolve:ssm:/aws/service/cloud9/amis/amazonlinux-2-x86_64 --region MY-REGION --connection-type CONNECT_SSM --subnet-id subnet-12a3456b
```

上記のコマンドでは:

- `--name` は、環境の名前を表します。このチュートリアルでは、`my-demo-environment` という名前を使用します。
- `--description` は、環境の説明 (オプション) を表します。
- `--instance-type` は、AWS Cloud9 で起動して新しい環境に接続する Amazon EC2 インスタンスのタイプを表します。この例では `t2.micro` を指定します。これは比較的小さい RAM および vCPU を使用しますが、このチュートリアルでは十分です。RAM と vCPUs の数が多いインスタンスタイプを指定すると、Amazon EC2 の AWS アカウントに追加料金が発生する可能性があります。使用可能なインスタンスタイプのリストについては、AWS Cloud9 コンソールの環境作成ウィザードを参照してください。
- EC2 インスタンスの作成に使用される Amazon マシンイメージ (AMI) の識別子を `--image-id` は指定します。インスタンスの AMI を選択するには、有効な AMI エイリアスまたは有効な AWS Systems Manager (SSM) パスを指定する必要があります。上記の例では、Amazon Linux 2 AMI の SSM パスが指定されています。

詳細については、AWS CLI コマンドリファレンスの「[create-environment-ec2](#)」を参照してください。

- `--region` は、環境を作成する AWS Cloud9 の AWS リージョンの ID を表します。使用可能な AWS リージョンのリストについては、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。
- `--connection-type CONNECT_SSM` は、Systems Manager を介して Amazon EC2 インスタンス AWS Cloud9 に接続する を指定します。このオプションにより、インスタンスへのインバウンドトラフィックが許可されなくなります。詳細については、「[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)」を参照してください。

Note

このオプションを使用するときは、AWS Cloud9SSMAccessRole サービスロールと AWS Cloud9SSMInstanceProfile を作成する必要があります (まだ作成されていない場合)。詳細については、「[を使用した Systems Manager のインスタンスプロファイルの管理 AWS CLI](#)」を参照してください。

- `--subnet-id` は、AWS Cloud9 使用するサブネットを表します。subnet-12a3456b は Amazon Virtual Private Cloud (VPC) のサブネットの ID に置き換えます。これは AWS Cloud9 と互換性がある必要があります。詳細については、「[AWS Cloud9 開発環境の VPC 設定](#)」の [VPC と他の VPC リソースを作成する](#) を参照してください。
 - AWS Cloud9 は、環境の IDE に接続されているすべてのウェブブラウザインスタンスが閉じられた後、環境の Amazon EC2 インスタンスをシャットダウンします。この期間を設定するには、`--automatic-stop-time-minutes` と分数を追加します。期間が短いと、AWS アカウントへの課金が少なくなる可能性があります。同様に、期間が長いと課金が高くなる可能性があります。
 - デフォルトでは、このコマンドを呼び出すエンティティが環境を所有します。これを変更するには、`--owner-id` と、所有エンティティの Amazon リソースネーム (ARN) を追加します。
3. このコマンドを正常に実行したら、新しく作成した環境の AWS Cloud9 IDE を開きます。これを行うには、「[で環境を開く AWS Cloud9](#)」を参照してください。次に、このトピックに戻り、に進み[ステップ 2: IDE のベーシック演習](#)、IDE AWS Cloud9 を使用して新しい環境を操作する方法について説明します。

環境を開いても 5 分以上経過しても IDE が表示され AWS Cloud9 ない場合は、ウェブブラウザ、AWS アクセス許可、インスタンス、または関連する VPC に問題がある可能性があります。可能な修正については、「[環境を開くことができない](#)」を参照してください。

ステップ 2: IDE のベーシック演習

このチュートリアルでは、IDE AWS Cloud9 を使用してアプリケーションを作成およびテストする方法をいくつか紹介します。

- [editor (エディタ)] ウィンドウを使用して、コードを作成および編集できます。

- [terminal (ターミナル)] ウィンドウまたは [Run Configuration (実行設定)] ウィンドウを使用して、デバッグせずにコードを実行できます。
- [Debugger (デバッガー)] ウィンドウを使用してコードをデバッグできます。

JavaScript および Node.js エンジンを使用して、これらの 3 つのタスクを実行します。他のプログラミング言語を使用する手順については、「[のチュートリアル AWS Cloud9](#)」を参照してください。

環境の準備をする

JavaScript コードの実行とデバッグに必要なツールのほとんどはインストール済みです。ただし、このチュートリアルではもう 1 つの Node.js パッケージが必要です。これを次のようにインストールします。

1. AWS Cloud9 IDE の上部にあるメニューバーで、Window、New Terminal を選択するか、既存のターミナルウィンドウを使用します。
2. ターミナルウィンドウ (IDE の下部にあるタブの 1 つ) では、以下のように入力します。

```
npm install readline-sync
```

結果が以下になることを確認します。npm WARN メッセージが表示されても、無視してかまいません。

```
+ readline-sync@1.4.10
added 1 package from 1 contributor and audited 5 packages in 0.565s
found 0 vulnerabilities
```

コードを記述する

コードを記述することから始めます。

1. メニューバーで [File (ファイル)]、[New File (新規ファイル)] の順に選択します。
2. 以下の JavaScript を新しいファイルに追加します。

```
var readline = require('readline-sync');
var i = 10;
var input;
```

```
console.log("Hello Cloud9!");
console.log("i is " + i);

do {
    input = readline.question("Enter a number (or 'q' to quit): ");
    if (input === 'q') {
        console.log('OK, exiting.')
    }
    else{
        i += Number(input);
        console.log("i is now " + i);
    }
} while (input !== 'q');

console.log("Goodbye!");
```

3. [File (ファイル)]、[Save (保存)] の順に選択し、ファイルを `hello-cloud9.js` として保存します。

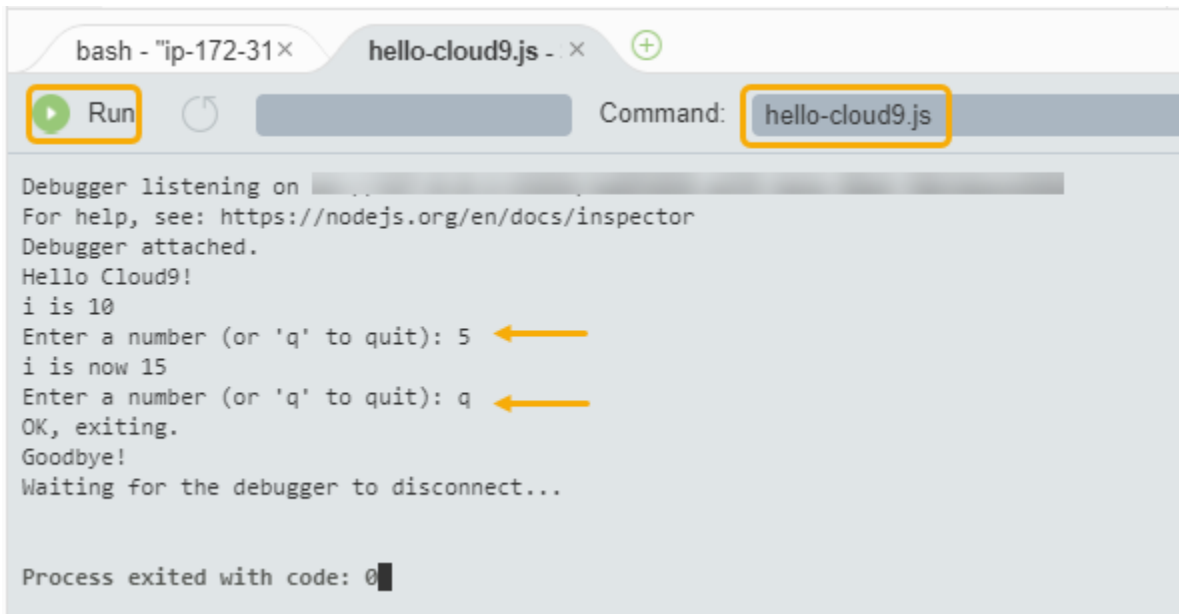
コードを実行する

次に、コードを実行できます。

使用しているプログラミング言語によっては、複数の方法でコードを実行できる場合があります。このチュートリアルで使用している JavaScript は、ターミナルウィンドウまたは [Run Configuration (実行設定)] ウィンドウを使用して実行できます。

[Run Configuration (実行設定)] ウィンドウを使用してコードを実行するには

1. メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
2. 新しい [設定の実行] ウィンドウ (IDE の下部にあるタブの 1 つ) で、[コマンド] フィールドに「`hello-cloud9.js`」と入力し、[実行] を選択します。
3. [Run Configuration] (実行設定) プロンプトがアクティブであることを確認してから、プロンプトで番号を入力してアプリケーションを操作します。
4. [Run Configuration (実行設定)] ウィンドウにコードの出力が表示されます。次のように表示されます。



The screenshot shows the AWS Cloud9 IDE interface. At the top, there are two tabs: 'bash - "ip-172-31' and 'hello-cloud9.js - x'. Below the tabs, there is a 'Run' button (a green play icon) and a 'Command:' field containing 'hello-cloud9.js'. The terminal output shows the following text:

```
Debugger listening on [redacted]
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Hello Cloud9!
i is 10
Enter a number (or 'q' to quit): 5
i is now 15
Enter a number (or 'q' to quit): q
OK, exiting.
Goodbye!
Waiting for the debugger to disconnect...

Process exited with code: 0
```

Yellow arrows point to the input '5' and 'q' in the terminal output.

ターミナルウィンドウを使用してコードを実行するには

1. 先ほど使用したターミナルウィンドウに移動します (または新しいターミナルウィンドウを開きます)。
2. ターミナルウィンドウのプロンプトで「ls」と入力し、コードファイルがファイルのリストにあることを確認します。
3. プロンプトで「node hello-cloud9.js」と入力して、アプリケーションを起動します。
4. プロンプトで番号を入力して、アプリケーションを操作します。
5. ターミナルウィンドウにコードの出力が表示されます。次のように表示されます。



The screenshot shows the AWS Cloud9 IDE interface. At the top, there are two tabs: 'node - "ip-172-31' and 'hello-cloud9.js - !x'. Below the tabs, there is a terminal window with the following text:

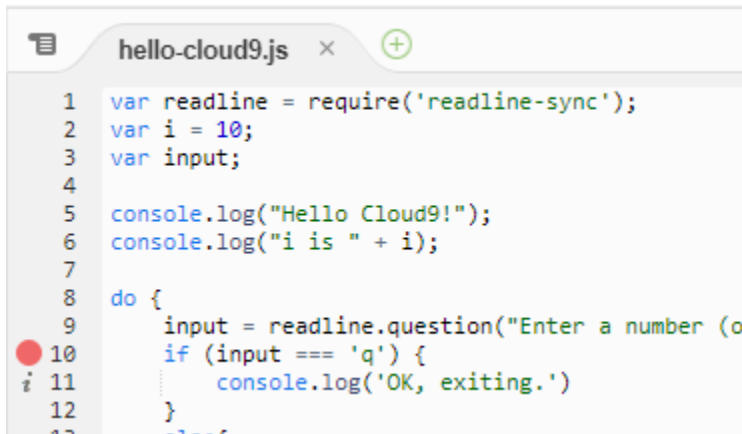
```
Admin:~/environment $ node hello-cloud9.js
Hello Cloud9!
i is 10
Enter a number (or 'q' to quit): 5
i is now 15
Enter a number (or 'q' to quit): q
OK, exiting.
Goodbye!
Admin:~/environment $
```

Yellow arrows point to the input '5' and 'q' in the terminal output.

コードをデバッグする

最後に、 [Debugger (デバッガー)] ウィンドウを使用してコードをデバッグできます。

1. 行 10 の横のマージンを選択して、行 10 のコード (if (input === 'q')) にブレークポイントを追加します。以下のように、その行番号の横に赤い円が表示されます。

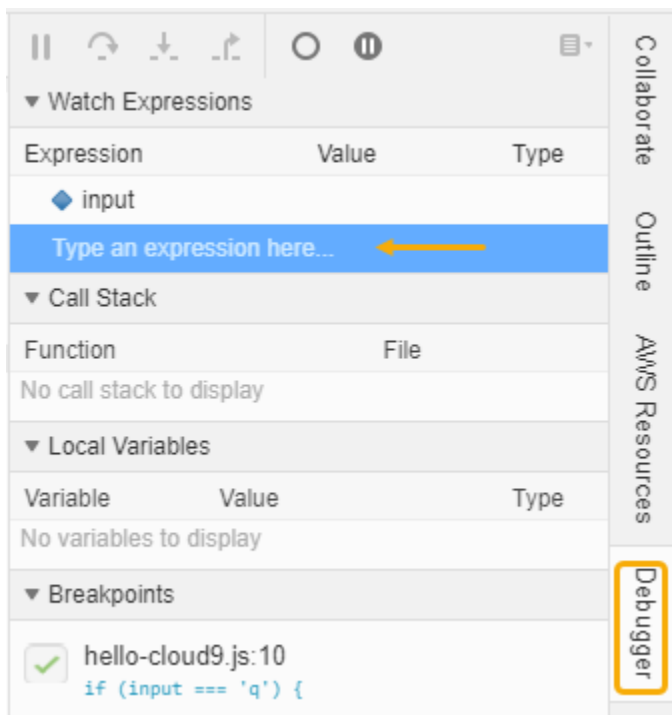


```

1  var readline = require('readline-sync');
2  var i = 10;
3  var input;
4
5  console.log("Hello Cloud9!");
6  console.log("i is " + i);
7
8  do {
9      input = readline.question("Enter a number (0
10     if (input === 'q') {
11         console.log('OK, exiting.')
12     }
13 } while (input !== 'q');
```

2. IDE の右側で [Debugger] ボタンを選択して、Debugger ウィンドウを開きます。または、メニューバーで [Window (ウィンドウ)]、[Debugger (デバッガー)] の順に選択します。

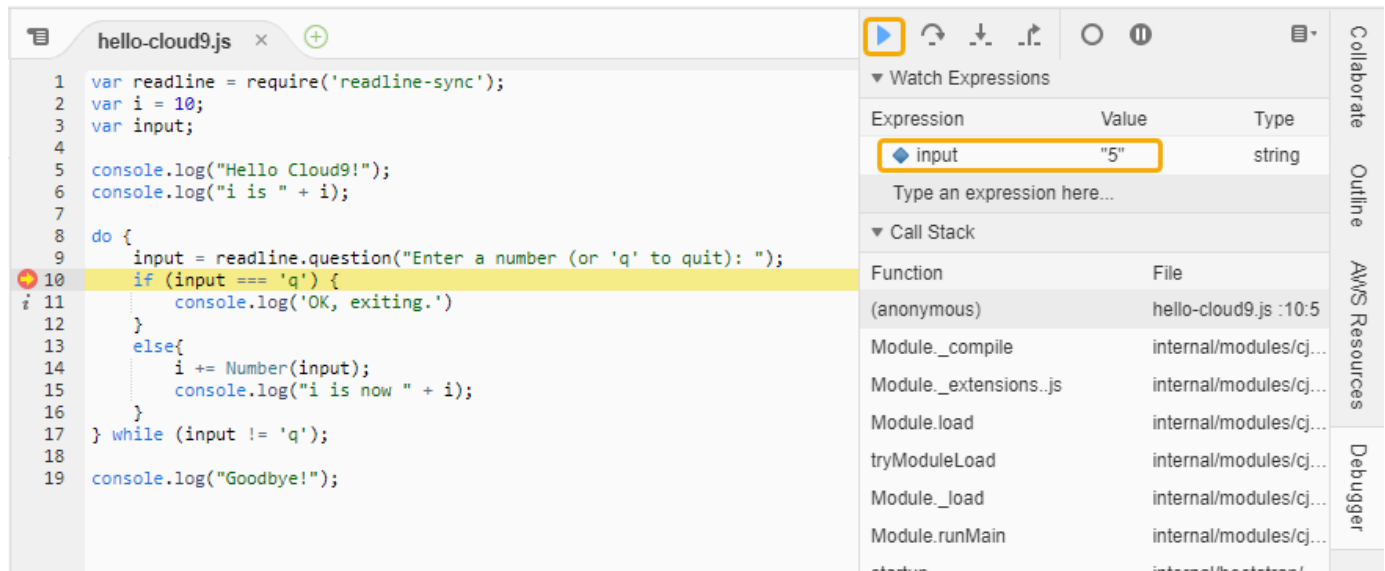
次に、[Debugger] (デバッガー) ウィンドウの [Watch Expressions] (ウォッチ式) セクションで [Type an expression here] (ここに式を入力) を選択して、input 変数を監視します。



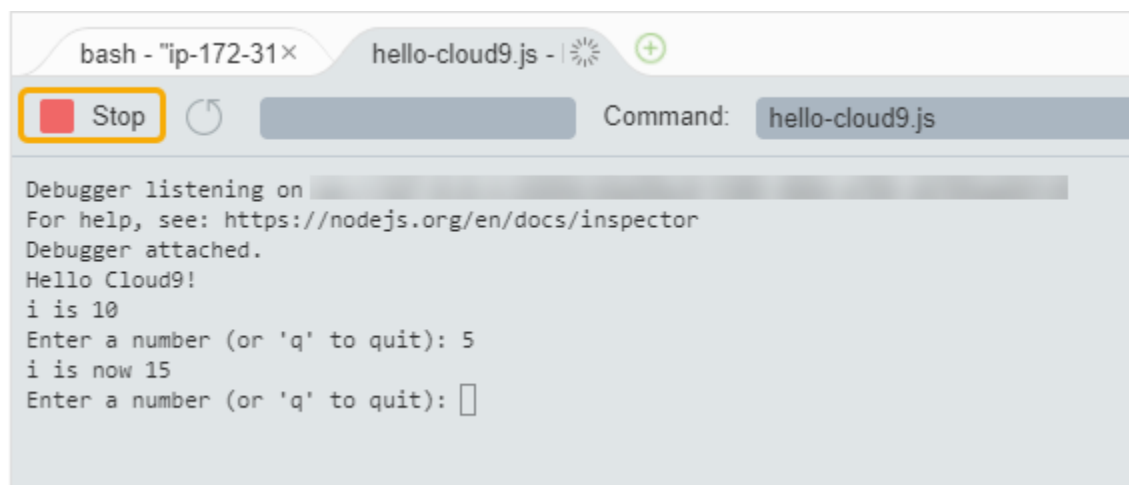
3. コードを実行するために先ほど使用した [Run Configuration (実行設定)] ウィンドウに移動します。 [Run(実行)] を選択します。

または、新しい [Run Configuration] (実行設定) ウィンドウを開いて、コードの実行を開始することもできます。そのためには、メニューバーから [Run] (実行)、[Run with] (環境を指定して実行)、[Node.js] の順に選択します。

4. [Run Configuration (実行設定)] プロンプトで番号を入力し、コードが行 10 で一時停止することを確認します。[Debugger] (デバッガー) ウィンドウに、[Watch Expressions] (ウォッチ式) に入力した値が表示されます。



5. [Debugger] (デバッガー) ウィンドウで、[Resume] (再開) を選択します。これは、前のスクリーンショットで強調表示されている青い矢印アイコンです。
6. [設定の実行] ウィンドウで [停止] を選択し、デバッガーを停止します。



ステップ 3: クリーンアップする

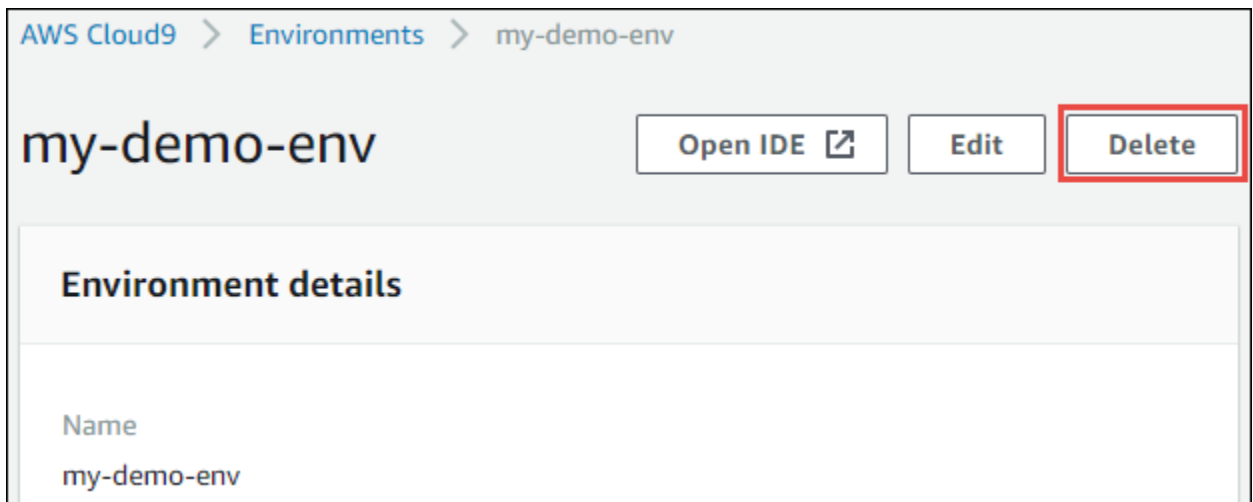
このチュートリアル AWS アカウント に関連する への継続的な課金を防ぐには、環境を削除します。

Warning

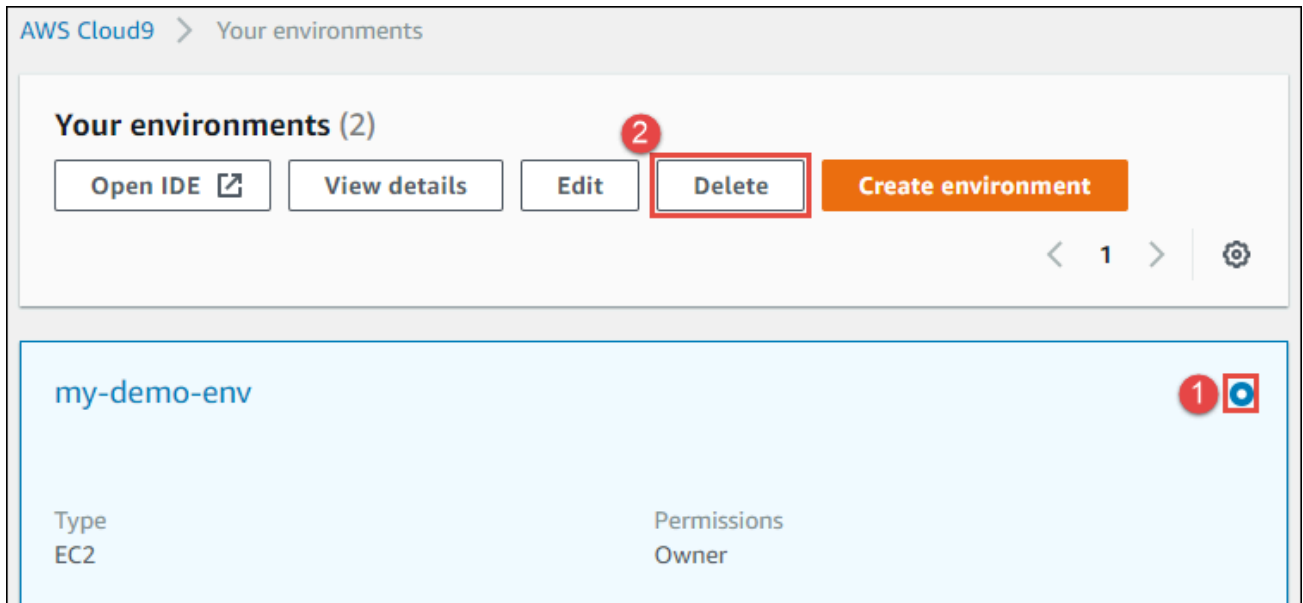
削除した環境を復元することはできません。

Delete the Environment by using the AWS Cloud9 console

1. ダッシュボードを開くには、IDE のメニューバーで、[AWS Cloud9]、[Go To Your Dashboard (ダッシュボードを開く)] を選択します。
2. 次のいずれかを行います。
 - [my-demo-environment] カード内のタイトルを選択し、もう一度 [Delete (削除)] を選択します。



- [my-demo-environment] を選択し、次に [Delete (削除)] を選択します。



3. [Delete (削除)] ダイアログボックスに「Delete」と入力し、[Delete (削除)] を選択します。削除オペレーションには数分かかります。

Note

このチュートリアルに厳密に従っていれば、環境は EC2 環境であり、AWS Cloud9 は、その環境に接続されていた Amazon EC2 インスタンスも終了します。

ただし、チュートリアルに従う代わりに SSH 環境を使用し、その環境が Amazon EC2 インスタンスに接続されている場合、そのインスタンス AWS Cloud9 は終了しません。後でそのインスタンスを終了しない場合、そのインスタンスに関連する Amazon EC2 の料金が引き続き発生する AWS アカウント 可能性があります。

Delete the Environment with the AWS CLI

1. コマンドを実行し AWS Cloud9 delete-environment、削除する環境の ID を指定します。

```
aws cloud9 delete-environment --region MY-REGION --environment-id
12a34567b8cd9012345ef67abcd890e1
```

前述のコマンドで、を環境が作成された AWS リージョンMY-REGIONに、を削除する環境の ID 12a34567b8cd9012345ef67abcd890e1 に置き換えます。

環境の作成時に ID を保存しなかった場合は、AWS Cloud9 コンソールを使用して ID を見つけることができます。コンソールで環境の名前を選択し、[Environment ARN (環境 ARN)] の最後の部分を見つけます。

2. このチュートリアル用に Amazon VPC を作成し、不要になった場合は、Amazon VPC コンソールを使用して VPC を削除します (<https://console.aws.amazon.com/vpc>)。

関連情報

[AWS Cloud9 コンソールの使用開始](#)に関する追加情報を以下に示します。

- EC2 環境を作成する場合、デフォルトでは環境にサンプルコードは含まれていません。サンプルコードを使用して環境を作成するには、次のトピックを参照してください。
- [IDE AWS Cloud9 での Amazon Lightsail インスタンスの操作](#)
- AWS Cloud9 開発環境の作成中に、Amazon EC2 instance. AWS Cloud9 created the instance and then connected the environment to it AWS Cloud9 を作成するように指示されます。別の方法として、既存のクラウドコンピューティングインスタンスや独自のサーバー (SSH 環境と呼ばれます) を使用することもできます。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。

次のステップ (オプション)

以下のトピックの一部またはすべてを調べて、引き続き理解を深めてください AWS Cloud9。

タスク	次のトピックを参照
環境で何ができるかに関する詳細はこちら。	での環境の使用 AWS Cloud9
他のコンピュータ言語を試します。	のチュートリアル AWS Cloud9
IDE AWS Cloud9 の詳細について説明します。	IDE を操作する の IDE AWS Cloud9 のツアー
テキストチャットサポートにより、リアルタイムで新しい環境を使用するように他のユーザーを招待します。	での共有環境の使用 AWS Cloud9
SSH 環境の作成 これらの環境では、AWS Cloud9 がユーザーに代わって作成した	での環境の作成 AWS Cloud9 および SSH 環境ホスト要件

タスク	次のトピックを参照
Amazon EC2 インスタンスではなく、ユーザーが作成したクラウドコンピューティングインスタンスまたはサーバーを使用します。	
AWS Toolkit を使用して、AWS Lambda 関数とサーバーレスアプリケーションでコードを作成、実行、デバッグします。	Toolkit を使用した AWS Lambda 関数の操作 AWS
AWS Cloud9 で を使用します Amazon Lightsail。	IDE AWS Cloud9 での Amazon Lightsail インスタンスの操作
AWS Cloud9 で を使用します AWS CodePipeline。	IDE AWS CodePipeline での AWS Cloud9 の使用
AWS Cloud9 AWS CLIは、 AWS CloudShell、 AWS クラウド開発キット (AWS CDK) AWS CodeCommit、 GitHub、 Amazon DynamoDB、 Node.js、 Python、 またはその他のプログラミング言語で使います。	のチュートリアル AWS Cloud9
AWS RoboMaker でインテリジェントなロボット工学アプリケーションのコードを使用します。	AWS RoboMaker デベロッパーガイド内にある AWS Cloud9を使って開発

コミュニティ AWS Cloud9 からのヘルプについては、「[AWS Cloud9 ディスカッションフォーラム](#)」を参照してください。(このフォーラムに参加すると、サインインが必要になる AWS 場合があります)。

AWS Cloud9 から直接サポートを受けるには AWS、 サポートページの[AWS サポート](#) オプションを参照してください。

での環境の使用 AWS Cloud9

開発環境は、プロジェクトのファイル AWS Cloud9 を保存し、アプリケーションを開発するためのツールを実行する の場所です。

AWS Cloud9 には、EC2 環境と SSH 環境の EC2 種類の開発環境が用意されています。これらの開発環境のキーとなる類似点と相違点を理解するには、[の SSH 環境と比較した EC2 環境 AWS Cloud9](#) を参照してください。

これらのトピックを 1 つ以上読んで AWS Cloud9 、 で 環境を操作する方法について説明します。

トピック

- [での環境の作成 AWS Cloud9](#)
- [を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)
- [で環境を開く AWS Cloud9](#)
- [の環境 AWS のサービス からの呼び出し AWS Cloud9](#)
- [での環境設定の変更 AWS Cloud9](#)
- [での共有環境の使用 AWS Cloud9](#)
- [Amazon EBS AWS Cloud9 ボリュームからの IDE の移動](#)
- [での環境の削除 AWS Cloud9](#)

での環境の作成 AWS Cloud9

AWS Cloud9 開発環境を作成するには、使用する計画に基づいて、提供されている手順のいずれかに従います AWS Cloud9。

何を選べばよいか明確でない場合、[EC2 環境を作成する](#) をお勧めします。

簡単なセットアップのために、EC2 環境を作成します。は、 に新しい Amazon EC2 インスタンス AWS Cloud9 を自動的に作成してセットアップします AWS アカウント。AWS Cloud9 また、はその新しいインスタンスを環境に自動的に接続します。

開発環境間の主な類似点と相違点については、「[の SSH 環境と比較した EC2 環境 AWS Cloud9](#)」を参照してください。

ソースコードの提供元	開発環境ホストの提供元	関連する手順
お客様	AWS Cloud9	EC2 環境の作成
お客様	お客様	SSH 環境の作成
Amazon Lightsail またはお客様	お客様 (Lightsail を使用)	IDE AWS Cloud9 での Amazon Lightsail インスタンスの操作
お客様 (AWS CodePipeline を使用)	AWS Cloud9 または	EC2 または SSH 環境、および IDE AWS CodePipeline での AWS Cloud9 の使用 を作成する
お客様 (AWS CodeCommit を使用)	AWS Cloud9 または	AWS CodeCommit の チュー トリアル AWS Cloud9
お客様 (GitHub を使用)	AWS Cloud9 または	EC2 または SSH 環境を作成し、 Git パネルインターフェイス を使用する

トピック

- [EC2 環境を作成する](#)
- [SSH 環境を作成する](#)

EC2 環境を作成する

この手順では、は EC2 環境と新しい Amazon EC2 インスタンス AWS Cloud9 を作成し、環境をこのインスタンスに接続します。は、必要に応じてインスタンスの起動、停止、再起動など、このインスタンスのライフサイクル AWS Cloud9 を管理します。この環境を削除した場合、AWS Cloud9 によってこのインスタンスが自動的に終了されます。

EC AWS Cloud9 EC2 開発環境は、[AWS Cloud9 コンソール](#)または[コード](#)を使用して作成できます。

 Note

この手順を完了すると、 に料金が発生する可能性があります AWS アカウント。これには、Amazon EC2 の使用に伴う料金も含まれる場合があります。詳細については、[Amazon EC2 の料金表](#)を参照してください。

 Warning

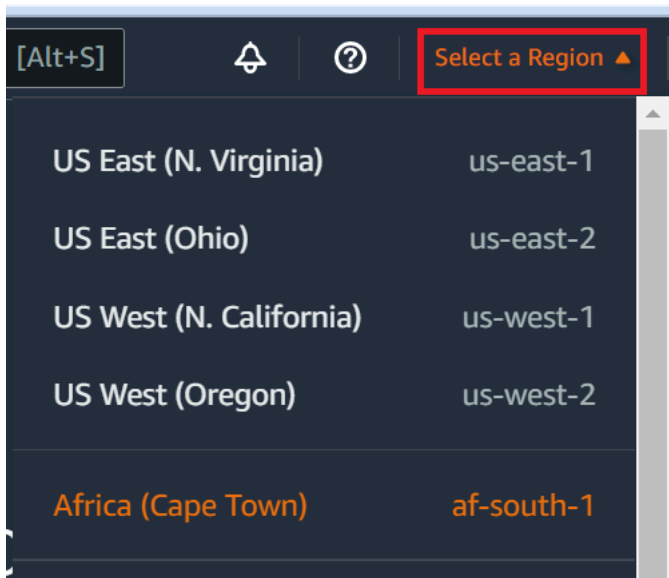
AWS Cloud9 および AWS Control Tower プロアクティブコントロール [CT.EC2.PR://www.8](#) との互換性の問題があります。このコントロールが有効な場合、AWS Cloud9で EC2 環境を作成することはできません。この問題の詳細については、[「トラブルシューティング AWS Cloud9」](#)を参照してください。

前提条件

AWS Cloud9 コンソールにサインインして環境を作成[セットアップ AWS Cloud9](#)できるように、「」のステップを完了します。

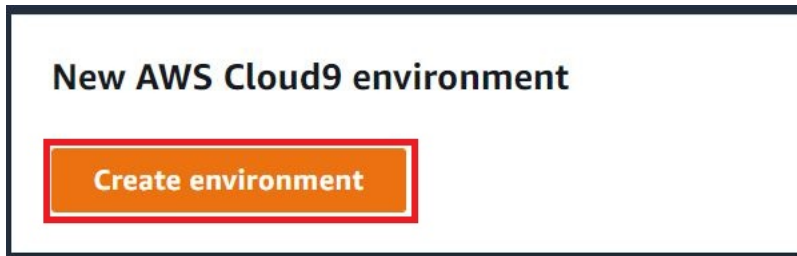
EC2 環境をコンソールで作成します。

1. AWS Cloud9 コンソールにサインインします。
 - ご自身が AWS アカウントを使用する唯一のユーザーである場合、または単一の AWS アカウントの IAM ユーザーである場合は、<https://console.aws.amazon.com/cloud9/> に移動します。
 - 組織で を使用している場合は AWS IAM アイデンティティセンター、AWS アカウント 管理者にサインイン手順を依頼してください。
 - 教室内の学生である場合は、インストラクターにサインインの手順をお問い合わせください。
2. AWS Cloud9 コンソールにサインインしたら、上部のナビゲーションバーで AWS リージョン を選択して環境を作成します。使用可能な のリストについては AWS リージョン、[AWS Cloud9](#) の「」を参照してくださいAWS 全般のリファレンス。



- 表示されている場所の1つで、大きな [環境を作成する] ボタンを選択します。

まだ AWS Cloud9 環境がない場合は、ウェルカムページにボタンが表示されます。



環境がすでにある場合 AWS Cloud9、ボタンは次のように表示されます。



- [Create environment] (環境の作成) ページで、[Name] (名前) に環境の名前を入力します。
- 環境の説明を追加するには、[Description] (説明) フィールドに入力します。
- [Environment type] (環境タイプ) で、[New EC2 instance] (新しい EC2 インスタンス) を選択して Amazon EC2 環境を作成します。
 - [New EC2 instance] (新しい EC2 インスタンス) — AWS Cloud9 から SSH 経由で直接接続できる新しい Amazon EC2 インスタンスを起動します。新しい Amazon EC2 インスタンスは、Systems Manager を使用して操作できます。詳細については、「[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)」を参照してください。

- [Existing compute] (既存のコンピューティング) — SSH ログインの詳細を必要とする既存の Amazon EC2 インスタンスを起動します。Amazon EC2 インスタンスにはインバウンドセキュリティグループルールが必要です。
- [Existing compute] (既存のコンピューティング) オプションを選択すると、サービスロールが自動的に作成されます。サービスロールの名前は、セットアップ画面の下部にある注記で確認できます。

 Note

自動シャットダウンは、既存のコンピューティングを使用して Amazon EC2 インスタンスを使用して作成された AWS Cloud9 環境では利用できません。

 Warning

環境の Amazon EC2 インスタンスを作成すると、Amazon EC2 の AWS アカウント に対する課金が発生する場合があります。Systems Manager を使用して EC2 インスタンスへの接続を管理する場合、追加コストはかかりません。

4. [Instance type] (インスタンスタイプ) で、実行するタスクの種類に必要なと思われる容量の RAM および vCPU を備えたインスタンスタイプを選択します。

 Warning

より多くの RAM と vCPUs を持つインスタンスタイプを選択すると、Amazon EC2 AWS アカウント の に追加料金が発生する可能性があります。どのインスタンスタイプがワークロードに適しているかについては、「[Amazon EC2 インスタンスタイプ](#)」ページを参照してください。

5. プラットフォームで、必要な Amazon EC2 インスタンスのタイプを選択します。Amazon Linux 2023、Amazon Linux 2、または Ubuntu 22.04 LTS。はインスタンス AWS Cloud9 を作成し、環境をそのインスタンスに接続します。

⚠ Important

EC2 環境には、[Amazon Linux 2023] オプションを選択することをお勧めします。Amazon Linux 2023 AMI は、安全で安定した、高パフォーマンスのランタイム環境を提供します。さらに、2024 年までの長期サポートも含まれています。詳細については、[AL2023 のページ](#)を参照してください。

6. [Timeout] (タイムアウト) の期間を選択します。このオプションは、自動休止状態になるまでの AWS Cloud9 の非アクティブ時間を決定します。環境の IDE に接続されているすべてのウェブブラウザインスタンスが閉じられると、は指定された時間を AWS Cloud9 待ってから、環境の Amazon EC2 インスタンスをシャットダウンします。

⚠ Warning

期間を長くするほど、AWS アカウントに対する課金が増える可能性があります。

7. [Network settings] (ネットワーク設定) パネルで、環境にアクセスする方法を、次の 2 つのオプションから選択します。
- AWS Systems Manager (SSM) – このメソッドは、インバウンドポートを開かずに SSM を使用して環境にアクセスします。
 - Secure Shell (SSH) — この方法では、SSH を使用して環境にアクセスします。受信ポートを開く必要があります。
8. VPC 設定を選択すると、環境の Amazon Virtual Private Cloud とサブネットが表示されます。は Amazon Virtual Private Cloud (Amazon VPC) AWS Cloud9 を使用して、新しく作成された Amazon EC2 インスタンスと通信します。このチュートリアルでは、事前に選択されたデフォルト設定を変更しないことをお勧めします。デフォルト設定では、AWS Cloud9 は、新しい環境と同じ AWS アカウント およびリージョンに単一のサブネットを持つデフォルトの VPC を使用しようとしています。Amazon VPC の設定方法に応じて、次のいずれかの指示セットを実行します。

何を選択すればいいのかわからない場合は、この手順の次のステップまでスキップすることをお勧めします。

過去のネットワーク設定 (アドバンスド) をスキップし、事前に選択したデフォルト設定のままにすると、は単一の subnet AWS Cloud9 でデフォルトの VPC を使用しようとしています。選択したインスタンスタイプに基づいてサブネット AWS Cloud9 を選択します。これらは、新しい環境と同じ AWS アカウントと AWS リージョンにあります。

 Important

環境タイプとして [Existing compute] (既存のコンピューティング) を選択した場合は、インスタンスをパブリックサブネットまたはプライベートサブネット内で起動できません。

- パブリックサブネット: インターネットゲートウェイをサブネットにアタッチして、インスタンスの SSM エージェントが Systems Manager と通信できるようにします。
- プライベートサブネット: NAT ゲートウェイを作成して、インスタンスがインターネットやその他の AWS のサービスと通信できるようにします。

現在、[AWS マネージド一時認証情報](#)を使用して、EC2 環境が IAM ユーザーなどの AWS エンティティ AWS のサービス に代わって にアクセスすることを許可することはできません。

サブネットの設定の詳細については、「[AWS Cloud9 開発環境の VPC 設定](#)」を参照してください。

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
いいえ	—	—	—	VPC が存在し ない場合は、作 成してください 。 新しい環境と 同じ AWS アカ ウント および リージョンに VPC を作成す るには、新し い VPC の作成 を選択し、画面 の指示に従いま す。詳細につい ては、「VPC と他の VPC リ ソースを作成す る」を参照して ください。 新しい環境 AWS アカウ ントとは異な る環境で VPC を作成するに は、「Amazon

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
				VPCs 」を参照 してください。
あり	あり	あり	あり	この手順の次 のステップにス キップします。 ネットワーク設 定 (アドバンス ド) をスキップ し、事前選択さ れたデフォルト 設定を変更しな い場合、AWS Cloud9 は、新 しい環境と同 じアカウントと リージョン内の 単一のサブネッ トでデフォルト の VPC を使用 しようとしま す。

AWS アカウントは Amazon VPC にアクセスできますか？	その VPC は新しい環境と同じ AWS アカウント およびリージョンにありますか？	VPC は AWS アカウントのデフォルト VPC ですか。	VPC 内に単一のサブネットがありますか。	以下の手順に従います
あり	あり	はい	いいえ	デフォルト VPC に複数のサブネットがある場合は、[Network settings (advanced) (ネットワーク設定 (高度))] を展開します。[Subnet] (サブネット) で、事前に選択されたデフォルト VPC で AWS Cloud9 が使用するサブネットを選択します。 デフォルト VPC にサブネットがない場合は、サブネットを作成します。これを行うには、[Create new subnet (新しいサブネットの作成)] を選択し、画面の

AWS アカウントは Amazon VPC にアクセスできますか？	その VPC は新しい環境と同じ AWS アカウント およびリージョンにありますか？	VPC は AWS アカウントのデフォルト VPC ですか。	VPC 内に単一のサブネットがありますか。	以下の手順に従います
				指示に従います。詳細については、「 のサブネットを作成する AWS Cloud9 」を参照してください。
はい	はい	なし	はい	[Network settings (ネットワーク設定)] を展開します。[Network (VPC) (ネットワーク (VPC))] で、AWS Cloud9 が使用する VPC を選択します。

AWS アカウントは Amazon VPC にアクセスできますか？	その VPC は新しい環境と同じ AWS アカウント およびリージョンにありますか？	VPC は AWS アカウントのデフォルト VPC ですか。	VPC 内に単一のサブネットがありますか。	以下の手順に従います
あり	はい	いいえ	いいえ	[Network settings (ネットワーク設定)] を展開します。 [Network (VPC) (ネットワーク (VPC))] で、AWS Cloud9 が使用する VPC を選択します。 選択した VPC に複数のサブネットがある場合は、 [Network settings (advanced) (ネットワーク設定 (高度))] を展開します。 Subnet で、選択した VPC AWS Cloud9 で使用するサブネットを選択します。

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
				選択した VPC にサブネットが ない場合は、サ ブネットを作成 します。これ を行うには、 [Create new subnet (新しい サブネットの 作成)] を選択 し、画面の指示 に従います。詳 細については、 「 のサブネッ トを作成する AWS Cloud9 」 を参照してくだ さい。

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
はい	なし	はい	—	AWS Cloud9 は、新しい環境 のアカウント AWS アカウ ントとは異なる でデフォルトの VPC を使用で きません。この リストで別のオ プションを選択 します。

AWS アカウントは Amazon VPC にアクセスできますか？	その VPC は新しい環境と同じ AWS アカウント およびリージョンにありますか？	VPC は AWS アカウントのデフォルト VPC ですか。	VPC 内に単一のサブネットがありますか。	以下の手順に従います
はい	いいえ	いいえ	はい	[Network settings (ネットワーク設定)] を展開します。 [Network (VPC) (ネットワーク (VPC))] で、AWS Cloud9 が使用する VPC を選択します。  Note VPC は、別のアカウントにある場合でも、新しい環境と同じリージョンに存在する必要が

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
				ありま す。

AWS アカウントは Amazon VPC にアクセスできますか？	その VPC は新しい環境と同じ AWS アカウント およびリージョンにありますか？	VPC は AWS アカウントのデフォルト VPC ですか。	VPC 内に単一のサブネットがありますか。	以下の手順に従います
はい	いいえ	いいえ	いいえ	[Network settings (ネットワーク設定)] を展開します。 [Network (VPC) (ネットワーク (VPC))] で、AWS Cloud9 が使用する VPC を選択します。 [Subnet (サブネット)] で、選択した VPC で AWS Cloud9 が使用するサブネットを選択します。 選択した VPC にサブネットがない場合は、新しい環境 AWS アカウントとは異なる環境で VPC のサブネットを作成するには、「Amazon

AWS アカウ ントは Amazon VPC にアクセ スできますか？	その VPC は新 しい環境と同じ AWS アカウ ント およびリー ジョンにありま すか？	VPC は AWS ア カウントのデ フォルト VPC ですか。	VPC 内に単一 のサブネットが ありますか。	以下の手順に従 います
				VPC ユーザーガイドの「共有 VPCs」を参 照してくださ い。  Note VPC お よびサ ブネッ トは、 別のア カウ ントにあ る場合 でも、 新し い環 境と同 じリー ジョン に存在 する 必要が ありま す。

これらの選択肢の詳細については、「[AWS Cloud9 開発環境の VPC 設定](#)」を参照してください。

9. 最大 50 個のタグを追加します。タグごとにキーと値を指定します。これを行うには、[Add new tag] (新しいタグを追加) を選択します。タグはリソースタグとして AWS Cloud9 環境にアタッチされ、CloudFormation スタック、Amazon EC2 インスタンス、Amazon EC2 セキュリティグループという基盤となるリソースに伝達されます。タグの詳細については、「[IAM ユーザーガイド](#)」の[AWS 「リソースタグを使用したアクセスの制御」](#)およびこのガイドの[「詳細情報」](#)を参照してください。

 Warning

これらのタグを作成後に更新した場合、変更は基になるリソースには反映されません。詳細については、[タグ](#)に関する詳細情報の「[基礎となるリソースへのタグ更新の伝播](#)」を参照してください。

10. [Create] (作成) を選択して環境を作成すると、ホームページにリダイレクトされます。アカウントが正常に作成されると、AWS Cloud9 コンソールの上部に緑色のフラッシュバーが表示されます。新しい環境を選択し、[Open in Cloud9] (Cloud9 で開く) を選択して IDE を起動できます。

[Delete](#)[View details](#) [Open in Cloud9](#) [Create environment](#)

アカウントの作成に失敗すると、AWS Cloud9 コンソールの上部に赤いフラッシュバーが表示されます。アカウントの作成に失敗する原因としては、ウェブブラウザ、AWS アクセス許可、インスタンス、または関連するネットワークの問題が考えられます。可能な解決方法は、「[AWS Cloud9 のトラブルシューティング](#)」セクションで参照できます。

 Note

AWS Cloud9 はIMDSv1 と IMDSv2 の両方をサポートしています。IMDSv1 よりもセキュリティレベルが強化されているため、IMDSv2 の導入をお勧めします。IMDSv2 の利点の詳細については、「[AWS セキュリティブログ](#)」を参照してください。IMDSv1 から IMDSv2 への移行の詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「[インスタンスメタデータサービスバージョン 2 の使用への移行](#)」を参照してください。

Note

環境がプロキシを使用してインターネットにアクセスしている場合は、依存関係をインストール AWS Cloud9 できるように、プロキシの詳細を に提供する必要があります。詳細については、「[依存関係をインストールできませんでした](#)」を参照してください。

コードで環境を作成する

コードを使用して で EC2 環境を作成するには AWS Cloud9、次のように EC2 環境の作成オペレーションを AWS Cloud9 呼び出します。

AWS CLI	create-environment-ec2
AWS SDK for C++	CreateEnvironmentEC2Request 、 CreateEnvironmentEC2Result
AWS SDK for Go	CreateEnvironmentEC2 、 CreateEnvironmentEC2Request 、 CreateEnvironmentEC2WithContext
AWS SDK for Java	CreateEnvironmentEC2Request 、 CreateEnvironmentEC2Result
AWS SDK for JavaScript	createEnvironmentEC2
AWS SDK for .NET	CreateEnvironmentEC2Request 、 CreateEnvironmentEC2Response
AWS SDK for PHP	createEnvironmentEC2
AWS SDK for Python (Boto)	create_environment_ec2
AWS SDK for Ruby	create_environment_ec2
AWS Tools for Windows PowerShell	New-C9EnvironmentEC2
AWS Cloud9 API	CreateEnvironmentEC2

Note

環境がプロキシを使用してインターネットにアクセスしている場合は、依存関係をインストール AWS Cloud9 できるように、プロキシの詳細を に提供する必要があります。詳細については、「[依存関係をインストールできませんでした](#)」を参照してください。

SSH 環境を作成する

AWS Cloud9 コンソールを使用して SSH AWS Cloud9 開発環境を作成します。CLI を使用して SSH 環境を作成することはできません。

前提条件

- まず、「[セットアップ AWS Cloud9](#)」のステップを完了していることを確認します。これにより、AWS Cloud9 コンソールにサインインして環境を作成できます。
- 環境 AWS Cloud9 に接続する既存のクラウドコンピューティングインスタンス (の Amazon EC2 インスタンスなど AWS アカウント) または独自のサーバーを特定します。
- 既存のインスタンスまたは独自のサーバーが、すべての [SSH ホスト要件](#)を満たしていることを確認します。これには、特定のバージョンの Python、Node.js、およびその他のコンポーネントをインストールしていること、ログイン後に AWS Cloud9 をスタートしたいディレクトリに具体的な許可を設定していること、関連した Amazon Virtual Private Cloud を設定していることなどが含まれます。

SSH 環境を作成する

- 前述の前提条件を完了していることを確認してください。
- SSH クライアントを使用して既存のインスタンスまたは独自のサーバーに接続します (まだ接続していない場合)。これにより、インスタンスまたはサーバーに必要な公開 SSH キー値を追加できます。詳細については、この手順で後ほど説明します。

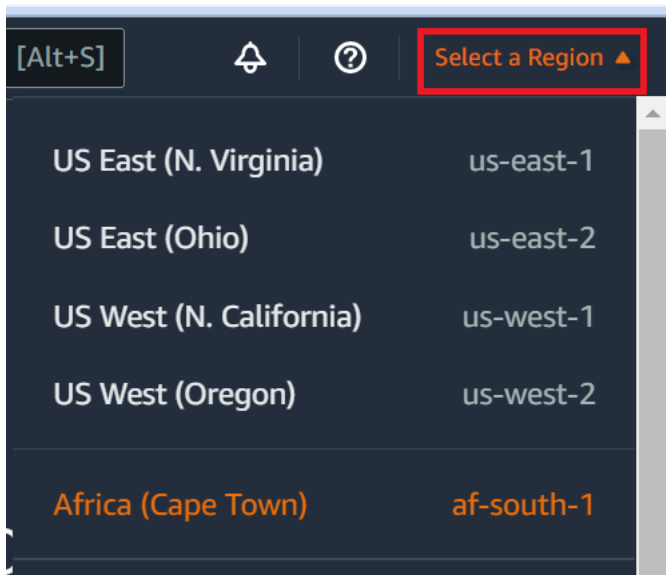
Note

既存の AWS クラウド コンピューティングインスタンスに接続するには、次のリソースの 1 つ以上を参照してください。

- Amazon EC2 については、「Amazon EC2 ユーザーガイド」の「[Linux インスタンスに接続](#)」を参照してください。
- Amazon Lightsail については、Amazon Lightsail ドキュメントの「[Linux/UNIX ベースの Lightsail インスタンスへの接続](#)」を参照してください。
- については AWS Elastic Beanstalk、「AWS Elastic Beanstalk デベロッパーガイド」の「[サーバーインスタンスの一覧表示と接続](#)」を参照してください。
- については AWS OpsWorks、「AWS OpsWorks ユーザーガイド」の「[SSH を使用して Linux インスタンスにログインする](#)」を参照してください。
- その他の方法については AWS のサービス、その特定のサービスのドキュメントを参照してください。

独自のサーバーに接続するには、SSH を使用します。SSH は macOS および Linux オペレーティングシステムに既にインストールされています。Windows で SSH を使用してサーバーに接続するには、[PuTTY](#) をインストールする必要があります。

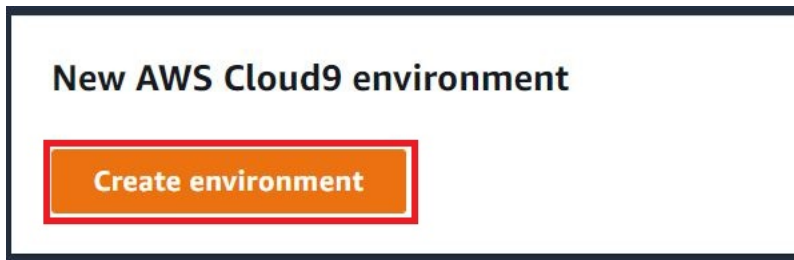
3. AWS Cloud9 コンソールにサインインするには、<https://console.aws.amazon.com/cloud9/> を参照してください。
4. AWS Cloud9 コンソールにサインインしたら、上部のナビゲーションバーで AWS リージョンを選択して環境を作成します。使用可能な のリストについては AWS リージョン、[AWS Cloud9](#) の「」を参照してくださいAWS 全般のリファレンス。



5. 今回、開発環境を初めて作成する場合は、ウェルカムページが表示されます。新しい AWS Cloud9 環境パネルで、環境の作成を選択します。

以前に開発環境を作成したことがある場合は、画面の左側にあるペインを展開することもできます。[Your environments] (自分の環境)、[Create environment] (環境の作成) の順に選択します。

ウェルカムページの場合:



または、[Your environments] (自分の環境) ページの場合:



6. [Create environment] (環境の作成) ページで、環境の名前を入力します。
7. [Description (説明)] に、環境に関する説明を入力します。このチュートリアルでは、This environment is for the AWS Cloud9 tutorial. を使用します。
8. [Environment type] (環境タイプ) で、以下のオプションから [Existing Compute] (既存のコンピューティング) を選択します。
 - 新しい EC2 インスタンス – SSH または SSM 経由で に直接接続 AWS Cloud9 できる Amazon EC2 インスタンスを起動します。
 - 既存のコンピューティング – SSH ログインの詳細とポート 22 を開く必要がある既存の Amazon EC2 インスタンスを起動します。 を介してインスタンス AWS Cloud9 に接続します [AWS Systems Manager](#)。
 - [Existing compute] (既存のコンピューティング) オプションを選択すると、サービスロールが自動的に作成されます。サービスロールの名前は、インターフェイスの下方にある [Systems Manager アクセス用のサービスロールとインスタンスプロファイル] セクションで確認できます。詳細については、「[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)」を参照してください。

⚠ Warning

環境に EC2 インスタンスを作成すると、Amazon EC2 AWS アカウント の に料金が発生する可能性があります。Systems Manager を使用して EC2 インスタンスへの接続を管理する場合、追加コストはかかりません。

⚠ Warning

AWS Cloud9 は SSH パブリックキーを使用してサーバーに安全に接続します。安全な接続を確立するには、以下の手順に従って公開キーを `~/.ssh/authorized_keys` ファイルに追加し、ログイン認証情報を入力します。[Copy key to clipboard] (キーをクリップボードにコピー) を選択して SSH キーをコピーするか、[View public SSH key] (SSH 公開キーを表示) を選択してキーを表示します。

9. [Existing compute] (既存のコンピューティング) パネルの [User] (ユーザー) に、この手順で先にインスタンスまたはサーバーに接続するときに使用したログイン名を入力します。例えば、AWS クラウド コンピューティングインスタンスの場合は、`ec2-user`、`ubuntu`、または `root` を入力します。

i Note

ログイン名をインスタンスやサーバーの管理者アクセス許可または管理者ユーザーに関連付けることをお勧めします。より具体的には、このログイン名がインスタンスやサーバーの Node.js インストールの所有者であることが推奨されます。これをチェックするには、インスタンスやサーバーのターミナルから、`ls -l $(which node)` (または `nvm` を使用している場合は `ls -l $(nvm which node)`) コマンドを実行します。このコマンドは、Node.js インストールの所有者名を表示します。また、インストールのアクセス許可、グループ名、場所も表示されます。

10. [Host] (ホスト) に、インスタンスまたはサーバーのパブリック IP アドレス (優先) またはホスト名を入力します。
11. ポート には、インスタンスまたはサーバーへの接続を試みるために AWS Cloud9 使用するポートを入力します。または、デフォルトのポートをそのまま使用します。

12. [Additional details - optional] (その他の詳細 - オプション) を選択し、環境パス、node.js バイナリへのパス、SSH ジャンプホスト情報を表示します。
13. 環境パスには、AWS Cloud9 開始するインスタンスまたはサーバーのディレクトリへのパスを入力します。この手順の前提条件で、以前、これを確認しました。空白のままにすると、AWS Cloud9 はログイン後に通常インスタンスまたはサーバーを起動するディレクトリを使用します。これは通常、ホームまたはデフォルトのディレクトリです。
14. [Path to Node.js binary] (Node.js バイナリへのパス) にパス情報を入力し、インスタンスまたはサーバーの Node.js バイナリへのパスを指定します。パスを取得するには、インスタンスまたはサーバーでコマンド **which node** (nvm を使用している場合は **nvm which node**) を実行できます。例えば、パスは `/usr/bin/node` のようになります。空白のままにした場合、AWS Cloud9 は接続の試行時に Node.js バイナリの場所の推測を試みます。
15. [SSH jump host] (SSH ジャンプホスト) に、インスタンスまたはサーバーが使用するジャンプホストに関する情報を入力します。USER_NAME@HOSTNAME:PORT_NUMBER 形式を使用します (例: `ec2-user@ip-192-0-2-0:22`)。

ジャンプホストは、以下の要件を満たしている必要があります。

- SSH を使用してパブリックインターネット経由で到達可能とする必要があります。
 - 特定のポートを経由したすべての IP アドレスからのインバウンドアクセスを許可する必要があります。
 - 既存のインスタンスまたはサーバーの `~/.ssh/authorized_keys` ファイルにコピーしたパブリック SSH キー値を、ジャンプホストの `~/.ssh/authorized_keys` ファイルにもコピーする必要があります。
 - Netcat をインストールする必要があります。
16. 最大 50 個のタグを追加します。タグごとにキーと値を指定します。これを行うには、[Add new tag] (新しいタグを追加) を選択します。タグはリソースタグとして AWS Cloud9 環境にアタッチされ、CloudFormation スタック、Amazon EC2 インスタンス、Amazon EC2 セキュリティグループという基盤となるリソースに伝達されます。タグの詳細については、[「IAM ユーザーガイド」のAWS「リソースタグを使用したアクセスの制御」](#)と、このガイドのタグに関する詳細[情報](#)を参照してください。

⚠ Warning

これらのタグを作成後に更新した場合、変更は基になるリソースには反映されません。詳細については、[タグ](#)に関する詳細情報の「[基礎となるリソースへのタグ更新の伝播](#)」を参照してください。

17. [Create] (作成) を選択して環境を作成すると、ホームページにリダイレクトされます。アカウントが正常に作成されると、AWS Cloud9 コンソールの上部に緑色のフラッシュバーが表示されます。新しい環境を選択し、[Open in Cloud9] (Cloud9 で開く) を選択して IDE を起動できます。

Delete

View details

 Open in Cloud9 

Create environment

アカウントの作成に失敗すると、AWS Cloud9 コンソールの上部に赤い点滅バーが表示されます。ウェブブラウザ、AWS アクセス許可、インスタンス、または関連するネットワークに問題があるため、アカウントの作成に失敗することがあります。アカウントが失敗する原因と考えられる問題の可能な解決方法については、「[AWS Cloud9 のトラブルシューティング](#)」セクションを参照してください。

i Note

環境がプロキシを使用してインターネットにアクセスしている場合は、依存関係をインストール AWS Cloud9 できるように、プロキシの詳細を に提供する必要があります。詳細については、「[依存関係をインストールできませんでした](#)」を参照してください。

を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager

EC2 環境用に作成された「no-ingress EC2 インスタンス」を使用すると AWS Cloud9、そのインスタンスでインバウンドポートを開くことなく Amazon EC2 インスタンスに接続できます。コンソール、コマンドラインインターフェイス、または [CloudFormation スタック](#) を使って EC2 環境を作成するときに、no-ingress オプションを選択できます。コンソールまたはコマンドラインインターフェイスを使用して環境を作成する方法の詳細については、「[the section called “ステップ 1: 環境を作成する”](#)」を参照してください。

⚠ Important

Systems Manager Session Manager 使って EC2 インスタンスへの接続を管理する場合、追加料金は発生しません。

コンソールの [Create environment] (環境の作成) ページで環境タイプを選択するときに、インバウンド接続を必要とする新しい EC2 インスタンスを選択するか、次を必要としない新しい no-ingress EC2 インスタンスを選択できます。

- [新しい EC2 インスタンス](#) — この設定の場合、インスタンスのセキュリティグループには、着信ネットワークトラフィックを許可するルールがあります。着信ネットワークトラフィックは[AWS Cloud9 接続のため承認された IP アドレス](#)に制限されています。オープンインバウンドポートを使用すると AWS Cloud9、は SSH 経由でインスタンスに接続できます。AWS Systems Manager Session Manager を使用する場合は、インバウンドポート (進入なし) を開くことなく、SSM を介して Amazon EC2 インスタンスにアクセスできます。この方法は、新しい Amazon EC2 インスタンスでのみ使用できます。詳細については、「[EC2 環境で Systems Manager を使用する利点](#)」を参照してください。
- [既存のコンピューティング](#) — この設定の場合、SSH ログインの詳細を必要とする既存の Amazon EC2 インスタンスにアクセスします。このインスタンスにはインバウンドセキュリティグループルールが必要です。このオプションを使用する場合は、サービスロールが自動的に作成されます。サービスロールの名前は、設定画面の下部にある注記で確認できます。

[AWS CLI](#) を使用して環境を作成する場合、create-environment-ec2 コマンドを呼び出す時に、--connection-type CONNECT_SSM オプションを設定して、no-ingress EC2 インスタンスを設定できます。必要なサービスロールとインスタンスプロファイルの作成に関する詳細については、「[を使用した Systems Manager のインスタンスプロファイルの管理 AWS CLI](#)」を参照してください。

no-ingress EC2 インスタンスを使用する環境の作成が完了したら、以下を確認します。

- Systems Manager Session Manager には、ユーザーに代わって EC2 インスタンスでアクションを実行する許可がおりています。詳細については、「[Systems Manager 許可の管理](#)」を参照してください。

- AWS Cloud9 ユーザーは Session Manager によって管理されるインスタンスにアクセスできます。詳細については、「[Session Manager によって管理されているインスタンスへのアクセスをユーザーに与える](#)」を参照してください。

EC2 環境で Systems Manager を使用する利点

[Session Manager](#) が AWS Cloud9 とその EC2 インスタンス間の安全な接続を処理できるようにするには、次の 2 つの主な利点があります。

- インスタンスのインバウンドポートを開く必要がない
- パブリックサブネットまたはプライベートサブネットでインスタンスを起動するオプション

No open inbound ports

AWS Cloud9 とその EC2 インスタンス間の安全な接続は、[Session Manager](#) によって処理されます。Session Manager は、インバウンドポートを開くことなく AWS Cloud9 が EC2 インスタンスに接続できるようにするフルマネージド Systems Manager 機能です。

Important

no-ingress 接続に Systems Manager を使用するオプションは、現在、新しい EC2 環境を作成する場合にのみ使用できます。

Session Manager セッションがスタートすると、ターゲットインスタンスへの接続が行われます。接続を確立すると、環境は Systems Manager サービスを通じてインスタンスと対話できるようになります。Systems Manager サービスは、Systems Manager Agent ([SSM Agent](#)) を通じてインスタンスと通信を行います。

デフォルトでは、SSM Agent は、EC2 環境によって使用されるすべてのインスタンスにインストールされます。

Private/public subnets

インスタンスのサブネットをネットワーク 設定 (アドバンスド) セクションで選択する時は、ご使用の環境のインスタンスに Systems Manager 経由でアクセスする場合は、プライベートサブネットまたはパブリックサブネットを選択できます。

▼ **Network settings (advanced)**

Network (VPC)
Launch your EC2 instance into an existing Amazon Virtual Private Cloud (VPC) or create a new one.

vpc- ▼  [Create new VPC](#)

Subnet
Select the subnet in which the EC2 instance is created. For a private subnet, ensure it has internet connectivity by adding a NAT gateway. Public or private IP depends on the subnet (public or private).

No preference (default subnet in any Availability Zone) ▼  [Create new subnet](#)

 Temporary managed credentials can't be used in private subnets.

No tags associated with the resource.

[Add new tag](#)

You can add 50 more tags.

プライベートサブネット

プライベートサブネットの場合、インスタンスが SSM サービスにまだ接続できることを確認してください。これは、[パブリックサブネットに NAT ゲートウェイを設定する](#)または [Systems Manager の VPC エンドポイントの設定](#)。によって行うことができます。

NAT ゲートウェイを使用する利点は、インターネットがプライベートサブネット内のインスタンスへの接続を開始できなくなることです。環境のインスタンスには、パブリック IP アドレスではなくプライベート IP アドレスが割り当てられます。したがって、NAT ゲートウェイはインスタンスからインターネットまたは他の AWS サービスにトラフィックを転送し、レスポンスをインスタンスに送り返します。

VPC オプションでは、Systems Manager

に、com.amazonaws.region.ssm、com.amazonaws.region.ec2メッセージ、およびcom.amazonaws.region.ssmessagesと、少なくとも 3 つのインターフェイスエンドポイントを作成します。詳細については、AWS Systems Manager ユーザーガイドの「[Systems Manager の VPC エンドポイントを作成する](#)」を参照してください。

⚠ Important

現在、環境の EC2 インスタンスがプライベートサブネットで起動されている場合、[AWS マネージド一時認証情報](#)を使用して EC2 環境が AWS エンティティ (IAM ユーザーなど) に代わって AWS サービスにアクセスすることを許可することはできません。

パブリックサブネット

開発環境が SSM を使用して EC2 インスタンスにアクセスしている場合は、起動先のパブリックサブネットで、インスタンスがパブリック IP アドレスに割り当てられていることを確認します。これを行うには、独自の IP アドレスを指定するか、パブリック IP アドレスの自動割り当てを有効にします。自動割り当て IP 設定を変更する時に含まれる指示については、Amazon VPC ユーザーガイド内の[VPC の IP アドレス指定](#)を参照してください。

環境インスタンスのプライベートサブネットとパブリックサブネットの設定の詳細については、「[のサブネットを作成する AWS Cloud9](#)」を参照してください。

Systems Manager 許可の管理

デフォルトでは、Systems Manager には、EC2 インスタンス上でアクションを実行する許可がありません。アクセスは AWS Identity and Access Management (IAM) インスタンスプロファイルを通じて提供されます。(インスタンスプロファイルは、起動時に EC2 インスタンスに IAM ロール情報を渡すコンテナです。)

AWS Cloud9 コンソールを使用して no-ingress EC2 インスタンスを作成すると、サービスロール (AWSCloud9SSMAccessRole) と IAM インスタンスプロファイル (AWSCloud9SSMInstanceProfile) の両方が自動的に作成されます。(IAM マネジメントコンソールで AWSCloud9SSMAccessRole を表示できます。インスタンスプロファイルは IAM コンソールに表示されません)。

⚠ Important

で初めて no-ingress EC2 環境を作成する場合は AWS CLI、必要なサービスロールとインスタンスプロファイルを明示的に定義する必要があります。詳細については、「[を使用した Systems Manager のインスタンスプロファイルの管理 AWS CLI](#)」を参照してください。

⚠ Important

AWS Cloud9 環境を作成し、AWSCloud9Administrator または AWSCloud9User ポリシーがアタッチされた Amazon EC2 Systems Manager を使用している場合は、特定の IAM アクセス許可を持つカスタムポリシーもアタッチする必要があります。「」を参照してください [SSM 環境作成用のカスタム IAM ポリシー](#)。これは、AWSCloud9Administrator ポリシーと AWSCloud9User ポリシーに伴うアクセス許可の問題が原因です。

セキュリティ保護を強化するために、AWS Cloud9 サービスにリンクされたロールには AWSServiceRoleforAWSCloud9、AWSCloud9ServiceRolePolicy ポリシー PassRole の制限があります。IAM ロールをサービスに Pass (適用) すると、そのサービスがロールを引き受け、ユーザーに代わってアクションを実行できる許可が得られます。この場合、アクセス PassRole 許可により、は AWSCloud9SSMAccessRole ロール (およびそのアクセス許可) のみを EC2 インスタンスに渡す AWS Cloud9 ことができます。これにより、EC2 インスタンスで実行できるアクションは、AWS Cloud9 が必要とするもののみに限定されます。

i Note

Systems Manager を使用してインスタンスにアクセスする必要がなくなった場合は、AWSCloud9SSMAccessRole サービスロールを削除できます。詳細については、IAM ユーザーガイドの「[ロールまたはインスタンスプロファイルを削除する](#)」を参照してください。

を使用した Systems Manager のインスタンスプロファイルの管理 AWS CLI

また、AWS CLI を使って no-ingress EC2 環境を作成できます。create-environment-ec2 を呼び出すとき、--connection-type オプションを CONNECT_SSM に設定します。

このオプションを使用する場合、AWSCloud9SSMAccessRole サービスロールと AWSCloud9SSMInstanceProfile は自動的に作成されません。そのため、必要なサービスプロファイルとインスタンスプロファイルを作成するには、次のいずれかを行います。

- コンソールを使って EC2 環境を作成すると、その後は自動的に AWSCloud9SSMAccessRole サービスロールと AWSCloud9SSMInstanceProfile が自動的に作成されます。作成後、サービスロールとインスタンスプロファイルは、AWS CLI を使って作成した追加 EC2 環境で利用できます。

- 次の AWS CLI コマンドを実行して、サービスロールとインスタンスプロファイルを作成します。

```
aws iam create-role --role-name AWSCloud9SSMAccessRole --path /service-role/ --
assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Effect":
"Allow", "Principal": {"Service": ["ec2.amazonaws.com", "cloud9.amazonaws.com"]
}, "Action": "sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSCloud9SSMAccessRole --policy-arn
arn:aws:iam::aws:policy/AWSCloud9SSMInstanceProfile
aws iam create-instance-profile --instance-profile-name AWSCloud9SSMInstanceProfile
--path /cloud9/
aws iam add-role-to-instance-profile --instance-profile-name
AWSCloud9SSMInstanceProfile --role-name AWSCloud9SSMAccessRole
```

Session Manager によって管理されているインスタンスへのアクセスをユーザーに与える

Systems Manager を介して EC2 インスタンスに接続されている AWS Cloud9 環境を開くには、ユーザーに API オペレーション のアクセス許可が必要です `StartSession`。このオペレーションにより、Session Manager セッション用マネージド EC2 インスタンスへの接続が開始されます。AWS Cloud9 特定の管理ポリシー (推奨) を使用するか、IAM ポリシーを編集して必要なアクセス許可を追加することで、ユーザーにアクセス権を付与できます。

Method	説明
使用 AWS Cloud9固有の管理ポリシー	Systems Manager によって AWS 管理される EC2 インスタンスへのアクセスをユーザーに許可するには、管理ポリシーを使用することをお勧めします。管理ポリシーは、標準の AWS Cloud9 ユースケースに対する一連のアクセス許可を提供し、IAM エンティティに簡単にアタッチできます。 すべてのマネージドポリシーには、<code>StartSession</code> API オペレーションを実行する許可も含まれています。以下は、固有の管理ポリシーです <code>AWS Cloud9</code>。

Method	説明
	• <code>AWSCloud9Administrator</code> (<code>arn:aws:iam::aws:policy/AWSCloud9Administrator</code>) • <code>AWSCloud9User</code> (<code>arn:aws:iam::aws:policy/AWSCloud9User</code>) • <code>AWSCloud9EnvironmentMember</code> (<code>arn:aws:iam::aws:policy/AWSCloud9EnvironmentMember</code>)
	 Important AWS Cloud9 環境を作成し、<code>AWSCloud9Administrator</code> または <code>AWSCloud9User</code> ポリシーがアタッチされた Amazon EC2 Systems Manager を使用している場合は、特定の IAM アクセス許可を持つカスタムポリシーもアタッチする必要があります。「」を参照してくださいSSM 環境作成用のカスタム IAM ポリシー。これは、<code>AWSCloud9Administrator</code> ポリシーと <code>AWSCloud9User</code> ポリシーに伴うアクセス許可の問題が原因です。
	詳細については、「AWS の マネージドポリシー AWS Cloud9」を参照してください。

Method	説明
IAM ポリシーを編集し、必要なポリシーステートメントを追加する	既存のポリシーを編集するため、StartSession APIのアクセス 許可を追加できます。AWS マネジメントコンソール または を使用してポリシーを編集するには AWS CLI、「IAM ユーザーガイド」の「IAM ポリシーの編集」に記載されている手順に従ってください。 ポリシーを編集するときに、ssm:startSession API オペレーションの実行を許可する policy statement (以下を参照) を追加します。

次のアクセス許可を使用すると、StartSession API オペレーションの実行が有効になります。ssm:resourceTag 条件キーは、インスタスが AWS Cloud9 EC2 開発環境 (Resource: arn:aws:ec2:*:*:instance/*) であることを条件として、任意のインスタス () に対して Session Manager セッションを開始できることを指定しますaws:cloud9:environment。

Note

次のマネージドポリシーには、AWSCloud9Administrator、AWSCloud9User、およびAWSCloud9EnvironmentMember というポリシーステートメントも含まれています。

```
{
  "Effect": "Allow",
  "Action": "ssm:StartSession",
  "Resource": "arn:aws:ec2:*:*:instance/*",
  "Condition": {
    "StringLike": {
      "ssm:resourceTag/aws:cloud9:environment": "*"
    },
    "StringEquals": {
      "aws:CalledViaFirst": "cloud9.amazonaws.com"
    }
  }
},
```

```
{
  "Effect": "Allow",
  "Action": [
    "ssm:StartSession"
  ],
  "Resource": [
    "arn:aws:ssm:*:*:document/*"
  ]
}
```

CloudFormation を使用して非進入 EC2 環境を作成する

[CloudFormation テンプレート](#)を使用して、no-ingress Amazon EC2 開発環境を定義する場合は、スタックを作成する前に次のことを行います。

1. AWSCloud9SSMAccessRole サービスロールとAWSCloud9SSMInstanceProfile インスタンスプロファイルを作成します。詳細については、「[CloudFormation テンプレートを使用したサービスロールとインスタンスプロファイルの作成](#)」を参照してください。
2. を呼び出す IAM エンティティのポリシーを更新します CloudFormation。こうすることで、EC2 インスタンスに接続する Session Manager セッションをスタートできます。詳細については、「[IAM ポリシーに Systems Manager 許可を追加する](#)」を参照してください。

CloudFormation テンプレートを使用したサービスロールとインスタンスプロファイルの作成

サービスロール AWSCloud9SSMAccessRole とインスタンスプロファイル

AWSCloud9SSMInstanceProfileを作成して、Systems Manager による開発環境をバックアップする EC2 インスタンスの管理を有効にします。

以前に を作成しAWSCloud9SSMAccessRole、no-ingress EC2 環境を作成[with the console](#)または[AWS CLI コマンドを実行](#)AWSCloud9SSMInstanceProfileした場合、サービスロールとインスタンスプロファイルは既に使用できます。

Note

非進入 EC2 環境の CloudFormation スタックを作成しようとしたが、最初に必要なサービスロールとインスタンスプロファイルを作成しなかったとします。そうすると、スタックは作成されず、次のエラーメッセージが表示されます。

インスタンスプロファイル `AWSCloud9SSMinStanceProfile` がアカウントに存在しません。

を使用してイングレスなし EC2 環境を初めて作成する場合は CloudFormation、テンプレートで `AWSCloud9SSMAccessRole` および `AWSCloud9SSMInstanceProfile` を IAM リソースとして定義できます。

サンプルテンプレートからのこの抜粋は、これらのリソースを定義する方法を示しています。AssumeRole アクションは、AWS Cloud9 環境とその EC2 インスタンスの両方へのアクセスを提供するセキュリティ認証情報を返します。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AWSCloud9SSMAccessRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - cloud9.amazonaws.com
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Description: 'Service linked role for AWS Cloud9'
      Path: '/service-role/'
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSCloud9SSMInstanceProfile
      RoleName: 'AWSCloud9SSMAccessRole'

  AWSCloud9SSMInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      InstanceProfileName: AWSCloud9SSMInstanceProfile
      Path: "/cloud9/"
      Roles:
        -
          Ref: AWSCloud9SSMAccessRole
```

IAM ポリシーに Systems Manager 許可を追加する

[CloudFormation テンプレート](#)で[サービスロールとインスタンスプロファイルを定義](#)した後、スタックを作成する IAM エンティティに Session Manager セッションをスタートする許可があることを確認してください。セッションは、Session Manager を使用した EC2 インスタンス への接続です。

Note

no-ingress EC2 環境用のスタックを作成する前に Session Manager セッションをスタートする許可を追加しない場合、AccessDeniedException エラーが返されます。

CloudFormationを呼び出すため、次のアクセス許可を IAM エンティティのポリシーに追加します。

```
{
    "Effect": "Allow",
    "Action": "ssm:StartSession",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
        "StringLike": {
            "ssm:resourceTag/aws:cloud9:environment": "*"
        },
        "StringEquals": {
            "aws:CalledViaFirst": "cloudformation.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ssm:StartSession"
    ],
    "Resource": [
        "arn:aws:ssm:*:*:document/*"
    ]
}
```


依存関係をダウンロードするため、Amazon S3 の VPC エンドポイントを設定する

AWS Cloud9 環境の EC2 インスタンスがインターネットにアクセスできない場合は、指定された Amazon S3 バケットの VPC エンドポイントを作成します。このバケットには、IDE を最新の状態に保つために必要な依存関係が含まれています。

Amazon S3 の VPC エンドポイントを設定するには、アクセスポリシーをカスタマイズする必要があります。アクセスポリシーは、ダウンロードする依存関係を含む信頼された S3 バケットのみへのアクセスを許可するようにします。

Note

VPC エンドポイントは AWS マネジメントコンソール、AWS CLI、または Amazon VPC API を使用して作成および設定できます。次の手順では、コンソールインターフェイスを使用して VPC エンドポイントを作成する方法について説明します。

Amazon S3 の VPC エンドポイントを作成および設定する

1. AWS マネジメントコンソール、Amazon VPC のコンソールページに移動します。
2. ナビゲーションバーで、[エンドポイント] を選択します。
3. エンドポイントページで、[エンドポイント作成] を選択します。
4. エンドポイントの作成ページで、検索フィールドに「s3」と入力し、戻るを押して、現在の AWS リージョンで、Amazon S3 で利用可能なエンドポイントをリストします。
5. 返却された Amazon S3 エンドポイントのリストから、ゲートウェイタイプを選択できます。
6. 次に、環境の EC2 インスタンスが含まれている VPC を選択します。
7. 次に、VPC のルートテーブルを選択します。こうすることで、関連付けられたサブネットはエンドポイントにアクセスできるようになります。環境の EC2 インスタンスは、これらのサブネットの 1 つにあります。
8. ポリシーセクションで、カスタムオプションを選択し、スタンダードポリシーを以下に置き換えます。

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "Access-to-C9-bucket-only",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::{bucket_name}/content/dependencies/*"
  }
]
}

```

Resource 要素に対して、AWS リージョンで {bucket_name} を利用可能なバケットの実際の名前に置き換えます。たとえば、欧州 (アイルランド) リージョン AWS Cloud9 でを使用している場合は、を指定します "Resource": "arn:aws:s3:::static-eu-west-1-prod-static-hld3vzaf7c4h/content/dependencies/".

次の表に、AWS リージョン が AWS Cloud9 利用可能な のバケット名を示します。

AWS Cloud9 リージョンの Amazon S3 バケット

AWS リージョン	バケット名
米国東部(オハイオ)	static-us-east-2-prod-static-1c3sfcvf9hy4m
米国東部 (バージニア北部)	static-us-east-1-prod-static-mft1klnc4hl
米国西部 (オレゴン)	static-us-west-2-prod-static-p21mksqx9zlr
米国西部 (北カリフォルニア)	static-us-west-1-prod-static-16d59zrrp01z0
アフリカ (ケープタウン)	static-af-south-1-prod-static-v6v7i5ypdppv
アジアパシフィック (香港)	static-ap-east-1-prod-static-171xhpfkrorh6

AWS リージョン	バケット名
アジアパシフィック (ムンバイ)	static-ap-south-1-prod-static-ykocre202i9d
アジアパシフィック (大阪)	static-ap-northeast-3-prod-static-ivmxqzrx2ioi
アジアパシフィック (ソウル)	static-ap-northeast-2-prod-static-1wxycctlhwiajm
アジアパシフィック (シンガポール)	static-ap-southeast-1-prod-static-13ibpyrx4vk6d
アジアパシフィック (シドニー)	static-ap-southeast-2-prod-static-1cjsl8bx27rfu
アジアパシフィック (東京)	static-ap-northeast-1-prod-static-4fwvbdisquj8
カナダ (中部)	static-ca-central-1-prod-static-g80lpejy486c
欧州 (フランクフルト)	static-eu-central-1-prod-static-14lbgls2vrkh
欧州 (アイルランド)	static-eu-west-1-prod-static-hld3vzaf7c4h
欧州 (ロンドン)	static-eu-west-2-prod-static-36lbg202837x
欧州 (ミラノ)	static-eu-south-1-prod-static-1379tzkd3ni7d
ヨーロッパ (パリ)	static-eu-west-3-prod-static-1rwpkf766ke58
ヨーロッパ (ストックホルム)	static-eu-north-1-prod-static-1qzw982y7yu7e

AWS リージョン	バケット名
中東 (バーレーン)	static-me-south-1-prod-static-gmljex38qtqx
南米 (サンパウロ)	static-sa-east-1-prod-static-1cl8k0y7opidt
イスラエル (テルアビブ)	static-il-central-1-prod-static-k02vrnhcesue

9. エンドポイントの作成 を選択します。

正しい設定情報を指定した場合は、作成されたエンドポイントの ID がメッセージに表示されます。

- IDE が Amazon S3 バケットにアクセスできることをチェックするには、メニューバー上の [Window]、[New Terminal (新しいターミナル)] を選択してターミナルセッションをスタートします。次に、以下のコマンドを実行し、{bucket_name} をリージョンのバケットの実際の名前に置き換えます。

```
ping {bucket_name}.s3.{region}.amazonaws.com.
```

例えば、米国東部 (バージニア北部) リージョンで S3 バケットのエンドポイントを作成した場合は、次のコマンドを実行します。

```
ping static-us-east-1-prod-static-mft1klnc4hl.s3.us-east-1.amazonaws.com
```

ping が応答を受け取ると、IDE がバケットとその依存関係にアクセスできることを確認します。

この特徴の詳細については、「AWS PrivateLink ガイド」にある [Amazon S3 用エンドポイント](#) を参照してください。

プライベート接続用 VPC エンドポイントの設定

Systems Manager を使用するアクセスのオプションを使用して、インスタンスをサブネットに起動する時には、セキュリティグループには受信ネットワークトラフィックを許可するインバウンドルールがありません。ただし、セキュリティグループには、インスタンスからのアウトバウンドトラ

フィックを許可するアウトバウンドルールがあります。これは、IDE AWS Cloud9 を最新の状態に保つために必要なパッケージとライブラリをダウンロードするために必要です。

インスタンスのアウトバウンドトラフィックとインバウンドトラフィックを防ぐには、Systems Manager 用の Amazon VPC エンドポイントを作成して設定します。インターフェイス VPC エンドポイント (インターフェイスエンドポイント) を使用すると、使用するサービスに接続できます [AWS PrivateLink](#)。AWS PrivateLink は、プライベート IP アドレスを使用して Amazon EC2 および Systems Manager APIs にプライベートにアクセスするために使用できるテクノロジーです。VPC エンドポイントを設定して Systems Manager を使用するには、この [ナレッジセンターのソース](#)が提供する指示に従います。

Warning

インバウンドまたはアウトバウンドのネットワークトラフィックを許可しないセキュリティグループを設定するとします。その後、IDE をサポートする EC2 AWS Cloud9 インスタンスにはインターネットアクセスがありません。したがって、[VPC の Amazon S3 エンドポイント](#)を作成して、信頼できる S3 バケットに含まれる依存関係へのアクセスを許可する必要があります。さらに、などの一部の は AWS のサービス、インターネットアクセスなしでは意図したとおりに動作しない AWS Lambda場合があります。

では AWS PrivateLink、VPC エンドポイントを介して処理されるギガバイトごとにデータ処理料金が発生します。これは、トラフィックの送信元または送信先に関係なく行われます。詳細については、[AWS PrivateLink 料金表](#)を参照してください。

で環境を開く AWS Cloud9

この手順では、AWS Cloud9で環境を開く方法について説明します。

Note

この手順では、AWS Cloud9 開発環境を既に作成していることを前提としています。環境を作成する時は、[環境を作成する](#)を参照してください。

1. 次のように AWS Cloud9 コンソールにサインインします。

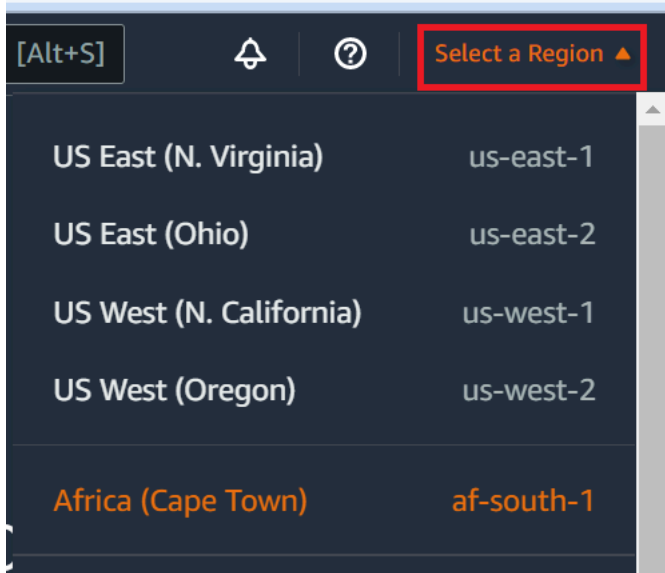
- ご自身が AWS アカウントを使用する唯一のユーザーである場合、または単一の AWS アカウントの IAM ユーザーである場合は、<https://console.aws.amazon.com/cloud9/> に移動します。

- 組織で 使用している場合は AWS IAM アイデンティティセンター、AWS アカウント 管理者にサインイン手順を依頼してください。

⚠ Important

[からサインアウト AWS アカウント](#)しても、IDE AWS Cloud9 には最大 5 分間アクセスできます。その後、必要なアクセス許可の有効期限が切れると、アクセスが拒否されます。

2. 上部のナビゲーションバーで、環境 AWS リージョン がある を選択します。



3. 環境リストの開きたい環境で、次のいずれかのアクションを実行します。

- カード内で、[Open in Cloud9] (Cloud9 で開く) リンクを選択します。
- カードを選択し、[Open in Cloud9] (Cloud9 で開く) ボタンを選択します。



環境がコンソールに表示されない場合は、次の 1 つ以上のアクションを試行して表示を試みてください。

- [Environments] (環境) ページのドロップダウンメニューバーで、以下の 1 つまたは複数を選択します。
- 環境を選択して、選択した AWS リージョン および 内で AWS エンティティが所有するすべての環境を表示します AWS アカウント。

- 選択した AWS リージョン と 内で AWS エンティティが招待されたすべての環境を表示するには、共有を選択します AWS アカウント。
- すべてのアカウント環境を選択して、選択した 内のすべての環境 AWS リージョン と、 AWS エンティティ AWS アカウント に表示許可がある環境を表示します。
- 自分がメンバーである環境が [Shared with you] (自分と共有) リストに表示されていない場合は、環境の所有者に確認してください。
- 上部のナビゲーションバーで、別のナビゲーションバーを選択します AWS リージョン。

の環境 AWS のサービス からの呼び出し AWS Cloud9

AWS Cloud9 開発環境 AWS のサービス から を呼び出すことができます。たとえば、次のアクションを実行できます。

- Amazon Simple Storage Service (Amazon S3) バケットにデータをアップロードおよびダウンロードします。
- Amazon Simple Notification Service (Amazon SNS) トピックを通じてブロードキャスト通知を送信します。
- Amazon DynamoDB (DynamoDB) データベースのデータの読み取り/書き込みを行う。

環境 AWS のサービス から を呼び出すには、いくつかの方法があります。たとえば、AWS Command Line Interface (AWS CLI) または AWS CloudShell を使用して、ターミナルセッションからコマンドを実行できます。環境内で実行したコードから AWS のサービス を呼び出すこともできます。これを行うには、JavaScript、Python、Ruby PHP Goなどのプログラミング言語に AWS SDKs を使用しますC++。詳細については、「[AWS CLI および aws-shell のサンプル](#)」、[AWS Command Line Interface ユーザーガイド](#)、および [AWS SDK](#) を参照してください。

AWS CLI、AWS CloudShell、またはコードが を呼び出すたびに AWS のサービス、、AWS CLI AWS CloudShell、またはコードは、呼び出しとともに一連の AWS アクセス認証情報を提供する必要があります。これらの認証情報は、呼び出し元にその呼び出しを行う適切なアクセス許可があるかどうかを判別します。認証情報に適切な許可がない場合は、呼び出しは失敗します。

環境に認証情報を提供するには、いくつかの方法があります。次の表は、いくつかのアプローチについて説明します。

環境タイプ	アプローチ
EC2	AWS マネージド一時認証情報を使用します。 EC2 環境では、このアプローチをお勧めします。AWS が管理する一時的な認証情報は、ユーザーに代わって EC2 環境の AWS アクセス認証情報を管理し、AWS セキュリティのベストプラクティスに従います。 EC2 環境を使用している場合は、このトピックの残りの部分はスキップできます。これは、AWS マネージド一時認証情報が環境で既に設定されているためです。 詳細については、「AWS マネージド一時認証情報」を参照してください。
EC2	IAM インスタンスプロファイルをインスタンスに添付します。 このアプローチは、何らかの理由で AWS マネージド一時認証情報を使用できない場合にのみ使用してください。AWS マネージド一時認証情報と同様に、インスタンスプロファイルはユーザーに代わって AWS アクセス認証情報を管理します。ただし、インスタンスプロファイルを作成して管理し、Amazon EC2 インスタンスに自分で直接アタッチする必要があります。 手順については、「インスタンスプロファイルを作成して使用し一時認証情報を管理する」を参照してください。
EC2 または SSH	永続的な AWS アクセス認証情報を環境内に保存します。 このアプローチは、一時的な AWS アクセス認証情報を使用するよりも安全性が低くなりま

環境タイプ	アプローチ
	す。ただし、SSH 環境で唯一サポートされているアプローチです。 手順については、「環境で永続的認証情報を作成して保存する」を参照してください。
EC2 または SSH	永続的な AWS アクセス認証情報をコードに直接挿入します。 セキュリティの AWS ベストプラクティスに従わないため、このアプローチはお勧めしません。 このアプローチは推奨しないため、このトピックでは説明しません。

インスタンスプロファイルを作成・使用しマネージド一時認証情報を管理する

Note

AWS Cloud9 SSH 開発環境でこの手順を使用することはできません。代わりに、「[環境で永続的認証情報を作成して保存する](#)」までスキップしてください。

インスタンスプロファイルの代わりに、AWS マネージド一時認証情報を使用することをお勧めします。何らかの理由で AWS マネージド一時認証情報を使用できない場合にのみ、これらの手順に従ってください。詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。

この手順では、IAM と Amazon EC2 を使用して、IAM インスタンスプロファイルを作成し、環境に接続する Amazon EC2 インスタンス環境にアタッチします。このインスタンスプロファイルは、ユーザーの代わりに一時認証情報を管理します。この手順は、AWS Cloud9の環境が既に作成されていることを前提としています。環境を作成するには、「[環境を作成する](#)」を参照してください。

これらのタスクは、[IAM と Amazon EC2 コンソール](#)、または[AWS コマンドラインインターフェイス \(AWS CLI\)](#) を使用して完了できます。

IAM コンソールを使用してインスタンスプロファイルを作成する

Note

インスタンスプロファイルを含む IAM ロールがすでにある場合は、この手順をスキップして、「[Amazon EC2 コンソールを使用してインスタンスプロファイルをインスタンスに添付する](#)」に進みます。

1. IAM コンソール (<https://console.aws.amazon.com/iam/>) にサインインします。

このステップでは、AWS アカウントの管理者レベルの認証情報使用にサインインすることをお勧めします。これが実行できない場合は、AWS アカウント 管理者にお問い合わせください。

2. ナビゲーションバーで [ロール] を選択します。

Note

IAM コンソールを使用してインスタンスプロファイルを単独で作成することはできません。インスタンスプロファイルを含む IAM ロールを作成する必要があります。

3. [ロールの作成] を選択してください。
4. [Select type of trusted entity] (信頼されたエンティティの種類を選択) ページで、AWS のサービスをすでに選択した状態にし、[Choose the service that will use this role] (このロールを使用するサービスを選択) で [EC2] を選択します。
5. [ユースケースの選択] で、[EC2] を選択します。
6. [Next: Permissions] (次のステップ: 許可) を選択します。
7. [Attach permissions policies (アクセス権限ポリシーをアタッチする)] ページで、ポリシーのリストの [AdministratorAccess] の横のボックスを選択して、[次へ: 確認] を選択します。

Note

AdministratorAccess ポリシーは、全体のすべての AWS アクションとリソースへの無制限のアクセスを許可します AWS アカウント。これは実験目的でのみ使用してください。詳細については、IAM ユーザーガイドの「[IAM ポリシー](#)」を参照してください。

8. [Review] (確認) ページの [Role Name] (ロール名) に、ロールの名前を入力します (例: my-demo-cloud9-instance-profile)。

9. [ロールの作成] を選択します。

「[Amazon EC2 コンソールを使用してインスタンスプロファイルをインスタンスにアタッチする](#)」に進みます。

を使用してインスタンスプロファイルを作成する AWS CLI

Note

インスタンスプロファイルを含む IAM ロールがすでにある場合は、スキップして、[AWS CLI を伴うインスタンスにインスタンスプロファイルを添付する](#)に進みます。
このトピックでは、で管理者レベルの認証情報 AWS CLI を使用してを設定することをお勧めします AWS アカウント。これが実行できない場合は、AWS アカウント 管理者にお問い合わせください。

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できます AWS CLI。

1. インスタンスプロファイルに必要な IAM ロール AWS の信頼関係を で定義します。これを行うには、次の内容のファイルを作成して保存します (例えば、my-demo-cloud9-instance-profile-role-trust.json)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      }
    }
  ]
}
```

```
    },  
    "Action": "sts:AssumeRole"  
  }  
]  
}
```

- ターミナルまたはコマンドプロンプトを使用して、このファイルを保存したディレクトリに切り替えます。
- インスタンスプロファイルの IAM ロールを作成します。これを行うには、IAM `create-role` コマンドを実行します。これを行うには、新しい IAM ロールの名前 (例えば、`my-demo-cloud9-instance-profile-role`) と、保存したファイルの名前を指定します。

```
aws iam create-role --role-name my-demo-cloud9-instance-profile-role --assume-role-policy-document file:///my-demo-cloud9-instance-profile-role-trust.json
```

- インスタンスプロファイル IAM ロールにアクセス AWS 許可をアタッチします。これを行うには、IAM `attach-role-policy` コマンドを実行します。既存の IAM ロールの名前と、という名前の AWS マネージドポリシーの Amazon リソースネーム (ARN) を指定します `AdministratorAccess`。

```
aws iam attach-role-policy --role-name my-demo-cloud9-instance-profile-role --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

Note

`AdministratorAccess` ポリシーは、全体のすべての AWS アクションとリソースへの無制限のアクセスを許可します AWS アカウント。これは実験目的でのみ使用してください。詳細については、IAM ユーザーガイドの「[IAM ポリシー](#)」を参照してください。

- インスタンスプロファイルを作成します。これを行うには、新しい IAM インスタンスプロファイルの名前を指定する IAM `create-instance-profile` コマンド (たとえば `my-demo-cloud9-instance-profile`) を実行します。

```
aws iam create-instance-profile --instance-profile-name my-demo-cloud9-instance-profile
```

- インスタンスプロファイルに IAM ロールを添付します。これを行うには、既存の IAM ロールとインスタンスプロファイルの名を指定する IAM `add-role-to-instance-profile` を実行します。

```
aws iam add-role-to-instance-profile --role-name my-demo-cloud9-instance-profile-role  
--instance-profile-name my-demo-cloud9-instance-profile
```

[AWS CLIを使用してインスタンスプロファイルを作成する](#)に進みます。

Amazon EC2 コンソールを使用してインスタンスプロファイルをインスタンスに添付する

1. <https://console.aws.amazon.com/ec2/> でAmazon EC2 コンソールにサインインします。

このステップでは、AWS アカウントの管理者の認証情報を使用してサインインすることをお勧めします。これが実行できない場合は、AWS アカウント 管理者にお問い合わせください。

2. ナビゲーションバーで、環境用と一致する AWS リージョン がリージョンセレクターに表示されていることを確認します。例えば、米国東部 (オハイオ) リージョンで環境を作成した場合は、このリージョンセレクターで米国東部 (オハイオ)を選択します。
3. [実行中のインスタンス] リンク、またはナビゲーションペインで [インスタンス] を展開して、[インスタンス] を選択します。
4. インスタンスのリストで、環境名を含む [名前] を持つインスタンスを選択します。たとえば、環境名が my-demo-environment の場合は、my-demo-environmentを含む [名前] を持つインスタンスを選択します。
5. [アクション] メニューで、[セキュリティ]、[IAM ロールの変更] の順に選択します。

 Note

インスタンスにロールをアタッチしていても、ロールにはインスタンスプロファイルが含まれています。

6. [IAM ロールの変更] ページの [IAM ロール] で、確認したロール、または前の手順で作成したロールの名前を選択し、[適用] を選択します。
7. 環境に戻り、AWS CLI を使用して `aws configure` コマンドを実行するか、AWS CloudShell を使用して `configure` コマンドを実行します。[AWS アクセスキー ID] または [AWS シークレットアクセスキー] には値を指定しないでください (これらの各プロンプトの後に Enter を押します)。デフォルトリージョン名には、最も AWS リージョン 近い または AWS リソースがあるリージョンを指定します。たとえば、米国東部 (オハイオ) リージョンの場合は、us-east-2 を使用します。リージョンのリストについては、「Amazon Web Services 全般のリファレンス」の

「[AWS リージョン とエンドポイント](#)」を参照してください。オプションで、[Default output format (デフォルト出力形式)] の値を指定します (json など)。

環境 AWS のサービス から の呼び出しを開始できるようになりました。AWS CLI、aws-shellまたはその両方を使用して呼び出すには AWS のサービス、[AWS CLI 「」](#) および [aws-shell サンプル](#)」を参照してください。コードから AWS のサービス を呼び出すには、他の[チュートリアルとサンプル](#)を参照してください。

を使用してインスタンスプロファイルをインスタンスにアタッチする AWS CLI

 Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用してこのセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できますAWS CLI。

1. Amazon EC2 associate-iam-instance-profile コマンドを実行します。インスタンスプロファイルの名前と、環境の Amazon EC2 インスタンスの ID と AWS リージョン ID を指定します。

```
aws ec2 associate-iam-instance-profile --iam-instance-profile Name=my-demo-cloud9-instance-profile --region us-east-2 --instance-id i-12a3b45678cdef9a0
```

上記のコマンドでは、us-east-2 をインスタンスの AWS リージョン ID で置き換え、i-12a3b45678cdef9a0 をインスタンス ID で置き換えます。

インスタンスの ID を取得するには、例えば、Amazon EC2 describe-instances コマンドを実行して環境の名前と AWS リージョン ID を指定します。

```
aws ec2 describe-instances --region us-east-2 --filters Name=tag:Name,Values=*my-environment* --query "Reservations[*].Instances[*].InstanceId" --output text
```

上記のコマンドでは、us-east-2 をインスタンスの AWS リージョン ID で置き換え、my-environment を環境の名前で置き換えます。

2. 環境に戻り、AWS CLI を使用して `aws configure` コマンドを実行するか、`aws-shell` を使用して `configure` コマンドを実行します。[AWS Access Key ID] (アクセスキー ID) または [AWS Secret Access Key] (シークレットアクセスキー) には値を指定しないでください。これらの各プロンプトが表示されたら、Enter を押します。デフォルトリージョン名には、最も近い または AWS リソースがあるリージョンを指定します AWS リージョン。たとえば、米国東部 (オハイオリージョン) の場合は、`us-east-2` を使用します。リージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「[AWS とエンドポイント](#)」を参照してください。オプションで、[Default output format (デフォルト出力形式)] の値を指定します (`json` など)。

環境 AWS のサービス からの呼び出しを開始できるようになりました。AWS CLI、`aws-shell` またはその両方を使用して呼び出すには AWS のサービス、[AWS CLI 「」](#) および [「aws-shell サンプル」](#) を参照してください。コードから AWS のサービスを呼び出すには、他の[チュートリアルとサンプル](#)を参照してください。

環境に永続的アクセス認証情報を作成して保存する

Note

AWS Cloud9 EC2 開発環境を使用している場合は、AWS 永続的なアクセス認証情報ではなく、AWS マネージド一時認証情報を使用することをお勧めします。AWS マネージド一時認証情報を使用するには、「」を参照してください [AWS マネージド一時認証情報](#)。

このセクションでは、AWS Identity and Access Management (IAM) を使用して永続的な認証情報のセットを生成します。AWS CLI、`aws-shell`、またはコードは、を呼び出すときにこの認証情報のセットを使用できます AWS のサービス。このセットには、AWS アクセスキー ID とシー AWS クレットアクセスキーが含まれます。これは、のユーザーに固有のものです AWS アカウント。アクセスキー ID と AWS シークレットアクセスキーがすでに AWS ある場合は、これらの認証情報を書き留めてから、「[環境に永続的アクセス認証情報を保存する](#)」に進みます。

永続的認証情報のセットは、[IAM コンソール](#) または [AWS CLI](#) を使用して作成できます。

プログラマチックアクセス権を付与する

ユーザーが の AWS 外部とやり取りする場合は、プログラムによるアクセスが必要です AWS マネジメントコンソール。プログラムによるアクセスを許可する方法は、 がアクセスするユーザーのタイプによって異なります AWS。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	(推奨) コンソール認証情報を一時的な認証情報として使用して AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、AWS Command Line Interface 「ユーザーガイド」のAWS 「ローカル開発用ログイン」を参照してください。 • AWS SDKs 「SDK および ツールリファレンスガイド」のAWS 「ローカル開発用のログイン」を参照してください。AWS SDKs
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、AWS Command Line Interface ユーザーガイドの「を使用する AWS CLI 用に AWS IAM アイデンティティセンターを設定する」を参照してください。 • AWS SDKs、ツール、API AWS APIs 「SDK および ツールリファレンスガイド」の「IAM アイデンティティセンター認証」を参照してください。AWS SDKs

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	一時的な認証情報を使用して AWS CLI、AWS SDKs、または AWS APIs。	「IAM ユーザーガイド 」の「 AWS リソースでの一時的な認証情報の使用 」の手順に従います。
IAM	(非推奨) 長期認証情報を使用して、AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDKs 「SDK とツールリファレンスガイド」の「長期認証情報を使用した認証」を参照してください。AWS SDKs • API AWS APIs 「IAM ユーザーのアクセスキーの管理」を参照してください。

を使用して永続的なアクセス認証情報を作成する AWS CLI

Note

このセクションでは、で管理者レベルの認証情報 AWS CLI を使用して を設定することをお勧めします AWS アカウント。これができない場合は、AWS アカウント 管理者に確認してください。

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できますAWS CLI。

IAM create-access-key コマンドを実行して、ユーザーの新しい AWS アクセスキーと対応する AWS シークレットアクセスキーを作成します。

```
aws iam create-access-key --user-name MyUser
```

前述のコマンドでは、MyUser をユーザーの名前に置き換えます。

表示される AccessKeyId と SecretAccessKey の値を安全な場所に保存します。IAM create-access-key コマンドを実行した後、を使用して AWS CLI ユーザーの AWS シークレットアクセスキーを表示できるのは今回だけです。必要に応じて後でユーザーの新しい AWS シークレットアクセスキーを生成するには、IAM ユーザーガイドの「[アクセスキーの作成、変更、表示 \(API、CLI、PowerShell\)](#)」を参照してください。

環境に永続的アクセス認証情報を保存する

この手順では、IDE AWS Cloud9 を使用して永続的な AWS アクセス認証情報を環境に保存します。この手順では、で環境を既に作成し AWS Cloud9、環境を開き、ウェブブラウザに AWS Cloud9 IDE を表示していることを前提としています。詳細については、「[環境を作成する](#)」と「[環境を開く](#)」を参照してください。

Note

次の手順では、環境変数を使用して永続的なアクセス認証情報を保存する方法を示します。AWS CLI または が環境にaws-shellインストールされている場合は、代わりに の **aws configure** コマンド AWS CLI を使用するか、の **configure** コマンドを使用して永続的なアクセス認証情報aws-shellを保存できます。指示については、AWS Command Line Interface ユーザーガイドの「[クイック設定](#)」を参照してください。

1. 環境を開いた状態で、IDE AWS Cloud9 で新しいターミナルセッションがまだ開始されていない場合は、新しいターミナルセッションを開始します。新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。
2. 一度に 1 つのコマンドを実行して、永続的なアクセス認証情報を表すローカル環境変数を設定します。これらのコマンドで、の後に AWS アクセスキー ID `AWS_ACCESS_KEY_ID`: を入力します。の後に `AWS_SECRET_ACCESS_KEY`、AWS シークレットアクセスキーを入力します。の後に `AWS_DEFAULT_REGION_ID`、最も AWS リージョン 近い (または任意の) に関連付けられた AWS リージョン 識別子を入力します AWS リージョン。利用可能な識別子の一覧については、「Amazon Web Services 全般のリファレンス」の「[AWS リージョン とエンドポイント](#)」を参照してください。たとえば、米国東部 (オハイオ) の場合は、`us-east-2` を使用します。

```
export AWS_ACCESS_KEY_ID=  
export AWS_SECRET_ACCESS_KEY=  
export AWS_DEFAULT_REGION=
```

3. 前述の環境変数は、ターミナルセッションにおいてのみ有効になります。これらの環境変数をターミナルセッション間で利用できるようにするには、以下のように、それらをユーザー環境変数としてシェルスクリプトファイルに追加する必要があります。
 - a. IDE の [Environment (環境)] ウィンドウで歯車アイコンを選択し、[Show Home in Favorites (お気に入りのホームを表示する)] を選択します。このステップを繰り返して、[Show Hidden Files (隠しファイルを表示する)] も同様に選択します。
 - b. `~/.bashrc` ファイルを開きます。
 - c. ファイルの末尾に次のコードを入力するか貼り付けます。これらのコマンドで、の後に AWS アクセスキー ID `AWS_ACCESS_KEY_ID`: を入力します。の後に `AWS_SECRET_ACCESS_KEY`、AWS シークレットアクセスキーを入力します。の後に `AWS_DEFAULT_REGION_ID`、最も AWS リージョン 近い (または任意の) に関連付けられた識別子を入力します AWS リージョン。利用可能な識別子の一覧については、「Amazon Web Services 全般のリファレンス」の「[AWS リージョン とエンドポイント](#)」を参照してください。たとえば、米国東部 (オハイオ) の場合は、`us-east-2` を使用します。

```
export AWS_ACCESS_KEY_ID=  
export AWS_SECRET_ACCESS_KEY=  
export AWS_DEFAULT_REGION=
```

- d. ファイルを保存します。
- e. `~/.bashrc` ファイルにこれらの新しい環境変数をロードします。

```
. ~/.bashrc
```

環境 AWS のサービス から の呼び出しを開始できるようになりました。AWS CLI または を使用して aws-shell を呼び出すには AWS のサービス、[AWS CLI 「」](#) および [「aws-shell サンプル」](#) を参照してください。コードから AWS のサービス を呼び出すには、他の[チュートリアルとサンプル](#)を参照してください。

での環境設定の変更 AWS Cloud9

AWS Cloud9 開発環境のプリファレンスまたは設定を変更できます。

- [環境設定を変更する](#)
- [コンソールで環境設定を変更する](#)
- [コードを使用して環境設定を変更する](#)

環境優先順位を変更する

1. 設定を変更する環境を開きます。環境 を開くには、「[環境を開く](#)」を参照してください。
2. AWS Cloud9 IDE のメニューバーで、AWS Cloud9「設定」を選択します。
3. [Preferences (優先順位)] ウィンドウで、[Project Settings (プロジェクト設定)] を選択します。
4. 使用できるプロジェクト設定を希望に応じて変更します。[Code Editor (Ace) (コードエディタ (Ace))] や [Find in Files (ファイルの検索)] などの設定を含みます。

Note

詳細については、「[変更可能なプロジェクト設定](#)」を参照してください。

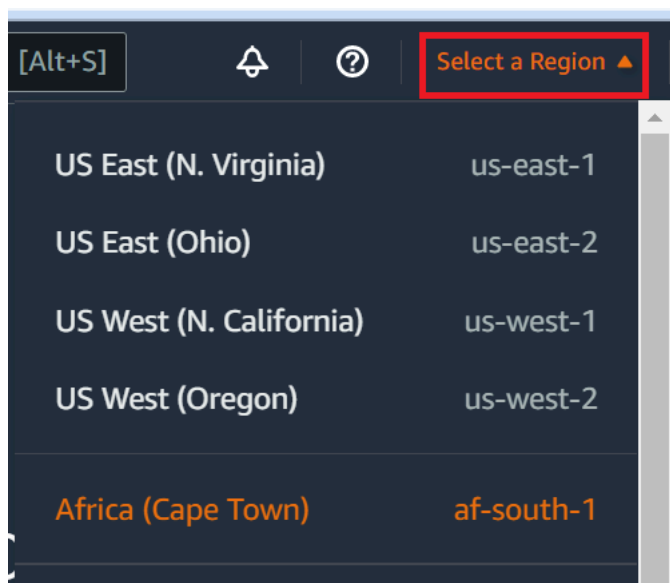
IDE AWS Cloud9 での環境のタイムアウトの調整

次のステップでは、IDE で Amazon EC2 AWS Cloud9 環境のタイムアウト期間を更新する方法の概要を示します。これは、環境が停止するまでの時間です。

1. 設定する環境を開きます。
2. [AWS Cloud9 IDE] のメニューバーで、[AWS Cloud9]、[設定] の順に選択します。
3. [設定] ウィンドウで、[Amazon EC2 インスタンス] セクションまでスクロールします。
4. 使用可能なリストからタイムアウト値を選択し、更新します。

コンソールで環境設定を変更する

1. 次のように AWS Cloud9 コンソールにサインインします。
 - を使用するののお客様だけであるか、単一の IAM ユーザー AWS アカウント である場合は AWS アカウント、<https://console.aws.amazon.com/cloud9/> にアクセスしてください。
 - 組織が を使用している場合は AWS IAM アイデンティティセンター、サインイン手順について AWS アカウント 管理者にお問い合わせください。
2. 上部のナビゲーションバーで、環境 AWS リージョン がある を選択します。



3. 環境のリストで変更したい設定の環境に対して、次のいずれかを実行します。
 - 環境のカードのタイトルを選択します。次のページで [View details] (詳細を表示) を選択します。
 - 環境のカードを選択し、[View details] (詳細を表示) ボタンを選択します。
4. 変更を行ってから、[Save changes (変更の保存)] を選択します。

AWS Cloud9 コンソールを使用して、次の設定を変更できます。

- EC2 環境の場合名前および説明。

- SSH 環境の場合: [Name (名前)]、[Description (説明)]、[User (ユーザー)]、[Host (ホスト)]、[Port (ポート)]、[Environment path (環境パス)]、[Node.js binary path (Node.js バイナリパス)]、および [SSH jump host (SSH ジャンプホスト)]。

その他の設定を変更するには、以下を実行します。

- EC2 環境に対して、次を行います。
 - [Type (タイプ)]、[Security groups (セキュリティグループ)]、[VPC]、[Subnet (サブネット)]、[Environment path (環境パス)]、または [Environment ARN (環境 ARN)] は変更できません。
 - [Permissions] (アクセス許可) または [Number of member] (メンバー数) については、「[環境メンバーのアクセスロールを変更する](#)」、「[自分のユーザーを削除する](#)」、「[IAM ユーザーを招待する](#)」、および「[別の環境メンバーを削除する](#)」を参照してください。
 - [EC2 instance type (EC2 インスタンスタイプ)]、[Memory (メモリ)]、または [vCPU] については、「[環境を移動またはサイズ変更する](#)」を参照してください。
- SSH 環境で、次を行います。
 - [Type (タイプ)] または [Environment ARN (環境 ARN)] は変更できません。
 - [許可] または [Number of members (メンバー数)] については、「[環境メンバーのアクセスロールを変更する](#)」、「[自分のユーザーを削除する](#)」、「[IAM ユーザーを招待する](#)」、および「[別の環境メンバーを削除する](#)」を参照してください。

環境がコンソールに表示されない場合は、次の 1 つ以上のアクションを試行して表示を試みてください。

- [Environments] (環境) ページのドロップダウンメニューバーで、以下の 1 つまたは複数を選択します。
 - 環境を選択して、選択した AWS リージョン および 内で AWS エンティティが所有するすべての環境を表示します AWS アカウント。
 - 選択した AWS リージョン と 内で AWS エンティティが招待されたすべての環境を表示するには、共有を選択します AWS アカウント。
 - すべてのアカウント環境を選択すると、選択した 内のすべての環境 AWS リージョン が表示され、AWS エンティティ AWS アカウント には表示するアクセス許可があります。
- 自分がメンバーである環境が [Shared with you] (自分と共有) リストに表示されていない場合は、環境の所有者に確認してください。
- 上部のナビゲーションバーで、別のナビゲーションバーを選択します AWS リージョン。

コードで環境設定を変更する

コードを使用して の環境設定を変更するには AWS Cloud9、次のように環境 AWS Cloud9 の更新オペレーションを呼び出します。

AWS CLI	update-environment
AWS SDK for C++	UpdateEnvironmentRequest 、 UpdateEnvironmentResult
AWS SDK for Go	UpdateEnvironment 、 UpdateEnvironmentRequest 、 UpdateEnvironmentWithContext
AWS SDK for Java	UpdateEnvironmentRequest 、 UpdateEnvironmentResult
AWS SDK for JavaScript	updateEnvironment
AWS SDK for .NET	UpdateEnvironmentRequest 、 UpdateEnvironmentResponse
AWS SDK for PHP	updateEnvironment
AWS SDK for Python (Boto)	update_environment
AWS SDK for Ruby	update_environment
AWS Tools for Windows PowerShell	Update-C9Environment
AWS Cloud9 API	UpdateEnvironment

での共有環境の使用 AWS Cloud9

共有環境は、複数のユーザーが参加するよう招待された AWS Cloud9 開発環境です。このトピックでは、 で環境を共有する手順 AWS Cloud9 と、共有環境に参加する方法について説明します。

所有する環境に参加するようユーザーを招待するには、次のいずれかの一連の手順に従います。招待したいユーザーの種類に基づいて選択します。

- 環境 AWS アカウント と同じ のユーザーの場合は、[環境と同じアカウントのユーザーを招待](#)する 必要があります。
- 環境 AWS アカウント と同じ の AWS Cloud9 管理者、特に AWS アカウント ルートユーザー、管理者ユーザー、または AWS 管理ポリシーがAWSCloud9Administratorアタッチされたユーザーである場合は、AWS Cloud9 管理者を自分で招待するか、[「環境と同じアカウントのユーザーを招待する」を参照するか](#)、AWS Cloud9 「管理者に環境と同じアカウントの管理者を招待する」または「他のユーザーを招待する AWS アカウント」を参照してください。 [AWS Cloud9](#)

共有環境のユースケース

共有環境は以下のユースケースに適しています。

- ペアプログラミング (ピアプログラミングともいいます): 単一の環境で 2 人のユーザーが一緒に同じコードに対して作業を行う場合です。ペアプログラミングでは通常、1 人のユーザーがコードを記述し、もう 1 人のユーザーがコードの記述を観察します。観察者はコードの記述者にその場でインプットおよびフィードバックを行います。2 人のポジションはプロジェクト中に頻繁に入れ替わります。共有環境がない場合、通常ペアプログラマーのチームが 1 台のマシンの前に座ります。同時にコードの書き込みができるのは 1 人のユーザーのみです。共有環境では、両方のユーザーが自分のマシンの前に座ることができます。さらに、たとえ物理的に異なるオフィスにいても、同時にコードの書き込みができます。
- コンピュータサイエンスクラス: これは、教師または教育助手が学生の環境にアクセスする場合に便利です。そうすることで、生徒の宿題を確認したり、彼らの環境の問題をリアルタイムで修正することができます。また、学生がクラスメートと一緒に共有の宿題プロジェクトで作業を行い、単一の環境でリアルタイムで一緒にコードを書くこともできます。それぞれが別の場所において、別のコンピューターオペレーティングシステムやウェブブラウザタイプを使用していても、これができます。
- その他、複数のユーザーが同じコードに対してリアルタイムで共同作業する必要がある状況。

環境メンバーのアクセスロールについて

で環境を共有したり、共有環境に参加する前に AWS Cloud9、共有環境のアクセス許可レベルを理解しておく必要があります。これらのアクセス許可レベルは環境メンバーアクセスロールと呼ばれます。

の共有環境 AWS Cloud9 には、所有者、読み取り/書き込み、読み取り専用の 3 つの環境メンバーアクセスロールがあります。

- 所有者は、環境に関する完全なコントロールを持っています。各環境の所有者は、環境の作成者である 1 人のみです。所有者は、以下のアクションができます。
- 環境のメンバーを追加、変更、および削除する
- ファイルを開く、表示、編集する
- コードを実行する
- 環境設定を変更する
- 他のメンバー とチャットする
- 既存のチャットメッセージを削除する

AWS Cloud9 IDE では、環境所有者が読み取り + 書き込みアクセスで表示されます。

- 読み取り/書き込みメンバーは以下を実行できます。
- ファイルを開く、表示、編集する
- コードを実行する
- IDE AWS Cloud9 内からさまざまな環境設定を変更する
- 他のメンバーとチャットする
- 既存のチャットメッセージを削除する

AWS Cloud9 IDE では、読み取り/書き込みメンバーは読み取り + 書き込みアクセスで表示されます。

- 読み取り専用メンバーは以下のアクションを実行できます。
- ファイルを開く、表示する
- 他のメンバーとチャットする
- 既存のチャットメッセージを削除する

AWS Cloud9 IDE では、読み取り専用メンバーは読み取り専用アクセスで表示されます。

ユーザーが環境所有者またはメンバーになる前に、ユーザーが次のいずれかの条件を満たしている必要があります。

- ユーザーが AWS アカウント ルートユーザーである。
- ユーザーが IAM 管理者ユーザーである。詳細については、IAM ユーザーガイドの「[最初の IAM 管理者ユーザーおよびグループの作成](#)」を参照してください。
- ユーザーは、IAM グループに属するユーザー、ロールを引き受けるユーザー、または ~~はロールを引き受けるフェデレーテッドユーザーであり、そのグループまたはロ~~

ルには AWS 管理ポリシー `AWSCloud9Administrator` または `AWSCloud9User` (または `AWSCloud9EnvironmentMember`、メンバーのみとなる) がアタッチされています。詳細については、「[AWS マネージド \(事前定義\) ポリシー](#)」を参照してください。

- 前述のマネージドポリシーのいずれかを IAM グループにアタッチするには、以下の手順で説明しているとおり、[AWS マネジメントコンソール](#) または [AWS コマンドラインインターフェイス \(AWS CLI\)](#) を使用します。
- ユーザーまたはフェデレーションユーザーが引き受ける前述のいずれかの管理ポリシーを使用して、IAM でロールを作成できます。詳細については、「IAM ユーザーガイド」の「[ロールの作成](#)」を参照してください。ユーザーまたはフェデレティッドユーザーがロールを引き受けるようにするには、IAM ユーザーガイドの「[IAM ロールを使用する](#)」を参照してください。

コンソールを使用して の AWS 管理ポリシー AWS Cloud9 をグループにアタッチする

次の手順では、コンソールを使用して の AWS 管理ポリシー AWS Cloud9 をグループにアタッチする方法について説明します。

1. まだサインインしていない場合は AWS マネジメントコンソール、 にサインインします。

このステップでは、AWS アカウントの IAM 管理者レベルの認証情報使用にサインインすることをお勧めします。これが実行できない場合は、AWS アカウント 管理者にお問い合わせください。

2. [IAM コンソール] を開きます。これを行うには、コンソールのナビゲーションバーで、[サービス] を選択します。次に、[IAM] を選択します。

3. [グループ] を選択します。

4. グループの名前を選択します。

5. [許可] タブの [マネージドポリシー] で、[ポリシーのアタッチ] を選択します。

6. ポリシー名のリストで、次のいずれかのボックスを選択します。

- [AWSCloud9User] (推奨) または [AWSCloud9Administrator]。グループの各ユーザーは環境の所有者になることができます。
- [AWSCloud9EnvironmentMember]。グループの各ユーザーはメンバーにのみなることができます。

(これらのポリシー名のいずれかがリストに表示されない場合は、[検索] ボックスにポリシー名を入力して表示させます。)

7. [Attach policy] (ポリシーのアタッチ) を選択します。

を使用して の AWS 管理ポリシー AWS Cloud9 をグループにアタッチする AWS CLI

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できますAWS CLI。

IAM attach-group-policy コマンドを実行して、 の AWS 管理ポリシー AWS Cloud9 をグループにアタッチします。ポリシーのグループ名と Amazon リソースネーム (ARN) を指定します。

```
aws iam attach-group-policy --group-name MyGroup --policy-arn arn:aws:iam::aws:policy/  
POLICY_NAME
```

前述のコマンドで、MyGroup をグループの名前に置き換えます。を次のいずれかの AWS マネージドポリシーの名前POLICY_NAMEに置き換えます。

- AWSCloud9User (推奨) または AWSCloud9Administrator。グループの各ユーザーは環境の所有者になることができます。
- AWSCloud9EnvironmentMember。グループの各ユーザーはメンバーにのみなることができます。

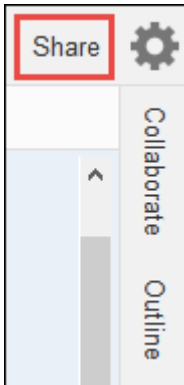
環境と同じアカウントのユーザーを招待する

このセクションの手順を使用して、で所有している AWS Cloud9 開発環境を同じアカウントの AWS アカウント ユーザーと共有します。

1. 招待したいユーザーが、以下のどのタイプのユーザーでもないとしします。招待したいユーザーが、対応する環境メンバーアクセスロールをすでに持っていることを確認してください。手順については、「[環境メンバーのアクセスロールについて](#)」を参照してください。

- AWS アカウント ルートユーザー。
- 管理者ユーザー。

- IAM グループに属するユーザー、ロールを引き受けるユーザー、またはロールを引き受けるフェデレーテッドユーザー。そのグループまたはロールには AWS 管理ポリシーがAWSCloud9Administratorアタッチされています。
2. お客様が所有しユーザーを招待する環境をまだ開いていない場合は、環境を開きます。
 3. IDE AWS Cloud9 のメニューバーで、次のいずれかを実行します。
 - [Window] (ウィンドウ)、[Share] (共有) の順に選択します。
 - [Share (共有)] ([Preferences (設定)] の歯車アイコンの横にあります) を選択します。



4. [Share this environment (この環境を共有)] ダイアログボックスの [Invite Members (メンバーの招待)] に、以下のいずれかを入力します。
 - IAM ユーザーを招待するには、ユーザーの名前を入力します。
 - AWS アカウント ルートユーザーを招待するには、`arn:aws:iam::123456789012:root` と入力します。を AWS アカウント ID 123456789012に置き換えます。
 - 役割を引き受けたユーザーまたは役割を引き受けたフェデレーションユーザーを招待するには、`arn:aws:sts::123456789012:assumed-role/MyAssumedRole/MyAssumedRoleSession` と入力します。を AWS アカウント ID 123456789012に置き換え、 を引き受けたロールの名前MyAssumedRoleに置き換えます。MyAssumedRoleSession を引き受けたロールのセッション名に置き換えます。
5. このユーザーを読み取り専用にするには、 [R] を選択します。このユーザーを読み取り/書き込み可能にするには、 [RW] です。
6. [招待] を選択します。

Note

このユーザーを読み取り/書き込みメンバーにすると、AWS セキュリティ認証情報を危険にさらす可能性のある情報を含むダイアログボックスが表示されます。以下の情報は、この問題の背景の詳細です。

環境を共有するのは信頼する相手のみにする必要があります。

読み取り/書き込みメンバーは、環境内の AWS CLI、AWS CloudShell、または AWS SDK コードを使用して、AWS ユーザーに代わってでアクションを実行できる場合があります。さらに、永続的な AWS アクセス認証情報を環境内に保存すると、そのメンバーはそれらの認証情報をコピーして環境外で使用する可能性があります。

環境から永続的な AWS アクセス認証情報を削除し、代わりに一時的な AWS アクセス認証情報を使用しても、この問題は完全には解決されません。これによりメンバーがこれらの一時的認証情報をコピーして環境外で利用できる機会は減少します (一時的認証情報は限られた時間のみ機能するため)。ただし、一時的な認証情報により、読み取り/書き込みメンバーがユーザーに代わって環境 AWS からでアクションを実行できるようになります。

7. ユーザーに、この環境を開いて使用できるようになったことを連絡します。

環境と同じアカウントの AWS Cloud9 管理者に自分や他のユーザーを招待させる

Note

[AWS マネージド一時認証情報](#)を使用している場合、IDE AWS Cloud9 のターミナルセッションを使用して、このセクションのコマンドの一部またはすべてを実行することはできません。AWS セキュリティのベストプラクティスに対処するために、AWS マネージド一時認証情報では一部のコマンドを実行できません。代わりに、これらのコマンドを AWS Command Line Interface () の別のインストールから実行できます AWS CLI。

以下のタイプのユーザーは、同じアカウントのどの環境にでも自分自身 (または同じ AWS アカウント内の他のユーザー) を招待できます。

- AWS アカウント ルートユーザー。
- 管理者ユーザー。
- IAM グループに属するユーザー、ロールを引き受けるユーザー、またはロールを引き受けるフェデレーティッドユーザー。そのグループまたはロールには AWS 管理ポリシーが `AWSCloud9Administrator` アタッチされています。

招待されたユーザーが前述のタイプのユーザーではないとします。ユーザーが、対応する環境メンバーアクセスロールをすでに持っていることを確認してください。手順については、「[環境メンバーのアクセスロールについて](#)」を参照してください。

ユーザーを招待するには、AWS CLI または AWS CloudShell を使用して `create-environment-membership` コマンドを実行します AWS Cloud9。

```
aws cloud9 create-environment-membership --environment-id
12a34567b8cd9012345ef67abcd890e1 --user-arn USER_ARN --permissions PERMISSION_LEVEL
```

前述のコマンドで、`12a34567b8cd9012345ef67abcd890e1` を環境の ID に置き換え、`PERMISSION_LEVEL` を `read-write` または `read-only` に置き換えます。また、`USER_ARN` は、以下のいずれかに置き換えます。

- IAM ユーザーを招待するには、`arn:aws:iam::123456789012:user/MyUser` と入力します。を AWS アカウント ID `123456789012` に置き換え、をユーザーの名前 `MyUser` に置き換えます。
- AWS アカウント ルートユーザーを招待するには、`arn:aws:iam::123456789012:root` と入力します。を AWS アカウント ID `123456789012` に置き換えます。
- 役割を引き受けたユーザーまたは役割を引き受けたフェデレーションユーザーを招待するには、`arn:aws:sts::123456789012:assumed-role/MyAssumedRole/MyAssumedRoleSession` と入力します。を AWS アカウント ID `123456789012` に置き換えます。`MyAssumedRole` を引き受けるロールの名前に置き換えます。また、`MyAssumedRoleSession` を引き受けるロールの名前に置き換えます。

たとえば、アカウント ID の AWS アカウント ルートユーザー `123456789012` を ID を持つ環境に読み取り/書き込みメンバー `12a34567b8cd9012345ef67abcd890e1` として招待するには、次のコマンドを実行します。

```
aws cloud9 create-environment-membership --environment-id
12a34567b8cd9012345ef67abcd890e1 --user-arn arn:aws:iam::123456789012:root --
permissions read-write
```

Note

を使用している場合は AWS CloudShell、前述のコマンドの `aws` プレフィックスを省略します。

共有環境を開く

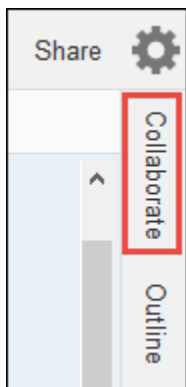
共有環境を開くには、AWS Cloud9 ダッシュボードを使用できます。AWS Cloud9 IDE を使用してアクションを実行し、共有環境で作業を完了します。例えば、ファイルを操作したり、他のチームメンバーとチャットしたりします。

1. 対応するアクセスポリシーが、ユーザーのグループまたはロールにアタッチされていることを確認してください。詳細については、「[環境メンバーのアクセスロールについて](#)」を参照してください。
2. 次のように AWS Cloud9 コンソールにサインインします。
 - を使用するののお客様だけであるか、単一の IAM ユーザー AWS アカウント である場合は AWS アカウント、<https://console.aws.amazon.com/cloud9/> にアクセスしてください。
 - 組織が IAM Identity Center を使用している場合は、サインイン手順について AWS アカウント管理者にお問い合わせください。
 - 教室内の学生である場合は、インストラクターにサインインの手順をお問い合わせください。
3. AWS Cloud9 ダッシュボードから共有環境を開きます。詳細については、「[環境を開く AWS Cloud9](#)」を参照してください。

このトピックの残りの部分で説明するように、[Collaborate (コラボレーション)] ウィンドウを使用して他のメンバーとやり取りできます。

Note

[Collaborate] (コラボレーション) ウィンドウが表示されていない場合は、[Collaborate] (コラボレーション)を選択します。[Collaborate (コラボレーション)] ボタンが非表示になっている場合は、メニューバーで [Window (ウィンドウ)、Collaborate (コラボレーション)] を選択します。

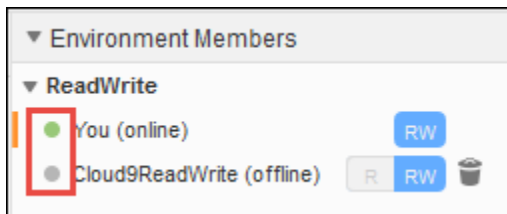


環境メンバーのリストを参照する

共有環境が開いている状態で、[Collaborate (コラボレーション)] ウィンドウで、メンバーが表示されていない場合は、[Environment Members (環境メンバー)] を展開します。

各メンバーの横にある円は、オンラインステータスを次のように示します。

- アクティブなメンバーには緑色の円が付いています。
- オフラインのメンバーには灰色の円が付いています。
- アイドルなメンバーにはオレンジ色の円が付いています。



コードを使用して環境メンバーのリストを取得するには、次のように環境メンバーシップの AWS Cloud9 記述オペレーションを呼び出します。

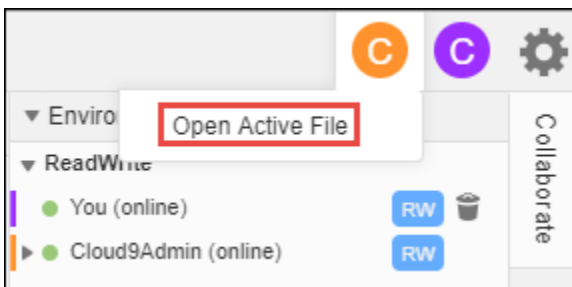
AWS CLI	describe-environment-memberships
AWS SDK for C++	DescribeEnvironmentMembershipsRequest 、 DescribeEnvironmentMembershipsResult
AWS SDK for Go	DescribeEnvironmentMemberships 、 DescribeEnvironmentMembershipsRequest 、 DescribeEnvironmentMembershipsWithContext

AWS SDK for Java	DescribeEnvironmentMembershipsRequest 、 DescribeEnvironmentMembershipsResult
AWS SDK for JavaScript	describeEnvironmentMemberships
AWS SDK for .NET	DescribeEnvironmentMembershipsRequest 、 DescribeEnvironmentMembershipsResponse
AWS SDK for PHP	describeEnvironmentMemberships
AWS SDK for Python (Boto)	describe_environment_memberships
AWS SDK for Ruby	describe_environment_memberships
AWS Tools for Windows PowerShell	Get-C9EnvironmentMembershipList
AWS Cloud9 API	DescribeEnvironmentMemberships

環境メンバーのアクティブなファイルを開く

このステップでは、環境メンバーのアクティブなファイルを開く方法を示します。

共有環境が開いている状態で、メニューバーでメンバー名を選択します。次に、[Open Active File] (アクティブなファイルを開く) を選択します。

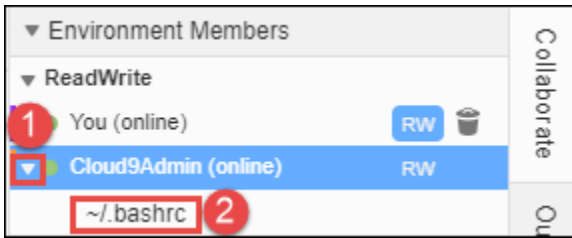


環境メンバーが開いているファイルを開く

このステップでは、環境メンバーが開いているファイルを開く方法を示します。

- 共有環境が開いている状態で、[Collaborate (コラボレーション)] ウィンドウで、メンバーが表示されていない場合は、[Environment Members (環境メンバー)] を展開します。

2. 環境で開きたいファイルを開いているユーザーの名前を展開します。
3. 開くファイルの名前を開きます (ダブルクリック)。



環境メンバーのアクティブなカーソルに移動する

このステップでは、環境メンバーのアクティブなカーソルを操作する方法を示します。

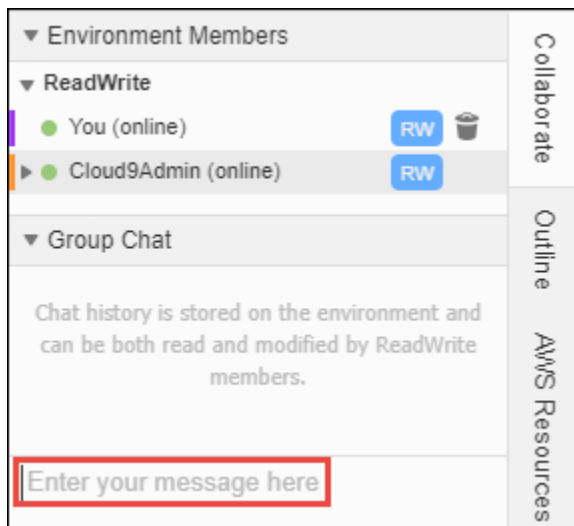
1. 共有環境が開いている状態で、[Collaborate (コラボレーション)] ウィンドウで、メンバーが表示されていない場合は、[Environment Members (環境メンバー)] を展開します。
2. ローカルサーバー名のコンテキスト (右クリック) メニューを開き、[Show Location] (場所を表示する) を選択します。

共有環境でのチャットの管理

このトピックでは、他の環境メンバーとチャットを使用する方法、共有環境でチャットメッセージを表示する方法、共有環境から削除する方法を示します。

他の環境メンバーとチャットする

共有環境を開いている状態で、[Collaborate (コラボレーション)] ウィンドウの下部にある [Enter your message here (ここにメッセージを入力してください)] にチャットメッセージを入力し、Enter を押します。



共有環境でチャットメッセージを表示する

共有環境が開いている状態で、[Collaborate] (コラボレーション) ウィンドウで、チャットメッセージが表示されていない場合は、[Group Chat] (グループチャット) を展開します。

共有環境からチャットメッセージを削除する

共有環境が開いている状態で、[Collaborate] (コラボレーション) ウィンドウで、[Group Chat] (グループチャット) のチャットメッセージをコンテキスト (右クリック) メニューを開きます。[Delete Message] (メッセージの削除) を選択します。

Note

チャットメッセージを削除すると、すべてのメンバーの環境から削除されます。

共有環境からすべてのチャットメッセージを削除する

共有環境が開いている状態で、[Collaborate] (コラボレーション) ウィンドウで、[Group Chat] (グループチャット) の任意の場所でコンテキスト (右クリック) メニューを開きます。次に、[Clear history] (履歴をクリア) を選択します。

Note

すべてのチャットメッセージを削除すると、すべてのメンバーの環境から削除されます。

環境メンバーのアクセスロールを変更する

このステップでは、環境メンバーのアクセスロールを変更する方法を示します。コードを使用してアクセスロールを変更し、AWS Cloud9 環境メンバーシップを更新することもできます。

1. アクセスロールを変更したいメンバーが含まれている自己所有の環境を開きます (環境 がまだ開いていない場合)。詳細については、「[環境を開く AWS Cloud9](#)」を参照してください。
2. メンバーのリストが表示されていない場合は、[Collaborate] (コラボレーション) ウィンドウで、[Environment Members] (環境メンバー) を展開します。
3. 次のいずれかのアクションを実行します。
 - アクセスロールを変更したいメンバーの名前の横にある [R] または [RW] を選択して、このメンバーをそれぞれ所有者または読み取り/書き込みメンバーにします。
 - 読み取り/書き込みメンバーを読み取り専用メンバーに変更するには、メンバー名のコンテキストメニュー (右クリック) を開き、[Revoke Write Access] (書き込みアクセス権を取り消す) を選択します。
 - 読み取り専用メンバーを読み取り/書き込みメンバーに変更するには、メンバー名のコンテキストメニュー (右クリック) を開き、[Grant Read+Write Access] (書き込み+読み取りアクセス権を付与する) を選択します。

Note

このユーザーを読み取り/書き込みメンバーにすると、AWS セキュリティ認証情報を危険にさらす可能性のある情報を含むダイアログボックスが表示されます。そのユーザーが AWS ユーザーに代わってアクションを実行することを信頼しない限り、ユーザーを読み取り/書き込みメンバーにしないでください。詳細については、「[環境と同じアカウントのユーザーを招待する](#)」の関連する注記を参照してください。

コードを使用して環境メンバーのアクセスロールを変更するには、次のように環境メンバーシップ AWS Cloud9 の更新オペレーションを呼び出します。

AWS CLI

[update-environment-membership](#)

AWS SDK for C++

[UpdateEnvironmentMembershipRequest](#)、[UpdateEnvironmentMembershipResult](#)

AWS SDK for Go	UpdateEnvironmentMembership 、 UpdateEnvironmentMembershipRequest 、 UpdateEnvironmentMembershipWithContext
AWS SDK for Java	UpdateEnvironmentMembershipRequest 、 UpdateEnvironmentMembershipResult
AWS SDK for JavaScript	updateEnvironmentMembership
AWS SDK for .NET	UpdateEnvironmentMembershipRequest 、 UpdateEnvironmentMembershipResponse
AWS SDK for PHP	updateEnvironmentMembership
AWS SDK for Python (Boto)	update_environment_membership
AWS SDK for Ruby	update_environment_membership
AWS Tools for Windows PowerShell	Update-C9EnvironmentMembership
AWS Cloud9 API	UpdateEnvironmentMembership

共有環境から自分のユーザーを削除する

このステップでは、共有環境からユーザーを削除する方法を示します。

Note

環境所有者の場合は、環境から自分のユーザーを削除することはできません。
自分のユーザーを環境から削除しても、IAM からは削除されません。

- 共有環境が開いている状態で、[Collaborate (コラボレーション)] ウィンドウで、メンバーが表示されていない場合は、[Environment Members (環境メンバー)] を展開します。
- 次のいずれかのアクションを実行します。
 - [You (自分)] の横で、ゴミ箱アイコンを選択します。

- [You] (あなた) のコンテキスト (右クリック) メニューを開き、[Leave environment] (環境を離れる) を選択します。

3. プロンプトが表示されたら、[Leave (離れる)] を選択します。

コードを使用して共有環境からユーザーを削除するには、次のように環境メンバーシップ AWS Cloud9 の削除オペレーションを呼び出します。

AWS CLI	delete-environment-membership
AWS SDK for C++	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResult
AWS SDK for Go	DeleteEnvironmentMembership 、 DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipWithContext
AWS SDK for Java	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResult
AWS SDK for JavaScript	deleteEnvironmentMembership
AWS SDK for .NET	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResponse
AWS SDK for PHP	deleteEnvironmentMembership
AWS SDK for Python (Boto)	delete_environment_membership
AWS SDK for Ruby	delete_environment_membership
AWS Tools for Windows PowerShell	Remove-C9EnvironmentMembership
AWS Cloud9 API	DeleteEnvironmentMembership

別の環境メンバーを削除する

このステップでは、ユーザー以外のメンバーを環境から削除する方法を示します。

Note

自分のユーザー以外のメンバーを環境から削除するには、環境所有者の認証情報を使用して AWS Cloud9 にサインインする必要があります。
メンバーを削除しても、そのユーザーは IAM から削除されません。

1. 削除したいメンバーが含まれる環境をまだ開いていない場合は、環境を開きます。詳細については、「[AWS Cloud9で環境を開く](#)」を参照してください。
2. [Collaborate (コラボレーション)] ウィンドウで、メンバーのリストが表示されていない場合は、[Environment Members (環境メンバー)] を展開します。
3. 次のいずれかを行います。
 - 削除したいメンバーの名前の横にあるごみ箱アイコンを選択します。
 - 削除するメンバー名のコンテキスト (右クリック) メニューを開き、[Revoke Access] (アクセスを取り消す) を選択します。
4. プロンプトが表示されたら、[Remove Member (メンバーの削除)] を選択します。

コードを使用して環境からメンバーを削除するには、次のように環境メンバーシップ AWS Cloud9 の削除オペレーションを呼び出します。

AWS CLI	delete-environment-membership
AWS SDK for C++	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResult
AWS SDK for Go	DeleteEnvironmentMembership 、 DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipWithContext

AWS SDK for Java	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResult
AWS SDK for JavaScript	deleteEnvironmentMembership
AWS SDK for .NET	DeleteEnvironmentMembershipRequest 、 DeleteEnvironmentMembershipResponse
AWS SDK for PHP	deleteEnvironmentMembership
AWS SDK for Python (Boto)	delete_environment_membership
AWS SDK for Ruby	delete_environment_membership
AWS Tools for Windows PowerShell	Remove-C9EnvironmentMembership
AWS Cloud9 API	DeleteEnvironmentMembership

環境共有のベストプラクティス

環境を共有する場合には、以下のプラクティスをお勧めします。

- 信頼できる読み取り/書き込みメンバーのみを環境に招待します。
- EC2 環境の場合、読み取り/書き込みメンバーは環境所有者の AWS アクセス認証情報を使用して環境から呼び出すことができます AWS のサービス。これは彼ら自身の認証情報の代わりです。これを防ぐために、環境所有者は環境の AWS マネージド一時認証情報を無効にすることができます。ただし、これを行うと環境所有者も呼び出しを行うことができなくなります。詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。
- をオンに AWS CloudTrail して、環境内のアクティビティを追跡します。詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。
- AWS アカウント ルートユーザーを使用して環境を作成および共有しないでください。代わりに、アカウント内の IAM ユーザーを使用します。詳細については、IAM ユーザーガイドの「[初回アクセスのみ: ルートユーザーの認証情報](#)」および「[IAM ユーザー](#)」を参照してください。

Amazon EBS AWS Cloud9 ボリュームからの IDE の移動

AWS Cloud9 開発環境は、ある Amazon EC2 インスタンスから別のインスタンスに移動できます。たとえば、次のようなアクションを実行したいかもしれません。

- 正常なインスタンスと比較して、障害がある、または予期しない方法で実行されている Amazon EC2 インスタンスから環境を移転します。
- 古いインスタンスから最新のシステム更新が適用されているインスタンスに環境を移転します。
- 環境が現在のインスタンスに対して使用率が高すぎるまたは低すぎるため、インスタンスのコンピューティングリソースを増やします。

プロジェクトファイルを保持したまま、新しい AWS Cloud9 EC2 環境に移行することで、AWS Cloud9 サポートされている AMI から別の AMI にアップグレードできます。別のバージョンの AMI にアップグレードする理由は以下の通りです。

- 現在の環境の AMI がサポート終了となり、サポート対象ではなくなった。
- 必要なパッケージが、現在の AMI では古くなっている。

環境の Amazon EC2 インスタンスに関連付けられている Amazon Elastic Block Store (Amazon EBS) ボリュームのサイズ変更もできます。たとえば、次のアクションの一方または両方を実行したいかもしれません。

- インスタンスのストレージ領域が不足している場合、ボリュームのサイズを増やします。
- 使用していない余分なストレージ領域の支払いを避けるために、ボリュームのサイズを減らします。

環境を移動またはサイズ変更する前に、環境で実行中のプロセスの停止または環境へのスワップファイルの追加を試すことができます。メモリ不足や CPU 使用率が高い場合の対処方法の詳細については、[トラブルシューティング](#)を参照してください。

Note

このトピックでは、Amazon EC2 インスタンス間の環境移動または Amazon EBS ボリュームのサイズ変更についてのみ説明します。自社サーバーの 1 つの環境をサイズ変更したり、

自社サーバーのいずれかのストレージ領域を変更する場合は、サーバーのドキュメンテーションを参照してください。

最後に、Amazon EBS リソースを暗号化して、インスタンスと添付の EBS ストレージ間で保管中のデータと転送中のデータの両方のセキュリティを調整できます。

環境の移動

移動プロセスをスタートする前に、以下の条件に注意してください。

- 環境を同じタイプの Amazon EC2 インスタンスに移動することはできません。移動する場合は、新しいインスタンスとして別の Amazon EC2 インスタンスタイプを選択する必要があります。

Important

環境を別の Amazon EC2 インスタンスタイプに移動する場合、そのインスタンスタイプは現在の AWS Cloud9 でサポートされる必要があります AWS リージョン。各リージョンごとに使用できるインスタンスタイプをチェックするには、[コンソールを使用して EC2 環境の作成](#)している時は表示されている [[Configure settings] (設定の構成) に移動します。インスタンスタイプセクションの選択は、コンソールの右上で AWS リージョン 選択されている によって決まります。

- インスタンスタイプを変更する前に、環境に関連付けられている Amazon EC2 インスタンスを停止する必要があります。インスタンスが停止している間、お客様およびメンバーは、停止したインスタンスに関連付けられた環境を使用することはできません。
- AWS はインスタンスを新しいハードウェアに移動しますが、インスタンスの ID は変更されません。
- インスタンスが Amazon VPC で実行されており、パブリック IPv4 アドレスがある場合、はアドレスを AWS 解放し、インスタンスに新しいパブリック IPv4 アドレスを付与します。インスタンスは、プライベート IPv4 アドレスおよび Elastic IP アドレスまたは IPv6 アドレスを保持します。
- インスタンスが停止している間のダウンタイムを計画しておいてください。このプロセスには数分かかることがあります。

環境を移動するには

1. (オプション) 新しいインスタンスタイプに既存のインスタンス上でインストールされていないドライバが必要な場合は、インスタンスに接続して、そのドライバをインストールします。詳細については、[Amazon EC2 ユーザーガイド](#)の「インスタンスのサイズ変更の互換性」を参照してください。
2. 現在 環境を表示しているすべてのウェブブラウザのタブを閉じます。

Important

現在環境を表示しているウェブブラウザタブをすべて閉じない場合、この手順の完了が妨げられる AWS Cloud9 可能性があります。具体的に AWS Cloud9 は、この手順中に間違ったタイミングで環境に関連付けられている Amazon EC2 インスタンスを再起動しようとする可能性があります。この手順を完了するまで、インスタンスは停止したままである必要があります。

3. まだサインインしていない場合は AWS マネジメントコンソール、<https://console.aws.amazon.com> でサインインします。

で管理者レベルの認証情報を使用してサインインすることをお勧めします AWS アカウント。これができない場合は、AWS アカウント 管理者に確認してください。

4. Amazon EC2 コンソールを開きます。そのためには、[サービス] リストで [EC2] を選択します。
5. AWS ナビゲーションバーで、移動する環境 AWS リージョン (米国東部 (オハイオ) など) を含むを選択します。
6. サービスナビゲーションペインで [インスタンス] を展開して、その後、[インスタンス] を選択します。
7. インスタンスのリストで、移動したい環境に関連付けられているインスタンスを選択します。EC2 環境の場合、インスタンス名はaws-cloud9- で始まり、環境名が続きます。例えば、環境名が my-demo-environment の場合、インスタンス名は aws-cloud9-my-demo-environment で始まります。
8. [インスタンスの状態] が [停止] でない場合は、[アクション]、[インスタンスの状態]、[停止] を選択します。プロンプトが表示されたら、[停止する] を選択します。インスタンスが停止するまで、数分かかる場合があります。
9. [インスタンスの状態] が停止すると、インスタンスを選択したままで、[アクション]、[インスタンスの設定]、[インスタンスタイプの変更] を選択します。

10. [インスタンスタイプの変更] ダイアログボックスで、使用する環境に新しい [インスタンスタイプ] を選択します。

 Note

インスタンスタイプがリストに表示されない場合は、そのインスタンスの設定と互換性がありません。例えば、仮想化のタイプが原因で、インスタンスに互換性がない場合があります。

11. (オプション) 選択したインスタンスタイプが EBS 最適化をサポートしている場合は、[EBS 最適化] を選択して EBS 最適化を有効にするか、[EBS 最適化] を選択解除して EBS 最適化を無効にします。

 Note

選択したインスタンスタイプがデフォルトで EBS 最適化される場合、[EBS 最適化] が選択され、この選択を解除することはできません。

12. [Apply] を選択して、新しい設定を受け入れます。

 Note

この手順で先ほど [インスタンスタイプ] に別のインスタンスタイプを選択しなかった場合は、[適用] を選択しても何も起こりません。

13. 環境を再び開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

前の手順の詳細については、Amazon EC2 ユーザーガイドの「[インスタンスタイプを変更する](#)」を参照してください。

AWS Cloud9 EC2 環境を別の Amazon マシンイメージ (AMI) に移動する

このトピックでは、AWS Cloud9 EC2 環境を 1 つの Amazon Linux AMI から別の AWS Cloud9 サポートされている AMI に移行する方法について説明します。

Note

OS バージョンを更新せずに環境を新しいインスタンスに移動する場合は、「[the section called “環境の移動”](#)」を参照してください。

以下のいずれかの手順を使用して、環境間でデータを移行できます。

アーカイブをローカルマシンにダウンロードして環境を移動するには

- 異なるベースイメージを使用して、同じアベイラビリティゾーンに新しい環境を作成します。
 - [the section called “EC2 環境を作成する”](#) セクションのステップを完了して、新しい環境を作成します。

Note

[プラットフォーム] を選択する際は、環境を移行するプラットフォームを選択します。

- デフォルトでは、環境は 10 GiB ボリュームで作成されます。アーカイブを新しい環境にアップロードまたは展開するのに十分なスペースがない場合は、[the section called “Amazon EBS ボリュームのサイズ変更”](#) 手順のステップを実行して Amazon EBS ボリュームサイズを変更します。
- IDE AWS Cloud9 で移行する環境を開きます。
 - IDE AWS Cloud9 がロードされたら、メニューから File > Download project を選択して、環境プロジェクトディレクトリの内容を含むアーカイブをダウンロードします。
 - 新しい環境で AWS Cloud9 IDE を開きます。
 - [ファイル] > [ローカルファイルをアップロード...] を選択してアーカイブをアップロードします。
 - (オプション) 古い .c9 ディレクトリを .c9.backup にバックアップするには、環境ターミナルで次のコマンドを実行します。

```
cp .c9 .c9.backup
```

後で設定ファイルを復元する場合は、これらのバックアップファイルが必要になる場合があります。

7. 次のコマンドを実行して、このパッケージを展開します。

```
tar xzvf <old_environment_name>.tar.gz -C ~/
```

8. プロジェクトディレクトリからアーカイブを削除するには、次のコマンドを実行します。

```
rm <old_environment_name>.tar.gz
```

新しい環境が予期したとおりに動作していることを確認します。

9. 古い環境を削除できるようになりました。

Amazon EBS ボリュームを使用して環境を移動するには

アーカイブをダウンロードできない場合、またはダウンロードしたアーカイブが大きすぎる場合は、Amazon EBS ボリュームを使用して移行できます。また、この方法によって、~/environment ディレクトリの外部にあるファイルをコピーできます。

1. 既存の環境で開いているすべての AWS Cloud9 IDE タブを閉じます。
2. 既存のインスタンスを停止するには、次のステップを実行します。
 - a. AWS Cloud9 コンソールで、環境を選択して移動し、詳細を表示します。
 - b. [環境の詳細] ページの [EC2 インスタンス] タブで、[EC2 インスタンスの管理] を選択します。
 - c. EC2 コンソールで、インスタンスを選択してインスタンスの詳細に移動します。
 - d. [インスタンスの状態] が [停止] に設定されていることを確認します。そうでない場合は、[インスタンス状態] ドロップダウンリストから [インスタンスの停止] を選択します。プロンプトが表示されたら、[停止] を選択します。インスタンスが停止するまで、数分かかる場合があります。
3. 異なるベースイメージを使用して、同じアベイラビリティーゾーンに新しい環境を作成します。
 - a. [the section called “EC2 環境を作成する”](#) セクションのステップを完了して、新しい環境を作成します。

 Note

[プラットフォーム] を選択する際は、環境を移行するプラットフォームを選択します。

- b. デフォルトでは、環境は 10 GiB ボリュームで作成されます。ソースボリュームから新しい環境へファイルを移動するのに十分な容量がない場合は、[the section called “Amazon EBS ボリュームのサイズ変更”](#) 手順のステップを完了して Amazon EBS ボリュームサイズを変更します。
4. ボリュームを既存のインスタンスからデタッチするには、以下の手順を実行します。
 - a. [インスタンスの概要] ページで、[ストレージ] タブを選択し、ボリュームを選択します。選択したボリュームのデバイス名は、[ルートデバイスの詳細] セクションの [ルートデバイス名] で指定されているものと同じである必要があります。
 - b. ボリュームの詳細ページで、[アクション] > [ボリュームのデタッチ] を選択します。
 - c. ボリュームが正常にデタッチされたら、[アクション] > [ボリュームのアタッチ] を選択し、ドロップダウンリストから新しい環境のインスタンスを検索して選択します。選択した Amazon EC2 インスタンスの名前には、というプレフィックスが付いた AWS Cloud9 環境名が含まれている必要があります aws-cloud9。
5. 新しい環境で AWS Cloud9 IDE を開きます。
6. 環境がロードされたら、新しくアタッチされたボリュームのデバイスを識別するために、ターミナルで次のコマンドを実行します。

```
lsblk
```

以下の出力例では、ルートデバイス nvme0n1p1 のパーティション nvme0n1 がすでにマウントされているため、nvme1n1p1 パーティションもマウントする必要があります。デバイスのフルパスは /dev/nvme1n1p1 です。

```
Admin:~/environment $ lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
nvme0n1              259:0    0  10G  0 disk
##nvme0n1p1         259:2    0  10G  0 part /
##nvme0n1p127       259:3    0   1M  0 part
##nvme0n1p128       259:4    0  10M  0 part /boot/efi
nvme1n1              259:1    0  10G  0 disk
##nvme1n1p1         259:5    0  10G  0 part
##nvme1n1p128       259:6    0   1M  0 part
```

 Note

出力は、ターミナルでこのコマンドを実行する場合は異なります。

7. 環境ターミナルで以下の手順を実行して、既存のボリュームをマウントします。

- a. ボリュームのパーティションがマウントされる一時ディレクトリを作成するには、次のコマンドを実行します。

```
MOUNT_POINT=$(mktemp -d)
```

- b. `lsblk` コマンドの出力例に基づいて、マウントするデバイスの次のパスを指定します。

```
MOUNT_DEVICE=/dev/nvme1n1p1
```

 Note

出力は、ターミナルでこのコマンドを実行する場合は異なります。

- c. 既存のボリュームをマウントするには、次のコマンドを実行します。

```
sudo mount $MOUNT_DEVICE $MOUNT_POINT
```

- d. 既存のボリュームが正常にマウントされているかどうかを確認するには、以下の手順を実行します。

- i. ボリュームが出力に含まれていることを確認するには、次のコマンドを実行します。

```
df -h
```

- ii. ボリュームの内容を確認するには、次のコマンドを実行します。

```
ls $MOUNT_POINT/home/ec2-user/environment/
```

8. (オプション) 古い `.c9` ディレクトリを `.c9.backup` にバックアップするには、環境ターミナルで次のコマンドを実行します。

```
cp .c9 .c9.backup
```


後で設定ファイルを復元する場合は、これらのバックアップファイルが必要になる場合があります。

9. 既存のボリュームから古い環境をコピーするには、次のコマンドを実行します。

```
cp -R $MOUNT_POINT/home/ec2-user/environment ~
```

 Note

必要に応じて、前述のコマンドを使用して環境ディレクトリの外部にあるファイルやディレクトリをコピーすることもできます。

新しい環境が予期したとおりに動作していることを確認します。

10. 前のデバイスのマウントを解除するには、次の 2 つのコマンドのいずれかを実行します。

```
sudo umount $MOUNT_DEVICE
```

```
sudo umount $MOUNT_POINT
```

11. [アクション] ドロップダウンリストから [ボリュームのデタッチ] を選択して、ステップ 3 でアタッチしたボリュームをデタッチします。
12. 古い環境とそのボリュームを削除できるようになりました。

 Note

ボリュームは環境の Amazon EC2 インスタンスにアタッチされていないため、手動で削除する必要があります。これを実行するには、[ボリュームの詳細] ページで [削除] を選択します。

環境で使用されている Amazon EBS ボリュームのサイズ変更

このステップでは、Amazon EBS ボリュームのサイズを変更する方法を説明します。

1. サイズを変更したい Amazon EBS ボリュームの Amazon EC2 インスタンスに関連付けられている環境を開きます。

2. 環境の AWS Cloud9 IDE で、次の内容のファイルを作成し、そのファイルを 拡張子 `.sh` (例:) で保存します `resize.sh`。

メモ

このスクリプトは、AL2023、Amazon Linux 2、Amazon Linux、または Ubuntu サーバーを実行する EC2 インスタンスに接続した Amazon EBS ボリュームに対して有効であり、IMDSv2 を使用するように設定されています。

このスクリプトは、Nitro ベースのインスタンスで NVMe ブロックデバイスとして公開される Amazon EBS ボリュームのサイズも変更します。Nitro システムをベースにしたインスタンスリストについては、「Amazon EC2 ユーザーガイド」の「[Nitro ベースのインスタンス](#)」を参照してください。

```
#!/bin/bash

# Specify the desired volume size in GiB as a command line argument. If not
# specified, default to 20 GiB.
SIZE=${1:-20}

# Get the ID of the environment host Amazon EC2 instance.
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 60")
INSTANCEID=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" -v
http://169.254.169.254/latest/meta-data/instance-id 2> /dev/null)
REGION=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/
latest/meta-data/placement/region 2> /dev/null)

# Get the ID of the Amazon EBS volume associated with the instance.
VOLUMEID=$(aws ec2 describe-instances \
--instance-id $INSTANCEID \
--query "Reservations[0].Instances[0].BlockDeviceMappings[0].Ebs.VolumeId" \
--output text \
--region $REGION)

# Resize the EBS volume.
aws ec2 modify-volume --volume-id $VOLUMEID --size $SIZE

# Wait for the resize to finish.
while [ \
```

```

"$(aws ec2 describe-volumes-modifications \
  --volume-id $VOLUMEID \
  --filters Name=modification-state,Values="optimizing","completed" \
  --query "length(VolumesModifications)" \
  --output text)" != "1" ]; do
sleep 1
done

# Check if we're on an NVMe filesystem
if [[ -e "/dev/xvda" && $(readlink -f /dev/xvda) = "/dev/xvda" ]]
then
# Rewrite the partition table so that the partition takes up all the space that it
can.
  sudo growpart /dev/xvda 1
# Expand the size of the file system.
# Check if we're on AL2 or AL2023
STR=$(cat /etc/os-release)
SUBAL2="VERSION_ID=\"2\""
SUBAL2023="VERSION_ID=\"2023\""
if [[ "$STR" == *"$SUBAL2"* || "$STR" == *"$SUBAL2023"* ]]
then
  sudo xfs_growfs -d /
else
  sudo resize2fs /dev/xvda1
fi

else
# Rewrite the partition table so that the partition takes up all the space that it
can.
  sudo growpart /dev/nvme0n1 1

# Expand the size of the file system.
# Check if we're on AL2 or AL2023
STR=$(cat /etc/os-release)
SUBAL2="VERSION_ID=\"2\""
SUBAL2023="VERSION_ID=\"2023\""
if [[ "$STR" == *"$SUBAL2"* || "$STR" == *"$SUBAL2023"* ]]
then
  sudo xfs_growfs -d /
else
  sudo resize2fs /dev/nvme0n1p1
fi
fi

```

3. IDE のターミナルセッションから、`resize.sh` ファイルが格納されているディレクトリに移動します。それから次のコマンドを実行し、20 を Amazon EBS ボリュームのリサイズとして希望する Gib 単位のサイズに置き換えます。

- ```
bash resize.sh 20
```
- ```
chmod +x resize.sh  
./resize.sh 20
```

が AWS Cloud9 使用する Amazon EBS ボリュームを暗号化する

このトピックでは、AWS Cloud9 開発環境で使用される EC2 インスタンスの Amazon EBS ボリュームを暗号化する方法について説明します。

Amazon EBS 暗号化では、次のデータが暗号化されます。

- ボリューム内の保管中のデータ
- ボリュームとインスタンスの間で移動されるすべてのデータ
- ボリュームから作成されたすべてのスナップショット
- それらのスナップショットから作成されたすべてのボリューム

Amazon EBS ボリュームには、AWS Cloud9 EC2 開発環境が使用する2つの暗号化オプションがあります。

- デフォルトでの暗号化 – 作成する新しい EBS ボリュームとスナップショットコピーの暗号化を強制するように AWS アカウント を設定できます。暗号化は、デフォルトでは、AWS リージョンのレベルで有効になっています。そのため、そのリージョン内の個々のボリュームまたはスナップショットに対して有効にすることはできません。さらに、Amazon EBS は、インスタンスの起動時に作成されるボリュームを暗号化します。そのため、EC2 環境を作成する前にこの設定を有効にする必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「[デフォルトで暗号化](#)」を参照してください。
- EC2 環境で使用される既存の Amazon EBS ボリュームの暗号化 — EC2 インスタンス用に作成済みの特定の Amazon EBS ボリュームを暗号化できます。このオプションでは、AWS Key Management Service (AWS KMS) を使用して暗号化されたボリュームへのアクセスを管理します。関連する手順については、「[が AWS Cloud9 使用する既存の Amazon EBS ボリュームを暗号化する](#)」を参照してください。

⚠ Important

AWS Cloud9 IDE がデフォルトで暗号化された Amazon EBS ボリュームを使用している場合、AWS Identity and Access Management のサービスにリンクされたロールは、これらの EBS ボリューム AWS KMS key の にアクセス AWS Cloud9 する必要があります。アクセスが提供されていない場合、IDE AWS Cloud9 の起動に失敗し、デバッグが困難になる可能性があります。

アクセスを提供するには、Amazon EBS AWS Cloud9ボリュームで使用される KMS キーにAWSServiceRoleForAWSCloud9、 のサービスにリンクされたロールを追加します。このタスクの詳細については、「規範ガイダンスパターン」の「[デフォルトの暗号化で Amazon EBS ボリュームを使用する AWS Cloud9 IDE を作成する](#)」を参照してください。
AWS

が AWS Cloud9 使用する既存の Amazon EBS ボリュームを暗号化する

既存の Amazon EBS ボリュームを暗号化するには、AWS KMS を使用して KMS キーを作成します。置き換えるボリュームのスナップショットを作成した後、KMS キーを使用してスナップショットのコピーを暗号化します。

次に、そのスナップショットを使用して暗号化されたボリュームを作成します。次に、EC2 インスタンスからデタッチし、暗号化されたボリュームをアタッチすることで、暗号化されていないボリュームを置き換えます。

最後に、カスターマネージドキーのキーポリシーを更新して、AWS Cloud9 サービスロールのアクセスを有効にする必要があります。

i Note

次の手順の中心は、カスターマネージドキーを使用してボリュームを暗号化する方法です。アカウントの AWS のサービスに AWS マネージドキーを使用することもできます。Amazon EBS のエイリアスは aws/ebs です。このデフォルトの暗号化オプションを選択した場合は、カスターマネージドキーを作成するステップ 1 をスキップします。また、キーポリシーを更新するステップ 8 をスキップします。これは、 のキーポリシーを変更できないためです AWS マネージドキー。

既存の Amazon EBS ボリュームを暗号化するには

1. AWS KMS コンソールで、対称 KMS キーを作成します。詳細については、AWS Key Management Service デベロッパーガイドの「[対称 KMS キーの作成](#)」を参照してください。
2. Amazon EC2 コンソールで、環境で使用されている Amazon EBS-backed インスタンスを停止します。[コンソールまたはコマンドラインを使用して、ドインスタンスを停止タグを追加できます](#)。
3. Amazon EC2 コンソールのナビゲーションペインで、[Snapshots] (スナップショット) を選択して、暗号化したい[既存のボリュームのスナップショットを作成します](#)。
4. Amazon EC2 コンソールのナビゲーションペインで、[スナップショット] を選択して、[スナップショットをコピーします](#)。スナップショットのコピーダイアログボックスで、暗号化を有効にするには、以下の操作を実行します。
 - このスナップショットを暗号化するを選択します。
 - [Master Key] (マスターキー) には、作成済みの KMS キーを選択します。(を使用している場合は AWS マネージドキー、(デフォルト) aws/ebs 設定のままにします。)
5. [暗号化されたスナップショット から新しいボリュームを作成します](#)。

Note

暗号化されたスナップショットから作成された新しい Amazon EBS ボリュームは、自動的に暗号化されます。

6. Amazon EC2 インスタンスから [古い Amazon EBS ボリュームをデタッチ](#)します。
7. [新しい暗号化ボリュームを](#) Amazon EC2 インスタンスに添付します。
8. [AWS マネジメントコンソール デフォルトビュー、ポリシービュー、または AWS KMS API を使用して、KMS キーの AWS マネジメントコンソール キーポリシー](#)を更新します。次のキーポリシーステートメントを追加して、AWS Cloud9 サービス が KMS キーAWSServiceRoleForAWSCloud9にアクセスできるようにします。

Note

を使用している場合は AWS マネージドキー、このステップをスキップします。

```
{
```

```

    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:{Partition}:iam::{AccountId}:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:{Partition}:iam::{AccountId}:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
    },
    "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": "true"
        }
    }
}
}

```

9. Amazon EC2 インスタンスを再起動します。Amazon EC2 インスタンスの再起動についての詳細は、「[インスタンスの停止と起動](#)」を参照してください。

での環境の削除 AWS Cloud9

使用しなくなった AWS Cloud9 開発環境 AWS アカウント に関連する への継続的な課金を防ぐには、環境を削除します。

- [コンソールで環境を削除する](#)
- [コードで環境を削除する](#)

コンソールで環境を削除する

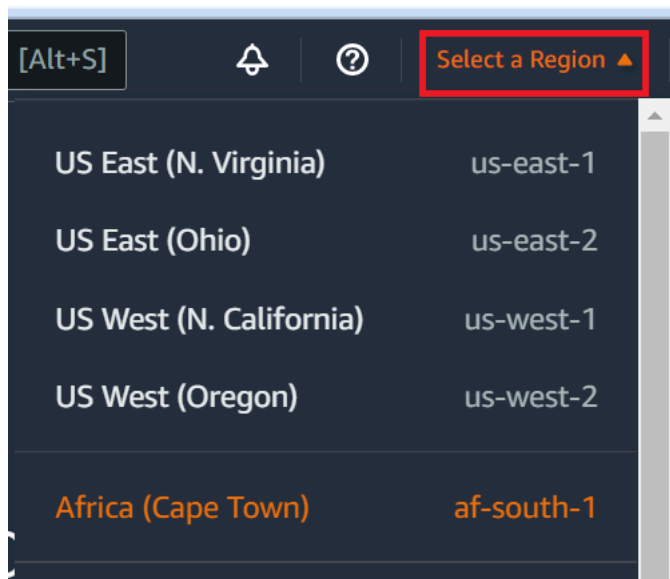
⚠ Warning

環境を削除すると、は環境を完全に AWS Cloud9 削除します。これには、すべての関連する設定、ユーザーデータおよびコミットされていないコードの完全な削除が含まれます。削除された環境は復元できません。

1. AWS Cloud9 コンソールにサインインします。

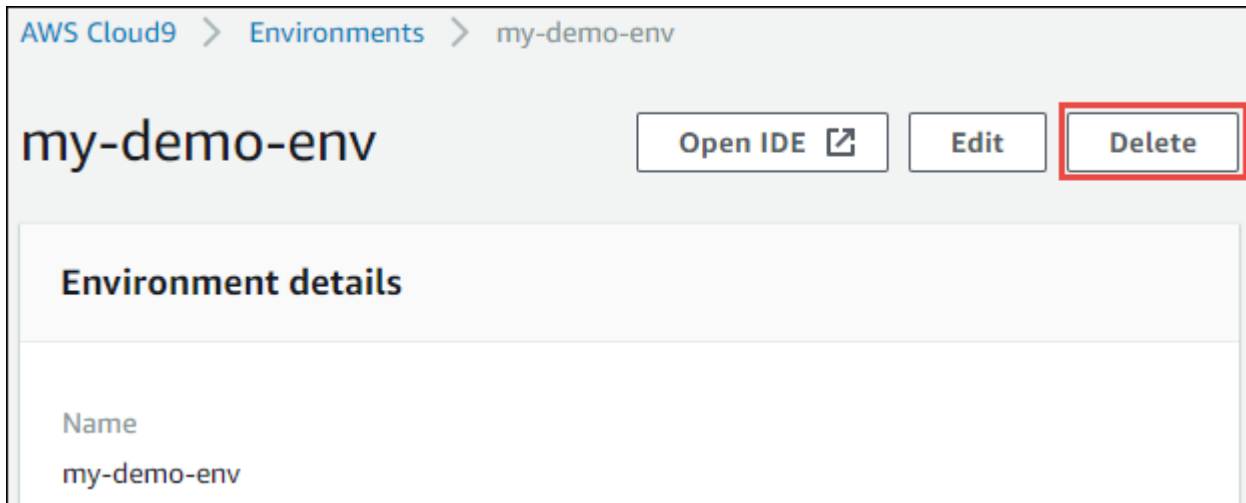
- を使用するののお客様だけ AWS アカウント であるか、単一の IAM ユーザーである場合は AWS アカウント、<https://console.aws.amazon.com/cloud9/> にアクセスしてください。
- 組織が を使用している場合は AWS IAM アイデンティティセンター、AWS アカウント 管理者にサインイン手順を依頼してください。

2. 上部のナビゲーションバーで、環境 AWS リージョン が配置されている を選択します。

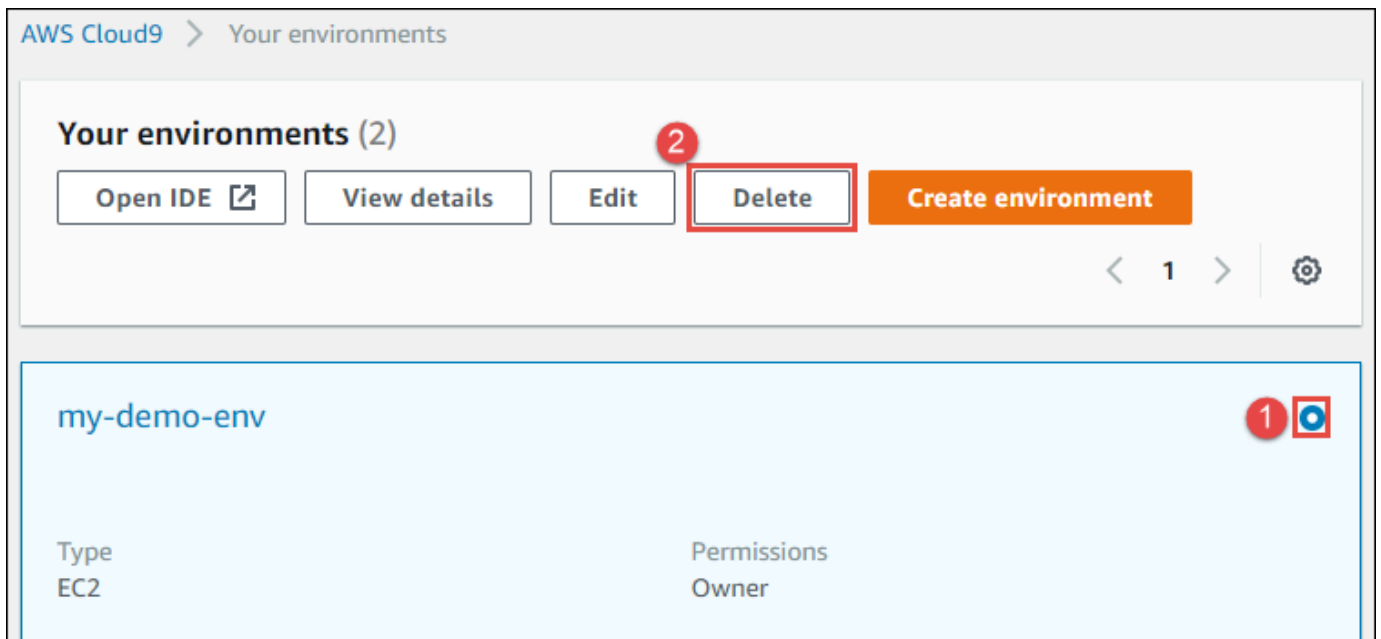


3. 環境のリストでは、削除したい環境に対して、次のいずれかを実行します。

- 環境のカードのタイトルを選択します。次のページで [Delete] (削除) を選択します。



- 環境のカードを選択し、[削除] ボタンを選択します。



4. [Delete (削除)] ダイアログボックスに「Delete」と入力し、[Delete (削除)] を選択します。

- EC2 環境

AWS Cloud9 は、その環境に接続された Amazon EC2 インスタンスも終了します。

Note

アカウントの削除に失敗すると、コンソールのウェブページの上部にバナーが表示されます。また、環境のカード (存在する場合) にも、環境の削除に失敗したことが示されます。

- SSH 環境

環境が Amazon EC2 インスタンスに接続されている場合、そのインスタンスは終了 AWS Cloud9 しません。後でそのインスタンスを終了しない場合は、AWS アカウント でそのインスタンスに関連する Amazon EC2 に対して継続的な料金が発生します。

5. 環境が SSH 環境の場合、はその環境に接続されたクラウドコンピューティングインスタンスまたは独自のサーバー上の非表示のサブディレクトリを AWS Cloud9 残します。削除したい場合、今ならそのサブディレクトリを安全に削除できます。サブディレクトリは .c9 という名前です。サブディレクトリは、環境を作成したときに指定した [Environment path] (環境パス) ディレクトリにあります。

環境がコンソールに表示されない場合は、次の 1 つ以上のアクションを試行して表示を試みてください。

- [Environments] (環境) ページのドロップダウンメニューバーで、以下の 1 つまたは複数を選択します。
 - 環境を選択して、選択した AWS リージョン および 内で AWS エンティティが所有するすべての環境を表示します AWS アカウント。
 - 選択した AWS リージョン と 内でエンティティが招待されたすべての環境を表示するには、共有を選択します AWS アカウント。AWS
 - すべてのアカウント環境を選択すると、選択した 内のすべての環境 AWS リージョン が表示され、AWS エンティティ AWS アカウント には表示するアクセス許可があります。
 - 自分がメンバーである環境が [Shared with you] (自分と共有) リストに表示されていない場合は、環境の所有者に確認してください。
 - 上部のナビゲーションバーで、別のナビゲーションバーを選択します AWS リージョン。

コードで環境を削除する

Warning

環境を削除すると、は環境を完全に AWS Cloud9 削除します。これには、すべての関連する設定、ユーザーデータおよびコミットされていないコードの完全な削除が含まれます。削除された環境は復元できません。

コードを使用して環境を削除するには AWS Cloud9、次のように環境 AWS Cloud9 の削除オペレーションを呼び出します。

AWS CLI	delete-environment
AWS SDK for C++	DeleteEnvironmentRequest 、 DeleteEnvironmentResult
AWS SDK for Go	DeleteEnvironment 、 DeleteEnvironmentRequest 、 DeleteEnvironmentWithContext
AWS SDK for Java	DeleteEnvironmentRequest 、 DeleteEnvironmentResult
AWS SDK for JavaScript	deleteEnvironment
AWS SDK for .NET	DeleteEnvironmentRequest 、 DeleteEnvironmentResponse
AWS SDK for PHP	deleteEnvironment
AWS SDK for Python (Boto)	delete_environment
AWS SDK for Ruby	delete_environment
AWS Tools for Windows PowerShell	Remove-C9Environment
AWS Cloud9 API	DeleteEnvironment

IDE AWS Cloud9 の使用

統合開発環境 (IDE) は、ソースコードエディタ、デバッガー、ビルドツールなど、コーディングの生産性を向上させるツールのセットを提供します。

Important

AWS Cloud9の使用時に推奨されるベストプラクティスを以下に示します。

- ソース管理を使用し、環境を頻繁にバックアップします。AWS Cloud9 は自動バックアップを実行しません。
- 環境でソフトウェアを定期的に更新します。AWS Cloud9 は自動ソフトウェア更新を実行しません。
- AWS アカウントで AWS CloudTrail をオンにして、環境内のアクティビティを追跡します。詳細については、「[を使用した AWS Cloud9 API コールのログ記録 AWS CloudTrail](#)」を参照してください
- 信頼されたユーザーとのみ環境を共有します。環境を共有すると、AWS アクセス認証情報が危険にさらされる可能性があります。詳細については、「[での共有環境の使用 AWS Cloud9](#)」を参照してください

これらのトピックを 1 AWS Cloud9 つ以上読んで、IDE を操作する方法について説明します。

トピック

- [IDE AWS Cloud9 のツアー](#)
- [IDE AWS Cloud9 での言語サポート](#)
- [IDE AWS Cloud9 での言語サポートの強化](#)
- [IDE AWS Cloud9 のメニューバーコマンドリファレンス](#)
- [IDE AWS Cloud9 でのテキストの検索と置換](#)
- [IDE AWS Cloud9 でのファイルのプレビュー](#)
- [IDE AWS Cloud9 で実行中のアプリケーションをプレビューする](#)
- [IDE AWS Cloud9 でのファイルリビジョンの操作](#)
- [IDE AWS Cloud9 でのイメージファイルの使用](#)
- [AWS Cloud9 IDE でのビルダー、ランナー、デバッガーの操作](#)

- [AWS Cloud9 IDE でのカスタム環境変数の使用](#)
- [IDE AWS Cloud9 でのプロジェクト設定の操作](#)
- [IDE AWS Cloud9 でのユーザー設定の操作](#)
- [AWS Cloud9 IDE での AWS プロジェクトおよびユーザー設定の操作](#)
- [IDE AWS Cloud9 でのキー割り当ての操作](#)
- [IDE AWS Cloud9 でのテーマの使用](#)
- [IDE AWS Cloud9 での初期化スクリプトの管理](#)
- [IDE の MacOS AWS Cloud9 デフォルトキー割り当てリファレンス](#)
- [IDE の MacOS Vim AWS Cloud9 キー割り当てリファレンス](#)
- [IDE の MacOS Emacs AWS Cloud9 キー割り当てリファレンス](#)
- [IDE の MacOS Sublime AWS Cloud9 キー割り当てリファレンス](#)
- [IDE の Windows/Linux AWS Cloud9 デフォルトキー割り当てリファレンス](#)
- [IDE の Windows/Linux Vim AWS Cloud9 キー割り当てリファレンス](#)
- [IDE の Windows/Linux Emacs AWS Cloud9 キー割り当てリファレンス](#)
- [IDE の Windows/Linux Sublime AWS Cloud9 キー割り当てリファレンス](#)
- [IDE AWS Cloud9 のコマンドリファレンス](#)

IDE AWS Cloud9 のツアー

このトピックでは、AWS Cloud9 統合開発環境 (IDE) の基本的なツアーを提供します。この演習を最大限に活用するには、以下のステップを順番に実行します。

トピック

- [前提条件](#)
- [ステップ 1: メニューバー](#)
- [ステップ 2: ダッシュボード](#)
- [ステップ 3: \[Environment \(環境\) \] ウィンドウ](#)
- [ステップ 4: エディタ、タブ、ペイン](#)
- [ステップ 5: コンソール](#)
- [ステップ 6: \[Open Files \(開いているファイル\) \] セクション](#)
- [ステップ 7: ガーター](#)
- [ステップ 8: ステータスバー](#)

- [ステップ 9: \[Outline \(アウトライン\) \] ウィンドウ](#)
- [ステップ 10: \[Go \(移動\) \] ウィンドウ](#)
- [ステップ 11: \[Immediate \(即時\) \] タブ](#)
- [ステップ 12: プロセスリスト](#)
- [ステップ 13: 設定](#)
- [ステップ 14: ターミナル](#)
- [ステップ 15: \[Debugger \(デバッガー\) \] ウィンドウ](#)
- [結論](#)

前提条件

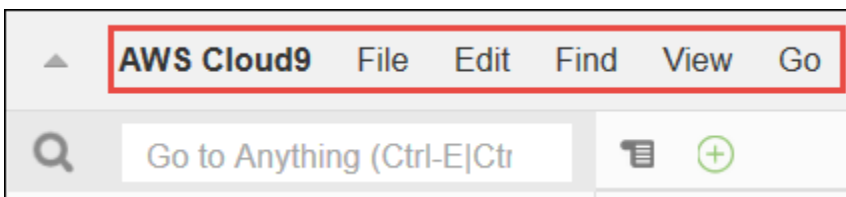
このツアーに参加するには、AWS アカウントとオープンな AWS Cloud9 開発環境が必要です。これらの操作を行う方法を理解するには、「[の開始方法 AWS Cloud9](#)」のステップに従います。または、「[セットアップ AWS Cloud9](#)」や「[での環境の使用 AWS Cloud9](#)」などの個別の関連トピックを試すこともできます。

Warning

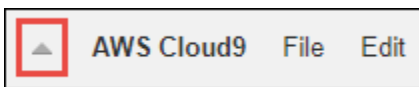
AWS Cloud9 開発環境があると、AWS アカウントに料金が発生する可能性があります。これらには、EC2 環境を使用している場合の Amazon EC2 の料金が含まれます。詳細については、[Amazon EC2 の料金表](#)を参照してください。

ステップ 1: メニューバー

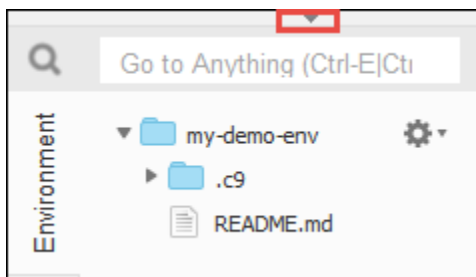
メニューバーは IDE の上部にあり、ファイルやコードの操作、IDE 設定の変更を行う一般的なコマンドが含まれています。また、メニューバーからコードのプレビューと実行も行うことができます。



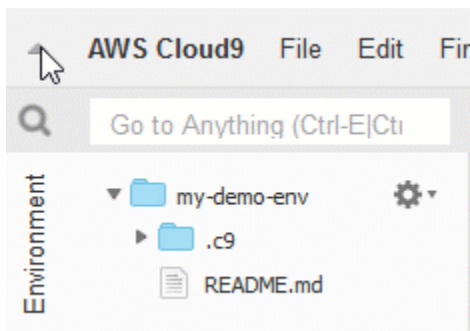
次のようにメニューバーの端の矢印を選択してメニューバーを非表示にできます。



メニューバーをもう一度表示するには、次のように、メニューバーがあった場所の中ほどにある矢印を選択します。



結果を以下と比較します。



このチュートリアル次のいくつかのセクションでは、IDE を使用して一連のファイル进行操作できます。これらのファイルを設定するには、[File (ファイル)]、[New File (新規ファイル)] の順にクリックします。

次に、次のテキストをコピーして Untitled1 エディタタブに貼り付けます。

```
fish.txt
-----
A fish is any member of a group of organisms that consist of
all gill-bearing aquatic craniate animals that lack limbs with
digits. They form a sister group to the tunicates, together
forming the olfactores. Included in this definition are
lampreys and cartilaginous and bony fish as well as various
extinct related groups.
```

ファイルを保存するには、[File (ファイル)]、[Save (保存)] の順に選択します。ファイルに fish.txt と名前を付け、[Save (保存)] を選択します。

これらの手順を繰り返し、2 つ目のファイルを次の内容で cat.txt として保存します。

```
cat.txt
```

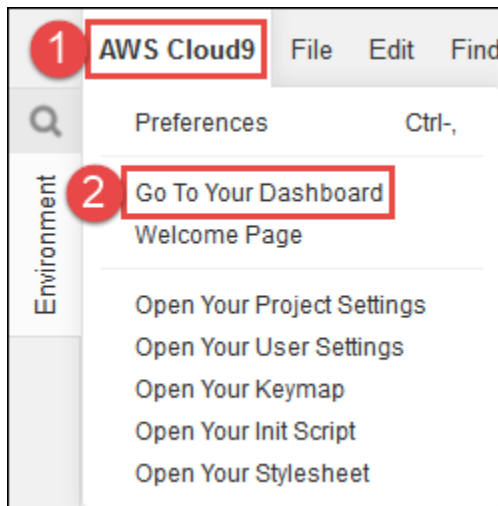
```
-----  
The domestic cat is a small, typically furry, carnivorous mammal.  
They are often called house cats when kept as indoor pets or  
simply cats when there is no need to distinguish them from  
other felids and felines. Cats are often valued by humans for  
companionship and for their ability to hunt.
```

IDE でこれを行うには、多くの場合いくつかの方法があります。たとえば、メニューバーを非表示にするには、端にある矢印を選択する代わりに、[View (表示)]、[Menu Bar (メニューバー)] を選択できます。新規ファイルを作成するには、[File (ファイル)]、[New File (新規ファイル)] を選択する代わりに、Alt-N (Windows/Linux の場合) または Control-N (MacOS の場合) を押すことができます。このチュートリアルを短くするために、1 つの方法についてのみ説明します。さらに IDE に精通したら、自由に試して最も使いやすい方法を見つけてください。

ステップ 2: ダッシュボード

ダッシュボードを使用すると、各環境にすばやくアクセスできます。ダッシュボードからは、環境の作成、環境を開く、および環境の設定の変更ができます。

ダッシュボードを開くには、メニューバーで、[AWS Cloud9]、[Go To Your Dashboard (ダッシュボードを開く)] の順に選択します。



環境の設定を表示するには、[my-demo-environment] カード内のタイトルを選択します。ダッシュボードに戻るには、ウェブブラウザの [戻る] ボタン、または [Environments (環境)] と呼ばれるナビゲーションのパンくずを使用します。

ご使用の環境の IDE を開くには、my-demo-environment カード内側の IDE を開くを選択します。

Note

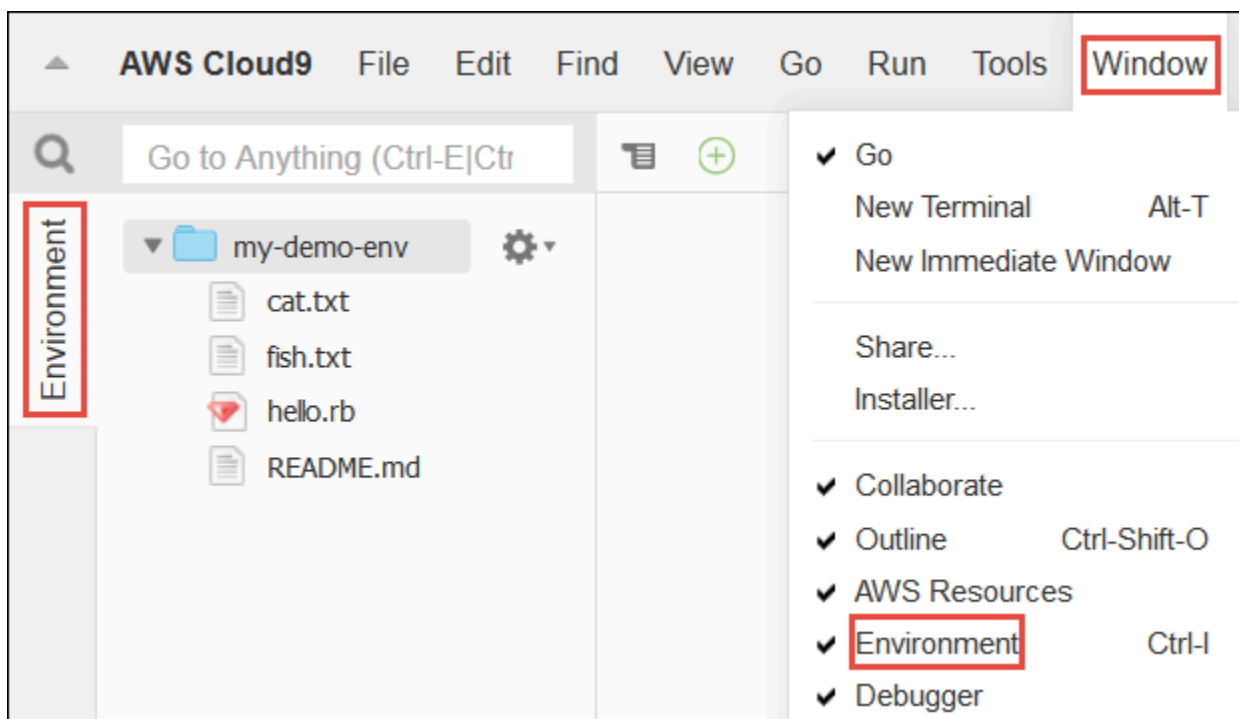
IDE が再表示されるまで数分かかる場合があります。

ステップ 3: [Environment (環境)] ウィンドウ

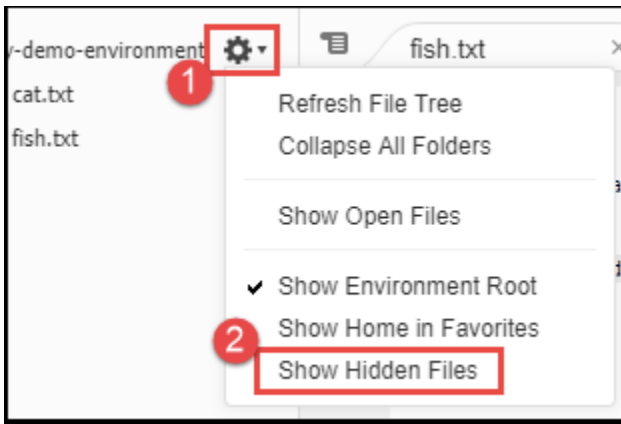
[Environment (環境)] ウィンドウには、環境のフォルダとファイルのリストが表示されます。また、隠しファイルなど、さまざまな種類のファイルを表示できます。

[Environment (環境)] ウィンドウを表示または非表示にするには、[Environment (環境)] ボタンを選択します。

[Environment] (環境) ウィンドウおよび [Environment] (環境) ボタンを表示または非表示にするには、メニューバーで [Window] (ウィンドウ)、[Environment] (環境) の順に選択します。



隠しファイルを表示または非表示にするには、[Environment (環境)] ウィンドウで歯車アイコンを選択してから、[Show Hidden Files (隠しファイルを表示する)] を選択します。



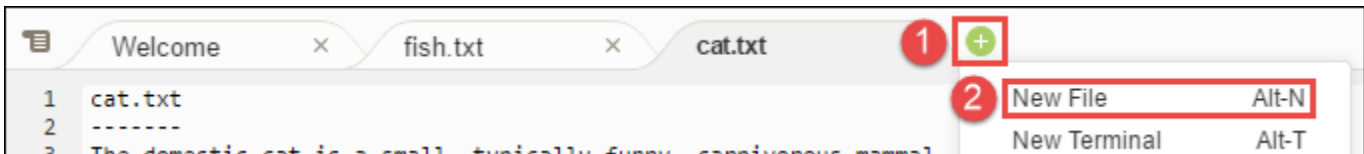
ステップ 4: エディタ、タブ、ペイン

エディタでは、コードの記述、ターミナルセッションの実行、IDE 設定の変更などを行うことができます。開いているファイルやターミナルセッションなどの各インスタンスは、タブとして表示されます。タブはペインにグループ化できます。タブはそのペインの端に表示されます。

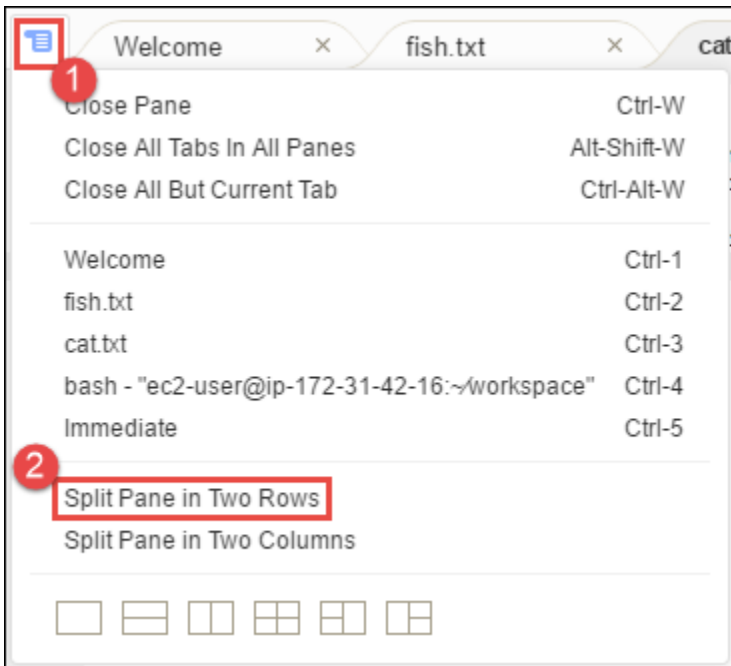


タブを表示または非表示にするには、メニューバーで [View (表示)]、[Tab Buttons (タブボタン)] を選択します。

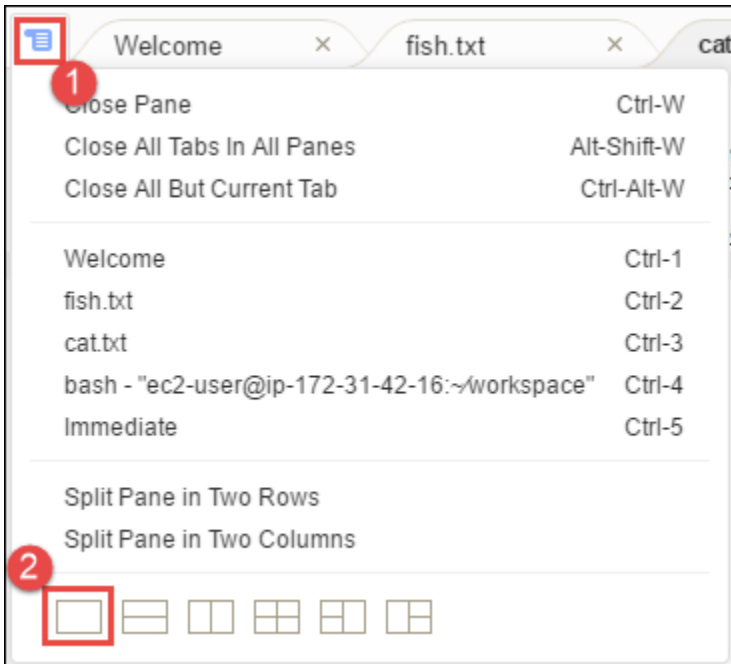
新しいタブを開くには、タブの並びの端にある [+] アイコンを選択します。続いて、次のように、使用可能なコマンドのいずれか ([New File (新規ファイル)] など) を選択します。



2 つのペインを表示するには、タブの並びの端にある、ドロップダウンメニューに似たアイコンを選択します。続いて、次のように、 [Split Pane in Two Rows (ペインを 2 行に分割する)] を選択します。

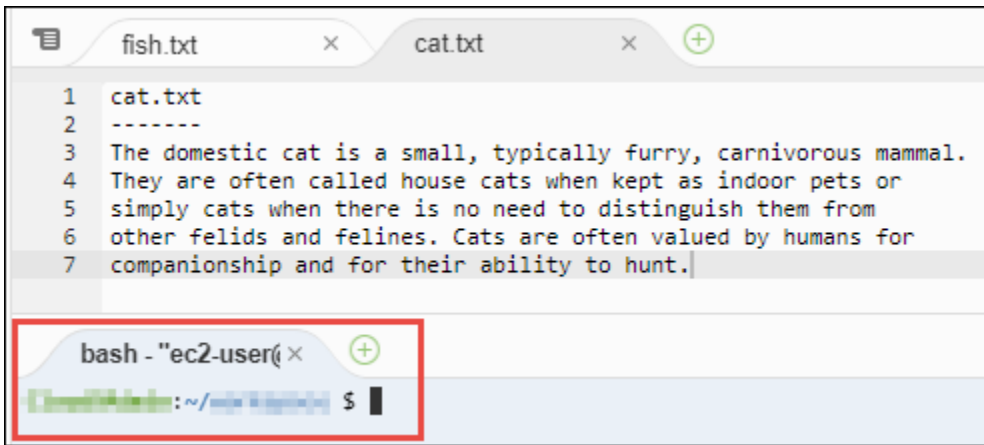


単一のペインに戻すには、もう一度ドロップダウンメニューのアイコンを選択し、次のように、1つの正方形のアイコンを選択します。



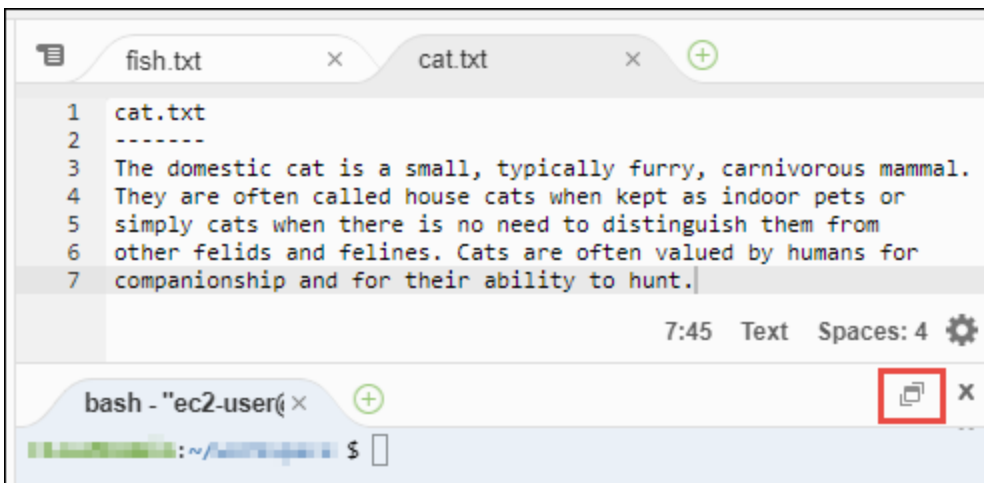
ステップ 5: コンソール

コンソールは、タブを作成および管理するための代替の場所です。デフォルトでは、[Terminal] タブが含まれますが、他のタイプのタブを含めることもできます。



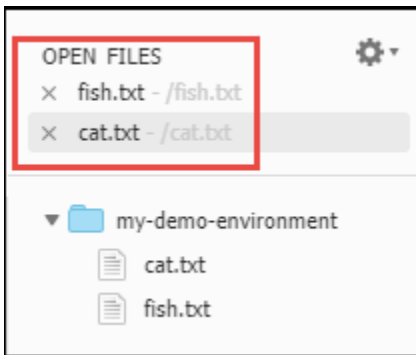
コンソールを表示または非表示にするには、メニューバーで [View (表示)]、[Console (コンソール)] の順に選択します。

コンソールを拡張または縮小するには、次のように、コンソールの端にあるサイズ変更アイコンを選択します。



ステップ 6: [Open Files (開いているファイル)] セクション

[Open Files (開いているファイル)] セクションには、エディタで現在開いているすべてのファイルが一覧表示されます。[Open Files (開いているファイル)] は、次のように [Environment (環境)] ウィンドウの一部です。

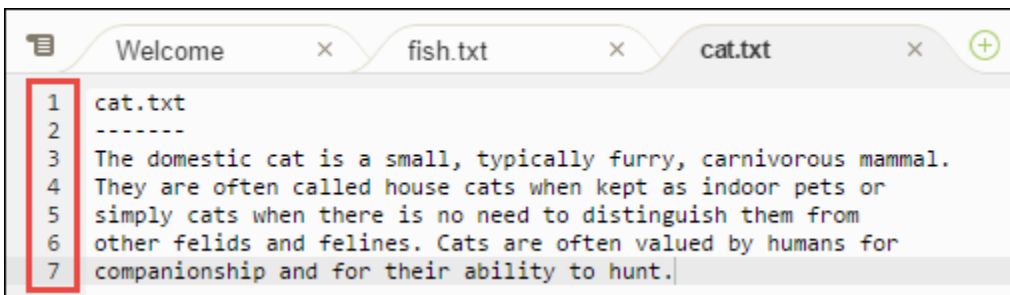


[Open Files (開いているファイル)] セクションを表示または非表示にするには、メニューバーで [View (表示)]、[Open Files (開いているファイル)] の順に選択します。

開いているファイルを切り替えるには、リストから目的のファイルを選択します。

ステップ 7: ガーター

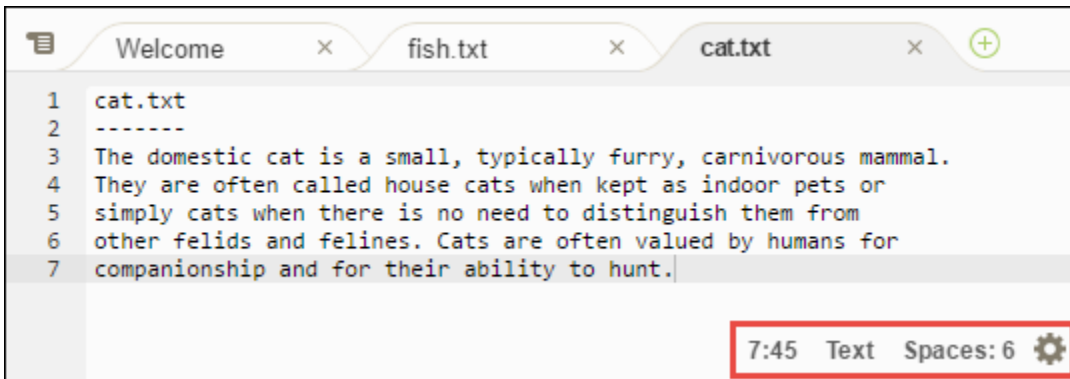
ガーターは、エディタの各ファイルの端にあり、ファイルの操作時に行番号やコンテキスト記号などを表示します。



ガーターを表示または非表示にするには、メニューバーで [View (表示)]、[Gutter (ガーター)] の順に選択します。

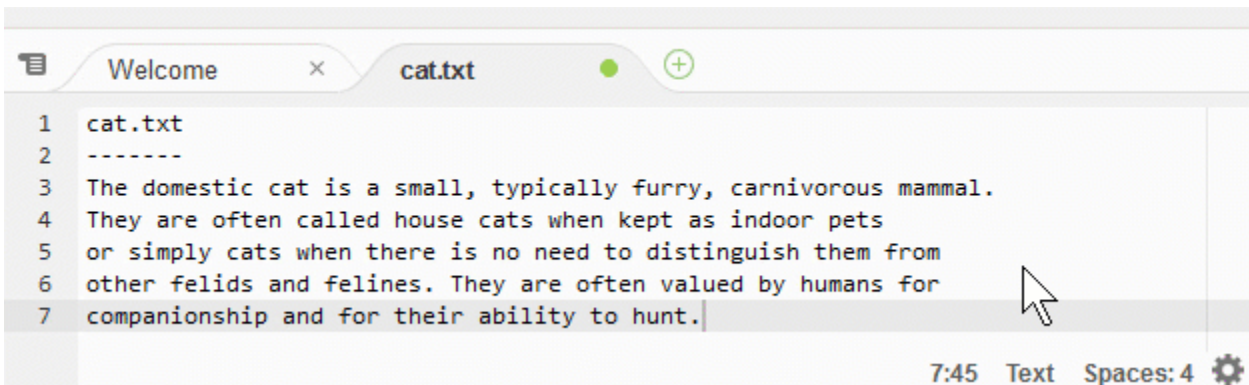
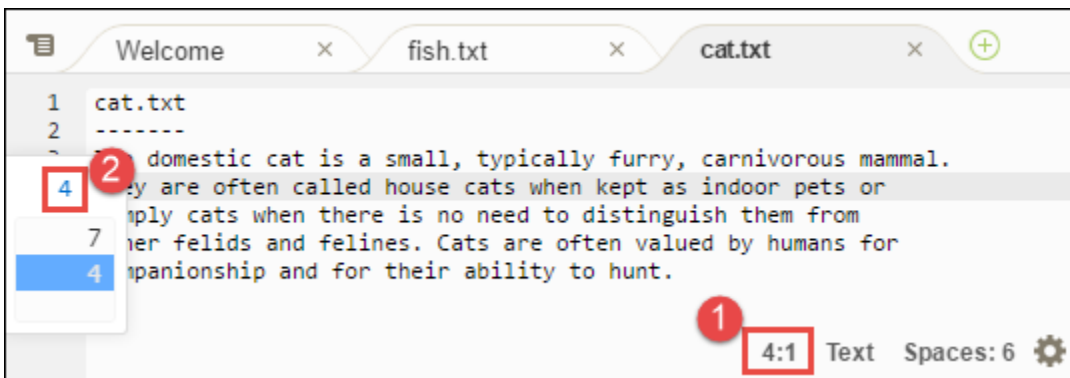
ステップ 8: ステータスバー

ステータスバーは、エディタの各ファイルの端にあり、行番号や文字番号、ファイルタイプの設定、スペースおよびタブの設定、および関連エディタ設定などが表示されます。

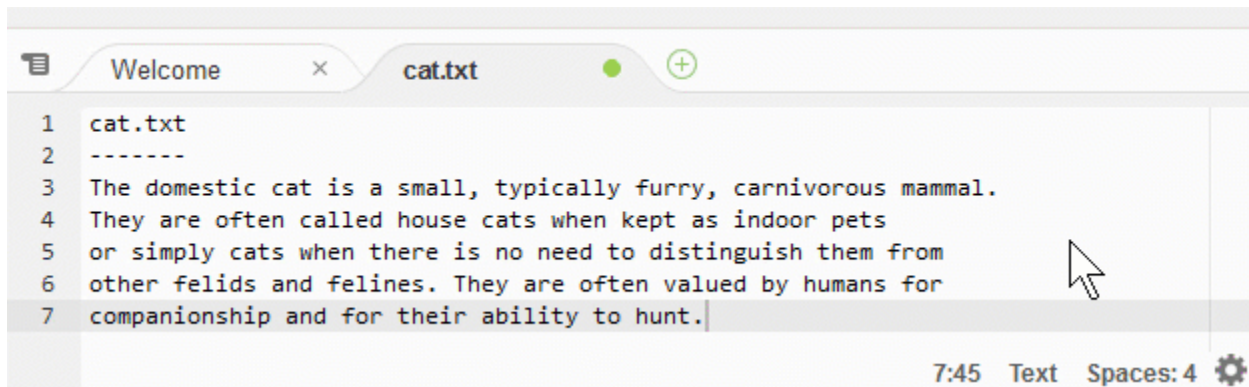
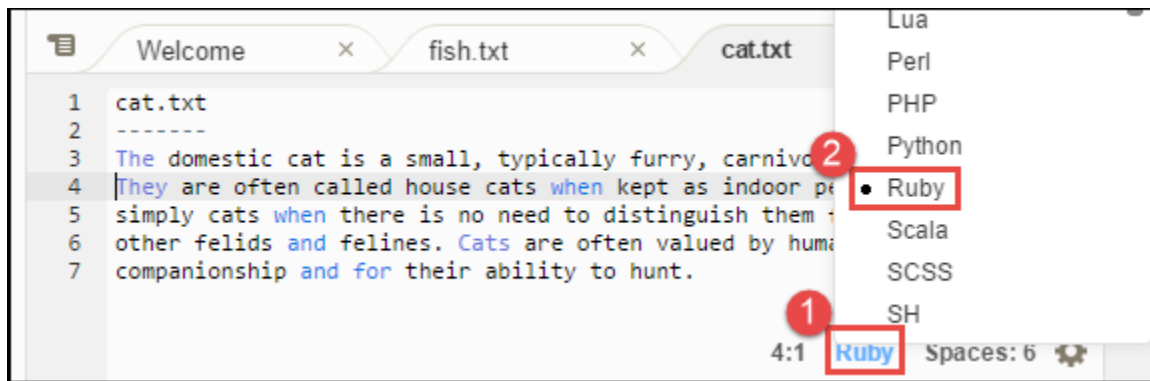


ステータスバーを表示または非表示にするには、メニューバーで [View (表示)]、[Status Bar (ステータスバー)] の順に選択します。

特定の行番号に移動するには、目的のファイルのタブを選択します。次にステータスバーで、行番号と文字番号を見つけます ([7:45] のような形式です)。次のように、行番号 (例: 4) を入力して、Enter を押します。



ファイルタイプの設定を変更するには、ステータスバーで異なるファイルタイプを選択します。たとえば、[cat.txt] の場合、[Ruby] を選択して構文の色の変化を確認します。プレーンテキストの色に戻すには、次のように [Plain Text (プレーンテキスト)] を選択します。



ステップ 9: [Outline (アウトライン)] ウィンドウ

[Outline (アウトライン)] ウィンドウを使用して特定のファイルの場所にすばやく移動できます。

[Outline] (アウトライン) ウィンドウおよび [Outline] (アウトライン) ボタンを表示または非表示にするには、メニューバーで [Window] (ウィンドウ)、[Outline] (アウトライン) の順に選択します。

[Outline (アウトライン)] ウィンドウの機能を確認するために、hello.rb という名前のファイルを作成します。以下のコードをファイルにコピーして保存します。

```
def say_hello(i)
  puts "Hello!"
  puts "i is #{i}"
end

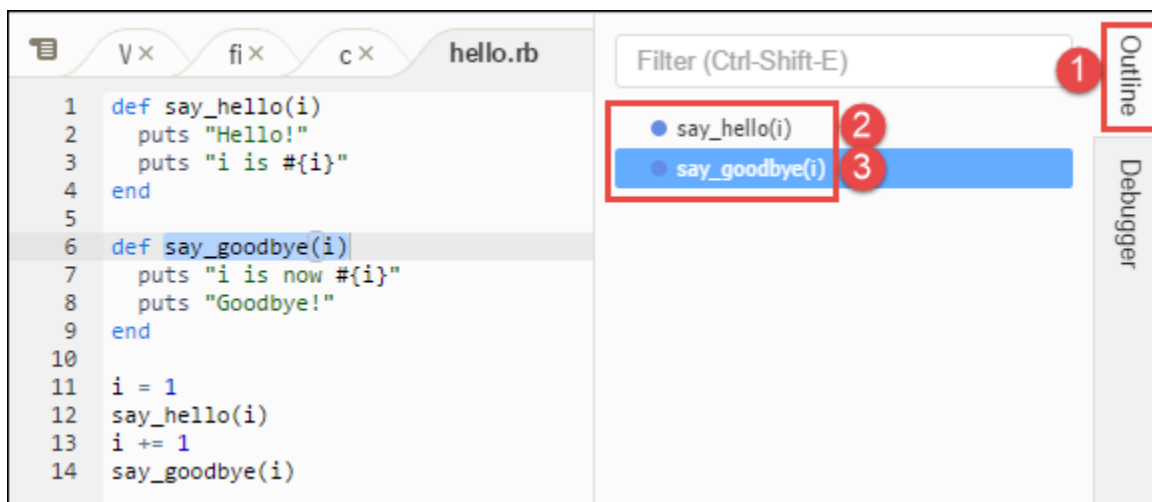
def say_goodbye(i)
  puts "i is now #{i}"
  puts "Goodbye!"
end

i = 1
say_hello(i)
```

```
i += 1
say_goodbye(i)
```

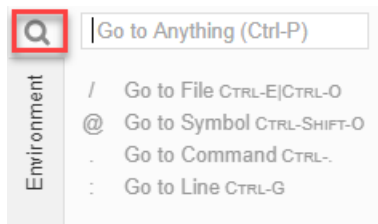
[Outline (アウトライン)] ウィンドウの内容を表示または非表示にするには、[Outline (アウトライン)] ボタンを選択します。

[Outline (アウトライン)] ウィンドウで、次のように [say_hello(i)] を選択してから [say_goodbye(i)] を選択します。



ステップ 10: [Go (移動)] ウィンドウ

[Go (移動)] ウィンドウを使用して、エディタでファイルを開き、記号の定義に移動し、コマンドを実行して、エディタのアクティブなファイルの行に移動します。



[Go (移動)] ウィンドウの内容を表示するには、[Go (移動)] ボタン (拡大鏡アイコン) をクリックします。

[Go] (移動) ウィンドウおよび [Go] (移動) ボタンを表示または非表示にするには、メニューバーで [Window] (ウィンドウ)、[Go] (移動) の順に選択します。

[Go (移動)] ウィンドウを開いた状態で、次の操作を実行できます。

- スラッシュ (/) に続けてファイル名の一部またはすべてを入力する。一致するファイルが一覧表示されたら、ファイルを選択してエディタで開きます。たとえば、「/fish」と入力すると、fish.txt が表示されます。「/.txt」と入力すると、fish.txt と cat.txt が表示されます。

Note

ファイルの検索範囲には、[Environment (環境)] ウィンドウの隠しファイルと隠しフォルダは含まれません。

- アットマーク (@) に続けて記号の名前を入力する。一致する記号が一覧表示されたら、記号を選択してエディタの該当箇所に移動します。たとえば、エディタで hello.rb ファイルが開いてアクティブになっている場合、「@hello」と入力すると say_hello(i) が表示され、「@say」と入力すると say_hello(i) と say_goodbye(i) の両方が表示されます。

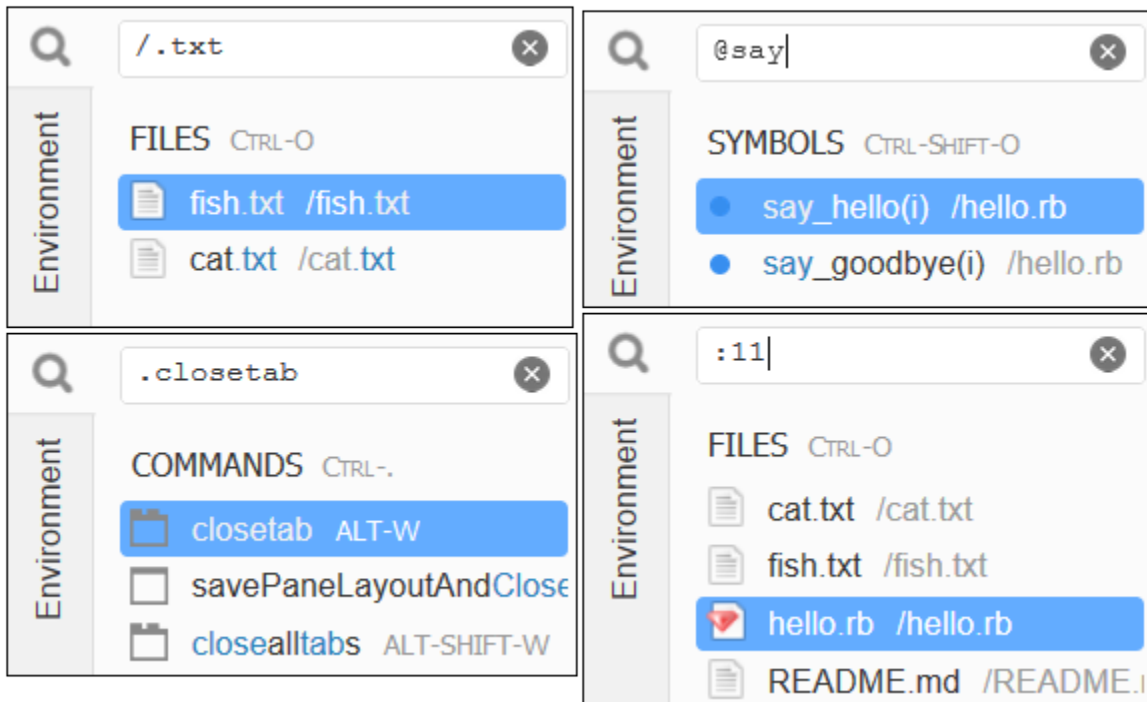
Note

エディタでアクティブになっているファイルが、サポートされている言語プロジェクトの一部である場合、記号の検索範囲は現在のプロジェクトになります。それ以外の場合、記号の検索範囲はエディタのアクティブなファイルに限定されます。詳細については、「[TypeScript のサポートと機能の強化](#)」を参照してください。

- ドット (.) に続けてコマンド名を入力する。コマンドが一覧表示されたら、コマンドを選択して実行します。たとえば、「.closetab」と入力して Enter キーを押すと、エディタの現在のタブが

閉じられます。使用可能なコマンドのリストについては、「[IDE AWS Cloud9 のコマンドリファレンス](#)」を参照してください。

- コロン (:) に続けて数字を入力し、エディタのアクティブなファイルの該当する行番号に移動する。たとえば、エディタで `hello.rb` ファイルが開いてアクティブになっている場合、「:11」と入力すると、そのファイルの行番号 11 に移動します。



現在のキーボードモードとオペレーティングシステム別に、これらの各アクションのキー割り当てを確認するには、メニューバーの [Go (移動)] メニューの利用可能な各 [Go To (移動)] コマンドを参照してください。

ステップ 11: [Immediate (即時)] タブ

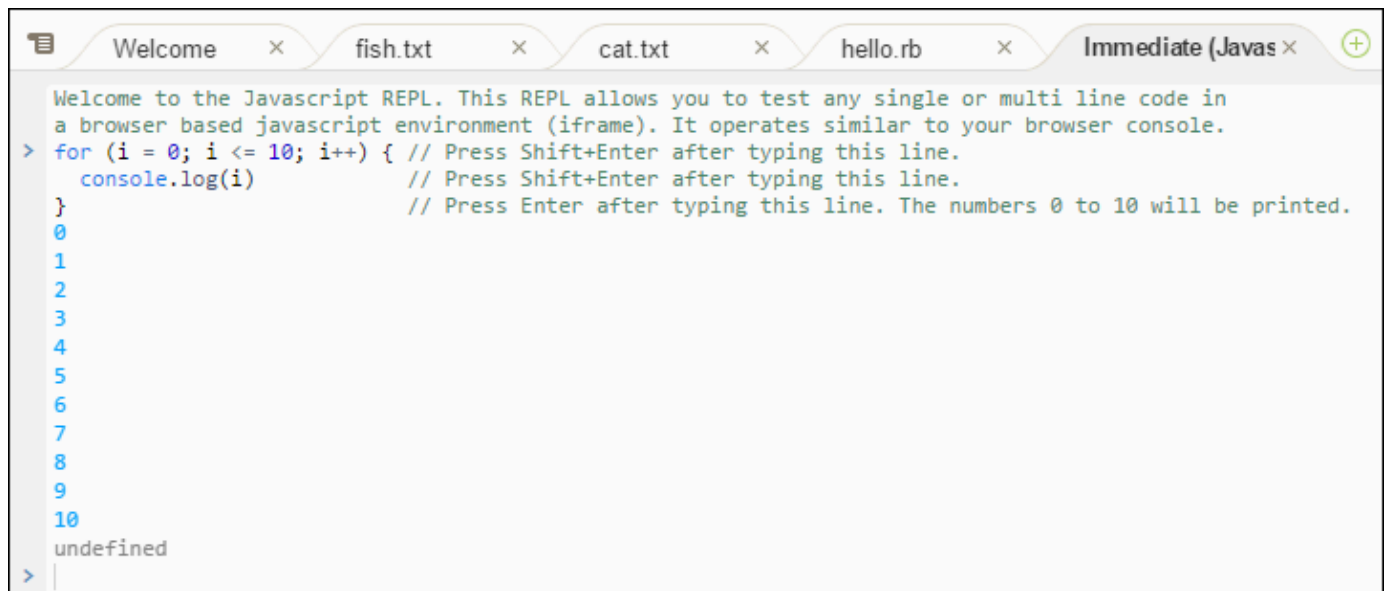
[Immediate (即時)] タブでは、JavaScript コードの小さなスニペットをテストできます。

[Immediate (即時)] タブの動作を確認するには、次の手順を実行します。

1. メニューバーで [Window (ウィンドウ)]、[New Immediate Window (新しい即時ウィンドウ)] の順に選択して [Immediate (即時)] タブを開きます。
2. [Immediate (即時)] タブで何らかのコードを実行します。これを実行するには、以下のコードをウィンドウに入力します。行 1 を入力した後に Shift-Enter を押して、行 2 の後でも再度

押します。行 3 の後で Enter を押します。(行 1 または行 2 を入力した後で Enter ではなく Shift-Enter を押すと、コードが意図したよりも前に実行されます)

```
for (i = 0; i <= 10; i++) { // Press Shift-Enter after typing this line.
  console.log(i)           // Press Shift-Enter after typing this line.
}                           // Press Enter after typing this line. The numbers 0 to
                             10 will be printed.
```



ステップ 12: プロセスリスト

[Process List (プロセスリスト)] には、すべての実行中のプロセスが表示されます。プロセスを停止したり、それ以上実行を望まないプロセスを強制終了することもできます。[Process List (プロセスリスト)] ウィンドウの動作を確認するには、次の手順を実行します。

1. メニューバーで [Tools (ツール)]、[Process List (プロセスリスト)] を選択して [Process List (プロセスリスト)] を表示します。
2. プロセスを検索します。[Process List (プロセスリスト)] で、プロセスの名前を入力します。
3. プロセスを停止または強制終了します。プロセスのリストでプロセスを選択し、[Kill (停止)] または [Force Kill (強制終了)] を選択します。

Process List

Process Name	CPU	MEM	Process Time	PID	User
kworker/0:1H	0.0%	0.0%	0:00	1491	root
init	0.0%	0.4%	0:00	1	root
ksoftirqd/0	0.0%	0.0%	0:00	3	root
kworker/0:0	0.0%	0.0%	0:00	4	root
kworker/0:0H	0.0%	0.0%	0:00	5	root
rcu_sched	0.0%	0.0%	0:00	7	root
rcu_bh	0.0%	0.0%	0:00	8	root
migration/0	0.0%	0.0%	0:00	9	root
kdevtmpfs	0.0%	0.0%	0:00	10	root
netns	0.0%	0.0%	0:00	11	root
perf	0.0%	0.0%	0:00	12	root
kworker/u30:1	0.0%	0.0%	0:00	13	root
xenwatch	0.0%	0.0%	0:00	15	root
kworker/u30:2	0.0%	0.0%	0:00	17	root

Kill

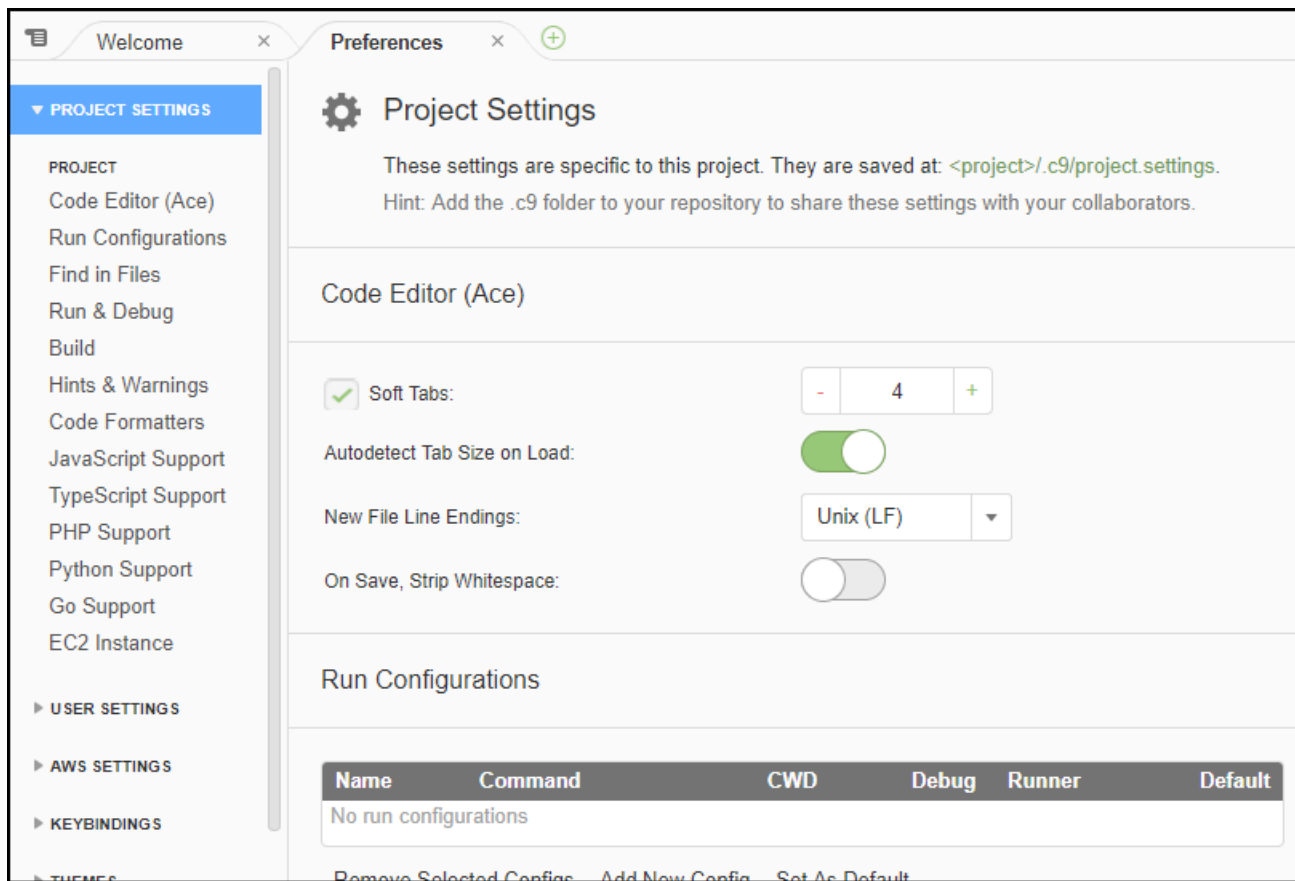
Force Kill

ステップ 13: 設定

設定には以下の設定が含まれます。

- エディタでソフトタブを使用するかどうか、無視するファイルタイプ、PHP や Python などの言語におけるコード補完機能など、現在の環境の設定のみ。
- 環境ごとのユーザー設定。色、フォント、エディタの動作など。
- キー割り当て。ファイルやエディタの操作で使用するショートカットキーの組み合わせなど。
- IDE 全体のテーマ。

設定を表示するには、メニューバーで、[AWS Cloud9]、[Preferences (設定)] の順に選択します。次のような内容が表示されます。



ステップ 14: ターミナル

IDE では 1 つ以上のターミナルセッションを実行できます。ターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。または、[Console (コンソール)] タブの横にある「プラス」アイコンを選択し、[New Terminal (新しいターミナル)] を選択します。

ターミナルでは、以下のコマンドの実行を試すことができます。たとえば、ターミナルで、`echo $PATH` と入力してから、Enter を押して、PATH 環境変数の値を出力します。

次のコマンドの実行を試すこともできます。たとえば、以下のようなコマンドを試します。

- **pwd**。現在のディレクトリへのパスを出力します。
- **aws --version** を使用して、に関するバージョン情報を出力します AWS CLI。
- **ls -l**。現在のディレクトリについての情報を出力します。



ステップ 15: [Debugger (デバッガー)] ウィンドウ

[Debugger (デバッガー)] ウィンドウを使用してコードをデバッグできます。たとえば、コード全体を一度に一部分ずつ段階を追って実行し、経過に伴う変数の値を確認したり、コールスタックを調べたりできます。

Note

この手順は [ベーシック IDE チュートリアル](#) のいずれかからの [ステップ 2: IDE のベーシック演習](#) に似ています。

[Debugger] (デバッガー) ウィンドウおよび [Debugger] (デバッガー) ボタンを表示または非表示にするには、メニューバーで [Window] (ウィンドウ)、[Debugger] (デバッガー) の順に選択します。

このチュートリアルでは、次の操作を行うことで、[Debugger (デバッガー)] ウィンドウと一部の JavaScript コードを試すことができます。

1. ターミナルセッションで次のコマンドを実行して、環境内の Node.js がインストールされていることをチェックします。**node --version**。Node.js がインストールされている場合は、Node.js のバージョン番号が出力されます。この場合は、この手順のステップ 3 (JavaScript コードを作成) までスキップできます。
2. Node.js をインストールする必要がある場合は、以下を実行します。
 - a. 次の 2 つのコマンドを一度に 1 つずつ実行し、環境に最新のアップデートがあることを確認し、Node Version Manager(nvm) をダウンロードします。(nvm は Node.js のバージョンをインストールして管理するために役立つシンプルな Bash シェルスクリプトです。詳細は、GitHub の [Node Version Manager](#) を参照してください。

Amazon Linux の場合:

```
sudo yum -y update
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh |
  bash
```

Ubuntu Server の場合:

```
sudo apt update
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh |
  bash
```

- b. テキストエディタを使用してシェルスクリプトファイル (たとえば ~/.bashrc) を更新し、nvm をロードできるようにします。たとえば、IDE の [Environment (環境)] ウィンドウで歯車アイコンを選択し、[Show Home in Favorites (お気に入りのホームを表示する)] を選択します。このステップを繰り返して、[Show Hidden Files (隠しファイルを表示する)] も同様に選択します。
 - c. ~/.bashrc ファイルを開きます。
 - d. ファイルの末尾に次のコードを入力するか貼り付けて、nvm のロードを有効にします。

Amazon Linux の場合:

```
export NVM_DIR="/home/ec2-user/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm.
```

Ubuntu Server の場合:

```
export NVM_DIR="/home/ubuntu/.nvm"  
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm.
```

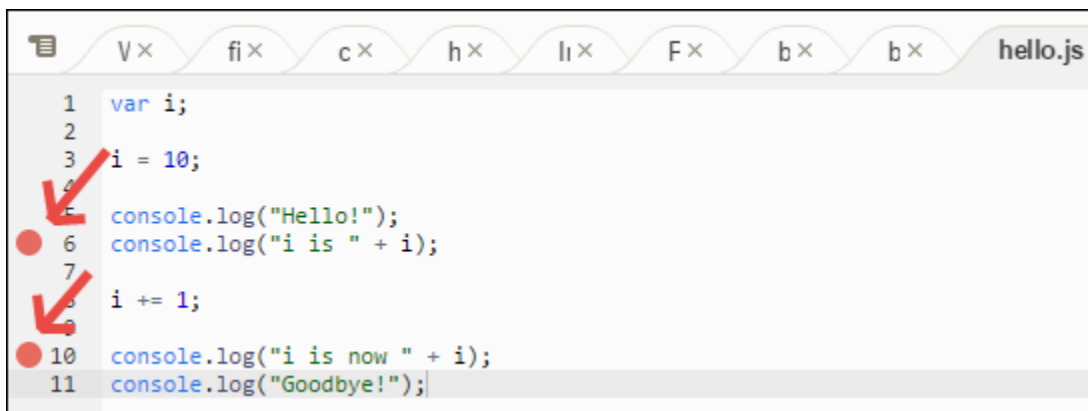
- e. ファイルを保存します。
- f. そのターミナルセッションを閉じて、新しいセッションを開始します。次に、次のコマンドを実行して Node.js の最新バージョンをインストールします。

```
nvm install node
```

- 3. JavaScript コードをいくつか作成してデバッグします。たとえば、ファイルを作成し、そのファイルに次のコードを追加して、hello.js として保存します。

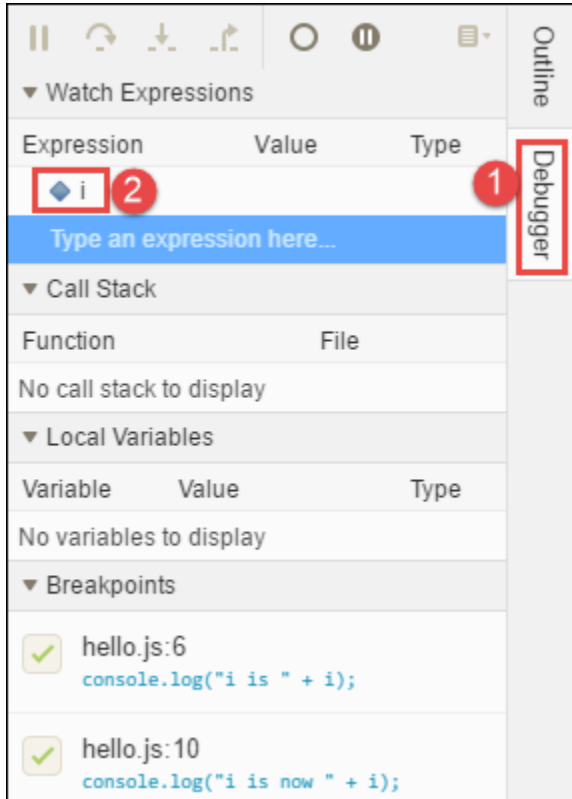
```
var i;  
  
i = 10;  
  
console.log("Hello!");  
console.log("i is " + i);  
  
i += 1;  
  
console.log("i is now " + i);  
console.log("Goodbye!");
```

- 4. コードにブレークポイントをいくつか追加します。たとえば、ガーターで、行 6 と行 10 の横のマージンを選択します。次のように、各行番号の横に赤い円が表示されます。

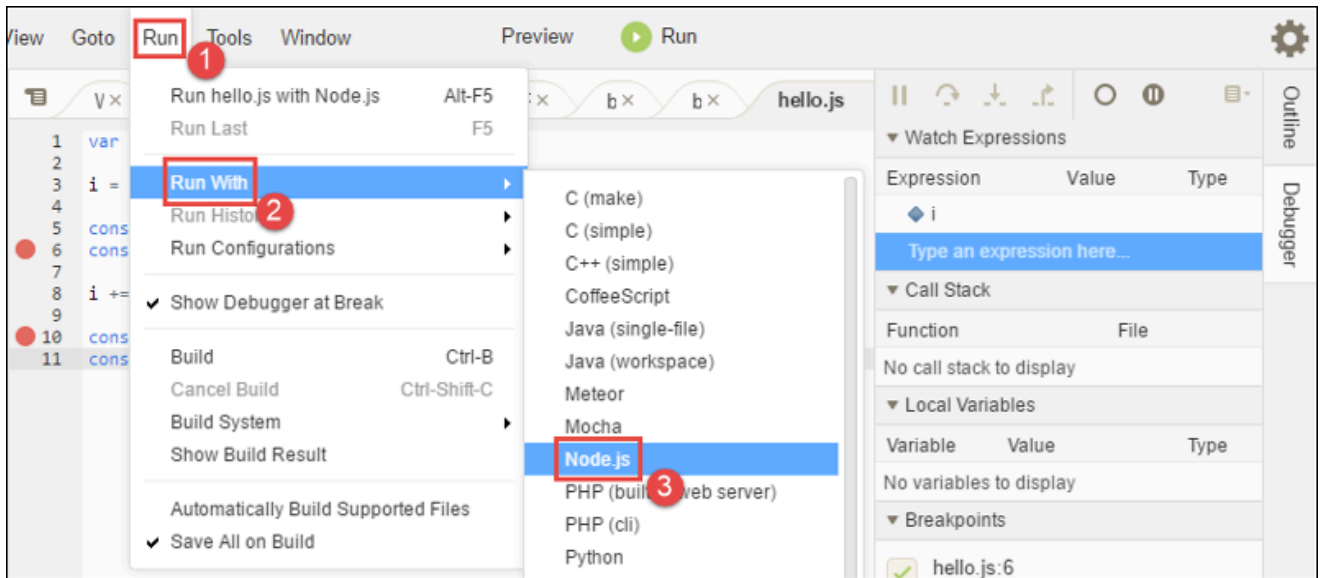


- 5. これで JavaScript コードをデバッグする準備が整いました。これを試すには、以下を実行します。

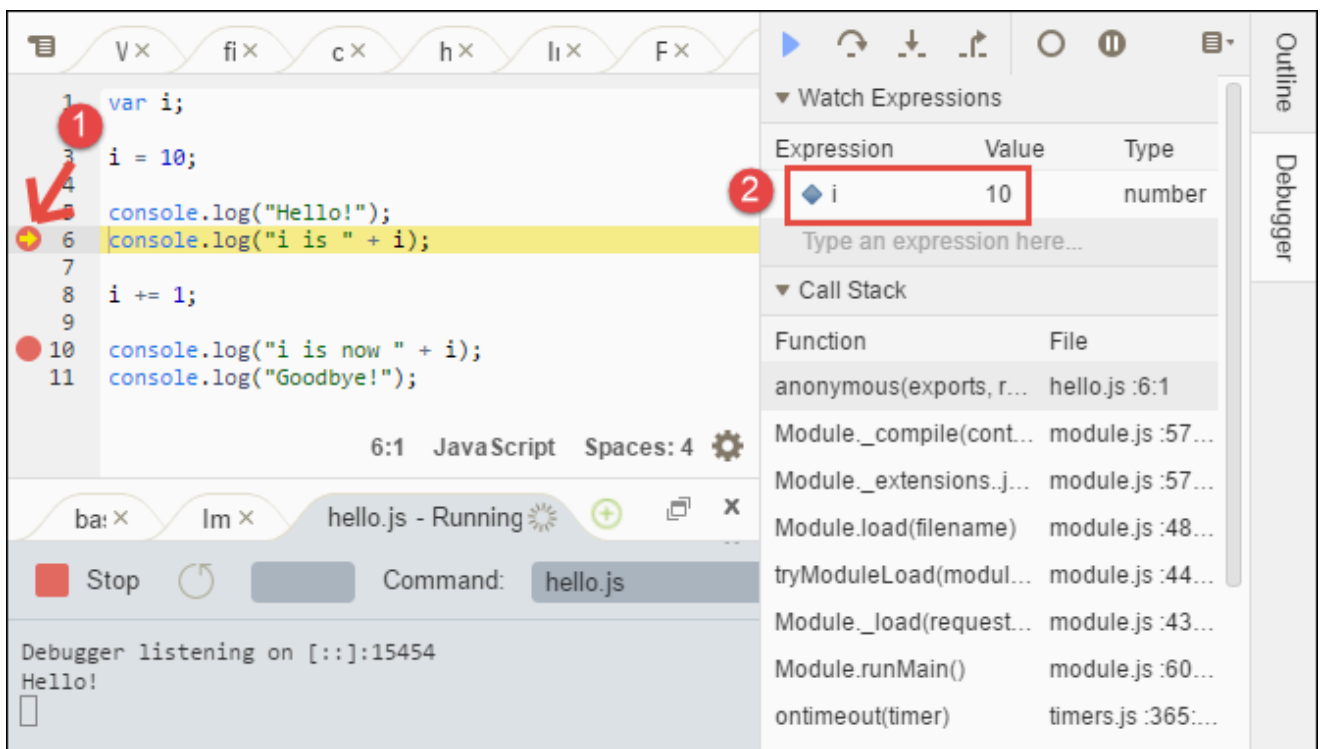
- a. [Debugger (デバッガー)] ウィンドウの内容を表示または非表示にするには、次のステップに示すように [Debugger (デバッガー)] ボタンを選択します。
- b. コードの実行中に `i` という変数の値を確認します。[Debugger (デバッガー)] ウィンドウの [Watch Expressions (ウォッチ式)] で、[Type an expression here (ここに式を入力)] を選択します。次のように文字 `i` を入力して、Enter を押します。



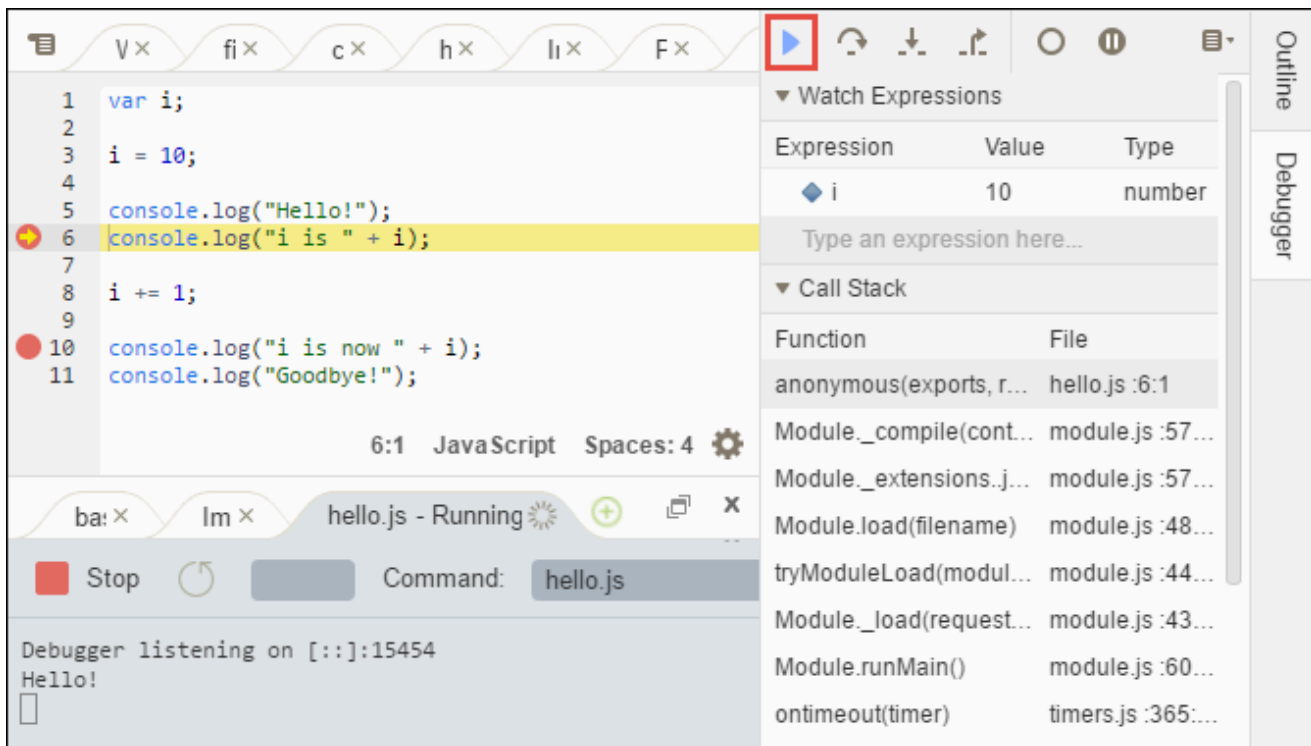
- c. コードの実行を開始します。次のように、[Run (実行)]、[Run With (次で実行)]、[Node.js] の順に選択します。



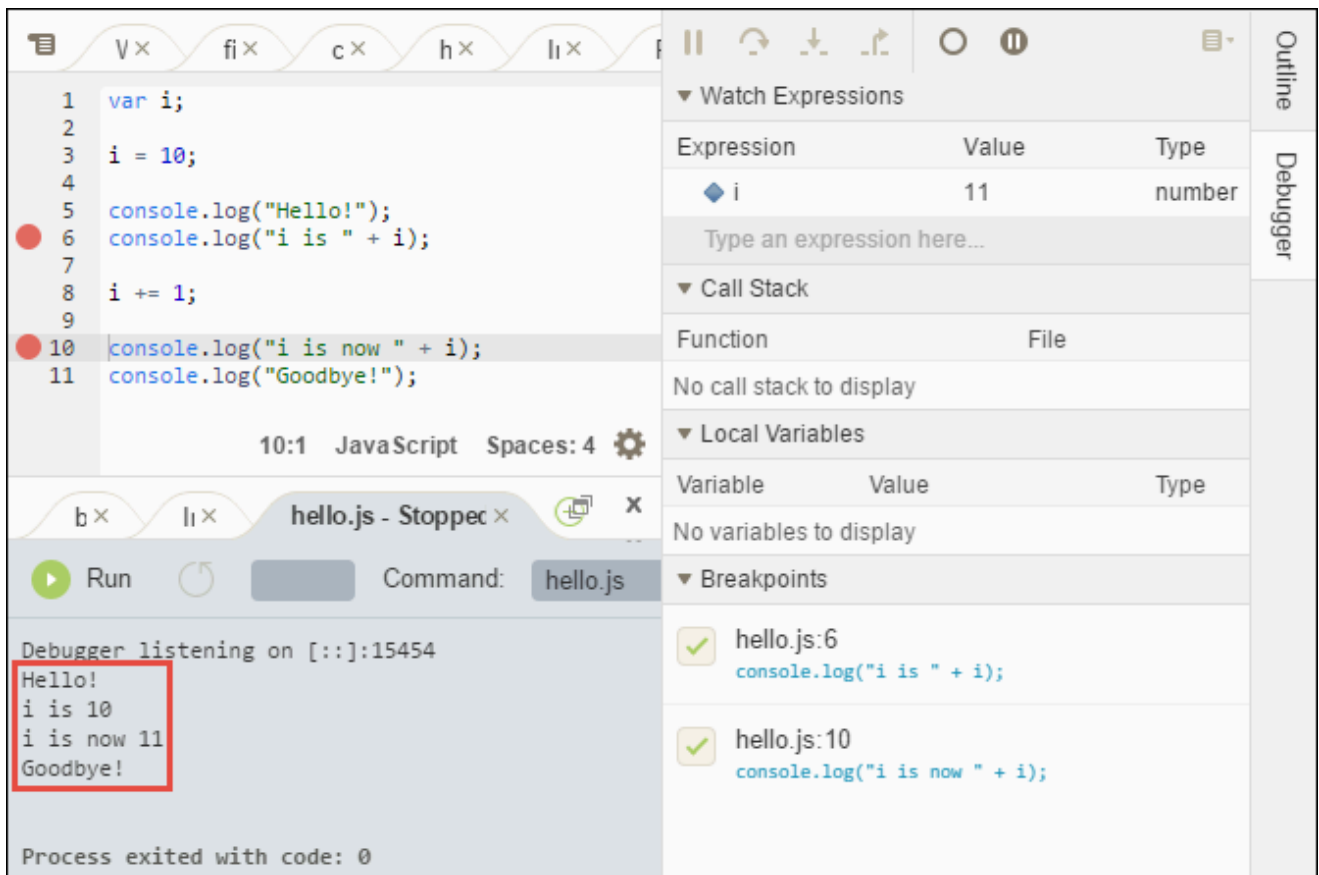
- d. コードは行 6 で実行を一時停止します。[Debugger (デバッガー)] ウィンドウの [iWatch Expressions (表現を見る)] に 値が表示されます。現在は 10 です。



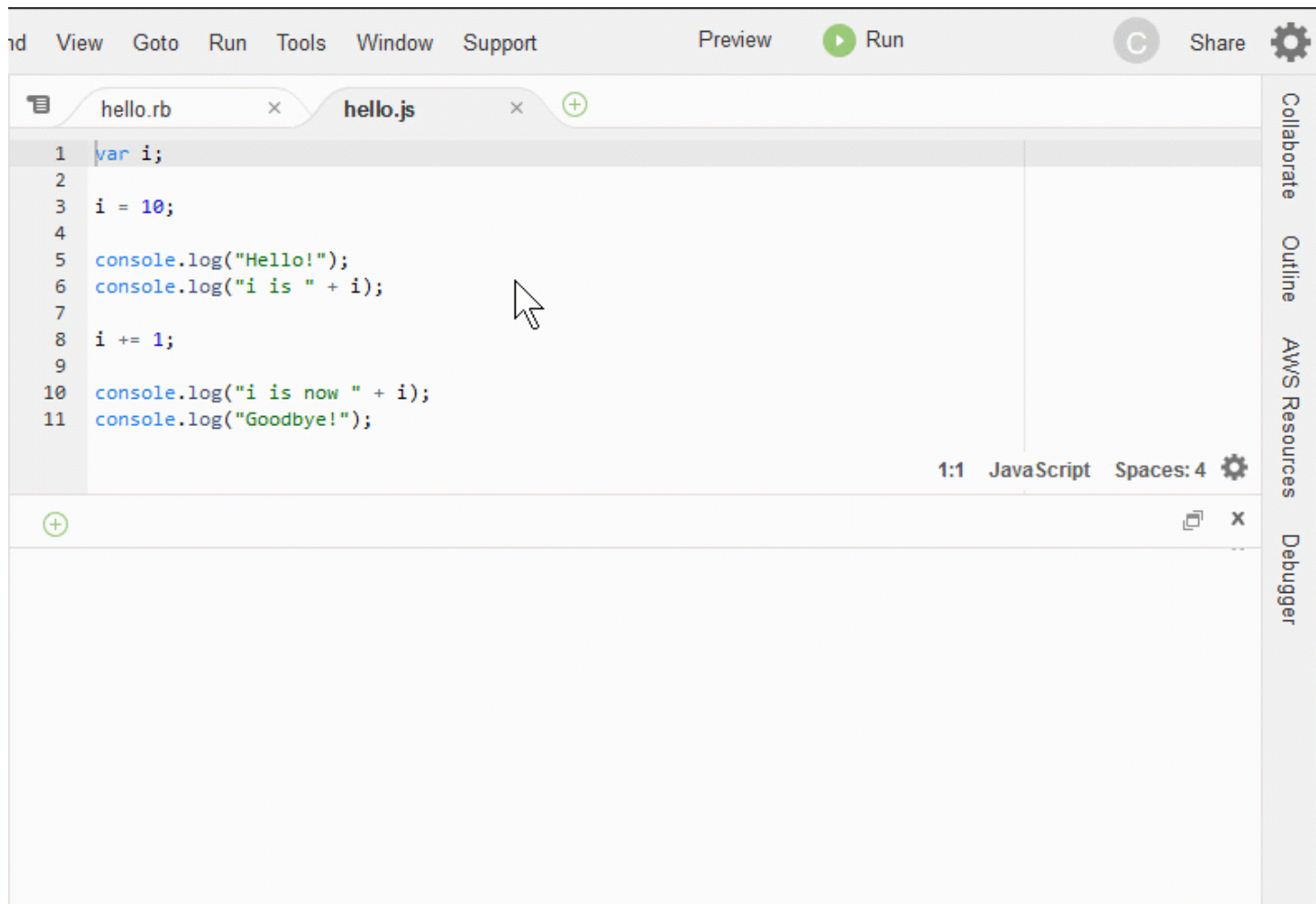
- e. 次のように、[Debugger (デバッガー)] ウィンドウで、[Resume (再開)] を選択します。青い矢印アイコンです。



- f. コードは行 10 で実行を一時停止します。[Debugger (デバッガー)] ウィンドウに、今度は `i` の新しい値が表示されます。現在は 11 です。
- g. [Resume (再開)] を再度選択します。コードが最後まで実行されます。次のように、出力がコンソールの [`hello.js`] タブに表示されます。



結果を以下と比較します。



結論

Warning

AWS Cloud9 開発環境があると、AWS アカウントに料金が発生する可能性があることに注意してください。これらには、EC2 環境を使用している場合の Amazon EC2 の料金が含まれます。詳細については、[Amazon EC2 の料金表](#)を参照してください。

親セクション (「[IDE を操作する](#)」) には、試すことができる追加のトピックがあります。ただし、IDE AWS Cloud9 のツアーが終了し、環境が不要になった場合は、「」で説明されているように、IDE とその関連リソースを削除してください [環境を削除する](#)。

IDE AWS Cloud9 での言語サポート

AWS Cloud9 IDE は多くのプログラミング言語をサポートしています。次の表に、サポートされている言語とそのレベルを示します。

言語	構文のハイライト ¹	UI の実行 ²	アウトライン表示	コードヒントと linting	コードの完了	デバッグ ³
C++	✓	✓	✓		✓ ⁵	✓ ⁴
C#	✓		✓		✓ ⁵	
CoffeeScript	✓	✓				
CSS	✓				✓	
Dart	✓					
Go	✓	✓	✓	✓	✓ ⁴	✓ ⁴
Haskell	✓					
HTML	✓	✓	✓		✓	
Java ⁶	✓	✓	✓	✓	✓	✓
JavaScript	✓	✓	✓	✓	✓	
Node.js	✓	✓	✓	✓	✓	✓
PHP	✓	✓	✓	✓	✓ ⁷	✓
Python	✓	✓	✓	✓	✓ ⁸	✓
Ruby	✓	✓	✓	✓	✓ ⁵	
シェルスクリプト	✓	✓	✓	✓	✓ ⁵	

言語	構文のハイライト ¹	UI の実行 ²	アウトライン表示	コードヒントと linting	コードの完了	デバッグ ³
TypeScript ⁹	✓	✓	✓	✓	✓	

Notes (メモ)

¹ AWS Cloud9 IDE は、さらに多くの言語の構文強調表示を提供します。詳細なリストについては、IDE のメニューバーで、[表示、構文] を選択してください。

² コマンドラインを使用せずに、✓ とマークされた言語のボタンをクリックするだけで、プログラムやスクリプトを実行できます。IDE で ✓ とマークされていない言語、または [Run, Run With] (実行、次で実行) メニューバーに表示されていない言語については、その言語のランナーを作成することができます。手順については、「[ビルダーまたはランナーを作成する](#)」を参照してください。

³ IDE に組み込まれているツールを使用して、✓ でマークされた言語のプログラムやスクリプトをデバッグできます。手順については、「[コードをデバッグする](#)」を参照してください。

⁴ この機能はこの言語で実験段階にあります。これは完全に実装されておらず、文書化もサポートもされていません。

⁵ この機能は、この言語のローカル関数のみをサポートします。

⁶ Java SE 11 機能の拡張サポートは、2 GiB 以上のメモリを持つ AWS Cloud9 EC2 開発環境でアクティブ化できます。詳細については、「[Java 開発のサポートの強化](#)」を参照してください。

⁷ カスタム PHP コードの完了 AWS Cloud9 に使用する のパスを指定するには、IDE AWS Cloud9 でプロジェクト、PHP サポート、Preferences の PHP コード完了設定の有効化を有効にし、カスタムコードへのパスをプロジェクト、PHP サポート、PHP 完了インクルードパス設定に追加します。

⁸ カスタム Python コードの完了 AWS Cloud9 に使用する のパスを指定するには、IDE AWS Cloud9 で、プロジェクト、Python サポート、設定で Python コード完了設定を有効にするを有効にし、カスタムコードへのパスをプロジェクト、Python サポート、PYTHONPATH設定に追加します。

⁹ AWS Cloud9 IDE は、言語プロジェクトのコンテキスト内で TypeScript (IDE でサポートされているバージョン 3.7.5) AWS Cloud9 など、一部のプログラミング言語をさらにサポートします。詳細については、「[言語プロジェクトを操作する](#)」を参照してください。

AWS Cloud9 統合開発環境 (IDE) でサポートされているプログラミング言語バージョン

以下の表は、AWS Cloud9 IDE の特定の AMIs でサポートされているプログラミング言語のバージョンの概要を示しています。Ubuntu 18 は 2023 年に販売終了になったため、プログラミング言語のバージョンを AWS Cloud9 で更新することはできません。

[言語]	Amazon Linux 2023	Amazon Linux 2	Ubuntu 18	Ubuntu 22
Python3	3.9	3.8	3.6	3.10
TypeScript	3.7.5	3.7.5	3.7.5	3.7.5
PHP	8.2	8.2	7.2	8.1
Ruby	3.2	3.0	3.0	3.2
Java	11、17	11	11	11、17
Python2	該当なし	2.7	該当なし	該当なし
C++*	23	17	17	23
Go	1.20	1.20	1.9	1.21
CoffeeScript	2.7	2.7	2.7	2.7

*次のコマンドを実行すると、希望するプログラミング言語のバージョンを使用して C++ ファイルをコンパイルできます。

```
g++ -std=c++[version-number] "$file" -o "$file.o"
```

IDE AWS Cloud9 での言語サポートの強化

AWS Cloud9 では、次の言語でコーディングする際の開発エクスペリエンスを向上させるためのサポートが強化されています。

- Java: 拡張機能では、コード補完、エラーの linting、コンテキスト固有のアクション、デバッグオプションなどの機能が提供されます。
- Typescript: 言語プロジェクトで、TypeScript の生産性向上機能にアクセスできます。

トピック

- [Java 開発のサポートの強化](#)
- [TypeScript のサポートと機能の強化](#)

Java 開発のサポートの強化

AWS Cloud9 は、Java を使用する際の開発エクスペリエンスを向上させるための拡張言語サポートを提供します。生産性に関する主な機能には、コード補完、エラーの linting、コードレンズ、ブレークポイントやステップニングなどのデバッグオプションが含まれます。

Important

強化された生産性機能は、Amazon EC2 インスタンスに接続されている AWS Cloud9 開発環境でのみ使用できます。

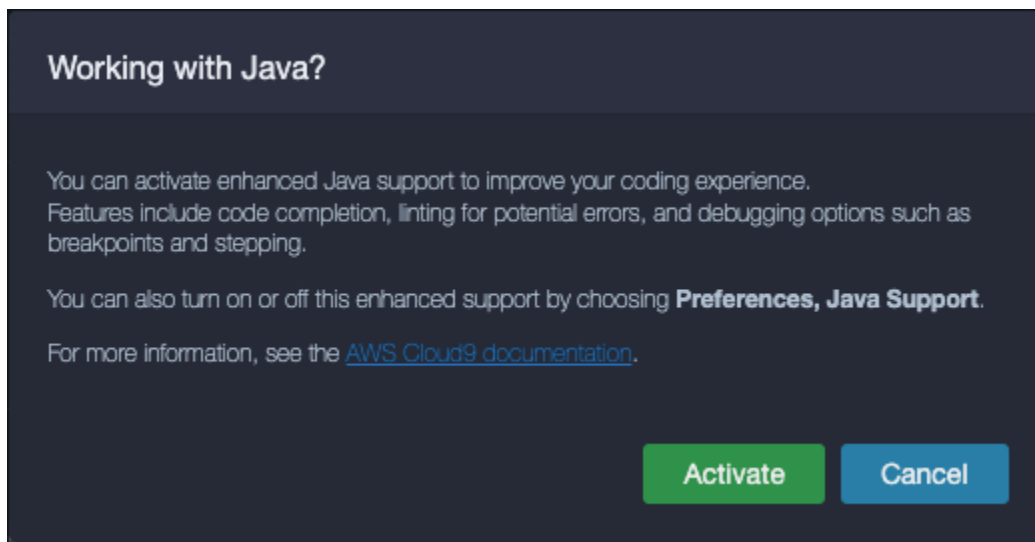
さらに、Java の拡張言語サポートを使用する際に最適な IDE エクスペリエンスを確保するために、AWS Cloud9 環境をサポートする Amazon EC2 コンピューティングインスタンスには 2 GiB 以上のメモリが必要です。が EC2 コンピューティングインスタンスに十分な RAM がないこと AWS Cloud9 を検出した場合、Java の拡張機能をアクティブ化するオプションは提供されません。

Java の拡張サポートの有効化とカスタマイズ

次の条件を満たすと、Java の拡張サポートを有効にするオプションが自動的に表示されます。

- AWS Cloud9 環境が 2 GiB 以上のメモリを持つ Amazon EC2 インスタンスに接続されている。
- Java 開発に関連付けられているファイルを使用しています。は次のファイル名と拡張子 AWS Cloud9 をチェックします: *.java、*.gradle (Gradle ビルドツールに関連)、pom.xml (Apache Maven ビルドツールに関連)。
- 2020 年 12 月 11 日以降に作成された AWS Cloud9 環境で作業している。現在、この日付より前に作成された開発環境で Java の生産性向上機能は使用できません。

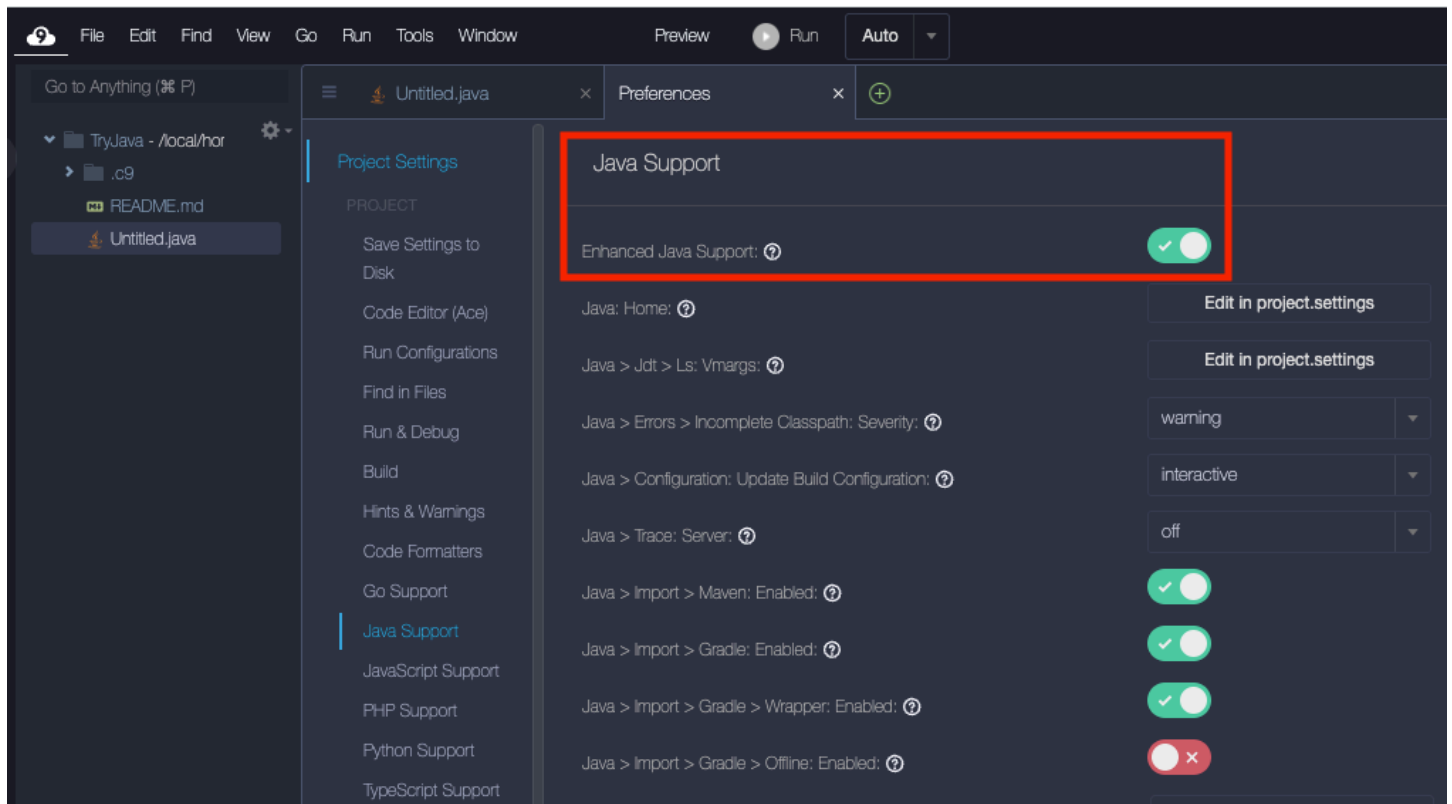
これらの条件が満たされると、Java のコーディングとデバッグのために生産性向上機能を有効にするかどうかを確認するダイアログボックスが表示されます。[Activate] (有効化) を選択すると、IDE でこの機能の使用を開始できます。



Note

AWS Cloud9 環境の作成時に起動される Amazon EC2 インスタンスには、Amazon Corretto 11 が既にインストールされています。Amazon Corretto は、Open Java Development Kit (OpenJDK) の、マルチプラットフォーム対応の本番稼働可能な、無償ディストリビューションです。つまり、AWS Cloud9 out-of-the-box Java アプリケーションの開発と実行を開始できます。

AWS Cloud9 インターフェイスを使用して、拡張言語とデバッグのサポートを手動でアクティブ化または非アクティブ化することもできます。[Preferences] (設定)、[Java Support] (Java サポート)、[Enhanced Java Support] (拡張 Java サポート) の順に選択します。



での Java 開発の拡張サポート AWS Cloud9 は、IDE の 2 つの拡張機能によって提供されます。

- Red Hat による Java(TM) の言語サポート
- Java のデバッガー

AWS Cloud9 インターフェイスを使用すると、これらの拡張機能のパフォーマンスをカスタマイズする幅広い設定にアクセスできます。拡張機能の設定を変更するには、[Preferences] (設定)、[Java Support] (Java サポート) の順に選択します。

これらの設定の詳細については、拡張機能の GitHub リポジトリの以下の関連する ReadMe ページを参照してください。

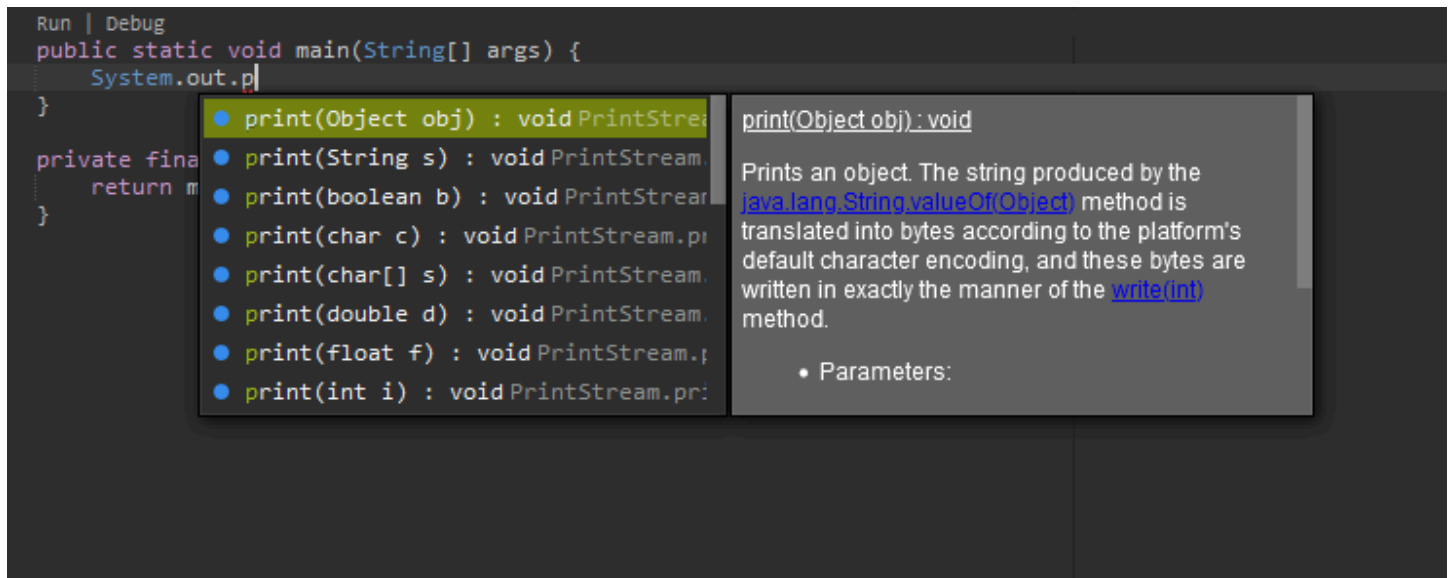
- [Red Hat による Java\(TM\) の言語サポート](#)
- [Java のデバッガー](#)

機能のハイライト

Java の拡張サポートを有効にすると、各種の生産性向上機能を使用できます。

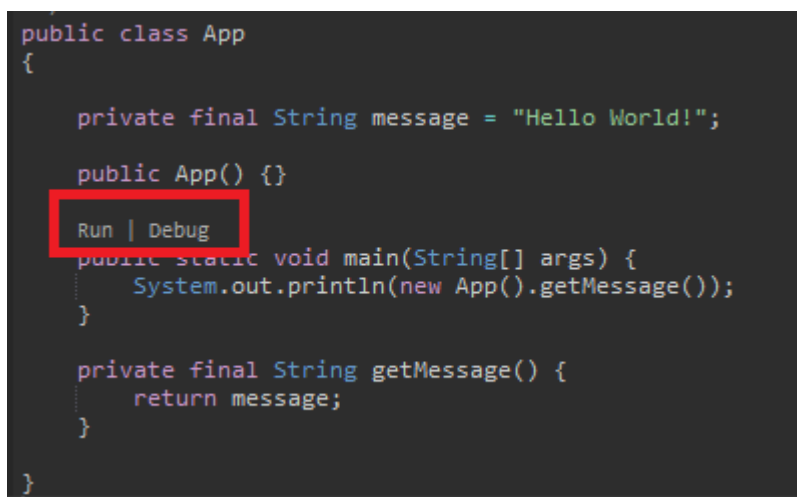
コード補完

コード補完を使用すると、入力しているコードに基づいてコンテキストに対応した候補がエディタに表示されます。例えば、オブジェクト名の後にドット (「.」) 演算子を入力すると、そのオブジェクトで使用可能なメソッドまたはプロパティがエディタに表示されます。



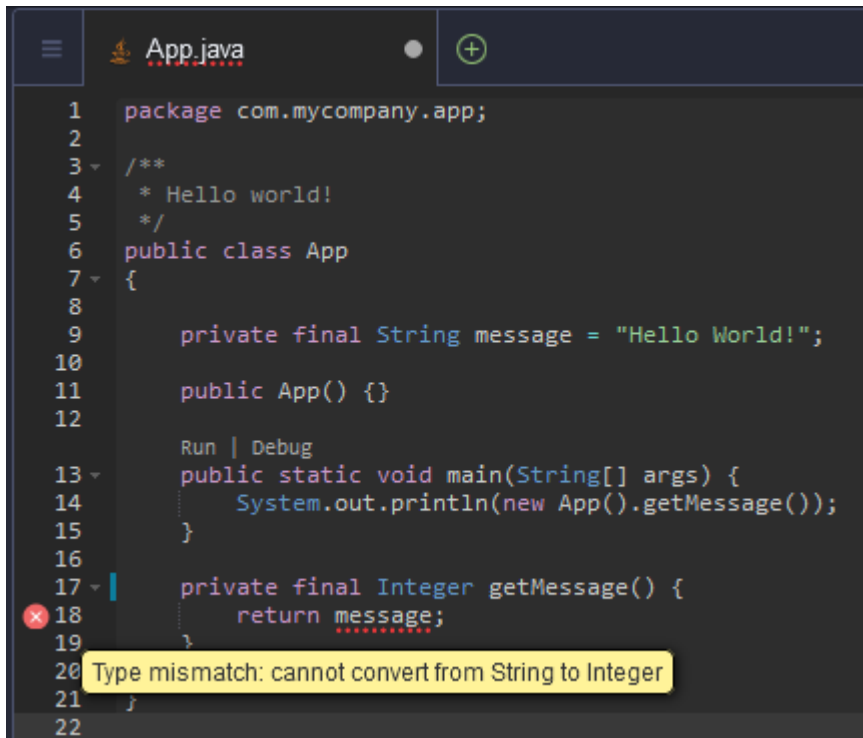
コードレンズ

コードレンズを使用すると、ソースコード内のコンテキスト固有のアクションに直接アクセスできます。Java 開発の場合、コードレンズは特定のメソッドを実行およびデバッグできるようにすることで、単体テストを容易にします。



コードの linting

コードの linting は、ビルドする前にエディタがコード内の潜在的なエラーをどのように強調表示するかを記述します。例えば、初期化されていない変数を使用しようとする場合や、割り当てようとしている値と変数の型が一致しない場合、linting ツールが呼び出されます。

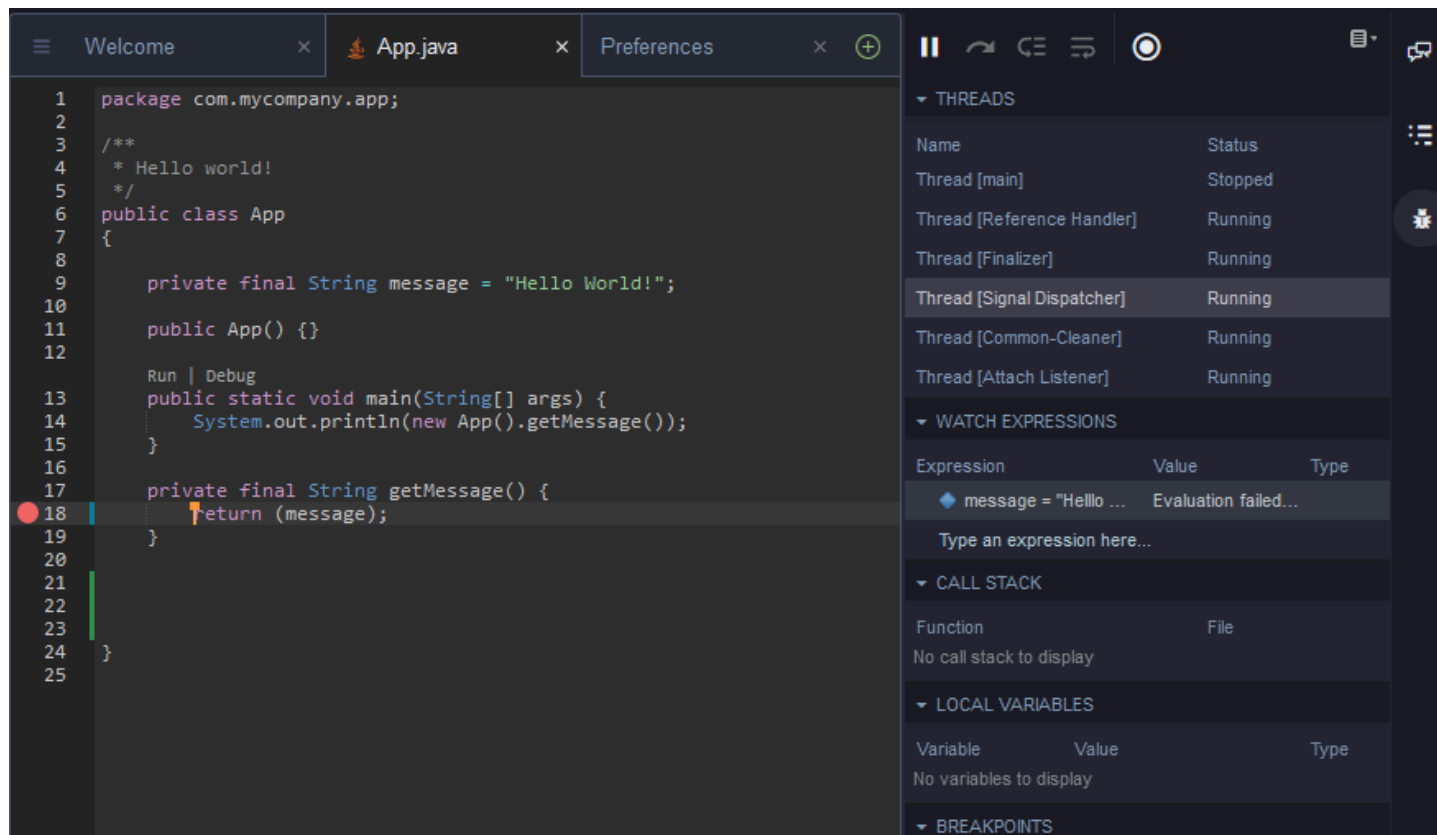


```
1 package com.mycompany.app;
2
3 /**
4  * Hello world!
5  */
6 public class App
7 {
8     private final String message = "Hello World!";
9
10    public App() {}
11
12    Run | Debug
13    public static void main(String[] args) {
14        System.out.println(new App().getMessage());
15    }
16
17    private final Integer getMessage() {
18        return message;
19    }
20 }
21
22
```

Type mismatch: cannot convert from String to Integer

デバッグオプション

ブレークポイントとウォッチ式を実装できます。ソースコードにブレークポイントを設定し、デバッガーペインを表示して関連する条件を定義します。

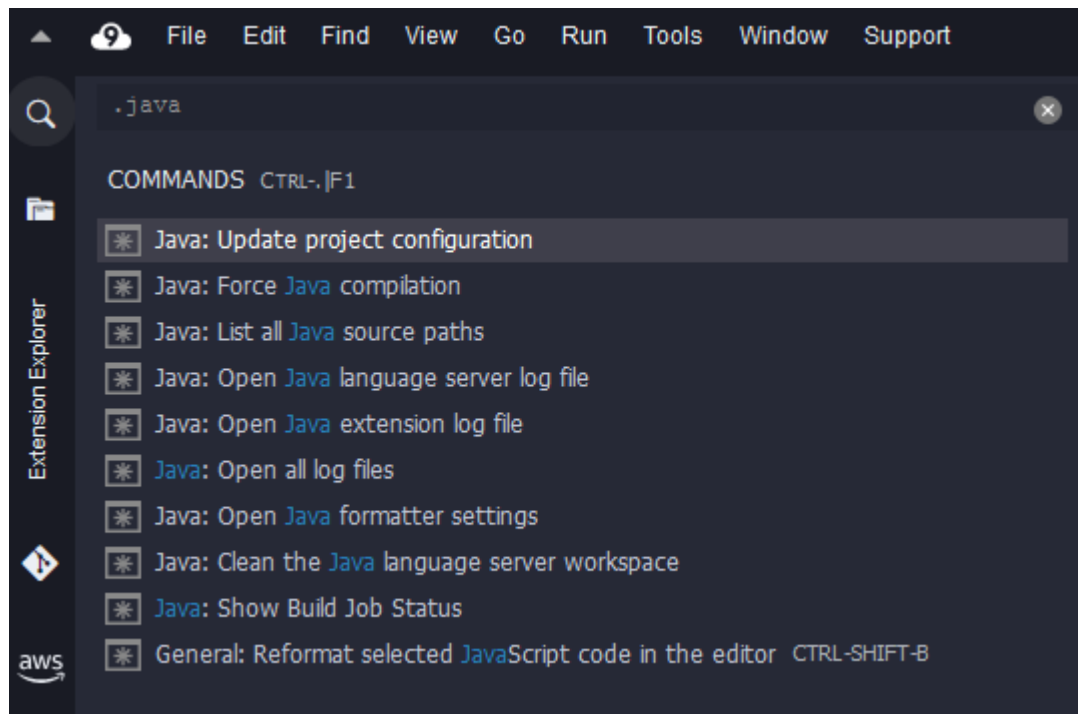


設定ファイルを使用したデバッグ

また、AWS Cloud9 および `launch.json` 設定ファイルを介して `tasks.json` がサポートする起動設定とタスクを使用することで、デバッグ設定を制御することもできます。起動設定とその使用方法の例については、「[Java デバッグ設定](#)」を参照してください。

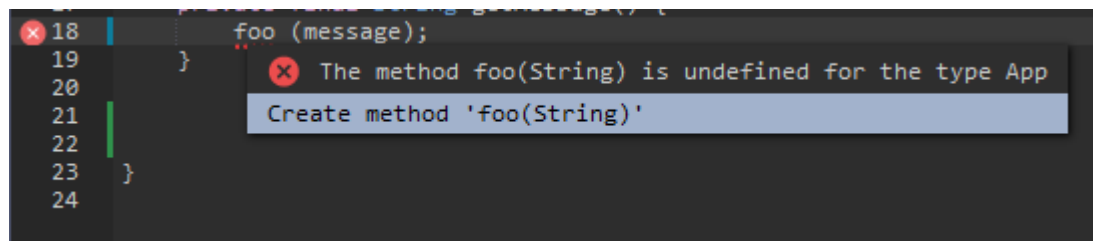
Java コマンド

コマンド AWS Cloud9 パネルからコマンドを実行するには、`Ctrl+.` または `F1` を押します。その後に「java」と入力して、関連するコマンドをフィルタリングします。



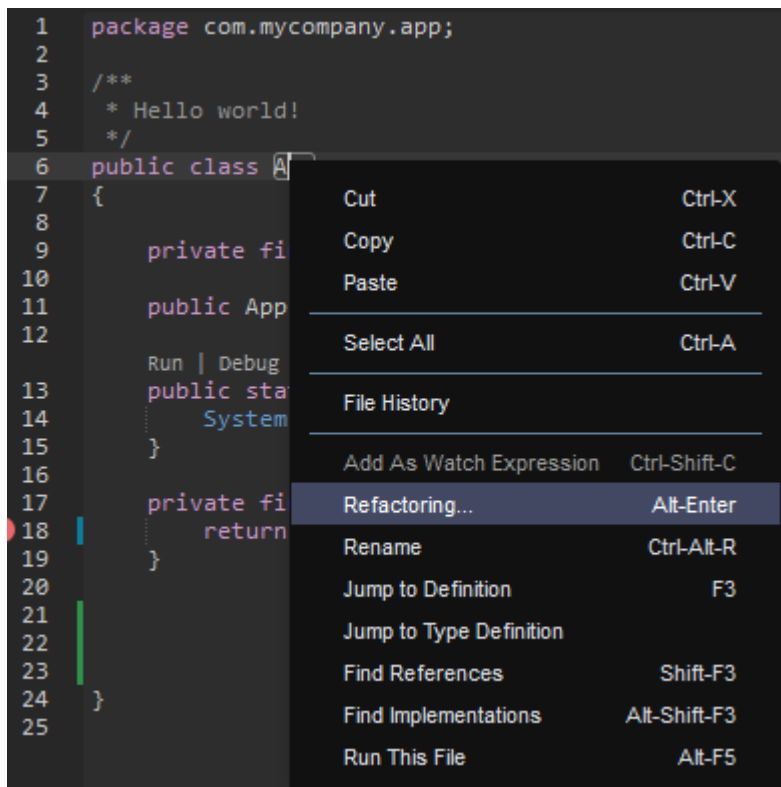
Quick fixes (簡易修正)

簡易修正を使用すると、欠落している要素のスタブを作成して、宣言されていない変数または未定義のメソッドを使用することで発生するエラーを解決できます。



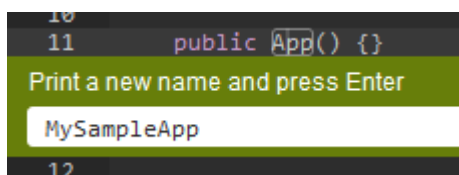
リファクタリング

リファクタリングを使用すると、動作を変更せずにコードを再構築できます。インポートの構成、コンストラクタの作成などのオプションにアクセスするには、項目のコンテキスト (右クリック) メニューを開いて、[Refactoring] (リファクタリング) を選択します。



名前の変更

名前の変更はリファクタリング機能の 1 つです。1 回のアクションで、選択した変数、関数、およびクラスの名前を、コード内で表示されるすべての場所で簡単に変更できます。名前を変更するには、項目のコンテキスト (右クリック) メニューを開き、[Rename] (名前の変更) を選択します。名前の変更は、コード内の名前のすべてのインスタンスに影響します。



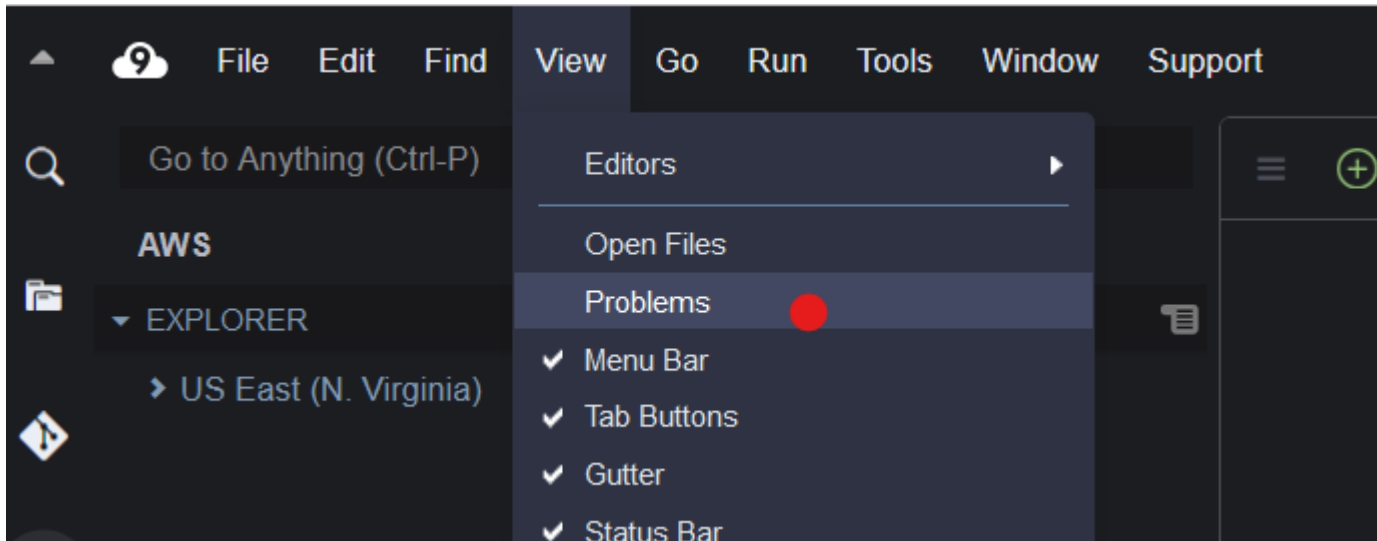
Java 開発用のオプションツール

Java の拡張サポートを提供する拡張機能には、Gradle および Maven オートメーションツールをプロジェクト開発に統合できる機能が含まれています。これらのツールは、AWS Cloud9 開発環境にプリインストールされていません。これらのオプションのビルドツールをインストールして使用方法については、次のリソースを参照してください。

- Gradle: [入門ガイド](#)
- Maven: [5 分で Maven](#)

Java 拡張機能の [問題] タブ

環境内の AWS Cloud9 java プロジェクトに関する問題は、AWS Cloud9 IDE の問題タブで表示およびトラブルシューティングできます。AWS Cloud9 IDE から [問題] タブを表示するには、メニューバーから [表示]、[問題] の順に選択します。



[問題] タブは、コンソール内の [+] アイコンを選択し、[Open Problems] (問題を開く) を選択して開くこともできます。タブから問題を選択すると、影響を受けるファイルが開き、問題の詳細が表示されます。

TypeScript のサポートと機能の強化

IDE を使用すると、言語プロジェクトを使用して TypeScript AWS Cloud9 の強化された生産性向上機能にアクセスできます。言語プロジェクトは、AWS Cloud9 開発環境の IDE 内の関連ファイル、フォルダ、および設定のコレクションです。

IDE を使用して環境で言語プロジェクトを作成するには、「[言語プロジェクトの作成](#)」を参照してください。

使用可能なプロジェクト生産性向上機能

IDE には、TypeScript AWS Cloud9 の以下のプロジェクト生産性機能が用意されています。

Autocomplete

エディタでファイルに入力するときに、コンテキストに応じた記号 (ある場合) のリストが挿入ポイントに表示されます。

リストの記号を挿入ポイントに挿入するには、上矢印キーまたは下矢印キーを使用して記号を選択し (まだ選択されていない場合)、Tab キーを押します。

Tab キーを押す前に、選択した記号に関するヒント情報が表示されます。

記号を挿入しないでリストを閉じるには、Esc キーを押します。

ガーターアイコン

アクティブなファイルでガーターにアイコンが表示される場合があります。これらのアイコンは、コードを実行する前にコード内に問題が検出された場合、これらの問題を警告やエラーとして示します。

問題の詳細を確認するには、問題のアイコン上でポインタを一時停止します。

Quick Fixes (簡易修正)

エディタのアクティブなファイルで、コーディングのエラーおよび警告に関する情報とそのコードに自動的に適用できる有効な修正を表示できます。エラーまたは警告の情報と有効な修正を表示するには、赤の点線の下線 (エラーの場合) またはグレーの点線の下線 (警告の場合) があるコードの一部を選択します。または、赤またはグレーの点線の下線があるコードにカーソルを置いたまま、Option-Enter (macOS の場合) または Alt-Enter (Linux または Windows の場合) を押します。提案された修正を適用するには、リストの修正を選択するか、矢印キーを使用して修正を選択して Enter キーを押します。簡易修正のオン/オフをマウスのクリックで切り替えるには、AWS Cloud9、[設定]、[User Settings (ユーザー設定)]、[言語]、[Hints & Warnings (ヒントと警告)]、[Show Available Quick Fixes on Click (クリックで利用可能な簡易修正を表示)] の順に選択します。

Find References (リファレンスの検索)

エディタのアクティブなファイルで、挿入ポイントにある記号に関するすべてのリファレンスを表示できます (これらのリファレンスに IDE からアクセスできる場合)。

これを行うには、挿入ポイントになる記号の内部で **Find References** コマンドを実行します。例:

- 挿入ポイントで右クリックし、[Find References (リファレンスの検索)] を選択します。
- メニューバーで、[Go (移動)]、[Find References (リファレンスの検索)] の順に選択します。
- macOS、Windows、Linux で、デフォルトの Shift-F3 を押します。

リファレンスが利用可能である場合は、アクティブなファイル上にペインが (記号の横に) 表示されます。ペインには、この記号のリファレンスがあるファイルのリストが含まれています。ペインには

リストの最初のリファレンスが表示されます。別のリファレンスを表示するには、そのリファレンスをリストで選択します。

ペインを閉じるには、ペインの閉じる (X) アイコンを選択するか、Esc キーを押します。

以下の場合、**Find References** コマンドが無効であるか、正常に機能しない場合があります。

- アクティブファイルのプロジェクトに、その記号に関するリファレンスがない場合。
- アクティブなファイルのプロジェクトで、IDE が記号のリファレンスの一部またはすべてを見つけない場合。
- IDE が、アクティブなファイルのプロジェクトで記号のリファレンス場所にアクセスできない場合。

Go to Definition (定義に移動)

エディタのアクティブなファイルで、記号からその記号の定義場所に移動できます (IDE がその定義にアクセスできる場合)。

これを行うには、挿入ポイントになる記号の内部で **Jump to Definition** コマンドを実行します。例:

- 挿入ポイントで右クリックし、[Jump to Definition (定義に移動)] を選択します。
- メニューバーで、[Go (移動)]、[Jump to Definition (定義に移動)] を選択します。
- macOS、Windows、Linux で、デフォルトの F3 を押します。

定義が利用可能な場合は、その定義が別のファイルにある場合でも、挿入ポイントに定義が表示されます。

以下の場合、**Jump to Definition** コマンドが無効であるか、正常に機能しない場合があります。

- 記号が該当言語のプリミティブシンボルである場合。
- アクティブなファイルのプロジェクトで IDE が定義場所を見つられない場合。
- アクティブなファイルのプロジェクトで IDE が定義場所にアクセスできない場合。

Go to Symbol (記号に移動)

プロジェクト内の特定の記号に移動できます。方法は次のとおりです。

1. プロジェクトのいずれかのファイルをエディタで開いてアクティブにします。ファイルがすでに開いている場合は、そのタブをエディタで選択し、ファイルをアクティブにします。
2. **Go to Symbol** コマンドを実行します。例:
 - ウィンドウの [Go (移動)] ボタン (虫眼鏡アイコン) を選択します。[Go to Anything (どこにでも移動)] ボックスに、「@」と入力し、記号の入力を開始します。
 - メニューバーで、[Go (移動)]、[Go To Symbol (記号に移動)] を選択します。[Go (移動)] ウィンドウで、[@] に続けて記号の入力を開始します。
 - macOS でデフォルトの Command-2 または Command-Shift-0 を押します。Windows/Linux ではデフォルトの Ctrl-Shift-0 を押します。[Go (移動)] ウィンドウで、[@] に続けて記号の入力を開始します。

例えば、プロジェクトで toString という名前の記号をすべてを見つけるには、@toString の入力を開始します (または、@ が既に表示されている場合、@ の後に toString の入力を開始します)。
3. 目的の記号が [Symbols (記号)] リストに表示されている場合は、その記号をクリックして選択します。または上矢印キー/下矢印キーを使用して記号を選択し、Enter キーを押します。挿入ポイントがその記号に切り替わります。

移動先の記号がアクティブなファイルのプロジェクトに存在しない場合、この手順は正常に動作しないことがあります。

言語プロジェクトの作成

AWS Cloud9 IDE でサポートされているプロジェクトの生産性向上機能で動作する言語プロジェクトを作成するには、次の手順に従います。

Note

サポートされているプロジェクト生産性向上機能は、言語プロジェクトの一部であるファイルでを使用することをお勧めします。サポートされているプロジェクト生産性向上機能は、言語プロジェクトの一部ではないファイルでも使用できますが、予期しない結果になることがあります。

たとえば、IDE を使用して、プロジェクトのパートではない環境のルートレベルにあるファイル内からリファレンスや定義を検索するとします。この場合、IDE が検索する対象は同じルートレベルにあるファイルに限定される可能性があります。その結果、同じ環境内でどこ

かにある言語プロジェクトで実際にリファレンスや定義が存在するにもかかわらず、リファレンスや定義が見つからないという結果になる場合があります。

TypeScript 言語プロジェクトの作成

1. 環境に TypeScript がインストールされていることを確認します。詳細については、「[ステップ 1: 必要なツールをインストールする](#)」の「[の TypeScript チュートリアル AWS Cloud9](#)」を参照してください。
2. IDE における環境のターミナルセッションから、プロジェクトを作成したいディレクトリに切り替えます。このディレクトリが存在しない場合は、作成してから切り替えます。たとえば、以下のコマンドでは、(~/environment 内の)環境のルートにディレクトリ (my-demo-project) を作成し、このディレクトリに切り替えます。

```
mkdir ~/environment/my-demo-project
cd ~/environment/my-demo-project
```

3. プロジェクトを作成する先のディレクトリのルートで、**--init** オプションを指定して TypeScript コンパイラーを実行します。

```
tsc --init
```

このコマンドが成功すると、TypeScript コンパイラーにより、プロジェクトのディレクトリのルートに `tsconfig.json` ファイルが作成されます。このファイルを使用してさまざまなプロジェクト設定を定義できます。たとえば、TypeScript コンパイラーのオプション、プロジェクトに含めるファイル、プロジェクトから除外するファイルなどを定義できます。

`tsconfig.json` の詳細については、以下を参照してください。

- [tsconfig.json Overview](#) (TypeScript ウェブサイト)
- [tsconfig.json Schema](#) (json.schemastore.org ウェブサイト)

IDE AWS Cloud9 のメニューバーコマンドリファレンス

次のリストでは、AWS Cloud9 IDE のデフォルトのメニューバーコマンドについて説明します。メニューバーが表示されていない場合は、IDE の上端にある細いバーを選択して表示します。

- [AWS Cloud9 メニュー](#)

- [\[File\] \(ファイル\) メニュー](#)
- [\[Edit\] \(編集\) メニュー](#)
- [\[Find\] \(検索\) メニュー](#)
- [\[View\] \(表示\) メニュー](#)
- [\[Go\] \(移動\) メニュー](#)
- [\[Run\] \(実行\) メニュー](#)
- [\[Tools\] \(ツール\) メニュー](#)
- [\[Window\] \(ウィンドウ\) メニュー](#)
- [\[Support\] \(サポート\) メニュー](#)
- [\[Preview\] \(プレビュー\) メニュー](#)
- [他のメニューバーコマンド](#)

AWS Cloud9 メニュー

コマンド	説明
詳細設定	次のいずれかを行います： • [Preferences (設定)] タブが開いていない場合は開きます。 • [Preferences (設定)] タブが開いていてもアクティブでない場合は、アクティブにします。 • [Preferences (設定)] タブがアクティブの場合は非表示にします。 「プロジェクト設定の操作」、「ユーザー設定の操作」、「キー割り当ての操作」、「テーマの操作」、および「初期化スクリプトの操作」を参照してください。
Go To Your Dashboard (ダッシュボードを開く)	別のウェブブラウザタブで AWS Cloud9 コンソールを開きます。 「環境を作成する」 、 「環

コマンド	説明
	境を開く 」、「 環境設定の変更 」、および「 環境の削除 」を参照してください。
ようこそページ	[Welcome (ようこそ)] タブを開きます。
Open Your Project Settings (プロジェクト設定を開く)	現在の環境の <code>project.settings</code> ファイルを開きます。「 プロジェクト設定の操作 」を参照してください。
Open Your User Settings (ユーザー設定を開く)	現在のユーザーの <code>user.settings</code> ファイルを開きます。「 ユーザー設定の操作 」を参照してください。
Open Your Keymap (キー割り当てを開く)	現在のユーザーの <code>keybindings.settings</code> ファイルを開きます。「 キー割り当ての操作 」を参照してください。
Open Your Init Script (Init スクリプトを開く)	現在のユーザーの <code>init.js</code> ファイルを開きます。「 初期化スクリプトの操作 」を参照してください。
Open Your Stylesheet (スタイルシートを開く)	現在のユーザーの <code>styles.css</code> ファイルを開きます。「 テーマの操作 」を参照してください。

[File] (ファイル) メニュー

コマンド	説明
New File (新しいファイル)	新しいファイルを作成します。
New From Template (テンプレートから新規作成)	選択したファイルテンプレートに基づいて新しいファイルを作成します。
開く	[Navigate (移動)] ウィンドウを表示して移動します。

コマンド	説明
[Open Recent] (最近使用したファイル)	選択したファイルを開きます。
[Save] (保存)	現在のファイルを保存します。
Save As (名前を付けて保存)	現在のファイルを別のファイル名で、または別の場所に、または別のファイル名で別の場所に保存します。
[Save All] (すべて保存)	未保存のファイルをすべて保存します。
Revert to Saved (最後に保存したファイルに戻す)	現在のファイルが最後に保存されてから以降の変更を破棄します。
Revert All to Saved (最後に保存したファイルにすべて戻す)	ファイルが最後に保存されてから、未保存のすべての変更を破棄します。
Show File Revision History (ファイルリビジョン履歴の表示)	エディタで現在のファイルの変更を表示および管理します。「 ファイルリビジョンの操作 」を参照してください。
Upload Local Files (ローカルファイルのアップロード)	[Upload Files (ファイルをアップロードする)] ダイアログボックスを表示します。ここで、ローカルコンピュータから環境にファイルをドラッグすることができます。
Download Project (プロジェクトのダウンロード)	環境内のファイルを .zip ファイルに結合します。これはローカルコンピュータにダウンロードできます。
Line Endings (行末処理)	[Windows] (キャリッジリターンとラインフィード) または [Unix] (ラインフィードのみ) の行末処理を行います。
Close File (ファイルを閉じる)	現在のファイルを閉じます。
Close All Files (すべてのファイルを閉じる)	開いているすべてのファイルを閉じます。

[Edit] (編集) メニュー

コマンド	説明
Undo	前回のアクションを元に戻します。
[Redo] (再実行)	最後に取り消したアクションをやり直します。
Cut (切り取る)	選択したものをクリップボードに移動します。
[Copy] (コピー)	選択したものをクリップボードにコピーします。
[Paste] (貼り付ける)	クリップボードの内容を選択ポイントにコピーします。
Keyboard Mode (キーボードモード)	使用するキー割り当てのセット (Default、Vim、Emacs、または Sublime など) です。「 キー割り当ての操作 」を参照してください。
Selection (選択)、Select All (すべて選択)	選択可能な内容をすべて選択します。
Selection (選択)、Split Into Lines (行に分割)	現在の行の末尾にカーソルを追加します。
Selection (選択)、Single Selection (単一選択)	以前の選択をすべてクリアします。
Selection, Multiple Selections, Add Cursor Up (選択、複数選択、カーソルを上追加)	アクティブなカーソルの 1 行上にカーソルを追加します。既にカーソルが追加されている場合は、そのカーソルの上に別のカーソルを追加します。
Selection (選択)、Multiple Selections (複数選択)、Add Cursor Down (カーソルを下追加)	アクティブなカーソルの 1 行下にカーソルを追加します。既にカーソルが追加されている場合は、そのカーソルの下に別のカーソルを追加します。
Selection (選択)、Multiple Selections (複数選択)、Move Active Cursor Up (アクティブなカーソルを上移動)	アクティブなカーソルの 1 行上に 2 番目のカーソルを追加します。2 番目のカーソルが既

コマンド	説明
	に追加されている場合は、2 番目のカーソルを 1 行うに移動します。
Selection (選択)、Multiple Selections (複数選択)、Move Active Cursor Down (アクティブなカーソルを下に移動)	アクティブなカーソルの 1 行下に 2 番目のカーソルを追加します。2 番目のカーソルが既に追加されている場合は、2 番目のカーソルを 1 行上に移動します。
Selection (選択)、Multiple Selections (複数選択)、Add Next Selection Match (次の選択一致を追加)	選択部分以降で一致する選択部分を含めます。
[選択]、[Multiple Selections (複数選択)]、[Add Previous Selection Match (前の一致する選択を追加)]	選択部分以前で一致する選択部分を含めます。
Selection (選択)、Multiple Selections (複数選択)、Merge Selection Range (選択範囲をマージ)	現在の行の末尾にカーソルを追加します。
Selection (選択)、Select Word Right (右の単語を選択)	カーソル右にある次の単語までを選択に含めます。
Selection (選択)、Select Word Left (左の単語を選択)	カーソル左にある次の単語までを選択に含めます。
Selection (選択)、Select to Line End (行末まで選択)	カーソルから現在の行の末尾までを選択に含める
Selection (選択)、Select to Line Start (行頭まで選択)	現在の行の行頭からカーソルまでを選択に含めます。
Selection (選択)、Select to Document End (ドキュメントの最後まで選択)	カーソルから現在のファイルの末尾までを選択に含めます。
Selection (選択)、Select to Document Start (ドキュメントの最初まで選択)	カーソルから現在のファイルの先頭までを選択に含めます。

コマンド	説明
Line (行)、Indent (インデント)	選択部分を 1 タブ分インデントします。
Line (行)、Outdent (アウトデント)	選択部分を 1 タブ分アウトデントします。
Line (行)、Move Line Up (1 行上に移動)	選択範囲を 1 行上に移動します。
Line (行)、Move Line Down (1 行下に移動)	選択範囲を 1 行下に移動します。
Line (行)、Copy Lines Up (行を上コピー)	行の内容をコピーし、1 つ上の行にコピーした内容を貼り付けます。
Line (行)、Copy Lines Down (行を下コピー)	行の内容をコピーし、1 つ下の行にコピーした内容を貼り付けます。
Line (行)、Remove Line (行を削除)	現在の行の内容を削除します。
Line (行)、Remove to Line End (行末まで削除)	カーソルから現在の行の末尾までを削除します。
Line (行)、Remove to Line Start (行頭まで削除)	現在の行の行頭からカーソルまでを削除します。
Line (行)、Split Line (行を分割)	カーソルの内容を行の末尾に移動し、独自の行にします。
Text (テキスト)、Remove Word Right (右の単語を削除)	カーソルの右にある単語を削除します。
Text (テキスト)、Remove Word Left (左の単語を削除)	カーソルの左にある単語を削除します。
Text (テキスト)、Align (配置)	行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動します (正しく配置されていない場合)。
Text (テキスト)、Transpose Letters (文字の入れ替え)	選択部分を移動します。

コマンド	説明
Text (テキスト)、To Upper Case (大文字に変換)	選択部分をすべて大文字に変更します。
Text (テキスト)、To Lower Case (小文字に変換)	選択部分をすべて小文字に変更します。
Comment (コメント)、Toggle Comment (コメントの切り替え)	選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除します。
Code Folding (コードの折りたたみ)、Toggle Fold (折りたたみの切り替え)	コードを折りたたむ、またはコードの折りたたみがある場合は削除します。
Code Folding (コードの折りたたみ)、Unfold (展開)	選択されたコードを展開します。
Code Folding (コードの折りたたみ)、Fold Other (他を折りたたむ)	現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたみます。
Code Folding (コードの折りたたみ)、Fold All (すべて折りたたむ)	折りたたむことができる要素をすべて折りたたみます。
Code Folding (コードの折りたたみ)、Unfold All (すべて展開)	ファイル全体でコードの折りたたみを展開します。
Code Formatting (コードフォーマット)、Apply Code Formatting (コードフォーマットの適用)	選択した JavaScript コードを再フォーマットします。
Code Formatting (コードフォーマット)、Open Language & Formatting Preferences (言語とフォーマットの設定を開く)	[設定] タブの [Project Settings (プロジェクト設定)] セクションを開き、プログラミング言語を設定します。

[Find] (検索) メニュー

詳細については、「[テキストの検索と置き換え](#)」を参照してください。

コマンド	説明
Find (検索)	現在のドキュメントの検索と置換バーを表示し、[Find (検索)] 式にフォーカスします。
Find Next (次を検索)	現在のドキュメントで最後に入力した検索クエリの次の一致に移動します。
Find Previous (前を検索)	現在のドキュメントで最後に入力した検索クエリの前の一致に移動します。
Replace (置換)	現在のドキュメントの検索と置換バーを表示し、[Replace With (置換後の文字列)] 式にフォーカスします。
Replace Next (次と置き換え)	現在のドキュメントの検索と置換のバーで [Find (検索)] の次の一致項目を [Replace With (置換後の文字列)] に置き換えます。
Replace Previous (前と置き換え)	現在のドキュメントの検索と置換のバーで [Find (検索)] の前の一致項目を [Replace With (置換後の文字列)] に置き換えます。
Replace All (すべて置換)	現在のドキュメントの検索と置換のバーで [Find (検索)] のすべての一致項目を [Replace With (置換後の文字列)] に置き換えます。
ファイルの検索	複数のファイルで検索と置換バーを表示します。

[View] (表示) メニュー

コマンド	説明
Editors (エディタ)	選択したエディタを表示します。

コマンド	説明
Open Files (開いているファイル)	[環境] ウィンドウに [Open Files (開いているファイル)] リストを表示するか、表示されている場合は非表示にします。
問題点	環境の Java プロジェクトの問題は、ターミナルの [問題] パネルで表示できます。問題を選択してターゲットファイルを開くことができます。
[メニューバー]	メニューバーを表示するか、表示されている場合は非表示にします。
Tab Buttons (タブボタン)	タブを表示するか、表示される場合は非表示にします。
Gutter (ガーター)	ガーターを表示するか、表示されている場合は非表示にします。
Status Bar (ステータスバー)	ステータスバーを表示するか、表示されている場合は非表示にします。
コンソール	[Console (コンソール)] ウィンドウを表示するか、表示されている場合は非表示にします。
Layout (レイアウト)、Single (単一)	単一のペインを表示します。
Layout (レイアウト)、Vertical Split (垂直分割)	上と下の 2 つのペインを表示します。
Layout (レイアウト)、Horizontal Split (水平分割)	2 つのペインを並べて表示します。
Layout (レイアウト)、Cross Split (クロス分割)	同じサイズの 4 つのペインを表示します。
Layout (レイアウト)、Split 1:2 (1:2 分割)	左側に 1 つのペインを表示し、右側に 2 つのペインを表示します。
Layout (レイアウト)、Split 2:1 (2:1 分割)	左側に 2 つのペインを表示し、右側に 1 つのペインを表示します。

コマンド	説明
Font Size (フォントサイズ)、Increase Font Size (フォントサイズを大きくする)	フォントサイズを大きくします。
Font Size (フォントサイズ)、Decrease Font Size (フォントサイズを小さくする)	フォントサイズを小さくします。
[Syntax] (構文)	現在のドキュメントの構文タイプを表示します。
Themes (テーマ)	IDE テーマのタイプを表示します。
Wrap Lines (行の折り返し)	現在のペインの端に単語を折り返すか、すでに折り返している場合は折り返しを停止します。
Wrap To Print Margin (印刷マージンに折り返す)]	現在の印刷マージンに単語を折り返すか、すでに折り返している場合は折り返しを停止します。

[Go] (移動) メニュー

コマンド	説明
Go To Anything (どこにでも移動)	[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示します。
Go To Symbol (記号に移動)	[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する
Go To File (ファイルに移動)	[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する
Go To Command (コマンドに移動)	[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示します。
Go To Line (行に移動)	[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示します。

コマンド	説明
Next Error (次のエラー)	次のエラーに移動します。
Previous Error (前のエラー)	前のエラーに移動します。
Word Right (右の単語)	1 つ右の単語に移動します。
Word Left (左の単語)	1 つ左の単語に移動します。
Line End (行末)	現在の行の末尾に移動します。
Line Start (行頭)	現在の行の行頭に移動します。
Jump to Definition (定義に移動)	変数の定義またはカーソルがある関数に移動します。
Jump to Matching Brace (一致する括弧に移動)	現在の範囲内の一致する記号に移動します。
Scroll to Selection (選択範囲にスクロール)	選択範囲が見やすいようにスクロールして移動します。

[Run] (実行) メニュー

コマンド	説明
実行	現在のアプリケーションを実行またはデバッグします。
Run Last (直前を実行)	最後に実行したファイルを実行またはデバッグします。
Run With (次で実行)	選択したランナーを使用して実行またはデバッグします。「 ビルダー、ランナー、デバッガーの操作 」を参照してください。
Run History (実行履歴)	実行履歴を表示します。

コマンド	説明
実行設定	実行設定を選択して実行またはデバッグするか、実行設定を作成または管理します。「 ビルダー、ランナー、デバッガーの操作 」を参照してください。
Show Debugger at Break] (ブレーク時にデバッガを表示)	実行中のコードがブレークポイントに達すると、[Debugger (デバッガー)] ウィンドウが表示されます。
Build	現在のファイルをビルドします。
Cancel Build (ビルドのキャンセル)	現在のファイルのビルドを停止します。
Build System (ビルドシステム)	選択されたビルドシステムを使用して構築します。
Show Build Result (ビルド結果の表示)	関連するビルド結果を表示します。
Automatically Build Supported Files (サポートされるファイルを自動的にビルドする)	サポートされるファイルを自動的にビルドします。
Save All on Build (ビルド時にすべてを保存)	ビルド時に、保存されていない関連ファイルをすべて保存します。

[Tools] (ツール) メニュー

コマンド	説明
Strip Trailing Space (末尾のスペースを削除)	行末の空白を削除します。
Preview (プレビュー)、Preview File (ファイルのプレビュー)	現在のドキュメントを [Preview (プレビュー)] タブでプレビューします。
[Preview (プレビュー)]、Preview Running Application (実行中のアプリケーションのプレビュー)]	現在のアプリケーションを別のウェブブラウザのタブでプレビューします。

コマンド	説明
Preview (プレビュー)、Configure Preview URL (プレビュー URL の設定)	[設定] タブの [Project Settings (プロジェクト設定)] セクションを開き、[実行とデバッグ]、[Preview URL (URL のプレビュー)] を設定します。
Preview (プレビュー)、Show Active Servers (アクティブサーバーの表示)	[Process List (プロセスリスト)] ダイアログボックスに利用可能なアクティブサーバーのアドレスを一覧表示します。
Process List (プロセスリスト)	[Process List (プロセスリスト)] ダイアログボックスを表示します。
Show Autocomplete (自動補完を表示)	コード補完のコンテキストメニューを表示します。
Rename Variable (変数名の変更)	選択部分の名前変更のリファクタリングを開始します。
Toggle Macro Recording (マクロ記録の切り替え)	キーストロークの記録を開始する、または既に記録している場合は停止します。
Play Macro (マクロの再生)	以前に記録されたキーストロークを再生します。

[Window] (ウィンドウ) メニュー

コマンド	説明
Go	[Go (移動)] ウィンドウを表示するか、表示されている場合は非表示にします。
New Terminal (新しいターミナル)	新しい [Terminal (ターミナル)] タブを開きます。
New Immediate Window (新しい即時ウィンドウ)	新しい [Immediate (即時)] タブを開きます。

コマンド	説明
共有	[Share this environment (この環境を共有する)] ダイアログボックスを表示します。
Installer (インストーラ)	[AWS Cloud9 Installer (インストーラー)] ダイアログボックスを表示します。
Collaborate (コラボレーション)	[Collaborate (コラボレーション)] ウィンドウを表示するか、表示されている場合は非表示にします。
Outline (アウトライン)	[Outline (アウトライン)] ウィンドウを表示するか、表示されている場合は非表示にします。
AWS リソース	[AWS リソース] ウィンドウを表示するか、表示されている場合は非表示にします。
環境	[Environment (環境)] ウィンドウを表示するか、表示されている場合は非表示にします。
Debugger (デバッガー)	[Debugger (デバッガー)] ウィンドウを表示するか、表示されている場合は非表示にします。
Navigation (移動)、Tab to the Right (タブで右に移動)	1 タブ分だけ右へ移動します。
Navigation (移動)、Tab to the Left (左のタブに移動)	1 タブ分だけ左に移動します。
Navigation (移動)、Next Tab in History (履歴の次のタブ)	次のタブに移動します。
Navigation (移動)、Previous Tab in History (履歴の前のタブ)	前のタブに移動します。

コマンド	説明
Navigation (移動)、Move Tab to Right (タブを右に移動)	現在のタブを右に移動します。タブが既に右端にある場合は、そこに分割タブを作成します。
Navigation (移動)、Move Tab to Left (タブを左に移動)	現在のタブを左に移動します。タブが既に左端にある場合は、そこに分割タブを作成します。
Navigation (移動)、Move Tab to Up (タブを上を移動)	現在のタブを 1 つ上に移動します。タブが既に上端にある場合は、そこに分割タブを作成します。
Navigation (移動)、Move Tab to Down (タブを下に移動)	現在のタブを 1 つ下に移動します。タブが既に下端にある場合は、そこに分割タブを作成します。
Navigation (移動)、Go to Pane to Right (ペイン 1 つ分を右に移動)	1 ペイン分だけ右に移動します。
Navigation (移動)、Go to Pane to Left (ペイン 1 つ分を左に移動)	1 ペイン分だけ左に移動します。
Navigation (移動)、Go to Pane to Up (ペイン 1 つ分を上を移動)	1 ペイン分だけ上に移動します。
Navigation (移動)、Go to Pane to Down (ペイン 1 つ分を下を移動)	1 ペイン分だけ下に移動します。
Navigation (移動)、Switch Between Editor and Terminal (エディタとターミナルの切り替え)	エディタと [Terminal (ターミナル)] タブを切り替えます。
Navigation (移動)、Next Pane in History (履歴の次のペイン)	次のペインに移動します。
Navigation (移動)、Previous Pane in History (履歴の前のペイン)	前のペインに移動します。

コマンド	説明
Saved Layouts (保存済みレイアウト)、Save (保存)	現在のレイアウトを保存します。このレイアウトに後で切り替えるには、[Saved Layouts (保存済みレイアウト)]、[LAYOUT-ID (レイアウト ID)] を選択します。
Saved Layouts (保存済みレイアウト)、Save and Close All (保存してすべて閉じる)	現在のレイアウトを保存し、すべてのタブとペインを閉じます。
Saved Layouts (保存済みレイアウト)、Show Saved Layouts in File Tree (保存済みレイアウトをファイルツリーで表示)	[Environment (環境)] ウィンドウの保存済みレイアウトをすべて表示します。
Tab (タブ)、Close Pane (ペインを閉じる)	現在のペインを閉じます。
Tab (タブ)、Close All Tabs In All Panes (すべてのペインでタブをすべて閉じる)	すべてのペインで開いているタブをすべて閉じます。
Tab (タブ)、Close All But Current Tab (現在のタブを除いてすべてのタブを閉じる)	現在のペインで現在のタブ以外の開いているすべてのタブを閉じます。
Tab (タブ)、Split Pane in Two Rows (ペインを 2 行に分割)	現在のペインを上下 2 つのペインに分割します。
Tab (タブ)、Split Pane in Two Columns (ペインを 2 列に分割)	現在のペインを左右 2 つのペインに分割します。
Presets (プリセット)、Full IDE (完全な IDE)	完全な IDE モードに切り替えます。
Presets (プリセット)、Minimal Editor (最小エディタ)	最小エディタモードに切り替えます。
Presets (プリセット)、Sublime Mode (Sublime モード)	Sublime モードに切り替えます。

[Support] (サポート) メニュー

コマンド	説明
ようこそページ	[Welcome (ようこそ)] タブを開きます。
[Get Help (Community) (ヘルプ (コミュニティ))	別のウェブブラウザタブで AWS Cloud9 オンラインコミュニティのウェブサイトを開きます。
Read Documentation (ドキュメントを読む)	AWS Cloud9 A ユーザーガイドを別のウェブブラウザタブで開きます。

[Preview] (プレビュー) メニュー

コマンド	説明
Preview File (ファイルのプレビュー)	現在のドキュメントを [Preview (プレビュー)] タブでプレビューします。
Preview Running Application (実行中のアプリケーションのプレビュー)	現在のアプリケーションを別のウェブブラウザのタブでプレビューします。
Configure Preview URL (プレビュー URL の設定)	[設定] タブの [Project Settings (プロジェクト設定)] セクションを開き、[実行とデバッグ]、[Preview URL (URL のプレビュー)] を設定します。
Show Active Servers (アクティブサーバーの表示)	[Process List (プロセスリスト)] ダイアログボックスに利用可能なアクティブサーバーのアドレスを一覧表示します。

他のメニューバーコマンド

コマンド	説明
実行	現在のアプリケーションを実行またはデバッグします。
共有	[Share this environment (この環境を共有する)] ダイアログボックスを開きます。
Preferences (設定) (歯車アイコン)	[Preferences (設定)] タブを開きます。

IDE AWS Cloud9 でのテキストの検索と置換

AWS Cloud9 統合開発環境 (IDE) の find and replace バーを使用して、1 つまたは複数のファイル内のテキストを検索して置き換えることができます。

- [1 つのファイルでテキストを検索する](#)
- [1 つのファイルでテキストを置き換える](#)
- [複数のファイルでテキストを検索する](#)
- [複数のファイルでテキストを置き換える](#)
- [検索と置換のオプション](#)

1 つのファイルでテキストを検索する

1. テキストを検索するファイルを開きます。ファイルがすでに開いている場合は、ファイルのタブを選択してそのファイルをアクティブにします。
2. メニューバーで [File (ファイル)]、[Find (検索)] の順に選択します。
3. 検索と置換のバーで、[Find (検索)] に検索するテキストを入力します。
4. 追加の検索オプションを指定するには、「[検索と置換のオプション](#)」を参照してください。
5. 一致するものがある場合、[検索] ボックスの 0 of 0 (0/0) はゼロ以外の数字に変わります。一致するものがある場合は、エディターは最初の一致項目に進みます。複数の一致がある場合、次の一致に進むには、[Find (検索)] ボックスで右矢印を選択するか、メニューバーで [Find (検索)]、[Find Next (次を検索)] を選択します。前の一致に進むには、[Find (検索)] ボックスで左矢印を選択するか、メニューバーで [Find (検索)]、[Find Previous (前を検索)] を選択します。

1 つのファイルでテキストを置き換える

1. テキストを置き換えるファイルを開きます。ファイルがすでに開いている場合は、ファイルのタブを選択してそのファイルをアクティブにします。
2. メニューバーで [Find (検索)]、[Replace (置換)] の順に選択します。
3. 検索と置換のバーで、[Find (検索)] に検索するテキストを入力します。
4. [Replace With (置換後の文字列)] で、[Find (検索)] に置き換えるテキストを入力します。
5. 追加の検索と置換のオプションを指定するには、「[検索と置換のオプション](#)」を参照してください。
6. 一致するものがある場合、[検索] ボックスの 0 of 0 (0/0) はゼロ以外の数字に変わります。一致するものがある場合は、エディターは最初の一致項目に進みます。複数の一致がある場合、次の一致に進むには、[Find (検索)] ボックスで右矢印を選択するか、メニューバーで [Find (検索)]、[Find Next (次を検索)] を選択します。前の一致に進むには、[Find (検索)] ボックスで左矢印を選択するか、メニューバーで [Find (検索)]、[Find Previous (前を検索)] を選択します。
7. 現在の一致を [Replace With (置換後の文字列)] のテキストに置き換えて次の一致に移動するには、[Replace (置換)] を選択します。すべての一致を [Replace With (置換後の文字列)] のテキストに置き換えるには [Replace All (すべて置換)] を選択します。

複数のファイルでテキストを検索する

1. メニューバーで [Find (検索)]、[Find in Files (ファイルの検索)] の順に選択します。
2. 検索と置換のバーで、[Find (検索)] に検索するテキストを入力します。
3. 追加の検索オプションを指定するには、「[検索と置換のオプション](#)」を参照してください。
4. [Find (検索)] ボタンの右側にあるボックス (*.*, -.*) のあるボックス) で、検索に含める、または除外するファイルのセットを入力します。例:
 - 空白、*、または *.*: すべてのファイルを検索します。
 - my-file.txt: my-file.txt という名前のファイルのみを検索します。
 - my*: my で始まるファイル名のファイルのみを検索します。
 - my*.txt: my で始まり、ファイル拡張子 .txt を持つファイル名のファイルのみを検索します。
 - my*.htm*: ファイル名が my で始まり、.htm で始まるファイル拡張子を持つすべてのファイルを検索します。

- `my*.htm`, `my*.html`: ファイル名が `my` で始まり、ファイル拡張子 `.htm` または `.html` を持つすべてのファイルを検索します。
 - `-my-file.txt`: `my-file.txt` という名前のファイルは検索しません。
 - `-my*`: `my` で始まるファイルは検索しません。
 - `-my*.htm*`: ファイル名が `my` で始まり、`.htm` で始まるファイル拡張子を持つファイルは検索しません。
 - `my*.htm*`, `-my*.html`: ファイル名が `my` で始まり、`.htm` で始まるファイル拡張子を持つすべてのファイルを検索します。ただし、ファイル名が `my` で始まり、拡張子が `.html` であるファイルは検索しません。
5. 前のボックスの横にあるドロップダウンリストで、次のいずれかを選択して、検索範囲を特定の場所のみにさらに制限します。
- [Environment (環境)]: [Environment (環境)] ウィンドウ内のファイルのみを検索します。
 - [プロジェクト (excludes.gitignore'd)]: `.gitignore` ファイルが存在する場合、環境内の `.gitignore` ファイルにリストされているファイルまたはファイルタイプを除く、環境内のファイルを検索します。
 - [Selection: (選択:)]: [Environment (環境)] ウィンドウで現在選択されているファイルのみを検索します。
- 
- Note**
- 単一のフォルダにのみ検索をさらに制限するには、[Environment (環境)] ウィンドウでフォルダを選択し、[Selection (選択)] を選択します。または、[Environment (環境)] ウィンドウでフォルダを右クリックし、コンテキストメニューで [Search In This Folder (このフォルダで検索)] を選択します。
- [お気に入り]: [環境] ウィンドウの [お気に入り] リスト内のファイルのみを検索します。
 - [Active File (アクティブなファイル)]: アクティブなファイルのみを検索します。
 - [Open Files (開いているファイル)]: [環境] ウィンドウの [Open Files (開いているファイル)] リスト内のファイルのみを検索します。
6. [Find (検索)] を選択します。
7. 一致するファイルに移動するには、[Search Results (検索結果)] タブでファイル名をダブルクリックします。特定の一致項目に移動するには、[Search Results (検索結果)] タブでその一致項目をダブルクリックします。

複数のファイルでテキストを置き換える

1. メニューバーで [Find (検索)]、[Find in Files (ファイルの検索)] の順に選択します。
2. 検索と置換のバーで、[Find (検索)] に検索するテキストを入力します。
3. 追加の検索オプションを指定するには、「[検索と置換のオプション](#)」を参照してください。
4. [Find (検索)] ボタンの右側にあるボックス (*.*, -.*) のあるボックス) で、検索に含める、または除外するファイルのセットを入力します。例:
 - 空白、*、または *.*: すべてのファイルを対象とします。
 - my-file.txt: my-file.txt という名前のファイルのみを対象とします。
 - my*: my で始まるファイル名のファイルのみを対象とします。
 - my*.txt: my で始まり、ファイル拡張子 .txt を持つファイル名のファイルのみを対象とします。
 - my*.htm*: ファイル名が my で始まり、.htm で始まるファイル拡張子を持つすべてのファイルを対象とします。
 - my*.htm, my*.html: ファイル名が my で始まり、.htm または .html のファイル拡張子を持つすべてのファイルを対象とします。
 - -my-file.txt: my-file.txt という名前のファイルは検索しません。
 - -my*: my で始まるファイルは検索しません。
 - -my*.htm*: ファイル名が my で始まり、.htm で始まるファイル拡張子を持つファイルは検索しません。
 - my*.htm*, -my*.html: ファイル名が my で始まり、.htm で始まるファイル拡張子を持つすべてのファイルを検索します。ただし、ファイル名が my で始まり、拡張子が .html であるファイルは検索しません。
5. 前のボックスの横にあるドロップダウンリストで、次のいずれかを選択して、検索範囲を特定の場所のみにさらに制限します。
 - [Environment (環境)]: [Environment (環境)] ウィンドウ内のファイルのみを対象とします。
 - [プロジェクト (excludes.gitignore'd)]: .gitignore ファイルが存在する場合、.gitignore ファイルにリストされているファイルまたはファイルタイプを除く、環境内のファイルを対象とします。
 - [Selection: / (選択: /)]: 現在選択されているすべてのファイルのみを対象とします。
 - [お気に入り]: [環境] ウィンドウの [お気に入り] リスト内のファイルのみを対象とします。
 - ~~Active File (アクティブなファイル): アクティブなファイルのみを対象とします。~~

- [Open Files (開いているファイル)]: [Environment (環境)] ウィンドウの [Open Files (開いているファイル)] リスト内のファイルのみを対象とします。
6. [Replace With (置換後の文字列)] で、置き換えるテキストを [Find (検索)] に入力します。
 7. [Replace (置換)] を選択します。

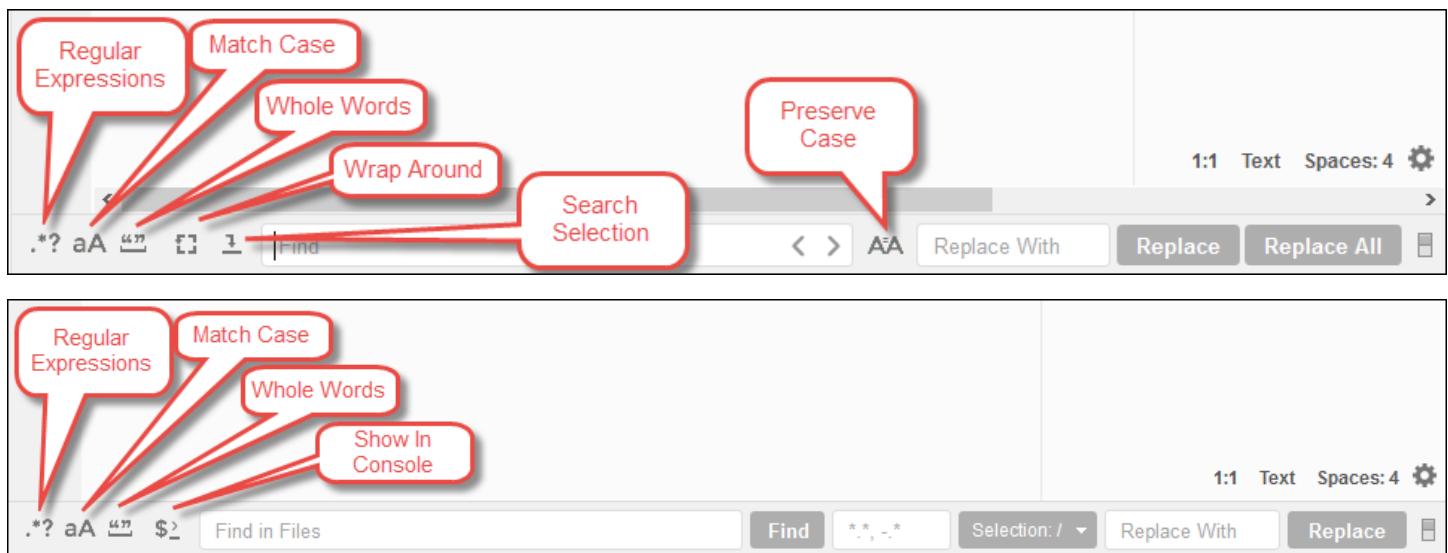
Note

置換操作は、範囲内のすべてのファイルで直ちに行われます。この操作は簡単に元に戻すことができません。置換操作を開始する前に何が変更されるのかを見たい場合は、代わりに [Find (検索)] を選択します。

8. 置換するファイルに移動するには、[Search Results (検索結果)] タブでファイル名をダブルクリックします。特定の置換項目に移動するには、[Search Results (検索結果)] タブでその置換項目をダブルクリックします。

検索と置換のオプション

検索および置換操作を変更するには、検索および置換バーの次のいずれかのボタンを選択します。



- Regular Expressions (正規表現): [Find (検索)] または [Find in Files (ファイルの検索)] で指定された正規表現に一致するテキストを検索します。詳細については、Mozilla 開発者ネットワークに掲載されている [JavaScript 正規表現](#) の「Writing a regular expression pattern」のトピックを参照してください。

- [Match Case (大文字と小文字を区別): [Find (検索)] または [Find in Files (ファイルの検索)] で指定された大文字と小文字に一致するテキストを検索します。
- [Whole Words (語全体) (語全体)]: [Find (検索)] または [Find in Files (ファイルの検索)] の標準の文字ルールを使用してテキストを検索する
- Wrap Around (折り返す): 1 つのファイルのみの検索で、次または前の一致に進むときにファイルの最後または先頭で停止しません。
- Search Selection (選択範囲の検索): 1 つのファイルのみの検索で、選択範囲内でのみ検索します。
- [Show Console (コンソールを表示)]: 複数のファイルの検索で、アクティブなペインの代わりに、[コンソール] に [検索結果] タブを表示します。
- [Preserve Case (大文字と小文字を保持)]: 1 つのファイルのみの検索で、テキストを置き換えるときに適用される大文字と小文字の区別を保持します。

IDE AWS Cloud9 でのファイルのプレビュー

IDE を使用して、IDE AWS Cloud9 内から AWS Cloud9 開発環境内のファイルをプレビューできます。

- [プレビューするファイルを開く](#)
- [ファイルプレビューの再ロード](#)
- [ファイルプレビュータイプの変更](#)
- [ウェブブラウザの別のタブでファイルプレビューを開く](#)
- [別のファイルプレビューに切り替える](#)

プレビューするファイルを開く

IDE AWS Cloud9 で次のいずれかのオプションを選択して、環境内のファイルプレビュータブを開きます。

- [環境] でプレビューするファイルを右クリックしてコンテキストメニューを開き、[プレビュー] を選択します。

Note

この方法でどのようなファイルもプレビューできますが、次のファイル拡張子を持つファイルのプレビューに最適です。

- .htm
- .html
- .pdf
- .svg
- .xhtml
- Markdown 形式のコンテンツを含むファイル。

- 次のいずれかのファイル拡張子を持つファイルを開きます。
 - .pdf
 - .svg
- プレビューするファイルが既に関開かれていて使用中の場合、メニューバーで、[Preview (プレビュー)]、[Preview File (ファイルのプレビュー)]、[FILE_NAME] の順に選択します。または、[Tools (ツール)]、[Preview (プレビュー)]、[Preview File (ファイルのプレビュー)]、[FILE_NAME] の順に選択します。[FILE_NAME] はプレビューするファイルの名前です。

Note

これらのコマンドは、以下のファイルタイプでのみ動作します。

- .htm
- .html
- .markdown
- .md
- .pdf
- .svg
- .txt: ファイルのコンテンツが Markdown 形式の場合はプレビューがよく機能します。
- .xhtml: ファイルにコンテンツプレゼンテーション情報が含まれているか情報を参照している場合はプレビューがよく機能します。

Note

ファイルプレビュータブの [Preview Settings (プレビュー設定)] メニューは現在機能していません。どのメニューコマンドを選択しても効果はありません。

ファイルプレビューの再ロード

ファイルプレビュータブで、[Refresh (最新情報に更新)] ボタン (円状の矢印) を選択します。

ファイルプレビュータイプの変更

ファイルプレビュータブで、プレビュータイプリストから以下のいずれかを選択します。

- **Browser (ブラウザ):** ウェブブラウザ形式でファイルをプレビューします (以下のファイルタイプのみ)。
 - .htm
 - .html
 - .pdf
 - .svg
 - .xhtml: ファイルにコンテンツプレゼンテーション情報が含まれているか情報を参照している場合はプレビューがよく機能します。
- **Raw Content (UTF-8) (未加工コンテンツ (UTF-8)):** ファイルのオリジナルコンテンツを Unicode Transformation Format 8-bit (UTF-8) 形式でプレビューします。一部のファイルタイプでは予期しないコンテンツが表示される場合があります。
- **Markdown:** Markdown 形式を含むファイルをプレビューします。その他のファイルタイプをプレビューしようとすると、予期しないコンテンツが表示される場合があります。

ウェブブラウザの別のタブでファイルプレビューを開く

ファイルプレビュータブで、[Pop Out Into New Window (新しいウィンドウで開く)] をクリックします。

別のファイルプレビューに切り替える

ファイルプレビュータブで、アドレスバーに別のファイルへのパスを入力します。アドレスバーは [Refresh (最新の情報に更新)] ボタンとプレビュータイプリストの間にあります。

IDE AWS Cloud9 で実行中のアプリケーションをプレビューする

AWS Cloud9 統合開発環境 (IDE) を使用して、IDE 内から実行中のアプリケーションをプレビューできます。

アプリケーションの実行

IDE 内からアプリケーションをプレビューする前に、アプリケーションが AWS Cloud9 開発環境で実行されている必要があります。以下のポートで HTTP を使用する必要があります。

- 8080
- 8081
- 8082

上記のすべてのポートでは、IP アドレスとして 127.0.0.1、localhost、0.0.0.0 のいずれかを必要する必要があります。

Note

IP アドレスとして 127.0.0.1、localhost、または 0.0.0.0 を使用し、ポート 8080、8081、または 8082 を経由して HTTP でアプリケーションを実行することは必須ではありません。ただし、そうしないと、IDE 内から実行中のアプリケーションをプレビューすることはできません。

Note

プレビューアプリケーションは IDE 内で実行され、iframe 要素内にロードされます。一部のアプリケーションサーバーは、X-Frame-Options ヘッダーなどの iframe 要素からのリクエストをデフォルトでブロックする場合があります。プレビューアプリケーションがプレビュータブに表示されない場合は、アプリケーションサーバーが iframe のコンテンツの表示を禁止していないことを確認してください。

特定のポートと IP アドレスでアプリケーションを実行するコードを記述するには、アプリケーションのドキュメントを参照してください。

アプリケーションを実行するには、「[コードの実行](#)」を参照してください。

この動作をテストするには、環境のルートで server.js という名前のファイルに、次の JavaScript コードを追加します。このコードは、Node.js という名前のファイルを使用してサーバーを実行します。

 Note

次の例で、text/html は返されたコンテンツの Content-Type です。コンテンツを別の形式で返すには、別の Content-Type を指定します。例えば、CSS ファイル形式には text/css を指定できます。

```
var http = require('http');
var fs = require('fs');
var url = require('url');

http.createServer( function (request, response) {
  var pathname = url.parse(request.url).pathname;
  console.log("Trying to find '" + pathname.substr(1) + "'...");

  fs.readFile(pathname.substr(1), function (err, data) {
    if (err) {
      response.writeHead(404, {'Content-Type': 'text/html'});
      response.write("ERROR: Cannot find '" + pathname.substr(1) + "'.");
      console.log("ERROR: Cannot find '" + pathname.substr(1) + "'.");
    } else {
      console.log("Found '" + pathname.substr(1) + "'.");
      response.writeHead(200, {'Content-Type': 'text/html'});
      response.write(data.toString());
    }
    response.end();
  });
}).listen(8080, 'localhost'); // Or 8081 or 8082 instead of 8080. Or '127.0.0.1'
                              instead of 'localhost'.
```

環境のルートで、server.py などの名前のファイルに次の Python コードを追加できます。次の例で、サーバーは Python を使用して実行されます。

```
import os
import http.server
import socketserver

ip = 'localhost' # Or '127.0.0.1' instead of 'localhost'.
port = '8080' # Or '8081' or '8082' instead of '8080'.
Handler = http.server.SimpleHTTPRequestHandler
httpd = socketserver.TCPServer((ip, int(port)), Handler)
```



```
httpd.serve_forever()
```

環境のルートで、`index.html` という名前のファイルに次の HTML コードを追加します。

```
<html>
  <head>
    <title>Hello Home Page</title>
  </head>
  <body>
    <p style="font-family:Arial;color:blue">Hello, World!</p>
  </body>
</html>
```

このファイルの HTML 出力をアプリケーションプレビュータブで確認するには、Node.js で `server.js` を実行するか、Python で `server.py` ファイルを実行します。次に、次のセクションのステップに従ってプレビューします。アプリケーションのプレビュータブで、URL の末尾に `/index.html` を追加して、Enter を押します。

実行中のアプリケーションのプレビュー

アプリケーションをプレビューする前に、以下の点を確認してください。

- アプリケーションはポート 8080、8081、または 8082 を経由して HTTP プロトコルを使用して実行している。
- 環境内のアプリケーションの IP アドレスは 127.0.0.1、localhost、または 0.0.0.0 である。
- アプリケーションコードファイルは IDE AWS Cloud9 で開いていてアクティブです。

これらの詳細をすべて確認したら、メニューバーで以下のいずれかのオプションを選択します。

- [Preview (プレビュー)]、Preview Running Application (実行中のアプリケーションのプレビュー)]
- [Tools (ツール)], [Preview (プレビュー)], [Preview Running Application (実行中のアプリケーションのプレビュー)]

これらのオプションのいずれかにより、環境内でアプリケーションプレビュータブが開き、アプリケーションの出力がタブに表示されます。

Note

アプリケーションプレビュータブにエラーが表示されるか、何も表示されない場合は、「[アプリケーションプレビュータブにエラーが表示される、または空白になる](#)」のトラブルシューティングステップに従ってください。アプリケーションまたはファイルをプレビューしようとしたときに、「ブラウザでサードパーティの Cookie が無効になっているため、プレビュー機能が無効になっています」という通知が表示される場合は、「[アプリケーションプレビューまたはファイルプレビュー通知:「サードパーティの Cookie が無効になっています」](#)」に記載されているトラブルシューティング手順に従ってください。

Note

アプリケーションがまだ実行していない場合は、アプリケーションプレビュータブにエラーが表示されます。この問題を解決するには、アプリケーションを実行または再起動し、メニューバーのコマンドをもう一度選択します。

例えば、アプリケーションが前述のポートや IP のいずれでも実行できないとします。または、アプリケーションをこれらのポートの複数で同時に実行する必要があるとします。例えば、アプリケーションをポート 8080 と 3000 で同時に実行する必要がある場合があります。このような場合、アプリケーションプレビュータブはエラーを表示するか、空白になる可能性があります。これは、環境内のアプリケーションプレビュータブは前述のポートと IP でのみ有効であり、アプリケーションは一度に 1 つのポートでのみ動作するためです。

アプリケーションプレビュータブで URL を他のユーザーと共有することはお勧めしません (URL の形式は

す `https://12a34567b8cd9012345ef67abcd890e1.vfs.cloud9.us-east-2.amazonaws.com/`。この形式では、12a34567b8cd9012345ef67abcd890e1 が環境に割り当てる ID です AWS Cloud9。us-east-2 は環境の AWS リージョンの ID です。) この URL を使用するには、環境の IDE が開いており、アプリケーションが同じウェブブラウザで実行されている必要があります。

IDE のアプリケーションプレビュータブまたは IDE 外の別のウェブブラウザタブ 0.0.0.0 を使用して 127.0.0.1、localhost、またはの IP にアクセスしようとすると、IDE AWS Cloud9 はデフォルトで、環境に接続されているインスタンスまたは独自のサーバーではなく、ローカルコンピュータに移動しようとします。

IDE の外部で実行中のアプリケーションのプレビューを他のユーザーに提供する方法については、「[実行中のアプリケーションをインターネット経由で共有する](#)」を参照してください。

アプリケーションのプレビューの再ロード

アプリケーションプレビューを再ロードするには、次のオプションを選択します。

アプリケーションプレビュータブで、[Refresh (最新情報に更新)] ボタン (円状の矢印) を選択します。

Note

このコマンドはサーバーを再起動しません。アプリケーションプレビュータブの内容を更新するだけです。

アプリケーションプレビュータイプの変更

アプリケーションのプレビュータイプを変更するには、次のオプションを選択します。

アプリケーションプレビュータブで、プレビュータイプリストから、以下のいずれかを選択します。

- [Browser (ブラウザ)]: ウェブブラウザ形式で出力をプレビューします。
- [Raw Content (UTF-8) (未加工コンテンツ (UTF-8))]: Unicode Transformation Format 8-bit (UTF-8) 形式で出力をプレビューします (該当する場合)。
- [Markdown]: Markdown 形式で出力をプレビューします (該当する場合)。

別のウェブブラウザのタブでアプリケーションプレビューを開く

別のウェブブラウザタブでアプリケーションプレビューを開くには、次のオプションを選択します。

アプリケーションプレビュータブで、[Pop Out Into New Window (新しいウィンドウで開く)] をクリックします。

Note

IDE は、同じウェブブラウザの少なくとも 1 AWS Cloud9 つの他のタブでも実行されている必要があります。そうでない場合、アプリケーションのプレビューは別のウェブブラウザタブに表示されません。

IDE は、同じウェブブラウザの少なくとも 1 AWS Cloud9 つの他のタブでも実行されている必要があります。そうでない場合、アプリケーションのプレビューは別のウェブブラウザタブに表示されません。アプリケーションプレビュータブにエラーが表示されるか、何も表示

されない場合は、「[アプリケーションプレビューまたはファイルプレビュー通知: 「サードパーティーの Cookie が無効になっています」](#)」のトラブルシューティングステップに従ってください。

別のプレビュー URL に切り替える

別のプレビュー URL に切り替えるには、次のオプションを選択します。

アプリケーションプレビュータブで、アドレスバーに別の URL へのパスを入力します。アドレスバーは [Refresh (最新の情報に更新)] ボタンとプレビュータイプリストの間にあります。

実行中のアプリケーションをインターネット経由で共有する

実行中のアプリケーションをプレビューした後、インターネット経由で他の人が利用できるようにすることができます。

Amazon EC2 インスタンスが環境に接続されている場合は、次の手順に従います。それ以外の場合は、サーバーのドキュメントを参照してください。

トピック

- [ステップ 1: インスタンスの ID と IP アドレスを取得する](#)
- [ステップ 2: インスタンスのセキュリティグループを設定する](#)
- [ステップ 3: インスタンスのサブネットをセットアップする](#)
- [ステップ 4: 実行中のアプリケーションの URL を共有する](#)

ステップ 1: インスタンスの ID と IP アドレスを取得する

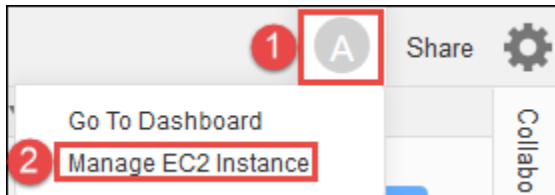
このステップでは、環境に接続されている Amazon EC2 インスタンスのインスタンス ID とパブリック IP アドレスを書き留めます。着信アプリケーションリクエストを許可するには、後のステップでインスタンス ID が必要になります。次に、パブリック IP アドレスを他のユーザーと共有して、実行中のアプリケーションにアクセスできるようにします。

1. Amazon EC2 インスタンスの ID を取得します。これを取得するには、次のいずれかを実行します。
 - 環境の IDE AWS Cloud9 のターミナルセッションで、次のコマンドを実行して Amazon EC2 インスタンスの ID を取得します。

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

インスタンス ID の形式は `i-12a3b456c789d0123` です。このインスタンス ID を書き留めます。

- 環境の IDE でメニューバーのユーザーアイコンを選択し、[Manage EC2 Instance (EC2 インスタンスの管理)] を選択します。



表示される Amazon EC2 コンソールで、[Instance ID (インスタンス ID)] 列に表示されるインスタンス ID を書き留めます。インスタンス ID の形式は `i-12a3b456c789d0123` です。

2. Amazon EC2 インスタンスのパブリック IP アドレスを取得します。これを取得するには、次のいずれかを実行します。

- 環境の IDE で、メニューバーの [共有] を選択します。[Share this environment (この環境を共有)] ダイアログボックスで、[Application (アプリケーション)] ボックスのパブリック IP アドレスを書き留めます。パブリック IP アドレスの形式は `192.0.2.0` です。
- 環境の IDE のターミナルセッションで、次のコマンドを実行して Amazon EC2 インスタンスのパブリック IP アドレスを取得します。

```
curl http://169.254.169.254/latest/meta-data/public-ipv4
```

パブリック IP アドレスの形式は `192.0.2.0` です。このパブリック IP アドレスをメモします。

- 環境の IDE でメニューバーのユーザーアイコンを選択し、[Manage EC2 Instance (EC2 インスタンスの管理)] を選択します。表示される Amazon EC2 コンソールの [説明] タブで、[IPv4 パブリック IP] フィールドのパブリック IP アドレスを書き留めます。パブリック IP アドレスの形式は `192.0.2.0` です。

Note

アプリケーションのパブリック IP アドレスは、アプリケーションのインスタンスが再起動するたびに変わる可能性があります。IP アドレスの変更を防ぐには、Elastic IP アド

レスを割り当てます。次に、そのアドレスを実行中のインスタンスに割り当てます。手順については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレスの割り当て](#)」および「[Elastic IP アドレスを実行中のインスタンスに関連付ける](#)」を参照してください。Elastic IP アドレスを割り当てると、AWS アカウント に料金が発生する可能性があります。詳細については、「[Amazon EC2 の料金表](#)」を参照してください。

ステップ 2: インスタンスのセキュリティグループを設定する

このステップでは、Amazon EC2 コンソールで、環境に接続されているインスタンスの Amazon EC2 セキュリティグループを設定します。ポート 8080、8081、または 8082 経由での着信 HTTP リクエストを許可するように設定します。

Note

ポート 8080、8081、または 8082 経由で HTTP を使用して実行することは必須ではありません。ただし、そうしないと、IDE 内から実行中のアプリケーションをプレビューすることはできません。詳細については、「[実行中のアプリケーションのプレビュー](#)」を参照してください。別のプロトコルやポートで実行している場合は、このステップで置き換えてください。

追加のセキュリティレイヤーについては、インスタンスが使用できる VPC 内のサブネットにネットワークアクセスコントロールリスト (ACL) を設定します。セキュリティグループとネットワーク ACL の詳細については、以下を参照してください。

- [ステップ 3: インスタンスのサブネットをセットアップする](#)
- Amazon VPC ユーザーガイドの[セキュリティ](#)
- Amazon VPC ユーザーガイドの[VPC のセキュリティグループ](#)
- Amazon VPC ユーザーガイドの[ネットワーク ACL](#)

1. 環境の IDE でメニューバーのユーザーアイコンを選択し、[Manage EC2 Instance (EC2 インスタンスの管理)] を選択します。次に、この手順のステップ 3 にスキップします。
2. この手順で「EC2 インスタンスを管理する」ステップや他のステップを選択したときにエラーが表示された場合は、AWS アカウントの管理者の認証情報を使用して Amazon EC2 コンソールにサインインします。次に、以下の手順を実行します。これを実行できない場合は、AWS アカウント 管理者にお問い合わせください。

- a. まだサインインしていない場合は、<https://console.aws.amazon.com/> AWS マネジメントコンソール でサインインします。
 - b. Amazon EC2 コンソールを開きます。これを行うには、ナビゲーションバーで、[Services] (サービス) を選択します。次に、[EC2] を選択します。
 - c. ナビゲーションバーで、環境 AWS リージョン がある を選択します。
 - d. [EC2 ダッシュボード] が表示された場合は、[実行中のインスタンス] を選択します。表示されない場合は、サービスナビゲーションペインで [Instances] (インスタンス) を展開し (まだ展開していない場合)、[Instances] (インスタンス) を選択します。
 - e. インスタンスのリストで、前にメモしたインスタンス ID と一致するインスタンス ID のインスタンスを選択します。
3. インスタンスの [Description] (説明) タブで、[Security groups] (セキュリティグループ) の横のセキュリティグループリンクを選択します。
 4. セキュリティグループを表示した状態で、[インバウンド] タブを確認します。[タイプ] が [カスタム TCP ルール]、[ポート範囲] が [8080]、[8081]、または [8082] に設定されているルールが既に存在する場合は、[キャンセル] を選択して、「[ステップ 3: インスタンスのサブネットをセットアップする](#)」にスキップします。それ以外の場合は、[編集] を選択します。
 5. [インバウンドのルールの編集] ダイアログボックスで、[ルールの追加] をクリックします。
 6. [タイプ] で [カスタム TCP ルール] を選択します。
 7. [Port Range] (ポート範囲) に、8080、8081、または 8082 と入力します。
 8. [ソース] で、[任意の場所] を選択します。

 Note

[ソース] で [任意の場所] を選択すると、任意の IP アドレスからの着信リクエストを許可することになります。これを特定の IP アドレスに制限するには、[Custom] (カスタム) を選択して IP アドレス範囲を入力します。または、[My IP] (マイ IP) を選択して、自分の IP アドレスからのリクエストのみに制限します。

9. [保存] を選択します。

ステップ 3: インスタンスのサブネットをセットアップする

Amazon EC2 コンソールおよび Amazon VPC コンソールを使用して、環境に接続されている Amazon EC2 インスタンスのサブネットを設定します。次に、ポート 8080、8081、または 8082 経由での着信 HTTP リクエストを許可します。

Note

ポート 8080、8081、または 8082 経由で HTTP を使用して実行することは必須ではありません。ただし、そうしないと、IDE 内から実行中のアプリケーションをプレビューすることはできません。詳細については、「[実行中のアプリケーションのプレビュー](#)」を参照してください。別のプロトコルまたはポートで実行している場合は、このステップで置き換えてください。

このステップでは、インスタンスが使用できる Amazon VPC 内のサブネットに対してネットワーク ACL を設定する方法について説明します。これは必須ではありませんが、お勧めします。ネットワーク ACL を設定すると、別のセキュリティレイヤーが追加されます。ネットワーク ACL の詳細については、以下を参照してください。

- Amazon VPC ユーザーガイドの[セキュリティ](#)
- Amazon VPC ユーザーガイドの[ネットワーク ACL](#)

1. Amazon EC2 コンソールのサービスナビゲーションペインで、[Instances] (インスタンス) を展開し (まだ展開していない場合)、[Instances] (インスタンス) を選択します。
2. インスタンスのリストで、前にメモしたインスタンス ID と一致するインスタンス ID のインスタンスを選択します。
3. インスタンスの [説明] タブで [サブネット ID] の値を書き留めます。サブネット ID の形式は subnet-1fab8aEX です。
4. Amazon VPC コンソールを開きます。これを行うには、AWS ナビゲーションバーでサービスを選択し、VPC を選択します。

このステップでは、AWS アカウントで管理者の認証情報を使用して Amazon VPC コンソールにサインインすることをお勧めします。これができない場合は、AWS アカウント 管理者に確認してください。

5. [VPC ダッシュボード] が表示されている場合は、[サブネット] を選択します。それ以外の場合は、サービスナビゲーションペインで [サブネット] を選択します。

6. サブネットのリストで、前にメモしたサブネット ID と一致するサブネット ID のサブネットを選択します。
7. [Summary] (概要) タブで、[Network ACL] (ネットワーク ACL) の横にあるネットワーク ACL のリンクを選択します。
8. ネットワーク ACL の一覧で、ネットワーク ACL を選択します (ネットワーク ACL は 1 つしかありません)。
9. [インバウンドのルール] タブで、ネットワーク ACL を確認します。[タイプ] が HTTP* (8080)、HTTP* (8081)、または HTTP* (8082) に設定されているルールがすでに存在する場合は、[ステップ 4: 実行中のアプリケーションの URL を共有する](#) に進みます。それ以外の場合は、[編集] を選択します。
10. [別のルールの追加] を選択します。
11. [Rule #] (ルール番号) にルール番号 (200 など) を入力します。
12. [タイプ] で [カスタム TCP ルール] を選択します。
13. [ポート範囲] に「8080」、「8081」、または「8082」と入力します。
14. [ソース] に、着信リクエストを許可する IP アドレスの範囲を入力します。例えば、任意の IP アドレスからの着信リクエストを許可するには、「0.0.0.0/0」と入力します。
15. [許可/拒否] を [許可] に設定して、[保存] を選択します。

ステップ 4: 実行中のアプリケーションの URL を共有する

アプリケーションを実行したら、アプリケーションの URL を与えることで、アプリケーションを他のユーザーと共有できます。そのためには、先ほどメモしておいたパブリック IP アドレスが必要になります。アプリケーションの完全な URL を記述するには、アプリケーションのパブリック IP アドレスを正しいプロトコルで開始してください。次に、アプリケーションポートがアプリケーションで使用するプロトコルのデフォルトポートでない場合は、ポート番号情報を追加します。アプリケーション URL の一例は、ポート 8080 経由で HTTP を使用する `http://192.0.2.0:8080/index.html` です。

結果のウェブブラウザタブにエラーが表示されるか、何も表示されない場合は、「[IDE の外部で実行中のアプリケーションを表示できない](#)」のトラブルシューティングステップに従ってください。

Note

アプリケーションのパブリック IP アドレスは、アプリケーションのインスタンスが再起動するたびに変わる可能性があります。この IP アドレスが変更されないようにするには、Elastic

IP アドレスを割り当て、そのアドレスを実行中のインスタンスに割り当てます。手順については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレスの割り当て](#)」および「[Elastic IP アドレスを実行中のインスタンスに関連付ける](#)」を参照してください。Elastic IP アドレスを割り当てると、AWS アカウント に料金が発生する可能性があります。詳細については、[Amazon EC2 の料金表](#)を参照してください。

ポート 8080、8081、または 8082 経由で HTTP を使用してアプリケーションを実行することは必須ではありません。ただし、そうしないと、IDE 内から実行中のアプリケーションをプレビューすることはできません。詳細については、「[実行中のアプリケーションのプレビュー](#)」を参照してください。

例えば、要求されたプロトコルまたはポート経由のトラフィックをブロックする VPN から送信されたリクエストがあるとします。この場合、これらのリクエストはアプリケーションの URL へのアクセスに失敗する可能性があります。リクエストは、要求されたプロトコルやポート経由のトラフィックを許可する別のネットワークから行う必要があります。詳細については、ネットワーク管理者にお問い合わせください。

IDE のアプリケーションプレビュータブの URL を他のユー

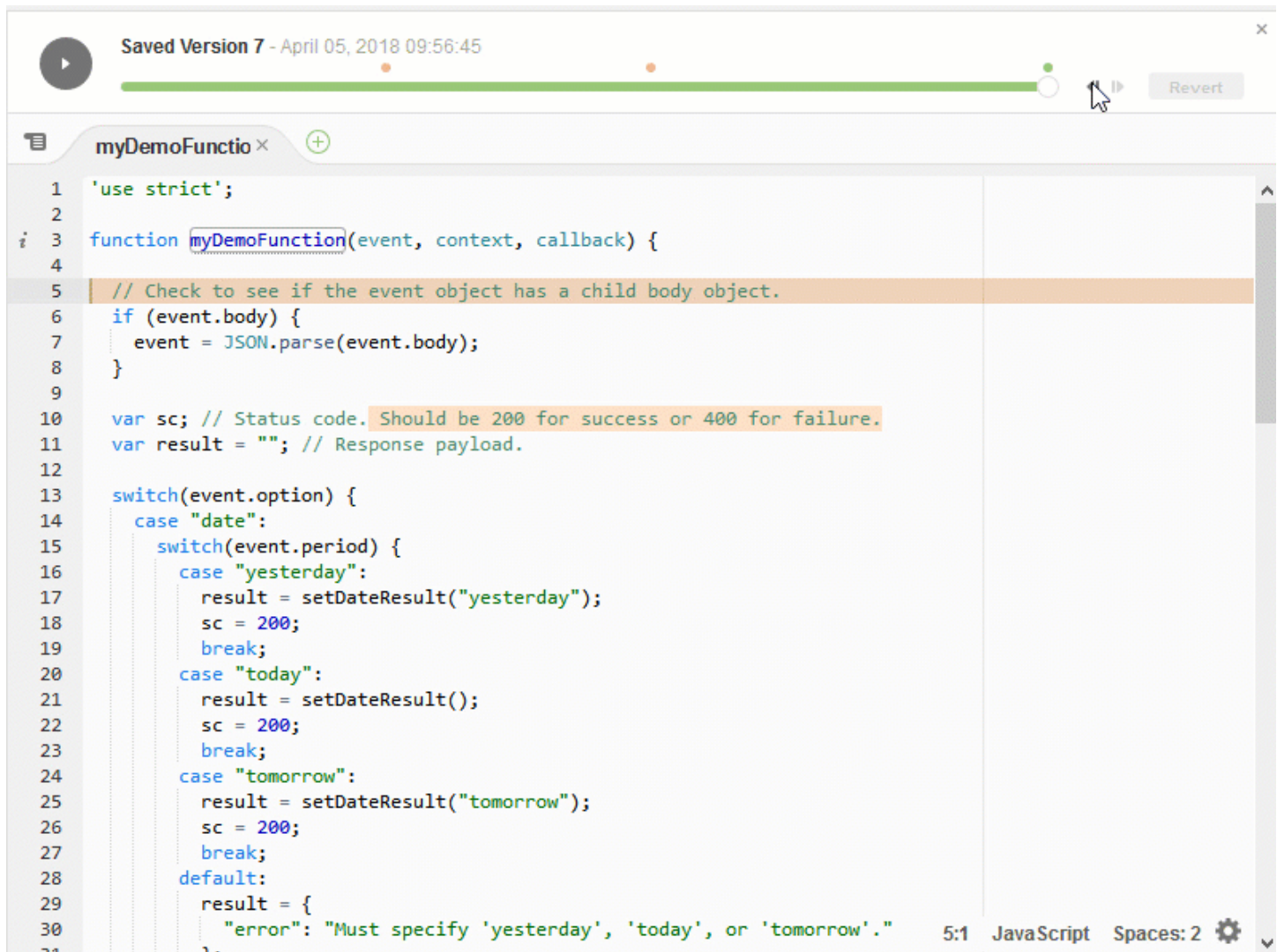
ザーと共有することはお勧めしません。(この URL の形式は で

す `https://12a34567b8cd9012345ef67abcd890e1.vfs.cloud9.us-`

`east-2.amazonaws.com/`。この形式では、12a34567b8cd9012345ef67abcd890e1 が環境に割り当てる ID です AWS Cloud9。us-east-2 は環境の ID AWS リージョン です。) この URL を使用するには、環境の IDE が開いており、アプリケーションが同じウェブブラウザで実行されている必要があります。

IDE AWS Cloud9 でのファイルリビジョンの操作

AWS Cloud9 統合開発環境 (IDE) のファイルリビジョン履歴ペインを使用して、AWS Cloud9 EC2 開発環境のファイルに対する変更を表示および管理できます。ファイルリビジョン履歴ペインは、SSH AWS Cloud9 開発環境のファイルでは使用できません。

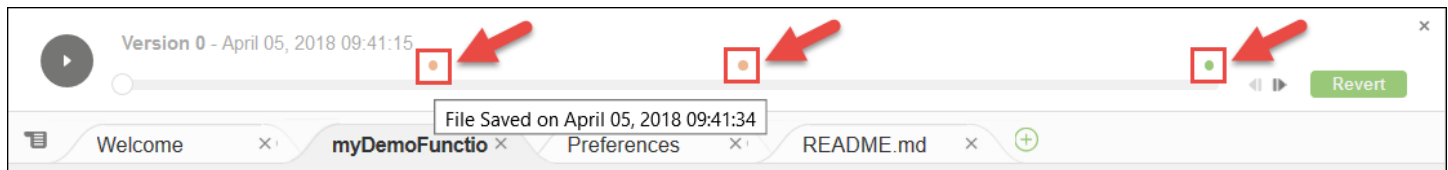


ファイルの [File Revision History (ファイルリビジョン履歴)] ペインを表示するには、エディタでファイルを開きます。次に、メニューバーで、[File (ファイル)]、[Show File Revision History (ファイルリビジョン履歴を表示する)] を選択します。

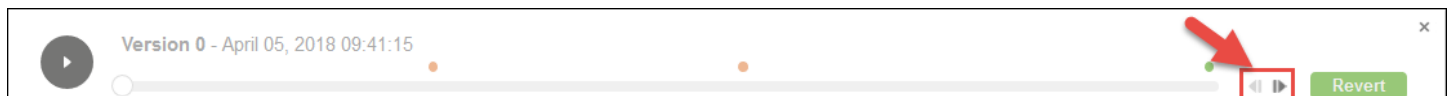
[File Revision History (ファイルリビジョン履歴)] ペインは、環境で最初にエディタでファイルを開いた後、その環境でのみ、IDE でファイルのリビジョン履歴の追跡を開始します。[File Revision History (ファイルリビジョン履歴)] ペインは、エディタ自体から行われたファイルのリビジョンのみを追跡します。他の方法 (ターミナル、Git、他のファイルリビジョンツールなど) で行われたファイルのリビジョンは追跡されません。

[File Revision History (ファイルリビジョン履歴)] ペインが表示されている間は、ファイルを編集できません。ペインを非表示にするには、[File (ファイル)]、[Show Revision History (リビジョン履歴を表示する)] を再度選択するか、ペインの隅にある [X] ([Close timeslider (タイムスライダを閉じる)]) を選択します。

ファイル保存アクションに関連付けられているファイルのバージョンにジャンプするには、リビジョンスライダの上にある [File Saved on (ここでファイルを保存)] ドットを選択します。



ファイルスライダでファイルの現在選択されているバージョンから 1 つ前または後のバージョンに移動するには、ステップ矢印 ([Step revision forward (前のリビジョンにステップ)] または [Step revision backward (後のリビジョンにステップ)]) のいずれかを選択します。



ファイルのバージョンを一度に 1 つずつ、リビジョン履歴の最初から最後まで自動的に進むには、再生ボタン ([Playback file history (ファイル履歴の再生)]) を選択します。

現在選択されているバージョンのファイルをリビジョン履歴の最新バージョンにするには、[Revert (元に戻す)] を選択します。

IDE AWS Cloud9 でのイメージファイルの使用

AWS Cloud9 統合開発環境 (IDE) を使用して、イメージファイルを表示および編集できます。

- [イメージの表示または編集](#)
- [イメージのサイズ変更](#)
- [イメージのクロップ](#)
- [イメージの回転](#)
- [イメージの反転](#)
- [イメージのズーム](#)
- [イメージのスムーズ化](#)

イメージの表示または編集

AWS Cloud9 IDE で、表示または編集するイメージの ファイルを開きます。サポートされているイメージファイルの種類には次のものがあります。

- .bmp

- .gif (表示のみ)
- .ico (表示のみ)
- .jpeg
- .jpg
- .png
- .tiff

イメージのサイズ変更

1. IDE でイメージファイルを開きます。
2. イメージ編集バーで、[Resize (サイズ変更)] を選択します。
3. イメージの幅を変更するには、新しい [Width (幅)] をピクセルで入力します。または、[Width (幅)] の横にある [-] または [+] を選択して、幅を一度に 1 ピクセルずつ変更します。
4. イメージの高さを変更するには、新しい [Height (高さ)] をピクセルで入力します。または、[Height (高さ)] の横にある [-] または [+] を選択して、高さを一度に 1 ピクセルずつ変更します。
5. イメージの幅と高さの比率を維持するには、[Maintain Aspect Ratio (アスペクト比を保つ)] のチェックをそのままにします。
6. イメージの新しいサイズを確定するには、イメージ編集バーで、幅 ([W]) と高さ ([H]) の測定値 (ピクセル) を確認します。
7. [サイズ変更] を選択します。
8. サイズ変更を破棄するには、メニューバーで [Edit (編集)]、[Undo (元に戻す)] の順に選択します。新しいサイズを維持するには、[File (ファイル)]、[Save (保存)] の順に選択します。

イメージのクロップ

1. IDE でイメージファイルを開きます。
2. 保持したいイメージの部分の上にポインタをドラッグします。
3. 選択の寸法を確認するには、次のように、イメージ編集バーで [Selection (選択)] の寸法を確認します。
 - 元のイメージの左端から選択範囲の左端までの距離 (ピクセル) ([L])
 - 元のイメージの上端から選択範囲の上端までの距離 (ピクセル) ([T])
 - 選択の幅 (ピクセル) ([W])

- 選択の高さ (ピクセル) ([H])
4. イメージ編集バーで、[Crop (クロップ)] を選択します。
 5. クロップを破棄するには、メニューバーで [Edit (編集)]、[Undo (元に戻す)] の順に選択します。クロップされた新しいイメージを維持するには、[File (ファイル)]、[Save (保存)] の順に選択します。

イメージの回転

1. IDE でイメージファイルを開きます。
2. イメージを左回りに回転させるには、イメージ編集バーで [Rotate 90 Degrees Left (左に 90 度回転)] を選択します。
3. イメージを右回りに回転させるには、イメージ編集バーで [Rotate 90 Degrees Right (右に 90 度回転)] を選択します。
4. 回転を破棄するには、メニューバーで [Edit (編集)]、[Undo (元に戻す)] の順に選択します。回転させた新しいイメージを維持するには、[File (ファイル)]、[Save (保存)] の順に選択します。

イメージの反転

1. IDE でイメージファイルを開きます。
2. イメージを水平に反転させるには、イメージ編集バーで [FlipH] を選択します。
3. イメージを垂直に反転させるには、イメージ編集バーで [FlipV] を選択します。
4. 反転を破棄するには、メニューバーで [Edit (編集)]、[Undo (元に戻す)] の順に選択します。反転させた新しいイメージを維持するには、[File (ファイル)]、[Save (保存)] の順に選択します。

イメージのズーム

1. IDE でイメージファイルを開きます。
2. イメージ編集バーで、使用可能なズーム率のいずれかを選択します (例: [75%]、[100%]、[200%])。

イメージのスムーズ化

1. IDE でイメージファイルを開きます。

2. イメージ編集バーで、[Smooth (スムーズ化)] を選択して、画素数を減らします。スムージングを破棄するには、[Smooth (スムーズ化)] の選択を解除します。
3. メニューバーで [File (ファイル)]、[Save (保存)] の順に選択します。

AWS Cloud9 IDE でのビルダー、ランナー、デバッガーの操作

ビルダーは、AWS Cloud9 統合開発環境 (IDE) にプロジェクトのファイルを構築する方法を指示します。ランナーは、特定のタイプのファイルを実行する方法を AWS Cloud9 IDE に指示します。ランナーは、デバッガーを使用してファイルのソースコードの問題を見つけることができます。

AWS Cloud9 IDE を使用して、次の方法でコードを構築、実行、デバッグできます。

- ビルダーを使用してプロジェクトのファイルを構築します。詳細については、「[プロジェクトのファイルの構築](#)」を参照してください。
- ランナーを使用してコードを実行 (および任意でデバッグ) します。詳細については、「[組み込みの構築、実行、およびデバッグのサポート](#)」および「[コードを実行する](#)」を参照してください。
- 組み込みランナーを変更して、元の定義とは異なる方法でコードを実行 (および任意でデバッグ) します。詳細については、「[組み込みのランナーの変更](#)」を参照してください。
- ランナーを使用して、ファイル名、コマンドラインオプション、デバッグモード、現在の作業ディレクトリ、および環境変数をカスタムに組み合わせて、コードを実行 (および任意でデバッグ) します。詳細については、「[実行設定の作成](#)」を参照してください。
- 独自のビルダーまたはランナーを作成します。詳細については、「[ビルダーまたはランナーの作成](#)」を参照してください。

組み込みの構築、実行、およびデバッグのサポート

AWS Cloud9 IDE には、複数の言語のコードの構築、実行、デバッグのサポートが組み込まれています。完全なリストについては、「[言語のサポート](#)」を参照してください。

ビルドの組み込みサポートは、メニューバーで [Run (実行)]、[Build System (ビルドシステム)] を選択するか、[Run (実行)]、[Build (構築)] メニューコマンドで使用できます。プログラミング言語またはツールのサポートを追加するには、「[ビルダーまたはランナーを作成する](#)」を参照してください。

実行の組み込みサポートは、[Run (実行)] ボタン、およびメニューバーの [Run (実行)]、[Run With (次で実行)]、および [Run (実行)]、[Run Configurations (実行設定)] メニューコマンドで

使用できます。プログラミング言語またはツールのサポートを追加するには、「[ビルダーまたはランナーを作成する](#)」または「[実行設定を作成する](#)」を参照してください。

デバッグの組み込みサポートは、[Debugger (デバッガー)] ウィンドウから使用できます。

[Debugger (デバッガー)] ウィンドウを表示するには、[Debugger (デバッガー)] ボタンを選択します。[Debugger (デバッガー)] ボタンが非表示になっている場合は、メニューバーで [Window (ウィンドウ)]、[Debugger (デバッガー)] の順に選択します。

プロジェクトのファイルを構築する

1. 構築したいコードに対応するファイルを開きます。
2. メニューバーで、[Run (実行)]、[Build System (ビルドシステム)] を選択し、使用するビルダーの名前がまだ選択されていない場合は選択します。使用したいビルダーがリストに表示されない場合は、この手順を停止して、「[ビルダーまたはランナーを作成する](#)」の手順を完了してから、この手順に戻ります。
3. [Run (実行)]、[Build (構築)] を選択します。

コードを実行する

1. 実行したいコードに対応するファイルをまだ開いて選択していない場合は、ファイルを選択して開きます。
2. メニューバーで、以下のいずれかを選択します。
 - 最も一致する組み込みランナーを使用してコードを実行するには、[Run (実行)]、[Run (実行)] の順に選択します。AWS Cloud9 が見つからない場合、このコマンドは無効になります。
 - AWS Cloud9 最後に使用した実行設定でコードを実行するには、「実行」、「最後に実行」を選択します。
 - 特定のランナーを使用してコードを実行するには、[Run (実行)]、[Run With (次で実行)] を選択し、ランナーの名前を選択します。使用したいランナーがリストに表示されない場合は、この手順を停止して、「[ビルダーまたはランナーを作成する](#)」の手順を完了してから、この手順に戻ります。
 - ファイル名、コマンドラインオプション、デバッグモード、現在の作業ディレクトリ、および環境変数のカスタムの組み合わせを使用して特定のランナーでコードを実行するには、[Run (実行)]、[Run Configurations (実行設定)] の順に選択します。表示された実行設定タブで、[Runner: Auto (ランナー: 自動)] を選択し、使用したいランナーを選択してから [Run (実行)] を選択します。使用したいランナーがリストに表示されない場合は、この手順を停止して、「[ビルダーまたはランナーを作成する](#)」の手順を完了してから、この手順に戻ります。

コードをデバッグする

1. コードの実行設定タブで、[Run in Debug Mode (デバッグモードで実行)] を選択します。バグアイコンが、白背景に緑に変わります。詳細については、「[コードを実行する](#)」および「[実行設定を作成する](#)」を参照してください。
2. 次のように、実行中に一時停止したいブレークポイントをコード内に設定します。
 - a. ブレークポイントを設定したい各ファイルを開きます。
 - b. ファイル内のブレークポイントを設定したい各ポイントで、行番号の左にあるガーターの空白部分を選択します。赤い円が表示されます。

ブレークポイントを削除するには、ガーターの既存のブレークポイントを選択します。

ブレークポイントを削除ではなく無効にするには、[Debugger (デバッガー)] ウィンドウの [Breakpoints (ブレークポイント)] で、無効にしたいブレークポイントに該当するボックスのチェックを外します。ブレークポイントを再度有効にするには、クリアしたボックスを選択します。

一度にすべてのブレークポイントを無効にするには、[Debugger (デバッガー)] ウィンドウで、[Deactivate All Breakpoints (すべてのブレークポイントを無効にする)] を選択します。すべてのブレークポイントを再度有効にするには、[Activate All Breakpoints (すべてのブレークポイントを有効にする)] を選択します。

[Debugger (デバッガー)] ウィンドウが非表示になっている場合は、[Debugger (デバッガー)] ボタンを選択します。[Debugger (デバッガー)] ボタンが非表示になっている場合は、メニューバーで [Window (ウィンドウ)]、[Debugger (デバッガー)] の順に選択します。

3. 次のように、実行が一時停止するポイントで取得したい値のウォッチ式を設定します。
 - a. [Debugger (デバッガー)] ウィンドウの [Watch Expressions (ウォッチ式)] で、[Type an expression here (ここに式を入力)] を選択します。
 - b. ウォッチしたい式を入力し、Enter を押します。

既存のウォッチ式を変更するには、ウォッチ式を右クリックして、[Edit Watch Expression (ウォッチ式の編集)] を選択します。変更を入力し、Enter を押します。

既存のウォッチ式を削除するには、ウォッチ式を右クリックして、[Remove Watch Expression (ウォッチ式の削除)] を選択します。

4. 「[コードを実行する](#)」で説明されているようにコードを実行します。

実行が一時停止するたびに、表示されたコードの任意の部分 (変数など) でポインタを一時停止することもできます。関連情報があればそれがツールチップで表示されます。

組み込みランナーを変更する

このステップでは、組み込みランナーを変更して、元の定義とは異なる方法でコードを実行 (および任意でデバッグ) する方法を示します。

1. メニューバーで、[Run (実行)]、[Run With (次で実行)] を選択し、変更したい組み込みランナーを選択します。
2. ランナーでコードの実行を停止するには、表示された実行設定タブで [Stop (停止)] を選択します。
3. [Runner: My Runner (ランナー: マイランナー)] を選択します。ここでは、[My Runner] は変更したいランナーの名前です。次に [Edit Runner (ランナーの編集)] を選択します。
4. 表示される [My Runner.run] タブで、ランナーの現在の定義を変更します。「[ビルダーまたはランナーを定義する](#)」を参照してください。
5. [File (ファイル)]、[Save As (名前を付けて保存)] の順に選択します。ファイルを同じ名前 (My Runner.run) で my-environment/.c9/runners ディレクトリに保存します。ここで、my-environment は AWS Cloud9 開発環境の名前です。

Note

組み込みランナーを変更した場合、それらの変更はその変更を行った環境にのみ適用されます。変更を別の環境に適用するには、他の他の環境を開いて、前述のステップに従って組み込みランナーを開き、編集し、同じ変更を保存します。

実行設定を作成する

このステップでは、ランナーを使用して、ファイル名、コマンドラインオプション、デバッグモード、現在の作業ディレクトリ、および環境変数をカスタムに組み合わせて、コードを実行 (および任意でデバッグ) する方法を示します。

メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。表示された実行設定タブで、次の操作を実行します。

1. [Run (実行)] および [Restart (再起動)] の横にあるボックスに、この実行設定の [Run (実行)]、[Run Configurations (実行設定)] メニューに表示する名前を入力します。
2. [Command (コマンド)] ボックスに、使用したいカスタムコマンドラインオプションを入力します。
3. この実行設定でランナーの事前定義されたデバッグ設定を使用するには、[Run in Debug Mode (デバッグモードで実行)] を選択します。バグアイコンが、白背景で緑に変わります。
4. この実行設定で特定の作業ディレクトリを使用するには、[CWD] を選択し、ディレクトリを選択して [Select (選択)] を選択します。
5. この実行設定で特定の環境変数を使用するには、[ENV] を選択し、各環境変数の名前と値を入力します。

この実行設定を使用するには、実行したいコードに対応するファイルを開きます。メニューバーで、[Run (実行)]、[Run Configurations (実行設定)] の順に選択してから、実行設定の名前を選択します。表示された実行設定タブで、[Runner: Auto (ランナー: 自動)] を選択し、使用したいランナーを選択してから [Run (実行)] を選択します。

Note

作成された実行設定は、その実行設定がある環境のみに適用されます。実行設定を別の環境に追加するには、他の環境を開いて、その環境で前述のステップに従って同じ実行設定を作成します。

ビルダーまたはランナーを作成する

このステップでは、独自のビルダーまたはランナーを作成する方法を示します。

1. ビルダーを作成するには、メニューバーで、[Run (実行)]、[Build System (ビルドシステム)]、[New Build System (新しいビルドシステム)] の順に選択します。ランナーを作成するには、メニューバーで、[Run (実行)]、[Run With (次で実行)]、[New Runner (新しいランナー)] の順に選択します。
2. 表示されたビルダータブ ([My Builder.build] というラベル) またはランナータブ ([My Runner.run] というラベル) で、ビルダーまたはランナーを定義します。「[ビルダーまたはランナーを定義する](#)」を参照してください。
3. ビルダーまたはランナーを定義した後、[File (ファイル)]、[Save As (名前を付けて保存)] の順に選択します。ビルダーの場合は、.build という拡張子でファイルを my-

environment/.c9/builders ディレクトリに保存します。my-environment は環境の名前です。ランナーの場合は、.run というファイル拡張子でファイルを my-environment/.c9/runners ディレクトリに保存します。my-environment は環境の名前です。指定されたファイル名が、[Run (実行)]、[Build System (ビルドシステム)] メニュー (ビルダーの場合) または [Run (実行)]、[Run With (次で実行)] メニュー (ランナーの場合) に表示される名前です。したがって、別のファイル名を指定しない限り、デフォルトでは表示名は [My Builder] (ビルダーの場合) または [My Runner] (ランナーの場合) になります。

このビルダーまたはランナーを使用するには、「[プロジェクトのファイルを構築する](#)」または「[コードを実行する](#)」を参照してください。

Note

作成したビルダーまたはランナーは、そのビルダーまたはランナーを作成した環境のみに適用されます。ビルダーまたはランナーを別の環境に追加するには、他の環境を開いて、その環境で前述のステップに従って同じビルダーまたはランナーを作成します。

ビルダーまたはランナーを定義する

このトピックでは、ビルダーまたはランナーを定義する方法について説明します。ビルダーまたはランナーを定義する前に、[必ずビルダーまたはランナーを作成](#)してください。

表示されたビルダーまたはランナータブで、JSON を使用してランナーまたはビルダーを定義します。次のコードをテンプレートとして使用して開始します。

ビルダーの場合は、このコードで開始します。

```
{
  "cmd": [],
  "info": "",
  "env": {},
  "selector": ""
}
```

ランナーの場合は、このコードで開始します。

```
{
```

```
"cmd": [],
"script": "",
"working_dir": "",
"info": "",
"env": {},
"selector": "",
"debugger": "",
"debugport": ""
}
```

上記のコードでは:

- **cmd:** が単一のコマンドとして AWS Cloud9 実行する文字列のカンマ区切りリストを表します。

がこのコマンド AWS Cloud9 を実行すると、リスト内の各文字列は 1 つのスペースで区切られます。たとえば、AWS Cloud9 は "cmd": ["ls", "\$file", "\$args"]として実行されls \$file \$args、AWS Cloud9 は現在のファイルへの\$fileフルパスと、ファイル名の後に入力された引数\$argsに置き換えられます。詳細については、このセクションで後述するサポートされている変数のリストを参照してください。

- **script:** ランナーがターミナルで実行する Bash スクリプトを表します (読みやすくするために行の配列として指定することもできます)。
- **working_dir:** ランナーが実行されるディレクトリを表します。
- **info:** 実行開始時にユーザーに表示したい任意の文字列を表します。この文字列には変数を含めることができます。たとえばRunning \$project_path\$file_name...、AWS Cloud9 は現在のファイルのディレクトリパス\$project_pathに、現在のファイルの名前部分\$file_nameに置き換えます。このセクションで後述するサポートされている変数のリストを参照してください。
- **env:** AWS Cloud9 が使用するコマンドライン引数の配列を表します。次に例を示します。

```
"env": {
  "LANG": "en_US.UTF-8",
  "SHLVL": "1"
}
```

- **selector:** このランナーに適用されるファイル名を識別 AWS Cloud9 するために使用する正規表現を表します。たとえば、source.py で Python ファイルを指定できます。
- **debugger:** このランナーと AWS Cloud9 互換性のある使用可能なデバッガーの名前を表します。たとえば、V8 デバッガーには v8 を指定できます。

- `debugport`: デバッグ中 AWS Cloud9 に使用するポート番号を表します。たとえば、使用するポート番号に 15454 を指定できます。

次の表に、使用できる変数を示します。

変数	説明
<code>\$file_path</code>	現在のファイルのディレクトリ。たとえば <code>/home/ec2-user/environment</code> や <code>/home/ubuntu/environment</code> です。
<code>\$file</code>	現在のファイルの完全パス。たとえば <code>/home/ec2-user/environment/hello.py</code> や <code>/home/ubuntu/environment/hello.py</code> です。
<code>\$args</code>	ファイル名の後に入力する任意の引数。たとえば <code>"5"</code> <code>"9"</code> です。
<code>\$file_name</code>	現在のファイルの名前の一部。たとえば <code>hello.py</code> です。
<code>\$file_extension</code>	現在のファイルの拡張子。たとえば <code>py</code> です。
<code>\$file_base_name</code>	ファイル拡張子を付けない現在のファイルの名前。たとえば <code>hello</code> です。
<code>\$packages</code>	パッケージフォルダの完全パス。
<code>\$project</code>	現在のプロジェクトフォルダの完全パス。
<code>\$project_path</code>	現在のプロジェクトのディレクトリ。たとえば <code>/home/ec2-user/environment/</code> や <code>/home/ubuntu/environment/</code> です。
<code>\$project_name</code>	ファイル拡張子を付けない現在のプロジェクトの名前。たとえば <code>my-demo-environment</code> です。

変数	説明
<code>\$project_extension</code>	現在のプロジェクトファイルの拡張子。
<code>\$project_base_name</code>	拡張子を付けない現在のプロジェクトの名前。
<code>\$hostname</code>	環境のホスト名。たとえば <code>192.0.2.0</code> です。
<code>\$hostname_path</code>	プロジェクトファイルへの相対パスを含む環境のホスト名。たとえば <code>https://192.0.2.0/hello.js</code> です。
<code>\$url</code>	環境にアクセスする完全な URL。たとえば <code>https://192.0.2.0.</code> です。
<code>\$port</code>	環境に割り当てられたポート。たとえば <code>8080</code> です。
<code>\$ip</code>	環境に対してプロセスを実行する IP アドレス。たとえば <code>0.0.0.0</code> です。

例として、次の `G++.build` という名前のビルダーファイルは、`-o` オプションを指定した `g++` コマンドを実行して現在のファイル (例: `hello.cpp`) をオブジェクトモジュールにコンパイルする GCC のビルダーを定義します。次に、オブジェクトモジュールを現在のファイルと同じ名前 (例: `hello`) でプログラムにリンクします。ここでは、`g++ -o hello hello.cpp` が同等の `bu` コマンドです。

```
{
  "cmd": [ "g++", "-o", "$file_base_name", "$file_name" ],
  "info": "Compiling $file_name and linking to $file_base_name...",
  "selector": "source.cpp"
}
```

別の例として、次の `Python.run` という名前のランナーファイルは、Python を使用し、任意の引数を指定して現在のファイルを実行するランナーを定義します。たとえば、現在のファイルが `hello.py` という名前であり、引数に 5 および 9 を指定する場合、同等のコマンドは `python hello.py 5 9` です。

```
{
  "cmd": [ "python", "$file_name", "$args" ],
  "info": "Running $file_name...",
  "selector": "source.py"
}
```

最後に、次の Print Run Variables.run という名前のランナーファイルは、使用できる変数ごとの値を単純に出力してから停止するランナーを定義します。

```
{
  "info": "file_path = $file_path, file = $file, args = $args, file_name = $file_name,
file_extension = $file_extension, file_base_name = $file_base_name, packages
= $packages, project = $project, project_path = $project_path, project_name
= $project_name, project_extension = $project_extension, project_base_name =
$project_base_name, hostname = $hostname, hostname_path = $hostname_path, url = $url,
port = $port, ip = $ip"
}
```

AWS Cloud9 IDE でのカスタム環境変数の使用

AWS Cloud9 統合開発環境 (IDE) は、カスタム環境変数の設定をサポートしています。IDE AWS Cloud9 でカスタム環境変数を設定するには、次の方法があります。

- [コマンドレベルのカスタム環境変数を設定する](#)
- [~/.bash_profile のカスタムユーザー環境変数を設定する](#)
- [ローカルのカスタム環境変数を設定する](#)
- [~/.bashrc のカスタムユーザー環境変数を設定する](#)
- [ENV リストのカスタム環境変数を設定する](#)

コマンドレベルのカスタム環境変数を設定する

AWS Cloud9 開発環境でコマンドを実行するときに、コマンドレベルのカスタム環境変数を設定できます。この動作をテストするには、次のコードを使用して、script.sh という名前のファイルを作成します。

```
#!/bin/bash
```



```
echo $MY_ENV_VAR
```

次のコマンドを実行すると、ターミナルに `Terminal session` が表示されます。

```
MY_ENV_VAR='Terminal session' sh ./script.sh
```

このトピックで説明されている複数の方法を使用してカスタム環境変数を設定すると、カスタム環境変数の値を取得する際に、この設定が他のすべてに優先されます。

~/.bash_profile のカスタムユーザー環境変数を設定する

環境の ~/.bash_profile ファイルにカスタムユーザー環境変数を設定できます。この動作をテストするには、次のコードを環境の ~/.bash_profile ファイルに追加します。

```
export MY_ENV_VAR='.bash_profile file'
```

次にコマンドラインから `sh ./script.sh` を実行すると、ターミナルに `.bash_profile file` が表示されます。(script.sh ファイルを前に説明したように作成済みであるとしします。)

ローカルのカスタム環境変数を設定する

export コマンドを実行して、ターミナルセッションでローカルのカスタム環境変数を設定できます。この動作をテストするには、ターミナルセッションで次のコマンドを実行します。

```
export MY_ENV_VAR='Command line export'
```

次にコマンドラインから `sh ./script.sh` を実行すると、ターミナルに `Command line export` が表示されます。(script.sh ファイルを前に説明したように作成済みであるとしします。)

同じカスタム環境変数を **export** コマンドを使用して ~/.bash_profile ファイルに設定した場合、カスタム環境変数の値を取得する際に、**export** コマンドの設定が優先されます。

~/.bashrc のカスタムユーザー環境変数を設定する

環境の ~/.bashrc ファイルにカスタムユーザー環境変数を設定できます。この動作をテストするには、次のコードを環境の ~/.bashrc ファイルに追加します。

```
export MY_ENV_VAR='.bashrc file'
```

次にコマンドラインから `sh ./script.sh` を実行すると、ターミナルに `.bashrc file` が表示されます。(script.sh ファイルを前に説明したように作成済みであるとします。)

同じカスタム環境変数を **export** コマンドを使用して `~/.bashrc` ファイルに設定した場合、カスタム環境変数の値を取得する際に、**export** コマンドの設定が優先されます。

ENV リストのカスタム環境変数を設定する

[Run (実行)] タブの [ENV] リストにカスタム環境変数を設定できます。

この動作をテストするには、以下を実行します。

1. メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
2. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Shell script (シェルスクリプト)] を選択します。
3. [ENV] を選択して、[MY_ENV_VARName (名前)] に と入力し、[ENV listValue (値)] には と入力します。
4. [Command (コマンド)] に「`./script.sh`」と入力します。
5. [Run (実行)] ボタンを選択します。ランナータブに ENV list が表示されます。(script.sh ファイルを前に説明したように作成済みであるとします。)

`~/.bash_profile` ファイルに同じカスタム環境変数を設定し、`~/.bashrc` ファイル、および [ENV] リストで、**export** コマンドを使用した場合、カスタム環境変数の値を取得する際に、`~/.bash_profile` ファイルの設定が優先されます。次に **export** コマンドの設定、`~/.bashrc` ファイルの設定、そして [ENV] リストの設定が続きます。

Note

[ENV] リストは、シェルスクリプトは別として、コードを使用してカスタム環境変数を取得および設定する唯一の方法です。

IDE AWS Cloud9 でのプロジェクト設定の操作

現在の AWS Cloud9 開発環境にのみ適用されるプロジェクト設定には、次の種類の設定が含まれます。

- ソフトタブを使用するかどうか、新しいファイル行終了などの、コードエディタ設定
- 無視するファイルタイプ
- 表示または抑制するヒントと警告の種類
- JavaScript、PHP、Python、および Go などのプログラミング言語のコードおよびフォーマットの設定
- 実行中およびビルド時に使用する設定の種類

プロジェクト設定が適用される環境は 1 つだけですが、1 つの環境のプロジェクト設定を他の環境に適用することができます。

- [プロジェクト設定の表示または変更](#)
- [環境の現在のプロジェクト設定を別の環境に適用](#)
- [プロジェクト設定のカスタマイズ](#)

プロジェクト設定の表示または変更

1. メニューバーで、AWS Cloud9、[設定] の順に選択します。
2. 現在の環境のプロジェクト設定を表示するには、[設定] タブのサイドナビゲーションペインで、[プロジェクト設定] を選択します。
3. 環境の現在のプロジェクト設定を変更するには、[Project Settings] (プロジェクト設定) ペインの設定を選択します。

プロジェクト設定の変更方法の詳細については、「[プロジェクト設定のカスタマイズ](#)」を参照してください。

環境の現在のプロジェクト設定を別の環境に適用

1. ソース環境とターゲット環境の両方で、IDE AWS Cloud9 のメニューバーで、AWS Cloud9 「プロジェクト設定を開く」を選択します。
2. ソース環境で、表示される[project.settings] タブの内容をコピーします。
3. ターゲット環境で、[project.settings] タブの内容を、ソース環境からコピーした内容で上書きします。
4. ターゲット環境で、[project.settings] タブを保存します。

プロジェクト設定のカスタマイズ

以下のセクションでは、[Preferences (設定)] タブの [Project Settings (プロジェクト設定)] ペインで変更できるプロジェクト設定の種類について説明します。

- [EC2 インスタンス](#)
- [コードエディタ \(Ace\)](#)
- [ファイルの検索](#)
- [ヒントと警告](#)
- [JavaScript のサポート](#)
- [Build](#)
- [実行およびデバッグ](#)
- [実行設定](#)
- [コードフォーマッタ](#)
- [TypeScript サポート](#)
- [PHP のサポート](#)
- [Python のサポート](#)
- [Go のサポート](#)

EC2 インスタンス

[Stop my environment (環境を停止する)]

環境の IDE に接続されたすべてのウェブブラウザインスタンスを閉じた後に、(使っていれば) 環境の Amazon EC2 インスタンスを自動的に停止するタイミングを選択します 1 週間から 30 分の範囲で期間を選択できます。IDE を終了した後に Amazon EC2 AWS Cloud9 インスタンスを自動的に停止しないように選択することもできます。

IDE の終了後 30 分未満でインスタンスを停止したい場合は、[コンソールインターフェイスを使用して手動で停止する](#)ことができます。

コードエディタ (Ace)

Soft Tabs (ソフトタブ)

選択すると、Tab を押すたびにタブ文字の代わりに指定された数のスペースが挿入されます。

Autodetect Tab Size on Load (ロード時にタブサイズを自動検出)

選択すると、タブサイズを推測 AWS Cloud9 しようとします。

New File Line Endings (新規ファイル改行位置)

新しいファイルに使用する行末のタイプです。

有効なオプションは以下のとおりです。

- [Windows (CRLF)] キャリッジリターンとラインフィードで行を終了します。
- [Unix (LF)] ラインフィードのみで行を終了します。

On Save, Strip Whitespace (保存時に空白を削除)

選択すると、ファイルが保存されるたびに、ファイルから不要なスペースやタブと見なされるものを削除 AWS Cloud9 しようとします。

ファイルの検索

[Ignore these Files (これらのファイルを無視)]

ファイルを検索する場合、 が AWS Cloud9 無視するファイルのタイプ。

[Maximum number of files to search (in 1000) (検索するファイルの最大数 (1000))]

ファイルを検索する場合、 が現在のスコープで AWS Cloud9 検出する 1,000 の倍数での最大ファイル数。

Hints and warnings (ヒントと警告)

警告レベル

有効にするメッセージの最小レベルです。

有効な値は次のとおりです。

- [Info (情報)] 情報、警告、およびエラーメッセージを有効にします。
- [Warning (警告)] 警告およびエラーメッセージを有効にします。
- [Error (エラー)] エラーメッセージだけを有効にします。

[Mark Missing Optional Semicolons (省略可能なセミコロンをマーク)]

有効にすると、コードで利用できるが使用されていないセミコロンに気づくたびに、 がファイルに AWS Cloud9 フラグを立てます。

[Mark Undeclared Variables (宣言されていない変数をマーク)]

有効にすると、コードで宣言されていない変数が検出されるたびに、`g` ファイルに AWS Cloud9 フラグを立てます。

[Mark Unused Function Arguments (未使用の関数引数をマーク)]

有効にすると、`h` は 関数で未使用の引数に気づくたびにファイルに AWS Cloud9 フラグを立てます。

[Ignore Messages Matching Regex (正規表現に一致するメッセージを無視する)]

AWS Cloud9 は、指定された正規表現に一致するメッセージを表示しません。詳細については、Mozilla 開発者ネットワークに掲載されている JavaScript 正規表現トピックの「[正規表現パターンの記述](#)」を参照してください。

JavaScript サポート

Customize JavaScript warnings with .eslintrc

有効にすると、`.eslintrc` ファイル AWS Cloud9 を使用して、有効または無効にする JavaScript 警告を決定します。詳細については、ESLint ウェブサイトの「[設定ファイルの形式](#)」を参照してください。

JavaScript library code completion (ライブラリコード補完)

AWS Cloud9 が自動コード補完を提案または実行するために使用する JavaScript ライブラリ。

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、JavaScript ファイルが保存されるたびにファイルのコードをフォーマット AWS Cloud9 しようとします。

Use builtin JSBeautify as code formatter (組み込み をコードフォーマッタとして使用する)

有効にすると、`i` の内部実装 AWS Cloud9 を使用して JSBeautify、ファイル内のコードの読みやすさの向上を試みます。

Custom Code Formatter (カスタムコードフォーマッタ)

ファイルでコードをフォーマットするときに `g` が実行 AWS Cloud9 を試みるコマンド JavaScript。

構築

Builder Path in environment (環境のビルダーパス)

カスタムビルド設定へのパスです。

Run and debug (実行とデバッグ)

Runner Path in Environment (環境のランナーパス)

カスタム実行設定へのパスです。

[Preview URL (プレビュー URL)]

環境のアプリケーションのプレビューに使用する URL。

実行設定

この環境設定をカスタム実行します。

Remove Selected Configs (選択した設定を削除)

選択した実行設定を削除します。

Add New Config (新しい設定を追加)

新しい実行設定を作成します。

Set As Default (デフォルトとして設定)

選択した実行設定をデフォルトの実行設定として設定します。

コードフォーマッタ

JSBeautify 設定

ファイル内のコードの可読性を高めるための設定。

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、コードファイルが保存されるたびにJSBeautify設定を適用 AWS Cloud9 しようします。

Use JSBeautify for JavaScript (のための の使用)

有効にすると、JavaScriptファイルが保存されるたびにJSBeautify設定を適用 AWS Cloud9 しようします。

Preserve Empty Lines (空の行を保持する)

有効にした場合、コードファイル内の空の行は削除 AWS Cloud9 されません。

Keep Array Indentation (配列のインデントを保持)

有効にすると、コードファイルの配列に要素宣言のインデント AWS Cloud9 が保持されます。

JSLint Strict Whitespace (JSLint 厳密な空白)

有効にすると、コードファイルに JSLint 空白ルールを適用 AWS Cloud9 しようします。詳細については、[JSLint Help](#) の「Whitespace」を参照してください。

[Braces (中括弧)]

コードでの中括弧の配置を指定します。

有効な値は次のとおりです。

- [Braces with control statement (制御ステートメントを含む中括弧)] では、必要に応じて、開始と終了の各中括弧を関連する制御ステートメントに合わせて移動します。

例えば、このコードは次のような形式になっています。

```
for (var i = 0; i < 10; i++) { if (i == 5) { console.log("Halfway done.") } }
```

ファイルを保存するときにこのコードに変わります。

```
for (var i = 0; i < 10; i++) {  
    if (i == 5) {  
        console.log("Halfway done.")  
    }  
}
```

- [Braces on own line (中括弧ごとに独自の行)] は、必要に応じて各中括弧をそれぞれの行に移動します。

例えば、このコードは次のような形式になっています。


```
for (var i = 0; i < 10; i++) { if (i == 5) { console.log("Halfway done.") }}
```

ファイルを保存するときにこのコードに変わります。

```
for (var i = 0; i < 10; i++) {if (i == 5)
{
    console.log("Halfway done.")
}
}
```

- [End Braces on own line (終了括弧ごとに独自の行)] は、必要に応じて各終了括弧をそれぞれの行に移動します。

例えば、このコードは次のような形式になっています。

```
for (var i = 0; i < 10; i++) {
    if (i == 5) { console.log("Halfway done.") }
}
```

ファイルを保存するときにこのコードに変わります。

```
for (var i = 0; i < 10; i++) {
    if (i == 5) {
        console.log("Halfway done.")
    }
}
```

Preserve Inline Blocks (インラインブロックの保持)

有効にした場合、インラインブロックの開始中括弧と終了中括弧が同じ行にある場合、それらの中括弧を別々の行に移動しようと AWS Cloud9 しません。

Space Before Conditionals (条件の前の空白)

有効にした場合、必要に応じて各条件宣言の前にスペース AWS Cloud9 を追加します。

Unescape Strings (未エスケープ文字列)

有効にすると、エスケープされた文字列をエスケープされていない同等の文字列 AWS Cloud9 に変換します。たとえば、`\n` を改行文字に変換し、`\r` をキャリッジリターン文字に変換します。

Indent Inner Html (インデント内部 Html)

有効にすると、HTML コードの AWS Cloud9 インデント<head>と<body>セクションが表示されます。

TypeScript サポート

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、TypeScriptファイルが保存されるたびにTypeScriptコードをフォーマット AWS Cloud9 しようとします。

Custom Code Formatter (カスタムコードフォーマッタ)

TypeScript コードのカスタムコードの書式設定へのパスです。

PHP サポート

Enable PHP code completion (コード補完を有効にする)

有効にすると、PHPコードを完了しようと AWS Cloud9 します。

PHP Completion Include Paths (パスを含む 補完)

コードを完了するために AWS Cloud9 が使用する場所PHP。例えば、AWS Cloud9 が補完に使用するカスタム PHPPHP ファイルが ~/environment ディレクトリのどこかにある場合、このパスに ~/environment を追加します。

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、PHPファイルが保存されるたびにPHPコードをフォーマット AWS Cloud9 しようとします。

Custom Code Formatter (カスタムコードフォーマッタ)

PHP コードのカスタムコードの書式設定へのパスです。

Python サポート

Enable Python code completion (コード補完を有効にする)

有効にすると、Pythonコードを完了しようと AWS Cloud9 します。がPythonコードを完了 AWS Cloud9 するために使用するパスを設定するには、PYTHONPATH 設定を使用します。

Python バージョン

使用する Python のバージョンを指定します。

PyLint command line options (コマンドラインオプション)

Python コードPylintで AWS Cloud9 に使用する のオプション。詳細については、PyLint ウェブサイトの「[PyLint ユーザーマニュアル](#)」を参照してください。

PYTHONPATH

AWS Cloud9 で使用する のPythonライブラリとパッケージへのパス。たとえば、~/environment ディレクトリにカスタム Python ライブラリとパッケージがある場合は、このパスに ~/environment を追加します。

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、Pythonファイルが保存されるたびにPythonコードをフォーマット AWS Cloud9 しようとします。

Custom Code Formatter (カスタムコードフォーマッタ)

Python コードのカスタムコードの書式設定へのパスです。

Go サポート

Enable Go code completion (コード補完を有効にする)

有効にすると、Goコードを完了しようと AWS Cloud9 します。

[Format Code on Save (保存時にコードを形式設定)]

有効にすると、Goファイルが保存されるたびにGoコードをフォーマット AWS Cloud9 しようとします。

Custom Code Formatter (カスタムコードフォーマッタ)

Go コードのカスタムコードの書式設定へのパスです。

環境のEC2 インスタンスを手動で停止する

[EC2 インスタンス](#)設定を使用すると、IDE に接続したすべてのウェブブラウザインスタンスを閉じてから 30 分後に、すぐに、環境の Amazon EC2 インスタンスを自動的に停止できます

コンソールを使用してインスタンスをすぐに手動で停止することもできます。

環境の EC2 インスタンスを手動で停止するには、次の手順を選択します。

1. IDE に接続されているすべてのウェブブラウザインスタンスを閉じたら、AWS Cloud9 コンソールで環境を選択します。
2. ペインの右上にある、使用していた環境の詳細を表示するボタンを選択し、詳細を表示を選択します。
3. 環境の詳細の中、EC2 インスタンスの下でインスタンスに移動を選択します。
4. Amazon EC2 コンソールの[Instance state] (インスタンスの状態) で、環境のインスタンスを選択するチェックボックスを選択します。[Instance state] (インスタンスの状態) は、インスタンスがまだ実行中であることを示している可能性があります。
5. [Instance state (インスタンスの状態)] を選択し、[Stop instance (インスタンスの停止)] を選択します。
6. 確認を求められたら、[Stop] を選択してください。インスタンスが停止するまで、数分かかる場合があります。

IDE AWS Cloud9 でのユーザー設定の操作

ユーザー設定は、AWS Identity and Access Management (IAM ユーザー) に関連付けられている各 AWS Cloud9 開発環境に適用される設定です。これらには、以下の設定が含まれます。

- アニメーションの有効化や変更されたタブのマーク付けなど、一般的なユーザーインターフェイスの設定
- File system navigation settings (ファイルシステムのナビゲーション設定)
- File find and search settings (ファイル検索と検索の設定)
- ターミナルセッションと出力の配色
- その他のコードエディタの設定 (フォントサイズ、コードの折りたたみ、全行選択、スクロールアニメーション、フォントサイズなど)

ユーザー設定を変更すると、はそれらの変更をクラウドに AWS Cloud9 プッシュし、IAM ユーザーと関連付けます。AWS Cloud9 また、は、IAM ユーザーに関連付けられたユーザー設定の変更についてクラウドを継続的にスキャンし、それらの設定を現在の環境に適用します。これを使用して、作業している AWS Cloud9 環境に関係なく、同じルックアンドフィールを体験できます。

Note

IDE 設定を保存および取得するには、は内部 APIs `getUserSettings` と `AWS Cloud9` を使用します `updateUserSettings`。

ユーザー設定は、次のように他のユーザーと共有できます。

- [ユーザー設定の表示または変更](#)
- [ユーザー設定の他のユーザーとの共有](#)
- [ユーザー設定のカスタマイズ](#)

ユーザー設定の表示または変更

1. メニューバーで、AWS Cloud9、[設定] の順に選択します。
2. 各環境のユーザー設定を表示するには、[Preferences (設定)] タブのサイドナビゲーションペインで、[User Settings (ユーザー設定)] を選択します。
3. [User Settings (ユーザー設定)] ペインで、各環境のユーザー設定を変更します。
4. 他のいずれかの環境に変更を適用するには、その環境を単に開きます。その環境が既に開いている場合は、その環境のウェブブラウザタブを最新の情報に更新します。

ユーザー設定の変更方法の詳細については、「[ユーザー設定のカスタマイズ](#)」を参照してください。

ユーザー設定の他のユーザーとの共有

1. ソース環境とターゲット環境の両方で、IDE AWS Cloud9 のメニューバーで、AWS Cloud9 「ユーザー設定を開く」を選択します。
2. ソース環境で、表示される [user.settings] タブの内容をコピーします。
3. ターゲット環境で、[user.settings] タブの内容を、ソース環境からコピーした内容で上書きします。
4. ターゲット環境で、[user.settings] タブを保存します。

ユーザー設定のカスタマイズ

以下のセクションでは、[Preferences] (設定) タブの [User Settings] (ユーザー設定) ペインで変更できるユーザー設定の種類について説明します。

- [全般](#)
- [User interface](#) (ユーザーインターフェイス)
- [コラボレーション](#)
- [Tree および Go パネル](#)
- [Find in files](#) (ファイルの検索)
- [Meta data](#) (メタデータ)
- [ウォッチャー](#)
- [ターミナル](#)
- [出力](#)
- [コードエディタ \(Ace\)](#)
- [Input](#) (入力)
- [Hints and warnings](#) (ヒントと警告)
- [Run and debug](#) (実行とデバッグ)
- [プレビュー](#)
- [Build](#)

General

Reset to Factory Settings (工場出荷時の設定にリセットする)

デフォルトへのリセットボタンを選択すると、はすべてのユーザー設定を AWS Cloud9 デフォルトのユーザー設定に AWS Cloud9 リセットします。確定するには、 [Reset settings (設定をリセットする)] を選択します。

Warning

このアクションは元に戻すことができません。

Warn Before Exiting (終了する前に警告する)

IDE を閉じようとするたびに、は終了することを確認するように AWS Cloud9 求めます。

[ユーザーインターフェイス]

Enable UI Animations (UI アニメーションを有効にする)

AWS Cloud9 は IDE でアニメーションを使用します。

Use an Asterisk (*) to Mark Changed Tabs (アスタリスク (*) を使用して変更されたタブをマークする)

AWS Cloud9 は、変更があるがコンテンツがまだ保存されていないタブにアスタリスク (*) を追加します。

Display Title of Active Tab as Browser Title (アクティブなタブのタイトルをブラウザのタイトルとして表示する)

AWS Cloud9 は、関連付けられたウェブブラウザタブのタイトルをアクティブなタブのタイトル (Untitled1、hello.js、ターミナル、設定など) に変更します。

Automatically Close Empty Panes (空のペインを自動的に閉じる)

環境を再ロードするたびに、は空のペイン AWS Cloud9 を自動的に閉じます。

Environment Files Icon and Selection Style (環境ファイルのアイコンと選択スタイル)

アイコンは環境ファイルに AWS Cloud9 を使用し、ファイル選択動作 AWS Cloud9 は を使用します。

有効な値を次に示します。

- デフォルト – デフォルトのアイコンとデフォルトのファイル選択動作 AWS Cloud9 を使用します。
- 代替 – 代替アイコンと代替ファイル選択動作 AWS Cloud9 を使用します。

コラボレーション

Disable collaboration security warning (コラボレーションセキュリティ警告を無効にする)

読み取り/書き込みメンバーが環境に追加されると、セキュリティ警告ダイアログボックスは表示 AWS Cloud9 されません。

Show Authorship Info (著作権情報を表示する)

AWS Cloud9 は、他の環境メンバーが入力したテキストに、ガーターの関連するハイライトを付けます。

Tree および Go パネル

Scope Go to Anything to Favorites (どこにでも移動の対象をお気に入りにする)

[Go] ウィンドウのGo (ファイルに移動は、[環境] ウィンドウのお気に入りに表示されます。

Enable Preview on Tree Selection (ツリーの選択でプレビューを有効にする)

AWS Cloud9 は、ダブルクリックではなくワンクリックで選択したファイルを表示します。

Hidden File Pattern (非表示ファイルのパターン)

AWS Cloud9 が非表示として扱うファイルのタイプ。

Reveal Active File in Project Tree (プロジェクトツリーのアクティブなファイルを表示する)

AWS Cloud9 は、環境ウィンドウでアクティブなファイルを強調表示します。

Download Files As (ファイルをダウンロードする形式)

ファイルのダウンロード時に AWS Cloud9 が使用する動作。

有効な値は次のとおりです。

- auto – 変更せずにファイル AWS Cloud9 をダウンロードします。
- tar.gz – 圧縮ファイルとしてTARファイル AWS Cloud9 をダウンロードします。
- zip – ファイルとして.zipファイル AWS Cloud9 をダウンロードします。

ファイルの検索

Search In This Path When 'Project' Is Selected (「プロジェクト」が選択されている場合このパスを検索する)

ファイルの検索バーで、検索範囲として [Project (プロジェクト)] を選択した場合に検索対象となるパス。

Show Full Path in Results (結果に完全パスを表示する)

[Search Results (検索結果)] タブに、一致する各ファイルの完全パスを表示します。

Clear Results Before Each Search (各検索の前に結果をクリアする)

[Search Results (検索結果)] タブから以前の検索結果をクリアした後で、現在の検索を開始します。

Scroll Down as Search Results Come In (検索結果の受信に合わせて下にスクロールする)

検索結果の表示に合わせて [Search Results (検索結果)] タブで検索結果の下部にスクロールします。

Open Files when Navigating Results with (Up and Down) (結果を (上下に) 移動するときにファイルを開く)

[Search Results (検索結果)] タブの結果リストで上下の矢印キーを押すたびに、一致する各ファイルを開きます。

メタデータ

Maximum of Undo Stack Items in Meta Data (メタデータ内の元に戻す項目の最大数)

元に戻すことができるアクションのリストに が AWS Cloud9 保持する項目の最大数。

ウォッチャー

Auto-Merge Files When a Conflict Occurs (ファイルが競合した場合は自動的にマージする)

AWS Cloud9 は、マージの競合が発生するたびにファイルを自動的にマージしようとします。

Terminal

文字色

[Terminal (ターミナル)] タブのテキストの色。

Background Color (背景色)

[Terminal (ターミナル)] タブの背景の色。

Selection Color (選択色)

[Terminal (ターミナル)] タブの選択されたテキストの色。

[Font Family] (フォントファミリー)

[Terminal (ターミナル)] タブのテキストのフォントスタイル。
フォントのサイズ

[Terminal (ターミナル)] タブのテキストのサイズ。
Antialiased Fonts (アンチエイリアスフォント)

AWS Cloud9 はターミナルタブでのテキストの表示を滑らかにしようとします。
Blinking Cursor (カーソル点滅)

AWS Cloud9 はターミナルタブでカーソルを継続的に点滅させます。
Scrollback (スクロールバック)

[Terminal (ターミナル)] タブで上下にスクロールできる行数。
をデフォルトエディタ AWS Cloud9 として使用する

をデフォルトのテキストエディタ AWS Cloud9 として使用します。

Output

文字色

出力を表示するタブのテキストの色。

Background Color (背景色)

出力を表示するタブのテキストの背景の色。

Selection Color (選択色)

出力を表示するタブの選択されたテキストの色。

Warn Before Closing Unnamed Configuration (名前を付けずに設定を閉じる前に警告する)

AWS Cloud9 を閉じる前に、保存されていない設定タブを保存するように求められます。

Preserve log between runs (実行間でログを保持)

AWS Cloud9 は、試行されたすべての実行のログを保持します。

コードエディタ (Ace)

Auto-pair Brackets, Quotes, etc. (かっこや引用符などを自動的にペアにする)

AWS Cloud9 は、角括弧、引用符、中括弧など、エディタタブで入力される関連する開始文字ごとに一致する終了文字を追加しようとします。

Wrap Selection with Brackets, Quote, etc. (選択部分をかっこや引用符などでくくる)

AWS Cloud9 は、テキストが選択され、括弧、引用符、中括弧などの関連する開始文字が入力された後、エディタタブのテキストの末尾に一致する終了文字を挿入しようとします。

Code Folding (コードの折りたたみ)

AWS Cloud9 は、関連するコード構文ルールに従って、エディタタブでコードのセクションを表示、展開、非表示、または折りたたもうとします。

Fade Fold Widgets (折りたたみウィジェットのフェード)

AWS Cloud9 エディタタブのコントロールの上にマウスを置くと、ガーターにコード折りたたみコントロールが表示されます。

空のテキストを選択してコピー

AWS Cloud9 では、テキストをコピーまたはカットできます。このオプションは、空のテキストをクリップボードにコピーするかどうかを決定します。

Full Line Selection (行全体の選択)

AWS Cloud9 は、エディタタブでトリプルクリックされた行全体を選択します。

Highlight Active Line (アクティブな行をハイライトする)

AWS Cloud9 はエディタタブでアクティブな行全体を強調表示します。

Highlight Gutter Line (ガーター行をハイライトする)

AWS Cloud9 は、エディタタブのアクティブな行の横にあるガーター内の場所を強調表示します。

Show Invisible Characters (非表示文字を表示する)

AWS Cloud9 は、改行、改行、スペース、タブなど、エディタタブで非表示文字と見なす内容を表示します。

Show Gutter (ガーターを表示する)

AWS Cloud9 にガーターが表示されます。

Show Line Numbers (行番号を表示する)

ガーターに行番号を表示する動作。

有効な値は次のとおりです。

- Normal (通常) – 行番号を表示します。
- Relative (相対) – アクティブな行を基準にした行番号を表示します。
- None (なし) – 行番号を非表示にします。

Show Indent Guides (インデントガイドを表示する)

AWS Cloud9 は、エディタタブでインデントされたテキストをより簡単に視覚化するためのガイドを表示します。

Highlight Selected Word (選択された単語をハイライトする)

AWS Cloud9 は、エディタタブでダブルクリックする単語全体を選択します。

Scroll Past the End of the Document (ドキュメントの末尾を超えてスクロールする)

エディタタブでユーザーが現在のファイルの末尾を超えてスクロールすることを許可する動作です。

有効な値は次のとおりです。

- Off (オフ) – 現在のファイルの末尾を超えたスクロールは許可されません。
- Half Editor Height (エディタの高さの半分) – エディタ画面の高さの半分まで、現在のファイルの末尾を超えたスクロールを許可します。
- Full Editor Height (エディタの高さ全体) – エディタ画面の高さ全体まで、現在のファイルの末尾を超えたスクロールを許可します。

Animate Scrolling (スクロールのアニメーション)

AWS Cloud9 は、エディタタブでスクロールアクション中にアニメーション動作を適用します。

[Font Family] (フォントファミリー)

エディタタブで使用するフォントのスタイルです。

フォントのサイズ

エディタタブで使用するフォントのサイズです。

Antialiased Fonts (アンチエイリアスフォント)

AWS Cloud9 はエディタタブでのテキストの表示を滑らかにしようとします。

Show Print Margin (印刷マージンを表示する)

エディタタブで指定された文字の位置の後に垂直線を表示します。

Mouse Scroll Speed (マウスのスクロール速度)

エディタタブでのマウスのスクロールの相対速度です。値が大きいほどスクロールが速くなります。

Cursor Style (カーソルのスタイル)

エディタタブでのポインターのスタイルと動作です。

有効な値を次に示します。

- Ace (エース) – ポインターを縦の棒として表示します。[Slim] (スリム) よりも幅が広くなります。
- Slim (スリム) – ポインターを細めの縦の棒として表示します。
- Smooth (スムーズ) – ポインターを[Slim] (スリム) よりも幅が広い縦の棒として表示し、[Slim] (スリム) よりも滑らかに点滅させます。
- Smooth and Slim (スムーズでスリム) – ポインターを細めの縦の棒として表示し、[Slim] (スリム) よりも滑らかに点滅させます。
- Wide (ワイド) – ポインターを広めの縦の棒として表示します。

Merge Undo Deltas (マージを元に戻すデルタ)

- Always (常に) – マージ競合を元に戻すことを許可します。
- Never (行わない) – マージ競合を元に戻すことを許可しません。
- Timed (時間指定) – 指定した時間後に、マージ競合を元に戻すことを許可します。

Enable Wrapping For New Documents (新しいドキュメントのラッピングを有効にする)

AWS Cloud9 はコードを新しいファイルにラップします。

Input

Complete As You Type (入力しながら補完する)

AWS Cloud9 は、入力時に可能なテキスト補完を表示しようとします。

Complete On Enter (Enter 後に補完)

AWS Cloud9 Enter キーを押すと、はテキスト補完の可能性を表示しようとします。

Highlight Variable Under Cursor (カーソルが置かれた変数をハイライトする)

AWS Cloud9 は、選択した変数へのコード内のすべての参照を強調表示します。

Use Cmd-Click for Jump to Definition (Cmd クリックを使用して定義にジャンプする)

AWS Cloud9 は、Command for Mac または Ctrl for Windows を押しながら選択したコードの元の定義に移動します。

ヒントと警告

Enable Hints and Warnings (ヒントと警告を有効にする)

AWS Cloud9 は、該当するヒントメッセージと警告メッセージを表示します。

クリック時に利用可能なクイックフィックスを表示

AWS Cloud9 コード内のキーワードをクリックすると、にリファクタリングの提案を含むツールヒントが表示されます。

[Ignore Messages Matching Regex (正規表現に一致するメッセージを無視する)]

AWS Cloud9 は、指定された正規表現に一致するメッセージを表示しません。詳細については、Mozilla 開発者ネットワークに掲載されている [JavaScript 正規表現](#) トピックの Writing a regular expression pattern を参照してください。

実行およびデバッグ

Save All Unsaved Tabs Before Running (実行前に未保存のすべてのタブを保存する)

関連付けられたコードを実行する前に、AWS Cloud9 は未保存のすべてのファイルを開いているタブで保存しようとします。

プレビュー

Preview Running Apps (実行中のアプリケーションをプレビューする)

AWS Cloud9 は、プレビューボタンが選択されるたびに、アクティブなタブにコードの出力のプレビューを表示しようとします。

Default Previewer (デフォルトのプレビューア)

形式は、AWS Cloud9 を使用してコード出力をプレビューします。

有効な値を次に示します。

- Raw (未加工) – コード出力をプレーンな形式で表示します。
- Browser (ブラウザ) – コード出力をウェブブラウザの推奨形式で表示します。

When Saving Reload Previewer (保存時にプレビューアを再ロードする)

この動作では、コードファイルが保存されるたびに AWS Cloud9 を使用してコード出力をプレビューします。

有効な値は次のとおりです。

- Only on Ctrl-Enter (Ctrl-Enter を押した場合のみ) – 現在のコードタブで Ctrl+Enter を押すたびにコード出力をプレビューします。
- Always (常に) – コードファイルを保存するたびにコード出力をプレビューします。

構築

Automatically Build Supported Files (サポートされるファイルを自動的にビルドする)

AWS Cloud9 ビルドアクションが開始され、コードがサポートされている形式である場合、は現在のコードを自動的にビルドしようとします。

AWS Cloud9 IDE での AWS プロジェクトおよびユーザー設定の操作

AWS のサービス 設定 タブAWS の設定ペインにある設定には、次の設定が含まれます。

- AWS リソースウィンドウ AWS リージョン に使用するもの
- AWS マネージド一時認証情報を使用するかどうか
- AWS Serverless Application Model (AWS SAM) テンプレートエディタをプレーンテキストまたはビジュアルモードで表示するかどうか

これらの設定を表示または変更するには、環境の IDE のメニューバーで [AWS Cloud9 Preferences (設定)] を選択します。

次のリストでは、プロジェクトレベルの設定は現在の AWS Cloud9 開発環境にのみ適用されます。一方、ユーザーレベル設定は、IAM ユーザーに関連付けられた各環境に適用されます。詳細については、「[環境の現在のプロジェクト設定を別の環境に適用する](#)」および「[ユーザー設定を他のユーザーと共有する](#)」を参照してください。

- [プロジェクトレベル設定](#)
- [ユーザーレベル設定](#)

プロジェクトレベル設定

AWS リージョン

AWS リソースウィンドウの Lambda セクション AWS リージョン に使用するもの。

AWS マネージド一時認証情報

オンにすると、環境 AWS のサービスの AWS CLI、または AWS SDK コードから を呼び出すときに AWS CloudShell、AWS マネージド一時認証情報が使用されます。詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。

ユーザーレベル設定

AWS SAM ビジュアルエディタを使用する

オンにすると、AWS リソースウィンドウの Lambda セクションを使用すると、AWS Serverless Application Model (AWS SAM) テンプレートエディタがビジュアルモードで表示されます。オフにすると、エディタはテキストモードで表示されます。

IDE AWS Cloud9 でのキー割り当ての操作

キー割り当ては、ショートカットキーの組み合わせを定義します。キー割り当ては、IAM ユーザーに関連付けられている各 AWS Cloud9 開発環境に適用されます。キー割り当てを変更すると、はそれらの変更をクラウドに AWS Cloud9 プッシュし、IAM ユーザーと関連付けます。AWS Cloud9 また、は、IAM ユーザーに関連付けられたキー割り当ての変更についてクラウドを継続的にスキャンし、それらの変更を現在の環境に適用します。

キー割り当ては他のユーザーと共有できます。

- [キー割り当ての表示または変更](#)

- [別のユーザーとキー割り当てを共有する](#)
- [キーボードモードを変更する](#)
- [オペレーティングシステムのキー割り当てを変更する](#)
- [特定のキー割り当てを変更する](#)
- [すべてのカスタムキー割り当てを削除する](#)

キー割り当ての表示または変更

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. お客様の各環境間のキー割り当てを表示するには、[Preferences (設定)] タブのサイドナビゲーションペインで、[Keybindings (キー割り当て)] を選択します。
3. 各環境間のキー割り当てを変更するには、[Keybindings] (キー割り当て) ペインで必要な設定を変更します。
4. 他の環境に変更を適用するには、単純にその環境を開きます。その環境が既に開いている場合は、その環境のウェブブラウザタブを最新の情報に更新します。

詳細については次を参照してください:

- [MacOS デフォルトキー割り当てリファレンス](#)
- [MacOS Vim キー割り当てリファレンス](#)
- [MacOS Emacs キー割り当てリファレンス](#)
- [MacOS Sublime キー割り当てリファレンス](#)
- [Windows/Linux デフォルトキー割り当てリファレンス](#)
- [Windows/Linux Vim キー割り当てリファレンス](#)
- [Windows/Linux Emacs キー割り当てリファレンス](#)
- [Windows/Linux Sublime キー割り当てリファレンス](#)

別のユーザーとキー割り当てを共有する

1. ソース環境とターゲット環境の両方で、IDE AWS Cloud9 のメニューバーで、AWS Cloud9 「キーマップを開く」を選択します。
2. ソース環境で、表示される [keybindings.settings] タブの内容をコピーします。

3. ターゲット環境で、[keybindings.settings] タブの内容を、ソース環境からコピーした内容で上書きします。
4. ターゲット環境で、[keybindings.settings] タブを保存します。

キーボードモードを変更する

AWS Cloud9 IDE がエディタでテキストを操作するために使用するキーボードモードは、IAM ユーザーに関連付けられた各環境で変更できます。

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. [Preferences (設定)] タブで、サイドナビゲーションペインの [Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、以下のいずれかのキーボードモードを選択します。
 - [デフォルト] では、デフォルトのキー割り当てを使用します。
 - [Vim] では Vim モードを使用します。詳細については、[Vim ヘルプファイル](#)のウェブサイト参照してください。
 - [Emacs] では Emacs モードを使用します。詳細については、GNU オペレーティングシステムのウェブサイトの「[Emacs エディタ](#)」を参照してください。
 - [Sublime] では Sublime モードを使用します。詳細については、[Sublime Text ドキュメント](#)のウェブサイト参照してください。

オペレーティングシステムのキー割り当てを変更する

AWS Cloud9 IDE が IAM ユーザーに関連付けられた各環境全体で認識するオペレーティングシステムのキー割り当てのセットを変更できます。

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. [Preferences (設定)] タブで、サイドナビゲーションペインの [Keybindings (キー割り当て)] を選択します。
3. [Operating System (オペレーティングシステム)] で、以下のいずれかのオペレーティングシステムを選択します。
 - Auto for the AWS Cloud9 IDE to attempt to detect which set of operating system keybindings to use。
 - MacOS for the AWS Cloud9 IDE は、macOS 形式でリストされているキー割り当てを使用します。

- Windows / Linux for the AWS Cloud9 IDE は、Windows および Linux 形式でリストされているキー割り当てを使用します。

特定のキー割り当てを変更する

IAM ユーザーに関連付けられた各環境の個々のキー割り当てを変更することができます。

同時に 1 つのキー割り当てを変更するには

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. [Preferences (設定)] タブで、サイドナビゲーションペインの [Keybindings (キー割り当て)] を選択します。
3. キー割り当てのリストで、変更するキー割り当てを [Keystroke] (キーストローク) 列で開きます (ダブルクリック)。
4. キーボードを使用して置換キーの組み合わせを指定し、Enter を押します。

Note

現在のキーの組み合わせを完全に削除するには、Windows または Linux の場合は Backspace を、macOS の場合は Delete を押します。

同時に複数のキー割り当てを変更するには

1. メニューバーで、[AWS Cloud9]、[Open Your Keymap (キーマップを開く)] の順に選択します。
2. `keybindings.settings` ファイルで、変更する各キー割り当てを定義します。構文の例を次に示します。

```
[
  {
    "command": "addfavorite",
    "keys": {
      "win": ["Ctrl-Alt-F"],
      "mac": ["Ctrl-Option-F"]
    }
  },
  {
```

```
"command": "copyFilePath",
"keys": {
  "win": ["Ctrl-Shift-F"],
  "mac": ["Alt-Shift-F"]
}
}
```

この例では、addFavorite と copyFilePath は、[設定] タブの [Keybindings (キー割り当て)] ペインの [Keystroke (キーストローク)] 列のキー割り当ての名前です。Windows または Linux および macOS の場合、必要なキー割り当ては、win と mac です。

変更を適用するには、keybindings.settings ファイルを保存します。変更は、少し時間が経過すると、[Keybindings] (キー割り当て) ペインに表示されます。

すべてのカスタムキー割り当てを削除する

すべてのカスタムキー割り当てを削除し、すべてのキー割り当てを自分の IAM ユーザーに関連付けられた各環境のデフォルト値に復元できます。

Warning

このアクションは元に戻すことができません。

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. [Preferences (設定)] タブで、サイドナビゲーションペインの [Keybindings (キー割り当て)] を選択します。
3. [Reset to Defaults (デフォルトにリセット)] を選択します。

IDE AWS Cloud9 でのテーマの使用

テーマは、IDE 全体の色を定義します。これは、IAM ユーザーに関連付けられたそれぞれの AWS Cloud9 開発環境 に適用されます。テーマを変更すると、はそれらの変更をクラウドに AWS Cloud9 プッシュし、IAM ユーザーと関連付けます。AWS Cloud9 また、は、IAM ユーザーに関連付けられたテーマの変更についてクラウドを継続的にスキャンします。は、これらの変更を現在の環境 AWS Cloud9 に適用します。

- [View or change your theme](#) (テーマの表示または変更)
- [Overall theme settings you can change](#) (変更可能なテーマの全体的な設定)
- [Theme overrides](#) (テーマの上書き)

テーマの表示または変更

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. お客様の各環境間のテーマを表示するには、[Preferences (設定)] タブのサイドナビゲーションペインで、[テーマ] を選択します。
3. お客様の各環境間のテーマを変更するには、[テーマ] ペインで目的の設定を変更します。コードを使用してテーマの一部を変更するには、[your stylesheet (スタイルシート)] リンクを選択します。
4. お客様の環境に変更を適用するには、その環境を開きます。その環境が既に開いている場合は、その環境のウェブブラウザタブを最新の情報に更新します。

変更可能なテーマの全体的な設定

[テーマ] ペインの [設定] タブで、次の種類のテーマ設定を変更できます。

Flat Theme (フラットテーマ)

IDE AWS Cloud9 全体に組み込みのフラットテーマを適用します。

Classic Theme (クラシックテーマ)

選択した組み込みのクラシックテーマを IDE AWS Cloud9 全体に適用します。

Syntax Theme (構文テーマ)

選択したテーマを IDE AWS Cloud9 全体のコードファイルに適用します。

Theme overrides (テーマの上書き)

Important

AWS Cloud9 は、`styles.css` ファイルを更新して IDE テーマを上書きすることをユーザーに許可する機能をサポートしなくなりました。ユーザーはエディタを使っ

て、`styles.css` ファイルを引き続き表示、編集、保存できます。ただし、IDE AWS Cloud9 のロード時にテーマオーバーライドは適用されません。

が `styles.css` ファイルが変更されたこと AWS Cloud9 を検出すると、IDE に次のメッセージが表示されます。

テーマの上書き Support は中止されました。この `styles.css` ファイルの内容は、IDE AWS Cloud9 のロードに適用されなくなります。

IDE のテーマを定義するためにスタイルシートを使用する必要がある場合は、[お問い合わせ](#) を直接実行します。

IDE AWS Cloud9 での初期化スクリプトの管理

Important

AWS Cloud9 は、ユーザーが初期化スクリプトをカスタマイズできる実験的な機能をサポートしなくなりました。以前は、このスクリプトは IDE で自動的に実行されました。ユーザーはエディタを使って、`init.js` ファイルを引き続き表示、編集、保存できます。ただし、カスタマイズした初期化スクリプトの実行は許可されなくなり、IDE の動作を変更できなくなります。

が `init.js` ファイルが変更されたこと AWS Cloud9 を検出すると、IDE に次のメッセージが表示されます。

初期化スクリプトの Support は終了しました。この `init.js` ファイルの内容は、IDE AWS Cloud9 のロード時に実行されなくなります。

IDE のカスタム初期化スクリプトを実行する必要がある場合は、[当社にお問い合わせ](#) ください。

初期化スクリプトは、すべてのプラグインのロード後に IDE で実行するカスタム初期化コードを定義します。これは、IAM ユーザーに関連付けられている各 AWS Cloud9 開発環境に適用されます。AWS Cloud9 また、は初期化スクリプトの変更を継続的にスキャンし、変更が発生した場合はユーザーに警告します。

初期化スクリプトを再実行する

初期化スクリプトを表示するには、メニューバーで、[AWS Cloud9]、[Open Your Init Script (初期化スクリプトを開く)] の順に選択します。

⚠ Important

エディタを使って `init.js` ファイルを編集して保存できますが、カスタマイズしたスクリプトは IDE で実行できません。

IDE の MacOS AWS Cloud9 デフォルトキー割り当てリファレンス

このステップでは、MacOS デフォルトキー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Default (デフォルト)] を選択します。
4. [Operating system (オペレーティングシステム)] で、[MacOS] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、AWS Cloud9 統合開発環境 (IDE) の MacOS オペレーティングシステムのデフォルトのキーボードモードキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Command-Shift-C	addwatchfromselection

説明	割り当て	コマンド
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Control-Space Option-Space	complete
コードを補完した後、上書きする	Control-Shift-Space Option-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Command-C	copy
選択したものをクリップボードに切り取る	Command-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Command-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Control-Option-G	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Command-G	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Command-Shift-G	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences

説明	割り当て	コマンド
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Command-Shift-B	formatcode
行に移動するボックスを表示する	Command-L	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Command-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Control-N	newfile
[Preferences (設定)] タブを表示する	Command-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Command-Option-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Command-V	paste

説明	割り当て	コマンド
エラーを修正するための候補を表示する	Command-F3	quickfix
前回のアクションを繰り返す	Command-Shift-Z Command-Y	redo
プレビューペインを最新の情報に更新する	Command-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Option-Command-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Option-Command-F	replace
初期化スクリプトを再実行する	Command-Enter	rerunInitScript
環境の再起動	Command-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Control-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Option-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Command-S	save
現在のファイルを別のファイル名でディスクに保存する	Command-Shift-S	saveas

説明	割り当て	コマンド
複数のファイルで検索と置換バーを表示する	Shift-Command-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Command-Option-P	showprocesslist
前回のアクションを元に戻す	Command-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Option-Control-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Option-Shift-W	closealltabs
現在のペインを閉じる	Command-Control-W	closepane
現在のタブを閉じる	Option-W	closetab
下のペインに移動	Control-Command-Down	gotopanedown
左のペインに移動	Control-Command-Left	gotopaneleft
右のペインに移動	Control-Command-Right	gotopaneright
上のペインに移動	Control-Command-Up	gottopaneup
左のタブに移動	Command-[	gototableft
右のタブに移動	Command-]	gototabright

説明	割り当て	コマンド
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Command-Option-Shift-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Command-Option-Shift-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Command-Option-Shift-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Command-Option-Shift-Up	movetabup
次のペインに移動する	Option-Esc	nextpane
次のタブに移動する	Option-Tab	nexttab
前のペインに移動する	Option-Shift-Esc	previouspane
前のタブに移動する	Option-Shift-Tab	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Option-Shift-T	reopenLastTab
ファイルツリーで現在のタブを表示する	Command-Shift-L	revealtab
10 番目のタブに移動する	Command-0	tab0
最初のタブに移動する	Command-1	tab1

説明	割り当て	コマンド
2 番目のタブに移動する	Command-2	tab2
3 番目のタブに移動する	Command-3	tab3
4 番目のタブに移動する	Command-4	tab4
5 番目のタブに移動する	Command-5	tab5
6 番目のタブに移動する	Command-6	tab6
7 番目のタブに移動する	Command-7	tab7
8 番目のタブに移動する	Command-8	tab8
9 番目のタブに移動する	Command	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Command-E Command-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Command-. F1	gotocommand
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Command-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Command-Shift-0	gotosymbol

説明	割り当て	コマンド
[Outline (アウトライン)] ウィンドウを表示する	Command-Shift-E	outline
[Console (コンソール)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Control-Esc	toggleconsole
[Environment (環境)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Command-U	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行上にカーソルを追加する、または既にカーソルが追加されている場合はその上にもう 1 つカーソルを追加する	Control-Option-Up	addCursorAbove
アクティブなカーソルの 1 行上に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行上に移動する	Control-Option-Shift-Up	addCursorAboveSkipCurrent
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加され	Control-Option-Down	addCursorBelow

説明	割り当て	コマンド
ている場合はその下にもう 1 つカーソルを追加する		
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Control-Option-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Control-Option-A	alignCursors
バックスペースで 1 つのスペースを削除する	Control-Backspace Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Control-]	blockindent
選択部分を 1 タブ分アウトデントする	Control-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Command-Z Command-Shift-Z Command-Y	cancelBrowserUndoInAce
選択部分を中央揃えにする	Control-L	centerselection
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Command-Option-Down	copylinesdown

説明	割り当て	コマンド
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Command-Option-Up	copylinesup
1 つのスペースを削除する	Delete Control-D delete Shift-Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Command-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Command-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Control-Shift-M	expandToMatching
選択されたコードを折りたたむ、または折りたたまれたユニットが選択されている場合は展開する	Command-Option-L Command-F1	fold
折りたたむことができる要素をすべて折りたたむ	Control-Command-Option-0	foldall
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Command-Option-0	fold0ther
1 行下に移動する	Down Control-N	golinedown
1 行上に移動する	Up Control-P	golineup
ファイルの末尾に移動する	Command-End Command-Down	gotoend
1 つ左のスペースに移動する	Left Control-B	gotoleft

説明	割り当て	コマンド
現在の行の末尾に移動する	Command-Right End Control-E	gotolineend
現在の行の行頭に移動する	Command-Left Home Control-A	gotolinestart
次のエラーに移動する	F4	goToNextError
1 ページ下に移動する	Page Down Control-V	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Shift-F4	goToPreviousError
1 つ右のスペースに移動する	Right Control-F	gotoright
ファイルの最初に移動する	Command-Home Command-Up	gotostart
1 つ左の単語に移動する	Option-Left	gotowordleft
1 つ右の単語に移動する	Option-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Control-P	jumptomatching
フォントサイズを大きくする	Command-+ Command-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Option-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Option-Shift-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Option-Down	movelinesdown

説明	割り当て	コマンド
選択範囲を 1 行上に移動する	Option-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする (オフの場合)	Insert	overwrite
1 ページ下に移動する	Option-Page Down	pagedown
1 ページ上に移動する	Option-Page Up	pageup
現在の行を削除する	Command-D	removeline
カーソルから現在の行の末尾までを削除する	Control-K	removetolineend
現在の行の行頭からカーソルまでを削除する	Command-Backspace	removetolinestart
カーソルの左にある単語を削除する	Option-Backspace Control-Option-Backspace	removewordleft
カーソルの右にある単語を削除する	Option-Delete	removewordright
以前に記録されたキーストロークを再生する	Command-Shift-E	replaymacro
選択可能な内容をすべて選択する	Command-A	selectall
下にある次の行を選択部分に含める	Shift-Down Control-Shift-N	selectdown
左にある次のスペースを選択部分に含める	Shift-Left Control-Shift-B	selectleft

説明	割り当て	コマンド
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Control-Option-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Control-Option-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Control-Option-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Control-Option-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Control-G	selectOrFindNext
前に一致する選択部分を選択または検索する	Control-Shift-G	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Command-Shift-End Command-Shift-Down	selecttoend

説明	割り当て	コマンド
カーソルから現在の行の末尾までを選択に含める	Command-Shift-Right Shift-End Control-Shift-E	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Command-Shift-Left Control-Shift-A	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Control-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Command-Shift-Home Command-Shift-Up	selecttostart
上にある次の行を選択部分に含める	Shift-Up Control-Shift-Up	selectup
カーソル左にある次の単語までを選択に含める	Option-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Option-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Command-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Command--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Command-Option-S	sortlines
現在の行の末尾にカーソルを追加する	Control-Option-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Control-O	splitline

説明	割り当て	コマンド
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Command-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Command-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Option-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Command-Option-E	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Control-W	toggleWordWrap
選択部分をすべて小文字に変更する	Control-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Control-U	touppercase
選択部分を移動する	Control-T	transposeletters
選択されたコードを展開する	Command-Option-Shift-L Command-Shift-F1	unfold

説明	割り当て	コマンド
ファイル全体でコードの折りたたみを展開する	Command-Option-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Command-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Control-Option-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Command-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Command-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Control-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Option-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Option-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Command-B	build
現在一時停止しているプロセスを再開する	F8 Command-\	resume
現在のアプリケーションを実行またはデバッグする	Option-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11 Command-;	stepinto
現在の関数範囲をステップアウトする	Shift-F11 Command-Shift-'	stepout
スタックの現在の式をステップオーバーする	F10 Command-'	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop

説明	割り当て	コマンド
現在のファイルのビルドを停止する	Control-Shift-C	stopbuild

IDE の MacOS Vim AWS Cloud9 キー割り当てリファレンス

このステップでは、MacOS Vim キー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Vim] を選択します。
4. [Operating system (オペレーティングシステム)] で、[MacOS] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、AWS Cloud9 IDE における MacOS オペレーティングシステム用の Vim キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Command-Shift-C	addwatchfromselection

説明	割り当て	コマンド
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Control-Space Option-Space	complete
コードを補完した後、上書きする	Control-Shift-Space Option-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Command-C	copy
選択したものをクリップボードに切り取る	Command-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Command-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Control-Option-G	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Command-G	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Command-Shift-G	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences

説明	割り当て	コマンド
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Command-Shift-B	formatcode
行に移動するボックスを表示する	Command-L	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Command-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Control-N	newfile
[Preferences (設定)] タブを表示する	Command-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Command-Option-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Command-V	paste

説明	割り当て	コマンド
エラーを修正するための候補を表示する	Command-F3	quickfix
前回のアクションを繰り返す	Command-Shift-Z Command-Y	redo
プレビューペインを最新の情報に更新する	Command-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Option-Command-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Option-Command-F	replace
初期化スクリプトを再実行する	Command-Enter	rerunInitScript
環境の再起動	Command-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Control-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Option-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Command-S	save
現在のファイルを別のファイル名でディスクに保存する	Command-Shift-S	saveas

説明	割り当て	コマンド
複数のファイルで検索と置換バーを表示する	Shift-Command-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Command-Option-P	showprocesslist
前回のアクションを元に戻す	Command-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Option-Control-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Option-Shift-W	closealltabs
現在のペインを閉じる	Command-Control-W	closepane
現在のタブを閉じる	Option-W	closetab
下のペインに移動	Control-Command-Down	gotopanedown
左のペインに移動	Control-Command-Left	gotopaneleft
右のペインに移動	Control-Command-Right	gotopaneright
上のペインに移動	Control-Command-Up	gottopaneup
左のタブに移動	Command-[	gototableft
右のタブに移動	Command-]	gototabright

説明	割り当て	コマンド
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Command-Option-Shift-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Command-Option-Shift-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Command-Option-Shift-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Command-Option-Shift-Up	movetabup
次のペインに移動する	Option-Esc	nextpane
次のタブに移動する	Option-Tab	nexttab
前のペインに移動する	Option-Shift-Esc	previouspane
前のタブに移動する	Option-Shift-Tab	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Option-Shift-T	reopenLastTab
ファイルツリーで現在のタブを表示する	Command-Shift-L	revealtab
10 番目のタブに移動する	Command-0	tab0
最初のタブに移動する	Command-1	tab1

説明	割り当て	コマンド
2 番目のタブに移動する	Command-2	tab2
3 番目のタブに移動する	Command-3	tab3
4 番目のタブに移動する	Command-4	tab4
5 番目のタブに移動する	Command-5	tab5
6 番目のタブに移動する	Command-6	tab6
7 番目のタブに移動する	Command-7	tab7
8 番目のタブに移動する	Command-8	tab8
9 番目のタブに移動する	Command	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Command-E Command-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Command-. F1	gotocommand
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Command-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Command-Shift-0	gotosymbol

説明	割り当て	コマンド
[Outline (アウトライン)] ウィンドウを表示する	Command-Shift-E	outline
[Console (コンソール)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Control-Esc	toggleconsole
[Environment (環境)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Command-U	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行上にカーソルを追加する、または既にカーソルが追加されている場合はその上にもう 1 つカーソルを追加する	Control-Option-Up	addCursorAbove
アクティブなカーソルの 1 行上に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行上に移動する	Control-Option-Shift-Up	addCursorAboveSkipCurrent
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加され	Control-Option-Down	addCursorBelow

説明	割り当て	コマンド
ている場合はその下にもう 1 つカーソルを追加する		
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Control-Option-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Control-Option-A	alignCursors
バックスペースで 1 つのスペースを削除する	Control-Backspace Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Control-]	blockindent
選択範囲を 1 行上に移動する	Control-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Command-Z Command-Shift-Z Command-Y	cancelBrowserUndoInAce
選択部分を中央揃えにする	Control-L	centerselection
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Command-Option-Down	copylinesdown

説明	割り当て	コマンド
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Command-Option-Up	copylinesup
1 つのスペースを削除する	Delete Control-D delete Shift-Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Command-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Command-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Control-Shift-M	expandToMatching
選択されたコードを折りたたむ、または折りたたまれたユニットが選択されている場合は展開する	Command-Option-L Command-F1	fold
折りたたむことができる要素をすべて折りたたむ	Control-Command-Option-0	foldall
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Command-Option-0	fold0ther
1 行下に移動する	Down Control-N	golinedown
1 行上に移動する	Up Control-P	golineup
ファイルの末尾に移動する	Command-End Command-Down	gotoend
1 つ左のスペースに移動する	Left Control-B	gotoleft

説明	割り当て	コマンド
現在の行の末尾に移動する	Command-Right End Control-E	gotolineend
現在の行の行頭に移動する	Command-Left Home Control-A	gotolinestart
次のエラーに移動する	F4	goToNextError
1 ページ下に移動する	Page Down Control-V	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Shift-F4	goToPreviousError
1 つ右のスペースに移動する	Right Control-F	gotoright
ファイルの最初に移動する	Command-Home Command-Up	gotostart
1 つ左の単語に移動する	Option-Left	gotowordleft
1 つ右の単語に移動する	Option-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Control-P	jumptomatching
フォントサイズを大きくする	Command-+ Command-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Option-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Option-Shift-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Option-Down	movelinesdown

説明	割り当て	コマンド
選択範囲を 1 行上に移動する	Option-Up	movelinesup
選択範囲を 1 行上に移動する	Shift-Tab	outdent
上書きモードをオンにする (オフの場合)	Insert	overwrite
1 ページ下に移動する	Option-Page Down	pagedown
1 ページ上に移動する	Option-Page Up	pageup
現在の行を削除する	Command-D	removeline
カーソルから現在の行の末尾までを削除する	Control-K	removetolineend
現在の行の行頭からカーソルまでを削除する	Command-Backspace	removetolinestart
カーソルの左にある単語を削除する	Option-Backspace Control-Option-Backspace	removewordleft
カーソルの右にある単語を削除する	Option-Delete	removewordright
以前に記録されたキーストロークを再生する	Command-Shift-E	replaymacro
選択可能な内容をすべて選択する	Command-A	selectall
下にある次の行を選択部分に含める	Shift-Down Control-Shift-N	selectdown
左にある次のスペースを選択部分に含める	Shift-Left Control-Shift-B	selectleft

説明	割り当て	コマンド
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Control-Option-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Control-Option-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Control-Option-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Control-Option-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Control-G	selectOrFindNext
前に一致する選択部分を選択または検索する	Control-Shift-G	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Command-Shift-End Command-Shift-Down	selecttoend

説明	割り当て	コマンド
カーソルから現在の行の末尾までを選択に含める	Command-Shift-Right Shift-End Control-Shift-E	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Command-Shift-Left Control-Shift-A	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Control-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Command-Shift-Home Command-Shift-Up	selecttostart
上にある次の行を選択部分に含める	Shift-Up Control-Shift-P	selectup
カーソル左にある次の単語までを選択に含める	Option-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Option-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Command-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Command--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Command-Option-S	sortlines
現在の行の末尾にカーソルを追加する	Control-Option-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Control-O	splitline

説明	割り当て	コマンド
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Command-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Command-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Option-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Command-Option-E	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Control-W	toggleWordWrap
選択部分をすべて小文字に変更する	Control-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Control-U	touppercase
選択部分を移動する	Control-T	transposeletters
選択されたコードを展開する	Command-Option-Shift-L Command-Shift-F1	unfold

説明	割り当て	コマンド
ファイル全体でコードの折りたたみを展開する	Command-Option-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Command-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Control-Option-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Command-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Command-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Control-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Option-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Option-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Command-B	build
現在一時停止しているプロセスを再開する	F8 Command-\	resume
現在のアプリケーションを実行またはデバッグする	Option-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11 Command-;	stepinto
現在の関数範囲をステップアウトする	Shift-F11 Command-Shift-'	stepout
スタックの現在の式をステップオーバーする	F10 Command-'	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop

説明	割り当て	コマンド
現在のファイルのビルドを停止する	Control-Shift-C	stopbuild

IDE の MacOS Emacs AWS Cloud9 キー割り当てリファレンス

このステップでは、MacOS Emacs キー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Emacs] を選択します。
4. [Operating system (オペレーティングシステム)] で、[MacOS] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、IDE の MacOS オペレーティングシステム用の Emacs AWS Cloud9 キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Command-Shift-C	addwatchfromselection

説明	割り当て	コマンド
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Control-Space Option-Space	complete
コードを補完して、上書きする	Control-Shift-Space Option-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Command-C	copy
選択したものをクリップボードに切り取る	Command-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Command-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Control-Option-G	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Command-G	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Command-Shift-G	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences

説明	割り当て	コマンド
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Command-Shift-B	formatcode
行に移動するボックスを表示する	Command-L	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Command-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Control-N	newfile
[環境設定] タブを表示する	Command-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Command-Option-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Command-V	paste

説明	割り当て	コマンド
エラーを修正するための候補を表示する	Command-F3	quickfix
前回のアクションを繰り返す	Command-Shift-Z Command-Y	redo
プレビューペインを最新の情報に更新する	Command-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Option-Command-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Option-Command-F	replace
初期化スクリプトを再実行する	Command-Enter	rerunInitScript
環境の再起動	Command-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Control-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Option-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Command-S	save
現在のファイルを別のファイル名でディスクに保存する	Command-Shift-S	saveas

説明	割り当て	コマンド
複数のファイルで検索と置換バーを表示する	Shift-Command-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Command-Option-P	showprocesslist
前回のアクションを元に戻す	Command-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Option-Control-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Option-Shift-W	closealltabs
現在のペインを閉じる	Command-Control-W	closepane
現在のタブを閉じる	Option-W	closetab
下のペインに移動	Control-Command-Down	gotopanedown
左のペインに移動	Control-Command-Left	gotopaneleft
右のペインに移動	Control-Command-Right	gotopaneright
上のペインに移動	Control-Command-Up	gottopaneup
左のタブに移動	Command-[	gototableft
右のタブに移動	Command-]	gototabright

説明	割り当て	コマンド
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Command-Option-Shift-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Command-Option-Shift-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Command-Option-Shift-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Command-Option-Shift-Up	movetabup
次のペインに移動する	Option-Esc	nextpane
次のタブに移動する	Option-Tab	nexttab
前のペインに移動する	Option-Shift-Esc	previouspane
前のタブに移動する	Option-Shift-Tab	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Option-Shift-T	reopenLastTab
ファイルツリーで現在のタブを表示する	Command-Shift-L	revealtab
10 番目のタブに移動する	Command-0	tab0
最初のタブに移動する	Command-1	tab1

説明	割り当て	コマンド
2 番目のタブに移動する	Command-2	tab2
3 番目のタブに移動する	Command-3	tab3
4 番目のタブに移動する	Command-4	tab4
5 番目のタブに移動する	Command-5	tab5
6 番目のタブに移動する	Command-6	tab6
7 番目のタブに移動する	Command-7	tab7
8 番目のタブに移動する	Command-8	tab8
9 番目のタブに移動する	Command	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Command-E Command-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Command-. F1	gotocommand
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Command-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Command-Shift-0	gotosymbol

説明	割り当て	コマンド
[Outline (アウトライン)] ウィンドウを表示する	Command-Shift-E	outline
[Console (コンソール)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Control-Esc	toggleconsole
[Environment (環境)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	Command-U	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行上にカーソルを追加する、または既にカーソルが追加されている場合はその上にもう 1 つカーソルを追加する	Control-Option-Up	addCursorAbove
アクティブなカーソルの 1 行上に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行上に移動する	Control-Option-Shift-Up	addCursorAboveSkipCurrent
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加され	Control-Option-Down	addCursorBelow

説明	割り当て	コマンド
ている場合はその下にもう 1 つカーソルを追加する		
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Control-Option-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Control-Option-A	alignCursors
バックスペースで 1 つのスペースを削除する	Control-Backspace Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Control-]	blockindent
選択範囲を 1 行上に移動する	Control-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Command-Z Command-Shift-Z Command-Y	cancelBrowserUndoInAce
選択部分を中央揃えにする	Control-L	centerselection
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Command-Option-Down	copylinesdown

説明	割り当て	コマンド
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Command-Option-Up	copylinesup
1 つのスペースを削除する	Delete Control-D delete Shift-Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Command-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Command-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Control-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Command-Option-L Command-F1	fold
折りたたむことができる要素をすべて折りたたむ	Control-Command-Option-0	foldall
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Command-Option-0	fold0ther
1 行下に移動する	Down Control-N	golinedown
1 行上に移動する	Up Control-P	golineup
ファイルの末尾に移動する	Command-End Command-Down	gotoend
1 つ左のスペースに移動する	Left Control-B	gotoleft

説明	割り当て	コマンド
現在の行の末尾に移動する	Command-Right End Control-E	gotolineend
現在の行の行頭に移動する	Command-Left Home Control-A	gotolinestart
次のエラーに移動する	F4	goToNextError
1 ページ下に移動する	Page Down Control-V	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Shift-F4	goToPreviousError
1 つ右のスペースに移動する	Right Control-F	gotoright
ファイルの最初に移動する	Command-Home Command-Up	gotostart
1 つ左の単語に移動する	Option-Left	gotowordleft
1 つ右の単語に移動する	Option-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Control-P	jumptomatching
フォントサイズを大きくする	Command-+ Command-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Option-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Option-Shift-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Option-Down	movelinesdown

説明	割り当て	コマンド
選択範囲を 1 行上に移動する	Option-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
1 ページ下に移動する	Option-Page Down	pagedown
1 ページ上に移動する	Option-Page Up	pageup
現在の行を削除する	Command-D	removeline
カーソルから現在の行の末尾 までを削除する	Control-K	removetolineend
現在の行の行頭からカーソル までを削除する	Command-Backspace	removetolinestart
カーソルの左にある単語を削除する	Option-Backspace Control-Option-Backspace	removewordleft
カーソルの右にある単語を削除する	Option-Delete	removewordright
以前に記録されたキーストロークを再生する	Command-Shift-E	replaymacro
選択可能な内容をすべて選択する	Command-A	selectall
下にある次の行を選択部分に含める	Shift-Down Control-Shift-N	selectdown
左にある次のスペースを選択部分に含める	Shift-Left Control-Shift-B	selectleft

説明	割り当て	コマンド
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Control-Option-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Control-Option-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Control-Option-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Control-Option-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Control-G	selectOrFindNext
前に一致する選択部分を選択または検索する	Control-Shift-G	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Command-Shift-End Command-Shift-Down	selecttoend

説明	割り当て	コマンド
カーソルから現在の行の末尾までを選択に含める	Command-Shift-Right Shift-End Control-Shift-E	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Command-Shift-Left Control-Shift-A	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Control-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Command-Shift-Home Command-Shift-Up	selecttostart
上にある次の行を選択部分に含める	Shift-Up Control-Shift-Up	selectup
カーソル左にある次の単語までを選択に含める	Option-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Option-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Command-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Command--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Command-Option-S	sortlines
現在の行の末尾にカーソルを追加する	Control-Option-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Control-O	splitline

説明	割り当て	コマンド
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Command-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Command-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Option-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Command-Option-E	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Control-W	toggleWordWrap
選択部分をすべて小文字に変更する	Control-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Control-U	touppercase
選択部分を移動する	Control-T	transposeletters
選択されたコードを展開する	Command-Option-Shift-L Command-Shift-F1	unfold

説明	割り当て	コマンド
ファイル全体でコードの折りたたみを展開する	Command-Option-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Command-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Control-Option-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Command-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Command-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Control-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Option-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Option-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Command-B	build
現在一時停止しているプロセスを再開する	F8 Command-\	resume
現在のアプリケーションを実行またはデバッグする	Option-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11 Command-;	stepinto
現在の関数範囲をステップアウトする	Shift-F11 Command-Shift-'	stepout
スタックの現在の式をステップオーバーする	F10 Command-'	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop

説明	割り当て	コマンド
現在のファイルのビルドを停止する	Control-Shift-C	stopbuild

IDE の MacOS Sublime AWS Cloud9 キー割り当てリファレンス

このステップでは、MacOS Sublime キー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Sublime] を選択します。
4. [Operating system (オペレーティングシステム)] で、[MacOS] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、AWS Cloud9 IDE の MacOS オペレーティングシステム用の Sublime キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Command-Shift-C	addwatchfromselection

説明	割り当て	コマンド
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Control-Space Option-Space	complete
コードを補完した後、上書きする	Control-Shift-Space Option-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Command-C	copy
選択したものをクリップボードに切り取る	Command-X	cut
カーソルから行の先頭までを削除する	Command-K Command-Backspace Command-Backspace	delete_to_hard_bol
カーソルから行の末尾までを削除する	Command-K Command-K Command-Delete Control-K	delete_to_hard_eol
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Command-F	find
選択範囲のすべての一致をハイライトする	Control-Command-G	find_all_under
選択範囲の次の一致をハイライトする	Option-Command-G	find_under

説明	割り当て	コマンド
ハイライト部分のカーソルとすべての一致をハイライトする	Command-D	find_under_expand
カーソルをハイライトし、ハイライト部分のすべての一致の概略を示す	Command-K Command-D	find_under_expand_skip
選択範囲の前の一致をハイライトする	Shift-Option-Command-G	find_under_previous
現在のドキュメント内で検索に一致するものをすべて選択する	Control-Option-G	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Command-G	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Shift-Command-G	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Control-Option-F	formatcode
行に移動するボックスを表示する	Control-G	gotoline

説明	割り当て	コマンド
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F12 Command-Option-Down	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Command-Shift-U	lambdaUploadFunction
現在の単語の末尾に移動する	Option-Right	moveToWordEndRight
現在の単語の先頭に移動する	Option-Left	moveToWordStartLeft
新規ファイルを作成する	Control-N	newfile
[Preferences (設定)] タブを表示する	Command-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Command-Option-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Command-V	paste
エラーを修正するための候補を表示する	Command-F3	quickfix
前回のアクションを繰り返す	Command-Shift-Z Command-Y	redo

説明	割り当て	コマンド
プレビューペインを最新の情報に更新する	Command-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Option-Command-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Command-Option-F	replace
検索式のすべての一致を検索と置換バーの式で置き換える	Control-Option-Enter	replaceall
検索式の次の一致を検索と置換バーの式で置き換える	Command-Option-E	replacenext
初期化スクリプトを再実行する	Command-Enter	rerunInitScript
環境の再起動	Command-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Control-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Option-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Command-S	save
現在のファイルを別のファイル名でディスクに保存する	Command-Shift-S	saveas

説明	割り当て	コマンド
複数のファイルで検索と置換バーを表示する	Command-Shift-F	searchinfiles
カーソルから単語の末尾までを選択に含める	Option-Shift-Right	selectToWordEndRight
カーソルから単語の先頭までを選択に含める	Option-Shift-Left	selectToWordStartLeft
[Process List (プロセスリスト)] ダイアログボックスを表示する	Command-Option-P	showprocesslist
前回のアクションを元に戻す	Command-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Option-Control-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Option-Shift-W	closealltabs
現在のペインを閉じる	Command-Control-W	closepane
現在のタブを閉じる	Option-W	closetab
下のペインに移動	Control-Command-Down	gotopanedown
左のペインに移動	Control-Command-Left	gotopaneleft
右のペインに移動	Control-Command-Right	gotopaneright

説明	割り当て	コマンド
上のペインに移動	Control-Command-Up	gottopaneup
左のタブに移動	Command-Shift-[Command-Option-Left	gototableft
右のタブに移動	Command-Shift-] Command-Option-Right	gototabright
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Command-Option-Shift-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Command-Option-Shift-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Command-Option-Shift-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Command-Option-Shift-Up	movetabup
次のタブに移動する	Control-Tab	nexttab
前のペインに移動する	Option-Shift-Esc	previouspane
前のタブに移動する	Control-Shift-Tab	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Command-Shift-T	reopenLastTab

説明	割り当て	コマンド
ファイルツリーで現在のタブを表示する	Command-E	revealtab
10 番目のタブに移動する	Command-0	tab0
最初のタブに移動する	Command-1	tab1
2 番目のタブに移動する	Command-2	tab2
3 番目のタブに移動する	Command-3	tab3
4 番目のタブに移動する	Command-4	tab4
5 番目のタブに移動する	Command-5	tab5
6 番目のタブに移動する	Command-6	tab6
7 番目のタブに移動する	Command-7	tab7
8 番目のタブに移動する	Command-8	tab8
9 番目のタブに移動する	Command	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Command-E Command-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Command-. F1	gotocommand

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Command-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Command-Shift-0	gotosymbol
[Outline (アウトライン)] ウィンドウを表示する	Command-Shift-R	outline
[Console (コンソール)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Control-`	toggleconsole
[Environment (環境)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Command-K Command-B	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行 上にカーソルを追加する、ま たは既にカーソルが追加され ている場合はその上にもう 1 つカーソルを追加する	Control-Shift-Up	addCursorAbove
アクティブなカーソルの 1 行 上に 2 つ目のカーソルを追 加する、または既に 2 つ目の カーソルが追加されている場	Control-Option-Shi ft-Up	addCursorAboveSkip Current

説明	割り当て	コマンド
合は 2 つ目のカーソルを 1 行上に移動する		
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加されている場合はその下にもう 1 つカーソルを追加する	Control-Shift-Down	addCursorBelow
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Control-Option-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Control-Option-A	alignCursors
バックスペースで 1 つのスペースを削除する	Control-Backspace Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Control-]	blockindent
選択部分を 1 タブ分アウトデントする	Control-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Command-Z Command-Shift-Z Command-S Command-Y	cancelBrowserUndoInAce

説明	割り当て	コマンド
選択部分を中央揃えにする	Command-K Command-C Control-L	centerselection
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Command-Option-Down	copylinesdown
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Command-Option-Up	copylinesup
1 つのスペースを削除する	Delete Control-D delete Shift-Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Command-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Command-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Control-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Command-Option-L Command-F1	fold
折りたたむことができる要素をすべて折りたたむ	Control-Command-Option-0	foldall
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Command-K Command-1	fold0ther
1 行下に移動する	Down Control-N	golinedown

説明	割り当て	コマンド
1 行上に移動する	Up Control-P	golineup
ファイルの末尾に移動する	Command-End Command-Down	gotoend
1 つ左のスペースに移動する	Left Control-B	gotoleft
現在の行の末尾に移動する	Command-Right End Control-E	gotolineend
現在の行の行頭に移動する	Command-Left Home Control-A	gotolinestart
次のエラーに移動する	Control-F6	goToNextError
1 ページ下に移動する	Page Down Control-V	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Control-Shift-F6	goToPreviousError
1 つ右のスペースに移動する	Right Control-F	gotoright
ファイルの最初に移動する	Command-Home Command-Up	gotostart
1 つ左の単語に移動する	Option-Left	gotowordleft
1 つ右の単語に移動する	Option-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
選択した行を 1 行に結合する	Command-J	joinlines
現在の範囲内の一致する記号に移動する	Control-M	jumptomatching
フォントサイズを大きくする	Command-= Command-+	largerfont

説明	割り当て	コマンド
カーソルの左にある数字を 1 減らす (数字の場合)	Option-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Option-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Control-Command-Down	movelinesdown
選択範囲を 1 行上に移動する	Control-Command-Up	movelinesup
選択範囲を 1 行上に移動する	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
1 ページ下に移動する	Option-Page Down	pagedown
1 ページ上に移動する	Option-Page Up	pageup
現在の行の内容を削除する	Control-Shift-K	removeline
カーソルから現在の行の末尾 までを削除する	Control-K	removetolineend
現在の行の行頭からカーソル までを削除する	Command-Backspace	removetolinestart
カーソルの左にある単語を削 除する	Option-Backspace Control-Option-Bac kspace	removewordleft
カーソルの右にある単語を削 除する	Option-Delete	removewordright
以前に記録されたキースト ロークを再生する	Control-Shift-Q	replaymacro

説明	割り当て	コマンド
選択可能な内容をすべて選択する	Command-A	selectall
下にある次の行を選択部分に含める	Shift-Down Control-Shift-N	selectdown
左にある次のスペースを選択部分に含める	Shift-Left Control-Shift-B	selectleft
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Control-Option-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Control-Option-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Control-Option-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Control-Option-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Control-G	selectOrFindNext
前に一致する選択部分を選択または検索する	Control-Shift-G	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup

説明	割り当て	コマンド
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Command-Shift-End Command-Shift-Down	selecttoend
カーソルから現在の行の末尾までを選択に含める	Command-Shift-Right Shift-End Control-Shift-E	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Command-Shift-Left Control-Shift-A	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Control-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Command-Shift-Home Command-Shift-Up	selecttostart
上にある次の行を選択部分に含める	Shift-Up Control-Shift-P	selectup
カーソル左にある次の単語までを選択に含める	Option-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Option-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Command-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Command--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	F5	sortlines

説明	割り当て	コマンド
現在の行の末尾にカーソルを追加する	Command-Shift-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Control-O	splitline
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Command-Option-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Command-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	Command-Option-[	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Option-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Control-Q	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Control-W	toggleWordWrap
選択部分をすべて小文字に変更する	Command-K Command-L	tolowercase
選択部分をすべて大文字に変更する	Command-K Command-U	touppercase
選択部分を移動する	Control-T	transposeletters

説明	割り当て	コマンド
選択されたコードを展開する	Command-Option-]	unfold
ファイル全体でコードの折りたたみを展開する	Command-K Command-0 Command-K Command-J	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Command-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Control-Option-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Command-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Command-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Control-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Option-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Option-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	F7 Command-B	build
現在一時停止しているプロセスを再開する	F8 Command-\	resume
現在のアプリケーションを実行またはデバッグする	Command-Shift-B	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11 Command-;	stepinto
現在の関数範囲をステップアウトする	Shift-F11 Command-Shift-'	stepout
スタックの現在の式をステップオーバーする	F10 Command-'	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop

説明	割り当て	コマンド
現在のファイルのビルドを停止する	Control-Break	stopbuild

IDE の Windows/Linux AWS Cloud9 デフォルトキー割り当てリファレンス

このステップでは、AWS Cloud9の Windows/Linux デフォルトキー割り当てリファレンスを選択する方法を示します。

1. メニューバーで、AWS Cloud9、[Preferences (設定)] の順に選択します。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Default (デフォルト)] を選択します。
4. [Operating System (オペレーティングシステム)] で、[Windows / Linux] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、AWS Cloud9 統合開発環境 (IDE) の Windows/Linux オペレーティングシステムのデフォルトのキーボードモードキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Ctrl-Shift-C	addwatchfromselection
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Ctrl-Space Alt-Space	complete
コードを補完した後、上書きする	Ctrl-Shift-Space Alt-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Ctrl-C	copy
選択したものをクリップボードに切り取る	Ctrl-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Ctrl-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Ctrl-Alt-K	findall
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Ctrl-K	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Ctrl-Shift-K	findprevious

説明	割り当て	コマンド
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Ctrl-Shift-B	formatcode
行に移動するボックスを表示する	Ctrl-G	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Ctrl-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Alt-N	newfile
[Preferences (設定)] タブを表示する	Ctrl-,	openpreferences

説明	割り当て	コマンド
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Alt-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Ctrl-V	paste
エラーを修正するための候補を表示する	Ctrl-F3	quickfix
前回のアクションを繰り返す	Ctrl-Shift-Z Ctrl-Y	redo
プレビューペインを最新の情報に更新する	Ctrl-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Ctrl-Alt-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Alt-Shift-F Ctrl-H	replace
初期化スクリプトを再実行する	Ctrl-Enter	rerunInitScript
環境の再起動	Ctrl-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Ctrl-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Alt-Shift-Q	reverttosavedall

説明	割り当て	コマンド
現在のファイルをディスクに保存する	Ctrl-S	save
現在のファイルを別のファイル名でディスクに保存する	Ctrl-Shift-S	saveas
複数のファイルで検索と置換バーを表示する	Ctrl-Shift-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Ctrl-Alt-P	showprocesslist
前回のアクションを元に戻す	Ctrl-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Ctrl-Alt-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Alt-Shift-W	closealltabs
現在のペインを閉じる	Ctrl-W	closepane
現在のタブを閉じる	Alt-W	closetab
下のペインに移動	Ctrl-Meta-Down	gotopanedown
左のペインに移動	Ctrl-Meta-Left	gotopaneleft
右のペインに移動	Ctrl-Meta-Right	gotopaneright
上のペインに移動	Ctrl-Meta-Up	gottopaneup

説明	割り当て	コマンド
左のタブに移動	Ctrl-[	gototableft
右のタブに移動	Ctrl-]	gototabright
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Ctrl-Meta-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Ctrl-Meta-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Ctrl-Meta-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Ctrl-Meta-Up	movetabup
次のペインに移動する	Ctrl-`	nextpane
次のタブに移動する	Ctrl-Tab Alt-`	nexttab
前のペインに移動する	Ctrl-Shift-`	previouspane
前のタブに移動する	Ctrl-Shift-Tab Alt-Shift-`	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Alt-Shift-T	reopenLastTab

説明	割り当て	コマンド
ファイルツリーで現在のタブを表示する	Ctrl-Shift-L	revealtab
10 番目のタブに移動する	Ctrl-0	tab0
最初のタブに移動する	Ctrl-1	tab1
2 番目のタブに移動する	Ctrl-2	tab2
3 番目のタブに移動する	Ctrl-3	tab3
4 番目のタブに移動する	Ctrl-4	tab4
5 番目のタブに移動する	Ctrl-5	tab5
6 番目のタブに移動する	Ctrl-6	tab6
7 番目のタブに移動する	Ctrl-7	tab7
8 番目のタブに移動する	Ctrl-8	tab8
9 番目のタブに移動する	Ctrl-9	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Ctrl-E Ctrl-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Ctrl-. F1	gotocommand

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Ctrl-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移 動)] モードで表示する	Ctrl-Shift-0	gotosymbol
[Outline (アウトライン)] ウィンドウを表示する	Ctrl-Shift-E	outline
[Console (コンソール)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	F6	toggleconsole
[Environment (環境)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Ctrl-I	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行 上にカーソルを追加する、ま たは既にカーソルが追加され ている場合はその上にもう 1 つカーソルを追加する	Ctrl-Alt-Up	addCursorAbove
アクティブなカーソルの 1 行 上に 2 つ目のカーソルを追 加する、または既に 2 つ目の カーソルが追加されている場	Ctrl-Alt-Shift-Up	addCursorAboveSkip Current

説明	割り当て	コマンド
合は 2 つ目のカーソルを 1 行上に移動する		
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加されている場合はその下にもう 1 つカーソルを追加する	Ctrl-Alt-Down	addCursorBelow
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Ctrl-Alt-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Ctrl-Alt-A	alignCursors
バックスペースで 1 つのスペースを削除する	Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Ctrl-]	blockindent
選択部分を 1 タブ分アウトデントする	Ctrl-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Ctrl-Z Ctrl-Shift-Z Ctrl-Y	cancelBrowserUndoInAce
選択部分を中央揃えにする	Ctrl-L	centerselection

説明	割り当て	コマンド
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Alt-Shift-Down	copylinesdown
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Alt-Shift-Up	copylinesup
選択を切り取る。何も選択していない場合は、スペースを 1 つ削除する	Shift-Delete	cut_or_delete
1 つのスペースを削除する	Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Ctrl-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Ctrl-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Ctrl-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Alt-L Ctrl-F1	fold
折りたたむことができる要素をすべて折りたたむ	Ctrl-Command-Option-0	foldall
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Alt-0	fold0ther
1 行下に移動する	Down	golinedown

説明	割り当て	コマンド
1 行上に移動する	Up	golineup
ファイルの末尾に移動する	Ctrl-End	gotoend
1 つ左のスペースに移動する	Left	gotoleft
現在の行の末尾に移動する	Alt-Right End	gotolineend
現在の行の行頭に移動する	Alt-Left Home	gotolinestart
次のエラーに移動する	Alt-E	goToNextError
1 ページ下に移動する	Page Down	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Alt-Shift-E	goToPreviousError
1 つ右のスペースに移動する	Right	gotoright
ファイルの最初に移動する	Ctrl-Home	gotostart
1 つ左の単語に移動する	Ctrl-Left	gotowordleft
1 つ右の単語に移動する	Ctrl-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Ctrl-P	jumptomatching
フォントサイズを大きくする	Ctrl-+ Ctrl-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Ctrl-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Ctrl-Shift-Up	modifyNumberUp

説明	割り当て	コマンド
選択範囲を 1 行下に移動する	Alt-Down	movelinesdown
選択範囲を 1 行上に移動する	Alt-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
1 ページ下に移動する	Option-Page Down	pagedown
1 ページ上に移動する	Option-Page Up	pageup
現在の行の内容を削除する	Ctrl-D	removeline
カーソルから現在の行の末尾 までを削除する	Alt-Delete	removetolineend
現在の行の行頭からカーソル までを削除する	Alt-Backspace	removetolinestart
カーソルの左にある単語を削除する	Ctrl-Backspace	removewordleft
カーソルの右にある単語を削除する	Ctrl-Delete	removewordright
以前に記録されたキーストロークを再生する	Ctrl-Shift-E	replaymacro
現在のファイルの 1 行下にスクロールする	Ctrl-Down	scrolldown
現在のファイルの 1 行上にスクロールする	Ctrl-Up	scrollup

説明	割り当て	コマンド
選択可能な内容をすべて選択する	Ctrl-A	selectall
下にある次の行を選択部分に含める	Shift-Down	selectdown
左にある次のスペースを選択部分に含める	Shift-Left	selectleft
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Ctrl-Alt-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Ctrl-Alt-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Ctrl-Alt-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Ctrl-Alt-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Alt-K	selectOrFindNext
前に一致する選択部分を選択または検索する	Alt-Shift-K	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup

説明	割り当て	コマンド
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Ctrl-Shift-End	selecttoend
カーソルから現在の行の末尾までを選択に含める	Alt-Shift-Right	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Alt-Shift-Left	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Ctrl-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Ctrl-Shift-Home	selecttostart
上にある次の行を選択部分に含める	Shift-Up	selectup
カーソル左にある次の単語までを選択に含める	Ctrl-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Ctrl-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Ctrl-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Ctrl--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Ctrl-Alt-S	sortlines

説明	割り当て	コマンド
現在の行の末尾にカーソルを追加する	Ctrl-Alt-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Ctrl-O	splitline
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Ctrl-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Ctrl-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Alt-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Ctrl-Alt-E	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Ctrl-Q	toggleWordWrap
選択部分をすべて小文字に変更する	Ctrl-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Ctrl-U	touppercase
選択部分を移動する	Alt-X	transposeletters

説明	割り当て	コマンド
選択されたコードを展開する	Alt-Shift-L Ctrl-Shift-F1	unfold
ファイル全体でコードの折りたたみを展開する	Alt-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Ctrl-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Ctrl-Alt-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Ctrl-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Ctrl-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Ctrl-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Alt-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Alt-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Ctrl-B	build
現在一時停止しているプロセスを再開する	F8	resume
現在のアプリケーションを実行またはデバッグする	Alt-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11	stepinto
現在の関数範囲をステップアウトする	Shift-F11	stepout
スタックの現在の式をステップオーバーする	F10	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop

説明	割り当て	コマンド
現在のファイルのビルドを停止する	Ctrl-Shift-C	stopbuild

IDE の Windows/Linux Vim AWS Cloud9 キー割り当てリファレンス

このステップでは、Windows/Linux Vim キー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Vim] を選択します。
4. [Operating System (オペレーティングシステム)] で、[Windows / Linux] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、IDE の Windows/Linux AWS Cloud9 オペレーティングシステム用の Vim キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Ctrl-Shift-C	addwatchfromselection
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Ctrl-Space Alt-Space	complete
コードを補完した後、上書きする	Ctrl-Shift-Space Alt-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Ctrl-C	copy
選択したものをクリップボードに切り取る	Ctrl-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Ctrl-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Ctrl-Alt-K	findall
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Ctrl-K	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Ctrl-Shift-K	findprevious

説明	割り当て	コマンド
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Ctrl-Shift-B	formatcode
行に移動するボックスを表示する	Ctrl-G	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Ctrl-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Alt-N	newfile
[Preferences (設定)] タブを表示する	Ctrl-,	openpreferences

説明	割り当て	コマンド
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Alt-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Ctrl-V	paste
エラーを修正するための候補を表示する	Ctrl-F3	quickfix
前回のアクションを繰り返す	Ctrl-Shift-Z Ctrl-Y	redo
プレビューペインを最新の情報に更新する	Ctrl-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Ctrl-Alt-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Alt-Shift-F Ctrl-H	replace
初期化スクリプトを再実行する	Ctrl-Enter	rerunInitScript
環境の再起動	Ctrl-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Ctrl-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Alt-Shift-Q	reverttosavedall

説明	割り当て	コマンド
現在のファイルをディスクに保存する	Ctrl-S	save
現在のファイルを別のファイル名でディスクに保存する	Ctrl-Shift-S	saveas
複数のファイルで検索と置換バーを表示する	Ctrl-Shift-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Ctrl-Alt-P	showprocesslist
前回のアクションを元に戻す	Ctrl-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Ctrl-Alt-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Alt-Shift-W	closealltabs
現在のペインを閉じる	Ctrl-W	closepane
現在のタブを閉じる	Alt-W	closetab
下のペインに移動	Ctrl-Meta-Down	gotopanedown
左のペインに移動	Ctrl-Meta-Left	gotopaneleft
右のペインに移動	Ctrl-Meta-Right	gotopaneright
上のペインに移動	Ctrl-Meta-Up	gottopaneup

説明	割り当て	コマンド
左のタブに移動	Ctrl-[	gototableft
右のタブに移動	Ctrl-]	gototabright
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Ctrl-Meta-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Ctrl-Meta-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Ctrl-Meta-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Ctrl-Meta-Up	movetabup
次のペインに移動する	Ctrl-`	nextpane
次のタブに移動する	Ctrl-Tab Alt-`	nexttab
前のペインに移動する	Ctrl-Shift-`	previouspane
前のタブに移動する	Ctrl-Shift-Tab Alt-Shift-`	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Alt-Shift-T	reopenLastTab

説明	割り当て	コマンド
ファイルツリーで現在のタブを表示する	Ctrl-Shift-L	revealtab
10 番目のタブに移動する	Ctrl-0	tab0
最初のタブに移動する	Ctrl-1	tab1
2 番目のタブに移動する	Ctrl-2	tab2
3 番目のタブに移動する	Ctrl-3	tab3
4 番目のタブに移動する	Ctrl-4	tab4
5 番目のタブに移動する	Ctrl-5	tab5
6 番目のタブに移動する	Ctrl-6	tab6
7 番目のタブに移動する	Ctrl-7	tab7
8 番目のタブに移動する	Ctrl-8	tab8
9 番目のタブに移動する	Ctrl-9	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Ctrl-E Ctrl-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Ctrl-. F1	gotocommand

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Ctrl-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Ctrl-Shift-0	gotosymbol
[Outline (アウトライン)] ウィンドウを表示する	Ctrl-Shift-E	outline
[Console (コンソール)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	F6	toggleconsole
[Environment (環境)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Ctrl-I	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行 上にカーソルを追加する、ま たは既にカーソルが追加され ている場合はその上にもう 1 つカーソルを追加する	Ctrl-Alt-Up	addCursorAbove
アクティブなカーソルの 1 行 上に 2 つ目のカーソルを追 加する、または既に 2 つ目の カーソルが追加されている場	Ctrl-Alt-Shift-Up	addCursorAboveSkip Current

説明	割り当て	コマンド
合は 2 つ目のカーソルを 1 行上に移動する		
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加されている場合はその下にもう 1 つカーソルを追加する	Ctrl-Alt-Down	addCursorBelow
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Ctrl-Alt-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Ctrl-Alt-A	alignCursors
バックスペースで 1 つのスペースを削除する	Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Ctrl-]	blockindent
選択部分を 1 タブ分アウトデントする	Ctrl-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Ctrl-Z Ctrl-Shift-Z Ctrl-Y	cancelBrowserUndoInAce

説明	割り当て	コマンド
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Alt-Shift-Down	copylinesdown
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Alt-Shift-Up	copylinesup
選択を切り取る。何も選択していない場合は、スペースを 1 つ削除する	Shift-Delete	cut_or_delete
1 つのスペースを削除する	Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Ctrl-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Ctrl-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Ctrl-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Alt-L Ctrl-F1	fold
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Alt-0	fold0ther
1 行下に移動する	Down	golinedown
1 行上に移動する	Up	golineup

説明	割り当て	コマンド
ファイルの末尾に移動する	Ctrl-End	gotoend
1 つ左のスペースに移動する	Left	gotoleft
現在の行の末尾に移動する	Alt-Right End	gotolineend
現在の行の行頭に移動する	Alt-Left Home	gotolinestart
次のエラーに移動する	Alt-E	goToNextError
1 ページ下に移動する	Page Down	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Alt-Shift-E	goToPreviousError
1 つ右のスペースに移動する	Right	gotoright
ファイルの最初に移動する	Ctrl-Home	gotostart
1 つ左の単語に移動する	Ctrl-Left	gotowordleft
1 つ右の単語に移動する	Ctrl-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Ctrl-P	jumptomatching
フォントサイズを大きくする	Ctrl-+ Ctrl-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Ctrl-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Ctrl-Shift-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Alt-Down	movelinesdown

説明	割り当て	コマンド
選択範囲を 1 行上に移動する	Alt-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
現在の行の内容を削除する	Ctrl-D	removeline
カーソルから現在の行の末尾 までを削除する	Alt-Delete	removetolineend
現在の行の行頭からカーソル までを削除する	Alt-Backspace	removetolinestart
カーソルの左にある単語を削除する	Ctrl-Backspace	removewordleft
カーソルの右にある単語を削除する	Ctrl-Delete	removewordright
以前に記録されたキースト ロークを再生する	Ctrl-Shift-E	replaymacro
現在のファイルの 1 行下にスクロールする	Ctrl-Down	scrolldown
現在のファイルの 1 行上にスクロールする	Ctrl-Up	scrollup
選択可能な内容をすべて選択する	Ctrl-A	selectall
下にある次の行を選択部分に 含める	Shift-Down	selectdown

説明	割り当て	コマンド
左にある次のスペースを選択部分に含める	Shift-Left	selectleft
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Ctrl-Alt-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Ctrl-Alt-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Ctrl-Alt-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Ctrl-Alt-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Alt-K	selectOrFindNext
前に一致する選択部分を選択または検索する	Alt-Shift-K	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイルの末尾までを選択に含める	Ctrl-Shift-End	selecttoend

説明	割り当て	コマンド
カーソルから現在の行の末尾までを選択に含める	Alt-Shift-Right	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Alt-Shift-Left	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Ctrl-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Ctrl-Shift-Home	selecttostart
上にある次の行を選択部分に含める	Shift-Up	selectup
カーソル左にある次の単語までを選択に含める	Ctrl-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Ctrl-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Ctrl-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Ctrl--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Ctrl-Alt-S	sortlines
現在の行の末尾にカーソルを追加する	Ctrl-Alt-L	splitIntoLines
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Ctrl-Shift-/	toggleBlockComment

説明	割り当て	コマンド
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Ctrl-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Alt-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Ctrl-Alt-E	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Ctrl-Q	toggleWordWrap
選択部分をすべて小文字に変更する	Ctrl-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Ctrl-U	touppercase
選択部分を移動する	Alt-X	transposeletters
選択されたコードを展開する	Alt-Shift-L Ctrl-Shift-F1	unfold
ファイル全体でコードの折りたたみを展開する	Alt-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Ctrl-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Ctrl-Alt-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Ctrl-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Ctrl-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Ctrl-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Alt-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Alt-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Ctrl-B	build
現在一時停止しているプロセスを再開する	F8	resume
現在のアプリケーションを実行またはデバッグする	Alt-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11	stepinto
現在の関数範囲をステップアウトする	Shift-F11	stepout
スタックの現在の式をステップオーバーする	F10	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop
現在のファイルのビルドを停止する	Ctrl-Shift-C	stopbuild

IDE の Windows/Linux Emacs AWS Cloud9 キー割り当てリファレンス

このステップでは、Windows / Linux Emacs キー割り当てリファレンスを選択する方法を示します AWS Cloud9。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。

3. [Keyboard Mode (キーボードモード)] で、[Emacs] を選択します。
4. [Operating System (オペレーティングシステム)] で、[Windows / Linux] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、IDE の Windows/Linux AWS Cloud9 オペレーティングシステム用の Emacs キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)
- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Ctrl-Shift-C	addwatchfromselection
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Ctrl-Space Alt-Space	complete
コードを補完した後、上書きする	Ctrl-Shift-Space Alt-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Ctrl-C	copy

説明	割り当て	コマンド
選択したものをクリップボードに切り取る	Ctrl-X	cut
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Ctrl-F	find
現在のドキュメント内で検索に一致するものをすべて選択する	Ctrl-Alt-K	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	Ctrl-K	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Ctrl-Shift-K	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree
選択した JavaScript コードを再フォーマットする	Ctrl-Shift-B	formatcode
行に移動するボックスを表示する	Ctrl-G	gotoline

説明	割り当て	コマンド
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F3	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Ctrl-Shift-U	lambdaUploadFunction
新規ファイルを作成する	Alt-N	newfile
[Preferences (設定)] タブを表示する	Ctrl-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Alt-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Ctrl-V	paste
エラーを修正するための候補を表示する	Ctrl-F3	quickfix
前回のアクションを繰り返す	Ctrl-Shift-Z Ctrl-Y	redo
プレビューペインを最新の情報に更新する	Ctrl-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Ctrl-Alt-R	renameVar

説明	割り当て	コマンド
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Alt-Shift-F Ctrl-H	replace
初期化スクリプトを再実行する	Ctrl-Enter	rerunInitScript
環境の再起動	Ctrl-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Ctrl-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Alt-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Ctrl-S	save
現在のファイルを別のファイル名でディスクに保存する	Ctrl-Shift-S	saveas
複数のファイルで検索と置換バーを表示する	Ctrl-Shift-F	searchinfiles
[Process List (プロセスリスト)] ダイアログボックスを表示する	Ctrl-Alt-P	showprocesslist
前回のアクションを元に戻す	Ctrl-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Ctrl-Alt-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Alt-Shift-W	closealltabs
現在のペインを閉じる	Ctrl-W	closepane
現在のタブを閉じる	Alt-W	closetab
下のペインに移動	Ctrl-Meta-Down	gotopanedown
左のペインに移動	Ctrl-Meta-Left	gotopaneleft
右のペインに移動	Ctrl-Meta-Right	gotopaneright
上のペインに移動	Ctrl-Meta-Up	gottopaneup
左のタブに移動	Ctrl-[	gototableft
右のタブに移動	Ctrl-]	gototabright
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Ctrl-Meta-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Ctrl-Meta-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一	Ctrl-Meta-Right	movetabright

説明	割り当て	コマンド
番右の場合はそこで分割タブを作成する		
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Ctrl-Meta-Up	movetabup
次のペインに移動する	Ctrl-`	nextpane
次のタブに移動する	Ctrl-Tab Alt-`	nexttab
前のペインに移動する	Ctrl-Shift-`	previouspane
前のタブに移動する	Ctrl-Shift-Tab Alt-Shift-`	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Alt-Shift-T	reopenLastTab
ファイルツリーで現在のタブを表示する	Ctrl-Shift-L	revealtab
10 番目のタブに移動する	Ctrl-0	tab0
最初のタブに移動する	Ctrl-1	tab1
2 番目のタブに移動する	Ctrl-2	tab2
3 番目のタブに移動する	Ctrl-3	tab3
4 番目のタブに移動する	Ctrl-4	tab4
5 番目のタブに移動する	Ctrl-5	tab5
6 番目のタブに移動する	Ctrl-6	tab6
7 番目のタブに移動する	Ctrl-7	tab7

説明	割り当て	コマンド
8 番目のタブに移動する	Ctrl-8	tab8
9 番目のタブに移動する	Ctrl-9	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Ctrl-E Ctrl-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Ctrl-. F1	gotocommand
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Ctrl-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Ctrl-Shift-0	gotosymbol
[Outline (アウトライン)] ウィンドウを表示する	Ctrl-Shift-E	outline
[Console (コンソール)] ウィンドウが非表示の場合は表示し、表示されている場合は非表示にする	F6	toggleconsole
[Environment (環境)] ウィンドウが非表示の場合は表示	Ctrl-I	toggletree

説明	割り当て	コマンド
し、表示されている場合は非表示にする		

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行上にカーソルを追加する、または既にカーソルが追加されている場合はその上にもう 1 つカーソルを追加する	Ctrl-Alt-Up	addCursorAbove
アクティブなカーソルの 1 行上に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行上に移動する	Ctrl-Alt-Shift-Up	addCursorAboveSkipCurrent
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加されている場合はその下にもう 1 つカーソルを追加する	Ctrl-Alt-Down	addCursorBelow
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Ctrl-Alt-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ	Ctrl-Alt-A	alignCursors

説明	割り当て	コマンド
空間に移動する (正しく配置されていない場合)		
バックスペースで 1 つのスペースを削除する	Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Ctrl-]	blockindent
選択部分を 1 タブ分アウトデントする	Ctrl-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Ctrl-Z Ctrl-Shift-Z Ctrl-Y	cancelBrowserUndoInAce
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Alt-Shift-Down	copylinesdown
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Alt-Shift-Up	copylinesup
選択を切り取る。何も選択していない場合は、スペースを 1 つ削除する	Shift-Delete	cut_or_delete
1 つのスペースを削除する	Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Ctrl-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Ctrl-Shift-L	expandtoline

説明	割り当て	コマンド
上にある次の一致する記号まで選択部分に含める	Ctrl-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Alt-L Ctrl-F1	fold
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Alt-0	foldOther
1 行下に移動する	Down	golinedown
1 行上に移動する	Up	golineup
ファイルの末尾に移動する	Ctrl-End	gotoend
1 つ左のスペースに移動する	Left	gotoleft
現在の行の末尾に移動する	Alt-Right End	gotolineend
現在の行の行頭に移動する	Alt-Left Home	gotolinestart
次のエラーに移動する	Alt-E	goToNextError
1 ページ下に移動する	Page Down	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Alt-Shift-E	goToPreviousError
1 つ右のスペースに移動する	Right	gotoright
ファイルの最初に移動する	Ctrl-Home	gotostart
1 つ左の単語に移動する	Ctrl-Left	gotowordleft
1 つ右の単語に移動する	Ctrl-Right	gotowordright

説明	割り当て	コマンド
選択部分を 1 タブ分インデントする	Tab	indent
現在の範囲内の一致する記号に移動する	Ctrl-P	jumptomatching
フォントサイズを大きくする	Ctrl-+ Ctrl-=	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Ctrl-Shift-Down	modifyNumberDown
カーソルの左にある数字を 1 増やす (数字の場合)	Ctrl-Shift-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Alt-Down	movelinesdown
選択範囲を 1 行上に移動する	Alt-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
現在の行の内容を削除する	Ctrl-D	removeline
カーソルから現在の行の末尾までを削除する	Alt-Delete	removetolineend
現在の行の行頭からカーソルまでを削除する	Alt-Backspace	removetolinestart
カーソルの左にある単語を削除する	Ctrl-Backspace	removewordleft
カーソルの右にある単語を削除する	Ctrl-Delete	removewordright

説明	割り当て	コマンド
以前に記録されたキーストロークを再生する	Ctrl-Shift-E	replaymacro
現在のファイルの 1 行下にスクロールする	Ctrl-Down	scrolldown
現在のファイルの 1 行上にスクロールする	Ctrl-Up	scrollup
選択可能な内容をすべて選択する	Ctrl-A	selectall
下にある次の行を選択部分に含める	Shift-Down	selectdown
左にある次のスペースを選択部分に含める	Shift-Left	selectleft
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Ctrl-Alt-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Ctrl-Alt-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Ctrl-Alt-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Ctrl-Alt-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Alt-K	selectOrFindNext

説明	割り当て	コマンド
前に一致する選択部分を選択 または検索する	Alt-Shift-K	selectOrFindPrevious
カーソルから現在のページの 末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの 先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスパー スまでを選択に含める	Shift-Right	selectright
カーソルから現在のファイル の末尾までを選択に含める	Ctrl-Shift-End	selecttoend
カーソルから現在の行の末尾 までを選択に含める	Alt-Shift-Right	selecttolineend
現在の行の行頭からカーソル までを選択に含める	Alt-Shift-Left	selecttolinestart
カーソルから現在のスコープ の次に一致する記号までを含 める	Ctrl-Shift-P	selecttomatching
カーソルから現在のファイル の先頭までを選択に含める	Ctrl-Shift-Home	selecttostart
上にある次の行を選択部分に 含める	Shift-Up	selectup
カーソル左にある次の単語ま でを選択に含める	Ctrl-Shift-Left	selectwordleft
カーソル右にある次の単語ま でを選択に含める	Ctrl-Shift-Right	selectwordright

説明	割り当て	コマンド
[Preferences (設定)] タブを表示する	Ctrl-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Ctrl--	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	Ctrl-Alt-S	sortlines
現在の行の末尾にカーソルを追加する	Ctrl-Alt-L	splitIntoLines
カーソルの内容を行の末尾に移動し、独自の行にする	Ctrl-0	splitline
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Ctrl-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Ctrl-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	F2	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Alt-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Ctrl-Alt-E	togglerecording

説明	割り当て	コマンド
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Ctrl-Q	toggleWordWrap
選択部分をすべて小文字に変更する	Ctrl-Shift-U	tolowercase
選択部分をすべて大文字に変更する	Ctrl-U	touppercase
選択部分を移動する	Alt-X	transposeletters
選択されたコードを展開する	Alt-Shift-L Ctrl-Shift-F1	unfold
ファイル全体でコードの折りたたみを展開する	Alt-Shift-0	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Ctrl-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Ctrl-Alt-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Ctrl-.	emmet_select_next_item

説明	割り当て	コマンド
前の編集可能なコード部分に移動する	Shift-Ctrl-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Ctrl-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Alt-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Alt-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	Ctrl-B	build
現在一時停止しているプロセスを再開する	F8	resume
現在のアプリケーションを実行またはデバッグする	Alt-F5	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11	stepinto

説明	割り当て	コマンド
現在の関数範囲をステップアウトする	Shift-F11	stepout
スタックの現在の式をステップオーバーする	F10	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop
現在のファイルのビルドを停止する	Ctrl-Shift-C	stopbuild

IDE の Windows/Linux Sublime AWS Cloud9 キー割り当てリファレンス

このステップでは、AWS Cloud9の Windows/Linux Sublime キー割り当てリファレンスを選択する方法を示します。

1. メニューバーで、AWS Cloud9、設定を選択してください。
2. [Preferences (設定)] タブで、[Keybindings (キー割り当て)] を選択します。
3. [Keyboard Mode (キーボードモード)] で、[Sublime] を選択します。
4. [Operating System (オペレーティングシステム)] で、[Windows / Linux] を選択します。

キー割り当ての詳細については、「[キー割り当ての使用](#)」を参照してください。

以下は、AWS Cloud9 IDE の Windows/Linux オペレーティングシステム用の Sublime キーボードモードのキー割り当てのリストです。

- [全般](#)
- [タブ](#)
- [パネル](#)
- [コードエディタ](#)
- [emmet](#)

- [ターミナル](#)
- [実行およびデバッグ](#)

General

説明	割り当て	コマンド
選択したものをウォッチ式として追加する	Ctrl-Shift-C	addwatchfromselection
切り取った選択部分をクリップボードから削除する	Esc	clearcut
コード補完のコンテキストメニューを表示する	Ctrl-Space	complete
コードを補完した後、上書きする	Ctrl-Shift-Space Alt-Shift-Space	completeoverwrite
選択したものをクリップボードにコピーする	Ctrl-C	copy
選択したものをクリップボードに切り取る	Ctrl-X	cut
カーソルから行の先頭までを削除する	Ctrl-Shift-Backspace Ctrl-K Ctrl-Backspace	delete_to_hard_bol
カーソルから行の末尾までを削除する	Ctrl-Shift-Delete Ctrl-K Ctrl-K	delete_to_hard_eol
コードを展開する (該当する場合)	Tab	expandSnippet
現在のドキュメントで検索と置換バーを表示する	Ctrl-F	find

説明	割り当て	コマンド
選択範囲のすべての一致をハイライトする	Alt-F3	find_all_under
選択範囲の次の一致をハイライトする	Ctrl-F3	find_under
ハイライト部分のカーソルとすべての一致をハイライトする	Ctrl-D	find_under_expand
カーソルの周囲をハイライト表示しハイライト部分のすべての一致をアウトラインする	Ctrl-K Ctrl-D	find_under_expand_skip
選択範囲の前の一致をハイライトする	Ctrl-Shift-F3	find_under_prev
現在のドキュメント内で検索に一致するものをすべて選択する	Ctrl-Alt-K	findAll
現在のドキュメントで最後に入力した検索クエリの次の一致に移動する	F3	findnext
現在のドキュメントで最後に入力した検索クエリの前の一致に移動する	Shift-F3	findprevious
エディタのアクティブなファイルで挿入ポイントにすべての既知のリファレンスを表示する	Shift-F3	findReferences
[Environment (環境)] ウィンドウを開き、ファイルのリストをアクティブにする	Shift-Esc	focusTree

説明	割り当て	コマンド
選択した JavaScript コードを再フォーマットする	Ctrl-Alt-F	formatcode
行に移動するボックスを表示する	Ctrl-G	gotoline
検索と置換バーを非表示にする (表示されている場合)	Esc	hidesearchreplace
変数の定義またはカーソルがある関数に移動する	F12	jumptodef
ローカル Lambda 関数が [AWS リソース] ウィンドウの [Lambda] セクションで選択されている場合、この関数をリモート関数として Lambda にアップロードすることを試行する	Ctrl-Shift-U	lambdaUploadFunction
現在の単語の末尾に移動する	Ctrl-Right	moveToWordEndRight
現在の単語の先頭に移動する	Ctrl-Left	moveToWordStartLeft
新規ファイルを作成する	Alt-N	newfile
[Preferences (設定)] タブを表示する	Ctrl-,	openpreferences
[Terminal (ターミナル)] タブを開き、ファイルのリストで選択したファイルの親フォルダに切り替える	Alt-L	opentermhere
クリップボードの現在の内容をカーソル位置に貼り付ける	Ctrl-V	paste

説明	割り当て	コマンド
エラーを修正するための候補を表示する	Ctrl-F3	quickfix
前回のアクションを繰り返す	Ctrl-Shift-Z Ctrl-Y	redo
プレビューペインを最新の情報に更新する	Ctrl-Enter	reloadpreview
選択部分の名前変更のリファクタリングを開始する	Ctrl-Alt-R	renameVar
現在のドキュメントの検索と置換バーを表示し、「置換後の文字列」式にフォーカスする	Ctrl-H	replace
検索式のすべての一致を検索と置換バーの式で置き換える	Ctrl-Alt-Enter	replaceall
検索式の次の一致を検索と置換バーの式で置き換える	Ctrl-Shift-H	replacenext
初期化スクリプトを再実行する	Ctrl-Enter	rerunInitScript
環境の再起動	Ctrl-R	restartc9
現在のファイルを最後に保存されたバージョンにリセットする	Ctrl-Shift-Q	reverttosaved
開いている各ファイルを保存されているバージョンにリセットする	Alt-Shift-Q	reverttosavedall
現在のファイルをディスクに保存する	Ctrl-S	save

説明	割り当て	コマンド
現在のファイルを別のファイル名でディスクに保存する	Ctrl-Shift-S	saveas
複数のファイルで検索と置換バーを表示する	Ctrl-Shift-F	searchinfiles
カーソルから単語の末尾までを選択に含める	Ctrl-Shift-Right	selectToWordEndRight
カーソルから単語の先頭までを選択に含める	Ctrl-Shift-Left	selectToWordStartLeft
[Process List (プロセスリスト)] ダイアログボックスを表示する	Ctrl-Alt-P	showprocesslist
前回のアクションを元に戻す	Ctrl-Z	undo

タブ

説明	割り当て	コマンド
現在のペインで現在のタブ以外の開いているすべてのタブを閉じる	Ctrl-Alt-W	closeallbutme
すべてのペインで開いているタブをすべて閉じる	Alt-Shift-W	closealltabs
現在のペインを閉じる	Ctrl-W	closepane
現在のタブを閉じる	Alt-W	closetab
下のペインに移動	Ctrl-Meta-Down	gotopanedown
左のペインに移動	Ctrl-Meta-Left	gotopaneleft

説明	割り当て	コマンド
右のペインに移動	Ctrl-Meta-Right	gotopaneright
上のペインに移動	Ctrl-Meta-Up	gottopaneup
左のタブに移動	Ctrl-Page Up	gototableft
右のタブに移動	Ctrl-Page Down	gototabright
現在のタブを 1 つ下のペインに移動する、またはタブが一番下の場合はそこで分割タブを作成する	Ctrl-Meta-Down	movetabdown
現在のタブを 1 つ左のペインに移動する、またはタブが一番左の場合はそこで分割タブを作成する	Ctrl-Meta-Left	movetableft
現在のタブを 1 つ右のペインに移動する、またはタブが一番右の場合はそこで分割タブを作成する	Ctrl-Meta-Right	movetabright
現在のタブを 1 つ上のペインに移動する、またはタブが一番上の場合はそこで分割タブを作成する	Ctrl-Meta-Up	movetabup
次のタブに移動する	Ctrl-Tab	nexttab
前のペインに移動する	Ctrl-Shift-`	previouspane
前のタブに移動する	Ctrl-Shift-Tab	previoustab
最後のタブに戻る	Esc	refocusTab
最後のタブを再度開く	Ctrl-Shift-T	reopenLastTab

説明	割り当て	コマンド
ファイルツリーで現在のタブを表示する	Ctrl-E	revealtab
10 番目のタブに移動する	Ctrl-0	tab0
最初のタブに移動する	Ctrl-1	tab1
2 番目のタブに移動する	Ctrl-2	tab2
3 番目のタブに移動する	Ctrl-3	tab3
4 番目のタブに移動する	Ctrl-4	tab4
5 番目のタブに移動する	Ctrl-5	tab5
6 番目のタブに移動する	Ctrl-6	tab6
7 番目のタブに移動する	Ctrl-7	tab7
8 番目のタブに移動する	Ctrl-8	tab8
9 番目のタブに移動する	Ctrl-9	tab9

パネル

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go to Anything (どこにでも移動)] モードで表示する	Ctrl-E Ctrl-P	gotoanything
[Go (移動)] ウィンドウを [Go to Command (コマンドに移動)] モードで表示する	Ctrl-. F1	gotocommand

説明	割り当て	コマンド
[Go (移動)] ウィンドウを [Go To Line (行に移動)] モードで表示する	Ctrl-0	gotofile
[Go (移動)] ウィンドウを [Go to Symbol (記号に移動)] モードで表示する	Ctrl-Shift-0	gotosymbol
[Outline (アウトライン)] ウィンドウを表示する	Ctrl-R Ctrl-Shift-R	outline
[Console (コンソール)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Ctrl-`	toggleconsole
[Environment (環境)] ウィ ンドウが非表示の場合は表示 し、表示されている場合は非 表示にする	Ctrl-K Ctrl-B	toggletree

コードエディタ

説明	割り当て	コマンド
アクティブなカーソルの 1 行 上にカーソルを追加する、ま たは既にカーソルが追加され ている場合はその上にもう 1 つカーソルを追加する	Ctrl-Alt-Up	addCursorAbove
アクティブなカーソルの 1 行 上に 2 つ目のカーソルを追 加する、または既に 2 つ目の カーソルが追加されている場	Ctrl-Alt-Shift-Up	addCursorAboveSkip Current

説明	割り当て	コマンド
合は 2 つ目のカーソルを 1 行上に移動する		
アクティブなカーソルの 1 行下にカーソルを追加する、または既にカーソルが追加されている場合はその下にもう 1 つカーソルを追加する	Ctrl-Alt-Down	addCursorBelow
アクティブなカーソルの 1 行下に 2 つ目のカーソルを追加する、または既に 2 つ目のカーソルが追加されている場合は 2 つ目のカーソルを 1 行下に移動する	Ctrl-Alt-Shift-Down	addCursorBelowSkipCurrent
行ごとにすべてのカーソルをアクティブなカーソルと同じ空間に移動する (正しく配置されていない場合)	Ctrl-Alt-A	alignCursors
バックスペースで 1 つのスペースを削除する	Shift-Backspace Backspace	backspace
選択部分を 1 タブ分インデントする	Ctrl-]	blockindent
選択部分を 1 タブ分アウトデントする	Ctrl-[	blockoutdent
フォーカスをエディタから IDE の別の場所に切り替えることができるかどうかを制御する	Ctrl-Z Ctrl-Shift-Z Ctrl-Y	cancelBrowserUndoInAce
選択部分を中央揃えにする	Ctrl-K Ctrl-C	centerselection

説明	割り当て	コマンド
行の内容をコピーし、1 つ下の行にコピーした内容を貼り付ける	Alt-Shift-Down	copylinesdown
行の内容をコピーし、1 つ上の行にコピーした内容を貼り付ける	Alt-Shift-Up	copylinesup
選択を切り取る。何も選択していない場合は、スペースを 1 つ削除する	Shift-Delete	cut_or_delete
1 つのスペースを削除する	Delete	del
選択部分の内容をコピーし、選択部分のすぐ下にコピーした内容を貼り付ける	Ctrl-Shift-D	duplicateSelection
現在の行の内容を選択に含める	Ctrl-Shift-L	expandtoline
上にある次の一致する記号まで選択部分に含める	Ctrl-Shift-M	expandToMatching
選択されたコードを折りたたむ。折りたたまれたユニットが選択されている場合は展開する	Alt-L Ctrl-F1	fold
現在の選択範囲を除き、折りたたむことができる要素をすべて折りたたむ	Ctrl-K Ctrl-1	foldOther
1 行下に移動する	Down	golinedown
1 行上に移動する	Up	golineup

説明	割り当て	コマンド
ファイルの末尾に移動する	Ctrl-End	gotoend
1 つ左のスペースに移動する	Left	gotoleft
現在の行の末尾に移動する	Alt-Right End	gotolineend
現在の行の行頭に移動する	Alt-Left Home	gotolinestart
次のエラーに移動する	Ctrl-F6	goToNextError
1 ページ下に移動する	Page Down	gotopagedown
1 ページ上に移動する	Page Up	gotopageup
前のエラーに移動する	Ctrl-Shift-F6	goToPreviousError
1 つ右のスペースに移動する	Right	gotoright
ファイルの最初に移動する	Ctrl-Home	gotostart
1 つ左の単語に移動する	Ctrl-Left	gotowordleft
1 つ右の単語に移動する	Ctrl-Right	gotowordright
選択部分を 1 タブ分インデントする	Tab	indent
カーソルから単語の先頭までを選択に含める	Ctrl-J	joinlines
現在の範囲内の一致する記号に移動する	Ctrl-M	jumptomatching
フォントサイズを大きくする	Ctrl-- Ctrl-= Ctrl-+	largerfont
カーソルの左にある数字を 1 減らす (数字の場合)	Alt-Down	modifyNumberDown

説明	割り当て	コマンド
カーソルの左にある数字を 1 増やす (数字の場合)	Alt-Up	modifyNumberUp
選択範囲を 1 行下に移動する	Ctrl-Shift-Down	movelinesdown
選択範囲を 1 行上に移動する	Ctrl-Shift-Up	movelinesup
選択部分を 1 タブ分アウトデントする	Shift-Tab	outdent
上書きモードをオンにする。 オンの場合はオフにする	Insert	overwrite
現在の行の内容を削除する	Ctrl-Shift-K	removeLine
カーソルから現在の行の末尾 までを削除する	Alt-Delete	removetolineend
現在の行の行頭からカーソル までを削除する	Alt-Backspace	removetolinestart
カーソルの左にある単語を削除する	Ctrl-Backspace	removewordleft
カーソルの右にある単語を削除する	Ctrl-Delete	removewordright
以前に記録されたキースト ロークを再生する	Ctrl-Shift-Q	replaymacro
現在のファイルの 1 行下にスクロールする	Ctrl-Down	scrollDown
現在のファイルの 1 行上にスクロールする	Ctrl-Up	scrollUp
選択可能な内容をすべて選択する	Ctrl-A	selectAll

説明	割り当て	コマンド
下にある次の行を選択部分に含める	Shift-Down	selectdown
左にある次のスペースを選択部分に含める	Shift-Left	selectleft
カーソルから、現在の行の残り部分を選択に含める	Shift-End	selectlineend
現在の行の先頭を、カーソルまで選択に含める	Shift-Home	selectlinestart
選択部分以降で一致する選択部分を含める	Ctrl-Alt-Right	selectMoreAfter
選択部分以前で一致する選択部分を含める	Ctrl-Alt-Left	selectMoreBefore
選択部分以降で次に一致する選択部分を含める	Ctrl-Alt-Shift-Right	selectNextAfter
選択部分以前で次に一致する選択部分を含める	Ctrl-Alt-Shift-Left	selectNextBefore
次に一致する選択部分を選択または検索する	Alt-K	selectOrFindNext
前に一致する選択部分を選択または検索する	Alt-Shift-K	selectOrFindPrevious
カーソルから現在のページの末尾までを選択に含める	Shift-Page Down	selectpagedown
カーソルから現在のページの先頭までを選択に含める	Shift-Page Up	selectpageup
カーソル右にある次のスペースまでを選択に含める	Shift-Right	selectright

説明	割り当て	コマンド
カーソルから現在のファイルの末尾までを選択に含める	Ctrl-Shift-End	selecttoend
カーソルから現在の行の末尾までを選択に含める	Alt-Shift-Right	selecttolineend
現在の行の行頭からカーソルまでを選択に含める	Alt-Shift-Left	selecttolinestart
カーソルから現在のスコープの次に一致する記号までを含める	Ctrl-Shift-P	selecttomatching
カーソルから現在のファイルの先頭までを選択に含める	Ctrl-Shift-Home	selecttostart
上にある次の行を選択部分に含める	Shift-Up	selectup
カーソル左にある次の単語までを選択に含める	Ctrl-Shift-Left	selectwordleft
カーソル右にある次の単語までを選択に含める	Ctrl-Shift-Right	selectwordright
[Preferences (設定)] タブを表示する	Ctrl-,	showSettingsMenu
以前の選択をすべてクリア	Esc	singleSelection
フォントサイズを小さくする	Ctrl-- Ctrl-Shift-= Ctrl-Shift-+	smallerfont
複数の行が選択されている場合は、ソート順に並べ替える	F9	sortlines
現在の行の末尾にカーソルを追加する	Ctrl-Shift-L	splitIntoLines

説明	割り当て	コマンド
選択部分をブロックコメント文字で囲む、または既にある場合は削除する	Ctrl-Shift-/	toggleBlockComment
選択された各行の先頭にコメント行文字を追加する、または既にある場合は削除する	Ctrl-/	togglecomment
コードを折りたたむ、またはコードの折りたたみがある場合は削除する	Ctrl-Shift-[	toggleFoldWidget
親コードを折りたたむ、または折りたたみがある場合は削除する	Alt-F2	toggleParentFoldWidget
キーストロークの記録を開始する、または既に記録している場合は停止する	Ctrl-Q	togglerecording
単語をラッピングする、または既にラッピング中の場合は単語のラッピングを停止する	Ctrl-Q	toggleWordWrap
選択部分をすべて小文字に変更する	Ctrl-K Ctrl-L	tolowercase
選択部分をすべて大文字に変更する	Ctrl-K Ctrl-U	touppercase
選択部分を移動する	Alt-X	transposeletters
選択されたコードを展開する	Ctrl-Shift-]	unfold
ファイル全体でコードの折りたたみを展開する	Ctrl-K Ctrl-0 Ctrl-K Ctrl-J	unfoldall

emmet

説明	割り当て	コマンド
単純計算式 (2*4 や 10/2 など) を評価して結果を出力する	Shift-Ctrl-Y	emmet_evaluate_math_expression
現在のファイル構文に応じて、CSS 風の省略形を HTML、XML、または CSS コードに展開する	Ctrl-Alt-E	emmet_expand_abbreviation
タブストップを使用して展開された CSS 風の省略形をトラバースする	Tab	emmet_expand_abbreviation_with_tab
次の編集可能なコード部分に移動する	Shift-Ctrl-.	emmet_select_next_item
前の編集可能なコード部分に移動する	Shift-Ctrl-,	emmet_select_previous_item
省略形を展開し、生成されたスニペットの最後の要素内に現在の選択部分を配置する	Shift-Ctrl-A	emmet_wrap_with_abbreviation

Terminal

説明	割り当て	コマンド
新しい [Terminal (ターミナル)] タブを開く	Alt-T	openterminal
エディタと [Terminal (ターミナル)] タブを切り替える	Alt-S	switchterminal

実行およびデバッグ

説明	割り当て	コマンド
現在のファイルをビルドする	F7 Ctrl-B	build
現在一時停止しているプロセスを再開する	F8	resume
現在のアプリケーションを実行またはデバッグする	Ctrl-Shift-B	run
最後に実行したファイルを実行またはデバッグする	F5	runlast
スタックの次の関数をステップインする	F11	stepinto
現在の関数範囲をステップアウトする	Shift-F11	stepout
スタックの現在の式をステップオーバーする	F10	stepover
現在のアプリケーションの実行またはデバッグを停止する	Shift-F5	stop
現在のファイルのビルドを停止する	Ctrl-Break	stopbuild

IDE AWS Cloud9 のコマンドリファレンス

AWS Cloud9 統合開発環境 (IDE) でコマンドを実行するには：

1. [Go (移動)] ボタン (拡大鏡) を選択して [Go (移動)] ウィンドウを表示します。[Go (移動)] ボタンが表示されない場合は、メニューバーで [Window (ウィンドウ)]、[Go (移動)] の順に選択します。

2. [Go to Anything (どこにでも移動)] ボックスで、[コマンドグループ (コードエディタなど)] の名前の入力をスタートします。グループには、共通のテーマまたは IDE の特徴を中心に編成された複数のコマンドが含まれます。
3. コマンド見出しで、実行する特定のコマンドをグループから選択します。

使用できるコマンドグループ

コマンドグループ	説明
AWS	AWS ツールキット のコマンド
Clipboard	コンテンツをコピーアンドペーストするコマンド
Code Editor	コードエディターのインターフェースをナビゲートし、エディターのコンテンツを操作するためのコマンド
Emmet	HTML および CSS コンテンツに使用される Emmet ツールキットを操作するためのコマンド
General	IDE の構成ファイルとプロジェクトファイルを管理する様々なコマンド
Panels	IDE インタフェースでパネルの表示を管理するためのコマンド
Run & Debug	でプロジェクトを実行およびデバッグするためのコマンド AWS Cloud9
Tabs	IDE インタフェースのタブの表示とナビゲーションを管理するコマンド
Terminal	コマンドラインターミナルの管理コマンド
Window	IDE ウィンドウ内のペインのレイアウトを管理するコマンド

他の AWS サービスの使用

を使用する場合は AWS Cloud9、Amazon Lightsail、AWS CodeStar、および と緊密に連携できます AWS CodePipeline。このセクションのトピックは、これを行う方法についてです。

Important

Toolkit AWS 機能は、AWS Lambda、AWS Serverless Application Model Amazon S3 などの主要な AWS サービスを操作するための便利なビジュアルインターフェイスを提供します。詳細については、「[Toolkit の使用 AWS](#)」を参照してください。

トピック

- [IDE AWS Cloud9 でのAmazon Lightsailインスタンスの操作](#)
- [IDE AWS CodePipeline での AWS Cloud9 の使用](#)
- [Amazon CodeCatalyst の使用](#)
- [IDE AWS CDK での AWS Cloud9 の使用](#)

IDE AWS Cloud9 でのAmazon Lightsailインスタンスの操作

AWS Cloud9 IDE を使用して、一般的なアプリケーションやフレームワークで事前設定されたAmazon Lightsailインスタンスでコードを使用できます。これらには WordPress、LAMP (Linux、Apache、MySQL、およびPHP)、Node.js、NGINX、Drupal、および Joomla が含まれます。Amazon Linux、Ubuntu、Debian、FreeBSD、および openSUSE などの Linux ディストリビューションが含まれています。

Lightsail は、便利で迅速なセットアップが可能な仮想プライベートサーバーソリューションを提供します。Lightsailは、コンピューティング、ストレージ、ネットワーク容量と、ウェブサイトやウェブアプリケーションをクラウドにデプロイおよび管理するための機能を提供します。Lightsail は、低価格で予測可能な月額料金のため、プロジェクトをすばやく開始することができます。詳細については、「[Amazon Lightsail の特徴](#)」を参照してください。

このトピックでは、 と互換性のある Linux ベースのLightsailインスタンスを作成してセットアップします AWS Cloud9。次に、SSH AWS Cloud9 開発環境を作成してLightsailインスタンスに接続します。

Note

これらの手順を完了すると、 に料金が発生する可能性があります AWS アカウント。Lightsail などのサービスに対して発生する可能性がある料金も含まれます。詳細については、[Amazon Lightsail の料金](#)を参照してください。

AWS Cloud9 IDE を使用して、サンプルコードを含まない Amazon Linux または Ubuntu Server を実行している Amazon EC2 インスタンスを操作するには、「」を参照してください [開始方法: ベーシックチュートリアル](#)。

- [ステップ 1: Linux ベースの Lightsail インスタンスを作成する](#)
- [ステップ 2: で使用するインスタンスを設定する AWS Cloud9](#)
- [ステップ 3: AWS Cloud9 SSH 開発環境を作成し接続する](#)
- [ステップ 4: AWS Cloud9 IDE を使用してインスタンスのコードを変更する](#)

ステップ 1: Linux ベースの Lightsail インスタンスを作成する

このステップでは、Lightsail コンソールを使用して、Linux ベースのディストリビューションでアプリケーションを実行する Amazon EC2 インスタンスを作成します。このインスタンスには自動的に以下のものが含まれます。

- パブリック IP アドレスとプライベート IP アドレス (静的パブリック IP は後で作成できます)。
- ポート 22 を介した SSH、ポート 80 を介した HTTP、ポート 443 を介した HTTPS を使用して、インスタンスにアクセスします (これらの設定は変更できます。)
- ブロックストレージディスク (後で追加のディスクをアタッチできます)。
- 組み込みのシステムレポート

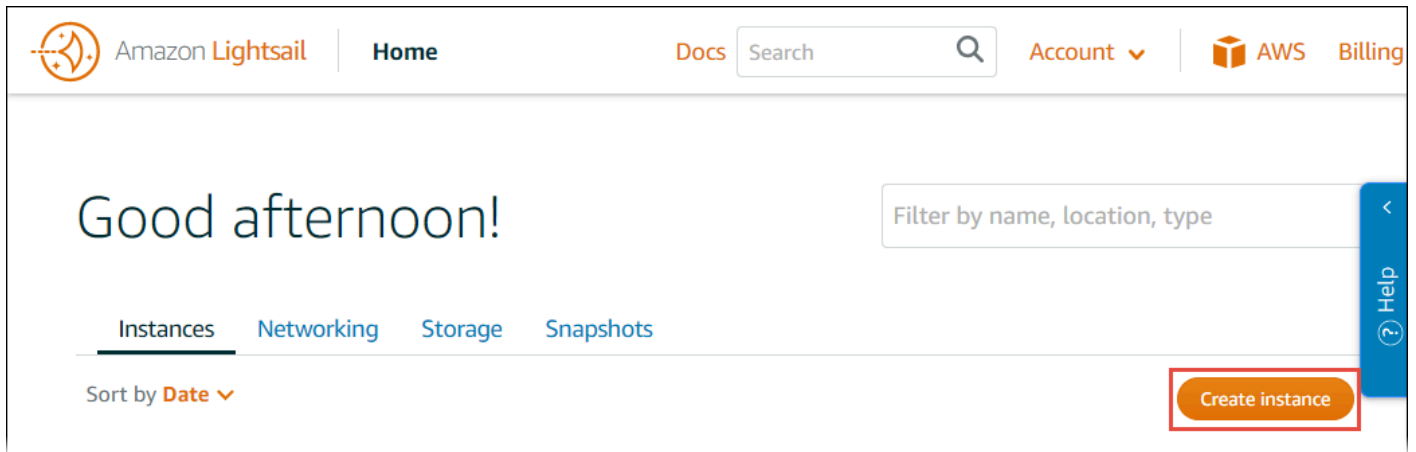
Lightsail コンソールでは、インスタンスを後でバックアップ、再起動、停止、または削除することができます。

1. を開き、<https://lightsail.aws.amazon.com> で Lightsail コンソールにサインインします。

AWS アカウントの IAM IAM 管理者ユーザーの認証情報を使用してサインインすることをお勧めします。IAM 管理者ユーザーとしてサインインできない場合は、AWS アカウント の管理者にチェックしてください。

2. プロンプトが表示されたら、コンソールで使用する言語を選択して、[保存] を選択します。

3. プロンプトが表示されたら、[では、始めましょう] を選択します。
4. [インスタンス] タブが既に選択されているホームページで、[インスタンスの作成] を選択します。



5. インスタンスの場所については、インスタンスを作成する場所 AWS リージョン AWS Cloud9 であることを確認します。詳細については、「Amazon Web Services 全般のリファレンス」の「[AWS Cloud9](#)」を参照してください。AWS リージョン、アベイラビリティゾーン、またはその両方を変更するには、AWS リージョンとアベイラビリティゾーンの変更を選択し、画面の指示に従います。
6. [インスタンスイメージの選択] に対しては、[プラットフォームの選択] には Linux/Unix をすでに選択しており、[Select a blueprint (設計図の選択)] には [アプリ + OS] がすでに選択されている状態で、設計図を選択します。

Pick your instance image ?

Select a platform



Linux/Unix
16 blueprints



Microsoft Windows
3 blueprints

Select a blueprint

Apps + OS

OS Only



WordPress
4.8.1



LAMP Stack
5.6.31



Node.js
8.4.0



Joomla
3.7.5



Magento
2.1.8-1



MEAN
3.4.7



Drupal
8.3.7-1



GitLab CE
9.5.0



Redmine
3.4.2-2



Nginx
1.12.1



Plesk Hosting Stack on Ubuntu
17.5.3

Note

アプリケーションを持たないインスタンスを作成する場合は、[アプリ + OS] の代わりに [OS のみ] を選択してから、ディストリビューションを選択します。

使用可能なオプションについては、Lightsail ウェブサイトの「[Amazon Lightsail インスタンスイメージの選択](#)」を参照してください。

7. [インスタンスプランの選択] で、プランを選択するか、選択済みのデフォルトプランのままにします。
8. [Name your instance] (インスタンスの名前の指定) に、インスタンスの名前を入力するか、推奨のデフォルト名のままにします。
9. インスタンス数には、作成するインスタンスの数を入力するか、デフォルトの 1 インスタンス ([x 1]) のままにします。

10[作成] を選択します。

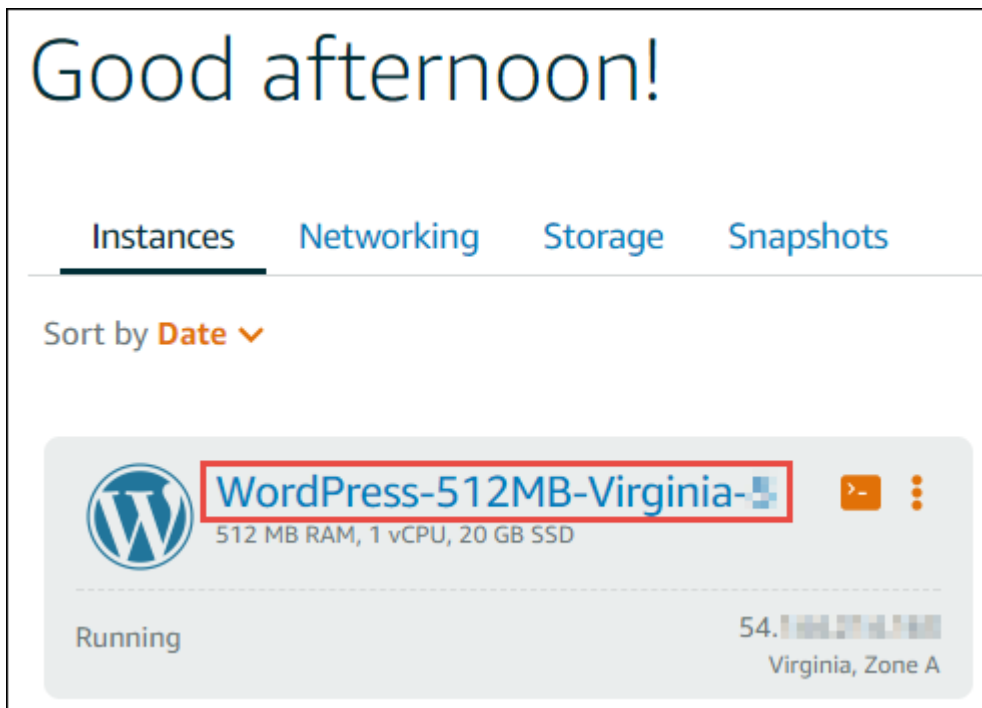
ステップ 2: インスタンスを使用するようにインスタンスを設定する AWS Cloud9

このステップでは、実行中のインスタンスに接続し、後で AWS Cloud9 が使用できるようにセットアップします。

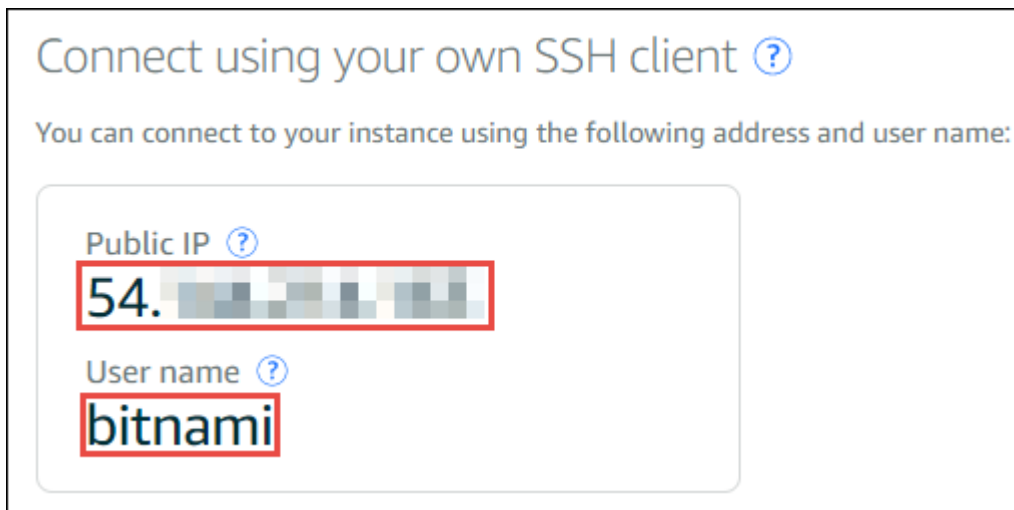
Note

以下の手順では、前のステップで [アプリ + OS] を選択したものとしします。代わりに [OS のみ] と、 [Ubuntu] 以外のディストリビューションを選択した場合は、以下の手順を適切に読み替える必要があります。

1. 前のステップで開いたままの Lightsail コンソールの [インスタンス] タブで、インスタンスのカードにあるインスタンス名を選択します。



2. [Connect] (接続) タブ上の [Connect using your own SSH client] (独自の SSH クライアントを使用して接続) で、[Public IP] (パブリック IP) と [User name] (ユーザー名) の値を書き留めます。後に必要になります。



3. [SSH を使用して接続] を選択します。
4. インスタンスに最新のシステム更新があることを確認してください。これを行うには、表示されたターミナルセッションで、コマンド **sudo apt update** を実行します。
5. Python がインストール済みであるかどうかをチェックし、インストール済みである場合は、バージョンが 2.7 であることを確認します。バージョンを確認するには、コマンド **python --version** を実行し、表示されるバージョン番号を書き留めます。バージョン番号が表示されないか、バージョンが 2.7 でない場合は、コマンド **sudo apt install -y python-minimal** を実行してインスタンスに Python 2.7 をインストールします。
6. Node.js がインストールされているかどうかを確認し、存在する場合はバージョンが 0.6.16 以降であることを確認します。バージョンを確認するには、コマンド **node --version** を実行し、表示されるバージョン番号を書き留めます。バージョン番号が表示されない場合、またはバージョンが 0.6.16 以降でない場合は、Node Version Manager (nvm) を使用してインスタンスに Node.js をインストールすることをお勧めします。

これを行うには、以下のコマンドを一度に 1 つずつ以下の順序で実行します。インスタンスを更新するには、インスタンスに Node Version Manager (nvm) をインストールし、インスタンスで nvm をアクティブにしてから、最新バージョンの Node.js をインスタンスにインストールします。

```
sudo apt update
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
. ~/.bashrc
nvm install node
```

7. コマンド **which node** を実行し、表示される値を書き留めます。後で必要になります。

Note

コマンドの出力 **which node** が のような場合 `/usr/sbin/node`、そのパスに Node.js AWS Cloud9 が見つかりません。代わりに、この手順の前のステップで説明したように、`nvm` を使用して Node.js をインストールします。コマンド `which node` をもう一度実行し、表示される新しい値を書き留めます。

8. インスタンスで [AWS Cloud9 インストーラをダウンロードして実行](#) します。

ステップ 3: SSH AWS Cloud9 開発環境を作成して接続する

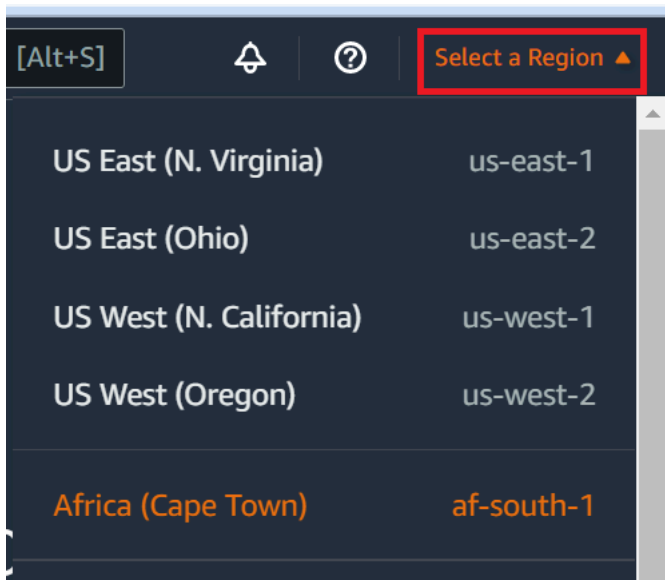
このステップでは、AWS Cloud9 コンソールとインスタンスのターミナルを使用して SSH 環境を作成し、その環境を実行中のインスタンスに接続します。

1. 前のステップでターミナルセッションを開いたままにして、次のように AWS Cloud9 コンソールにサインインします。
 - を使用するののお客様だけであるか、単一の IAM ユーザー AWS アカウント である場合は AWS アカウント、<https://console.aws.amazon.com/cloud9/> にアクセスしてください。
 - 組織が を使用している場合は AWS IAM アイデンティティセンター、サインイン手順について AWS アカウント 管理者にお問い合わせください。

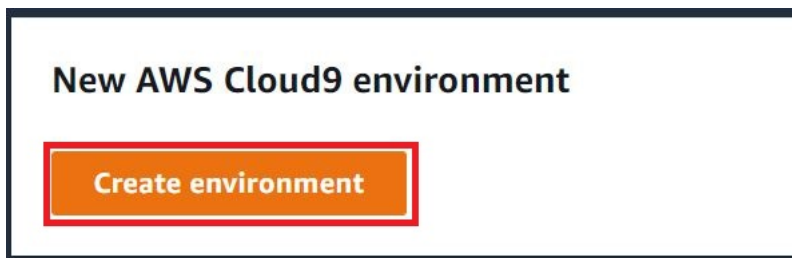
Note

このステップでは、2 つの異なる AWS のサービス を同時に使用します。IAM 管理者ユーザーとして Lightsail コンソールにサインインしたものの、別のエンティティに新しい SSH 環境を所有させたいとします。この場合は、別のウェブブラウザを開き、そのエンティティとして AWS Cloud9 コンソールにサインインすることを提案します。

2. AWS Cloud9 コンソールで、フレームワークでインスタンスを作成した AWS リージョン と一致する を選択します。



3. ようこそページが表示された場合は、新しい AWS Cloud9 環境 で環境の作成 を選択します。それ以外の場合は、[Create environment (環境の作成)] を選択します。



または:



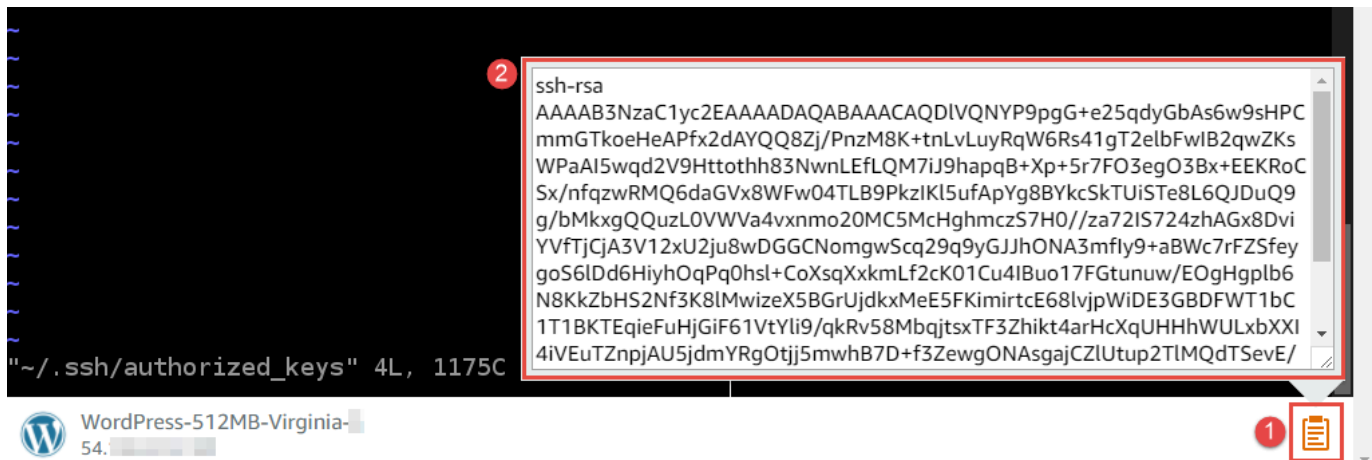
4. [名前環境] ページで、[Name (名前)] に環境の名前を入力します。
5. [説明] フィールドに、環境の説明を追加します。
6. [環境タイプ] で、[既存のコンピューティング] を選択します。[ユーザー] オプションと [ホスト] オプションを表示するには、このオプションを選択する必要があるため、これは重要です。
7. [User] (ユーザー) に [User name] (ユーザー名) の値を入力します。
8. [Host] (ホスト) に [Public IP] (パブリック IP) の値を入力します。
9. [Port (ポート)] は、デフォルト値の [22] のままにしておきます。
10. [その他の詳細] を展開します。
11. 環境パスには、ログイン後 から AWS Cloud9 始まるパスを入力します~/。これはユーザーのホームディレクトリのルートです。

- 12.[Node.js binary path] (Node.js バイナリパス) に、前に書き留めたコマンド **which node** の値を入力します。
- 13.[SSH jump host (SSH ジャンプホスト)] は空白のままにします。
- 14.この環境用に が AWS Cloud9 作成するパブリック SSH キーをシステムクリップボードに保存します。これを行うには、[Copy key to clipboard (キーをクリップボードにコピーする)] を選択します。

Note

コピーしたパブリック SSH キーの値を確認するには、[View public SSH key (パブリック SSH キーの表示)] を展開します。

- 15.インスタンスにコピーしたパブリック SSH キーの値を保存します。これを行うには、インスタンスにインストール済みである一般的なテキストエディタ **vi** を使用します。
- インスタンスのターミナルセッションで、コマンド **vi ~/.ssh/authorized_keys** を実行します。
 - 表示される vi エディタで、ファイルの最後に移動し、挿入モードに切り替えます。これを行うには **I** を押してから **A** を押します。(vi エディタの下部に -- INSERT -- が表示されます)。
 - Enter** を 2 回押して、ファイルの末尾に 2 つの改行を追加します。
 - コピーしたばかりのパブリック SSH キー値を含むシステムクリップボードの内容をターミナルセッションのクリップボードに貼り付けます。これを行うには、ターミナルセッションウィンドウの一番下にあるクリップボードボタンをクリックし、システムクリップボードの内容をボックスに貼り付けます。



- ターミナルセッションのクリップボードの内容を vi エディタに貼り付けます。これを行うには、vi エディター内の挿入ポイントで **Ctrl + Shift + V** を押します。

- f. ファイルを保存します。これを行うには、Esc を押してコマンドモードを入力します。(vi エディターの下部から [-- INSERT --] が消えます)。:wq と入力します (ファイルを write して、vi エディタを quit します)。次に Enter を押します。

16.AWS Cloud9 コンソールに戻り、次のステップを選択します。

- 17.[Review choices (選択内容を確認する)] ページで、[Create environment (環境の作成)] を選択します。が環境 AWS Cloud9 を作成し、環境の AWS Cloud9 IDE を表示するまで待ちます。これには数分間かかる場合があります。

が環境 AWS Cloud9 を作成すると、環境の AWS Cloud9 IDE が表示されます。

が少なくとも 5 分後に IDE を表示 AWS Cloud9 しない場合、ウェブブラウザ、AWS アクセス許可、インスタンス、または関連する Virtual Private Cloud (VPC) に問題がある可能性があります。環境の修正については、[トラブルシューティング](#)の「環境を開くことができない」を参照してください。

ステップ 4: AWS Cloud9 IDE を使用してインスタンスのコードを変更する

新しい環境に IDE が表示されたので、Lightsail ターミナルセッションの代わりに IDE のターミナルセッションを使用できます。この IDE では、豊富なコード編集エクスペリエンスを提供しており、複数のプログラミング言語とランタイムデバグがサポートされています。また、IDE には、色テーマ、ショートカットキーバインディング、プログラミング言語固有の構文の色付けとコードの書式設定なども含まれます。

IDE の使用方法を学ぶためには、「[IDE AWS Cloud9 のツアー](#)」を参照してください。

インスタンス上のコードを変更する方法については、以下のリソースをお勧めします。

- All [Lightsail ウェブサイトの「powered by Bitnami」 Lightsail イメージ](#)のアプリケーションのパスワードの取得
- Drupal: Bitnami ウェブサイトの [BitnamiDrupal For AWS クラウド](#)、および Drupal ウェブサイトの[チュートリアルとサイトレシピ](#)
- GitLab CE: Bitnami ウェブサイトの [BitnamiGitLab CE for AWS クラウド](#)、および GitLab ウェブサイトの[GitLab ドキュメンテーション](#)
- うJoomla: Bitnami ウェブサイトの [BitnamiJoomla! For AWS クラウド](#)、および Joomla! ウェブサイトでの [Joomla! の使用開始](#)
- LAMP スタック: Bitnami ウェブサイトの [BitnamiLAMP for AWS クラウド](#)

- Magento: Bitnami ウェブサイトの [BitnamiMagento For AWS クラウド](#)、および Magento ウェブサイトの [Magento ユーザーガイド](#)
- MEAN: Bitnami ウェブサイトの [BitnamiMEAN For AWS クラウド](#)
- NGINX: Bitnami ウェブサイトの [BitnamiNGINX For AWS クラウド](#)、および NGINX ウェブサイトでの [NGINXWiki](#)
- Node.js: Bitnami ウェブサイトの [BitnamiNode.Js For AWS クラウド](#)、および Node.js ウェブサイトの [使用開始ガイド](#)
- Ubuntu の Plesk ホスティングスタック: [Amazon Lightsail での Plesk のセットアップと設定](#)。
- Redmine: Bitnami ウェブサイトの [Bitnami Redmine For AWS クラウド](#)、および Redmine ウェブサイトの [使用開始](#)
- WordPress: [Lightsail ウェブサイトの Amazon Lightsail インスタンス、および Bitnami ウェブサイトの「Bitnami WordPress For AWS クラウド」から WordPress の使用を開始する](#)

IDE AWS CodePipeline での AWS Cloud9 の使用

AWS Cloud9 統合開発環境 (IDE) を使用して、と互換性のあるリポジトリでソースコードを操作できます AWS CodePipeline。

CodePipeline は、ソフトウェアおよびそれに対する継続的な変更をリリースするために必要なステップのモデル化、視覚化、およびオートメーションに使用できる継続的な配信サービスです。CodePipeline を使用して、ソフトウェアリリースプロセスのさまざまなステージをすばやくモデル化して設定できます。詳細については、「[AWS CodePipeline ユーザーガイド](#)」を参照してください。

Note

これらの手順を完了すると、AWS アカウントに料金が発生する可能性があります。これには、Amazon EC2、CodePipeline、Amazon S3 などのサービス、および CodePipeline でサポートされている AWS サービスに対して発生する可能性のある料金が含まれます。詳細については、「[Amazon EC2 料金表](#)」、「[AWS CodePipeline 料金表](#)」、「[Amazon S3 料金表](#)」、および「[クラウドサービス料金表](#)」を参照してください。

- [ステップ 1: ソースコードリポジトリを作成または識別する](#)
- [ステップ 2: AWS Cloud9 開発環境を作成し、それをコードリポジトリに接続して、コードをアップロードする](#)

- [ステップ 3: 使用する準備をする AWS CodePipeline](#)
- [ステップ 4: でパイプラインを作成する AWS CodePipeline](#)

ステップ 1: ソースコードリポジトリを作成または識別する

このステップでは、CodePipeline と互換性があるソースコードリポジトリを作成または識別します。

このトピックの後半では、ソフトウェアのソースコードをそのリポジトリにアップロードします。CodePipeline は、同様に作成した関連パイプラインを使用して、そのリポジトリにアップロードされたソースコードを構築、テスト、デプロイします。

ソースコードリポジトリは、CodePipeline がサポートする次のリポジトリタイプの一つである必要があります。

- AWS CodeCommit。CodeCommit 内に使用したいリポジトリが既に存在する場合は、「[ステップ 2: AWS Cloud9 開発環境を作成し、それをコードリポジトリに接続して、コードをアップロードする](#)」に進んでください。それ以外の場合、CodeCommit を使用するには、この順で、AWS CodeCommit のサンプルの指示を守ってから、このトピックに戻ります。
 - [ステップ 1: 必要なアクセス権限を持つ IAM グループをセットアップする](#)
 - [ステップ 2: AWS CodeCommit にレポジトリを作成する](#)
- Amazon S3。使用するバケットが Amazon S3 にすでにある場合は、[ステップ 2: AWS Cloud9 開発環境の作成、コードリポジトリへの接続、コードのアップロードに進んでください](#)。それ以外の場合、Amazon S3 を使用するには、Amazon Simple Storage Service ユーザーガイドの手順をこの順序で実行してから、このトピックに戻ります。
 - [Amazon S3 へのサインアップ](#)
 - [バケットの作成](#)
- GitHub。GitHub に既にリポジトリがある場合は、そのリポジトリをクローンし、開発環境に[Git パネル](#)インターフェイスを使って、ローカルコピーを作成します。GitHub でアカウントやリポジトリをまだセットアップしていない場合は、手順については、[関連資料](#)を参照してください。

ステップ 2: AWS Cloud9 開発環境を作成し、コードリポジトリに接続してコードをアップロードする

このステップでは、AWS Cloud9 コンソールで AWS Cloud9 開発環境を作成します。次に CodePipeline が使用するリポジトリに環境を接続します。最後に、環境の AWS Cloud9 IDE を使用してソースコードをリポジトリにアップロードします。

環境を作成するには、「[環境を作成する](#)」の指示に従ってから、このトピックに戻ります。(環境がすでに作成されている場合は、それを使用することができます。新しく作成する必要はありません。)

環境をリポジトリに接続し、その後ソースコードをリポジトリにアップロードしていない場合にアップロードするには、次の一連の手順のいずれかを使用します。選択するセットはソースコードを保存するリポジトリのタイプによって異なります。

[Repository type]	指示
CodeCommit	「AWS CodeCommit サンプル」の指示に従います。 • ステップ 3: 環境をリモートリポジトリに接続する • ステップ 4: リモートリポジトリのクローンを環境に作成する • ステップ 5: ファイルをリポジトリに追加する (このステップのソースコードの代わりに各自のものを使用します)
Amazon S3	• 「」および「サンプル AWS CloudShell」で説明されているように、環境に AWS CLI または をインストールして設定します。 AWS CLIAWS CloudShell • ソースコードをバケットにアップロードするには、環境 AWS CloudShell の AWS CLI または を使用して aws s3 cp コマンドを実行します。(の場合は AWS CloudShell、コマンドawsから を削除できます)。

[Repository type]	指示
GitHub	GitHub でホストされているリポジトリをクローンし、 Git パネル インターフェイスを使って操作できます。

環境をリポジトリに接続すると、AWS Cloud9 IDE からリポジトリにソースコードの変更をプッシュするたびに、CodePipeline はビルド、テスト、デプロイする関連パイプラインを介してそれらの変更を自動的に送信します。関連パイプラインはこのトピックの後半で作成します。

ステップ 3: を使用する準備をする AWS CodePipeline

このステップでは、[Team Setup](#) で作成または識別した IAM グループに特定の AWS 管理ポリシーをアタッチします。これにより、グループのユーザーによる CodePipeline でのパイプラインの作成および使用が有効になります。

前に CodePipeline を使用したことがある場合は、[ステップ 4: AWS CodePipeline でパイプラインを作成する](#)に進んでください。

このステップでは、「AWS CodePipeline ユーザーガイド」の「[ステップ 3: IAM 管理ポリシーを使用して割り当てる AWS CodePipeline](#)」の手順に従って、このトピックに戻ります。

ステップ 4: でパイプラインを作成する AWS CodePipeline

このステップでは、このトピックの前半で作成または識別したリポジトリを使用する CodePipeline でパイプラインを作成します。

このステップでは、AWS CodePipeline ユーザーガイドの「[AWS CodePipeline でパイプラインを作成する A](#)」の指示に従います。

パイプラインが作成された後、CodePipeline はリポジトリのソースコードの現行バージョンを、構築、テスト、デプロイするため、パイプラインを通して送信します。次に、ソースコードの変更を IDE AWS Cloud9 からリポジトリにプッシュするたびに、CodePipeline はそれらの変更をパイプラインを介して自動的に送信し、構築、テスト、デプロイします。

パイプラインを表示するには、AWS CodePipeline ユーザーガイド における [AWS CodePipeline でパイプラインの詳細と履歴](#) の指示 A に従います。

Amazon CodeCatalyst の使用

Amazon CodeCatalyst は、ソフトウェア開発チーム向けのクラウドベースのコラボレーションスペースです。CodeCatalyst では、継続的インテグレーション/デリバリー (CI/CD) ツールを使用して、コードでの作業やコラボレーション、アプリケーションの構築、テスト、デプロイを 1 か所で行うことができます。を CodeCatalyst スペースに接続することで、AWS リソース AWS アカウントをプロジェクトに接続できます。また、CodeCatalyst を使用すると、ソフトウェアを迅速かつ確実に配信できます。CodeCatalyst の詳細については、Amazon CodeCatalyst ガイドの「[What is Amazon CodeCatalyst?](#)」(Amazon CodeCatalyst とは) を参照してください。

開発環境は、プロジェクトのソースリポジトリに保存されているコードを操作するために CodeCatalyst で使用できるクラウドベースの開発環境です。CodeCatalyst で開発環境を作成できます。その間に、サポートされている統合開発環境 (IDE) を使用して、プロジェクト固有の CodeCatalyst でコードに取り組むことができます。または、空の開発環境を作成して、サードパーティのリポジトリからコードを複製し、サポートされている IDE で作業することもできます。

CodeCatalyst コンソールで開発環境にアクセスするために使用される AWS Cloud9 IDE は、で実行される AWS Cloud9 IDE とは異なります AWS。CodeCatalyst AWS Cloud9 IDE では、CodeCatalyst に自動的にログインし、IDE 内の aws-explorer オプションを使用してサービスにアクセスできます。AWS Toolkit の詳細については、AWS Cloud9 ガイドの [AWS Toolkit for AWS Cloud9](#) を参照してください。

トピック

- [での Amazon CodeCatalyst の開始方法 AWS Cloud9](#)
- [から AWS Cloud9 Amazon CodeCatalyst への移行](#)
- [レプリケーションツールを使用する](#)
- [レプリケーションプロセスに関するよくある質問](#)
- [Amazon CodeCatalyst の開発環境](#)

での Amazon CodeCatalyst の開始方法 AWS Cloud9

このセクションでは、CodeCatalyst の使用を開始する方法の概要を説明します。このセクションのトピックでは、Amazon CodeCatalyst AWS Cloud9 で を使用する方法と CodeCatalyst で環境を AWS Cloud9 レプリケートする方法について説明します。後のトピックでは、CodeCatalyst 開発環境を作成する方法と、IDE AWS Cloud9 を使用して開発環境にアクセスする方法についても説明します。

AWS ツールキットは、AWS クラウド アカウント、サービス、リソースへの迅速なアクセスを提供する IDE 固有のソフトウェア開発キット (SDKs) です。AWS Toolkit で CodeCatalyst アカウントから、CodeCatalyst 開発環境、スペース、およびプロジェクトを便利なインターフェイスで表示、編集、管理できます。AWS Toolkits で利用できる AWS クラウド サービスと機能の詳細については、「[とは AWS Toolkit for Visual Studio Code](#)」、「[AWS Toolkit for AWS Cloud9](#)」、「[とは AWS Toolkit for JetBrains](#)」を参照してください。ガイド [とは AWS Toolkit for JetBrains](#)。

IDE で CodeCatalyst AWS Cloud9 を使用するには、CodeCatalyst コンソール内で作成した既存のスペース、プロジェクト、開発環境が必要です。

Note

CodeCatalyst の IDE AWS Cloud9 のファイルシステム内で、同じ名前のフォルダ内にプロジェクトという名前のサブフォルダを作成しないでください。このようにすると、このディレクトリ内のどのファイルにもアクセスできなくなります。この問題は、`/projects/projects` のファイルパスに影響します。`/test/projects` や `/projects/test/projects` などのファイルパスは、この問題の影響を受けません。これは既知の問題であり、IDE File Explorer AWS Cloud9 にのみ影響します。

Note

現在、CodeCatalyst の IDE AWS Cloud9 のファイルシステムを使用して、同じ名前のフォルダ内にプロジェクトという名前のサブフォルダを作成することはできません。AWS Cloud9 IDE File Explorer からこのディレクトリ内のファイルにアクセスすることはできませんが、コマンドラインを使用してアクセスできます。別のフォルダ名を使用してください。この問題は `/projects/projects` のファイルパスにのみ影響します。`/test/projects` や `/projects/test/projects` などのファイルパスは正常に機能します。これは既知の問題であり、IDE File Explorer AWS Cloud9 にのみ影響します。

から AWS Cloud9 Amazon CodeCatalyst への移行

AWS Cloud9 CodeCatalyst のは、とやり取りするためのフルマネージドエクスペリエンスを提供します AWS Cloud9。Amazon CodeCatalyst で現在の AWS Cloud9 コードリソースを手動でレプリケートできます。以下のセクションでは、このプロセスを詳しく説明します。コードリソースを移動して複製するには、CodeCatalyst 内にスペースを作成します。スペースは、会社、部門、またはグ

ループを表します。CodeCatalyst で作成するプロジェクト、メンバー、および関連するクラウドリソースを追加するためのスペースを作成する必要があります。ユーザーがプロジェクトへの招待を受け入れると、CodeCatalyst は自動的にユーザーをスペースに追加します。Space administrator ロールを持つユーザーは、スペースを管理できます。

このスペース内にプロジェクトを作成し、ソースリポジトリを追加します。プロジェクトは、開発チームとタスクをサポートする CodeCatalyst のコラボレーションスペースです。プロジェクトを作成したら、リソースを追加、更新、または削除できます。プロジェクトダッシュボードをカスタマイズして、チームの作業の進捗状況を監視することもできます。1つのスペースに複数のプロジェクトを含めることができます。追加するソースリポジトリの数は、AWS Cloud9 環境で既に使用しているリポジトリの数によって異なります。このプロジェクトを作成し、該当するソースリポジトリを追加したら、AWS Cloud9 環境に戻って、環境データを CodeCatalyst の新しいリポジトリにレプリケートする必要がある場合があります。何をするかは、AWS Cloud9 にあるソースリポジトリのタイプによって異なります。

スペース、プロジェクト、ソースリポジトリを作成したら、開発環境で を使用して CodeCatalyst AWS Cloud9 で環境を起動できます。開発環境はクラウドベースの開発環境です。CodeCatalyst で開発環境を使用して、プロジェクトのソースリポジトリに保存されているコードを操作できます。CodeCatalyst で開発環境を作成して、サポートされている統合開発環境 (IDE) を備えたプロジェクト固有の開発環境でコードを処理することもできます。

レプリケーションツールを使用して、現在の AWS Cloud9 コードリソースを CodeCatalyst にレプリケートすることもできます。これは、環境でダウンロードして実行するツールです AWS Cloud9 。既に CodeCatalyst にサインアップしてスペースを作成した場合、ツールは自動的にこのスペース内にプロジェクトを作成し、コードリソースを CodeCatalyst の新しいリポジトリに複製します。手動レプリケーションプロセスと同様です。これは、AWS Cloud9 にあるソースリポジトリのタイプによって異なります。例えば、GitHub リポジトリがある場合でも、CodeCatalyst コンソールの [GitHub 拡張機能] を使用して、これらのリポジトリを複製する必要があります。

- [ステップ 1. Amazon CodeCatalyst にサインアップしてスペースを作成する](#)
- [ステップ 2. スペースにプロジェクトを作成する](#)
- [ステップ 3. プロジェクトにソースリポジトリを作成する](#)
- [Step 4. CodeCatalyst のソースリポジトリに AWS Cloud9 コードリソースを複製する](#)
- [ステップ 5. を使用して CodeCatalyst で開発環境を作成する AWS Cloud9](#)

ステップ 1. Amazon CodeCatalyst にサインアップしてスペースを作成する

既存のスペースやプロジェクトへの招待を受けなくても、Amazon CodeCatalyst にサインアップできます。サインアップすると、スペースとプロジェクトが作成されます。使用した既存の AWS アカウント ID を入力できます AWS Cloud9。これは請求目的で AWS アカウント も使用できます。AWS アカウント ID を検索する方法については、[AWS アカウント「ID とそのエイリアス」](#)を参照してください。以下の手順に従って、Amazon CodeCatalyst プロファイルにサインアップし、スペースを作成し、スペースのアカウントを追加します。

新規ユーザーとしてサインアップするには

1. [CodeCatalyst コンソール](#)を開きます。
2. [Welcome] ページで、[サインアップ] を選択します。

AWS ビルダー ID の作成ページが表示されます。AWS ビルダー ID は、サインインするために作成する ID です。この ID は AWS アカウント ID と同じではありません。AWS ビルダー ID の詳細については、AWS「サインインユーザーガイド」の[AWS「ビルダー ID およびその他の AWS 認証情報」](#)を参照してください。

3. [メールアドレス] に、CodeCatalyst に関連付けるメールアドレスを入力します。その後、[Next] を選択します。
4. 名前に、AWS ビルダー ID を使用するアプリケーションに表示する名前と姓を入力します。

この名前は AWS Builder ID プロファイル名です。必要に応じて、後で名前を変更できます。

[次へ] を選択します。[E メール確認] ページが表示されます。指定した E メールアドレスに確認コードが送信されます。

5. [確認コード] に、受信したコードを入力し、[確認] を選択します。

5 分経ってもコードが届かず、スパムまたは迷惑メールフォルダーにもコードが見つからない場合は、[コードを再送信] を選択します。

6. コードが確認されたら、パスワードを入力して、[パスワードの確認] を選択します。

カスタマーアグリーメントと AWS サービス条件を読み、同意 AWS したことを確認するチェックボックスを選択し、プロファイルの作成を選択します。

7. [エイリアスの作成] ページで、CodeCatalyst で使用するエイリアスを入力します。他の CodeCatalyst ユーザーは、コメントやプルリクエストであなたを @mention するために、このエイリアスを使用します。CodeCatalyst プロファイルには、AWS ビルダー ID のフルネームと

CodeCatalyst エイリアスの両方が含まれます。CodeCatalyst エイリアスを変更することはできません。

フルネームとエイリアスは、CodeCatalyst のさまざまな領域に表示されます。例えば、アクティビティフィードにはプロフィール名が表示されますが、プロジェクトメンバーはあなたのエイリアスを使用してあなたを @mention します。

[エイリアスの作成] を選択します。ページが更新され、[スペースの作成] セクションが表示されます。

8. [スペース名] にスペースの名前を入力し、[次へ] を選択します。

この名前を変更することはできません。

9. [AWS アカウント ID] には、スペースに接続するアカウントの 12 桁の ID をリンクします。

[AWS アカウント 検証トークン] に、生成されたトークン ID をコピーします。トークンは自動的にコピーされます。ただし、AWS 接続リクエストの承認中に保存することもできます。

10. Verify in AWS を選択します。

11. [Amazon CodeCatalyst スペースの確認] ページが AWS マネジメントコンソール で開きます。

これは [Amazon CodeCatalyst スペース] ページです。ページにアクセスするには、ログインが必要な場合があります。

このページにアクセスするには、[AWS マネジメントコンソール](#) にある Amazon CodeCatalyst スペースにサインインします。

の検証トークンフィールド AWS マネジメントコンソール には、CodeCatalyst で生成されたトークンが自動的に入力されます。

12. [スペースを確認] を選択します。

アカウント確認成功メッセージが表示され、アカウントがスペースに追加されたことを示します。

デフォルトでは、CodeCatalyst の無料利用枠を使用します。変更したい場合は、[このスペースについてスタンダードティアを有効にするか、IAM ロールを追加するには、スペースの詳細を表示します] を選択します。

CodeCatalyst の料金レベルの詳細については、「[Amazon CodeCatalyst - 料金表](#)」を参照してください。

[CodeCatalyst スペースの詳細] ページが AWS マネジメントコンソール で開きます。これは [Amazon CodeCatalyst スペース] ページです。ページにアクセスするには、ログインが必要な場合があります。

13. [\[Amazon CodeCatalyst に移動\]](#) を選択します。
14. CodeCatalyst の作成ページで、[スペースの作成] を選択します。

スペースの作成中は、ステータスメッセージが表示されます。スペースが作成されると、CodeCatalyst はそのスペースのページを開きます。ビューのデフォルトは [プロジェクト] タブに設定されています。

 Note

アクセス許可エラーまたはバナーが表示された場合は、ページを更新してページをもう一度表示してみてください。

CodeCatalyst にサインアップしてスペースを作成したら、レプリケーションプロセスの次のステップは、このスペース内にプロジェクトを作成することです。

ステップ 2. スペースにプロジェクトを作成する

以下のステップは、前のステップで作成したスペースに空のプロジェクトを作成する方法の概要です。このプロジェクトでは、後で必要なリソースを手動で追加できます。プロジェクトを作成する前に、Space administrator ロールを持っている必要があります。また、プロジェクトを作成するスペースに参加する必要があります。スペースを作成すると、CodeCatalyst は自動的に Space administrator ロールを割り当てます。Space administrator ロールは CodeCatalyst で最も強力なロールです。このロールとそのアクセス許可の詳細については、「[Space administrator ロール](#)」を参照してください。

空のプロジェクトを作成するには

1. プロジェクトを作成するスペースに移動します。
2. スペースダッシュボードで、[プロジェクトの作成] を選択します。
3. [最初から開始] を選択します。
4. [プロジェクトに名前を付ける] に、プロジェクトに割り当てる名前を入力します。名前はスペース内で一意でなければなりません。
5. [プロジェクトを作成] を選択します。

プロジェクトを作成したら、レプリケーションプロセスの次のステップは、1 つ以上のソースリポジトリを作成することです。

ステップ 3. プロジェクトにソースリポジトリを作成する

作成したプロジェクト内に、ソースリポジトリを作成する必要があります。このリポジトリには README.md ファイルという 1 つのファイルが含まれており、いつでも編集または削除できます。ソースリポジトリを作成したときに行った選択によっては、.gitignore ファイルも含まれる場合があります。

ソースリポジトリを作成するには

1. [CodeCatalyst コンソール](#)を開きます。
2. プロジェクトに移動します。
3. ナビゲーションペインで [コード] を選択してから、[ソースリポジトリ] を選択します。
4. [リポジトリの追加] を選択し、[リポジトリの作成] を選択します。
5. [リポジトリ名] で、リポジトリの名前を指定します。

リポジトリ名は、プロジェクト内で一意である必要があります。リポジトリ名の要件の詳細については、「[CodeCatalyst のソースリポジトリのクォータ](#)」を参照してください。

6. (オプション) [説明] に、プロジェクト内の他のユーザーがリポジトリの用途を理解しやすいように、リポジトリの説明を追加します。
7. (オプション) プッシュする予定のコードの種類に応じた .gitignore ファイルを追加します。
8. [作成] を選択します。

Note

CodeCatalyst は、ユーザーがリポジトリを作成するときに、README.md ファイルをリポジトリに追加します。また、CodeCatalyst は main という名前のデフォルトブランチにリポジトリの初期コミットを作成します。README.md ファイルは編集または削除できますが、デフォルトブランチを変更または削除することはできません。

9. ソースリポジトリのクローン URL と PAT を取得するには、[クローンリポジトリ] を選択します。
10. HTTPS クローン URL と PAT をそれぞれコピーするには、[コピー] を選択します。次に、クローン URL と PAT を、検索できる場所に保存します。

クローン URL と PAT はステップ 4 で使用され、CODECATALYST_SOURCE_REPO_CLONE_URL および CODECATALYST_PAT として参照されます。

プロジェクト内にソースリポジトリを作成したら、このソースリポジトリに AWS Cloud9 データを複製します。

ステップ 4. CodeCatalyst のソースリポジトリへの AWS Cloud9 コードリソースのレプリケート

AWS Cloud9 環境内のソースリポジトリのタイプによって、作成した CodeCatalyst ソースリポジトリにコードリソースを取得するためのレプリケーション方法が決まります。オプションは以下のとおりです:

- [でのGitHubリポジトリの使用 AWS Cloud9](#)
- [GitLab GitHubや Bitbucket などの 以外の リポジトリを で使用する AWS Cloud9](#)
- [AWS Cloud9 で空のリポジトリを使用する](#)。このオプションは、AWS Cloud9 でソースリポジトリを使用しないことを意味します。

CodeCatalyst で GitHub リポジトリを使用する

GitHub リポジトリ拡張機能を使用すると、Amazon CodeCatalyst プロジェクトの AWS Cloud9 からリンクされたGitHubリポジトリを使用できます。以下のステップは、CodeCatalyst カタログから GitHub 拡張機能をインストールする方法の概要です。また、これらのステップは、既存の GitHub アカウントを CodeCatalyst スペースに接続し、GitHub リポジトリを CodeCatalyst プロジェクトにリンクする方法も示しています。

この方法の最初のステップは、CodeCatalyst カタログから [GitHub リポジトリ] 拡張機能をインストールすることです。この拡張機能をインストールするには、以下のステップを実行します。

Important

[Github リポジトリ] 拡張機能のインストールと設定の一環として、拡張機能を GitHub アカウントにインストールする必要があります。そのためには、GitHub アカウント管理者と CodeCatalyst スペース管理者である必要があります。

ステップ 1. CodeCatalyst カタログから拡張機能をインストールするには

1. [CodeCatalyst コンソール](#)を開きます。
2. 自分のスペースに移動します。

Tip

複数のスペースに所属している場合は、表示するスペースを上部のナビゲーションバーで選択できます。

3. 検索バーの横にある上部のメニューバーの [カタログ] アイコンを選択して、CodeCatalyst カタログに移動します。[GitHub リポジトリ] を検索するか、カテゴリに基づいて拡張機能をフィルタリングできます。
4. (オプション) 関連するアクセス許可など、拡張機能の詳細を確認するには、[GitHub リポジトリ] 拡張名を選択します。
5. [インストール] を選択します。拡張機能に必要なアクセス許可を確認し、続行する場合は、[インストール] を再度選択します。

[GitHub リポジトリ] 拡張をインストールすると、[GitHub リポジトリ] 拡張機能の詳細ページが表示され、接続された GitHub アカウントとリンクされた GitHub リポジトリを表示および管理できます。

[GitHub リポジトリ] 拡張機能をインストールしたら、GitHub アカウントを CodeCatalyst スペースに接続します。GitHub アカウントを接続するには、以下のステップを実行します。

ステップ 2. GitHub アカウントを CodeCatalyst に接続するには

1. [接続済み Github アカウント] タブで、[GitHub アカウントを接続] を選択して、GitHub の外部サイトに移動します。
2. GitHub 認証情報を使用して GitHub アカウントにサインインし、Amazon CodeCatalyst をインストールするアカウントを選択します。
3. CodeCatalyst に現在および将来のすべてのリポジトリへのアクセスを許可するかどうかを選択します。または、CodeCatalyst で使用する特定の GitHub リポジトリを選択します。デフォルトのオプションは、GitHub スペース内のすべての GitHub リポジトリです。
4. CodeCatalyst に付与されているアクセス許可を確認してから、[インストール] を選択します。

GitHub アカウントを CodeCatalyst に接続すると、[GitHub リポジトリ] 拡張機能の詳細ページの [GitHub アカウント] タブで、接続されたアカウントを確認できます。

CodeCatalyst で GitHub リポジトリを使用するための最後のステップは、リポジトリを使用する CodeCatalyst プロジェクトにリポジトリをリンクすることです。GitHub リポジトリを CodeCatalyst プロジェクトにリンクするには、プロセス全体のステップ 3 で説明されている以下のステップを実行します。

ステップ 3. GitHub リポジトリ拡張機能の詳細ページから GitHub リポジトリを CodeCatalyst プロジェクトにリンクするには

1. [リンクされた GitHub リポジトリ] タブで、[GitHub リポジトリをリンク] を選択します。
2. [GitHub アカウント] として、リンクするリポジトリを含む GitHub アカウントを選択します。
3. [GitHub リポジトリ] として、CodeCatalyst プロジェクトにリンクするリポジトリを選択します。
4. [CodeCatalyst プロジェクト] として、GitHub リポジトリをリンクする CodeCatalyst プロジェクトを選択します。
5. [Link (リンク)] を選択します。

これで、CodeCatalyst リポジトリには、プッシュしたばかりの更新されたファイルとコミットが保存されているはずです。これで、このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。開発環境の詳細については、「[CodeCatalyst の開発環境](#)」を参照してください。

これで、このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。これを行う手順については、「[ステップ 5: CodeCatalyst AWS Cloud9 でを使用して開発環境を作成する](#)」で説明しています。

CodeCatalyst で GitHub 以外のリポジトリを使用する

以外のGitHubリポジトリ AWS Cloud9 を使用して環境をレプリケートする前に、Amazon CodeCatalyst で個人用アクセストークン (PAT) を作成する必要があります。次のセクションでは、このトークンの作成方法の概要を説明します。

Amazon CodeCatalyst で個人アクセストークンを作成する

プロジェクトで作成したソースリポジトリには、Git クライアントのあるローカルコンピューターまたは統合開発環境 (IDE) でアクセスできます。そのためには、アプリケーション固有のパスワードを入力する必要があります。この目的で使用する個人アクセストークン (PAT) を作成できます。作成

した個人アクセストークン (PAT) は、CodeCatalyst のすべてのスペースとプロジェクトのユーザー ID に関連付けられます。作成した PAT の名前と有効期限を表示したり、不要になった PAT を削除したりできます。PAT シークレットは、作成時にのみコピーできます。

個人アクセストークン (PAT) を作成するには

1. <https://codecatalyst.aws/> で CodeCatalyst コンソールを開きます。
2. 上部のメニューバーでプロファイルバッジを選択し、[My 設定] を選択します。

 Tip

ユーザープロフィールも確認できます。これを行うには、プロジェクトまたはスペースのメンバーページで、メンバーリストから自分の名前を選択します。

3. [個人アクセストークン] で [作成] を選択します。
4. [PAT 名] に、個人アクセストークン (PAT) のわかりやすい名前を入力します。
5. [有効期限] では、デフォルトの日付のままにしておくか、カレンダーアイコンを選択して、カスタムの日付を選択します。有効期限のデフォルトは、現在の日付から 1 年です。
6. [作成] を選択します。

 Tip

このトークンは、ソースリポジトリの [クローンリポジトリC] を選択したときにも作成できます。

7. PAT シークレットをコピーするには、[コピー] を選択します。PAT シークレットを、検索できる場所に保存します。

 Important

PAT シークレットは 1 回だけ表示されます。ウィンドウを閉じると、再表示できなくなります。PAT シークレットを安全な場所に保存しなかった場合は、別のシークレットを作成できます。

ソースリポジトリの PAT を作成したら、以下のセクションで説明するように、AWS Cloud9 環境にリモートリポジトリを追加し、このリポジトリにデータをプッシュして、AWS Cloud9 環境から CodeCatalyst にデータをレプリケートします。

AWS Cloud9 環境にリモートリポジトリを追加する

GitHub リポジトリではないリポジトリを実行しているとします。AWS Cloud9 環境にリモートリポジトリを追加し、CodeCatalyst のソースリポジトリにデータをプッシュできます。このプロセスを完了するには、以下のコマンドを実行します。

IDE AWS Cloud9 内から、CodeCatalyst のレプリケーションプロセスのステップ 3 で作成したソースリポジトリを指すリモートリポジトリを追加します。コマンドの CODECATALYST_SOURCE_REPO_CLONE_URL を、[ステップ 3 のステップ 10 で保存したクローン URL に置き換えます。プロジェクトにソースリポジトリを作成する](#)

```
git remote add codecatalyst CODECATALYST_SOURCE_REPO_CLONE_URL
```

次のコマンドを使用して、新しいブランチをソースリポジトリにプッシュします。パスワードの入力を求められたら、[ステップ 3 のステップ 10 で保存した CODECATALYST_PAT パスワードを使用します。プロジェクトにソースリポジトリを作成する](#)

```
git checkout -b replication && git push codecatalyst replication
```

以下は、このコマンドを実行したときの出力の例を示しています。

```
Switched to a new branch 'replication'
Password for 'https://[aws-account-id]@[aws-region].codecatalyst.aws/v1/MySpace222581768915/Replication/Repository':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 982 bytes | 122.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://[aws-account-id].codecatalyst.aws/v1/MySpace222581768915/Replication/Repository
* [new branch] replication # replication
```

このブランチは、CodeCatalyst で作成したソースリポジトリにあります。このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。開発環境の詳細については、「[CodeCatalyst の開発環境](#)」を参照してください。

これで、このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。これを行う手順については、「[ステップ 5: CodeCatalyst AWS Cloud9 でを使用して開発環境を作成する](#)」で説明しています。

で空のリポジトリを使用する AWS Cloud9

空のリポジトリ AWS Cloud9 を使用して から環境をレプリケートする前に、まず Amazon CodeCatalyst で個人用アクセストークン (PAT) を作成します。次のセクションでは、このトークンの作成方法の概要を説明します。

Amazon CodeCatalyst で個人アクセストークンを作成する

プロジェクトで作成したソースリポジトリには、Git クライアントのあるローカルコンピューターまたは統合開発環境 (IDE) でアクセスできます。そのためには、アプリケーション固有のパスワードを入力する必要があります。この目的で使用する個人アクセストークン (PAT) を作成できます。作成した個人アクセストークン (PAT) は、CodeCatalyst のすべてのスペースとプロジェクトのユーザー ID に関連付けられます。作成した PAT の名前と有効期限を表示したり、不要になった PAT を削除したりできます。PAT シークレットは、作成時にのみコピーできます。

個人アクセストークン (PAT) を作成するには

1. <https://codecatalyst.aws/> で CodeCatalyst コンソールを開きます。
2. 上部のメニューバーでプロファイルバッジを選択し、[My 設定] を選択します。

 Tip

ユーザープロファイルも確認できます。これを行うには、プロジェクトまたはスペースのメンバーページで、メンバーリストから自分の名前を選択します。

3. [個人アクセストークン] で [作成] を選択します。
4. [PAT 名] に、個人アクセストークン (PAT) のわかりやすい名前を入力します。
5. [有効期限] では、デフォルトの日付のままにしておくか、カレンダーアイコンを選択して、カスタムの日付を選択します。有効期限のデフォルトは、現在の日付から 1 年です。
6. [作成] を選択します。

 Tip

このトークンは、ソースリポジトリの [クローンリポジトリC] を選択したときにも作成できます。

7. PAT シークレットをコピーするには、[コピー] を選択します。PAT シークレットを、検索できる場所に保存します。

 Important

PAT シークレットは 1 回だけ表示されます。ウィンドウを閉じると、再表示できなくなります。PAT シークレットを安全な場所に保存しなかった場合は、別のシークレットを作成できます。

ソースリポジトリの PAT を作成したら、以下のセクションで説明するように、AWS Cloud9 環境で空のリポジトリを開始し、CodeCatalyst で作成したソースリポジトリをポイントして、AWS Cloud9 環境から CodeCatalyst にデータをレプリケートします。

で空のリポジトリを開始する AWS Cloud9

に設定されているソースリポジトリがない場合は AWS Cloud9、で空のリポジトリを開始します AWS Cloud9。さらに、CodeCatalyst で作成したソースリポジトリを指定し、Git を通じて複製するファイルを追加してプッシュします。以下のステップを実行し、次のコマンドを実行して AWS Cloud9、ファイルを CodeCatalyst にレプリケートします。

1. AWS Cloud9 環境から、次のコマンドを実行して空のリポジトリを開始します。

```
git init -b main
```

すると、次に示すような出力が表示されます。

```
Initialized empty Git repository in /home/ec2-user/environment/.git/
```

2. CodeCatalyst からソースリポジトリの URL をクローンします。CodeCatalyst コンソール内で作成した CodeCatalyst プロジェクトに移動し、ナビゲーションペインで [コード] を選択し、[ソースリポジトリ] を選択します。

3. ソースリポジトリのリストから必要なリポジトリを選択し、[クローンリポジトリ] を選択して、クローン URL をコピーします。
4. クローンした URL を使用して CodeCatalyst リポジトリを追加し、既に空のリポジトリにあるコンテンツを CodeCatalyst にプッシュします。

```
git remote add origin [...]  
git push origin --force
```

5. 複製するファイルを追加します。環境ディレクトリ内のすべてのファイルを複製する場合は、`git add -A` を実行します。

```
git add -A .  
git commit -m "replicate"
```

6. 関連性のない 2 つの履歴をマージします。マージコンフリクトが発生した場合の対処:

```
git merge origin/main --allow-unrelated-histories
```

7. 次のコマンドを実行して、変更を CodeCatalyst のソースリポジトリにプッシュバックします。パスワードの入力を求められたら、[ステップ 3 のステップ 10 で生成した個人アクセストークン \(CODECATALYST_PAT\)](#) を入力します。[プロジェクトにソースリポジトリを作成する](#)

```
Admin:~/environment (main) $ git push origin main  
Password for 'https://222581768915@git.us-west-2.codecatalyst.aws/v1/  
MySpace222581768915/Replication/Replication':
```

この手順を完了すると、CodeCatalyst リポジトリには、プッシュしたばかりの更新されたファイルとコミットが格納されます。これで、このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。そのためのステップは、次のセクションで概説されています。

ステップ 5: CodeCatalyst AWS Cloud9 で を使用して開発環境を作成する

次の手順では、AWS Cloud9 と先ほどレプリケートしたデータを使用して CodeCatalyst で開発環境を作成する方法の概要を説明します。

を使用して開発環境を作成するには AWS Cloud9

1. <https://codecatalyst.aws/> で CodeCatalyst コンソールを開きます。
2. 開発環境を作成するプロジェクトに移動します。

3. ナビゲーションペインで、[概要] を選択し、次に [My 開発環境] セクションに移動します。
4. [開発環境を作成] を選択します。
5. ドロップダウンメニュー AWS Cloud9 から を選択します。
6. [リポジトリのクローン] を選択します。

 Note

現在、CodeCatalyst はサードパーティのリポジトリのクローニングをサポートしていませんが、開発環境を作成して、選択した IDE からサードパーティのリポジトリをその中にクローンすることはできます。

7. 次のいずれかを行います。
 - a. クローンするリポジトリを選択し、[既存のブランチで作業する] を選択し、[既存のブランチ] ドロップダウンメニューからブランチを選択します。
 - b. クローンするリポジトリを選択し、[新しいブランチで作業する] を選択し、[ブランチ名] フィールドにブランチ名を入力し、[ブランチの作成元] ドロップダウンメニューから新しいブランチを作成するブランチを選択します。
8. オプションで、開発環境のエイリアスを追加します。
9. オプションで、[開発環境設定] 編集ボタンを選択して、開発環境のコンピューティング、ストレージ、またはタイムアウト設定を編集します。
10. [作成] を選択します。開発環境の作成中は、開発環境のステータス列に [開始中] と表示され、開発環境が作成されると、ステータス列に [実行中] と表示されます。

レプリケーションツールを使用する

AWS Cloud9 CodeCatalyst のは、とやり取りするためのフルマネージドエクスペリエンスを提供します AWS Cloud9。お客様が CodeCatalyst AWS Cloud9 で 使用できるように、レプリケーションツールを作成しました。AWS Cloud9 環境でスクリプトをコピーして実行したら、プロンプトに従ってスクリプトを実行し、コードリソースを から CodeCatalyst AWS Cloud9 にレプリケートします。レプリケーションツールとプロセスの詳細については、以下に概説する「[レプリケーションプロセスに関するよくある質問](#)」を参照してください。

Note

このレプリケーションプロセスは、既存の AWS Cloud9 環境には影響しません。レプリケーションプロセスが完了すると、開発環境、ソースリポジトリ、プロジェクト、スペースを削除でき、AWS Cloud9 環境には影響しません。このツールはコードリソースを CodeCatalyst AWS Cloud9 の にのみコピーし、既存の AWS Cloud9 環境を削除または設定しません。このレプリケーションツールは、最初に選択された AWS アカウントのグループにリリースされました。その結果、特定の AWS アカウントに表示されない場合があります。

Note

ツールをダウンロードする前に、Amazon CodeCatalyst にサインアップしてスペースを作成することをお勧めします。CodeCatalyst へのサインアップの詳細については、「[Amazon CodeCatalyst へのサインアップとスペースの作成](#)」を参照してください。

Amazon CodeCatalyst AWS Cloud9 で 使用する利点

次のセクションでは、CodeCatalyst AWS Cloud9 で 使用する際に得られるパフォーマンス上の利点と機能強化の概要を説明します。

- CodeCatalyst は、フルマネージド型の開発環境を使用して、ソフトウェア開発ライフサイクル全体を 1 か所から管理できる統合エクスペリエンスを提供します。
- 起動時の Amazon EBS ボリュームサイズのオプションが強化されました。
- エフェメラル環境をサポートし、開発環境のコンピューティングをオンデマンドでスケールできるようになりました。
- カスタムイメージの指定によりカスタム AMI がサポートされます。
- Devfile のサポートにより、設定をコードとして記述できます。

レプリケーションツールを使用して CodeCatalyst で AWS Cloud9 コードリソースをレプリケートする

次の手順では、レプリケーションツールをコピーして実行し、レプリケーションプロセスを完了する方法について詳しく説明します。

1. 以下のスクリプトをコピーして、必ず AWS Cloud9 環境内で実行してください。

```
curl https://dx5z5embsyrja.cloudfront.net -o /tmp/replicate-tool.tar.gz && tar
--no-same-owner --no-same-permissions -xvf /tmp/replicate-tool.tar.gz -C /tmp &&
node /tmp/cloud9-replication-tools
```

2. [オプション] レプリケーションツールはユーザーの AWS アカウント ID を使用してテレメトリを行います。これは、ツールの使用中に発生する可能性のある問題をより適切に特定できるようにすることを目的としています。tool starts、tool fails、tool is cancelled by user、tool completes successfully、および tool creates a Dev Environment for the user についてテレメトリイベントを送信します。レプリケーションツールでテレメトリを無効にする場合は、後述の「[レプリケーションツールのテレメトリを無効にする](#)」を参照してください。
3. AWS Cloud9 環境でレプリケーションツールをコピーして実行したら、ブラウザでアクセス URL に移動し、10 分以内に許可をクリックして、を AWS アカウント AWS ビルダー ID にリンクする必要があります。リンクは一度だけ開いてください。複数回開くとエラーが発生し、最初からやり直す必要があります。AWS Builder ID の詳細については、「[サインインユーザーガイド](#)」の AWS 「[Builder ID を使用したサインイン](#)」を参照してください。AWS これにより、CodeCatalyst でコードリソースを複製する目的でレプリケーションツールがコードリソースにアクセスできるようになります。
4. 使用するスペースを選択します。スペースが 1 つしかない場合は、そのスペースが選択されます。スペースの詳細については、『Amazon CodeCatalyst ユーザーガイド』の「[CodeCatalyst のスペース](#)」を参照してください。
5. CodeCatalyst でコードを複製するか、新しい開発環境で試すかを選択します。CodeCatalyst でコードを直接複製することをお勧めします。開発環境の詳細については、『Amazon CodeCatalyst ユーザーガイド』の「[CodeCatalyst の開発環境](#)」を参照してください。
6. プロジェクトの名前を入力するか、Enter キーを押して指定されたデフォルトの名前を使用します。
7. プロンプトが表示されたら、CodeCatalyst の新しいソースリポジトリにファイルをコピーする方法を選択します。ルートフォルダーを単一の CodeCatalyst リポジトリにプッシュするか、サブフォルダーを別の CodeCatalyst リポジトリにプッシュするかを選択できます。
8. ツールが完成したら、ターミナルメッセージに記載されている URL から CodeCatalyst コンソール内のプロジェクトに移動し、CodeCatalyst のコードリソースにアクセスします。

この手順を完了すると、CodeCatalyst リポジトリには、プッシュしたばかりの更新されたファイルとコミットが格納されます。これで、このブランチから開発環境を作成して、AWS Cloud9 で開くことができます。

レプリケーションツールのテレメトリを無効にする

以下のステップは、環境変数を設定してレプリケーションツールのテレメトリを無効にする方法の概要です。

1. AWS Cloud9 環境でターミナルを開く
2. 次のコマンドのいずれかを実行します。

```
export CLOUD9_REPLICATION_TOOL_TELEMETRY=off
```

または

```
export CLOUD9_REPLICATION_TOOL_TELEMETRY=0
```

3. 上記のコマンドのいずれかを実行すると、環境変数が設定され、レプリケーションツールのテレメトリが無効になります。テレメトリを無効にした後、レプリケーションツールのスクリプトをコピーして再実行し、プロセスを開始する必要があります。

レプリケーションツールのフィードバック

問題が発生した場合や、レプリケーションツールの使用経験についてフィードバックしたい場合は、サポートケースを作成して送信してください。サポートケースの作成については、「[サポートケースの作成とケース管理](#)」を参照してください。

AWS Cloud9 と Amazon CodeCatalyst の違い

次の表は、CodeCatalyst AWS Cloud9 での AWS Cloud9 との違いの一部を示しています。

AWS Cloud9	AWS Cloud9 Amazon CodeCatalyst での
プライベート VPC は非常にうまく機能します AWS Cloud9。	プライベート VPC の使用は、現在 CodeCatalyst の AWS Cloud9 ではサポートされていません。

AWS Cloud9	AWS Cloud9 Amazon CodeCatalyst での
AWS Cloud9 は、事前設定された AWS マネージド認証情報をサポートします。	CodeCatalyst AWS Cloud9 で の認証情報を手動で設定する必要があります。
30 分から 7 日間の間隔を設定し、 でシャットダウンを無効にすることができます AWS Cloud9。	CodeCatalyst では、 の間隔を 15 分から 20 時間 AWS Cloud9 に設定でき、シャットダウンを無効にすることはできません。
AWS Cloud9 は Ubuntu および AL2 OS プラットフォームをサポートしています。	AWS Cloud9 CodeCatalyst の は、Ubuntu と AL2 を含む MDE ユニバーサルイメージとカスタムイメージをサポートしています。この詳細については、「Amazon CodeCatalyst ユーザーガイド」の「 Universal devfile images 」を参照してください。
アップロードとダウンロードは でサポートされています AWS Cloud9	現在、CodeCatalyst AWS Cloud9 での のアップロードとダウンロードはサポートされていません。ユーザーは Amazon S3 バケットを使用してアップロードおよびダウンロードする必要があります。
コラボレーションは で利用できます。 AWS Cloud9	コラボレーションは現在、CodeCatalyst の AWS Cloud9 では使用できません。

レプリケーションプロセスに関するよくある質問

次のセクションでは、レプリケーションツールとレプリケーションプロセスに関するよくある質問のいくつかに回答します。

質問: CodeCatalyst で AWS Cloud9 環境をレプリケートした場合、AWS Cloud9 環境は影響を受けますか？

回答: いいえ、環境のレプリケーションは CodeCatalyst AWS Cloud9 のコードリソースのみを にコピーし、作業を継続できるようにします。のコードリソースと環境は、いかなる方法でも影響を受け AWS Cloud9 ません。

質問: ロールバックする場合、AWS Cloud9 環境は影響を受けますか？

回答: いいえ。CodeCatalyst 開発環境、ソースリポジトリ、プロジェクト、およびスペースを削除しても、AWS Cloud9 環境に影響はありません。

質問: 新しい拠点は HIPAA や SOC などの規格に準拠していますか？

回答: CodeCatalyst の開発環境は現在、これらの標準に準拠していません。これらの規格への準拠は今後予定されています。

質問: コードリソースはどこに行きますか？

回答: コードリソースは CodeCatalyst のプロジェクト内のソースリポジトリにコピーされます。

質問: 使用量は制限されますか？

回答: レプリケーションプロセスの一環として、無料利用枠内で 16 GB の開発環境を作成します。つまり、最大 4 つの開発環境を持つことができます。料金、ストレージ、および利用可能なさまざまな料金レベルの詳細については、「[Amazon CodeCatalyst - 料金表](#)」を参照してください。

質問: コンピューティングはどこに行きますか？

回答: 既存のコンピューティングに変更はありません。現状のままです。

質問: CodeCatalyst で既存の AWS アカウント認証情報を使用できますか？それらの認証情報は自動的に転送されますか？

回答: CodeCatalyst で AWS アカウント認証情報を手動で設定できます。自動的に転送されません。

質問: どれくらいの費用がかかりますか？

回答: CodeCatalyst は無料で使い始めることができます。料金、ストレージ、および利用可能なさまざまな料金レベルの詳細については、「[Amazon CodeCatalyst - 料金表](#)」を参照してください。

質問: CodeCatalyst のデータ複製プロセスとデータストレージは安全ですか？

回答: はい。コードリソースをコピーするには git push と https を使用します。CodeCatalyst はデータをサービス内に安全に保存します。すべてのデータは、転送時と保管時のいずれも暗号化されます。CodeCatalyst におけるデータ保護の詳細については、『Amazon CodeCatalyst ユーザーガイド』の「[Amazon CodeCatalyst におけるデータ保護](#)」を参照してください。

質問: どのレプリケーション方法を選択すればよいですか？

回答: レプリケーションツールには 2 つのアプローチがあります。コードリソース AWS Cloud9 から CodeCatalyst にコピーするには、単一の CodeCatalyst ソースリポジトリにプッシュするか、各サブフォルダを個別の CodeCatalyst ソースリポジトリに変換します。ソースリポジトリな

ど、CodeCatalyst の概念に関する予備知識を必要としないため、最初の方法を使用することをお勧めします。このアプローチは、AWS Cloud9 で使い慣れているのと同様のセットアップで作業しながら、CodeCatalyst での AWS Cloud9 の使い方を調べる良い出発点となります。

2 番目のオプションは、ルート AWS Cloud9 環境フォルダの下にあるサブフォルダを個別に使用する場合に最適です。このアプローチでは、ルートフォルダの下にあるファイルは複製されません。CodeCatalyst のソースリポジトリの詳細については、『Amazon CodeCatalyst ユーザーガイド』の「[CodeCatalyst のソースリポジトリ](#)」を参照してください。

質問: レプリケーションプロセスで生成される個人アクセストークンはどのようなもので、なぜ必要なのですか? 紛失した場合、再度生成することはできますか?

回答: 個人アクセストークンは CodeCatalyst のユーザー ID に関連付けられます。ローカルの変更を git で CodeCatalyst ソースリポジトリにプッシュするときのパスワードとして必要です。トークンとその生成方法の詳細については、『Amazon CodeCatalyst ユーザーガイド』の「[Amazon CodeCatalyst での個人アクセストークンの管理](#)」を参照してください。

質問: レプリケーションプロセス中にエラーが発生した場合はどうなりますか?

回答: レプリケーションツールの使用中にエラーが発生した場合は、まずツールをもう一度試してください。エラーがソースリポジトリに関するものである場合は、複製されたコードリソースを CodeCatalyst ソースリポジトリに手動でプッシュできます。ローカルリポジトリはすでに CodeCatalyst アップストリームで動作するように設定されているので、これですまいくははずです。問題が解決しない場合は、サポートケースを作成して送信してください。サポートケースの作成については、「[サポートケースの作成とケース管理](#)」を参照してください。

質問: my AWS BuilderID を使用してレプリケーションツールを認証し、アクセス許可を付与する必要があるのはなぜですか?

回答: レプリケーションプロセス中、レプリケーションツールが CodeCatalyst 内の複数のリソース (プロジェクト、開発環境、ソースリポジトリ) を読み書きし、ユーザーに代わってローカルコンテンツをコピーする必要があるため、これを行うユーザーのアクセス許可が必要です。

質問: CodeCatalyst に移行した場合、レイテンシーに変化はありますか?

回答: 実行するアクションによって は、レイテンシーが減少する場合があります。これは、CodeCatalyst サーバーが PDX リージョンでホストされているためです。

質問: インストール済みのソフトウェアはすべて引き継がれますか?

回答: いいえ、転送されるのはコードリソースだけです。バイナリ、設定、およびインストールされたソフトウェアは転送されません。

Amazon CodeCatalyst の開発環境

以下のセクションでは、IDE を使用して CodeCatalyst AWS Cloud9 で開発環境を作成および管理する方法の概要を説明します。

- [開発環境の作成](#)
- [開発環境設定を開く](#)
- [開発環境の再開](#)
- [開発環境の削除](#)
- [開発環境のリポジトリ devfile の編集](#)
- [リポジトリの複製](#)
- [開発環境のトラブルシューティング](#)

開発環境の作成

開発環境は複数の方法で作成できます。

- CodeCatalyst で [概要]、[開発環境]、または [ソースリポジトリ] の各ページから CodeCatalyst ソースリポジトリを使用して開発環境を作成します。
- 開発環境から CodeCatalyst のソースリポジトリに接続していない空の開発環境を作成します。
- 任意の IDE で開発環境を作成し、この開発環境内に CodeCatalyst ソースリポジトリを複製します。

ブランチおよびリポジトリごとに 1 つの開発環境を作成できます。プロジェクトは複数のリポジトリを持つことができます。開発環境は CodeCatalyst アカウントにのみ関連付けられ、CodeCatalyst アカウントでのみ管理できます。開発環境を開いて、サポートされている IDE のいずれかを使用して操作できます。特定の IDE を選択すると、その IDE でのみ開発環境を開くことができます。別の IDE を使用する場合は、ナビゲーションバーで開発環境を選択し、[編集] を選択するか、新しい開発環境を作成することで IDE を変更できます。デフォルトでは、開発環境は 2 コアプロセッサ、4 GB の RAM、16 GB の永続ストレージで作成されます。

CodeCatalyst で開発環境を作成する方法の詳細については、『Amazon CodeCatalyst ガイド』の「[開発環境の作成](#)」を参照してください。

CodeCatalyst で開発環境を作成するための詳細と手順については、「Amazon CodeCatalyst ユーザーガイド」の「[開発環境の作成](#)」を参照してください。

Note

サードパーティのソースリポジトリを使用して開発環境を作成できるようになりました。サードパーティのソースリポジトリを CodeCatalyst 内のプロジェクトにリンクする方法については、「Amazon CodeCatalyst ユーザーガイド」の「[ソースリポジトリのリンク](#)」を参照してください。

Important

開発環境は、CodeCatalyst 認証情報にアクセスできる devfile を介してスクリプトを実行できます。信頼できないリソースを開く前に、リポジトリを確認してください。

開発環境設定を開く

CodeCatalyst コンソールで開発環境を作成すると、特定の開発環境設定を表示できます。

1. CodeCatalyst コンソールから、AWS Cloud9 IDE を介して開発環境に移動します。
2. AWS Cloud9 サイドバーから [aws-explorer] を選択します。
3. [デベロッパーツール] ナビゲーションペインで、[CodeCatalyst] を展開し、[設定を開く] を選択して、[開発環境設定] ビューを開きます。
4. [開発環境設定] ビューで、以下のセクションに開発環境のオプションが表示されます。
 - [エイリアス:] 開発環境に割り当てられている [エイリアス] を表示および変更します。
 - [ステータス:] 現在の開発環境のステータス、割り当てられているプロジェクトを表示し、開発環境を停止します。
 - [Devfile:] 開発環境の Devfile の名前と場所を表示します。Devfile を開くには、[エディタで開く] を選択します。
 - Compute Settings (コンピューティング設定): 開発環境のサイズとデフォルトのタイムアウトまでの長さを変更します。

Note

開発環境を作成した後に、開発環境に割り当てたストレージ容量を変更することはできません。

 Note

ターミナルから Amazon CodeCatalyst AWS CLI を使用する場合は、CodeCatalyst コマンドを実行する前に、必ず `AWS_PROFILE=codecatalyst` を設定する必要があります。
CodeCatalyst

開発環境の再開

開発環境の `$HOME` ディレクトリのすべての内容は永続的に保存されます。必要に応じて開発環境での作業を停止し、後で開発環境での作業を再開できます。開発環境の作成時に [タイムアウト] フィールドで選択した時間を超えて開発環境のアイドル状態が続いたとします。この場合、セッションは自動的に停止します。

開発環境は CodeCatalyst からのみ再開できます。開発環境の再開の詳細については、『Amazon CodeCatalyst ガイド』の「[開発環境の再開](#)」を参照してください。

 Note

開発環境の再開には数分かかる場合があります。

開発環境の削除

開発環境に保存されているコンテンツを使い終わったら、そのコンテンツを削除できます。開発環境を削除する前に、必ずコードの変更をコミットして元のソースリポジトリにプッシュしてください。開発環境を削除すると、開発環境のコンピューティングとストレージの請求は停止されます。

開発環境は、CodeCatalyst の [Dev Environments] (開発環境) ページからのみ削除できます。開発環境の削除方法の詳細については、『Amazon CodeCatalyst ガイド』の「[開発環境の削除](#)」を参照してください。

開発環境のリポジトリ devfile の編集

開発環境の設定を変更するには、devfile を編集します。devfiles を使用して、チーム全体で開発環境を標準化できます。devfile は、CodeCatalyst のソースリポジトリのルートから編集できます。または、サポートされている IDE で devfile を編集することもできます。サポートされている IDE で devfile を編集する場合は、ソースリポジトリに変更をコミットしてプッシュするか、プルリクエストを作成します。これにより、チームメンバーは devfile の編集内容を確認して承認できます。

 Note

devfile にはパブリックコンテナイメージのみを含めることができます。

 Note

依存関係がない場合、一部の AWS Cloud9 IDE 機能はカスタム では機能しない場合があります。devfile。Linux x64 以外の一部のプラットフォームで動作させるには、追加の作業が必要になる場合があります。

で開発環境devfileのリポジトリを編集するには AWS Cloud9

1. CodeCatalyst コンソールから、AWS Cloud9 IDE を介して開発環境に移動します。
2. AWS Cloud9 サイドバーから、aws-explorer を選択します。
3. [デベロッパーツール] ナビゲーションペインで、[CodeCatalyst ツールキット] メニューを選択します。
4. [Open Devfile] (devfile を開く) を選択します。
5. devfile を編集し、ファイルを保存します。
6. メニューサイドバーから Git 拡張機能である [Source Control] を選択します。
7. [Message] (メッセージ) テキストフィールドに、変更をステージングする前のメッセージを入力します。
8. コミットの準備をするには、[Stage All Changes (+)] (すべての変更をステージング (+)) アイコンを選択します。
9. Git コマンドを表示するには、リポジトリ名の横にある [メニュー] アイコンを選択します。
10. [Commit] (コミット) と [Push] (プッシュ) を選択します。
11. AWS Toolkit メニューから、[開発環境の更新] を選択します。

[Commit] (コミット) と [Push] (プッシュ) を選択します。更新した devfile が保存され、変更がコミットされてプッシュされます。

Note

カスタム devfile を使用して起動したい開発環境が機能しないとしましょう。これは、devfile に AWS Cloud9 との互換性がないことが原因と考えられます。トラブルシューティングを行うには、devfile を参照してください。問題が解決しない場合は、削除して新しいものを作成してみてください。

開発環境の devfile は、CodeCatalyst で編集することもできます。詳細については、Amazon CodeCatalyst ガイドの「[Configuring your Dev Environment](#)」(開発環境の設定)を参照してください。

リポジトリの複製

ソースリポジトリ内の複数のファイル、ブランチ、コミットを効果的に操作するために、ソースリポジトリをローカルコンピューターにクローンできます。その場合、Git クライアントまたは IDE を使用して変更を加えます。CodeCatalyst から、他のGitホストプロバイダーと同じ方法で、またコマンドラインを使用して、IDE AWS Cloud9 Git拡張機能を使用できます。サードパーティのリポジトリをクローンする方法については、「[Git リポジトリを初期化またはクローンする](#)」を参照してください。

ソースリポジトリから開発環境を作成して CodeCatalyst でクローンする方法の詳細については、『Amazon CodeCatalyst ガイド』の「[ソースリポジトリの概念](#)」を参照してください。

開発環境のトラブルシューティング

開発環境で問題が発生した場合は、Amazon CodeCatalyst ガイドの「[Troubleshooting problems with Dev Environments](#)」(開発環境の問題のトラブルシューティング)を参照してください。

Note

ターミナルから Amazon CodeCatalyst AWS CLI を使用する場合は、CodeCatalyst コマンドを実行する前に、必ず `AWS_PROFILE=codecatalyst` を設定する必要があります。
CodeCatalyst

開発環境で問題が発生した場合は、Amazon CodeCatalyst ガイドの「[Troubleshooting problems with Dev Environments](#)」(開発環境の問題のトラブルシューティング)を参照してください。

IDE AWS CDK での AWS Cloud9 の使用

AWS CDK サービスを使用すると、[AWS Cloud Development Kit \(AWS CDK\)](#) アプリケーションまたはアプリを操作できます。の詳細については、AWS CDK [AWS Cloud Development Kit \(AWS CDK\) デベロッパーガイド](#)を参照してください。

AWS CDK アプリケーションは、[コンストラクト](#)と呼ばれる構成要素で構成されています。これらの構成要素には、CloudFormation スタックとその中の AWS リソースの定義が含まれます。AWS CDK Explorer を使用すると、AWS CDK ツリービューで定義されている[スタック](#)と[リソース](#)を確認できます。このビューには、AWS Cloud9 エディタ内の開発者ツールペインからアクセスできます。

このセクションでは、AWS Cloud9 エディタAWS CDK で にアクセスして使用方法について説明します。

AWS CDK アプリケーションの使用

AWS Cloud9 統合開発環境 (IDE) で AWS CDK Explorer を使用して、AWS CDK アプリケーションを視覚化して操作します。

前提条件

AWS CDK コマンドラインインターフェイスをインストールします。手順については、デベロAWS Cloud Development Kit (AWS CDK) ッパーガイドの[AWS 「CDK の開始方法」](#)を参照してください。

Important

インストールする AWS CDK バージョンは 1.17.0 以降である必要があります。実行中のバージョンは、**cdk --version** コマンドを使用して確認できます。

AWS CDK アプリケーションを視覚化する

IDE AWS CDK Explorer を使用すると、アプリケーションの CDK AWS Cloud9 コンストラクトに保存されている[スタック](#)と[リソース](#)を管理できます。AWS CDK Explorer は、tree.json ファイルで定義されている情報を使用して、リソースをツリービューに表示します。このファイルは、**cdk synth** コマンドを実行したときに作成されます。デフォルトでは、tree.json ファイルはアプリケーションの cdk.out ディレクトリにあります。

Toolkit AWS CDK Explorer の使用を開始するには、CDK アプリケーションを作成します。

1. [AWS CDK デベロッパーガイド](#)で、[Hello World のチュートリアル](#)の最初のいくつかのステップを完了します。

 Important

「スタックのデプロイ」ステップに達したら、操作を中止してこのガイドに戻ってください。

 Note

チュートリアルで提供されているコマンド (`mkdir`、`cdk init` など) は、オペレーティングシステムのコマンドラインインターフェイス、または VS Code エディタ内のターミナルウィンドウで実行できます。

2. CDK チュートリアルの必要なステップを完了したら、IDE エディタで作成した CDK AWS Cloud9 コンテンツを開きます。
3. AWS ナビゲーションペインで、CDK 見出しを展開します。CDK アプリケーションとその関連リソースが CDK Explorer のツリービューに表示されます。また、AWS Cloud9 内のターミナルで次のコマンドを実行して、CDK 機能が有効であることを確認できます。

```
mkdir mycdkapp
cd mycdkapp
cdk init app --language=typescript
cdk synth
cdk bootstrap
```

重要な注意事項

- CDK アプリケーションを AWS Cloud9 エディタにロードすると、複数のフォルダを一度にロードできます。各フォルダには、前のイメージに示すように、複数の CDK アプリを含めることができます。AWS CDK Explorer は、プロジェクトのルートディレクトリとその直接サブディレクトリでアプリケーションを検索します。

- チュートリアル最初のいくつかのステップを実行すると、最後に実行したコマンドが **cdk synth** であることに気付く場合があります。このコマンドは、AWS CDK アプリケーションを CFN に変換することで CloudFormation テンプレートを合成します。また、副産物として `tree.json` ファイルを生成します。CDK アプリケーションに変更を加えた場合は、**cdk synth** コマンドを再度実行して、変更がツリービューに反映されていることを確認します。変更の 1 つの例として、アプリケーションへのリソースの追加があります。

AWS CDK アプリケーションで他のオペレーションを実行する

AWS Cloud9 エディタを使用して、コマンドラインインターフェイスを使用するのと同じ方法で CDK アプリケーションで他のオペレーションを実行できます。たとえば、エディタでコードファイルを更新し、AWS Cloud9 ターミナルウィンドウを使用してアプリケーションをデプロイできます。

これらのタイプのアクションを試すには、AWS Cloud9 エディタを使用して、AWS CDK デベロッパーガイドの [Hello World チュートリアル](#) を続行します。最後のステップである「アプリケーションのリソースの破棄」を必ず実行してください。実行しないと、AWS アカウントで予想外のコストがかかる可能性があります。

Git パネルを使用したビジュアルソース管理

の Git パネル AWS Cloud9 は、重要な Git 機能を使用するための便利なビジュアルインターフェイスを提供します。

Git パネルインターフェイスのオプションを使用すると、リポジトリの初期化またはリモートリポジトリのクローン作成、ステージングエリアへのファイルの追加、作業ディレクトリにステージングされたファイルをコミット、上流のリポジトリへの変更のプッシュなど、完全なソース管理ライフサイクルを管理できます。

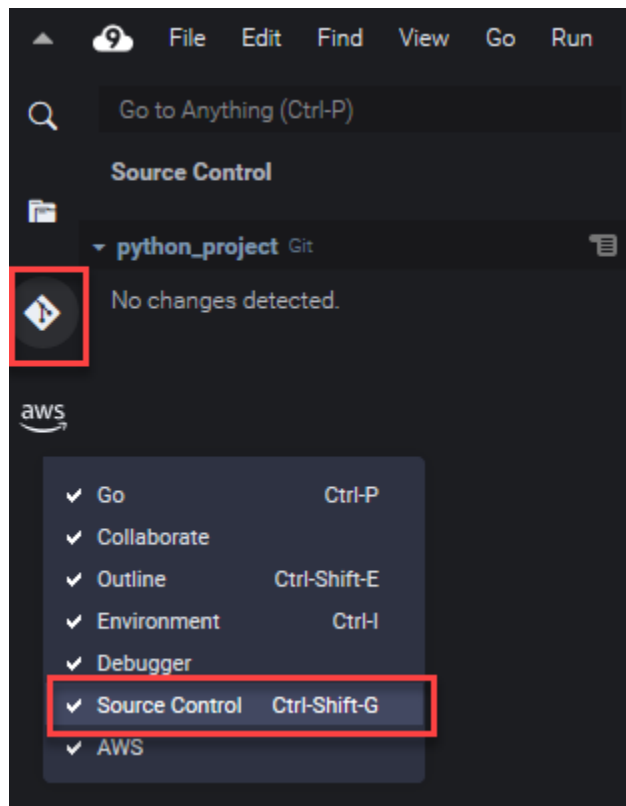
ブランチの作成やマージなど、Git のコアコラボレーションやプロジェクト管理の特徴は、Git パネルインターフェイスで数回クリックするだけですぐに実装できます。さらに、マージの競合は、IDE のエディタウインドウを使用して識別および解決できます。

Important

Git パネルは、Amazon EC2 インスタンスで作成された AWS Cloud9 環境でのみ使用できます。EC2 環境ではなく、[SSH 開発環境](#)を使用する場合、この機能は使用できません。さらに、Git パネルは、2020 年 12 月 11 日以降に作成された新しい AWS Cloud9 環境でのみデフォルトで使用できます。この日付よりも前に作成された開発環境の Git パネルの有効化に取り組んでいます。

インターフェイスにアクセスして操作するには、Window、ソースコントロールを選択します。または、IDE のサイドパネルのどこかを右クリックして、[Source Control] (ソースコントロール) を選択すると、ソース管理を取得できます。その後、IDE インターフェイスに表示されている Git アイコンを選択します。

キーの組み合わせ Ctrl-Shift-G は、Git パネルの表示を切り替えるためにも使用できます。



Note

Git パネルドキュメントのスクリーンショットは、Jett Dark テーマが適用された AWS Cloud9 IDE を示しています。IDE を別のテーマで使用している場合、インターフェース要素によっては表示方法が異なります。Git パネルを開くには、ソースコントロール Git アイコンの代わりに、ラベル付きのリンクを選択することができます。

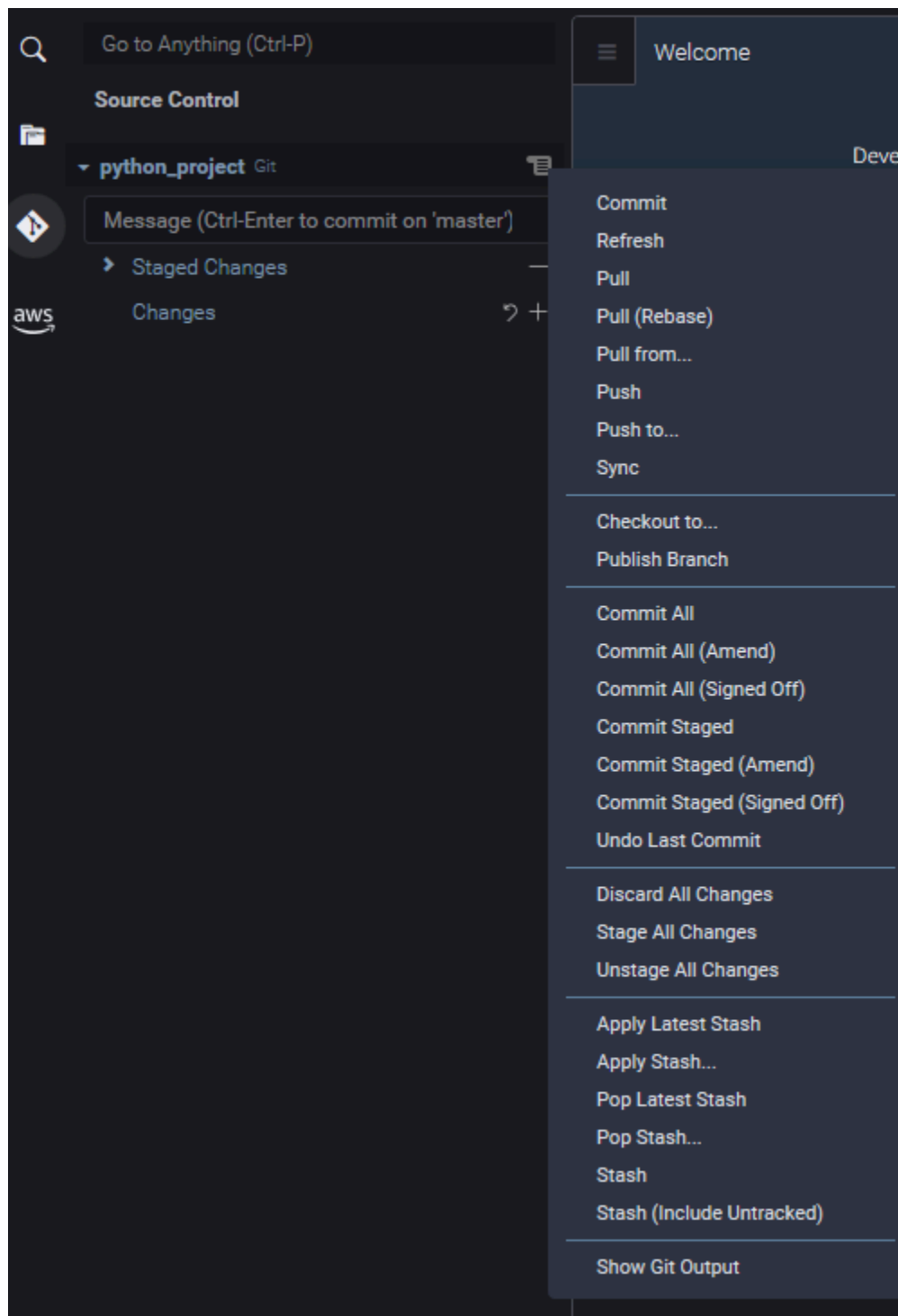
トピック

- [Git パネルを使用したソース管理の管理](#)
- [リファレンス: Git パネルで利用可能な Git コマンド](#)

Git パネルを使用したソース管理の管理

の Git パネル拡張機能 AWS Cloud9 は、コア Git コマンドとアドバンスド Git コマンドの両方への便利なユーザーインターフェイスアクセスを提供します。

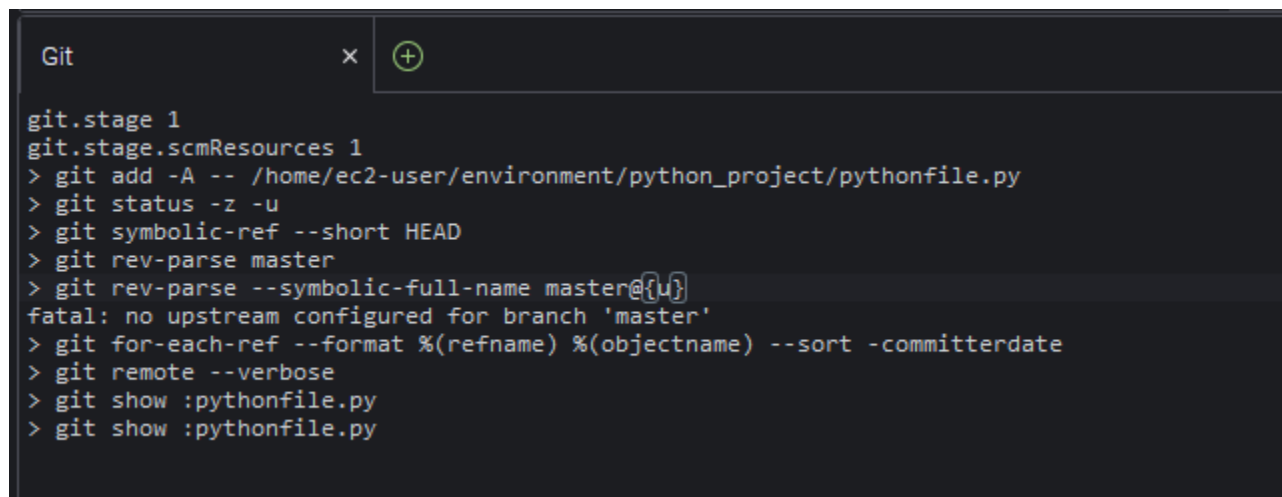
このセクションは、ソース管理のためにキーとなる Git の特徴にアクセスする方法を示します。この手順では、Git パネルメニューを使用して、リポジトリとそのコンテンツに対して Git コマンドを実行します。



Git パネルの検索ボックスに名前を入力することで、サポートされている Git コマンドにアクセスすることもできます。



また、Git パネルインターフェイスと対話するときに実行される実際の Git コマンドを表示できます。コマンドラインアクティビティを表示するには、Git パネルメニューに移動し、Git 出力を表示を選択します。

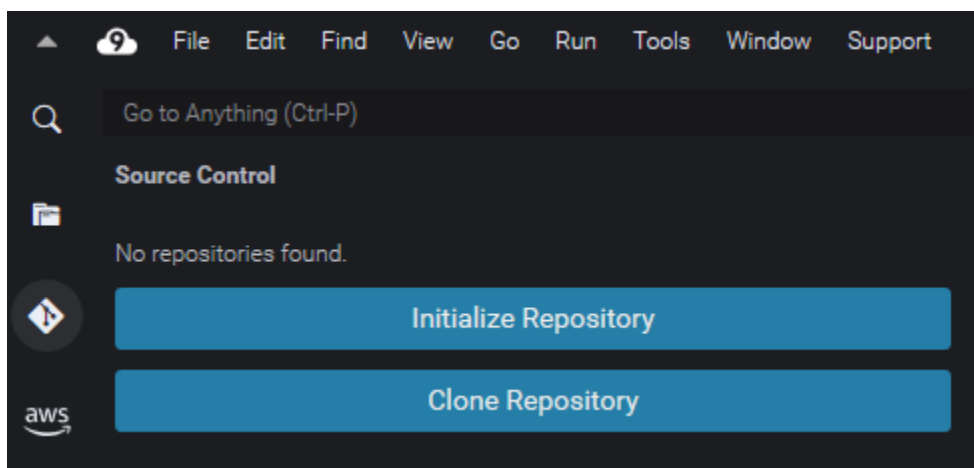


Git リポジトリを初期化またはクローン化するには

Gitリポジトリ(「repo」)には、プロジェクトの最初から完全な履歴が含まれています。リポジトリは、ステージングされたファイルをそのレポジトリにコミットするたびにキャプチャされたプロジェクトコンテンツのすべてのスナップショットで構成されます。

Gitパネルは、Gitリポジトリを取得する両方の方法をサポートしています。

- 既存のディレクトリを Git リポジトリとして初期化します。
- 既存のリポジトリをクローンし、ローカルディレクトリにコピーします。



Note

リポジトリの初期化およびクローン化のためのインタフェースオプションは、環境内のワークスペースフォルダに Git リポジトリをまだ追加していない場合にのみ使用できます。リポジトリの作業ディレクトリがすでに存在する場合、Git パネルウィンドウに作業ディレクトリのステータスとステージングエリアが表示されます。[Git パネル] メニューを使用して、リポジトリに対して実行できる Git コマンドにアクセスすることもできます。

リポジトリを初期化またはクローン化するには

1. Git パネルがまだ利用できない場合は、ウィンドウ、ソースコントロールをクリックし、Git アイコンを選択します。

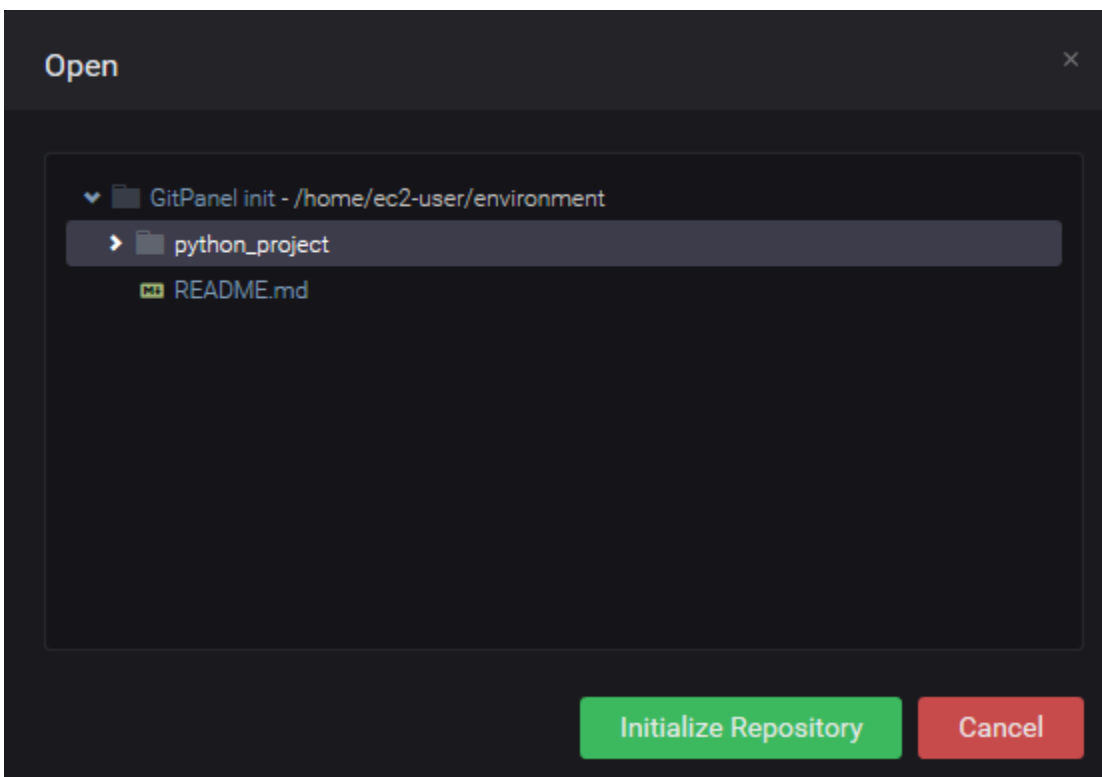
Note

キーボードショートカット `Ctrl+Shift+G` を使用してGitパネルを開くこともできます。

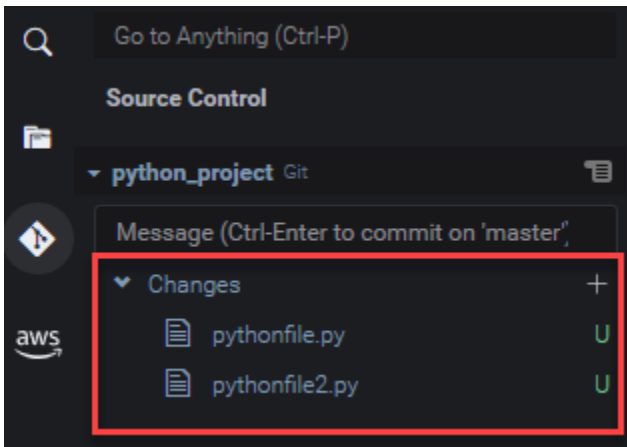
2. 新しい repo を初期化するか、既存の repo をクローンするかを選択します。

Initialize a repository

- Git パネルでリポジトリを初期化するを選択します。
- 次に、Git repo が初期化されるワークスペースフォルダを選択します。フォルダへのパスを入力したり、パスを選択したり、ダイアログボックスでフォルダを選択したりできます。
- ダイアログボックスを使用している場合は、移動先フォルダを選択し、リポジトリの初期化するを選択します。



選択したフォルダで Git repo を初期化すると、Git パネルには、そのフォルダに既に存在する任意のファイルが追跡されていない状態で表示され、Git ステージングエリアに追加できる状態になります。



Clone a repository

- Git パネルウィンドウでリポジトリのクローンを作成するを選択します。
- 次に、クローンを作りたいリモート repo の URL (GitHub でホストされている repo をクローンする `https://github.com/my-own-repo/my-repo-project-name.git` など) を入力し、戻るを押します。
- 表示されたダイアログボックスで、クローンされた repo のワークスペースフォルダを選択し、リポジトリの場所を選択を選択します。

Note

外部サイト (GitHub など) でホストされているリポジトリにアクセスする場合、プロセスを完了するには、サイトのサインイン認証情報も入力する必要があります。

選択したフォルダにリモート repo をクローンした後、`git pull` コマンドを使用して、ローカルリポジトリをリモートリポジトリ内の最新の変更と同期します。詳細については、[「リモートリポジトリを操作する」](#)を参照してください。

ファイルのステージングとコミット

Git リポジトリを取得した後、2 ステップのプロセスを使用してコンテンツを入力をスタートできます。

1. 追跡されていないコンテンツまたは最近変更されたコンテンツをステージングエリアに追加します。

2. ステージングエリアのファイルを作業ディレクトリにコミットします。

⚠ Important

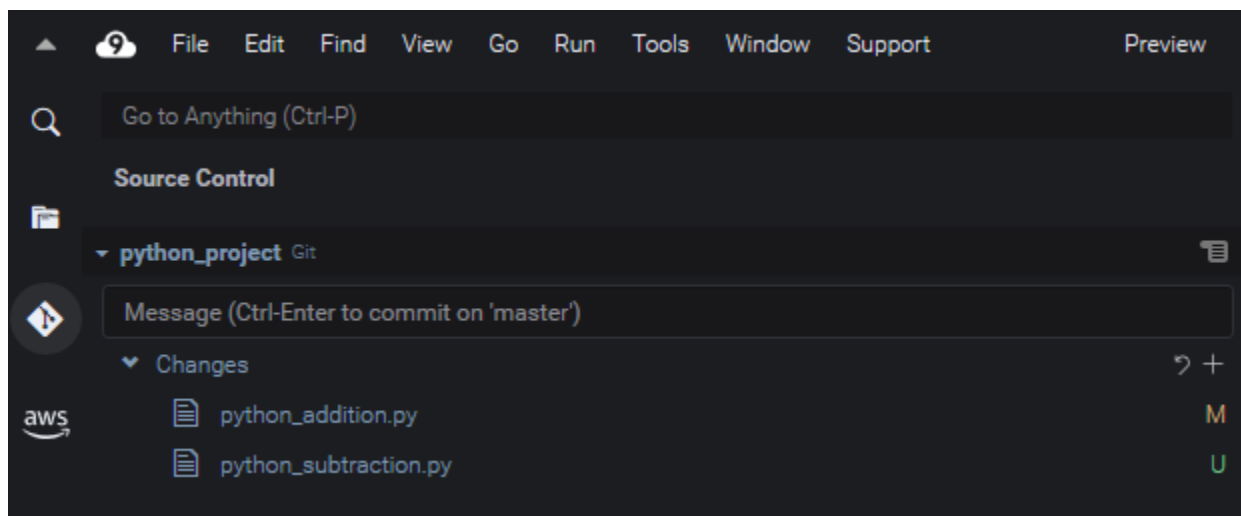
作業ディレクトリ内のすべてのファイルをリポジトリにコミットしたくない場合があります。たとえば、ランタイムに生成されたファイルをプロジェクトのリポジトリに追加したくなることはまずありません。Git パネルを使用すると、.gitignore ファイルでリストに追加して、ファイルが無視するようにマークできます。

.gitignore でリストを更新するには、ステージングエリアに追加されていないファイルを右クリックし、.gitignore にファイルを追加を選択します。IDE が .gitignore ファイルを開き、選択したファイルの名前が無視されるファイルリストに追加されます。

.gitignore でパターンマッチングを使ったファイルの種類を除外に関する情報については、関連するgit-scm.comサイトのリファレンスを参照してください。

Stage files

ステージングエリアに追加されていない未追跡ファイル (「U」というラベル) および変更されたファイル (「M」というラベル) は、Git パネルペインの [Changes] (変更) の下でリストされます。

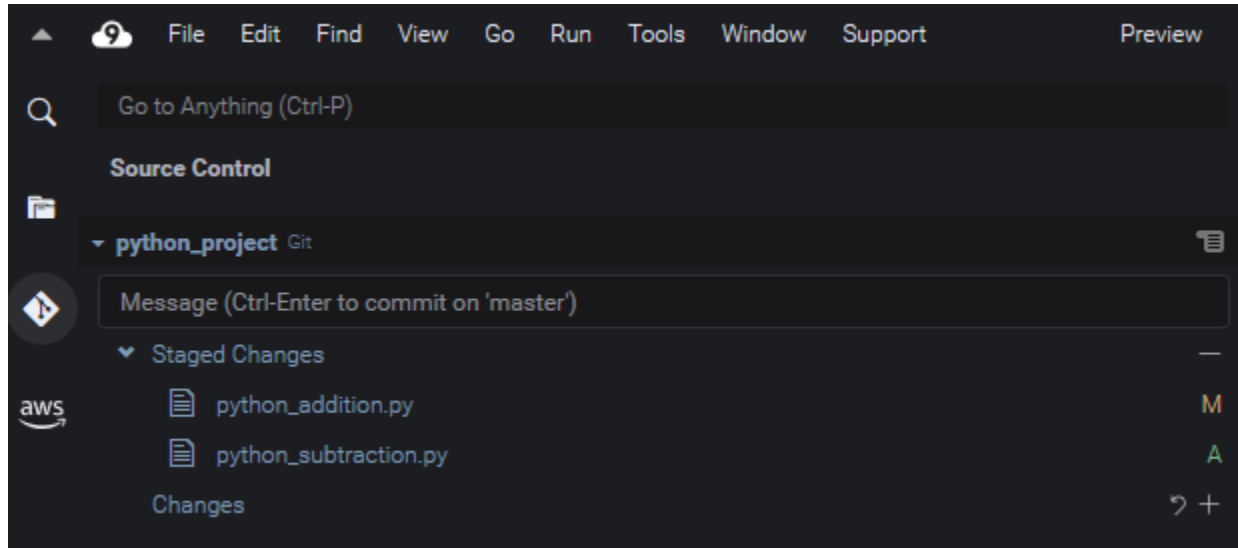


Git パネルインターフェイスを使用して、特定のファイル、またはすべての未追跡ファイルおよび変更されたファイルをステージングエリアに追加できます。

- 特定のファイル: ファイルを一時停止し、+を選択して、ステージングエリアに追加します。または、ファイルを右クリックしてステージの変更を選択します。

- すべてのファイル: Git パネルメニューに移動し、すべての変更をステージングを選択します。

リポジトリのインデックスに追加されたファイルは、ステージングされた変更の下でリストにされます。以前に追跡されていないファイルは「A」というラベルが付けられ、ステージングされたことが示されます。



Note

特定の変更またはすべての変更をステージング解除できます。1つのファイルの場合は、そのファイルを一時停止し、-を選択します。または、右クリックして変更をステージング解除を選択します。すべての変更をステージング解除するには、Git パネルメニューに移動し、すべての変更をステージング解除を選択します。

Commit files

Gitの `commit` コマンドを使用して、ステージングされたファイルをリポジトリ内の永続的なスナップショットとしてキャプチャします。Git パネルインターフェイスを使用して、コミットするファイルを選択できます。

- ステージングエリアでファイルをコミットする: Git パネルメニューに移動し、コミットを選択するか、ステージングされたファイルをコミットします。
- 作業ディレクトリ内のすべてのファイルをコミットする: [Git パネル] メニューに移動し、すべてコミットを選択します。(このオプションは `git commit` を呼び出す前に、ファイルをステージングエリアに追加するため、`git add` を使用します。)

Note

また、Gitパネルでファイルをコミットするときには、amend および signed-off オプションの使用も可能です。amend オプションは、最新のコミットのコミットメッセージを変更します。sign-off オプションは、Git ログでコミットを実行した人を特定できます。

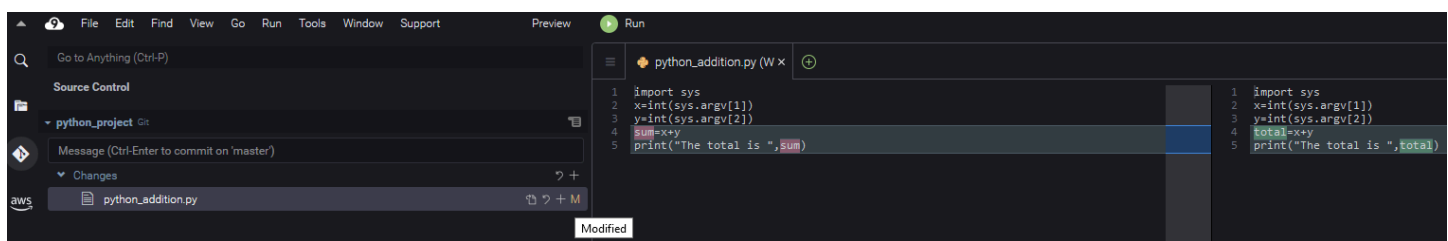
また、Git パネルメニューに移動して、コミットを選択し、前回のコミットを取り消しを選択して、コミットを反転させることもできます。

異なるファイルバージョンの表示

ステージングまたはコミット後に変更されたファイルのバージョンを比較できます。

- 変更の下でリストされたファイル: 「M」を選択すると、作業ディレクトリのバージョンと、最後にステージングまたは最後に repo をコミットしたバージョンの違いが表示されます。
- ステージングされた変更の下でリストされたファイル: ステージングエリアのバージョンと最後に repo をコミットしたバージョンの違いを表示するには、「M」を選択します。

「M」を選択すると、IDE ウィンドウに 2 つのバージョンのファイルの違いが表示されます。一方の側には、リポジトリ内で現行版として追跡されているバージョンが表示されます。もう一方の側には、まだコミットされていない変更されたバージョンが表示されます。



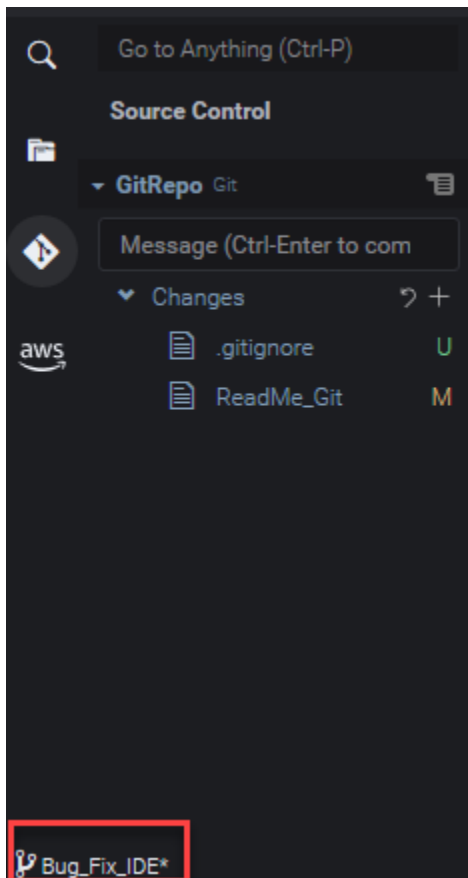
ブランチの操作

Git のおかげで、リポジトリのメインブランチとは独立したブランチの新たな特徴の操作を許可するワークフロー管理が大幅に容易になります。複数のブランチをシームレスに切り替えることができ、メインブランチに簡単に構築できるソースコードが常に存在します。

ブランチを作成する

ブランチを作成するには、ブランチに名前を付け、開始点を選択します。

1. Git パネルメニューで、チェックアウト先を選択。または、Git パネルの下部に表示される現在のブランチの名前を選択することもできます。



2. 新しいブランチを作成するためのオプションを選択します。
 - 新しいブランチを作成する: 新しいブランチは、現在のブランチの最後のコミットから始まります。
 - 新しいブランチの作成元: 新しいブランチは、後続の画面で選択したブランチの最後のコミットから始まります。
3. 新しいブランチの名前を入力します。
4. ブランチの開始点として特定のブランチを指定する場合は、一覧から選択します。

新しいブランチに切り替えた後、Git パネルの下部を表示することで、現在のブランチの名前を確認できます。

Note

リモートリポジトリで作業している場合は、[新しいブランチを発行](#)して、アップストリームのリモートリポジトリに発行して、他のユーザーによる自分のコンテンツへのアクセスを許可します。

ブランチを切り替える

Git でソース管理のキーとなる利点の1つは、ブランチを切り替えるだけで様々なプロジェクトの間をジャンプできることです。

Important

リポジトリにこみっとしていないファイルが現在のブランチにある場合は、ブランチを切り替えることはできません。まず、仕事を[コミット](#)するか、仕事を[stash](#)して、作業ディレクトリをクリーンアップする必要があります。

1. Git パネルの下部にある現在のブランチの名前を選択します。または、Git パネルに移動し、チェックアウト先を選択します。
2. 表示されたリストからブランチを選択します。

切り替え後、リポジトリの作業ディレクトリは、選択したブランチに最後にコミットされたファイルバージョンで更新されます。

ブランチをマージ

個別ブランチの特徴に対する作業が終了したら、通常、変更をメインプロジェクトに統合したくなるでしょう。Git では、この種の統合は、あるブランチ (例えば、特徴ブランチ) を別のブランチ (通常はリポジトリのメインブランチまたはデフォルトのブランチ) にマージすることによって促進されます。

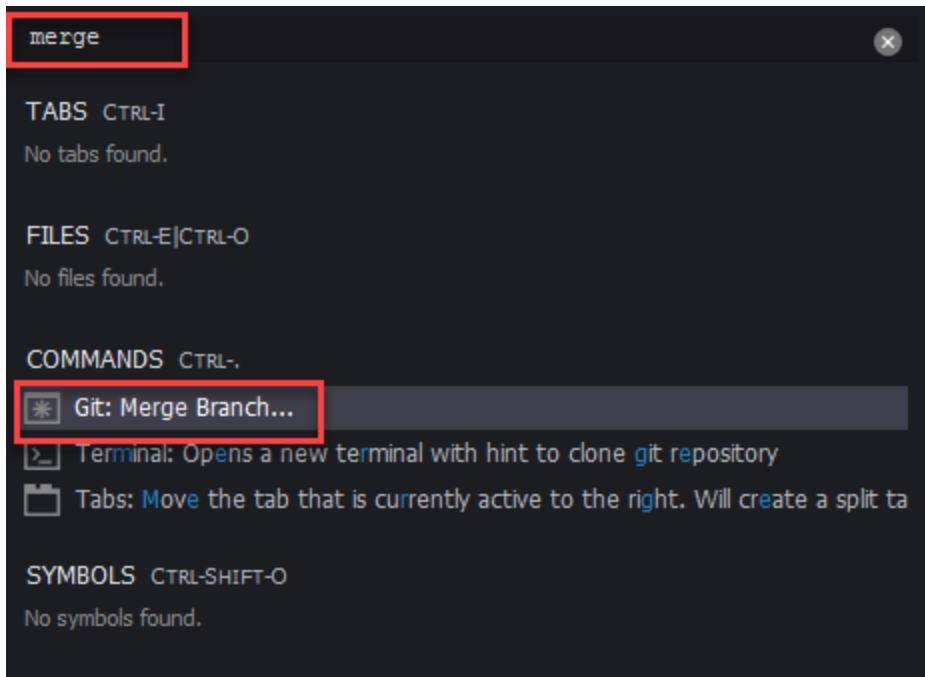
1. 別のブランチをマージするブランチを選択するには、Git パネルメニューに移動し、チェックアウト先を選択してください。

または、Git パネルの下部にある現在のブランチの名前を選択します。

2. 表示されるリストから、切り替えるブランチを選択します。

3. Git パネルの [検索] ボックスで、「マージ」という言葉の入力をスタートします。

コマンドリストの下で Git: Merge Branch が表示されるときは、それを選択します。

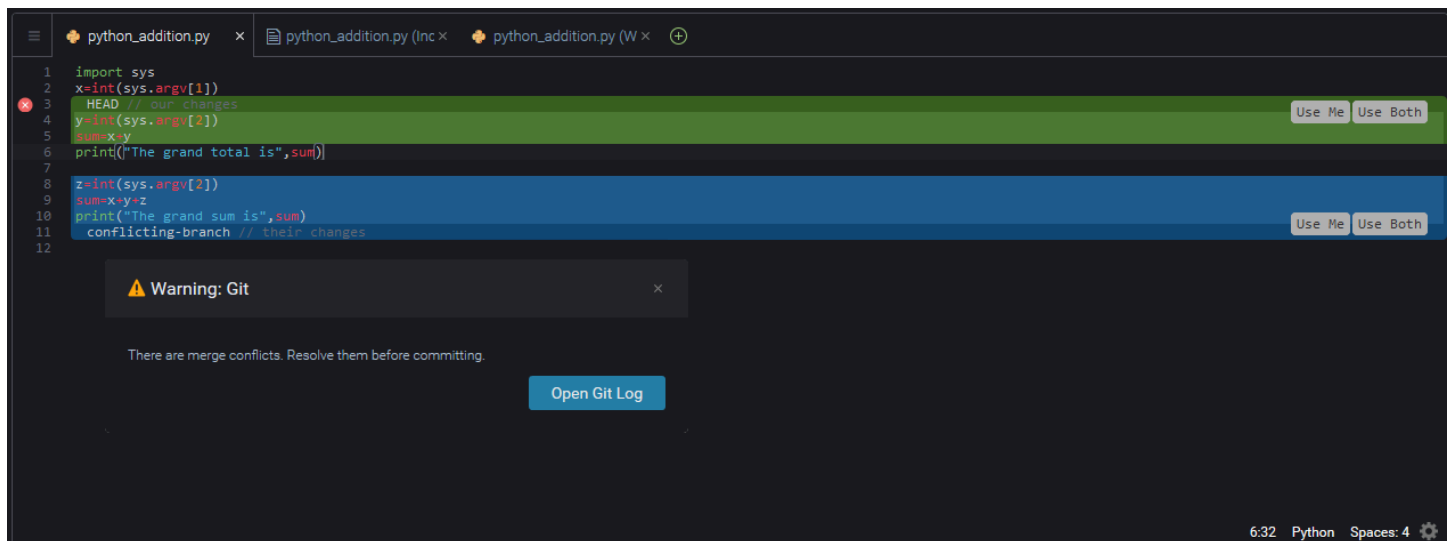


4. 表示されたリストから、ターゲットブランチにマージするブランチを選択します。

マージが競合せずに完了すると、Git パネルインターフェイスが更新され、マージされた変更を含むターゲットブランチが表示されます。

[ブランチのマージ](#)の時に、同じコンテンツに加えられた互換性のない変更起因するマージの競合が発生する可能性があります。このような場合は、マージをコミットする前に競合を解決する必要があるという警告が表示されます。

IDE のコードエディタウィンドウを使用して、2 つのブランチで競合するコンテンツを特定し、変更を加えて相違を解決できます。



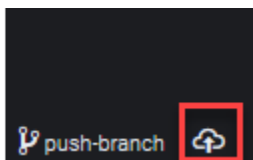
リモートリポジトリを操作する

インターネットまたはネットワークでホストされているリモートリポジトリは、チームメンバーがローカルの責任にコミットした変更の共有を許可して、コラボレーションを促進します。データをアップロードおよびダウンロードする Git コマンドを使用することで、「ダウンストリーム」(ローカル)リポジトリの内容が「アップストリーム」(リモート)リポジトリの内容と同期するようにします。

リモートリポジトリへのブランチの発行

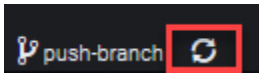
ローカルリポジトリのブランチを作成した後は、お客様にとってプライベートであり、お客様がプッシュしてリモートリポジトリに「アップストリーム」するまで、共同作業者は利用できません。

1. 現在のブランチを発行するには、[Git パネル] メニューに移動して、ブランチの発行 を選択します。または、Git パネルの下部にあるブランチ名の横にあるクラウドシンボルをクリックします。



2. 必要に応じて、リモートリポジトリにアクセスするため、サインイン認証情報を入力します。

ブランチがリモートリポジトリに正常にパブリッシュされると、Git パネルの下部にあるブランチ名の横に同期シンボルが表示されます。これを選択すると、ローカルリポジトリとリモートリポジトリの内容が同期されます。



ローカルリポジトリとリモートリポジトリ間でコンテンツをプッシュおよびプルする

Git を使用して共有プロジェクトで共同作業を行う場合は、通常、リモートリポジトリからローカルリポジトリに他のチームメンバーによる最近の変更取得でスタートします。また、ローカル repo への変更をコミットした後、リモートリポジトリにプッシュして、チームの残るメンバーがアクセスできるようにします。これらのアクションのパフォーマンスは、コマンド `git pull` および `git push` で行います。

Note

ほとんどのホストされたリポジトリ (GitHub 上のリポジトリなど) との間で変更をプッシュおよびプルするときは、サインイン認証情報を入力する必要があります。

Pull changes from remote

Git パネルインタフェースを通じて `git pull` コマンドを使用すると、リモートリポジトリ内のブランチにコミットされた最新の変更でローカルリポジトリを更新できます。

1. Git パネルメニューで、チェックアウト先を選択。
2. ブランチのリストで、変更をプルしたいローカルブランチを選択します。
3. 次に、を選択します。[Git パネル] メニューに移動し、[プル元] を選択します。
4. リモートリポジトリを選択し、そのリポジトリ内のブランチを選択して変更をプルします。

プルを実行した後、リポジトリの作業ディレクトリのリモート repo から取得したファイルにアクセスできます。ファイルを変更したら、変更をリモートブランチにプッシュできます。

Push changes to remote

Git パネルインターフェース経由で `git push` コマンドを使用すると、ローカルリポジトリ内の指定されたブランチの最新の変更を使ってリモートリポジトリを更新できます。

1. Git パネルメニューで、チェックアウト先を選択。
2. ブランチのリストで変更をプッシュしたいローカルブランチを選択します。
3. 次に、[Git パネル] メニューに移動し、プッシュ先を選択します。

4. リモトリポジトリを選択し、そのリポジトリ内のブランチを選択して変更をプッシュします。

プッシュを実行した後、他のチームメンバーは、自分のリポジトリのローカルコピーにプルダウンして、お客様が行った変更にアクセスできます。

ファイルの stash と取得

Git の stash 特徴を使用すると、最初にステージングしたファイルや変更したファイルをコミットせずに、ブランチを切り替えることができます。stash の特徴は、作業ディレクトリとステージングエリアの現在のステータスをキャプチャし、後で使用するために保存します。この特徴は、未完成のコンテンツまだ取り組んでおり、ブランチを遅滞なく切り替える必要がある場合に便利です。

stash 作業

1. 作業ディレクトリの現在の状態を stash (一時的に対比) するには、[Git パネル] メニューで、以下のいずれかのオプションを選択します。
 - stash: 作業ディレクトリ内のすべての変更またはステージングされたファイルが stash に追加されます。未追跡ファイルは追加されません。
 - stash (未追跡を含む): まだ追跡されていないファイルも含めて、作業ディレクトリ内のすべてのファイルが stash に追加されます。
2. 将来の取得のために stash の識別に役立つオプションのメッセージを入力します。

stash の後、Git パネルのインターフェイスがリフレッシュされ、クリーンアップされた作業ディレクトリが表示されます。

stash を取得する

1. stash を取得して作業ディレクトリに適用するには、[Git パネル] メニューで、以下のオプションの 1 つを選択します。
 - stash を適用: 選択した stash を作業ディレクトリに適用し、後で使えるように stash を保持します。
 - stash をポップ: 選択した stash を作業ディレクトリに適用し、stash スタックから stash を削除します。

 Note

また、stash スタックに追加された最後の stash の適用またはポップを選択することもできます。

2. stash を選択して、作業ディレクトリに適用します。

Git パネルインターフェイスが更新され、stash が適用されて作業ディレクトリが表示されます。

リファレンス: Git パネルで利用可能な Git コマンド

の Git パネルメニュー AWS Cloud9 は、コアコマンドとアドバンスド git コマンドの両方への便利なユーザーインターフェイスアクセスを提供します。

ブランチのマージや削除に使用されるコマンドなど、特定の git コマンドは、Git パネルの検索フィールドを通じてのみ使用できます。

また、Git パネルがコマンドを実行し、リポジトリと連携する方法をカスタマイズすることもできます。デフォルト設定を変更するには、まず AWS Cloud9、Preferences (設定) を選択します。[Preferences (設定)] ウィンドウで、[Project Settings (プロジェクト設定)] を選択します。

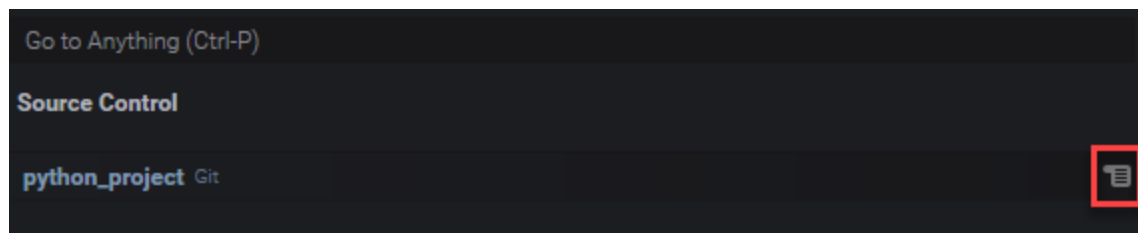
情報アイコンの上にカーソルを合わせると、設定の簡単な説明の読み取りができます。

Note

公式の Git サイトからリストされている Git コマンドの詳細なドキュメントにアクセスできます: <https://git-scm.com/doc>。

Git パネルメニューから利用可能な Git コマンドのリファレンス

リポジトリ名の反対側の記号を選択して、Git パネルメニューにあるオプションにアクセスします。



Git パネルメニュー

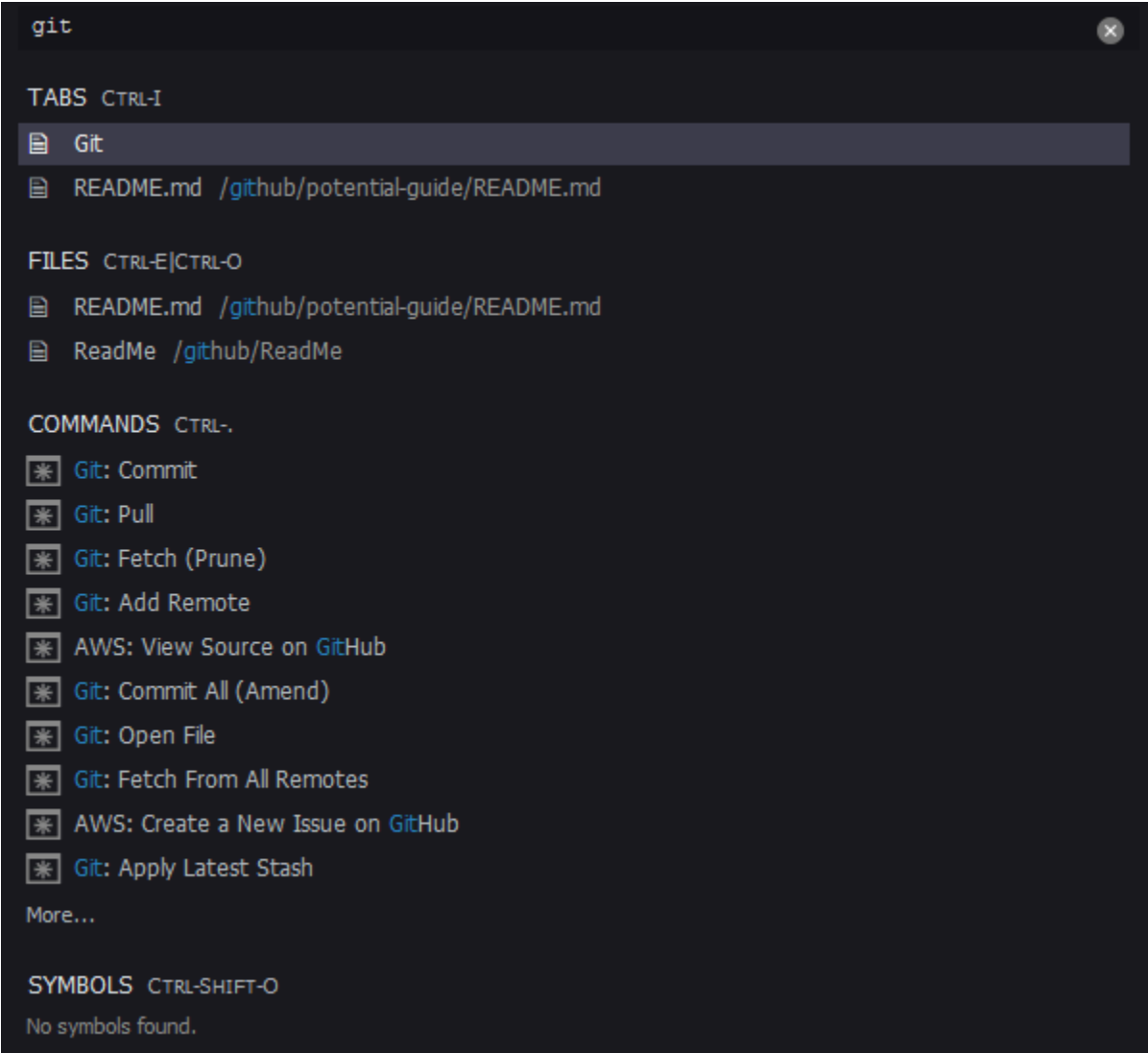
メニューオプション	説明
コミット	リポジトリの作業ディレクトリにステージングエリアを追加したコンテンツをコミットします。コミットメッセージを追加します。
更新	GitPanel インターフェイスを更新して、作業ディレクトリとステージングエリアのステータスを表示します。
プル	リモートリポジトリからローカルリポジトリに最新の変更をプルします。
プル (リベース)	リモートブランチからプルされたリモート変更、ローカルの変更を再適用します。
...からプッシュ	ローカルリポジトリ内のブランチにコミットされた変更をリモートリポジトリ内のブランチにプッシュします。
プッシュ	ローカルリポジトリにコミットされた変更内容をリモートリポジトリにプッシュします。
プッシュ先...	ローカルリポジトリ内のブランチにコミットされた変更をリモートリポジトリ内のブランチにプッシュします。
同期	git pull コマンドの後に git push コマンドを実行してローカルリポジトリとリモートリポジトリのコンテンツを同期します。

メニューオプション	説明
チェックアウト先...	既存のブランチに切り替えるか、ブランチを作成してそのブランチに切り替えます。
ブランチの発行	ローカルリポジトリで作成されたプライベートブランチを公開し、リモートリポジトリで使えるようにします。
すべてコミット	ステージングされたファイルとステージングされていないファイルの両方をリポジトリにコミットします。(git commit コマンドを実行する前に git add -A コマンドを実行して、ステージングエリアにファイルを追加します)。
すべてコミット (修正)	前回のコミットのメッセージを変更します。(git commit コマンドを実行するときに、-amend オプションを追加します)。
すべてコミット (サインオフ)	Git ログでコミットを実行したユーザーを識別します。(git commit コマンドを実行する時に、-signed-off オプションを追加します)。
ステージングしたものをコミット	ステージングしたファイルのみをリポジトリにコミットします。
ステージングしたものをコミット (修正)	前回のコミットのメッセージを変更します。(git commit コマンドを実行する時に、-amend オプションを追加します)。
ステージングしたものをコミット (サインオフ)	Git ログでコミットを実行したユーザーを識別します。(git commit コマンドを実行する時に、-signed-off オプションを追加します)。
[Undo Last Commit] (前回のコミットを元に戻す)	前のコミットを元に戻します。ファイルはステージングエリアに戻されます。
すべての変更を破棄	リポジトリのステージングエリアからすべてのファイルとフォルダを削除します。

メニューオプション	説明
すべての変更をステージング	追跡していないコンテンツおよび変更されたコンテンツをステージングエリアに追加します。
すべての変更のステージング解除	ステージングエリアからすべてのファイルを移動します。ステージングされていないファイルはリポジトリにコミットできません。
最新の stash を適用する	スタック stash に追加された最後の stash を作業ディレクトリに適用します。stash はスタックに残ります。
Stash を適用する...	stash スタックから選択された stash を作業ディレクトリに適用します。stash はスタックに残ります。
最新 stash をポップ	スタック stash に追加された最後の stash を作業ディレクトリに適用します。その後、stash はスタックから削除されます。
Stash のポップ...	選択した stash を作業ディレクトリに適用します。その後、stash はスタックから削除されます。
Stash	作業ディレクトリ内の変更されたファイルとステージングされたファイルを名前がついた stash に追加します。
Stash (追跡されていないものを含む)	追跡していないファイルを含む作業ディレクトリ内のすべてのファイルを名前が付いた stash に追加します。
Git 出力を表示	Git パネルインターフェースと連携するときに実行される Git コマンドを示すウィンドウが表示されます。

Git パネルの検索フィールドから利用可能な Git コマンド

検索ボックスに「git」と入力すると、Git パネルメニューでは利用できないサポートされている Git コマンドにアクセスすることもできます。



次の表に、この方法でアクセスできる選択した Git コマンドの説明を示します。

選択された Git コマンド

メニューオプション	説明
Git: リモートを追加する	リモートリポジトリへの接続を Git 設定ファイルに追加する
Git: ブランチを削除する	指定されたブランチを削除します。
Git: 取得	リモートリポジトリ内のブランチからコンテンツをダウンロードします。git pull とは対照的に、リモート変更はローカルリポジトリにマージされません。

メニューオプション	説明
Git: ブランチをマージする	あるブランチで行った変更を別のブランチに統合します。詳細については、 ブランチのマージ手順 を参照してください。

Toolkit の使用 AWS

AWS Explorer ウィンドウから AWS Toolkit AWS のサービス を使用してナビゲートおよび操作できます。

Toolkit を使用する理由 AWS

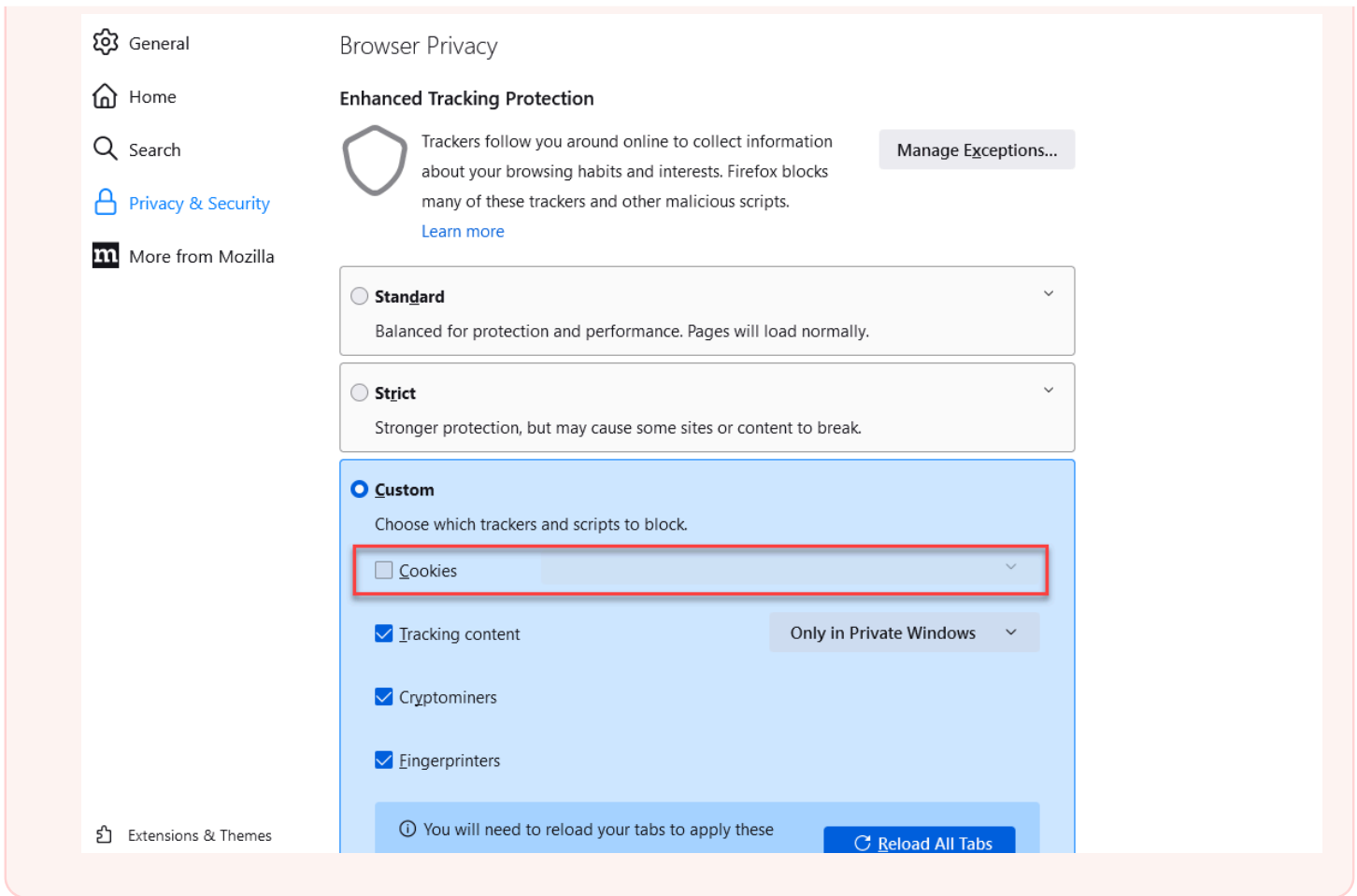
Toolkit AWS は、AWS Cloud9 統合開発環境 (IDE) の拡張機能です。この拡張機能 AWS のサービスを使用すると、 にアクセスしてさまざまな を操作できます。Toolkit は、Lambda プラグインによって提供される機能を AWS 置き換えます AWS Cloud9。詳細については、「[Toolkit の無効化 AWS](#)」を参照してください。

Important

AWS ツールキットのサポートは、 の統合機能です AWS Cloud9。現在、サードパーティーの拡張機能を使用して AWS Cloud9 IDE をカスタマイズすることはできません。

Warning

IDE で Mozilla Firefox AWS Cloud9 を優先ブラウザとして使用している場合は、AWS Cloud9 ウェブビューと AWS Toolkit がブラウザで正しく動作しないようにするサードパーティーの Cookie 設定があります。この問題の回避策として、下の画像に示すように、ブラウザ設定の「プライバシーとセキュリティ」セクションで [Cookies] をブロックしていないことを確認する必要があります。



現在、次の AWS のサービス および リソースには AWS Toolkit 拡張機能からアクセスできます。

- [AWS App Runner](#)
- [API Gateway](#)
- [CloudFormation スタック](#)
- [CloudWatch Logs](#)
- [AWS Lambda](#)
- [リソース](#)
- [Amazon S3 バケットとオブジェクト](#)
- [AWS Serverless Application Model applications](#)
- [Step Functions とステートマシン](#)
- [Systems Manager オートメーションドキュメント](#)
- [IDE での Amazon ECR AWS Cloud9 の使用](#)

- [AWS IoT](#)
- [???](#)
- [Amazon EventBridge](#)
- [の使用 AWS Cloud Development Kit \(AWS CDK\)](#)

AWS ツールキットの有効化

Toolkit AWS が環境で使用できない場合は、設定タブで有効にできます。

AWS ツールキットを有効にするには

1. メニューバーで、AWS Cloud9、[設定] を選択します。
2. [設定] タブでは、サイドナビゲーションペインで、[AWS 設定] を選択します。
3. [AWS リソース] ペインで、[AWS ツールキット] を有効にして、緑の背景にチェックマークを表示します。

AWS Toolkit を有効にすると、統合開発環境 (IDE) が更新され、更新された Enable AWS Toolkit 設定が表示されます。IDE の片側にある [環境] オプションの下に [AWS ツールキット] オプションも表示されます。

Important

AWS Cloud9 環境の EC2 インスタンスがインターネットにアクセスできない場合 (つまり、アウトバウンドトラフィックが許可されていない場合)、AWS Toolkit を有効にして IDE を再起動した後にメッセージが表示されることがあります。このメッセージは、AWS ツールキットが必要とする依存関係をダウンロードできなかったことを示します。この場合、AWS Toolkit も使用できません。

この問題を解決するには、Amazon S3 用の VPC エンドポイントを作成します。これにより、IDE を最新の状態に保つために必要な依存関係 AWS リージョン を含む 内の Amazon S3 バケットへのアクセスが許可されます。

詳細については、「[依存関係をダウンロードするため、Amazon S3 の VPC エンドポイントを設定する](#)」を参照してください。

AWS Toolkit のアクセス認証情報の管理

AWS Toolkit は、さまざまな とやり取りします AWS のサービス。アクセスコントロールを管理するには、AWS Toolkit サービスの IAM エンティティに、この範囲のサービスに必要なアクセス許可があることを確認します。すばやく開始するには、[AWS マネージド一時認証情報](#)を使用して必要なアクセス許可を取得します。これらのマネージド認証情報は、IAM ユーザーなどの AWS エンティティ AWS のサービス に代わって へのアクセス権を EC2 環境に付与することで機能します。

ただし、開発環境の EC2 インスタンスをプライベートサブネットで起動した場合、AWS マネージド一時認証情報は使用できません。したがって、代わりに、独自の認証情報のセットを手動で作成 AWS のサービス することで、AWS Toolkit が にアクセスすることを許可できます。このセットはプロファイルと呼ばれます。プロファイルには、アクセスキーと呼ばれる長期的な認証情報が必要です。これらのアクセスキーは、IAM コンソールから取得できます。

AWS Toolkit のアクセス認証情報を提供するプロファイルを作成する

1. アクセスキー (アクセスキー ID とシークレットアクセスキーで構成) を入手するため、IAM コンソール (<https://console.aws.amazon.com/iam>) にアクセスします。
2. ナビゲーションバーからユーザーを選択し、AWS ユーザー名を選択します (チェックボックスではありません)。
3. [Security credentials] (セキュリティ認証情報) タブを選択してから [Create access key] (アクセスキーの作成) を選択します。

Note

既にアクセスキーを持っても、シークレットキーにアクセスできない場合は、古いキーを非アクティブにして、新しいキーを作成します。

4. アクセスキー ID とシークレットアクセスキーを表示するダイアログボックスで、.csv ファイルのダウンロードを選択し、この情報は安全な場所に保存してください。
5. アクセスキーをダウンロードしたら、AWS Cloud9 環境を起動し、Window, New Terminal を選択してターミナルセッションを開始します。
6. ターミナルウィンドウで、次のコマンドを実行します。

```
aws configure --profile toolkituser
```

この場合、`toolkituser` は使用中のプロファイル名ですが、独自のプロファイル名を選択できます。

7. コマンドラインで、IAM コンソールで以前にダウンロードした AWS Access Key ID および AWS Secret Access Key を入力します。

- には Default region name、を指定します AWS リージョン (例: `us-east-1`)。
- Default output format として、ファイル形式 (`json` など) を指定します。

 Note

プロファイルを設定するためのオプションについては、AWS Command Line Interface ユーザーガイドの「[設定の基本](#)」を参照してください。

8. プロファイルを作成したら、AWS Toolkit を起動し、[AWS Toolkit メニュー](#)に移動して、Connect to AWSを選択します。
9. AWS 認証情報プロファイルの選択 フィールドで、ターミナルで作成したプロファイル (など `profile:toolkituser`) を選択します。

選択したプロファイルに有効なアクセス認証情報が含まれている場合、AWS Explorer ペインが更新されて、アクセスできる AWS のサービス が表示されます。

IAMロールを使用して、EC2 インスタンスのアプリケーションに許可を付与する

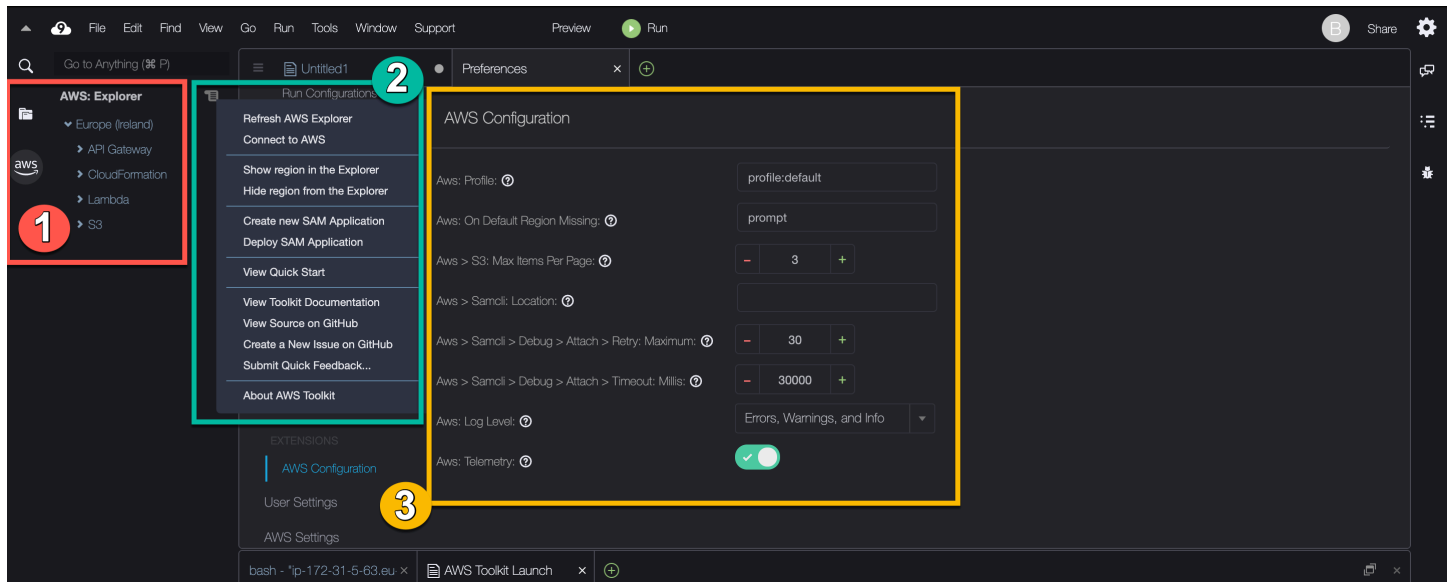
また、IAM ロールを使用して、EC2 インスタンス上で実行するアプリケーションの一時認証情報を管理することもできます。このロールは、アプリケーションが他の AWS リソースを呼び出すときに使用できる一時的なアクセス許可を提供します。EC2 インスタンスを起動するときに、そのインスタンスに関連付ける IAM ロールを指定します。これにより、AWS のサービスに対してAPI リクエストを行ったときに、そのインスタンスで実行するアプリケーションは、このロールが提供する一時認証情報を使用できるようになります。

ロールを作成したら、インスタンスプロファイルを作成して、このロールおよび関連するアクセス許可をインスタンスに割り当てます。インスタンスプロファイルは、インスタンスにアタッチされ、インスタンス上で実行されるアプリケーションにロールの一時認証情報を提供できます。

詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する](#)」を参照してください。

Toolkit AWS コンポーネントの特定

次のスクリーンショットは、AWS Toolkit の 3 つの主要な UI コンポーネントを示しています。



1. AWS Explorer ウィンドウ: Toolkit からアクセスできる AWS のサービス を操作するために使用されます。統合開発環境 (IDE) の左側にある AWS オプションを使用して、AWS Explorer の表示と非表示を切り替えることができます。このインターフェイスコンポーネントの使用と AWS のサービス へのさまざまなアクセスの詳細については AWS リージョン、「」を参照してください [AWS Explorer を使用して複数のリージョンのサービスとリソースを操作する](#)。
2. ツールキットメニュー: への接続の管理 AWS、AWS Explorer ウィンドウの表示のカスタマイズ、サーバーレスアプリケーションの作成とデプロイ、GitHub リポジトリの操作、ドキュメントへのアクセスに使用されます。詳細については、「[Toolkit AWS メニューへのアクセスと使用](#)」を参照してください。
3. AWS 設定ペイン: Toolkit を使用して操作 AWS のサービス する の動作をカスタマイズするために使用します。詳細については、「[AWS 設定ペインを使用した AWS Toolkit 設定の変更](#)」を参照してください。

Toolkit の無効化 AWS

ツールキットは、設定タブ AWS で無効にできます。

AWS ツールキットを無効にするには

1. メニューバーで、AWS Cloud9、[設定] を選択します。
2. [設定] タブでは、サイドナビゲーションペインで、[AWS 設定] を選択します。
3. AWS リソースペインで、AWS AWS ツールキットをオフにします。

AWS Toolkit を無効にすると、統合開発環境 (IDE) が更新され、環境オプションの下にある IDE の側面にある AWS Toolkit オプションが削除されます。

AWS ツールキットのトピック

- [Toolkit の操作と設定 AWS](#)
- [AWS Toolkit AWS App Runner での の使用](#)
- [Toolkit を使用した API Gateway の使用 AWS](#)
- [AWS Toolkit を使用した AWS CloudFormation スタックの操作](#)
- [Toolkit を使用した AWS Lambda 関数の操作 AWS](#)
- [リソースの使用](#)
- [Toolkit を使用した Amazon S3 の使用 AWS](#)
- [Toolkit AWS SAM を使用した の操作 AWS](#)
- [Amazon CodeCatalyst の使用](#)
- [???](#)

Toolkit の操作と設定 AWS

リソースにアクセスし、次の AWS Toolkit インターフェイス要素を使用して設定を変更できます。

- [AWS Explorer ウィンドウ](#): 異なる AWS のサービス からのアクセス AWS リージョン。
- [AWS ツールキットメニュー](#): サーバーレスアプリケーションの作成とデプロイ、表示または非表示 AWS リージョン、ユーザーサポートへのアクセス、Git リポジトリの操作を行います。
- [AWS 設定ペイン](#): Toolkit AWS のサービス で を操作する方法に影響する設定を変更します AWS 。

AWS Explorer を使用して複数のリージョンのサービスとリソースを操作する

AWS Explorer ウィンドウでは、そのサービスに関連付けられている特定のリソースを選択して AWS のサービス 操作できます。[AWS Explorer] で、サービス名のノード (API Gateway や Lambda など) を選択します。次に、そのサービスに関連する特定のリソース (REST API や Lambda 関数など) を選択します。特定のリソースを選択すると、アップロード/ダウンロード、呼び出し、コピーなど、使用可能な操作オプションがメニューに表示されます。

次の例を考えます。AWS アカウント 認証情報が Lambda 関数にアクセスできる場合は、 にリストされている Lambda ノードを展開し AWS リージョン、呼び出す、またはコードとして AWS Cloud9 IDE にアップロードする特定の Lambda 関数を選択します。また、ノードのタイトルを右クリックしてコンテキストメニューを開き、AWS Serverless Application Modelを使用するアプリケーションの作成を開始できます。

Note

統合開発環境 (IDE) で AWS Explorer ウィンドウを表示するオプションが表示されない場合は、AWS Toolkit を有効にしていることを確認します。有効になっていることを確認したら、もう一度試してください。詳細については、「[AWS ツールキットの有効化](#)」を参照してください。

[AWS Explorer] ウィンドウでは、複数の AWS リージョンでホストされているサービスを表示することもできます。

選択したリージョン AWS のサービス から にアクセスするには

1. [AWS Explorer] ウィンドウで、[ツールキット] メニューの「Show region in Explorer (Explorer でリージョンを表示)」を選択します。
2. Explorer AWS リストに表示するリージョンの選択から、 を選択します AWS リージョン。

選択したリージョンが [AWS Explorer] ウィンドウに追加されます。利用可能なサービスやリソースにアクセスするには、リージョン名の前にある矢印 (>) を選択してください。

Note

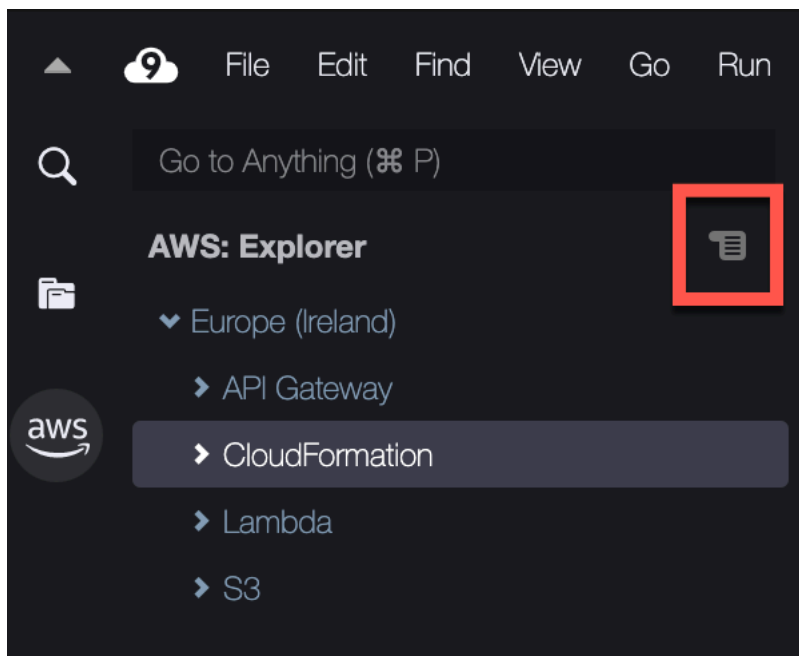
次のオプションを使用して、AWS リージョン Explorer ウィンドウで選択した を非表示にすることもできます。AWS

- リージョンを右クリックしてコンテキストメニューを開き、[Hide region from the Explorer] (Explorer からリージョンを非表示) を選択します。
- AWS ツールキットメニューで、Explorer からリージョンの非表示を選択し、非表示にするリージョンを選択します。

Toolkit AWS メニューへのアクセスと使用

-AWS ツールキットは、[サーバーレスアプリケーション](#)の作成およびデプロイするオプションへのアクセスを提供します。このメニューを使用して、接続の管理、[AWS: Explorer] ウィンドウの更新、ドキュメントへのアクセス、GitHub リポジトリの操作を行うことができます。

[ツールキット] メニューにアクセスするには、[AWS Explorer] ウィンドウ内の反対側にあるスクロールアイコン [AWS: Explorer] を選択します。



次の表は、[Toolkit] (ツールキット) メニューで使用可能なオプションの概要を示しています。

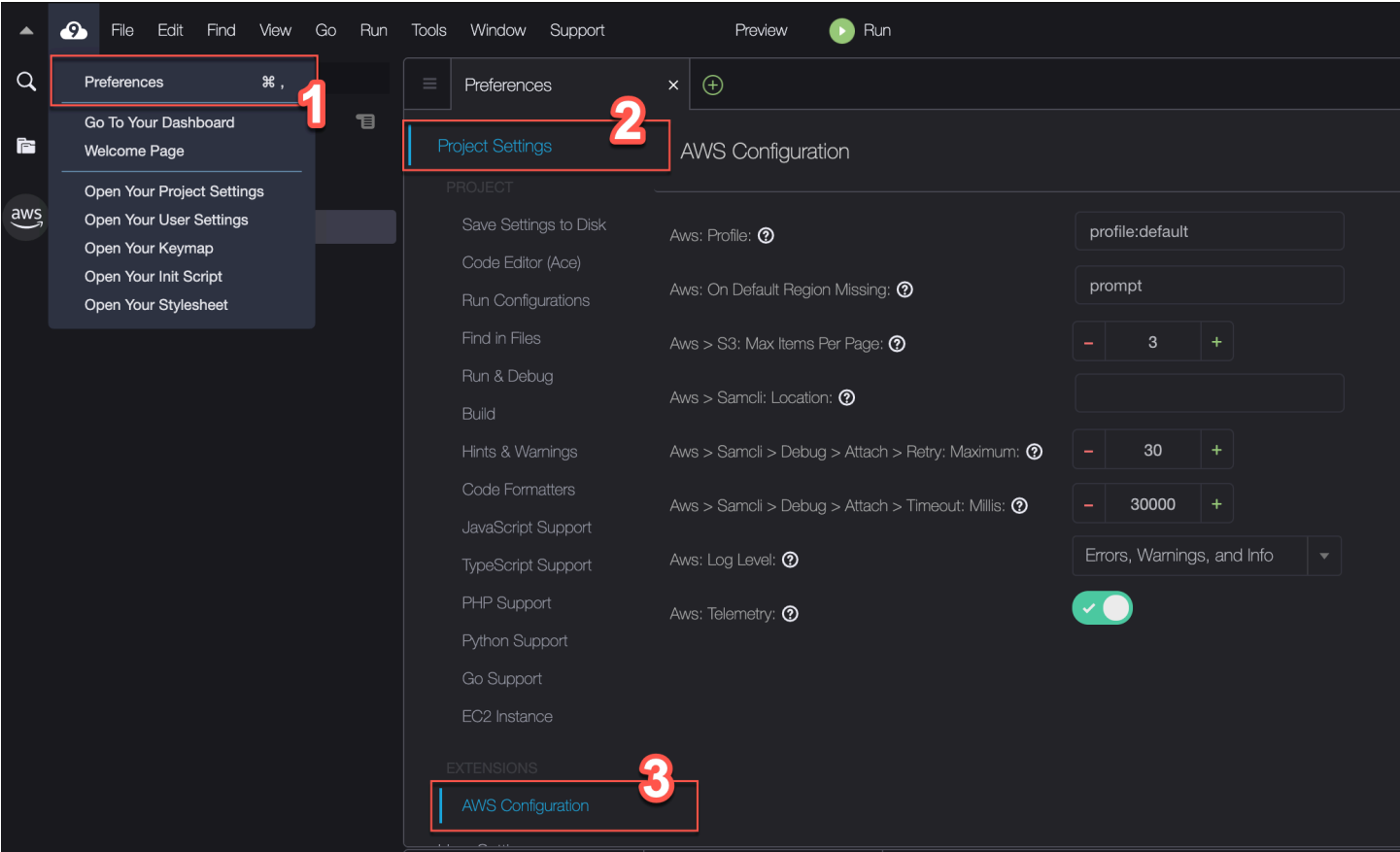
[Toolkit] (ツールキット) メニューオプション

メニューオプション	説明
Explorer の更新 AWS	このオプションを選択して AWS Explorer を更新し、前回ウィンドウを開いてから変更された AWS のサービス を表示します。
に接続する AWS	プロファイルに保存されている認証情報 AWS アカウント を使用して AWS Toolkit を に接続します。詳細については、「 AWS Toolkit のアクセス認証情報の管理 」を参照してください。
Explorer にリージョンを表示	AWS Explorer ウィンドウ AWS リージョンに を表示します。詳細については、「 AWS Explorer を使用して複数のリージョンのサービスとリソースを操作する 」を参照してください。
Explorer からリージョンを非表示	AWS Explorer ウィンドウで を非表示 AWS リージョン にします。詳細については、 AWS Explorer を使用して複数のリージョンのサービスとリソースを操作する を参照してください。
新しい SAM アプリケーションを作成	新しい AWS サーバーレスアプリケーションのコードファイルのセットを生成します。SAM アプリケーションの作成およびデプロイの詳細については、「 Toolkit AWS SAM を使用したの操作 AWS 」を参照してください。
SAM アプリケーションをデプロイ	サーバーレスアプリケーションを にデプロイします AWS。SAM アプリケーションの作成およびデプロイの詳細については、「 Toolkit AWS SAM を使用したの操作 AWS 」を参照してください。
クイックスタートを表示	クイックスタートガイドを開きます。
ツールキットドキュメントを表示	AWS Toolkit のユーザーガイドを開きます。

メニューオプション	説明
GitHub でソースを表示	Toolkit の GitHub リポジトリを開きます AWS 。
GitHub で新しい問題を作成	Github で AWS Toolkit の新しい問題ページを開きます
クイックフィードバックを送信	Toolkit AWS 開発チームに一方向のプライベートフィードバックを送信します。会話やバグ修正を必要とする問題については、[Create a New Issue on Github] (Github で新しい問題を作成) メニューオプションを選択して、Github で問題を送信します。
AWS Toolkit について	実行しているツールキットのバージョンと、設定されている Amazon OS に関する情報を表示します。

AWS 設定ペインを使用した AWS Toolkit 設定の変更

AWS 設定ペインにアクセスするには、 [AWS Cloud9,Preferences (設定)] を選択します。次に、 [Project Settings] の下にある [Preferences (設定)] ウィンドで、 [AWS 設定] を選択します。



次の表は、[AWS 設定] ペインで利用できるオプションの概要を示しています。

メニューオプション	説明
AWS: プロファイル	認証情報を取得する認証情報プロファイルの名前を設定します。
AWS: デフォルトのリージョンがありません	AWS リージョン 選択した認証情報プロファイルのデフォルトが AWS Explorer ウィンドウで使えない場合に実行するアクションを示します。以下の 3 つのオプションから選択できます。 - prompt (プロンプト) (デフォルト): 実行したいアクションを尋ねられます。 - add (追加): リージョンは [AWS Explorer] ウィンドウに表示されます。

メニューオプション	説明
	• ignore (無視): アクションは実行されません。
AWS > S3: ページあたりの最大項目数	一度に表示される Amazon S3 オブジェクトまたはフォルダの数を [AWS Explorer] ウィンドウで表示します。最大数が表示されたら、ロードを増加 を選択して次のバッチを表示することができます。 このフィールドで受け入れられる値の範囲は 3 ~ 1000 です。この設定は、一度に表示されるオブジェクトまたはフォルダの数にのみ適用されます。作成したすべてのバケットが一度に表示されます。デフォルトでは、各 に最大 100 個のバケットを作成できます AWS アカウント。
AWS > Samcli: 場所	作成、構築、パッケージ化、および サーバーレスアプリケーション のデプロイに使用する SAM CLI の場所を示します。
AWS > Samcli > デバッグ > アタッチ > 再試行: 最大:	ツールキットが接続を中止する前に SAM CLI デバッガーのアタッチを試みる回数を指定します。デフォルトの試行回数のクォータは 30 です。 SAMCLI 内のデバッグモードで Lambda AWS 関数をローカルで呼び出すと、デバッガーをアタッチできます。

メニューオプション	説明
AWS > Samcli > Debug > Attach> Timeout: Millis:	ツールキットがあきらめる前に SAM CLI デバッガーにアタッチを試みる時間を指定します。デフォルトのタイムアウトは30,000ミリ秒（30秒）です。 SAMCLI 内のデバッグモードで Lambda AWS 関数をローカルで呼び出すと、デバッガーをアタッチできます。
AWS : ログレベル:	記録されるワークフローイベントのカテゴリを設定します。使用可能なレベルは以下のとおりです。 - エラーのみ - エラーおよび警告 - エラー、警告、情報 (デフォルトオプション) - エラー、警告、情報、詳細、デバッグ
AWS : テレメトリ	への使用状況データの送信を有効または無効にします AWS。デフォルトで有効

Toolkit を使用した API Gateway の使用 AWS

API Gateway を使用すると、リアルタイムの双方向通信アプリケーションを有効にする RESTful API と WebSocket API を作成できます。API Gateway を使用した API の作成と管理の詳細については、API Gateway デベロッパーガイドを参照してください。

Toolkit では AWS 、入力として渡される REST リソース、メソッドタイプ、およびデータを指定することで、REST API への呼び出しを設定できます。

API Gateway での REST API の呼び出し

Important

AWS Toolkit を使用して API メソッドを呼び出すと、元に戻すことができないリソースが変更される可能性があります。たとえば、POST メソッドを呼び出すと、呼び出しが成功すれば API のリソースが更新されます。

Toolkit AWS から で API Gateway を呼び出すことができます AWS 。

REST API を呼び出すには

1. [AWS Explorer] ウィンドウで、API Gateway ノードを選択し、現在の AWS リージョンで利用できる REST API のリストを表示します。
2. REST API を右クリックし、AWSで呼び出しを選択します。

Note

このコンテキストメニューを使用して、REST API の URL、名前、Amazon リソースネーム (ARN) をコピーできます。

[Invoke methods] (メソッドの呼び出し) ウィンドウが表示されます。API への呼び出しを設定できます。

3. [Select a resource] (リソースの選択) で、操作する REST リソースを選択します。
4. [Select a method] (メソッドの選択) で、次のいずれかのメソッドタイプを選択します。
 - GET: API を通じてアクセスされるバックエンドサービスからリソースを取得します。
 - オプション: API Gateway でサポートされているメソッドとオペレーションに関する情報を要求します。
 - POST: API を通じてアクセスされるバックエンドサービス上に新しいリソースを作成します。
5. API メソッド呼び出しに入力を提供するには、クエリ文字列または JSON 形式のペイロードを使用します。

- クエリ文字列: 形式 `parameter1=value1&parameter2=value2` を使用してクエリ文字列を入力します (クエリ文字列を使用する前に、[マッピングテンプレート](#)を作成して、統合バックエンドに送信する前の受信ウェブリクエストを変換します)。
- JSON 形式: [Invoke methods] (メソッドの呼び出し) ウィンドウのラージテキストフィールドで JSON 形式のペイロードを定義できます。

たとえば、次のペイロードを含むPOSTメソッドを使って、新しいリソースを追加できます。

```
{"type": "soda", "price" : 3.99}
```

6. [呼び出し] ボタンを選択して、REST API リソースを呼び出してください。

REST API レスポンスは [AWS リモート呼び出し] タブをクリックします。レスポンス本文には、JSON 形式のリソースデータが含まれます。

AWS Toolkit AWS App Runner での の使用

[AWS App Runner](#) を使用すると、AWS クラウド内で、ソースコードやコンテナイメージから、スケラブルでセキュアなウェブアプリケーションに対して、迅速かつ費用対効果の高い方法で直接デプロイを行うことができます。これを使用すると、新しいテクノロジーを学習したり、使用するコンピューティングサービスを決定したり、AWS リソースをプロビジョニングして設定する方法を知っている必要はありません。

を使用して AWS App Runner、ソースイメージまたはソースコードに基づいてサービスを作成および管理できます。ソースイメージを使用する場合は、イメージリポジトリに保存されているパブリックまたはプライベートコンテナイメージを選択できます。App Runner は以下のイメージリポジトリプロバイダーをサポートしています。

- Amazon Elastic Container Registry (Amazon ECR): AWS アカウントにプライベートイメージを保存します。
- Amazon Elastic Container Registry Public (Amazon ECR Public): パブリックに読み取り可能なイメージを保存します。

ソースコードオプションを選択した場合、サポートされているリポジトリプロバイダーによって管理されているソースコードリポジトリからデプロイできます。現在、App Runner でソースコードリポジトリプロバイダーとしてサポートされているのは、[GitHub](#) です。

前提条件

Toolkit を使用して App Runner を操作するには、以下 AWS が必要です。

- AWS アカウント
- が機能する AWS Toolkit のバージョン AWS App Runner

これらのコア要件に加えて、関連するすべての IAM ユーザーが App Runner サービスを操作するアクセス許可を持っている必要があります。また、コンテナイメージの URI や GitHub リポジトリへの接続など、サービスソースに関する特定の情報を確実に取得します。この情報は、App Runner サービスを作成するときに必要です。

App Runner の IAM アクセス許可の設定

App Runner に必要なアクセス許可をすばやく付与するには、既存の AWS 管理ポリシーを関連する AWS Identity and Access Management (IAM) エンティティにアタッチします。特に、ユーザーまたはグループにポリシーをアタッチできます。App Runner には、IAM ユーザーにアタッチできる 2 つのマネージドポリシーが用意されています。

- `AWSAppRunnerFullAccess`: ユーザーがすべての App Runner アクションを実行できるようにします。
- `AWSAppRunnerReadOnlyAccess`: App Runner リソースの詳細をリストおよび表示できます。

Amazon Elastic Container Registry (Amazon ECR) からサービスソースとしてプライベートリポジトリを選択した場合は、App Runner サービス用に次のアクセスロールも作成する必要があります。

- `AWSAppRunnerServicePolicyForECRAccess`: アカウント内の Amazon Elastic Container Registry (Amazon ECR) イメージへのアクセスを App Runner に許可します。

このロールは、AWS ツールキットのコマンドペインでサービスインスタンスを設定するときに、自動的に作成できます。

Note

`AWSServiceRoleForAppRunner` サービスにリンクされたロールにより AWS App Runner、は次のタスクを完了できます。

- Amazon CloudWatch Logs Logs ロググループにログをプッシュします。

- Amazon CloudWatch Events ルールを作成して、Amazon Elastic Container Registry (Amazon ECR) イメージプッシュを購読します。

サービスにリンクされたロールを手動で作成する必要はありません。AWS App Runner で、AWS マネジメントコンソール または AWS Toolkit によって呼び出される API オペレーションを使用して を作成すると、はこのサービスにリンクされたロールを自動的に AWS App Runner 作成します。

詳細については、AWS App Runner デベロッパーガイドの「[App Runner の Identity and Access Management](#)」を参照してください。

App Runner のサービスソースの取得

AWS App Runner を使用して、ソースイメージまたはソースコードからサービスをデプロイできます。

Source image

ソースイメージからデプロイする場合は、プライベートまたはパブリックイメージレジストリからその AWS イメージのリポジトリへのリンクを取得します。

- Amazon ECR プライベートレジストリ: Amazon ECR コンソール (<https://console.aws.amazon.com/ecr/repositories>) を使用するプライベートリポジトリの URI をコピーします。
- Amazon ECR パブリックレジストリ: Amazon ECR Public Gallery (<https://gallery.ecr.aws/>) を使用するパブリックリポジトリの URI をコピーします。

Note

プライベート Amazon ECR リポジトリの URI は、AWS Toolkit の AWS Explorer から直接取得することもできます。

- [AWS Explorer] を開き、[ECR] ノードを展開して、その AWS リージョンのリポジトリを一覧表示します。
- リポジトリを右クリックしてコンテキストメニューを開き、[Copy Repository URI] (リポジトリ URI をコピー) を選択して、リンクをクリップボードにコピーします。

Toolkit コマンドペインでサービスインスタンスを設定するときに、イメージリポジトリの URI AWS を指定します。

詳細については、AWS App Runner デベロッパーガイドの「[ソースイメージに基づいた App Runner サービス](#)」を参照してください。

Source code

ソースコードを AWS App Runner サービスにデプロイするには、そのコードを Git リポジトリに保存する必要があります。この Git リポジトリは、サポートされているリポジトリプロバイダーが管理する必要があります。App Runner でソースコードリポジトリプロバイダーとしてサポートされているのは、[GitHub](#) です。

GitHub リポジトリの設定については、GitHub の[入門ガイドドキュメント](#)を参照してください。

GitHub リポジトリから App Runner サービスにソースコードをデプロイする場合、App Runner は GitHub への接続を確立します。リポジトリがプライベートである (つまり GitHub でパブリックにアクセスできない) 場合は、App Runner に接続の詳細情報を指定する必要があります。

Important

GitHub 接続を作成するには、App Runner コンソール (<https://console.aws.amazon.com/apprunner>) を使用して、GitHub から AWS へリンクする接続を作成する必要があります。Toolkit の AWS コマンドペインを使用してサービスインスタンスを設定するときに、GitHub 接続ページで使用可能な接続を選択できます。

詳細については、AWS App Runner デベロッパーガイドの「[App Runner 接続の管理](#)」を参照してください。

App Runner サービスインスタンスは、コードの構築と実行を許可するマネージドランタイムを提供します。AWS App Runner は現在、次のランタイムをサポートしています。

- Python マネージドランタイム
- Node.js マネージランタイム

サービス設定の一環として、App Runner サービスがサービスをビルドして開始する方法に関する情報を指定します。この情報は、[Command Palette] (コマンドパレット) を使用して入力するか、YAML 形式の [App Runner 設定ファイル](#) を指定します。このファイルの値は、App Runner にサービスを構築して開始し、ランタイムコンテキストを提供する方法を指示します。これには、

関連するネットワーク設定と環境変数が含まれます。設定ファイルの名前は `apprunner.yaml` です。設定ファイルは、アプリケーションのリポジトリのルートディレクトリに自動的に追加されます。

料金

アプリケーションが使用するコンピューティングリソースとメモリリソースに対して課金されます。また、デプロイを自動化する場合は、1 か月間のすべての自動デプロイを提供する各アプリケーションに対して設定された月額料金も支払います。ソースコードからデプロイする場合は、App Runner がソースコードからコンテナを構築するのにかかる時間に対して、構築料金を支払います。

詳細については、[AWS App Runner の料金](#)を参照してください。

トピック

- [App Runner サービスの作成](#)
- [App Runner サービスの管理](#)

App Runner サービスの作成

AWS Explorer を使用して AWS Toolkit で App Runner サービスを作成できます。特定の でサービスを作成することを選択すると AWS リージョン、AWS Toolkit のコマンドペインに、アプリケーションが実行されるサービスインスタンスを設定する方法が示されます。

App Runner サービスを作成する前に、[前提条件](#)を満たしていることを確認してください。これには、関連する IAM 権限の提供と、デプロイする特定のソースリポジトリの確認が含まれます。

App Runner サービスを作成するには

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. App Runner ノードを右クリックして、[Create Service] (サービスの作成) を選択します。

AWS Toolkit コマンドペインが表示されます。

3. [Select a source code location type] (ソースコードの場所タイプを選択する) では、[ECR] または [Repository] (リポジトリ) を選択します。

[ECR] を選択した場合は、Amazon Elastic Container Registry が管理するリポジトリ内のコンテナイメージを指定します。[Repository] (リポジトリ) を選択した場合は、サポートされているリ

ポジトリプロバイダーが管理するソースコードリポジトリを指定します。現在、App Runner でソースコードリポジトリプロバイダーとしてサポートされているのは、[GitHub](#) です。

ECR からのデプロイ

1. [Select or enter an image repository] (イメージリポジトリを選択または入力する) では、Amazon ECR プライベートレジストリまたは Amazon ECR Public Gallery によって管理されるイメージリポジトリの URL を選択または入力します。

Note

Amazon ECR Public Gallery からリポジトリを指定する場合は、自動デプロイがオフになっていることを確認してください。App Runner は、ECR Public リポジトリ内のイメージに対する自動デプロイをサポートしていません。

自動デプロイはデフォルトでオフになっています。この場合、コマンドペインヘッダーのアイコンには斜線が表示されます。自動デプロイをオンにすると、このオプションには追加料金がかかることを示すメッセージが表示されます。

2. コマンドペインのステップに「No tags found」(タグが見つかりません) と表示された場合は、前のステップに戻って、タグ付けされたコンテナイメージを含むリポジトリを選択します。
3. [Port] (ポート) に、サービスが使用する IP ポート (ポート 8000 など) を入力します。
4. (オプション) [Configure environment variables] (環境変数の設定) で、サービスインスタンスの動作のカスタマイズに使用する環境変数が含まれているファイルを指定します。
5. Amazon ECR プライベートレジストリを使用している場合は、ECR アクセスロール (AppRunnerECRAccessRole) が必要です。このロールにより、App Runner は、アカウント内の Amazon Elastic Container Registry (Amazon ECR) イメージにアクセスできます。コマンドペインヘッダーの「+」アイコンを選択して、このロールを作成します。イメージの保存場所が、イメージを公開している Amazon ECR Public である場合、アクセスロールは必要ありません。
6. [Name your service] (サービスの名前) で、一意の名前を入力し、Enter を押します。名前にスペースを含めることはできません。
7. [Select instance configuration] (インスタンス設定の選択) で、サービスインスタンスの CPU ユニットとメモリ (両方とも GB) の組み合わせを選択します。

サービスの作成時に、そのステータスは [Creating] (作成中) から [Running] (実行中) に変わります。

8. サービスの実行が開始されたら、サービスを右クリックしてコンテキストメニューを開き、[Copy Service URL] (サービス URL をコピー) を選択します。
9. デプロイ済みのアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

リモートリポジトリからのデプロイ

1. 「接続を選択する」で、GitHub をリンクする接続を選択します AWS。選択可能な接続は、App Runner コンソールの [GitHub connections] (GitHub 接続) ページに表示されます。
2. [Select a remote GitHub repository] (リモート GitHub リポジトリの選択) では、リモートリポジトリの URL を選択または入力します。

AWS Cloud9 ソースコントロール管理で既に設定されているリモートリポジトリを選択できます。リポジトリがリストにない場合は、リポジトリへのリンクを貼り付けることもできます。

3. [Select a branch] (ブランチの選択) では、デプロイするソースコードの Git ブランチを選択します。
4. [Choose configuration source] (設定ソースの選択) では、ランタイム設定の定義方法を指定します。

[Use configuration file] (設定ファイルを使用) を選択すると、サービスインスタンスは `apprunner.yaml` 設定ファイルで定義された設定を使用します。このファイルは、アプリケーションのリポジトリのルートディレクトリにあります。

[Configure all settings here] (ここですべて設定する) を選択した場合は、コマンドペインを使用して、以下の項目を指定します。

- [Runtime] (ランタイム): [Python 3] または [Nodejs 12] を選択します。
 - [Build command] (ビルドコマンド): サービスインスタンスのランタイム環境でアプリケーションをビルドするコマンドを入力します。
 - [Start command] (開始コマンド): サービスインスタンスのランタイム環境でアプリケーションを開始するコマンドを入力します。
5. [Port] (ポート) に、サービスが使用する IP ポート (ポート 8000 など)を入力します。
 6. (オプション) [Configure environment variables] (環境変数の設定) で、サービスインスタンスの動作のカスタマイズに使用する環境変数が含まれているファイルを指定します。
 7. [Name your service] (サービスの名前) で、一意の名前を入力し、Enter を押します。名前にスペースを含めることはできません。

8. [Select instance configuration] (インスタンス設定の選択) では、サービスインスタンスの CPU ユニット数とメモリ (GB) を選択します。

サービスの作成中に、そのステータスは [Creating] (作成中) から [Running] (実行中) に変わります。
9. サービスの実行が開始されたら、サービスを右クリックしてコンテキストメニューを開き、[Copy Service URL] (サービス URL をコピー) を選択します。
10. デプロイ済みのアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

Note

App Runner サービスの作成に失敗すると、Explorer でのサービスのステータス表示が [Create failed] (作成に失敗しました) になります。トラブルシューティングの情報については、App Runner デベロッパーガイドの「[サービスの作成に失敗した場合](#)」を参照してください。

App Runner サービスの管理

App Runner サービスを作成したら、AWS Explorer ペインを使用して以下のアクティビティを実行して管理できます。

- [App Runner サービスの一時停止と再開](#)
- [App Runner サービスの展開](#)
- [App Runner のログストリームの表示](#)
- [App Runner サービスの削除](#)

App Runner サービスの一時停止と再開

ウェブアプリケーションを一時的に無効にしてコードの実行を停止する必要がある場合は、AWS App Runner サービスを一時停止できます。App Runner は、サービスのコンピューティング容量をゼロに削減します。アプリケーションを再度実行する準備ができたら、App Runner サービスを再開します。App Runner は、新しいコンピューティングキャパシティをプロビジョニングし、アプリケーションをデプロイして、アプリケーションを実行します。

Important

App Runner が稼働しているときにのみ課金されます。したがって、コストを管理するために、必要に応じてアプリケーションを一時停止および再開できます。これは、開発およびテストシナリオで特に役立ちます。

App Runner サービスを一時停止するには

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Pause] (一時停止) を選択します。
4. 表示されるダイアログボックスで、[Confirm] (確認) を選択します。

サービスが一時停止している間に、サービスのステータスは [Running] (実行中) から [Pausing] (一時停止中) に変わり、その後 [Paused] (一時停止) に変わります。

App Runner サービスを再開するには

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Resume] (再開) を選択します。

サービスが再開している間に、サービスのステータスは [Resuming] (再開中) から [Running] (実行中) に変わります。

App Runner サービスの展開

サービスの手動デプロイオプションを選択した場合は、サービスへの各デプロイを明示的に開始する必要があります。

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Start Deployment] (デプロイの開始) を選択します。
4. アプリケーションがデプロイされている間に、サービスのステータスは [[Deploying] (デプロイ中) から [Running] (実行中) に変わります。

5. アプリケーションが正常にデプロイされたことを確認するには、同じサービスを右クリックし、[Copy Service URL] (サービス URL のコピー) を選択します。
6. デプロイされたウェブアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

App Runner のログストリームの表示

CloudWatch Logs を使用して、App Runner などのサービスのログストリームをモニタリング、保存、およびアクセスできます。ログストリームは、同じソースを共有する一連のログイベントです。

1. App Runner を展開して、サービスインスタンスのリストを表示します。
2. 特定のサービスインスタンスを展開して、ロググループのリストを表示します。ロググループは、保持、監視、アクセス制御について同じ設定を共有するログストリームのグループです。
3. ロググループを右クリックし、[View Log Streams] (ログストリームの表示) を選択します。
4. コマンドペインで、グループからログストリームを選択します。

IDE AWS Cloud9 には、ストリームを構成するログイベントのリストが表示されます。古いイベントまたは新しいイベントをエディタにロードするかを選択できます。

App Runner サービスの削除

Important

App Runner サービスを削除すると、そのサービスは完全に削除され、保存されたデータは削除されます。サービスを再作成する必要がある場合は、App Runner がソースを再度取得し、コードリポジトリの場合はビルドする必要があります。ウェブアプリケーションは、新しい App Runner ドメインを取得します。

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Delete Service] (サービスの削除) を選択します。
4. Toolkit AWS コマンドペインで delete と入力し、Enter キーを押して確認します。

削除されるサービスのステータスには、[Deleting] (削除中) が表示され、その後このサービスはリストから削除されます。

AWS Toolkit を使用した AWS CloudFormation スタックの操作

Toolkit AWS は[AWS CloudFormation](#)スタックをサポートします。Toolkit を使用して AWS、CloudFormation スタックを削除できます。

CloudFormation スタックの削除

Toolkit を使用して AWS スタックを表示および削除できます CloudFormation。

前提条件

- AWS Cloud9 環境で使用している認証情報に、CloudFormation サービスへの適切な読み取り/書き込みアクセスが含まれていることを確認します。AWS Explorer で CloudFormation の下に「Error loading CloudFormation resources」(CloudFormation リソースの読み込みエラー) のようなメッセージが表示される場合は、これらの認証情報にアタッチされているアクセス許可をチェックしてください。アクセス許可に加えた変更は、AWS Explorer に反映されるまでに数分かかります。

CloudFormation スタックを削除するには

1. Explorer でAWS、削除する CloudFormation スタックのコンテキスト (右クリック) メニューを開きます。
2. [Delete CloudFormation Stack] (CloudFormation スタックを削除する) を選択します。
3. 表示されるメッセージで、[はい] を選択して削除を確認します。

スタックが削除されると、AWS Explorer に表示されなくなります。

Toolkit を使用した CloudWatch Logs の使用 AWS

Amazon CloudWatch Logs を使用して、すべてのシステム、アプリケーション、および AWS のサービス 使用する のログを、スケーラブルな単一のサービスに一元化できます。これにより、ログを簡単に表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。詳細については、Amazon CloudWatch ユーザーガイドの「[Amazon CloudWatch Logs とは](#)」を参照してください。

以下のトピックでは、AWS Toolkit を使用して AWS アカウントの CloudWatch Logs を操作する方法について説明します。

トピック

- [Toolkit を使用した CloudWatch ロググループとログストリームの表示 AWS](#)
- [ログストリームでの CloudWatch ログイベントの操作](#)

Toolkit を使用した CloudWatch ロググループとログストリームの表示 AWS

ログストリームは、同じソースを共有する一連のログイベントです。CloudWatch Logs のログのソースごとに、別個のログストリームとなります。

ロググループは、保持、モニタリング、アクセス制御について同じ設定を共有するログストリームのグループです。ロググループを定義して、各グループに入れるストリームを指定できます。1 つのロググループに追加できるログストリームの数に制限はありません。

詳細については、Amazon CloudWatch ユーザーガイドの「[ロググループとログストリームの操作](#)」を参照してください。

トピック

- [CloudWatch Logs ノードのロググループとログストリームの表示](#)

CloudWatch Logs ノードのロググループとログストリームの表示

1. Explorer がまだ開いていない場合は AWS 、開きます。
2. CloudWatch Logs ノードをクリックして、ロググループのリストを展開します。

現在の のロググループは AWS リージョン 、CloudWatch Logs ノードの下に表示されます。

3. 特定のロググループのログストリームを表示するには、ロググループの名前を右クリックしてコンテキストメニューを開き、[View Log Streams] (ログストリームの表示) を選択します。
4. ロググループのコンテンツは、[Select a log stream] (ログストリームの選択) の下に表示されます。

リストから特定のストリームを選択するか、フィールドにテキストを入力してストリームをフィルタリングできます。

ストリームを選択すると、そのストリーム内のイベントが IDE の [Log Streams] (ログストリーム) ウィンドウに表示されます。各ストリームのログイベントを操作する方法については、「[CloudWatch ログイベントの操作](#)」を参照してください。

ログストリームでの CloudWatch ログイベントの操作

[ログストリーム] ウィンドウを開くと、各ストリームのログイベントにアクセスできます。ログイベントは、モニタリングされているアプリケーションまたはリソースによって記録されたアクティビティのレコードです。

トピック

- [ログストリーム情報の表示とコピー](#)
- [ログストリームエディタのコンテンツをローカルファイルに保存します。](#)

ログストリーム情報の表示とコピー

ログストリームを開くと、[Log Stream] (ログストリーム) ウィンドウに、そのストリームのログイベントのシーケンスが表示されます。

1. 表示するログストリームを見つけるには、[Log Stream] (ログストリーム) ウィンドウを開きます。詳細については、「[CloudWatch ロググループとログストリームの表示](#)」を参照してください。

イベントをリストする各行にはタイムスタンプがつき、ログに記録された時刻を示します。

2. 次のオプションを使用して、ストリームのイベントに関する情報を表示およびコピーできます。
 - イベントを時間別に表示: [Load newer events] (新しいイベントのロード) または [Load older events] (古いイベントのロード) を選択して、最新のログイベントや古いログイベントを表示します。

Note

[Log Stream] (ログストリーム) エディタは、最新の 10,000 行のログイベントまたは 1 MB のログデータのいずれか小さい方のバッチを最初にロードします。[Load newer events] (新しいイベントのロード) を選択すると、最後のバッチをロードした後にログに記録されたイベントがエディタに表示されます。[Load older events] (古いイベントのロード) を選択すると、現在表示されているイベントの前に発生したイベントのバッチがエディタに表示されます。

- ログイベントのコピー: コピーするイベントを選択し、メニューを右クリックしてコンテキストメニューを開き、[Copy] (コピー) を選択します。

- ログストリーム名をコピー: [Log Stream] (ログストリーム) ウィンドウのタブを右クリックしてコンテキストメニューを開き、[Copy Log Stream Name] (ログストリーム名のコピー) を選択します。

ログストリームエディタのコンテンツをローカルファイルに保存します。

CloudWatch ログストリームエディタのコンテンツを log ローカルマシン上のファイルにダウンロードできます。

Note

このオプションを使用すると、ログストリームエディタに現在表示されているログイベントのみをファイルに保存できます。例えば、ログストリームの合計サイズが 5 MB で、エディタに 2 MB しかロードされていないとします。この場合、これを保存したファイルにも 2 MB のログデータしか含まれません。保存するデータをさらに表示するには、エディタで新しいイベントをロードまたは古いイベントをロードを選択します。

1. コピーするログストリームを検索するには、[Log Stream] (ログストリーム) ウィンドウを開きます (「[CloudWatch ロググループとログストリームの表示](#)」を参照)。
2. [Log Stream] (ログストリーム) ウィンドウのタブを右クリックしてコンテキストメニューを開き、[Save Current Log Content to File] (現在のログコンテンツをファイルに保存) を選択します。
3. ダイアログボックスを使用して、ログファイルのダウンロードフォルダを選択または作成し、[Save] (保存) を選択します。

Toolkit を使用した AWS Lambda 関数の操作 AWS

Toolkit は [AWS Lambda](#) 関数 AWS をサポートしています。Toolkit は、Lambda プラグインによって以前提供された機能を AWS 置き換えます AWS Cloud9。Toolkit を使用すると AWS、[サーバーレスアプリケーション](#)の一部である Lambda 関数のコードを作成できます。さらに、Lambda 関数をローカルまたは AWS で呼び出すことができます。

Lambda は、カスタムコードまたはさまざまな AWS のサービスで生成されるイベントに応答してコードを実行する、フルマネージドのコンピューティングサービスです。これらのサービスには、Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、Amazon Kinesis、Amazon Simple Notification Service (Amazon SNS)、Amazon Cognito などがあります。

⚠ Important

サーバーレスアプリケーションモデル (SAM) が提供するリソースを使用する Lambda アプリケーションを構築する場合は、「[Toolkit AWS SAM を使用した の操作 AWS](#)」を参照してください。

トピック

- [リモートの Lambda 関数を呼び出す](#)
- [Lambda 関数のダウンロード、アップロード、削除](#)

リモートの Lambda 関数を呼び出す

Toolkit AWS を使用すると、さまざまな方法で[AWS Lambda](#)関数を操作できます。

Lambda の詳細については、[デベロッパーガイドAWS Lambda](#) を参照してください。

i Note

AWS マネジメントコンソール または別の方法で を使用して Lambda 関数を既に作成しているとします。AWS ツールキットから呼び出すことができます。Toolkit を使用してデプロイ AWS できる新しい関数を作成するには AWS Lambda、まず[サーバーレスアプリケーションを作成](#)する必要があります。

前提条件

- で設定した認証情報に、AWS Lambda サービスへの適切な読み取り/書き込みアクセスが含まれていることを確認します。AWS Explorer の [Lambda] の下に、「Error loading Lambda resources」(Lambda リソースのロードエラー) のようなメッセージが表示される場合は、これらの認証情報にアタッチされているアクセス許可をチェックしてください。アクセス許可に加えた変更は、AWS ツールキットの AWS Explorer に反映されるまで数分かかります。

Lambda 関数の呼び出し

Important

AWS Toolkit を使用して API メソッドを呼び出すと、元に戻すことができないリソースが変更される可能性があります。たとえば、POST メソッドを呼び出すと、呼び出しが成功すれば API のリソースが更新されます。

Toolkit AWS を使用して、で Lambda 関数を呼び出すことができます AWS。

1. [AWS Explorer] で、呼び出したい Lambda 関数の名前を選択し、コンテキストメニューを開きます。
2. 呼び出しを選択します AWS。
3. 開いている [呼び出し関数] ウィンドウで、Lambda 関数が必要とするペイロードのオプションを選択します。(ペイロードは、入力として Lambda 関数に提供したい JSON です)。[Browse] (参照) を選択して、ペイロードとして使用するファイルを選択するか、ドロップダウンフィールドを使用してペイロードのテンプレートを選択します。この場合、テキストボックスに示されているように、Lambda 関数は、入力としての文字列として表示される可能性があります。

[Invoke] (呼び出し) を選択して Lambda を呼び出し、ペイロードで渡します。

AWS Lambda タブに Lambda 関数の出力が表示されます。

Lambda 関数のダウンロード、アップロード、削除

Toolkit AWS には、AWS Cloud9 IDE で Lambda 関数をインポートおよびアップロードするためのオプションが用意されています。

Lambda 関数のダウンロード

Lambda 関数をダウンロードすることで、関数を記述するプロジェクトファイルを AWS クラウドからダウンロードし、IDE AWS Cloud9 で操作することもできます。

Lambda 関数のダウンロード

1. AWS Explorer の [Lambda] ノードの下で、関数を右クリックしてコンテキストメニューを開き、[Download] (ダウンロード) を選択します。
2. 新しいプロジェクトの WorkSpace フォルダの選択を求められた時には、次のいずれかを試すことができます。
 - Lambda プロジェクトと同じ名前でサブフォルダを作成するために提案されたフォルダを選択します。
 - 別のフォルダを選択して、参照するダイアログボックスを開き、プロジェクトサブフォルダに別の親フォルダを選択します。

IDE で新しいエディタウィンドウが開きます。

ダウンロードされた Lambda 関数の実行とデバッグの設定

ダウンロードした Lambda 関数をサーバーレスアプリケーションとして実行およびデバッグするには、`launch.json` ファイルで起動設定を定義する必要があります。で作成された Lambda 関数は、起動設定に含まれない AWS マネジメントコンソール 場合があります。その場合は、手動で追加する必要があります。

起動設定に Lambda 関数を追加するには

1. Lambda 関数をダウンロードしたら、[Environment] (環境) ウィンドウを開いてフォルダとファイルを表示します。
2. 次に、Lambda 関数が `/home/ec2-user/.c9/launch.json` ファイルに含まれていることを確認します。存在しない場合は、次の操作を実行して、関数のコードに CodeLens リンクを追加します。
 1. Lambda 関数を定義するソースコードファイル (`.js` ファイルや `.py` ファイルなど) を開きます。次に、Lambda 関数を `launch.json` ファイルに追加するために使用できる CodeLens リンクがあるかどうかを確認します。CodeLens は関数の上に表示され、Add Debug Config リンクが含まれています。
 2. IDE の左側にある [Go] (移動) (虫眼鏡アイコン) を選択し、「sam hint」と入力して AWS: Toggle SAM hints in source files コマンドを表示します。コマンドを選択して実行します。
 3. Lambda ソースコードファイルを閉じて、再度開きます。

4. ファイルを再度開いた後で CodeLens がソースコードで使用可能である場合は、Add Debug Config を選択して起動設定を追加します。
 3. SAM ヒントオプションを切り替えた後でも CodeLens を追加できない場合は、次の手順を実行して起動設定を追加します。
 1. IDE の左側にある [Go] (移動) (虫眼鏡アイコン) を選択し、「config」と入力して AWS: SAM Debug Configuration Editor コマンドを表示します。コマンドを選択して実行します。
 2. [SAM Debug Configuration Editor] (SAM デバッグ設定エディタ) が表示されます。このエディタを使用して、起動設定のプロパティを定義できます。詳細については、「[サーバーレスアプリケーションの実行とデバッグのため、SAM テンプレートを使用](#)」で「[configuring launch properties](#)」のステップを参照してください。
-  **Note**

Lambda 関数に SAM アプリケーション用の `template.yaml` がない場合は、これを追加する必要があります。詳細については、「[AWS SAM テンプレートの作成](#)」を参照してください。
3. エディタで必要な設定情報の入力を完了すると、起動設定が `launch.json` ファイルに追加されます。

Lambda 関数の起動設定を定義したら、次の手順に従って実行できます。

1. IDE の上部で、[Auto] (自動) の横にある矢印を選択し、該当する起動設定を選択します。
2. 続いて、[Run] (実行) を選択します。

Lambda 関数のアップロード

既存の Lambda 関数をローカルコードで更新できます。この方法でコードを更新しても、デプロイに AWS Serverless Application Model CLI は使用されず、AWS CloudFormation スタックも作成されません。これにより、Lambda でサポートされている任意のランタイムを使って Lambda 関数をアップロードできます。

AWS Toolkit を使用して Lambda 関数をアップロードするには、いくつかのインターフェイスオプションがあります。

[環境] ウィンドウまたは [コマンドペイン] からアップロードする

1. プロジェクトファイルの [Environment] (環境) ウィンドウで、アップロードする Lambda アプリケーションの `template.yaml` を右クリックしてコンテキストメニューを開き、[Upload Lambda] (Lambda のアップロード) を選択します。

または、Ctrl+P を押して [どこにでも移動] ペインを開き、「lambda」と入力して [AWS Lambda のアップロード] コマンドにアクセスします。続いて、このコマンドを選択してアップロードプロセスを開始します。

2. 次に、アップロード AWS リージョン する を選択します。
3. ここで、Lambda 関数をアップロードするためのオプションを選択します。

.zip アーカイブをアップロード

1. メニューから [ZIP アーカイブ] を選択します。
2. ファイルシステムから .zip AWS Cloud9 ファイルを選択し、開く を選択します。

ディレクトリをそのままアップロード

1. メニューから [ディレクトリ] を選択します。
2. AWS Cloud9 ファイルシステムからディレクトリを選択し、開くを選択します。
4. イベントを処理する Lambda 関数ハンドラーを指定します。関数が呼び出されると、Lambda はこのハンドラーメソッドを実行します。

Note

Lambda 関数を選択するときは、表示されるリストから選択できます。選択する関数がわからない場合は、ツールキットで利用可能な Lambda 関数の Amazon リソース番号 (ARN) を入力できます。

このコードを Lambda 関数の最新バージョンとして公開するかどうかを尋ねるダイアログが表示されます。[はい] を選択して、公開を確認します。

Note

フォルダの親フォルダを右クリックしてコンテキストメニューを開き、[Upload Lambda] (Lambda のアップロード) を選択して、Lambda アプリケーションをアップ

ロードすることもできます。親フォルダがアップロードのために自動的に選択されます。

AWS Explorer からアップロードする

1. AWS Explorer で、インポートする Lambda 関数の名前を右クリックしてコンテキストメニューを開きます。
2. [Lambda のアップロード] を選択します。
3. Lambda 関数をアップロードするための 3 つのオプションから選択します。

事前に作成された .zip アーカイブをアップロード

1. メニューから [ZIP アーカイブ] を選択します。
2. ファイルシステムから .zip AWS Cloud9 ファイルを選択し、開く を選択します。
3. モーダルダイアログでアップロードを確認します。こうして、.zip ファイルがアップロードされ、デプロイ後にすぐに Lambda が更新されます。

ディレクトリをそのままアップロード

1. メニューから [ディレクトリ] を選択します。
2. AWS Cloud9 ファイルシステムからディレクトリを選択し、開くを選択します。
3. ディレクトリの構築を求められたら、[いいえ] を選択します。
4. モーダルダイアログでアップロードを確認します。これにより、ディレクトリがそのままアップロードされ、デプロイ後にすぐに Lambda が更新されます。

ディレクトリの構築とアップロード

1. メニューから [ディレクトリ] を選択します。
2. AWS Cloud9 ファイルシステムからディレクトリを選択し、開くを選択します。
3. ディレクトリの構築を求められたら、[はい] を選択します。
4. モーダルダイアログでアップロードを確認します。これにより、AWS SAM CLI `sam build` コマンドを使用して ディレクトリにコードが構築され、デプロイ後すぐに Lambda が更新されます。

リモートアクセス用の Lambda 関数のデプロイ

ローカル関数をサーバーレス SAM アプリケーションとしてデプロイすることで、リモートで使用可能にすることができます。

Lambda 関数を SAM アプリケーションとしてデプロイするには

1. [AWS Explorer] ウィンドウで、[Lambda] ノードを右クリックしてコンテキストメニューを開き、[Deploy SAM Application] (SAM アプリケーションのデプロイ) を選択します。
2. コマンドペインで、関数をサーバーレスアプリケーションとして定義する [\[YAML テンプレート\]](#) を選択します。
3. 次に、Lambda デプロイ用の Amazon S3 バケットを選択します。デプロイ用のバケットを作成することもできます。
4. 次に、デプロイ先の CloudFormation スタックの名前を入力します。既存のスタックを指定すると、コマンドはスタックを更新します。新しいスタックを指定すると、コマンドはスタックを作成します。

スタックの名前を入力すると、Lambda 関数が SAM アプリケーションとしてデプロイを開始します。デプロイが成功すると、SAM Lambda アプリケーションをリモートで使えるようになります。これにより、他の AWS Cloud9 開発環境からダウンロードまたは呼び出すことができます。

Lambda 関数をゼロから作成する場合は、「[Toolkit を使用してサーバーレスアプリケーションを作成する AWS](#)」のステップに従うことをお勧めします。

Lambda 関数の削除

同じコンテキスト (右クリック) メニューを使用して Lambda 関数を削除することもできます。

Warning

この手順は、[CloudFormation](#) に関連付けられている Lambda 関数の削除には使用しないでください。例えば、このガイドで前述した[サーバーレスアプリケーションの作成](#)時に作成した Lambda 関数を削除しないでください。これらの関数は、CloudFormation スタックを通じて削除する必要があります。

1. AWS Explorer で、削除したい Lambda 関数の名前を選択し、そのコンテキスト (右クリック) メニューを開きます。
2. [削除] を選択します。
3. 表示されるメッセージで、[はい] を選択して削除を確認します。

関数が削除されると、AWS Explorer に表示されなくなります。

リソースの使用

AWS Explorer にデフォルトでリスト AWS のサービス されている へのアクセスに加えて、リソースに移動し、数百のリソースから選択してインターフェイスに追加できます。では AWS、リソースは操作できるエンティティです。追加されるリソースには、Amazon AppFlow、Amazon Kinesis Data Streams、AWS IAM ロール、Amazon VPC、Amazon CloudFront ディストリビューションなどがあります。

使用可能なリソースを表示するには、[Resources] (リソース) を選択し、リソースタイプを展開して、そのタイプで使用可能なリソースを一覧表示します。たとえば、AWS::Lambda::Function リソースタイプを選択すると、さまざまな関数、そのプロパティ、および属性を定義するリソースにアクセスできます。

リソースタイプを [Resources] (リソース) に追加すると、次の方法でリソースタイプとそのリソースを操作できます。

- このリソースタイプ AWS リージョン で現在の で利用可能な既存のリソースのリストを表示します。
- リソースを記述する JSON ファイルの読み取り専用バージョンを表示します。
- リソースのリソース識別子をコピーします。
- リソースタイプの目的と、リソースをモデリングするためのスキーマ (JSONおよび YAML形式) を説明する AWS ドキュメントを表示します。

リソースにアクセスするための IAM アクセス許可

に関連付けられたリソースにアクセスするには、特定のアクセス AWS Identity and Access Management 許可が必要です AWS のサービス。例えば、IAM エンティティ (ユーザーまたはロールなど) は、AWS::Lambda::Function リソースにアクセスするために Lambda アクセス許可を必要とします。

サービスリソースのアクセス許可に加えて、IAM エンティティには、AWS Toolkit が AWS Cloud Control API オペレーションを呼び出すことを許可するアクセス許可が必要です。Cloud Control API オペレーションでは、IAM ユーザーまたはロールがリモートリソースにアクセスして更新できます。

Toolkit インターフェイスを使用してこれらの API オペレーションを呼び出す IAM エンティティに AWS 管理ポリシー PowerUserAccess をアタッチすることで、アクセス許可をすばやく付与できます。このマネージドポリシーでは、API オペレーションの呼び出しなど、アプリケーション開発タスクを実行するために一定の範囲のアクセス許可が付与されます。

リモートリソースで使用可能な API オペレーションを定義する特定のアクセス許可については、「AWS クラウドコントロール API ユーザーガイド」<https://docs.aws.amazon.com/cloudcontrolapi/latest/userguide/security.html>を参照してください。

既存のリソースの操作

1. AWS Explorer で、[Resources] (リソース) を選択します。

リソースタイプのリストは、[Resources] (リソース) ノードの下に表示されます。

2. リソースタイプのテンプレートを定義する構文について説明したドキュメントがあります。このドキュメントにアクセスするには、そのリソースタイプを右クリックしてコンテキストメニューを開き、[View Documentation] (ドキュメントの表示) を選択します。

Note

ドキュメントページにアクセスできるように、ブラウザのポップアップブロッカーをオフにするよう求められることがあります。

3. リソースタイプに既に存在するリソースを表示するには、そのタイプのエントリを展開します。

使用可能なリソースのリストが、リソースタイプの下に表示されます。

4. 特定のリソースを操作するには、その名前を右クリックしてコンテキストメニューを開き、以下のいずれかのオプションを選択します。

- [Copy Identifier] (識別子をコピー): 特定のリソースの識別子をクリップボードにコピーします。たとえば、AWS::DynamoDB::Table リソースは TableName プロパティを使用して識別できます。
- [Preview] (プレビュー): リソースを記述する JSON 形式のテンプレートの読み取り専用バージョンを表示します。

Toolkit を使用した Amazon S3 の使用 AWS

以下のトピックでは、AWS Toolkit を使用して [Amazon S3](#) バケットとオブジェクトを操作する方法について説明します AWS アカウント。

トピック

- [Amazon S3 バケットの操作](#)
- [Amazon S3 オブジェクトの操作](#)

Amazon S3 バケットの操作

Amazon S3 に保存したオブジェクトはすべて、バケット内にあります。ファイルシステムでディレクトリを使用してファイルをグループ化するのと同じ方法で、バケットを使用して関連するオブジェクトをグループ化できます。

トピック

- [Amazon S3 バケットの作成](#)
- [Amazon S3 バケットにフォルダを追加](#)
- [Amazon S3 バケットの削除](#)
- [Amazon S3 アイテムの表示の設定](#)

Amazon S3 バケットの作成

1. AWS Explorer で、[S3] ノードを右クリックしてコンテキストメニューを開き、[Create Bucket] (バケットの作成) を選択します。
2. [バケット名] フィールドに、バケットの有効な名前を入力します。[入力] を押して確認します。

新しいバケットが S3 ノードの下で表示されます。

メモ

S3 バケットは、パブリックにアクセス可能な URL として使用できるため、グローバルに一意的なバケット名を選択する必要があります。他のアカウントで同じ名前のバケットがすでに作成されている場合は、別の名前を使用する必要があります。バケットを作成できない場合は、[出力] タブの [AWS ツールキットのログ] をチェックできます。例えば、既に使用されているバケット名を使用する

と、BucketAlreadyExists エラーが発生します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バケットの制約と制限](#)」を参照してください。

バケットを作成した後、バケット名と Amazon リソースネーム (ARN) をクリップボードにコピーできます。バケットエントリを右クリックしてコンテキストメニューを開き、メニューから関連するオプションを選択します。

Amazon S3 バケットにフォルダを追加

フォルダーのオブジェクトをグループ化して、バケットのコンテンツを整理します。他のフォルダ内にフォルダを作成することもできます。

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを表示します。
2. バケットまたはフォルダを右クリックしてコンテキストメニューを開き、[Create Folder] (フォルダの作成) を選択します。
3. フォルダ名を入力し、Enter を押します。

[AWS Explorer] ウィンドウで、選択したバケットおよびフォルダの下に、新しいフォルダが表示されます。

Amazon S3 バケットの削除

バケットを削除すると、バケットに含まれるフォルダとオブジェクトも削除されます。バケットの削除前に、削除を確認するメッセージが表示されます。

Note

フォルダのみを削除するには、バケット全体ではなく、AWS マネジメントコンソールを使用します。

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを展開します。
2. 削除するバケットのコンテキストメニューを開き、[削除] を選択します。
3. バケット名を入力して削除を確認し、Enter を押します。

 Note

バケットにオブジェクトが含まれている場合は、バケットを空にしてから削除します。何千ものオブジェクトのすべてのバージョンを削除する必要がある場合は、時間がかかる可能性があります。削除プロセスが完了すると、通知が表示されます。

Amazon S3 アイテムの表示の設定

多数の Amazon S3 オブジェクトまたはフォルダを使用している場合は、一度に表示する数を指定すると便利です。最大数が表示されているときは、[Load More] (さらにロード) を選択して、次のバッチを表示できます。

1. メニューバーで、AWS Cloud9、[設定] を選択します。
2. [設定] ウィンドウで、[プロジェクト設定] を拡張し、[拡張機能] セクションをクリックして [AWS 設定] を選択します。
3. [AWS Configuration (設定)] ペインで、[AWS > S3: ページあたりの最大項目数] 設定に移動します。
4. さらにロードすることを選択する前に、デフォルト値を、表示したい S3 項目の数に変更します。

 Note

許容値の範囲は 3 ~ 1000 です。この設定は、一度に表示されるオブジェクトまたはフォルダの数にのみ適用されます。作成したすべてのバケットが一度に表示されます。デフォルトでは、AWS アカウントにつき最大で 100 個のバケットを作成できます。

Amazon S3 オブジェクトの操作

オブジェクトとは、Amazon S3 に保存される基本エンティティです。オブジェクトは、オブジェクトデータとメタデータで構成されます。

トピック

- [Amazon S3 バケットにファイルをアップロード](#)
- [Amazon S3 オブジェクトのダウンロード](#)

- [Amazon S3 オブジェクトの削除](#)
- [Amazon S3 オブジェクトの署名付き URL の生成](#)

Amazon S3 バケットにファイルをアップロード

ツールキットインターフェイスまたはコマンドを使用して、バケットにファイルをアップロードできます。

どちらの方法でも、ユーザーの環境からファイルをアップロードし、AWS クラウドに S3 オブジェクトとして保存できます。ファイルは、バケットまたはバケットのコンテンツを整理するフォルダにアップロードできます。

インターフェイスを使用して S3 バケットにファイルをアップロード

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを表示します。
2. バケットまたはバケット内のフォルダのコンテキストメニュー (右クリック) を開き、[ファイルのアップロード] を選択します。

メモ

S3 オブジェクトのコンテキストメニューを開く (右クリック) 場合は、[親にアップロード] を選択できます。これで、選択したファイルを含むフォルダまたはバケットへのファイル追加が有効になります。

3. 環境のファイルマネージャを使用してファイルを選択し、アップロードを選択します。

選択したファイルは S3 オブジェクトとしてバケットまたはフォルダにアップロードされます。各オブジェクトのエントリは、保存したオブジェクトのサイズと更新したのはどれくらい前かを説明します。オブジェクトのリスト上で一時停止して、パス、サイズ、およびオブジェクトが最後に変更された時刻を表示できます。

コマンドを使用して S3 バケットに現在のファイルをアップロードします。

1. アップロードするファイルを選択するには、ファイルのタブを選択します。
2. Ctrl+Pを押してコマンドペインを表示します。
3. Go To Anything (どこにでも移動) の場合、フレーズ `upload file` の入力をスタートして `AWS: Upload File` コマンドを表示します。表示されたら、コマンドを選択します。

4. ステップ 1: アップロードするファイルの選択の場合、選択したファイルを選択するか、別のファイルを参照できます。
5. ステップ 2: アップロードする S3 バケットを選択する場合、リストからバケットを選択します。

選択したファイルは S3 オブジェクトとしてバケットまたはフォルダにアップロードされます。各オブジェクトのエントリは、保存したオブジェクトのサイズと更新したのはどれくらい前かを説明します。オブジェクトのリスト上で一時停止して、パス、サイズ、およびオブジェクトが最後に変更された時刻を表示できます。

Amazon S3 オブジェクトのダウンロード

Amazon S3 バケット内のオブジェクトは、AWS クラウドから AWS Cloud9 環境内のフォルダにダウンロードできます。

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを表示します。
2. バケットまたはバケット内のフォルダで、オブジェクトのコンテキストメニュー (右クリック) を開き、[Download as (形式を指定してダウンロード)] を選択します。
3. 環境のファイルマネージャを使用して、送信先フォルダを選択し、ファイル名を入力し、[ダウンロード] を選択します。

ファイルがダウンロードされたら、AWS Cloud9内で開くことができます。

Amazon S3 オブジェクトの削除

バージョニング非対応バケットにある場合、オブジェクトは完全に削除できます。ただし、バージョニングが有効なバケットの場合、削除リクエストはそのオブジェクトを完全に削除しません。代わりに、Amazon S3 はバケットに削除マーカを挿入します。詳細については、[Amazon Simple Storage Service ユーザーガイド]の「[オブジェクトバージョンの削除](#)」を参照してください。

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを表示します。
2. バケットまたはバケット内のフォルダで、オブジェクトのコンテキストメニュー (右クリック) を開き、[削除] を選択します。
3. 削除を確認するには、[Delete] (削除) を選択します。

Amazon S3 オブジェクトの署名付き URL の生成

署名付き URL を使用すると、オブジェクトの所有者はオブジェクトをダウンロードするための期限付きのアクセス許可を付与することにより、プライベート Amazon S3 オブジェクトを共有できます。詳細については、「Amazon S3 ユーザーガイド」の「[署名付き URL を使用したオブジェクトの共有](#)」を参照してください。

1. [AWS Explorer] で、S3 ノードを選択して、バケットのリストを表示します。
2. バケット内またはバケット内のフォルダで、オブジェクトを右クリックし、[Generate Presigned URL] (署名付き URL の生成) を選択します。
3. Toolkit AWS コマンドペインで、URL を使用してオブジェクトにアクセスできる分数を入力します。Enter を押して確認します。

IDE の下部のステータスに、このオブジェクト用の署名付き URL がクリップボードにコピーされたことが示されます。

Toolkit AWS SAM を使用した の操作 AWS

Toolkit AWS は、[サーバーレスアプリケーション](#)をサポートします。Toolkit を使用すると AWS 、[AWS Lambda](#)関数を含むサーバーレスアプリケーションを作成し、アプリケーションを AWS CloudFormation スタックにデプロイできます。

サーバーレスアプリケーションの作成

この例では、AWS Toolkit を使用してサーバーレスアプリケーションを作成する方法を示します。サーバーレスアプリケーションの実行とデバッグの詳細については、「[サーバーレスアプリケーションの実行とデバッグ](#)」を参照してください。

サーバーレスアプリケーションを作成するために必要な前提条件には、AWS SAM CLI と AWS CLI が含まれます。これらは に含まれています AWS Cloud9。CLI AWS SAM がインストールされていない場合、または古い場合は、インストールまたはアップグレードの実行が必要になる場合があります。AWS SAM CLI をインストールする方法については、[AWS SAM 「CLI のインストール」](#)を参照してください。また、CLI AWS SAM をアップグレードする方法については、[AWS SAM 「CLI のアップグレード」](#)を参照してください。

Toolkit を使用してサーバーレスアプリケーションを作成する AWS

この例では、[AWS Serverless Application Model \(AWS SAM\)](#) を使用して AWS Toolkit でサーバーレスアプリケーションを作成する方法を示します。

1. [AWS Explorer] で [Lambda] ノードを右クリックしてコンテキストメニューを開き、[Create Lambda SAM Application] (Lambda SAM アプリケーションの作成) を選択します。

 Note

または、[AWS: Explorer] のリストからメニューアイコンを選択し、[Create Lambda SAM Application] (Lambda SAM アプリケーションの作成) を選択します。

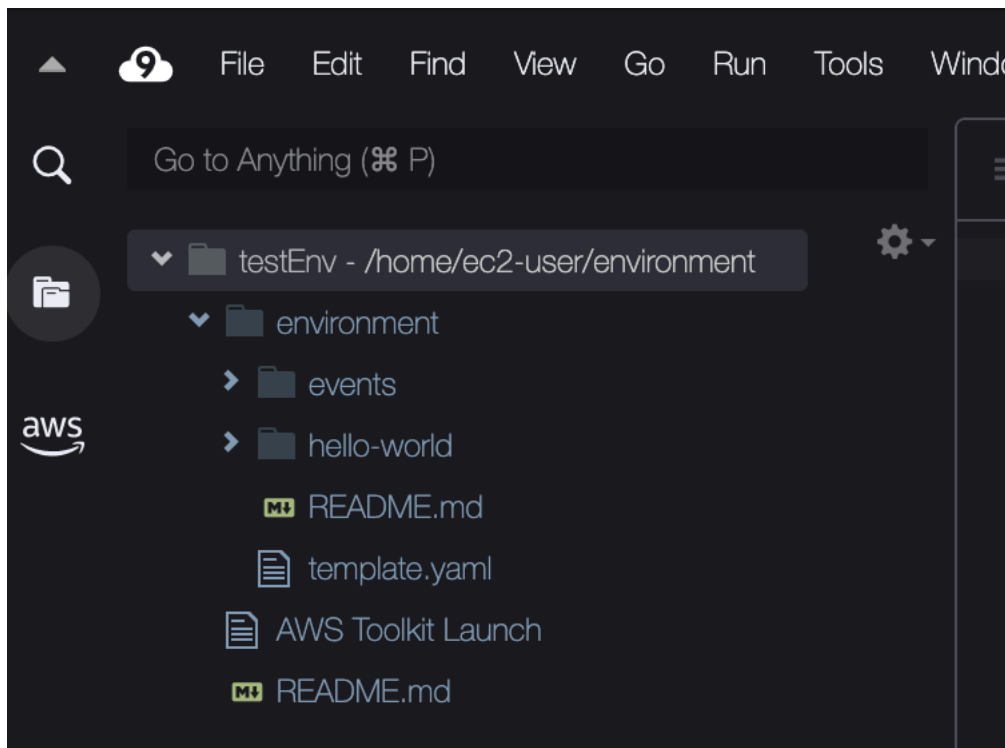
2. SAM アプリケーションのランタイムを選択します。この例では、[nodejs12.x] を選択します。

 Note

「(Image)」を含むランタイムの 1 つを選択した場合、アプリケーションのパッケージタイプは Image です。「(Image)」を含まないランタイムの 1 つを選択した場合、アプリケーションは Zip タイプです。Image および Zip パッケージタイプの違いの詳細については、AWS Lambda デベロッパーガイドの「[Lambda デプロイパッケージ](#)」を参照してください。

3. サーバーレスアプリケーションに以下のいずれかのテンプレートを選択します。
 - AWS SAM Hello World: 従来の「Hello World」メッセージを返す Lambda 関数を含む基本的なテンプレート。
 - AWS Step Functions サンプルアプリ: 株式取引ワークフローを実行するサンプルアプリケーション。ステップ関数は、関与する Lambda 関数の相互作用をオーケストレーションします。
4. 新しいプロジェクトの場所を選択します。既存のワークスペースフォルダがある場合は、これを選択できます。それ以外の場合は、別のフォルダを参照して選択します。[Select a different folder] (別のフォルダを選択) を選択すると、ダイアログボックスが表示され、フォルダの場所を選択できます。
5. 新しいアプリケーションの名前を入力します。この例では my-sam-app-nodejs を使用します。Enter キーを押すと、AWS Toolkit は少し時間を取ってプロジェクトを作成します。

プロジェクトが作成されると、[環境] ウィンドウにアプリケーションのファイルを表示できます。[Explorer] ウィンドウに表示されていることを確認します。



サーバーレスアプリケーションの実行とデバッグ

AWS Toolkit を使用して、サーバーレスアプリケーションをデバッグし、開発環境でローカルで実行する方法を設定できます。AWS Serverless Application Model (AWS SAM) テンプレートで定義されたサーバーレスアプリケーションをデバッグできます。このテンプレートは、シンプルな YAML 構文を使用して、サーバーレスアプリケーションを構成する関数、API、データベース、API、データベース、イベントソースのマッピングなどのリソースを記述します。

AWS SAM テンプレートの詳細については、「デベロッパーガイド」の[AWS SAM 「テンプレートの構造」](#)を参照してください。AWS Serverless Application Model

また、SAM テンプレートにコミットされていないサーバーレスアプリケーションを迅速にデバッグすることもできます。

インラインアクションを使用して対象となる AWS Lambda 関数を識別することで、デバッグ動作の設定を開始します。SAM テンプレートによって定義されたインフラストラクチャを使用するには、関連する YAML 形式のファイルでインラインアクションを使用します。テンプレートなしで関数を直接テストするには、アプリケーションファイルの Lambda ハンドラーのコンテキストに応じたリンクを使用します。

Note

この例では、JavaScript を使用するアプリケーションをデバッグします。ただし、AWS Toolkit で使用できるデバッグ機能は、次の言語とランタイムで使用できます。

- JavaScript – Node.js 10.x, 12.x, 14.x
- Python – 3.7、3.8、3.9、3.10 (Python 2.7 および 3.6 サーバーレスアプリケーションは実行できますが、AWS Toolkit によってデバッグすることはできません)。

言語の選択は、コンテキストに応じたリンクが適格な Lambda ハンドラーを示す方法にも影響します。詳細については、「[コードからサーバーレス関数を直接実行およびデバッグ](#)」を参照してください。

サーバーレスアプリケーションの実行とデバッグのため、SAM テンプレートを使用

SAM テンプレートを使用して実行およびデバッグされるアプリケーションの場合、YAML 形式のファイルには、アプリケーションの動作と使用するリソースが記述されます。AWS Toolkit を使用してサーバーレスアプリケーションを作成すると、`template.yaml` という名前のファイルがプロジェクト用に自動的に生成されます。

この手順では、[サーバーレスアプリケーションの作成](#) で作成したサンプルアプリケーションを使用します。

SAM テンプレートを使用してサーバーレスアプリケーションを実行およびデバッグ

1. サーバーレスアプリケーションを構成するアプリケーションファイルを表示するには、[環境] ウィンドウに移動します。
2. アプリケーションフォルダ (例:my-sample-app) で `template.yaml` ファイルを開きます。
3. `template.yaml` で、[Edit Launch Configuration] (起動設定の編集) を選択します。

新しいエディタに表示される `launch.json` ファイルは、デフォルト属性があるデバッグ設定を提供します。

4. 次の設定プロパティの値を編集または確認します。

- "name" – 読みやすい名前を入力して、[実行] ビューの [設定] ドロップダウンフィールドに表示します。

- "target" – 値が "template" であることを確認します。これにより、SAM テンプレートがデバッグセッションのエントリポイントになります。
- "templatePath" – template.yaml ファイルに相対パスまたは絶対パスを入力。
- "logicalId" – 名前が SAM テンプレートの [Resources] (リソース) セクションに指定されている名前と一致することを確認します。この場合はタイプ `AWS::Serverless::Function` の `HelloWorldFunction` です。

launch.json ファイル内のこれらと他の入力に関する詳細は、「[サーバーレスアプリケーションのデバッグ用設定オプション](#)」を参照してください。

5. デバッグ設定に問題がなければ、launch.json を保存します。次に、RUN の横の緑色の [play] (再生) ボタンを選択してデバッグを開始します。

 Note

SAM アプリケーションの実行に失敗した場合は、[出力] ウィンドウをチェックして、Docker イメージが構築されていないためにエラーが発生しているかどうかを確認します。環境でディスク容量の解放が必要になる場合があります。

詳細については、「[AWS Cloud9 環境に十分なディスク容量がないため、AWS Toolkit で SAM アプリケーションをローカルで実行中にエラーが発生しました](#)」を参照してください。

デバッグセッションを開始すると、[DEBUG CONSOLE] (デバッグコンソール) パネルにデバッグ出力が表示され、Lambda 関数から返された値が表示されます。SAM アプリケーションをデバッグする場合、AWS ツールキットを [Output] (出力) パネルの [Output] (出力) チャンネルとして選択します。

 Note

Windows ユーザーの場合、このプロセス中に Docker マウントエラーが表示されたら、[Docker Settings] (Docker 設定) で共有ドライブの認証情報を更新する必要があります。Docker マウントエラーは次のように表示されます。

```
Fetching lambci/lambda:nodejs10.x Docker container image.....
2019-07-12 13:36:58 Mounting C:\Users\<username>\AppData\Local\Temp\ ...
as /var/task:ro,delegated inside runtime container
Traceback (most recent call last):
```

```
...requests.exceptions.HTTPError: 500 Server Error: Internal Server Error ...
```

コードからサーバーレス関数を直接実行およびデバッグ

AWS SAM アプリケーションをテストするときは、Lambda 関数のみを実行およびデバッグすることを選択できます。SAM テンプレートに定義されている他のリソースは除外します。このアプローチでは、インラインアクションを使用して、直接呼び出すことができる、ソースコード内の Lambda 関数ハンドラーを特定します。

コンテキスト対応リンクによって検出される Lambda ハンドラーは、アプリケーションで使っている言語とランタイムによって異なります。

言語/ランタイム	Lambda 関数をコンテキスト対応リンクで特定するための条件
JavaScript(Node.js 10.x、12.x、14.x)	関数には以下の特徴があります。 - 最大 3 つのパラメータがあるエクスポートされた関数です。 - WorkSpaces フォルダー内の親フォルダーに <code>package.json</code> ファイルがあります。
Python (3.7、3.8、3.9、3.10)	関数には以下の特徴があります。 - 上位レベルの関数です。 - Workspace フォルダー内の親フォルダーに <code>requirements.txt</code> ファイルがあります。

サーバーレスアプリケーションをアプリケーションコードから直接実行およびデバッグ

- サーバーレスアプリケーションのファイルを表示するには、エディタの横にあるフォルダアイコンを選択して、アプリケーションフォルダに移動します。
- アプリケーションフォルダ (my-sample-app など) で、関数フォルダ (この例では hello-world) を展開し、app.js ファイルを開きます。

3. 対象の Lambda ハンドラー関数を識別するインラインアクションで、Add Debug Configuration を選択します。[Add debug configuration] (デバッグ設定の追加) オプションが表示されない場合は、コードレンズを有効にする必要があります。コードレンズを有効にするには、「[the section called “AWS Toolkit コードレンズの有効化”](#)」を参照してください。Lambda 関数ハンドラーのインラインアクション内の [Add Debug Configuration] オプションにアクセスします。
4. SAM アプリケーションを実行するランタイムを選択します。
5. launch.json ファイルのエディターでは、次の設定プロパティの値を編集または確認します。
 - "name" – 分かりやすい名前を入力します。
 - "target" – 値が "code" で、Lambda 関数ハンドラーを直接呼び出せることを確認します。
 - "lambdaHandler" – 関数を呼び出すために Lambda が呼び出すコード内のメソッドの名前を入力します。例えば、JavaScript のアプリケーションでは、デフォルトは app.lambdaHandler です。
 - "projectRoot" – Lambda 関数を含むアプリケーションファイルにパスを入力します。
 - "runtime" – Lambda 実行環境の有効なランタイム ("nodejs.12x" など) を入力または確認します。
 - "payload" – 以下のいずれかのオプションを選択して、Lambda 関数に入力として提供するイベントペイロードを定義します。
 - "json": イベントペイロードを定義する JSON 形式のキーバリューペア。
 - "path": イベントペイロードとして使用されるファイルへのパス。
6. デバッグ設定が満足なものであれば、[RUN] の横の緑の再生矢印を選択して、デバッグをスタートします。

デバッグセッションが開始すると、[DEBUG CONSOLE] (デバッグコンソール) パネルにデバッグ出力が表示され、Lambda 関数から返された値が表示されます。SAM アプリケーションをデバッグする場合、AWS ツールキットを [Output] (出力) パネルの [Output] (出力) チャンネルとして選択します。

 Note

エラーメッセージに Docker が記載されている場合は、この[注記](#)を参照してください。

ローカル Amazon API Gateway リソースの実行とデバッグ

で指定された AWS SAM API Gateway ローカルリソースを実行またはデバッグできます `template.yaml`。これを行うには、`type=aws-sam`での AWS Cloud9 起動設定を実行します `invokeTarget.target=api`。

Note

API Gateway は、2 種類の API をサポートしています。REST API と HTTP API です。ただし、AWS Toolkit の API Gateway 機能は REST APIs のみをサポートします。時に、HTTP API は「API Gateway V2 API」と呼ばれます。

ローカル API Gateway リソースを実行およびデバッグ

1. 以下のいずれかのアプローチを選択し、AWS SAM API Gateway リソース用の起動設定を作成します。
 - オプション 1: AWS SAM プロジェクトのハンドラーソースコード (特に .js、.cs、または .py ファイル) にアクセスし、Lambda ハンドラーにカーソルを合わせて、[Add debug configuration] (デバッグ設定の追加) を選択します。[Add debug configuration] (デバッグ設定の追加) オプションが表示されない場合は、コードレンズを有効にします。コードレンズを有効にするには、「[the section called “AWS Toolkit コードレンズの有効化”](#)」を参照してください。次に、メニューで API イベントとマークされた項目を選択します。
 - オプション 2 `launch.json` を編集して、次の構文を使用して新しい起動設定を作成します。

```
{
  "type": "aws-sam",
  "request": "direct-invoke",
  "name": "myConfig",
  "invokeTarget": {
    "target": "api",
    "templatePath": "n12/template.yaml",
    "logicalId": "HelloWorldFunction"
  },
  "api": {
    "path": "/hello",
    "httpMethod": "post",
    "payload": {
      "json": {}
    }
  }
}
```

```
    }  
  },  
  "sam": {},  
  "aws": {}  
}
```

2. [Run] (実行) ボタンの横のドロップダウンメニューで、起動設定を選択します (上の例では myConfig という名前です)。
3. (オプション) Lambda プロジェクトコードにブレークポイントを追加します。
4. 緑の「再生」ボタンの横にある [Run (実行)] ボタンを選択します。
5. 出力ペインで、結果を表示します。

設定

invokeTarget.target プロパティ値 api を使用すると、ツールキットは起動設定の検証と動作を変更して、api フィールドをサポートします。

```
{  
  "type": "aws-sam",  
  "request": "direct-invoke",  
  "name": "myConfig",  
  "invokeTarget": {  
    "target": "api",  
    "templatePath": "n12/template.yaml",  
    "logicalId": "HelloWorldFunction"  
  },  
  "api": {  
    "path": "/hello",  
    "httpMethod": "post",  
    "payload": {  
      "json": {}  
    },  
    "queryString": "abc=def&qrs=tuv",  
    "headers": {  
      "cookie": "name=value; name2=value2; name3=value3"  
    }  
  },  
  "sam": {},  
  "aws": {}  
}
```

例の値を、次のように置き換えます。

`invokeTarget.logicalId`

API リソース。

パス

起動設定が要求する API パス (例: `"path": "/hello"`)。

`invokeTarget.templatePath` で指定した `template.yaml` から解決された有効な API パスでなければなりません。

`httpMethod`

「delete」、「get」、「head」、「options」、「patch」、「post」、「put」のいずれかの動詞とすることができます。

`payload`

リクエストで送信する、`lambda.payload` フィールドと同じ構造とルールを持つ JSON ペイロード (HTTP 本文)。

`payload.path` は JSON ペイロードを含むファイルを指します。

`payload.json` は JSON ペイロードをインラインで指定します。

ヘッダー

名前と値のペアのオプションのマップ。リクエストに含める HTTP ヘッダーを指定するのに使用します。

```
"headers": {
  "accept-encoding": "deflate, gzip;q=1.0, *;q=0.5",
  "accept-language": "fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5",
  "cookie": "name=value; name2=value2; name3=value3",
  "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36",
}
```


querystring

(オプション) この文字列を使用してリクエストの querystring を設定します (例: "querystring": "abc=def&ghi=jkl")。

aws

AWS 接続情報の提供方法。詳細については、[サーバーレスアプリケーションのデバッグ用設定オプション](#) の AWS 接続 (aws) プロパティ表を参照してください。

SAM

CLI AWS SAM がアプリケーションを構築する方法。詳細については、[サーバーレスアプリケーションのデバッグ用設定オプション](#) の AWS SAM CLI (「sam」) のプロパティを参照してください。

サーバーレスアプリケーションの同期

この例では、AWS Toolkit for Visual Studio Codeを使用して、前のトピック(「[サーバーレスアプリケーションの作成](#)」)で作成したサーバーレスアプリケーションを AWS に同期する方法を示します。

前提条件

- グローバルに一意な Amazon S3 バケット名を必ず選択します。
- で設定した認証情報に、Amazon S3、CloudFormation AWS Lambda、および Amazon API Gateway のサービスへの適切な読み取り/書き込みアクセスが含まれていることを確認します。
- デプロイタイプが Image であるアプリケーションの場合、グローバルに一意の Amazon S3 バケット名と、デプロイに使用する Amazon ECR リポジトリ URI の両方があることを確認します。

サーバーレスアプリケーションの同期

1. [AWS Explorer] ウィンドウで、[Lambda] ノードを右クリックしてコンテキストメニューを開き、[SAM アプリケーションを同期] を選択します。
2. AWS リージョン デプロイする を選択します。
3. デプロイに使用する template.yaml ファイルを選択します。
4. このデプロイで使用できる Amazon S3 バケットの名前を入力します。バケットは、デプロイ先のリージョンにある必要があります。

⚠ Warning

Amazon S3 バケット名は、Amazon S3 内のどの既存バケット名とも重複しないグローバルに一意な名前である必要があります。次の例に示す名前に一意の識別子を追加するか、別の名前を選択します。

5. サーバーレスアプリケーションに、パッケージタイプの Image を伴う関数が含まれているのであれば、このデプロイで利用できる Amazon ECR リポジトリの名前を入力します。リポジトリは、デプロイ先のリージョンにある必要があります。
6. デプロイしたスタックの名前 (新しいスタック名または既存のスタック名) を入力します。
7. コンソールの [AWS ツールキット] タブ上のデプロイの成功を確認します。

エラーが発生すると、右下にメッセージが表示されます。

この場合は、[AWS ツールキット] タブのテキストで詳細をチェックします。以下に示しているのは、エラーの詳細の例です。

```
Error with child process: Unable to upload artifact HelloWorldFunction referenced
  by CodeUri parameter of HelloWorldFunction resource.
S3 Bucket does not exist. Execute the command to create a new bucket
aws s3 mb s3://pbart-my-sam-app-bucket
An error occurred while deploying a SAM Application. Check the logs for more
information by running the "View AWS Toolkit Logs" command from the Command
Palette.
```

この例では、Amazon S3 バケットが存在しないため、エラーが発生しています。

デプロイが完了すると、アプリケーションが [AWS Explorer] に表示されます。アプリケーションの一部として作成した Lambda 関数の呼び出し方法については、「[リモートの Lambda 関数を呼び出す](#)」を参照してください。

AWS クラウドからのサーバーレスアプリケーションの削除

サーバーレスアプリケーションを削除するには、以前に AWS クラウドにデプロイした CloudFormation スタックを削除します。この手順を実行しても、ローカルホストからアプリケーションディレクトリは削除されないことにご注意ください。

1. [AWS Explorer] を開きます。

2. [AWS Explorer] ウィンドウで、削除したいデプロイされたアプリケーションを含むリージョンを展開し、CloudFormationを拡張します。
3. 削除するサーバーレスアプリケーションに対応する CloudFormation スタックの名前のコンテキスト (右クリック) メニューを開きます。次に、[Delete CloudFormation Stack] (CloudFormation スタックを削除) を選択します。
4. 選択したスタックを削除したいことを確認する場合は、[削除] を選択します。

スタックの削除が成功すると、AWS Toolkit は AWS Explorer の CloudFormation リストからスタック名を削除します。

AWS Toolkit コードレンズの有効化

このステップでは、AWS Toolkit コードレンズを有効にする方法を示します。

1. メニューバーで、[AWS Cloud9]、[設定] の順に選択します。
2. [設定] タブで、サイドバーの [AWS ツールキット] を選択します。
3. コードレンズを有効にするには、[Enable Code Lenses] (コードレンズを有効にする) を選択します。

サーバーレスアプリケーションのデバッグ用設定オプション

インラインアクションを使用すると、Lambda 関数を直接呼び出すか、SAM テンプレートを使用してプロパティを簡単に検索して定義できます。また、"lambda" (関数の実行方法)"sam"、(CLI がアプリケーションを構築する方法)、"aws" (AWS 接続情報の提供方法) AWS SAM のプロパティを定義することもできます。

AWS SAM: Lambda ハンドラーの直接呼び出し/テンプレートベースの Lambda 呼び出し

プロパティ	説明
type	起動設定を管理する拡張機能を指定します。CLI AWS SAM を使用してローカルでビルドおよびデバッグaws-samするには、常に に設定します。
name	起動設定のデバッグリストに表示される読みやすい名前を指定します。

プロパティ	説明
<code>request</code>	指定された拡張子 (aws-sam) が実行する構成の種類を指定します。常に <code>direct-invoke</code> に設定され、Lambda 関数をスタートします。
<code>invokeTarget</code>	リソースを呼び出すためのエントリポイントを指定します。 Lambda 関数を直接呼び出すには、次の <code>invokeTarget</code> フィールドに値を設定: • <code>target-code</code> に設定します。 • <code>lambdaHandler</code> – 呼び出す予定の Lambda 関数ハンドラの名前。 • <code>projectRoot</code> – Lambda ハンドラーを含むアプリケーションファイルのパス。 SAM テンプレートを使用して Lambda リソースを呼び出すには、次の <code>invokeTarget</code> フィールドに値を設定: • <code>target-template</code> に設定します。 • <code>templatePath</code> – SAM テンプレートファイルへのパス。 • <code>logicalId</code> – 呼び出す リソース名 <code>AWS::Lambda::Function</code> または <code>AWS::Serverless::Function</code>。リソース名は、YAML 形式の SAM テンプレートで検索できます。

Lambda ("**lambda**") のプロパティ

プロパティ	説明
<code>environmentVariables</code>	オペレーショナルパラメータを関数に渡します。例えば、Amazon S3 バケットに書き込む場合は、バケット名を環境変数として設定します。書き込み先のバケット名をハードコーディングしないでください。

プロパティ	説明
payload	入力として Lambda 関数に提供されるイベントペイロード用に 2 つのオプションを提供します。 "json": イベントペイロードを定義する JSON 形式のキーバリューのペア。 "path": イベントペイロードとして使用されるファイルへのパス。
memoryMB	、呼び出された Lambda 関数のために提供されたメモリのメガバイトを指定します
runtime	Lambda 関数で使用するランタイムを指定します。詳細については、「 AWS Lambda ランタイム 」を参照してください。
timeoutSec	デバッグセッションがタイムアウトするまでの許可される時間を秒単位で設定します。

AWS Toolkit 拡張機能は CLI AWS SAM を使用して、サーバーレスアプリケーションをローカルで構築およびデバッグします。AWS SAM CLI コマンドの動作は、launch.json ファイル"sam"の設定のプロパティを使用して設定できます。

AWS SAM CLI ("sam") プロパティ

プロパティ	説明	デフォルトの値
buildArguments	sam build コマンドが Lambda ソースコードを構築する方法を設定します。構築オプションを表示するには、「AWS Serverless Application Model デベロッパーガイド」の「sam build」を参照してください。	空の文字列

プロパティ	説明	デフォルトの値
containerBuild	AWS Lambdaのような Docker コンテナ内に関数を構築するかどうかを示します。	false
dockerNetwork	Lambda Docker コンテナが接続する既存の Docker ネットワークの名前または ID を、デフォルトのブリッジネットワークとともに指定します。指定しない場合、Lambda コンテナはデフォルトのブリッジ Docker ネットワークにのみ接続します。	空の文字列
localArguments	追加のローカル呼び出し引数。	空の文字列
skipNewImageCheck	コマンドが Lambda ランタイム用の最新 Docker イメージのプルダウンをスキップするかどうかを指定します。	false
template	パラメータを使用してそれに対して顧客の値を入力することで、SAM テンプレートをカスタマイズします。詳細については、「AWS CloudFormation ユーザーガイド」の「 パラメータ 」を参照してください。	"parameters": {}

AWS 接続 ("aws") プロパティ

プロパティ	説明	デフォルトの値
credentials	認証情報ファイルから特定のプロファイル (など <code>profile:default</code>) を選択して、AWS 認証情報を取得します。	既存の共有設定 AWS ファイルまたは共有 AWS AWS 認証情報ファイルによって提供される認証情報。
Region	サービスの AWS リージョン (us-east-1 など) を設定します。	アクティブな認証情報プロファイルに関連付けられているデフォルトの AWS リージョン。

Toolkit AWS Step Functions を使用した の操作 AWS

Toolkit AWS では、 がサポートされています [AWS Step Functions](#)。Step Functions を使用すると、ビジネスクリティカルなアプリケーションをサポートする AWS Lambda 関数やその他の AWS サービスのワークフローを定義するステートマシンを作成できます。

AWS ツールキットを使用して、Step Functions で以下を実行できます。

- ステートマシンを作成して発行します。ステートマシンは、個々のステップで構成されるワークフローです。
- ステートマシンのワークフローを定義するファイルをダウンロードします。
- 直接入力するか、または選択した入力を使用して、ステートマシンのワークフローを実行します。

トピック

- [前提条件](#)
- [ステートマシンの作成と発行](#)
- [AWS Toolkit でステートマシンを実行する](#)
- [ステートマシン定義ファイルをダウンロードし、そのワークフローを視覚化する](#)

前提条件

Step Functions はコードを実行し、AWS リソースにアクセスできます (Lambda 関数の呼び出しなど)。セキュリティを維持するために、IAM ロールを使用してこれらのリソースへの Step Functions アクセスを許可する必要があります。

AWS Toolkit を使用すると、ステートマシンを作成する AWS リージョンで有効な自動生成された IAM ロールを利用できます。ステートマシン用に独自の IAM ロールを作成するには、「AWS Step Functions デベロッパーガイド」の「[IAM と AWS Step Functions 連携する方法](#)」を参照してください。

ステートマシンの作成と発行

AWS Toolkit を使用してステートマシンを作成するときは、ビジネスケースのワークフローを定義するスターターテンプレートを選択します。それにより、テンプレートを特定のニーズに応じて編集または置き換えることができます。ステートマシンの構造を表すファイルにステートマシンを定義する方法の詳細については、「AWS Step Functions デベロッパーガイド」の「[Amazon ステートメント言語](#)」を参照してください。

1. [AWS Explorer] ペインで、[Step Functions] のコンテキスト (右クリック) メニューを開き、[Create a new Step Function state machine] (新しい Step Function ステートマシンを作成) を選択します。
2. コマンドパネルで、ステートマシンのワークフローのスターターテンプレートを選択します。
3. 次に、ステートマシンを定義する Amazon ステートメント言語 (ASL、Amazon States Language) ファイルの形式を選択します。

エディタが開き、ステートマシンのワークフローを定義する ASL ファイルが表示されます。

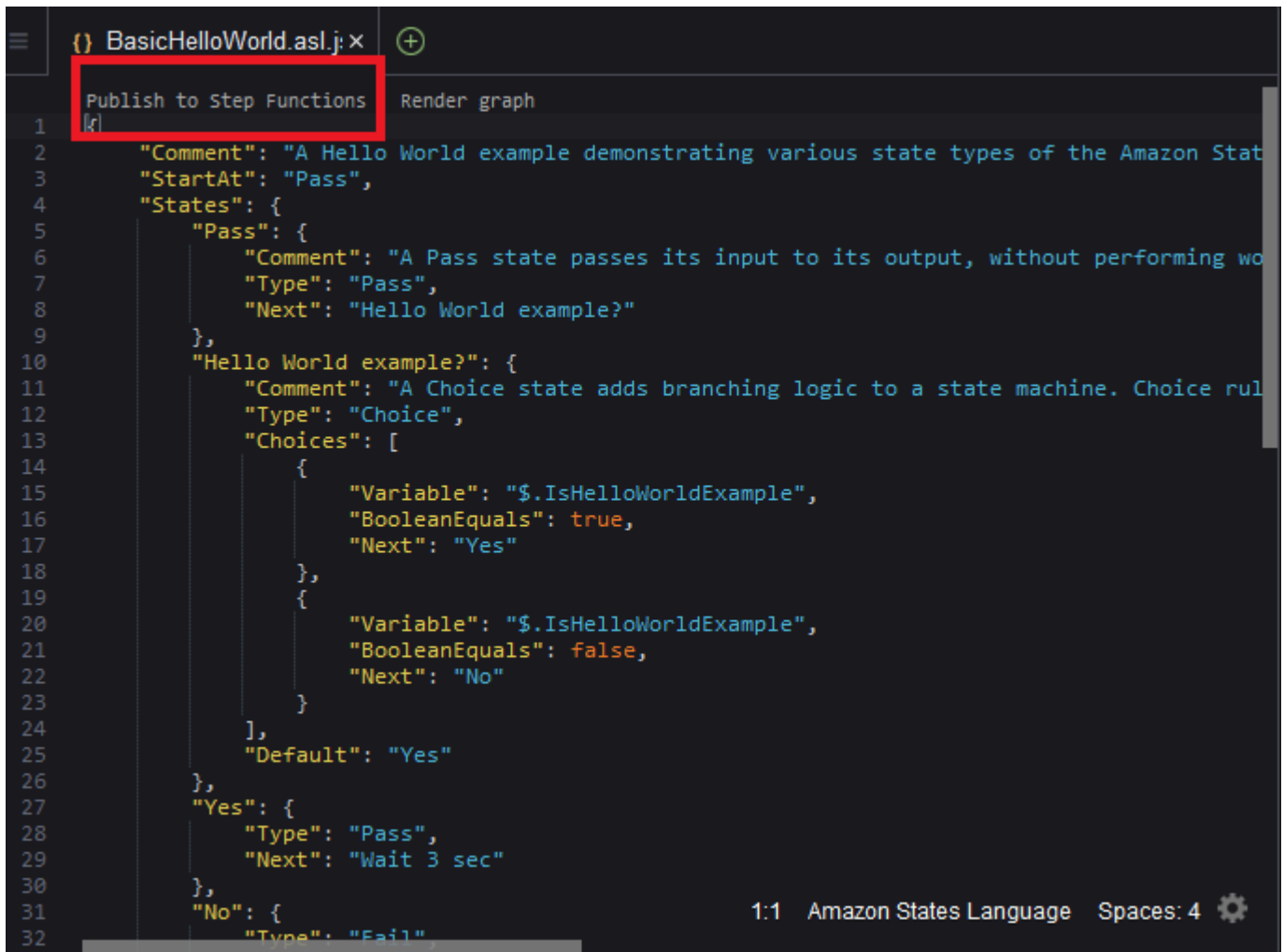
Note

ASL ファイルを編集してワークフローをカスタマイズする方法については、「[ステートマシンの構造](#)」を参照してください。

4. ASL ファイルで、Step Functions に発行を選択して、ステートマシンを AWS クラウドに追加します。

Note

ASL ファイルで [Render graph] (グラフをレンダリング) を選択して、ステートマシンのワークフローを視覚化することもできます。



The screenshot shows the AWS Cloud9 editor interface. At the top, there's a tab labeled 'BasicHelloWorld.asl.j: x'. Below the tab, there are two buttons: 'Publish to Step Functions' (highlighted with a red box) and 'Render graph'. The main area contains ASL (Amazon States Language) code for a 'Hello World' example. The code defines a state machine with a 'Pass' state and a 'Hello World example?' state, which branches into 'Yes' and 'No' paths. The 'Yes' path leads to a 'Wait 3 sec' state, and the 'No' path leads to a 'Fail' state. The bottom status bar indicates '1:1 Amazon States Language Spaces: 4'.

```
1 {} BasicHelloWorld.asl.j: x
2
3 "Comment": "A Hello World example demonstrating various state types of the Amazon States Language",
4 "StartAt": "Pass",
5 "States": {
6   "Pass": {
7     "Comment": "A Pass state passes its input to its output, without performing any work",
8     "Type": "Pass",
9     "Next": "Hello World example?"
10  },
11  "Hello World example?": {
12    "Comment": "A Choice state adds branching logic to a state machine. Choice rules evaluate a series of conditions until one is met, then execute the corresponding path.",
13    "Type": "Choice",
14    "Choices": [
15      {
16        "Variable": "$.IsHelloWorldExample",
17        "BooleanEquals": true,
18        "Next": "Yes"
19      },
20      {
21        "Variable": "$.IsHelloWorldExample",
22        "BooleanEquals": false,
23        "Next": "No"
24      }
25    ],
26    "Default": "Yes"
27  },
28  "Yes": {
29    "Type": "Pass",
30    "Next": "Wait 3 sec"
31  },
32  "No": {
33    "Type": "Fail",
34    "Error": "Failed",
35    "Next": null
36  }
37 }
```

5. コマンドパネルで、ステップ関数をホストする AWS リージョンを選択します。
6. 次に、新しいステップ関数を作成するか、既存のステップ関数を更新するかを選択できます。

Quick Create

このオプションにより、[step-functions/latest/dg/concepts-standard-vs-express.html](https://docs.aws.amazon.com/step-functions/latest/dg/concepts-standard-vs-express.html) を使用して、ASL ファイルから新しいステップ関数を作成できます。次の項目を指定する必要があります。

- ステップ関数がコードを実行して AWS リソースにアクセスできるようにする IAM ロール。(ステートマシンを作成する AWS リージョンに対して有効な自動生成された IAM ロールを選択できます)。
- 新しい関数の名前。

ステートマシンが正常に作成されたことを確認し、AWS ツールキット出力タブで ARN を取得できます。

Quick Update

AWS リージョンにステートマシンがすでに存在する場合は、現在の ASL ファイルで更新するステートマシンを選択できます。

ステートマシンが正常に更新されたことを確認し、AWS ツールキット出力タブで ARN を取得できます。

ステートマシンを作成すると、[AWS Explorer] ペインの [Step Functions] に表示されます。すぐに表示されない場合は、[Toolkit] (ツールキット) メニューの [Refresh Explorer] (Explorer を更新) を選択します。

AWS Toolkit でステートマシンを実行する

AWS Toolkit を使用してリモートステートマシンを実行できます。実行状態のマシンは JSON テキストを入力として受け取り、その入力をワークフローの最初の状態に渡します。それぞれの状態で JSON を入力として受け取ります。また、通常 JSON を出力として次の状態に渡します。詳細については、「[Step Functions の入出力処理](#)」を参照してください。

1. [AWS Explorer] ペインで、[Step Functions] を選択します。次に、特定のステートマシンのコンテキスト (右クリック) メニューを開き、[Start Execution] (実行の開始) を選択します。
2. [Start Execution] (実行の開始) ペインで、下のフィールドに直接テキストを入力するか、ローカルデバイスからファイルをアップロードすることにより、ステートマシンのワークフローに JSON 形式入力を追加します。
3. [Execute] (実行) を選択します。

Toolkit AWS 出力タブには、ワークフローが開始されたことの確認と、プロセス ID の ARN が表示されます。そのプロセス ID を使用して、ワークフローが正常に実行されたかどうかを AWS

Step Functions コンソールで確認できます。また、ワークフローの開始時刻と終了時刻のタイムスタンプを確認することもできます。

ステートマシン定義ファイルをダウンロードし、そのワークフローを視覚化する

ステートマシンをダウンロードするには、そのステートマシンの構造を表す JSON テキストを含むファイルをダウンロードします。それにより、ファイルを編集して新しいステートマシンを作成したり、既存のステートマシンを更新したりできます。詳細については、「AWS Step Functions デベロッパーガイド」の「[Amazon ステートメント言語](#)」を参照してください。

1. [AWS Explorer] ペインで、[Step Functions] を選択します。次に、特定のステートマシンのコンテキスト (右クリック) メニューを開き、[Download Definition] (定義をダウンロード) を選択します。

Note

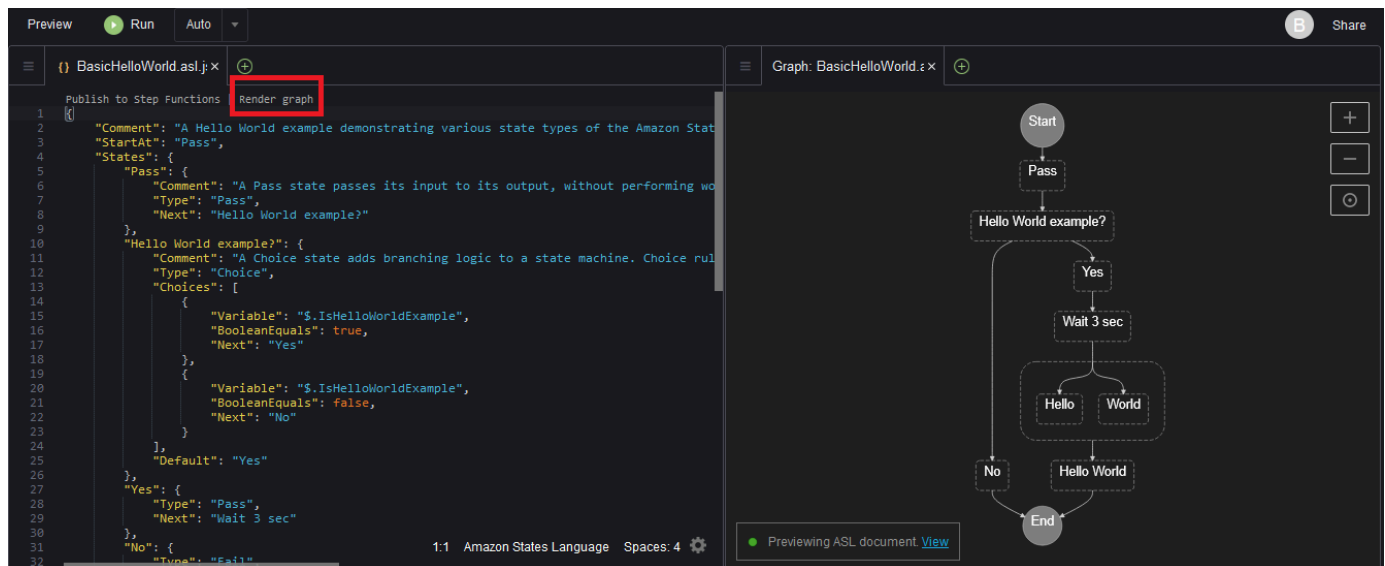
コンテキストメニューでは、[Copy Name] (名前をコピー) と [Copy ARN] (ARN をコピー) も選択できます。

2. [Save] (保存) ダイアログボックスで、ダウンロードしたステートマシンファイルを保存する環境内のフォルダを選択し、[Save] (保存) を選択します。

ステートマシンのワークフローを定義する JSON 形式ファイルがエディタに表示されます。

3. ワークフローを視覚化するには、[Render graph] (グラフをレンダリング) を選択します。

ウィンドウにフローチャートが表示され、ステートマシンのワークフロー内の状態のシーケンスが表示されます。



Systems Manager オートメシヨンドキュメントの使用

を使用すると AWS Systems Manager、インフラストラクチャを可視化して制御できます AWS。Systems Manager は、複数の からの運用データを表示 AWS のサービスし、AWS リソース 全体の運用タスクを自動化するために使用できる統合ユーザーインターフェイスを提供します。

[システムマネージャのドキュメント](#) は、マネージドインスタンスで Systems Manager が実行する アクションを定義します。オートメシヨンドキュメントは、Systems Manager ドキュメントの 1 つのタイプであり、一般的なメンテナンスやデプロイのタスクを実行するのに使用します。これには、Amazon マシンイメージ (AMI) の作成や更新が含まれます。このトピックでは、AWS Toolkit を使用してオートメシヨンドキュメントを作成、編集、公開、削除する方法について説明します。

トピック

- [仮定と前提条件](#)
- [Systems Manager Automation ドキュメントの IAM アクセス許可](#)
- [Systems Manager オートメシヨンドキュメントを新規作成する](#)
- [Systems Manager オートメシヨンドキュメントの発行](#)
- [既存の Systems Manager オートメシヨンドキュメントの編集](#)
- [バージョンの使用](#)
- [Systems Manager オートメシヨンドキュメントの削除](#)
- [Systems Manager オートメシヨンドキュメントの実行](#)

- [AWS Toolkit での Systems Manager オートメーションドキュメントのトラブルシューティング](#)

仮定と前提条件

始める前に、以下の条件を満たしていることを確認してください。

- Systems Manager に精通しています 詳細については、「[AWS Systems Manager ユーザーガイド](#)」を参照してください。
- Systems Manager オートメーションのユースケースに精通している。詳細については、「AWS Systems Manager ユーザーガイド」の「[AWS Systems Manager オートメーション](#)」を参照してください。

Systems Manager Automation ドキュメントの IAM アクセス許可

Systems Manager オートメーションドキュメントを作成、編集、公開、削除するには、必要な AWS Identity and Access Management (IAM) アクセス許可を含む認証情報プロファイルが必要です。次のポリシードキュメントは、プリンシパルポリシーで使用する必須の IAM アクセス許可を定義します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:ListDocuments",
        "ssm:ListDocumentVersions",
        "ssm:DescribeDocument",
        "ssm:GetDocument",
        "ssm:CreateDocument",
        "ssm:UpdateDocument",
        "ssm:UpdateDocumentDefaultVersion",
        "ssm>DeleteDocument"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

IAM ポリシーの更新方法については、IAM ユーザーガイドの「[IAM ポリシーの作成](#)」を参照してください。

Systems Manager オートメーションドキュメントを新規作成する

オートメーションドキュメントは、AWS ツールキットを使用して、JSON または YAML 形式で作成できます。オートメーションドキュメントを作成すると、そのドキュメントはタイトルのないファイルに表示されます。ファイルに名前を付けて保存できます。ただし、ファイルは公開 AWS するまでにアップロードされません。

オートメーションドキュメントを新規作成するには

1. 左側のナビゲーションペインで検索アイコンを選択するか、[Ctrl+P] を押して、検索ペインを開きます。
2. 検索ペインで、「systems manager」という語句の入力を開始し、[AWS: Create a new Systems Manager Document Locally] コマンドが表示されたら選択します。
3. 「Hello World」の例のためにスターターテンプレートの 1 つを選択します。
4. ドキュメントの形式として、JSON または YAML を選択します。

エディタに新しいオートメーションドキュメントが表示されます。

Note

最初にローカルでオートメーションドキュメントを作成しても、そのドキュメントは AWS に自動的に表示されません。実行する前に、公開する必要があります AWS。

Systems Manager オートメーションドキュメントの発行

AWS Toolkit でオートメーションドキュメントを作成または編集したら、それを公開できます AWS。

オートメーションドキュメントを発行するには

1. 「[既存の Systems Manager オートメーションドキュメントの編集](#)」の手順に従って、発行するオートメーションドキュメントを開きます。
2. 左側のナビゲーションペインで検索アイコンを選択するか、[Ctrl+P] を押して、検索ペインを開きます。
3. 検索ペインで、「systems manager」という語句の入力を開始し、[AWS: Publish a new Systems Manager Document] コマンドが表示されたら選択します。
4. ステップ 1/3 AWS リージョン で、ドキュメントを発行する を選択します。
5. [Step 2 of 3] (ステップ 2/3) で、[Quick Create] (クイック作成) を選択して、オートメーションドキュメントを作成します。または、[Quick Update] (クイック更新) を選択して、そのリージョンの既存のオートメーションドキュメントを更新します。

Note

更新できるのは、自身が所有するオートメーションドキュメントのみです。[Quick Update] (クイック更新) を選択し、リージョン内にドキュメントを所有していない場合は、更新する前にドキュメントを発行する必要があることを通知するメッセージが表示されます。

6. [Step 3 of 3] (ステップ 3/3) で、前のステップの選択に応じて、新しいオートメーションドキュメントの名前を入力するか、更新する既存のドキュメントを選択します。

Note

で既存のオートメーションドキュメントに更新を発行すると AWS、新しいバージョンがドキュメントに追加されます。ドキュメントに複数のバージョンがある場合は、[デフォルトのバージョン](#)を設定できます。

既存の Systems Manager オートメーションドキュメントの編集

AWS Explorer を使用して、既存の Systems Manager オートメーションドキュメントを検索します。既存のドキュメントを開くと、AWS Cloud9 エディタにタイトルのないファイルとして表示されます。次の 3 種類のオートメーションドキュメントをダウンロードできます。

- Owned by Amazon (Amazon が所有): 実行時にパラメータを指定して使用できる事前設定された SSM ドキュメント。
- 所有: 作成して公開したドキュメント AWS。
- 共有先: AWS アカウント ID に基づいて所有者が共有したドキュメント。

更新できるドキュメントのタイプ AWS は、自分が所有するドキュメントのみです。共有タイプまたは Amazon 所有タイプのオートメーションドキュメントをダウンロードして、AWS Cloud9で編集することもできます。ただし、に発行するときは AWS、新しいドキュメントを作成するか、所有している既存のドキュメントを更新する必要があります。他のユーザーまたは Amazon が所有しているドキュメントの新規バージョンは作成できません。

詳細については、「AWS Systems Manager ユーザーガイド」の「[AWS Systems Manager ドキュメント](#)」を参照してください。

1. AWS Explorer の Systems Manager で、ダウンロードする SSM ドキュメントのカテゴリを選択します。Amazon 所有、自分所有、または自分と共有。
2. 特定のドキュメントで、コンテキスト (右クリック) メニューを開き、[Download as YAML] (YAML としてダウンロード) または [Download as JSON] (JSON としてダウンロード) を選択します。

フォーマット済みの SSM ドキュメントが新しい編集タブに表示されます。

編集が完了したら、AWS: 新しい Systems Manager Document コマンドを発行して、AWS クラウドに新しいドキュメントを作成するか、所有している既存のドキュメントを更新できます。

バージョンの使用

Systems Manager オートメーションドキュメントでは、変更の管理にバージョンが使用されます。AWS Toolkit では、ドキュメントのデフォルトバージョンを設定できます。これは、ドキュメントの実行時に使用されるバージョンです。

デフォルトバージョンを設定するには

- AWS Explorer で、デフォルトバージョンを設定するドキュメントに移動し、ドキュメントのコンテキスト (右クリック) メニューを開き、デフォルトバージョンの設定を選択します。

 Note

選択したドキュメントに 1 つのバージョンしかない場合、デフォルトを変更することはできません。

Systems Manager オートメーションドキュメントの削除

AWS Toolkit で所有しているオートメーションドキュメントを削除できます。オートメーションドキュメントを削除すると、ドキュメントとドキュメントのすべてのバージョンが削除されます。

 Important

- 削除は、元に戻すことができない破壊的なアクションです。
- 既に開始されているオートメーションドキュメントを削除しても、その実行時に作成または変更された AWS リソースは削除されません。
- 削除は、ドキュメントを所有している場合にのみ許可されます。

オートメーションドキュメントを削除するには

1. AWS Explorer ペインの Systems Manager で、Owned by Me を展開してドキュメントを一覧表示します。
2. 削除するドキュメントのコンテキスト (右クリック) メニューを開き、[Delete document] (ドキュメントの削除) を選択します。
3. 表示される警告ダイアログボックスで、[Delete] (削除) を選択して確認します。

Systems Manager オートメーションドキュメントの実行

オートメーションドキュメントが に公開されたら AWS、オートメーションドキュメントを実行して、 でユーザーに代わってタスクを実行できます AWS アカウント。Automation ドキュメントを実行するには、AWS マネジメントコンソール、APIs、 、AWS CLIまたは を使用します AWS Tools for PowerShell。オートメーションドキュメントの実行方法については、「AWS Systems Manager ユーザーガイド」の「[シンプルなオートメーションワークフローを実行する](#)」を参照してください。

または、AWS SDKs を使用してオートメーションドキュメントを実行する場合は、[AWS SDK リファレンス](#)を参照してください。APIs

Important

オートメーションドキュメントを実行すると、に新しいリソースが作成され AWS、請求コストが発生する可能性があります。オートメーションドキュメントを実行する前に、アカウントにどのようなリソースが作成されるかを把握することを強くお勧めします。

AWS Toolkit での Systems Manager オートメーションドキュメントのトラブルシューティング

オートメーションドキュメントを AWS Toolkit に保存しましたが、に表示されません AWS マネジメントコンソール。

AWS Toolkit にオートメーションドキュメントを保存しても、オートメーションドキュメントはに発行されません AWS。オートメーションドキュメントの公開の詳細については、「[Systems Manager オートメーションドキュメントの発行](#)」を参照してください。

オートメーションドキュメントの発行がアクセス許可エラーで失敗しました。

AWS 認証情報プロファイルに、オートメーションドキュメントを発行するために必要なアクセス許可があることを確認します。アクセス許可ポリシーの例については、「[Systems Manager Automation ドキュメントの IAM アクセス許可](#)」を参照してください。

オートメーションドキュメントを に公開しましたが AWS、AWS Explorer ペインに表示されません。

AWS Explorer ペインで参照しているのと同じ AWS リージョンにドキュメントを公開していることを確認します。

オートメーションドキュメントを削除しましたが、作成したリソースの料金が現在も請求されています。

オートメーションドキュメントを削除しても、ドキュメントによって作成または変更されたリソースは削除されません。Billing [AWS Management Console](#) から作成した AWS リソースを特定し、料金を調べて、そこから削除するリソースを選択できます。

IDE での Amazon ECR AWS Cloud9 の使用

Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブルな AWS マネージドコンテナレジストリサービスです。AWS Toolkit Explorer から複数の Amazon ECR サービス関数にアクセスできます。

- リポジトリの作成
- リポジトリまたはタグ付けされたイメージの AWS App Runner サービスの作成。
- イメージタグおよびリポジトリの URI または ARN にアクセスする。
- イメージタグとリポジトリを削除する。

CLI やその他のプラットフォームをインストールすることで、AWS Cloud9 コンソールから Amazon ECR AWS 関数の全範囲にアクセスすることもできます。

Amazon ECR の詳細については、Amazon Elastic Container Registry ユーザーガイドの「[Amazon ECR とは?](#)」を参照してください。

前提条件

以下は、AWS Cloud9 Amazon EC2 環境の AWS Cloud9 IDE にプリインストールされています。IDE から Amazon ECR AWS Cloud9 サービスにアクセスするために必要です。

IAM 認証情報

AWS コンソールで作成して認証に使用した IAM ロール。IAM の詳細については、「[AWS Identity and Access Management ユーザーガイド](#)」を参照してください。

[Docker の設定]

Docker は Amazon AWS Cloud9 Amazon EC2 AWS Cloud9 環境の IDE にプリインストールされています。Docker の詳細については、「[Install Docker Engine](#)」を参照してください。

AWS CLI バージョン 2 の設定

AWS CLI バージョン 2 は、AWS Cloud9 Amazon EC2 AWS Cloud9 環境の IDE にプリインストールされています。AWS CLI バージョン 2 の詳細については、「[CLI バージョン AWS 2 のインストール、更新、アンインストール](#)」を参照してください。

トピック

- [での Amazon ECR サービスの使用 AWS Cloud9](#)

での Amazon ECR サービスの使用 AWS Cloud9

Amazon Elastic Container Registry (Amazon ECR) サービスには、IDE の AWS Explorer AWS Cloud9 から直接アクセスできます。Amazon ECR を使用して、プログラムイメージを Amazon ECR リポジトリにプッシュできます。使用を開始するには、次のステップに従います。

1. イメージの構築に必要な情報を含む Dockerfile を作成します。
2. その Dockerfile からイメージをビルドし、処理のためにイメージにタグを付けます。
3. Amazon ECR インスタンス内にリポジトリを作成します。
4. リポジトリにタグ付けされたイメージをプッシュします。

セクション

- [前提条件](#)
- [1. Dockerfile の作成](#)
- [2. Dockerfile からイメージをビルドする](#)
- [3. 新しいリポジトリの作成](#)
- [4. イメージのプッシュ、プル、削除](#)

前提条件

AWS Toolkit for の Amazon ECR 機能を使用する前に AWS Cloud9、まずこれらの[前提条件](#)を満たしていることを確認してください。これらの前提条件は、AWS Cloud9 Amazon EC2 AWS Cloud9 環境の IDE にプリインストールされており、Amazon ECR にアクセスするために必要です。

1. Dockerfile の作成

Docker は Dockerfile というファイルを使用して、リモートリポジトリにプッシュおよび保存できるイメージを定義します。ECR リポジトリにイメージをアップロードする前に、Dockerfile を作成し、その Dockerfile からイメージをビルドします。

Dockerfile の作成

1. Dockerfile を保存するディレクトリに移動するには、AWS Cloud9 IDE 内の左側のナビゲーションバーにある [Toggle Tree] (ツリーの切り替え) オプションを選択します。

2. Dockerfile という名前の新しいファイルを作成します。

Note

AWS Cloud9 IDE では、ファイルタイプまたはファイル拡張子を選択するように求められる場合があります。これが発生した場合は、plaintext. AWS Cloud9 IDE に「dockerfile」拡張子が付いていることを選択します。ただし、使用することは推奨されていません。これは、拡張機能が特定のバージョンの Docker または他の関連アプリケーションと競合する可能性があるためです。

IDE を使用した Dockerfile AWS Cloud9 の編集

Dockerfile にファイル拡張子がある場合は、そのファイルのコンテキスト (右クリック) メニューを開き、ファイル拡張子を削除します。拡張子を持つ Dockerfile は、Docker の特定のバージョンやその他の関連アプリケーションと競合する可能性があります。

Dockerfile からファイル拡張子を削除したら、次の操作を行います。

1. IDE で空の Dockerfile AWS Cloud9 を直接開きます。
2. 次の例の内容を Dockerfile にコピーします。

Example Dockerfile イメージテンプレート

```
FROM ubuntu:22.04

# Install dependencies
RUN apt-get update && \
    apt-get -y install apache2

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \
    echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
    echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh
```

```
EXPOSE 80
```

```
CMD /root/run_apache.sh
```

これは Ubuntu 22.04 イメージを使用する Dockerfile です。実行命令は、パッケージキャッシュを更新します。ウェブサーバー用のいくつかのソフトウェアがインストールされてから、「Hello World!」ウェブサーバーのドキュメントルートに書き込まれます。EXPOSE の命令はコンテナ上のポート 80 を公開し、CMD の命令はウェブサーバーを起動します。

3. Dockerfile を保存します。

2. Dockerfile からイメージをビルドする

作成した Dockerfile には、プログラムのイメージを構築するために必要な情報が含まれています。そのイメージを Amazon ECR インスタンスにプッシュする前に、まずイメージをビルドします。

Dockerfile からイメージをビルドする

1. Dockerfile を含むディレクトリに移動するには、Docker のインスタンスと統合された Docker CLI または CLI を使用します。
2. Dockerfile で定義されているイメージをビルドするには、Dockerfile と同じディレクトリから Docker ビルドコマンドを実行します。

```
docker build -t hello-world .
```

3. イメージが正しく作成されたことを検証するには、Docker イメージ コマンドを実行します。

```
docker images --filter reference=hello-world
```

Example

出力は次のとおりです。

REPOSITORY SIZE	TAG	IMAGE ID	CREATED

hello-world 241MB	latest	e9ffedc8c286	4 minutes ago

4. Ubuntu 22.04 に基づいて新しくビルドされたイメージを実行するには、echo コマンドを使用します。

 Note

このステップは、イメージの作成やプッシュには必要ありません。ただし、プログラムイメージの実行時の動作を確認できます。

```
FROM ubuntu:22.04
CMD ["echo", "Hello from Docker in Cloud9"]
```

次に Dockerfile を実行して構築します。このコマンドは、Dockerfile と同じディレクトリから実行する必要があります。

```
docker build -t hello-world .
docker run --rm hello-world
```

Example

出力は次のとおりです。

```
Hello from Docker in Cloud9
```

Docker run コマンドの詳細については、Docker ウェブサイトの「[Docker run reference](#)」を参照してください。

3. 新しいリポジトリの作成

Amazon ECR インスタンスにイメージをアップロードするには、保存できる新しいリポジトリを作成します。

新しい Amazon ECR リポジトリの作成

1. IDE AWS Cloud9 ナビゲーションバーから、AWS ツールキットアイコンを選択します。
2. AWS Explorer メニューを拡張します。
3. AWS リージョンに関連付けられているデフォルトを見つけます AWS アカウント。次に、これを選択すると、IDE AWS Cloud9 経由のサービスのリストが表示されます。
4. ECR オプションのコンテキスト (右クリック) メニューを開いて、新しいリポジトリの作成プロセスを開始します。次に、[Create Repository] (リポジトリの作成) を選択します。
5. プロセスを完了するには、プロンプトに従います。
6. プロセスが完了したら、AWS Explorer メニューの ECR セクションから新しいリポジトリにアクセスできます。

4. イメージのプッシュ、プル、削除

Dockerfile からイメージを構築してリポジトリを作成したら、イメージを Amazon ECR リポジトリにプッシュできます。さらに、Docker と AWS CLI で AWS Explorer を使用すると、次の操作を実行できます。

- イメージをリポジトリからプルします。
- リポジトリに保存されているイメージを削除します。
- リポジトリを削除します。

デフォルトレジストリでの Docker の認証

Amazon ECR インスタンスと Docker インスタンス間でデータを交換するには、認証が必要です。レジストリで Docker を認証するには

1. IDE AWS Cloud9 内でターミナルを開きます。
2. get-login-password メソッドを使用してプライベート ECR レジストリを認証し、リージョンと AWS アカウント ID を入力します。

```
aws ecr get-login-password \
  --region <region> \
| docker login \
  --username AWS \
  --password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```


⚠ Important

前述のコマンドで、**region** と **AWS_account_id** を、AWS アカウントに固有の情報に置き換えます。有効な **region** の値は us-east-1 です。

イメージへのタグ付けとレポジトリへのプッシュ

のインスタンスで Docker を認証したら AWS、イメージをレポジトリにプッシュします。

1. `docker images` コマンドを使用して、ローカルに保存したイメージを表示し、タグ付けするイメージを特定します。

```
docker images
```

Example

出力は次のとおりです。

REPOSITORY SIZE	TAG	IMAGE ID	CREATED
hello-world 241MB	latest	e9ffedc8c286	4 minutes ago

2. Docker コマンドを使用してイメージをビルドします。

```
docker tag hello-world:latest AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world:latest
```

3. レポジトリに Docker プッシュコマンドでタグ付けされたイメージをプッシュします。

⚠ Important

ローカルレポジトリの名前が AWS Amazon EC2 レポジトリと同じであることを確認します。この例では、両方のレポジトリの名前が「hello-world」である必要があります。Docker によるイメージのプッシュの詳細については、「[Docker イメージをプッシュする](#)」を参照してください。

```
docker push AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world:latest
```

Example

出力は次のとおりです。

```
The push refers to a repository [AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest:
sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b size: 6774
```

タグ付けされたイメージがリポジトリに正常にアップロードされたら、Explorer タブから Refresh Explorer を選択して AWS Toolkit AWS を更新します。その後、IDE AWS Cloud9 の AWS Explorer メニューに表示されます。

Amazon ECR からのイメージのプル

- イメージは、Docker タグコマンドのローカルインスタンスにプルできます。

```
docker pull AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world:latest
```

Example

出力は次のとおりです。

```
amazonaws.com/hello-world:latest
latest: Pulling from hello-world
Digest: sha256:e02c521fd65eae4ef1acb746883df48de85d55fc85a4172a09a124b11b339f5e
Status: Image is up to date for 922327013870.dkr.ecr.us-west-2.amazonaws.com/hello-world:latest
```

Amazon ECR リポジトリからのイメージの削除

IDE からイメージを削除する方法は 2 AWS Cloud9 つあります。最初の方法は AWS Explorer を使用することです。

1. AWS Explorer から ECR メニューを展開します。
2. イメージを削除するリポジトリを展開します。
3. 削除するイメージに関連付けられているイメージタグのコンテキストメニュー (右クリック) を開きます。
4. そのタグに関連付けられているすべての保存されたイメージを削除するには、[タグの削除...] を選択します。

CLI AWS を使用したイメージの削除

- AWS `ecr batch-delete-image` コマンドを使用して、リポジトリからイメージを削除することもできます。

```
aws ecr batch-delete-image \  
    --repository-name hello-world \  
    --image-ids imageTag=latest
```

Example

出力は次のとおりです。

```
{  
  "failures": [],  
  "imageIds": [  
    {  
      "imageTag": "latest",  
      "imageDigest":  
"sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b"  
    }  
  ]  
}
```

Amazon ECR インスタンスからのリポジトリの削除

IDE からリポジトリを削除する方法は 2 AWS Cloud9 つあります。最初の方法は Explorer AWS を使用することです。

1. AWS Explorer から ECR メニューを展開します。
2. 削除するリポジトリのコンテキスト (右クリック) メニューを開きます。
3. [Delete Repositor...] (リポジトリを削除...) を選択します。

CLI からの Amazon ECR AWS リポジトリの削除

- リポジトリは、AWS `ecr delete-repository` コマンドで削除できます。

Note

通常、まずリポジトリに含まれるイメージを削除しないと、リポジトリを削除することはできません。ただし、`--force` フラグを追加すると、リポジトリとそのすべてのイメージを 1 ステップで削除できます。

```
aws ecr delete-repository \  
--repository-name hello-world \  
--force
```

Example

出力は次のとおりです。

```
--repository-name hello-world --force  
{  
  "repository": {  
    "repositoryUri": "922327013870.dkr.ecr.us-west-2.amazonaws.com/hello-  
world",  
    "registryId": "922327013870",  
    "imageTagMutability": "MUTABLE",  
    "repositoryArn": "arn:aws:ecr:us-west-2:922327013870:repository/hello-  
world",
```

```
"repositoryName": "hello-world",
"createdAt": 1664469874.0
}
}
```

IDE AWS IoT での AWS Cloud9 の使用

AWS Cloud9 IDE AWS IoT で を使用すると、ワークフローの中断を最小限に抑えながら、AWS IoT サービスとやり取りできます AWS Cloud9。このガイドでは、IDE AWS Cloud9 で利用可能な AWS IoT サービス機能の使用を開始する方法について説明します。詳細については、「AWS IoT デベロッパーガイド」の「[AWS IoTとは](#)」を参照してください。

AWS IoT 前提条件

IDE AWS IoT で の使用を開始するには、AWS アカウント と AWS Cloud9 AWS Cloud9 のセットアップがすべての要件を満たしていることを確認してください。AWS IoT サービスに固有の AWS アカウント 要件と AWS ユーザーアクセス許可の詳細については、「AWS IoT デベロッパーガイド」の[AWS IoT 「Core の開始方法」](#)を参照してください。

AWS IoT モノ

AWS IoT はデバイスを AWS のサービス および AWS リソースに接続します。モノと呼ばれるオブジェクトを使用して AWS IoT 、デバイスを に接続できます。モノは、特定のデバイスまたは論理エンティティの表現です。物理的なデバイスやセンサー (電球や壁のスイッチなど) は、モノとして扱うことができます。詳細については、AWS IoT 「AWS IoT デベロッパーガイド」の「[によるデバイスの管理 AWS IoT](#)」を参照してください。

AWS IoT モノの管理

AWS Cloud9 IDE には、モノの管理を効率的にするいくつかの機能があります。AWS IoT モノを管理するには、次の手順に従います。

- [Create a thing](#)
- [Attach a certificate to a thing](#)
- [Detach a certificate from a thing](#)
- [Delete a thing](#)

モノを作成するには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. モノを右クリックしてコンテキストメニューを開き、[Create Thing] (モノの作成) を選択します。
3. モノの名前を [Thing Name] (モノの名前) に入力し、プロンプトに従います。
4. このステップが完了すると、モノのアイコンに続けて、指定した名前が [Thing] (モノ) セクションに表示されます。

モノに証明書をアタッチするには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Things] (モノ) サブセクションで、証明書をアタッチするモノを見つけます。
3. モノを右クリックしてコンテキストメニューを開き、[Attach Certificate] (証明書のアタッチ) を選択して、証明書のリストを含む入力セレクトアを開きます。
4. リストから、モノにアタッチする証明書に対応する証明書 ID を選択します。
5. このステップが完了すると、証明書がアタッチ先のモノの項目として、AWS Explorer でアクセス可能になります。

モノから証明書をデタッチするには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [モノ] サブセクションで、証明書からデタッチしたい [モノ] を検索します。
3. モノを右クリックしてコンテキストメニューを開き、[Attach Certificate] (証明書のアタッチ) を選択します。
4. このステップが完了すると、デタッチされた証明書は AWS Explorer の モノの下に表示されなくなります。ただし、[Certificates] (証明書) サブセクションからは引き続きアクセスできます。

モノを削除するには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Things] (モノ) サブセクションで、削除するモノを見つけます。

- モノを右クリックしてコンテキストメニューを開き、[Delete Thing] (モノの削除) を選択します。
- このステップが完了すると、削除したモノは [Things] (モノ) サブセクションで利用できなくなります。

 Note

削除できるのは、証明書がアタッチされていないモノのみです。

AWS IoT 証明書

証明書は、ユーザーの AWS IoT サービスとデバイス間のセキュアな接続を作成するための一般的な方法です。X.509 証明書は、X.509 パブリックキーインフラストラクチャ規格を使用して、パブリックキーと証明書内の ID を関連付けるための、デジタル証明書です。AWS IoT 証明書の詳細については、AWS IoT デベロッパーガイドの [「認証 \(IoT\)」](#) を参照してください。

証明書の管理

AWS ツールキットには、AWS Explorer から直接証明書を管理する AWS IoT さまざまな方法が用意されています。その概要を以下のステップで説明します。

- [Create a certificate](#)
- [Change a certificate status](#)
- [Attach a policy to a certificate](#)
- [Delete a certificate](#)

AWS IoT 証明書を作成するには

X.509 証明書は、 のインスタンスに接続するために使用されます AWS IoT。

1. AWS Explorer から IoT IoTサービスセクションを展開し、証明書を開きます (右クリック) 。
2. ダイアログボックスを開くには、コンテキストメニューから [Create Certificate] (証明書の作成) を選択します。
3. RSA キーペアと X.509 証明書を保存するには、ローカルファイルシステム内のディレクトリを選択します。

 Note

- デフォルトのファイル名には、証明書 ID がプレフィックスとして含まれます。
- X.509 証明書のみが AWS アカウントサービスを通じて AWS IoT に保存されます。
- RSA key pair は 1 回だけ発行でき、プロンプトが表示されたら、ファイルシステム内の安全な場所に保存してください。
- 証明書またはキーペアをファイルシステムに保存できない場合、AWS Toolkit は から証明書を削除します AWS アカウント。

証明書のステータスを変更するには

個々の証明書のステータスは AWS Explorer の証明書 ID の横に表示され、アクティブ、非アクティブ、または取り消すことができます。

 Note

- 証明書を使用してデバイスを AWS IoT サービスに接続する前に、アクティブなステータスが必要です。
- 非アクティブ証明書は、以前に非アクティブ化されたか、デフォルトで非アクティブであるかにかかわらず、アクティブ化することができます。
- 取消済み証明書は再有効化できません。

1. AWS Explorer から、IoT サービスセクションを展開します。
 2. [Certificates] (証明書) サブセクションで、変更する証明書を見つけます。
 3. 証明書を右クリックしてコンテキストメニューを開き、その証明書で使用可能なステータス変更オプションを表示します。
- 証明書のステータスが非アクティブである場合、[activate] (アクティブ化) を選択してステータスをアクティブに変更します。
 - 証明書に アクティブ ステータスがある場合、[非アクティブ] を選択してステータスを非アクティブに変更します。

- 証明書に アクティブ または 非アクティブ ステータスがある場合は、[取り消す] を選択し、ステータスを失効しましたに変更します。

 Note

これらのステータス変更の各アクションは、[Things] (モノ) サブセクションに表示中のモノにアタッチされている証明書を選択した場合に使用できます。

IoT ポリシーを証明書にアタッチするには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Certificates] (証明書) サブセクションで、変更する証明書を見つけます。
3. 証明書を右クリックしてコンテキストメニューを表示し、[Attach Policy] (ポリシーのアタッチ) を選択して、使用可能なポリシーのリストを含む入力セレクトを開きます。
4. 証明書にアタッチするポリシーを選択します。
5. このステップが完了すると、選択したポリシーがサブメニュー項目として証明書に追加されます。

証明書から IoT ポリシーをデタッチするには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Certificates] (証明書) サブセクションで、変更する証明書を見つけます。
3. 証明書を展開し、デタッチするポリシーを見つけます。
4. ポリシーを右クリックしてコンテキストメニューを開き、[Detach] (デタッチ) を選択します。
5. このステップが完了すると、ポリシーは証明書からアクセスできなくなります。ただし、[Policy] (ポリシー) サブセクションからはアクセスできます。

証明書を削除するには

1. AWS Explorer から IoT IoTサービスの見出しを展開します。
2. [Certificates] (証明書) サブセクションで、削除する証明書を見つけます。

3. 証明書を右クリックしてコンテキストメニューを開き、[Delete Certificate] (証明書の削除) を選択します。

 Note

モノにアタッチされている証明書や、アクティブなステータスの証明書は削除できません。ポリシーがアタッチされた証明書を削除できます。

AWS IoT ポリシー

AWS IoT コアポリシーは JSON ドキュメントを通じて定義されます。各ポリシーには、少なくとも 1 つのステートメントを含めます。ポリシーは AWS IoT AWS、とデバイスが相互にやり取りする方法を定義します。ポリシードキュメントの作成方法の詳細については、AWS IoT デベロッパーガイドの「[IoT ポリシー](#)」を参照してください。

 Note

名前付きポリシーは、バージョン管理されているため、ロールバックできます。AWS Explorer では、IoT ポリシーは AWS IoT サービスのポリシーサブセクションに一覧表示されます。ポリシーを展開すると、ポリシーバージョンを表示できます。デフォルトバージョンはアスタリスク (*) で示されます。

ポリシーの管理

AWS Cloud9 IDE には、AWS IoT サービスポリシーを管理する方法がいくつか用意されています。以下は、VS Code の AWS Explorer から直接ポリシーを管理または変更する方法です。

- [Create a policy](#)
- [Upload a new policy version](#)
- [Edit a policy version](#)
- [Change the policy version default](#)
- [Change the policy version default](#)

AWS IoT ポリシーを作成するには

Note

Explorer から新しいポリシーを作成できます AWS 。ただし、ポリシーを定義する JSON ドキュメントがファイルシステムに既に存在している必要があります。

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Policies] (ポリシー) サブセクションを右クリックしてコンテキストメニューを開き、[Policy Name] (ポリシー名) 入力フィールドから、[Create Policy from Document] (ドキュメントからポリシーを作成) を選択します。
3. 名前を入力し、プロンプトに従って、ファイルシステムから JSON ドキュメントを選択するよう指示するダイアログを開きます。
4. ポリシー定義を含む JSON ファイルを選択します。これが完了すると、ポリシーは AWS Explorer で利用可能になります。

新しい AWS IoT ポリシーバージョンをアップロードするには

ポリシーの新しいバージョンを作成するには、JSON ドキュメントをポリシーにアップロードします。

Note

AWS Explorer を使用して新しいバージョンを作成するには、新しい JSON ドキュメントがファイルシステムに存在する必要があります。

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Policies] (ポリシー) サブセクションを展開して、AWS IoT ポリシーを表示します。
3. 更新するポリシーを右クリックしてコンテキストメニューを開き、[Create new version from Document] (ドキュメントから新しいバージョンを作成) を選択します。
4. ダイアログボックスが開いたら、ポリシー定義の更新を含む JSON ファイルを選択します。

新しいバージョンは、AWS Explorer のポリシーからアクセスできます。

AWS IoT ポリシーバージョンを編集するには

を使用してポリシードキュメントを開いて編集できます AWS Cloud9。ドキュメントの編集が終了したら、ファイルシステムに保存します。次に、AWS Explorer から AWS IoT サービスにアップロードします。

1. AWS Explorer から、IoT サービスセクションを展開します。
2. ポリシー を拡張し、更新するポリシーを見つけます。
3. [Policy Name] (ポリシー名) を開くには、[Document] (ドキュメント) から [Create Policy] (ポリシーの作成) を選択します。
4. 更新するポリシーを展開し、編集するポリシーバージョンを右クリックしてコンテキストメニューを開きます。
5. でポリシーバージョンを開くには AWS Cloud9、コンテキストメニューから表示を選択してポリシーバージョンを開きます。
6. ポリシードキュメントが開いたら、編集して変更を保存します。

Note

この時点で、ポリシーに加えた変更は、ローカルファイルシステムにのみ保存されます。Explorer でバージョンを更新して追跡するには、AWS 「」の手順を繰り返します [Upload a new policy version](#)。

新しいポリシーバージョンのデフォルトを選択するには

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Policies] (ポリシー) サブセクションを展開し、更新するポリシーを見つけます。
3. 更新するポリシーを展開し、設定するポリシーバージョンを右クリックしてコンテキストメニューを開き、[Set as Default] (デフォルトとして設定) を選択します。

これが完了すると、選択した新しいデフォルトバージョンの横に星が表示されます。

ポリシーを削除するには

Note

ポリシーまたはポリシーバージョンを削除する前に、以下の条件を満たしていることを確認してください。

- 証明書にアタッチされているポリシーは削除できません。
- デフォルト以外のバージョンを持つポリシーは削除できません。
- ポリシーのデフォルトバージョンは、別のデフォルトバージョンを選択した場合にのみ削除できます。それ以外は、ポリシー全体が削除されます。
- ポリシー全体を削除する前に、同じポリシーのデフォルト以外のバージョンをすべて削除する必要があります。

1. AWS Explorer から、IoT サービスセクションを展開します。
2. [Policies] (ポリシー) サブセクションを展開し、更新するポリシーを見つけます。
3. 更新するポリシーを展開し、削除するポリシーバージョンを右クリックしてコンテキストメニューを開き、[Delete] (削除) を選択します。
4. バージョンが削除されると、AWS Explorer からは表示されなくなります。
5. ポリシーのデフォルトバージョンのみが残っている場合は、親ポリシーを右クリックしてコンテキストメニューを開き、[Delete] (削除) を選択します。

Amazon Elastic Container Service を使用する

AWS Cloud9 IDE は、[Amazon Elastic Container Service \(Amazon ECS\)](#) の一部をサポートしています。IDE を使用して Amazon ECS AWS Cloud9 リソースを管理できます。例えば、タスク定義を作成できます。

トピック

- [AWS Toolkit for の Amazon ECS Exec AWS Cloud9](#)

AWS Toolkit for の Amazon ECS Exec AWS Cloud9

AWS Toolkit for を使用すると、Amazon Elastic Container Service (Amazon ECS) コンテナで単一のコマンドを発行できます AWS Cloud9。この操作は、Amazon Exec の機能を使用して行うことができます。

Important

Amazon ECS Exec を有効または無効にすると、AWS アカウントでの ECS リソースの状態が変わります。変更には、サービスの停止と再起動が含まれます。さらに、Amazon ECS Exec が有効になっている間にリソースの状態を変更すると、予期しない結果が生じる可能性があります。Amazon ECS の詳細については、Amazon ECS デベロッパーガイドの「[Amazon ECS Exec を使用してデバッグする](#)」を参照してください。

Amazon Exec の前提条件

Amazon ECS Exec の機能を使用する前に、いくつかの前提条件を満たす必要があります。

Amazon ECS の要件

タスクが Amazon EC2 または でホストされているかどうか AWS Fargate、および Amazon ECS Exec のバージョン要件によって異なります。

- Amazon EC2 を使用する場合は、2021 年 1 月 20 日以降にリリースされた Amazon ECS 最適化 AMI を、エージェントバージョン 1.50.2 以降で使用する必要があります。詳細については、Amazon ECS デベロッパーガイドの「[Amazon ECS に最適化された AMI](#)」を参照してください。
- を使用する場合は AWS Fargate、プラットフォームバージョン 1.4.0 以降を使用する必要があります。詳細については、Amazon ECS デベロッパーガイドの「[AWS Fargate プラットフォームのバージョン](#)」を参照してください。

AWS アカウント設定と IAM アクセス許可

Amazon ECS Exec の機能を使用するには、既存の Amazon Exec のクラスターが AWS アカウントに関連付けられている必要があります。Amazon ECS Exec は Systems Manager を使用してクラスター内のコンテナとの接続を確立します。Amazon ECS で SSM サービスと通信するには、特定のタスクの IAM ロールのアクセス許可が必要です。

Amazon ECS Exec に固有の IAM ロールとポリシーについては、Amazon ECS デベロッパーガイドの「[ECS Exec に必要な IAM アクセス許可](#)」を参照してください。

Amazon ECS Exec の操作

AWS Toolkit for の AWS Explorer から直接 Amazon ECS Exec を有効または無効にできます AWS Cloud9。Amazon ECS Exec を有効にしたら、Amazon ECS メニューからコンテナを選択し、それらに対してコマンドを実行します。

Amazon ECS Exec の有効化

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 変更するサービスを含むクラスターを拡張します。
3. サービスのコンテキストメニュー (右クリック) を開き、[Enable Command Execution] (コマンドの実行を有効にする) を選択します

Important

このステップにより、サービスの新規デプロイが開始されます。これには数分かかることがあります。詳細については、このセクションの冒頭にある注意事項を参照してください。

Amazon ECS Exec の無効化

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 必要なサービスを含むクラスターを展開します。
3. サービスのコンテキストメニュー (右クリック) を開き、[Disable Command Execution] (コマンド実行を無効にする) を選択します

Important

このステップにより、サービスの新規デプロイが開始されます。これには数分かかることがあります。詳細については、このセクションの冒頭にある注意事項を参照してください。

コンテナに対するコマンドの実行

AWS Explorer を使用してコンテナに対してコマンドを実行するには、Amazon ECS Exec を有効にする必要があります。有効になっていない場合は、このセクションの「[Amazon ECS Exec の有効化](#)」の手順を参照してください。

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 必要なサービスを含むクラスターを展開します。
3. サービスを展開して、関連するコンテナを一覧表示します。
4. コンテナのコンテキストメニュー (右クリック) を開き、[Run Command in Container] (コンテナでコマンドを実行) を選択します
5. 実行中のタスクのリストを含むプロンプトが開きます。必要なタスクの ARN を選択します。

Note

実行中のタスクが 1 つだけの場合、プロンプトは開きません。代わりに、タスクは自動選択されます。

6. プロンプトが表示されたら、実行するコマンドを入力し、Enter キーを押して続行します。

Amazon EventBridge スキーマの使用

Toolkit AWS for AWS Cloud9 は、[Amazon EventBridge](#) をサポートしています。AWS Toolkit for を使用すると AWS Cloud9、スキーマなど、EventBridge の特定の側面を操作できます。

トピック

- [Amazon EventBridge スキーマの使用](#)

Amazon EventBridge スキーマの使用

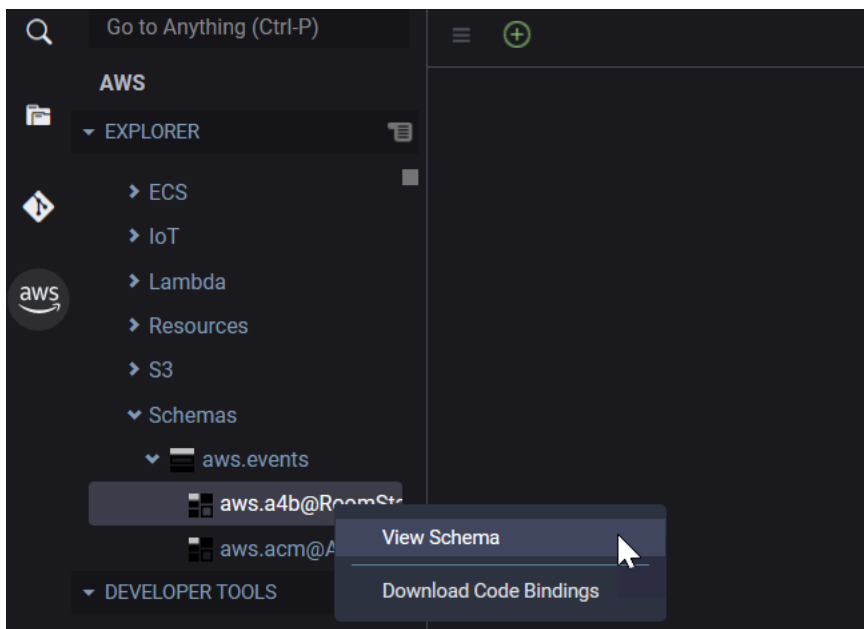
AWS Toolkit for を使用して AWS Cloud9、[Amazon EventBridge スキーマ](#)でさまざまなオペレーションを実行できます。

前提条件

使用する EventBridge スキーマは、お客様の AWS アカウントに存在している必要があります。存在していない場合は、スキーマを作成またはアップロードします。詳細については、Amazon EventBridge ユーザーガイドの「[Amazon EventBridge スキーマ](#)」を参照してください。

使用可能なスキーマの表示

1. AWS Explorer で、[スキーマ] を展開します。
2. 表示するスキーマが含まれているレジストリの名前を展開します。たとえば、AWS が提供するスキーマの多くは aws.events レジストリにあります。
3. エディタでスキーマを表示するには、スキーマを右クリックしてコンテキストメニューを開き、[View Schema] (スキーマの表示) を選択します。



使用可能なスキーマの検索

AWS Explorer で、次のいずれかの操作を行います。

- 検索するスキーマのタイトルの入力を開始します。AWS Explorer で、一致を含むスキーマタイトルがハイライト表示されます (ハイライト表示されたタイトルを表示するには、レジストリを展開する必要があります)。

- [Schemas] (スキーマ) を右クリックしてコンテキストメニューを開き、[Search Schemas] (スキーマを検索) を選択します。または、[Schemas] (スキーマ) を展開し、検索するスキーマが含まれているレジストリを右クリックしてコンテキストメニューを開き、[Search Schemas in Registry] (レジストリのスキーマを検索) を選択します。[EventBridge Schemas Search] (EventBridge スキーマの検索) ダイアログボックスで、検索するスキーマのタイトルの入力を開始します。一致部分を含むスキーマタイトルがダイアログボックスに表示されます。

ダイアログボックスにスキーマを表示するには、スキーマのタイトルを選択します。

使用可能なスキーマのコードの生成

1. AWS Explorer で、[スキーマ] を展開します。
2. コードを生成するスキーマが含まれているレジストリの名前を展開します。
3. スキーマのタイトルを右クリックしてコンテキストメニューを開き、[Download code bindings] (コードバインディングのダウンロード) を選択します。
4. 表示されたウィザードのページで、以下を選択します。
 - スキーマのバージョン
 - コードのバインド言語
 - 生成されたコードを保存するローカル開発マシン上のワークスペースフォルダ

のチュートリアル AWS Cloud9

を初めてお使いですか AWS Cloud9？「[開始方法: ベーシックチュートリアル](#)」で IDE の演習を行います。

これらのチュートリアルとサンプルコードを試して、さまざまなプログラミング言語と AWS サービス AWS Cloud9 でを使用して知識と自信を深めてください。

トピック

- [AWS CLI および の aws-shell チュートリアル AWS Cloud9](#)
- [AWS CodeCommit の チュートリアル AWS Cloud9](#)
- [の Amazon DynamoDB チュートリアル AWS Cloud9](#)
- [AWS CDK の チュートリアル AWS Cloud9](#)
- [の LAMP チュートリアル AWS Cloud9](#)
- [の WordPress チュートリアル AWS Cloud9](#)
- [の Java チュートリアル AWS Cloud9](#)
- [の C++ チュートリアル AWS Cloud9](#)
- [の Python チュートリアル AWS Cloud9](#)
- [の .NET チュートリアル AWS Cloud9](#)
- [の Node.js チュートリアル AWS Cloud9](#)
- [の PHP チュートリアル AWS Cloud9](#)
- [AWS の SDK for Ruby AWS Cloud9](#)
- [の Go チュートリアル AWS Cloud9](#)
- [の TypeScript チュートリアル AWS Cloud9](#)
- [の Docker チュートリアル AWS Cloud9](#)
- [関連チュートリアル](#)

AWS CLI および の aws-shell チュートリアル AWS Cloud9

次のチュートリアルでは、AWS Cloud9 開発環境で AWS Command Line Interface (AWS CLI)、aws-shell、またはその両方を設定できます。AWS CLI と aws-shell は、のすべての部分とやり取りするための一貫したインターフェイスを提供する統合ツールです AWS。AWS CLI の代わり

にを使用して、操作するコマンド AWS マネジメントコンソール をすばやく実行できます。これらのコマンドの一部は AWS、AWS CLI または で実行できます AWS CloudShell。

の詳細については AWS CLI、[AWS Command Line Interface 「ユーザーガイド」](#) を参照してください。aws-shell については、以下のリソースを参照してください。

- GitHub ウェブサイトの「[aws-shell](#)」
- pip ウェブサイトの「[aws-shell](#)」

とやり取り AWS CLI するために で実行できるコマンドのリストについては AWS、[AWS CLI 「コマンドリファレンス」](#) を参照してください。aws プレフィックスなしでコマンドを起動する以外は AWS CloudShell、 で同じコマンドを使用できます。

このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 環境に AWS CLI、aws-shell、またはその両方をインストールする](#)
- [ステップ 2: 環境で認証情報管理を設定する](#)
- [ステップ 3: 環境内の AWS CLI または aws-shell を使用して基本的なコマンドを実行する](#)
- [ステップ 4: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 環境に AWS CLI、aws-shell、またはその両方をインストールする

このステップでは、AWS Cloud9 IDE を使用して AWS CLI、aws-shell、またはその両方を環境にインストールし、操作するコマンドを実行できます AWS。

AWS Cloud9 EC2 開発環境を使用していて、のみを使用する場合は AWS CLI、「」に進むことができます [ステップ 3: 環境内の AWS CLI または aws-shell を使用して基本的なコマンドを実行する](#)。これは、AWS CLI が EC2 環境に既にインストールされており、AWS アクセス認証情報のセットが環境に既に設定されているためです。詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。

EC2 環境を使用していない場合は、以下を実行して AWS CLI をインストールします。

1. 環境を開いた状態で、IDE で、が既にインストール AWS CLI されているかどうかを確認します。ターミナルで `aws --version` コマンドを実行します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。) AWS CLI がインストールされている場合、Python のバージョン番号や Amazon EC2 インスタンスまたは独自のサーバーのオペレーティングシステムのバージョン番号などの情報とともに、バージョン番号が表示されます。AWS CLI がインストールされている場合は、「」に進みます [ステップ 2: 環境で認証情報管理を設定する](#)。
2. をインストールするには AWS CLI、AWS Command Line Interface ユーザーガイドの「[AWS Command Line Interface](#)」のインストール」を参照してください。たとえば、Amazon Linux を実行する EC2 環境の場合は、ターミナルで次の 3 つのコマンドを 1 つずつ実行して、AWS CLI をインストールします。

```
sudo yum -y update           # Install the latest system updates.
sudo yum -y install aws-cli  # Install the AWS CLI.
aws --version                # Confirm the AWS CLI was installed.
```

Ubuntu Server を実行する EC2 環境の場合は、ターミナルで代わりに次の 3 つのコマンドを 1 つずつ実行して、AWS CLI をインストールします。

```
sudo apt update              # Install the latest system updates.
sudo apt install -y awscli   # Install the AWS CLI.
aws --version                # Confirm the AWS CLI was installed.
```

aws-shell をインストールする場合は、以下を実行します。

1. 環境が開いている状態で、IDE で aws-shell が既にインストールされているかどうかをチェックします。ターミナルで **aws-shell** コマンドを実行します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。) aws-shell がインストールされている場合は、aws> プロンプトが表示されます。aws-shell がインストール済みである場合は、「[ステップ 2: 環境で認証情報管理を設定する](#)」に進んでください。
2. aws-shell をインストールするには、pip を使用します。pip を使用するには、Python がインストールされている必要があります。

Python が既にインストールされているかどうかをチェックするには (および必要に応じてインストールするには)、Python のサンプルの [ステップ 1: Python をインストールする](#) にある指示に従ってから、このトピックに戻ってください。

pip がインストール済みであるかどうかを確認するには、ターミナルで、**pip --version** コマンドを実行します。pip がインストールされている場合は、バージョン番号が表示されます。pip がインストールされていない場合は、ターミナルで次の 3 つのコマンドを 1 つずつ実行して、pip をインストールします。

```
wget https://bootstrap.pypa.io/get-pip.py # Get the pip install file.
sudo python get-pip.py                    # Install pip. (You might need to run
'sudo python2 get-pip.py' or 'sudo python3 get-pip.py' instead, depending on how
Python is installed.)
rm get-pip.py                             # Delete the pip install file, as it is
no longer needed.
```

3. pip を使用して aws-shell をインストールするには、次のコマンドを実行します。

```
sudo pip install aws-shell
```

ステップ 2: 環境で認証情報管理を設定する

AWS CLI または aws-shell を使用して AWS サービスを呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は、AWS CLI または aws-shell に、その呼び出しを行うための適切なアクセス許可があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

AWS Cloud9 EC2 開発環境を使用している場合は、「」に進むことができます[ステップ 3: 環境内の AWS CLI または aws-shell を使用して基本的なコマンドを実行する](#)。これは、認証情報が既に EC2 環境で設定されているためです。詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。

EC2 環境を使用していない場合は、手動で環境内に認証情報を保存する必要があります。これを行うには、[環境 AWS のサービスからの呼び出し AWS Cloud9](#) の指示を実行してから、このトピックに戻ります。

ステップ 3: 環境内の AWS CLI または aws-shell を使用して基本的なコマンドを実行する

このステップでは、環境の AWS CLI または aws-shell を使用して Amazon S3 にバケットを作成し、使用可能なバケットを一覧表示してから、バケットを削除します。

1. aws-shell を使用するがまだ起動していない場合は、aws-shell コマンドを実行して aws-shell を起動します。aws> プロンプトが表示されます。
2. バケットを作成します。aws s3 mb AWS CLI または aws-shell で s3 mb コマンドを実行し、作成するバケットの名前を指定します。この例では、という名前のバケットを使用します。ここでcloud9-123456789012-bucket、123456789012は AWS アカウント ID です。別の名前を使用する場合は、このステップ全体でそれを置き換えてください。

```
aws s3 mb s3://cloud9-123456789012-bucket # For the AWS CLI.
s3 mb s3://cloud9-123456789012-bucket    # For the aws-shell.
```

Note

バケット名は、AWS アカウントだけでなく AWS、すべての で一意である必要があります。上記で提案されたバケット名は、一意のバケット名を作成するために便利です。BucketAlreadyExists というエラーを含むメッセージが表示された場合は、別のバケット名で再度コマンドを実行する必要があります。

3. 使用可能なバケットを一覧表示します。で aws s3 ls コマンドを実行する AWS CLI か、aws-shell で s3 ls コマンドを実行します。使用可能なバケットのリストが表示されます。
4. バケットを削除します。aws s3 rb AWS CLI または aws-shell で s3 rb コマンドを実行し、削除するバケットの名前を指定します。


```
aws s3 rb s3://cloud9-123456789012-bucket # For the AWS CLI.  
s3 rb s3://cloud9-123456789012-bucket    # For the aws-shell.
```

バケットが削除されたかどうかを確認するには、で **aws s3 ls** コマンドを再度実行する AWS CLI か、aws-shell で **s3 ls** コマンドを再度実行します。削除されたバケットの名前がリストに表示されていないことを確認します。

Note

バケットを使用し続ける場合は、削除する必要はありません。詳細については、Amazon Simple Storage Service ユーザーガイドの[バケットへのオブジェクトの追加](#)を参照してください。AWS CLI コマンドリファレンスにある[s3 コマンド](#)も参照してください。(バケットを削除しない場合、AWS アカウントに対して継続的な料金が発生する可能性があることに注意してください)。

の実験を続行するには AWS CLI、AWS Command Line Interface 「ユーザーガイド」の「[Amazon Web Services の使用](#)」および[AWS CLI 「コマンドリファレンス」](#)を参照してください。aws-shell を使用して実験を続行するには、[AWS CLI コマンドリファレンス](#)を参照してください。コマンドをスタートするときに aws プレフィックスを付けないように注意してください。

ステップ 4: クリーンアップする

aws-shell を使用している場合、**.exit** コマンドまたは **.quit** コマンドを実行して使用を停止できます。

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

AWS CodeCommit の チュートリアル AWS Cloud9

AWS CodeCommit チュートリアルを使用して、CodeCommit のリモートコードリポジトリとやり取りする AWS Cloud9 開発環境を設定できます。CodeCommit は、AWS クラウドでの Git リポジトリのプライベートな保存と管理を有効にするソースコード制御サービスです。CodeCommit の詳細については、[AWS CodeCommit ユーザーガイド](#)を参照してください。

このチュートリアルに従い、このサンプルを作成すると、に料金が発生する可能性があります AWS アカウント。これには、Amazon EC2 や CodeCommit などのサービスで発生する可能性がある料金

も含まれます。詳細については、「[Amazon EC2 コスト](#)」および「[AWS CodeCommit コスト](#)」を参照してください。

- [前提条件](#)
- [ステップ 1: 必要なアクセス許可を持つ IAM グループを設定する](#)
- [ステップ 2: リポジトリを作成する AWS CodeCommit](#)
- [ステップ 3: 環境をリモートリポジトリに接続する](#)
- [ステップ 4: リモートリポジトリのクローンを環境に作成する](#)
- [ステップ 5: リポジトリにファイルを追加する](#)
- [ステップ 6: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なアクセス許可を持つ IAM グループを設定する

AWS 認証情報が の管理者ユーザーに関連付けられていて AWS アカウント、そのユーザーを使用して CodeCommit を操作する場合、「」の [ステップ 2: リポジトリを作成する AWS CodeCommit](#) 「」に進みます。

このステップは、[AWS マネジメントコンソール](#) または [AWS コマンドラインインターフェイス \(AWS CLI\)](#) を使って完了できます。

コンソールを使用して、必要なアクセス許可を持つ IAM グループを設定する

1. まだサインインしていない場合は AWS マネジメントコンソール、 にサインインします。

このステップでは、AWS アカウントの管理者ユーザーの認証情報を使用してサインインすることをお勧めします。これが実行できない場合は、AWS アカウント 管理者にお問い合わせください。

2. [IAM コンソール] を開きます。これを行うには、コンソールのナビゲーションバーで、[サービス] を選択します。次に、[IAM] を選択します。
3. [グループ] を選択します。
4. グループの名前を選択します。
5. [アクセス許可] タブの [管理ポリシー] で [ポリシーのアタッチ] を選択します。
6. ポリシー名のリストで、次のいずれかのボックスを選択します。
 - AWSCodeCommitPowerUser を選択すると、CodeCommit のすべての機能とリポジトリ関連リソースにアクセスできます。ただし、これによりCodeCommit リポジトリを削除したり AWS のサービス、Amazon CloudWatch Events などの他の でリポジトリ関連のリソースを作成または削除したりすることはできません。
 - AWSCodeCommitFullAccess を選択して、の CodeCommit リポジトリおよび関連リソースを完全に制御します AWS アカウント。これにはリポジトリを削除する機能も含まれます。

どちらのポリシー名もリストに表示されない場合は、[Filter] (フィルター) ボックスにポリシー名を入力して表示させます。

7. [ポリシーをアタッチ] を選択します。

これらの AWS 管理ポリシーがグループに付与するアクセス許可のリストを確認するには、「AWS CodeCommit ユーザーガイド」の[AWS 「 の 管理 \(事前定義\) ポリシー AWS CodeCommit 」](#)を参照してください。

ステップ [2: リポジトリを作成する AWS CodeCommit](#)に進みます。

を使用して必要なアクセス許可を持つ IAM グループを設定する AWS CLI

IAM attach-group-policy コマンドを実行し、グループの名前と、必要なアクセス許可を記述する AWS 管理ポリシーの Amazon リソースネーム (ARN) を指定します。構文は次のとおりです。

```
aws iam attach-group-policy --group-name MyGroup --policy-arn POLICY_ARN
```

前述のコマンドで、MyGroup をグループの名前に置き換えます。を AWS マネージドポリシーの ARN POLICY_ARNに置き換えます。

- `arn:aws:iam::aws:policy/AWSCodeCommitPowerUser` を選択すると、CodeCommit のすべての機能とリポジトリ関連リソースにアクセスできます。ただし、CodeCommit リポジトリを削除したり AWS のサービス、Amazon CloudWatch Events などの他のでリポジトリ関連のリソースを作成または削除したりすることはできません。
- `arn:aws:iam::aws:policy/AWSCodeCommitFullAccess` は、の CodeCommit リポジトリおよび関連リソースを完全に制御できます AWS アカウント。これにはリポジトリを削除する機能も含まれます。

これらの AWS 管理ポリシーがグループに付与するアクセス許可のリストを確認するには、「AWS CodeCommit ユーザーガイド」の[AWS 「の 管理 \(事前定義\) ポリシー AWS CodeCommit」](#)を参照してください。

ステップ 2: CodeCommit でリポジトリを作成する

このステップでは、CodeCommit コンソールを使用して CodeCommit にリモートコードリポジトリを作成します。

すでに CodeCommit リポジトリがある場合は、「[ステップ 3: 環境をリモートリポジトリに接続する](#)」に進みます。

このステップは、[AWS マネジメントコンソール](#)または [AWS コマンドラインインターフェース\(AWS CLI\)](#)を使って完了できます。

コンソールを使用して CodeCommit でリポジトリを作成する

1. 前のステップで管理者ユーザー AWS マネジメントコンソール として にサインインしていて、管理者ユーザーを使用してリポジトリを作成したくないとします。次に、 からサインアウトします AWS マネジメントコンソール。
2. <https://console.aws.amazon.com/codecommit> で CodeCommit コンソールを開きます。
3. コンソールのナビゲーションバーで、リージョンセレクタを使用して、リポジトリの作成先の AWS リージョン を選択します (例: 米国東部 (オハイオ))。
4. ウェルカムページが表示された場合は、[今すぐ始める] を選択します。それ以外の場合は、[リポジトリの作成] を選択します。
5. [Create repository] (リポジトリの作成) ページの [Repository name] (レポジトリ名) に、新しいレポジトリの名前を入力します (例: MyDemoCloud9Repo)。 (別の名前を選択する場合は、このサンプル全体でそれに置き換えてください。)

6. (オプション) [Description] (説明) に、リポジトリに関する説明を入力します。例えば、This is a demonstration repository for the AWS Cloud9 sample. と入力します。
7. [リポジトリの作成] を選択します。[リポジトリに接続] ペインが表示されます。[閉じる] を選択します。このトピックの後半で別の方法でリポジトリに接続するためです。

「[ステップ 3: 環境をリモートリポジトリに接続する](#)」に進みます。

を使用して CodeCommit でリポジトリを作成する AWS CLI

AWS CodeCommit create-repository コマンドを実行します。リポジトリの名前、オプションの説明、リポジトリ AWS リージョン を作成する を指定します。

```
aws codecommit create-repository --repository-name MyDemoCloud9Repo --repository-description "This is a demonstration repository for the AWS Cloud9 sample." --region us-east-2
```

前述のコマンドで、us-east-2 を AWS リージョン の ID に置き換えてリポジトリを作成します。サポートされているリージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「[AWS CodeCommit](#)」を参照してください。

(別のリポジトリ名を使用する場合は、このサンプル全体でそれに置き換えてください。)

ステップ 3: 環境をリモートリポジトリに接続する

このステップでは、IDE AWS Cloud9 を使用して、前のステップで作成または識別した CodeCommit リポジトリに接続します。

Note

ビジュアルインターフェイスを使って Git を操作したい場合は、リモートリポジトリをクローンすることができます。次に、IDE の [Git パネル](#) 機能を使用してファイルを追加できます。

使用する AWS Cloud9 開発環境の種類に応じて、次の一連の手順のいずれかを完了します。

環境タイプ	以下の手順に従います
EC2 環境	1. IDE のターミナルセッションから、次の 2 つのコマンドを実行します。 <pre>git config --global credential l.helper '!aws codecommit credential l-helper \$@' git config --global credential l.UseHttpPath true</pre> 詳細については、「AWS CodeCommit ユーザーガイド」の「ステップ 2: EC2 AWS Cloud9 開発環境で AWS CLI と統合して認証情報ヘルパーを設定する」を参照してください。 AWS Cloud9 AWS CodeCommit 2. このトピック後半の「ステップ 4: リモートリポジトリのクローンを環境に作成する」に進みます。
SSH 環境	1. Git が環境にまだインストールされていない場合は、IDE のターミナルセッションを使用してインストールします。詳細については、「AWS CodeCommit ユーザーガイド」の「Linux、macOS、または Unix の AWS CodeCommit リポジトリへの SSH 接続のセットアップステップ」の「ステップ 2: インストールGit」を参照してください。 2. AWS CodeCommit ユーザーガイドの「Linux、macOS、または Unix のリポジトリへの SSH 接続のセットアップ手順」の「ステップ 3: Linux、macOS、または Unix の認証情報を設定する」を完了します。 AWS CodeCommit macOS にサインイン AWS マネジメントコンソールして IAM コンソールを開くように指示され

環境タイプ	以下の手順に従います
	た場合は、 の管理者ユーザーの認証情報を使用してサインインすることをお勧めします AWS アカウント。これができない場合は、 AWS アカウント 管理者に確認してください。 3. このトピック後半の「ステップ 4: リモートリポジトリのクローンを作成する」に進みます。

ステップ 4: リモートリポジトリのクローンを作成する

このステップでは、IDE AWS Cloud9 を使用して CodeCommit のリモートリポジトリを環境にクローンします。

リポジトリのクローンを作成するには、**git clone** コマンドを実行します。`CLONE_URL` をリポジトリのクローン URL に置き換えます。

```
git clone CLONE_URL
```

EC2 環境には、`https://` で始まる HTTPS クローン URL を指定します。SS 環境 には、`ssh://` で始まる SSH クローン URL を指定します。

リポジトリの完全なクローン URL を取得するには、AWS CodeCommit 「[ユーザーガイド](#)」の [AWS CodeCommit 「コンソールを使用してリポジトリの詳細を表示する」](#) を参照してください。

リポジトリにファイルがない場合は、「You appear to have cloned an empty repository.」のような警告メッセージが表示されます。これは想定される動作です。これについては後で説明します。

ステップ 5: リポジトリにファイルを追加する

このステップでは、AWS Cloud9 環境にクローンしたリポジトリに 3 つのシンプルなファイルを作成します。次に、クローンしたリポジトリの Git ステージング領域にファイルを追加します。最後に、ステージングされたファイルをコミットして、そのコミットを CodeCommit のリモートリポジトリにプッシュします。

クローンされたリポジトリにすでにファイルが含まれている場合、操作は不要で、このサンプルの残りの部分はスキップできます。

リポジトリにファイルを追加するには

1. 新しいファイルを作成します。メニューバーで [File (ファイル)]、[New File (新規ファイル)] の順に選択します。
2. ファイルに次のコンテンツを入力し、File, Save を選択して、AWS Cloud9 環境の MyDemoCloud9Repo ディレクトリbird.txtに としてファイルを保存します。

```
bird.txt
-----
Birds are a group of endothermic vertebrates, characterized by feathers,
toothless beaked jaws, the laying of hard-shelled eggs, a high metabolic
rate, a four-chambered heart, and a lightweight but strong skeleton.
```

 Note

このファイルが正しいディレクトリに保存されたことを確認するには、[Save As] (名前を付けて保存)ダイアログボックスで MyDemoCloud9Repo フォルダを選択します。次に、[Folder] (フォルダ) に /MyDemoCloud9Repo が表示されていることを確認します。

3. 次の内容で、insect.txt および reptile.txt という名前のファイルを 2 つ作成します。ファイルを MyDemoCloud9Repo ディレクトリに保存します。

```
insect.txt
-----
Insects are a class of invertebrates within the arthropod phylum that
have a chitinous exoskeleton, a three-part body (head, thorax, and abdomen),
three pairs of jointed legs, compound eyes, and one pair of antennae.
```

```
reptile.txt
-----
Reptiles are tetrapod (four-limbed vertebrate) animals in the class
Reptilia, comprising today's turtles, crocodilians, snakes,
amphisbaenians, lizards, tuatara, and their extinct relatives.
```

4. ターミナルで、**cd** コマンドを実行して MyDemoCloud9Repo ディレクトリに切り替えます。

```
cd MyDemoCloud9Repo
```

5. **git status** コマンドを実行して、ファイルが正しく MyDemoCloud9Repo ディレクトリに保存されたことを確認します。3 つのファイルはすべて追跡されていないファイルとして表示されます。

```
Untracked files:
(use "git add <file>..." to include in what will be committed)

    bird.txt
    insect.txt
    reptile.txt
```

6. **git add** コマンドを実行して、ファイルを Git ステージング領域に追加します。

```
git add --all
```

7. **git status** コマンドを実行して、ファイルが正しく Git ステージング領域に追加されたことを確認します。3 つのファイルがすべてコミットされる変更としてリストされます。

```
Changes to be committed:
(use "git rm --cached <file>..." to unstage)

    new file:   bird.txt
    new file:   insect.txt
    new file:   reptile.txt
```

8. **git commit** コマンドを実行して、ステージングされたファイルをコミットします。

```
git commit -m "Added information about birds, insects, and reptiles."
```

9. **git push** コマンドを実行して、コミットを CodeCommit のリモートリポジトリにコマンドをプッシュします。

```
git push -u origin master
```

10. ファイルが正常にプッシュされたかどうかを確認します。CodeCommit コンソール (<https://console.aws.amazon.com/codecommit>) を開きます (まだ開いていない場合)。
11. 右上のナビゲーションバーで、リポジトリを作成した AWS リージョン を選択します (例: 米国東部 (オハイオ))。

12. [ダッシュボード] ページで、[MyDemoCloud9Repo] を選択します。3 つのファイルが表示されます。

CodeCommit リポジトリの試用を続行するには、AWS CodeCommit ユーザーガイドの「[リポジトリの内容を参照する](#)」を参照してください。

Git を初めて使用する場合で CodeCommit リポジトリに手を加えたくない場合は、[\[Git を試す\]](#) ウェブサイトのサンプル Git リポジトリを試してください。

ステップ 6: クリーンアップする

このサンプルの使用が完了した AWS アカウント 後に に継続的に課金されないようにするには、CodeCommit リポジトリを削除します。手順については、AWS CodeCommit 「ユーザーガイド」の [AWS CodeCommit 「リポジトリの削除」](#) を参照してください。

必ず環境も削除してください。手順については、「[環境の削除](#)」を参照してください。

の Amazon DynamoDB チュートリアル AWS Cloud9

このチュートリアルでは、Amazon DynamoDB と連携する AWS Cloud9 開発環境を設定できます。

DynamoDB は、フルマネージド型の NoSQL データベースサービスです。DynamoDB を使用して、任意の量のデータを保存および取得できるデータベーステーブルを作成し、任意のレベルのリクエストトラフィックを処理できます。DynamoDB によって自動的に、そのテーブルのデータとトラフィックが多数のサーバーに分散されます。サーバーの数は、指定のリクエスト容量と保存されているデータを処理するのに十分であるように選択されます。このような分散処理の間も、パフォーマンスは一定で、高速です。詳細については、AWS ウェブサイトの「[Amazon DynamoDB](#)」を参照してください。

このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。これには、Amazon EC2 や DynamoDB などのサービスで発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon DynamoDB 料金表](#)」を参照してください。

その他の AWS データベースサービスの詳細については、AWS ウェブサイトの[Amazon Relational Database Service \(RDS\)](#)、[Amazon ElastiCache](#)、および「[Amazon Redshift](#)」を参照してください。AWS ウェブサイトで[AWS Database Migration Service](#)も参照してください。

- [前提条件](#)
- [ステップ 1: AWS CLI、AWS CloudShell、またはその両方を環境にインストールして設定する](#)

- [ステップ 2: テーブルを作成する](#)
- [ステップ 3: テーブルに 1 つの項目を追加する](#)
- [ステップ 4: テーブルに複数の項目を追加する](#)
- [ステップ 5: グローバルセカンダリインデックスを作成する](#)
- [ステップ 6: テーブルから項目を取得する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 環境に AWS CLI、AWS CloudShell、またはその両方をインストールして設定する

このステップでは、IDE AWS Cloud9 を使用して AWS CLI、またはその両方を環境にインストールして設定し AWS CloudShell、コマンドを実行して DynamoDB とやり取りできるようにします。次に、AWS CLI を使用してベーシックな DynamoDB コマンドを実行し、インストールと設定をテストします。

1. AWS CLI または の認証情報管理をセットアップ AWS CloudShell し AWS CLI、AWS CloudShell、またはその両方を環境にインストールするには、「」および[AWS CLI AWS CloudShell 「サンプル」](#) のステップ 1 と 2 に従って、このトピックに戻ります。環境に AWS CLI、AWS CloudShell またはその両方を既にインストールして設定している場合は、再度実行する必要はありません。
2. 、aws-shell、またはその両方のインストールと設定をテストするには AWS CLI、環境のターミナルセッションから DynamoDB **list-tables** コマンドを実行して、既存の DynamoDB テーブル

がある場合は一覧表示します。新しいターミナルセッションを開始するには、メニューバーで、[Windows (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。

```
aws dynamodb list-tables # For the AWS CLI.  
dynamodb list-tables     # For the aws-shell.
```

Note

このサンプルでは、aws-shell を使用している場合に aws で始まる各コマンドの aws を省略します。aws-shell を開始するには、**aws-shell** コマンドを実行します。aws-shell の使用を停止するには、**.exit** コマンドまたは **.quit** コマンドを実行します。

このコマンドが正常に実行されると、既存の DynamoDB テーブル (ある場合) のリストが TableNames 配列として出力されます。まだ DynamoDB テーブルがない場合、TableNames 配列は空になります。

```
{  
  "TableNames": []  
}
```

DynamoDB テーブルがある場合は、TableNames 配列にテーブル名のリストが表示されます。

ステップ 2: テーブルを作成する

このステップでは、DynamoDB でテーブルを作成してテーブルの名前、レイアウト、シンププライマリキー、データスループット設定を指定します。

Weather というこのサンプルテーブルには、アメリカのいくつかの都市の天気予報に関する情報が含まれています。このテーブルには、以下のタイプの情報 (DynamoDB では個々の情報は属性と呼ばれます) が保持されます。

- 必須の一意の都市 ID (CityID)
- 必須の予報日 (Date)
- 都市名 (City)
- 州名 (State)
- 予測天気 (Conditions)

- 予測温度 (Temperatures)
 - 予測最高気温、華氏 (HighF)
 - 予測最低気温、華氏 (LowF)

テーブルを作成するには、IDE AWS Cloud9 のターミナルセッションで DynamoDB **create-table** コマンドを実行します。

```
aws dynamodb create-table \  
--table-name Weather \  
--attribute-definitions \  
    AttributeName=CityID,AttributeType=N AttributeName=Date,AttributeType=S \  
--key-schema \  
    AttributeName=CityID,KeyType=HASH AttributeName=Date,KeyType=RANGE \  
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

このコマンドで:

- `--table-name` はテーブル名を表します (このサンプルでは Weather)。テーブル名は、アカウント AWS 内の各 AWS リージョン内で一意である必要があります。
- `--attribute-definitions` はテーブル項目を一意に識別するために使用する属性を表します。このテーブルの各項目は、ISO-8601 形式の文字列として表される数字の ID 属性と Date 属性の組み合わせで一意に識別されます。
- `--key-schema` はテーブルのキースキーマを表します。このテーブルには CityID と Date の複合プライマリキーがあります。つまり、テーブルの各項目は CityID 属性値と Date 属性値を持つ必要がありますが、テーブル内で CityID 属性値と Date 属性値が両方とも同じ 2 つの項目が存在することはできません。
- `--provisioned-throughput` はテーブルの読み取り/書き込み容量を表します。DynamoDB では、サイズが 4 KB までの項目について 1 秒あたり最大 5 つの強力な整合性のある読み込み、またはサイズが 4 KB までの項目について 1 秒あたり最大 5 つの結果整合性のある読み込みが許可されます。また、DynamoDB ではサイズが 1 KB までの項目について 1 秒あたり最大 5 回の書き込みが許可されます。

 Note

プロビジョニングされたスループットを高く設定すると、AWS アカウントに追加料金が発生する可能性があります。

この件および他の DynamoDB のコマンドの詳細については、AWS CLI コマンドリファレンスの「[dynamodb](#)」を参照してください。

このコマンドが成功した場合、作成する新しいテーブルに関する概要情報が表示されます。テーブルが正常に作成されたことを確認するには、テーブルの名前 (--table-name) を指定して DynamoDB **describe-table** コマンドを実行します。

```
aws dynamodb describe-table --table-name Weather
```

テーブルが正常に作成されると、TableStatus の値が CREATING から ACTIVE に変わります。テーブルが正常に作成されてからこのステップの先に進んでください。

ステップ 3: テーブルに 1 つの項目を追加する

このステップでは、作成したテーブルに項目を追加します。

1. weather-item.json という名前のファイルを作成し、次の内容を記述します。新規ファイルを作成するには、メニューバーで [File (ファイル)]、[New File (新規ファイル)] の順に選択します。ファイルを保存するには、[File (ファイル)]、[Save (保存)] の順に選択します。

```
{
  "CityID": { "N": "1" },
  "Date": { "S": "2017-04-12" },
  "City": { "S": "Seattle" },
  "State": { "S": "WA" },
  "Conditions": { "S": "Rain" },
  "Temperatures": { "M": {
    "HighF": { "N": "59" },
    "LowF": { "N": "46" }
  }
}
```

このコードでは、N は数値である属性値です。S は文字列の属性値です。M はマップ属性であり、一連の属性と値のペアです。項目を操作する場合は常に属性のデータ型を指定する必要があります。その他の使用可能な属性のデータ型については、[データ型](#)の Amazon DynamoDB デベロッパーガイドのデータ型を参照してください。

2. テーブル名 (--table-name) と JSON 形式の項目のパス (--item) を指定して DynamoDB **put-item** コマンドを実行します。

```
aws dynamodb put-item \  
--table-name Weather \  
--item file://weather-item.json
```

コマンドが成功すると、エラーなしで実行され、確認メッセージは表示されません。

3. テーブルの現在のコンテンツを確認するには、テーブルの名前 (--table-name) を指定して DynamoDB **scan** コマンドを実行します。

```
aws dynamodb scan --table-name Weather
```

コマンドが成功した場合は、テーブルおよび追加した項目に関する概要情報が表示されます。

ステップ 4: テーブルに複数の項目を追加する

このステップでは、Weather テーブルに複数の項目を追加します。

1. more-weather-items.json という名前のファイルを作成し、次の内容を記述します。

```
{  
  "Weather": [  
    {  
      "PutRequest": {  
        "Item": {  
          "CityID": { "N": "1" },  
          "Date": { "S": "2017-04-13" },  
          "City": { "S": "Seattle" },  
          "State": { "S": "WA" },  
          "Conditions": { "S": "Rain" },  
          "Temperatures": { "M": {  
            "HighF": { "N": "52" },  
            "LowF": { "N": "43" }  
          }  
        }  
      }  
    }  
  ],  
}
```

```

    "PutRequest": {
      "Item": {
        "CityID": { "N": "1" },
        "Date": { "S": "2017-04-14" },
        "City": { "S": "Seattle" },
        "State": { "S": "WA" },
        "Conditions": { "S": "Rain" },
        "Temperatures": { "M": {
          "HighF": { "N": "49" },
          "LowF": { "N": "43" }
        }
      }
    }
  },
  {
    "PutRequest": {
      "Item": {
        "CityID": { "N": "2" },
        "Date": { "S": "2017-04-12" },
        "City": { "S": "Portland" },
        "State": { "S": "OR" },
        "Conditions": { "S": "Thunderstorms" },
        "Temperatures": { "M": {
          "HighF": { "N": "59" },
          "LowF": { "N": "43" }
        }
      }
    }
  },
  {
    "PutRequest": {
      "Item": {
        "CityID": { "N": "2" },
        "Date": { "S": "2017-04-13" },
        "City": { "S": "Portland" },
        "State": { "S": "OR" },
        "Conditions": { "S": "Rain" },
        "Temperatures": { "M": {
          "HighF": { "N": "51" },
          "LowF": { "N": "41" }
        }
      }
    }
  }
}

```

```

    }
  },
  {
    "PutRequest": {
      "Item": {
        "CityID": { "N": "2" },
        "Date": { "S": "2017-04-14" },
        "City": { "S": "Portland" },
        "State": { "S": "OR" },
        "Conditions": { "S": "Rain Showers" },
        "Temperatures": { "M": {
          "HighF": { "N": "49" },
          "LowF": { "N": "39" }
        }
      }
    }
  },
  {
    "PutRequest": {
      "Item": {
        "CityID": { "N": "3" },
        "Date": { "S": "2017-04-12" },
        "City": { "S": "Portland" },
        "State": { "S": "ME" },
        "Conditions": { "S": "Rain" },
        "Temperatures": { "M": {
          "HighF": { "N": "59" },
          "LowF": { "N": "40" }
        }
      }
    }
  },
  {
    "PutRequest": {
      "Item": {
        "CityID": { "N": "3" },
        "Date": { "S": "2017-04-13" },
        "City": { "S": "Portland" },
        "State": { "S": "ME" },
        "Conditions": { "S": "Partly Sunny" },
        "Temperatures": { "M": {

```



```

        "HighF": { "N": "54" },
        "LowF": { "N": "37" }
    }
}
},
{
    "PutRequest": {
        "Item": {
            "CityID": { "N": "3" },
            "Date": { "S": "2017-04-14" },
            "City": { "S": "Portland" },
            "State": { "S": "ME" },
            "Conditions": { "S": "Mostly Sunny" },
            "Temperatures": { "M": {
                "HighF": { "N": "53" },
                "LowF": { "N": "37" }
            }
        }
    }
}
}
]
}

```

このコードでは、前のステップで定義された単一の項目と同様に、8 つの Item オブジェクトがテーブルに追加する 8 つの項目を定義します。ただし、次のステップで **DynamoDB batch-write-item** コマンドを実行する際に、JSON 形式のオブジェクトを指定する必要があります。各 Item オブジェクトが PutRequest オブジェクトに含められます。次に、これらの PutRequest オブジェクトを、テーブルと同じ名前の親配列に含める必要があります。

2. 追加する JSON 形式の項目のパス (`--request-items`) を指定して DynamoDB **batch-write-item** コマンドを実行します。

```

aws dynamodb batch-write-item \
--request-items file://more-weather-items.json

```

コマンドが成功すると、次のメッセージが表示され、項目が正常に追加されたことを確認できます。

```
{
```

```
"UnprocessedItems": {}  
}
```

3. テーブルの現在のコンテンツを確認するには、DynamoDB **scan** コマンドをもう一度実行します。

```
aws dynamodb scan --table-name Weather
```

コマンドが成功した場合は、9 件の項目が表示されます。

ステップ 5: グローバルセカンダリインデックスを作成する

DynamoDB **scan** コマンドを実行して項目に関する情報を取得するには、特にテーブルのサイズが大きくなった場合や、取得する情報のタイプが複雑な場合は、時間がかかる場合があります。1 つ以上のセカンダリインデックスを作成して、処理速度を上げ情報を簡単に取得できるようにします。このステップでは、これを行うために DynamoDB がサポートしている 2 種類のセカンダリインデックスについて学習します。これらは、ローカルセカンダリインデックスおよびグローバルセカンダリインデックスと呼ばれます。ここではグローバルセカンダリインデックスを作成します。

これらのセカンダリインデックスの種類を理解するには、最初にテーブルの項目を一意に識別するプライマリキーの詳細について理解する必要があります。DynamoDB では、シンプルプライマリキーまたは複合プライマリキーがサポートされています。シンプルプライマリキーには単一の属性があり、その属性値はテーブルの項目ごとに一意である必要があります。この属性は、パーティションキー (またはハッシュ属性) と呼ばれ、DynamoDB はこれを使用して項目をパーティション化し、アクセスを高速化します。また、テーブルで 2 つの属性を組み合わせた複合プライマリキーを持つこともできます。最初の属性はパーティションキーであり、2 番目の属性はソートキー (範囲属性ともいいます) です。複合プライマリキーがあるテーブルの場合、2 つの項目が同じパーティションキー値を持つことはできますが、ソートキー値も同じにはできません。Weather テーブルには複合プライマリキーがあります。

ローカルセカンダリインデックスは、テーブル自体と同じパーティションキーを持ちますが、このインデックスタイプでは異なるソートキーを持つことができます。グローバルセカンダリインデックスは、テーブル自体とはどちらも異なるパーティションキーとソートキーを持つことができます。

たとえば、Weather 項目にアクセスするプライマリキーに CityID を既に使用しているとします。Weather 項目に State を使用してアクセスするには、パーティションキー CityID (テーブル自体と同じである必要があります) とソートキー State を持つローカルセカンダリインデックスを

作成できます。Weather 項目に City を使用してアクセスするには、パーティションキー City とソートキー Date を持つグローバルセカンダリインデックスを作成できます。

ローカルセカンダリインデックスは、テーブルを作成するときにのみ作成できます。Weather テーブルが既に存在するため、そこにローカルセカンダリインデックスを追加することはできません。ただし、グローバルセカンダリインデックスを追加することはできます。1 つ追加してみましょう。

 Note

セカンダリインデックスを作成すると、AWS アカウントに追加料金が発生する可能性があります。

1. weather-global-index.json という名前のファイルを作成し、次の内容を記述します。

```
[
  {
    "Create": {
      "IndexName": "weather-global-index",
      "KeySchema": [
        {
          "AttributeName": "City",
          "KeyType": "HASH"
        },
        {
          "AttributeName": "Date",
          "KeyType": "RANGE"
        }
      ],
      "Projection": {
        "ProjectionType": "INCLUDE",
        "NonKeyAttributes": [
          "State",
          "Conditions",
          "Temperatures"
        ]
      },
      "ProvisionedThroughput": {
        "ReadCapacityUnits": 5,
        "WriteCapacityUnits": 5
      }
    }
  }
]
```

```
}  
]
```

このコードでは:

- グローバルセカンダリインデックスの名前は `weather-global-index` です。
- `City` 属性はパーティションキー (ハッシュ属性)、`Date` 属性はソートキー (範囲属性) です。
- `Projection` は、このインデックスを使用したテーブル検索に一致する各項目について、デフォルトで (ハッシュ属性と範囲属性に加えて) 取得する属性を定義します。このサンプルでは、`State`、`Conditions`、`HighF` (Temperatures の一部)、および `LowF` (Temperatures の一部) 属性 (および `City` 属性と `Date` 属性) は、一致する項目ごとに取得されます。
- テーブルと同様に、グローバルセカンダリインデックスではプロビジョニングされたスループット設定を定義する必要があります。
- `IndexName`、`KeySchema`、`Projection`、および `ProvisionedThroughput` 設定が `Create` オブジェクトに含まれている必要があります。このオブジェクトは、次のステップで `DynamoDB update-table` コマンドを実行して作成するグローバルセカンダリインデックスを定義します。

2. DynamoDB **update-table** コマンドを実行します。

```
aws dynamodb update-table \  
--table-name Weather \  
--attribute-definitions \  
    AttributeName=City,AttributeType=S AttributeName=Date,AttributeType=S \  
--global-secondary-index-updates file://weather-global-index.json
```

このコマンドで:

- `--table-name` は更新するテーブルの名前です。
- `--attribute-definitions` はインデックスに含める属性です。パーティションキーは常にリストの最初に表示され、ソートキーは常にリストの 2 番目に表示されます。
- `--global-secondary-index-updates` はグローバルセカンダリインデックスを定義するファイルへのパスです。

このコマンドが成功した場合、作成する新しいグローバルセカンダリインデックスに関する概要情報が表示されます。グローバルセカンダリインデックスが正常に作成されたことを確認するには、テーブルの名前 (`--table-name`) を指定して `DynamoDB describe-table` コマンドを実行します。

```
aws dynamodb describe-table --table-name Weather
```

グローバルセカンダリインデックスが正常に作成された場合、TableStatus の値が UPDATING から ACTIVE に変わり、IndexStatus の値が CREATING から ACTIVE に変わります。グローバルセカンダリインデックスが正常に作成されてからこのステップの先に進んでください。これには数分間かかる場合があります。

ステップ 6: テーブルから項目を取得する

テーブルから項目を取得する方法は複数あります。このステップでは、テーブルのプライマリキー、テーブルのその他の属性、およびグローバルセカンダリインデックスをそれぞれ使用して項目を取得します。

項目のプライマリキーの値に基づいてテーブルから単一の項目を取得するには

項目のプライマリキーバリューがわかっている場合は、DynamoDB コマンド **get-item**、**scan**、または **query** を実行して一致する項目を取得できます。各コマンドの主な相違点は次のとおりです。

- **get-item** は、指定されたプライマリキーを持つ項目の属性のセットを返します。
- **scan** は、テーブルまたはセカンダリインデックスの各項目にアクセスして、1 つ以上の項目または項目属性を返します。
- **query** はプライマリキー値に基づいて項目を探します。複合プライマリキー (パーティションキーとソートキー) のあるテーブルまたはセカンダリインデックスをクエリできます。

このサンプルで、これらのコマンドをそれぞれ使用して、CityID 属性の値が 1、Date 属性の値が 2017-04-12 の項目を取得する方法を説明します。

1. DynamoDB **get-item** コマンドを実行するには、テーブルの名前 (--table-name)、プライマリキーバリュー (--key)、および表示する項目の属性値 (--projection-expression) を指定します。Date は DynamoDB で予約されたキーワードであるため、Date 属性値のエイリアス (--expression-attribute-names) も指定する必要があります (State も予約されたキーワードであり、このためのエイリアスは後のステップで指定します)。

```
aws dynamodb get-item \  
--table-name Weather \  
--key {key} \  
--projection-expression {expression}
```

```
--key '{ "CityID": { "N": "1" }, "Date": { "S": "2017-04-12" } }' \  
--projection-expression \  
  "City, #D, Conditions, Temperatures.HighF, Temperatures.LowF" \  
--expression-attribute-names '{ "#D": "Date" }'
```

このコマンドや他のコマンドで、項目の属性をすべて表示するには、`--projection-expression` を含めません。このサンプルでは、`--projection-expression` を含めていないため、`--expression-attribute-names` を含める必要もありません。

```
aws dynamodb get-item \  
--table-name Weather \  
--key '{ "CityID": { "N": "1" }, "Date": { "S": "2017-04-12" } }'
```

2. DynamoDB **scan** コマンドを実行するには、以下を指定します。

- テーブルの名前 (`--table-name`)。
- 実行する検索 (`--filter-expression`)。
- 使用する検索基準 (`--expression-attribute-values`)。
- 一致した項目で表示する属性の種類 (`--select`)。
- 表示する項目の属性値 (`--projection-expression`)。
- 属性に DynamoDB の予約済みキーワードが使用されている場合、その属性のエイリアス (`--expression-attribute-names`)。

```
aws dynamodb scan \  
--table-name Weather \  
--filter-expression "(CityID = :cityID) and (#D = :date)" \  
--expression-attribute-values \  
  '{ ":cityID": { "N": "1" }, ":date": { "S": "2017-04-12" } }' \  
--select SPECIFIC_ATTRIBUTES \  
--projection-expression \  
  "City, #D, Conditions, Temperatures.HighF, Temperatures.LowF" \  
--expression-attribute-names '{ "#D": "Date" }'
```

3. DynamoDB **query** コマンドを実行するには、以下を指定します。

- テーブルの名前 (`--table-name`)。
- 実行する検索 (`--key-condition-expression`)。
- 検索で使用する属性値 (`--expression-attribute-values`)。
- 一致した項目で表示する属性の種類 (`--select`)。

- 表示する項目の属性値 (--projection-expression)。
- 属性に DynamoDB の予約済みキーワードが使用されている場合、その属性のエイリアス (--expression-attribute-names)。

```
aws dynamodb query \  
--table-name Weather \  
--key-condition-expression "(CityID = :cityID) and (#D = :date)" \  
--expression-attribute-values \  
  '{ ":cityID": { "N": "1" }, ":date": { "S": "2017-04-12" } }' \  
--select SPECIFIC_ATTRIBUTES \  
--projection-expression \  
  "City, #D, Conditions, Temperatures.HighF, Temperatures.LowF" \  
--expression-attribute-names '{ "#D": "Date" }'
```

結果を取得するためにスキャンする必要がある項目数が、**scan** コマンドでは 9 項目すべてであるのに対して **query** コマンドでは 1 項目のみであることに注目してください。

項目のプライマリキーの値に基づいてテーブルから複数の項目を取得するには

項目のプライマリキーバリューがわかっている場合は、DynamoDB **batch-get-item** コマンドを実行して一致する項目を取得できます。このサンプルで、CityID 属性の値が 3、Date 属性の値が 2017-04-13 または 2017-04-14 の項目を取得する方法を説明します。

取得する項目を記述するファイルへのパス (--request-items) を指定して、DynamoDB **batch-get-item** コマンドを実行します。

```
aws dynamodb batch-get-item --request-items file://batch-get-item.json
```

このサンプルでは、batch-get-item.json ファイルのコードは、Weather テーブルで、CityID が 3 であり Date が 2017-04-13 または 2017-04-14 である項目を検索するように指定します。検出された各項目について、City、State、Date、および HighF (Temperatures の一部) の属性値が表示されます (存在する場合)。

```
{  
  "Weather" : {  
    "Keys": [  
      {  
        "CityID": { "N": "3" },  
        "Date": { "S": "2017-04-13" }  
      }  
    ]  
  }  
}
```

```

    },
    {
      "CityID": { "N": "3" },
      "Date": { "S": "2017-04-14" }
    }
  ],
  "ProjectionExpression": "City, #S, #D, Temperatures.HighF",
  "ExpressionAttributeNames": { "#S": "State", "#D": "Date" }
}
}

```

一致するすべての項目をテーブルから取得するには

テーブルの属性値について一部でもわかっていることがあれば、DynamoDB **scan** コマンドを実行して一致する項目を取得できます。このサンプルで、Conditions 属性の値に Sunny が含まれ、HighF 属性の値 (Temperatures の一部) が 53 より大きい日を取得する方法を説明します。

DynamoDB **scan** コマンドを実行し、以下を指定します。

- テーブルの名前 (--table-name)。
- 実行する検索 (--filter-expression)。
- 使用する検索基準 (--expression-attribute-values)。
- 一致した項目で表示する属性の種類 (--select)。
- 表示する項目の属性値 (--projection-expression)。
- 属性に DynamoDB の予約済みキーワードが使用されている場合、その属性のエイリアス (--expression-attribute-names)。

```

aws dynamodb scan \
--table-name Weather \
--filter-expression \
  "(contains (Conditions, :sun)) and (Temperatures.HighF > :h)" \
--expression-attribute-values \
  '{ ":sun": { "S" : "Sunny" }, ":h": { "N" : "53" } }' \
--select SPECIFIC_ATTRIBUTES \
--projection-expression "City, #S, #D, Conditions, Temperatures.HighF" \
--expression-attribute-names '{ "#S": "State", "#D": "Date" }'

```


一致するすべての項目をグローバルセカンダリインデックスから取得するには

グローバルセカンダリインデックスを使用して検索するには、**DynamoDBquery** コマンドを使用します。このサンプルで、weather-global-index セカンダリインデックスを使用して、日付が Portland および 2017-04-13 の、2017-04-14 という名前の都市の予測天気を取得する方法を説明します。

DynamoDB **query** コマンドを実行し、以下を指定します。

- テーブルの名前 (--table-name)。
- グローバルセカンダリインデックスの名前 (--index-name)。
- 実行する検索 (--key-condition-expression)。
- 検索で使用する属性値 (--expression-attribute-values)。
- 一致した項目で表示する属性の種類 (--select)。
- 属性に DynamoDB の予約済みキーワードが使用されている場合、その属性のエイリアス (--expression-attribute-names)。

```
aws dynamodb query \  
--table-name Weather \  
--index-name weather-global-index \  
--key-condition-expression "(City = :city) and (#D between :date1 and :date2)" \  
--expression-attribute-values \  
  '{ ":city": { "S" : "Portland" }, ":date1": { "S": "2017-04-13" }, ":date2": { "S":  
  "2017-04-14" } }' \  
--select SPECIFIC_ATTRIBUTES \  
--projection-expression "City, #S, #D, Conditions, Temperatures.HighF" \  
--expression-attribute-names '{ "#S": "State", "#D": "Date" }'
```

ステップ 7: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、テーブルを削除する必要があります。テーブルを削除すると、グローバルセカンダリインデックスも削除されます。環境も削除する必要があります。

テーブルを削除するには、テーブルの名前 () を指定して **DynamoDBdelete-table** --table-name コマンドを実行します。

```
aws dynamodb delete-table --table-name Weather
```

コマンドが成功した場合は、テーブルに関する情報が表示され、そこで TableStatus 値が DELETING になっています。

テーブルが正常に削除されたことを確認するには、DynamoDB **describe-table** コマンドを実行して、テーブルの名前 (--table-name) を指定してます。

```
aws dynamodb describe-table --table-name Weather
```

テーブルが正常に削除された場合は、Requested resource not found というフレーズを含むメッセージが表示されます。

環境を削除するには、「[環境の削除](#)」を参照してください。

AWS CDK の チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud Development Kit (AWS CDK) AWS Cloud9 開発環境で を 使用する方法を示します。AWS CDK は、開発者が AWS インフラストラクチャコンポーネントをコードとしてモデル化するために使用できる一連のソフトウェアツールとライブラリです。

AWS CDK には、多くのタスクをすばやく解決するために使用できる AWS コンストラクトライブラリが含まれています AWS。たとえば、Fleet コンストラクトを使用すると、コードをホストのフリートに完全かつ安全にデプロイできます。独自のコンストラクトを作成してアーキテクチャのさまざまな要素をモデル化し、これらを他のユーザーと共有したり、コミュニティに公開したりできます。詳細については、「[AWS Cloud Development Kit 開発者ガイド](#)」を参照してください。

このチュートリアルに従って、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。これには、Amazon EC2、Amazon SNS、Amazon SQS などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」、「[Amazon SNS 料金表](#)」および「[Amazon SQS 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを実行する](#)
- [ステップ 4: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは、が TypeScript プログラミング言語で記述されたサンプルを実行する AWS CDK ために必要なすべてのツールを環境にインストールします。

1. [Node Version Manager \(nvm\)](#)。後で Node.js をインストールするために使用します。
2. [Node.js](#)。サンプルで必要であり、Node Package Manager、または TypeScript AWS CDK 以降をインストールするために使用する **npm** が含まれています。
3. [TypeScript](#)。このサンプルに必要です。(では、他のいくつかのプログラミング言語 AWS CDK もサポートされています)。
4. [AWS CDK](#)。

ステップ 1.1: Node Version Manager (nvm) をインストールする

1. AWS Cloud9 IDE のターミナルセッションで、最新のセキュリティ更新プログラムとバグ修正がインストールされていることを確認します。これを行うには、**yum update** (Amazon Linux 用)または **apt update** コマンド (Ubuntu Server 用) を実行します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。)

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

2. **nvm** がインストール済みであるかどうかを確認します。これを行うには、**--version** オプションを使用して **nvm** コマンドを実行します。

```
nvm --version
```

成功すると、出力に **nvm** バージョン番号が表示されます。この場合、「[ステップ 1.2: Node.js をインストールする](#)」までスキップできます。

3. **nvm** をダウンロードしてインストールします。これを行うには、インストールスクリプトを実行します。この例では、v0.33.0 がインストールされていますが、**nvm** の最新バージョンは[こちら](#)で確認できます。

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

4. **nvm** の使用を開始します。ターミナルセッションを閉じて再起動するか **nvm** をロードするコマンドを含む ~/.bashrc ファイルを入手してください。

```
. ~/.bashrc
```

ステップ 1.2: Node.js をインストールする

1. Node.js がインストール済みであるかどうかを確認します。インストール済みである場合は、そのバージョンが 16.17.0 以上であることを確認します。このサンプルは Node.js 16.17.0 を使用してテスト済みです。IDE でターミナルセッションがまだ開いている状態でチェックするには、**--version** オプションと共に **node** コマンドを実行します。

```
node --version
```

Node.js がインストール済みである場合は、出力にバージョン番号が表示されます。バージョン番号が v16.17.0 である場合は、「[ステップ 1.3: TypeScript をインストールする](#)」に進みます。

2. **nvm** コマンドを **install** アクションと共に実行して、Node.js 16 をインストールします。

 Note

nvm install node を実行して、Node.js の長期サポート (LTS) バージョンをインストールすることもできます。AWS Cloud9 support は、Node.js の LTS バージョンを追跡します。

```
nvm install v16
```

3. Node.js 16 の使用を開始します。これを行うには、**nvm** コマンドを実行します。次のように **alias** アクション、エイリアスを作成するバージョン番号、このエイリアスで使用するバージョンを指定します。

```
nvm alias default 16
```

 Note

上のコマンドは、Node.js 16 をデフォルトバージョンの Node.js として設定します。または、**alias** アクション (たとえば、**nvm use 16.17.0**) の代わりに **use** アクションと共に **nvm** コマンドとを使用できます。ただし、**use** アクションを使用すると、このバージョンの Node.js の実行先は現在実行中のターミナルセッションに限られます。

4. Node.js 16 を使用していることを確認するには、**node --version** コマンドを再度実行します。正しいバージョンをインストールしている場合は、出力にバージョン v16 と表示されます。

ステップ 1.3: TypeScript をインストールする

1. TypeScript がインストール済みであるかどうかを確認します。これを行うには、IDE でターミナルセッションがまだ開いている状態で、**--version** オプションを使ってコマンドライン TypeScript コンパイラーを実行します。

```
tsc --version
```

TypeScript がインストール済みである場合は、出力に TypeScript のバージョン番号が表示されます。TypeScript がインストール済みである場合は、[ステップ 1.4: をインストールする AWS CDK](#)に進みます。

2. TypeScript をインストールします。これを行うには、**install** アクション、**-g** オプション、および TypeScript パッケージの名前を指定して **npm** コマンドを実行します。これにより、TypeScript がグローバルパッケージとして環境にインストールされます。

```
npm install -g typescript
```

3. TypeScript がインストールされていることを確認します。これを行うには、**--version** オプションを使用してコマンドライン TypeScript コンパイラーを実行します。

```
tsc --version
```

TypeScript がインストール済みである場合は、出力に TypeScript のバージョン番号が表示されます。

ステップ 1.4: をインストールする AWS CDK

1. が既に AWS CDK インストールされているかどうかを確認します。これを行うには、IDE でターミナルセッションがまだ開いている状態で、**--version** オプションを使って、**cdk** コマンドを実行します。

```
cdk --version
```

AWS CDK がインストールされている場合、出力には AWS CDK バージョンとビルド番号が含まれます。[ステップ 2: コードを追加する](#)に進んでください。

2. をインストールするには、**npm** コマンドを **install** アクション、インストールする AWS CDK パッケージの名前、環境にパッケージをグローバルにインストールする **-g** オプションとともに AWS CDK 実行します。

```
npm install -g aws-cdk
```

3. がインストールされ、正しく参照 AWS CDK されていることを確認します。これを行うには、**--version** オプションを使用して **cdk** コマンドを実行します。

```
cdk --version
```

成功すると、AWS CDK バージョンとビルド番号が表示されます。

ステップ 2: コードを追加する

このステップでは、が AWS CloudFormation スタックを AWS CDK プログラムでデプロイするために必要なすべてのソースコードを含むサンプル TypeScript プロジェクトを作成します。このスタックは、AWS アカウントに Amazon SNS トピックと Amazon SQS キューを作成し、キューをトピックにサブスクライブします。

1. IDE でターミナルセッションがまだ開いている状態で、プロジェクトのソースコードを保存するディレクトリ (例: ~/environment/hello-cdk ディレクトリ) を環境に作成します。次に、このディレクトリに切り替えます。

```
rm -rf ~/environment/hello-cdk # Remove this directory if it already exists.
mkdir ~/environment/hello-cdk  # Create the directory.
cd ~/environment/hello-cdk     # Switch to the directory.
```

2. このディレクトリを、AWS CDK の TypeScript 言語プロジェクトとして設定します。これを行うには、**init** アクション、**sample-app** テンプレート、**--language** オプション、およびプログラミング言語の名前を指定して **cdk** コマンドを実行します。

```
cdk init sample-app --language typescript
```

これにより、ディレクトリ内に以下のファイルとサブディレクトリが作成されます。

- 隠しサブディレクトリ **.git** と隠しファイル **.gitignore**。これにより、プロジェクトは Git などのソース管理ツールと互換性を持ちます。
- **lib** サブディレクトリ。hello-cdk-stack.ts ファイルが含まれます。このファイルには、AWS CDK スタックのコードが含まれています。このコードは、この手順の次のステップで説明します。
- **bin** サブディレクトリ。hello-cdk.ts ファイルが含まれます。このファイルには、AWS CDK アプリのエントリポイントが含まれています。
- **node_modules** サブディレクトリ。必要に応じてアプリケーションとスタックで利用できるサポート用のコードパッケージが含まれます。

- 隠しファイル `.npmignore`。コードのビルド時に **npm** で使用しない種類のサブディレクトリやファイルのリストが含まれます。
 - `cdk.json` ファイル。このファイルの情報を使用すると **cdk** コマンドを実行しやすくなります。
 - `package-lock.json` ファイル。このファイルの情報をを使用して、**npm** はビルドや実行に伴うエラーを減らすことができます。
 - `package.json` ファイル。このファイルの情報を使用すると、**npm** コマンドの実行が容易になり、ビルドと実行に伴うエラーも減る可能性があります。
 - **npm** および `npm run` で実行できる便利なコマンドを一覧表示する `README.md` ファイル AWS CDK。
 - `tsconfig.json` ファイル。このファイルの情報を使用すると、**tsc** コマンドの実行が容易になり、ビルドと実行に伴うエラーも減る可能性があります。
3. [Environment (環境)] ウィンドウで、`lib/hello-cdk-stack.ts` ファイルを開き、このファイル内で次のコードを参照します。

```
import sns = require('@aws-cdk/aws-sns');
import sqs = require('@aws-cdk/aws-sqs');
import cdk = require('@aws-cdk/cdk');

export class HelloCdkStack extends cdk.Stack {
  constructor(parent: cdk.App, name: string, props?: cdk.StackProps) {
    super(parent, name, props);

    const queue = new sqs.Queue(this, 'HelloCdkQueue', {
      visibilityTimeoutSec: 300
    });

    const topic = new sns.Topic(this, 'HelloCdkTopic');

    topic.subscribeQueue(queue);
  }
}
```

- `Stack`、`App`、`StackProps`、`Queue`および `Topic` クラスは、CloudFormation スタックとそのプロパティ、実行可能プログラム、Amazon SQS キュー、および Amazon SNS トピックをそれぞれ表します。

- HelloCdkStack クラスは、このアプリケーションの CloudFormation スタックを表します。このスタックには、このアプリケーションの新しい Amazon SQS キューおよび Amazon SNS トピックが含まれます。
4. [Environment (環境)] ウィンドウで、bin/hello-cdk.ts ファイルを開き、このファイル内で次のコードを参照します。

```
#!/usr/bin/env node
import cdk = require('@aws-cdk/cdk');
import { HelloCdkStack } from '../lib/hello-cdk-stack';

const app = new cdk.App();
new HelloCdkStack(app, 'HelloCdkStack');
app.run();
```

このコードは、lib/hello-cdk-stack.ts の HelloCdkStack クラスをロードし、インスタンス化して、実行します。

5. **npm** を使用して TypeScript コンパイラを実行してコーディングエラーを確認し、を有効に AWS CDK してプロジェクトの bin/hello-cdk.js ファイルを実行します。これを行うには、プロジェクトのルートディレクトリから、**npm** コマンドを **run** アクションで実行します。**build** コマンドの値は package.json ファイルで次のように指定します。

```
npm run build
```

上のコマンドは、TypeScript コンパイラを実行します。このコンパイラは、bin/hello-cdk.d.ts ファイルと lib/hello-cdk-stack.d.ts ファイルのサポートを追加します。また、このコンパイラは、hello-cdk.js ファイルおよび hello-cdk-stack.js ファイルを、hello-cdk.ts ファイルおよび hello-cdk-stack.ts ファイルとしてトランスパイルします。

ステップ 3: コードを実行する

このステップでは、bin/hello-cdk.js ファイル内のコードに基づいて CloudFormation スタックテンプレートを作成する AWS CDK ように に指示します。次に、スタックをデプロイ AWS CDK するように に指示します。これにより、Amazon SNS トピックと Amazon SQS キューが作成され、キューがトピックにサブスクライブされます。次に、トピックからキューにメッセージを送信して、トピックとキューが正常にデプロイされていることを確認します。

1. で CloudFormation スタックテンプレート AWS CDK を作成します。これを行うには、IDE でターミナルセッションがまだ開いている状態で、プロジェクトのルートディレクトリから、**synth** アクションとスタック名を使って、**cdk** コマンドを実行します。

```
cdk synth HelloCdkStack
```

成功すると、出力に CloudFormation スタックテンプレートResourcesのセクションが表示されます。

2. 特定の AWS アカウントと AWS リージョンの組み合わせの環境に AWS CDK アプリケーションを初めてデプロイする場合は、ブートストラップスタックをインストールする必要があります。このスタックには、さまざまなオペレーションを完了 AWS CDK するために必要なさまざまなリソースが含まれています。たとえば、このスタックには、デプロイプロセス中に テンプレートとアセットを保存 AWS CDK するために使用する Amazon S3 バケットが含まれています。ブートストラップスタックをインストールするには、**cdk** コマンドを **bootstrap** アクションと共に実行します。

```
cdk bootstrap
```

Note

オプションを指定cdk bootstrapせずに を実行すると、デフォルトの AWS アカウントと AWS リージョンが使用されます。プロファイルとアカウント/リージョンの組み合わせを指定して、特定の環境をブートストラップすることもできます。例えば、次のようになります。

```
cdk bootstrap --profile test 123456789012/us-east-1
```

3. CloudFormation スタックテンプレート AWS CDK を実行してスタックをデプロイします。これを行うには、プロジェクトのルートディレクトリから、**cdk** コマンドに **deploy** アクションとスタック名を指定して実行します。

```
cdk deploy HelloCdkStack
```

成功すると、エラーなしでデプロイされた HelloCdkStack スタックが出力に表示されます。

Note

出力に、スタックが環境を定義しておらず、AWS 認証情報を標準の場所から取得できなかったか、リージョンが設定されていないというメッセージが表示された場合は、AWS 認証情報が IDE で正しく設定されていることを確認し、**cdk deploy** コマンドを再度実行します。詳細については、「[の環境 AWS のサービス からの呼び出し AWS Cloud9](#)」を参照してください。

4. Amazon SNS トピックと Amazon SQS キューが正常にデプロイされたことを確認するには、トピックにメッセージを送信し、キューでメッセージが受信されたことをチェックします。これを行うには、AWS Command Line Interface (AWS CLI) や などのツールを使用できます AWS CloudShell。これらのツールの詳細については、[AWS CLI および の aws-shell チュートリアル AWS Cloud9](#) を参照してください。

たとえば、IDE でターミナルセッションを開いたままトピックにメッセージを送信するには、AWS CLI を使用して Amazon SNS **publish** コマンドを実行し、メッセージの件名と本文、トピックの AWS リージョン、トピックの Amazon リソースネーム (ARN) を指定します。

```
aws sns publish --subject "Hello from the AWS CDK" --message "This is a message from the AWS CDK." --topic-arn arn:aws:sns:us-east-2:123456789012:HelloCdkStack-HelloCdkTopic1A234567-8BCD9EFGHIJ0K
```

前述のコマンドで、をトピック CloudFormation に割り当てる ARN `arn:aws:sns:us-east-2:123456789012:HelloCdkStack-HelloCdkTopic1A234567-8BCD9EFGHIJ0K` に置き換えます。この ID を取得するには、Amazon SNS **list-topics** コマンドを実行できます。

```
aws sns list-topics --output table --query 'Topics[*].TopicArn'
```

成功すると、**publish** コマンドの出力に、発行されたメッセージの MessageId 値が表示されます。

キューでメッセージが受信されたことをチェックするには、Amazon SQS **receive-message** コマンドを実行して、キューの URL を提供します。

```
aws sqs receive-message --queue-url https://queue.amazonaws.com/123456789012/HelloCdkStack-HelloCdkQueue1A234567-8BCD9EFGHIJ0K
```

前述のコマンドで、`QueueUrl` をキュー CloudFormation に割り当てる ARN `https://queue.amazonaws.com/123456789012/HelloCdkStack-HelloCdkQueue1A234567-8BCD9EFGHIJ0K`に置き換えます。この URL を取得するため、Amazon SQS **list-queues** コマンドを実行できます。

```
aws sqs list-queues --output table --query 'QueueUrls[*]'
```

成功すると、**receive-message** コマンドの出力に、受信されたメッセージの情報が表示されます。

ステップ 4: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、CloudFormation スタックを削除する必要があります。これにより、Amazon SNS トピックと Amazon SQS キューが削除されます。環境も削除する必要があります。

ステップ 4.1: スタックを削除する

IDE でターミナルセッションがまだ開いている状態で、プロジェクトのルートディレクトリから、**destroy** アクションとスタック名で **cdk** コマンドを実行します。

```
cdk destroy HelloCdkStack
```

スタックの削除を確認するメッセージが表示されたら、「y」と入力し、Enter キーを押します。

成功すると、エラーなしで HelloCdkStack スタックが削除されたことが出力に表示されます。

ステップ 4.2: 環境を削除する

環境を削除するには、[での環境の削除 AWS Cloud9](#) を参照してください。

の LAMP チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境内で LAMP (Linux、Apache HTTP Server、MySQL、および PHP) をセットアップして実行できます。

このチュートリアルに従い、このサンプルを作成すると、に料金が発生する可能性があります AWS アカウント。これには、Amazon Elastic Compute Cloud (Amazon EC2) AWS のサービス などの の 料金が含まれます。詳細については、「[Amazon EC2 の料金](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: ツールのインストール](#)
- [ステップ 2: MySQL を設定する](#)
- [ステップ 3: ウェブサイトの設定](#)
- [ステップ 4: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に開いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: ツールのインストール

このステップでは、以下のツールをインストールします。

- Apache HTTP Server: ウェブサーバーホスト。
- PHP: 特にウェブ開発に適しており、HTML に埋め込むことができるスクリプト言語。
- MySQL: データベース管理システム。

その後、このステップを終了するには、Apache HTTP サーバーを起動後、MySQL を起動します。

1. インスタンスに最新のセキュリティ更新プログラムとバグ修正が適用されていることを確認します。これを行うには、IDE AWS Cloud9 のターミナルセッションで、**yum update** for (Amazon Linux) または **apt update** for (Ubuntu Server) コマンドを実行します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。)

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu サーバーの場合:

```
sudo apt -y update
```

2. Apache HTTP Server が既にインストールされているかどうかを確認します。これを行うには、**httpd -v** (Amazon Linux) または **apache2 -v** (Ubuntu Server) コマンドを実行します。

成功すると、Apache HTTP Server のバージョン番号が出力に表示されます。

エラーが発生する場合は、**install** コマンドを実行して Apache HTTP Server をインストールします。

Amazon Linux の場合:

```
sudo yum install -y httpd24
```

Ubuntu サーバーの場合:

```
sudo apt install -y apache2
```

3. **php -v** コマンドを実行して PHP が既にインストールされているかどうかを確認します。

成功すると、出力に PHP のバージョン番号が表示されます。

エラーが発生する場合は、**install** コマンドを実行して PHP をインストールします。

Amazon Linux の場合:

```
sudo yum install -y php56
```

Ubuntu サーバーの場合:

```
sudo apt install -y php libapache2-mod-php php-xml
```

4. **mysql --version** コマンドを実行して MySQL が既にインストールされているかどうかを確認します。

成功すると、出力に MySQL のバージョン番号が表示されます。

エラーが発生する場合は、**install** コマンドを実行して MySQL をインストールします。

Amazon Linux の場合:

```
sudo yum install -y mysql-server
```

Ubuntu サーバーの場合:

```
sudo apt install -y mysql-server
```

5. Apache HTTP Server、PHP、および MySQL をインストールしたら、Apache HTTP Server を起動後、次のコマンドを実行して、起動したことを確認します。

Amazon Linux の場合 (このコマンドを 2 回実行する必要がある場合があります):

```
sudo service httpd start && sudo service httpd status
```

Ubuntu Server の場合 (コマンドプロンプトに戻るには、q を押します):

```
sudo service apache2 start && sudo service apache2 status
```

6. MySQL を起動後、次のコマンドを実行して、起動したことを確認します。

Amazon Linux の場合:

```
sudo service mysqld start && sudo service mysqld status
```

Ubuntu Server の場合 (コマンドプロンプトに戻るには、q を押します):

```
sudo service mysql start && sudo service mysql status
```

ステップ 2: MySQL を設定する

このステップでは、MySQL セキュリティのベストプラクティスに従うようにMySQL を設定します。これらのセキュリティのベストプラクティスには、ルートアカウントのパスワードの設定や、ローカルホストの外部からアクセス可能なルートアカウントの削除が含まれます。他に注意すべきベストプラクティスとしては、匿名ユーザーの削除、テスト用データベースの削除、test_ で始まる名前のデータベースに誰でもアクセスできる権限の削除があります。

その後、MySQL コマンドラインクライアントの起動と終了を行い、このステップを終了します。

1. MySQL をインストールするための MySQL セキュリティのベストプラクティスを実装するには、AWS Cloud9 IDE のターミナルセッションで次のコマンドを実行します。

```
sudo mysql_secure_installation
```

2. プロンプトが表示されたら、指定されたとおりに次の質問に回答します。

Amazon Linux の場合:

1. root の現在のパスワードを入力 (ない場合は Enter キー) – Enter を押してください (パスワードがない場合)。
2. root のパスワードの設定 – Y と入力し、Enter を押してください。
3. 新しいパスワード – パスワードを入力し、Enter を押してください。
4. 新しいパスワードの再入力 – パスワードを再入力し、Enter を押してください。(後に使用できるように、パスワードは安全な場所に保存してください。)
5. 匿名ユーザーの削除 – Y と入力し、Enter を押してください。
6. root のリモートログインを禁止 – Y と入力し、Enter を押してください。
7. テストデータベースを削除し、アクセスする – Y と入力し、Enter を押してください。
8. 権限テーブルを再ロード – Y と入力し、Enter を押してください。

Ubuntu サーバーの場合:

1. パスワード検証プラグインのセットアップ – y と入力し、Enter を押します。
2. パスワード検証ポリシーには 3 つのレベルがあります – 0、1、または 2 と入力し、Enter を押します。
3. 新しいパスワード – パスワードを入力し、Enter を押します。

4. 新しいパスワードの再入力 – パスワードを再入力し、Enter を押します。後に使用できるように、パスワードは安全な場所に保存してください。
 5. 提供されたパスワードを続行しますか? – y と入力し、Enter を押します。
 6. 匿名ユーザーの削除 – y と入力し、Enter を押します。
 7. ルートのリモートログインを禁止 – y と入力し、Enter を押します。
 8. テストデータベースを削除し、アクセスする – y と入力し、Enter を押します。
 9. 権限テーブルを再ロード – y と入力し、Enter を押します。
3. MySQL を直接操作するには、次のコマンドを実行して、ルートユーザーとして MySQL コマンドラインクライアントをスタートします。プロンプトが表示されたら、先ほど設定した root ユーザーのパスワードを入力し、Enter を押します。MySQL コマンドラインクライアントを使用している場合、プロンプトは mysql> に変更されます。

```
sudo mysql -uroot -p
```

4. MySQL コマンドラインクライアントを終了するには、次のコマンドを実行します。プロンプトは \$ に戻ります。

```
exit;
```

ステップ 3: ウェブサイトの設定

このステップでは、Apache HTTP Server のデフォルトウェブサイトのルートに、推奨される所有者とアクセス許可を設定します。その後、そのデフォルトウェブサイトのルート内に PHP ベースのウェブページを作成します。

次に、この EC2 環境に関連付けられている Amazon Virtual Private Cloud (Amazon VPC) にある Amazon EC2 のセキュリティグループと Amazon EC2 とネットワークのアクセスコントロールリスト (ACL) を設定して、着信ウェブトラフィックでそのウェブページの表示を有効にします。各 EC2 環境は、Amazon EC2 のセキュリティグループと Amazon VPC のネットワーク ACL の両方に関連付ける必要があります。ただし、AWS アカウント のデフォルトネットワーク ACL では環境のすべての送受信トラフィックが許可されていますが、デフォルトのセキュリティグループでは、ポート 22 経由で SSH を使用した着信トラフィックのみが許可されています。詳細については、「[the section called “Amazon VPC の設定”](#)」を参照してください。

その後、AWS Cloud9 IDE の外部からウェブページを正常に表示して、このステップを終了します。

1. Apache HTTP Server のデフォルトウェブサイトのルート (/var/www/html) に、推奨される所有者とアクセス許可を設定します。これを行うには、IDE のターミナルセッションで、次の 6 つのコマンドを一度に 1 AWS Cloud9 つずつ次の順序で実行します。各コマンドの動作を理解するには、各コマンドの後の # の情報をお読みください。

Amazon Linux の場合:

```
sudo groupadd web-content # Create a group named web-content.

sudo usermod -G web-content -a ec2-user # Add the user ec2-user (your default user
for this environment) to the group web-content.

sudo usermod -G web-content -a apache # Add the user apache (Apache HTTP Server) to
the group web-content.

sudo chown -R ec2-user:web-content /var/www/html # Change the owner of /var/www/
html and its files to user ec2-user and group web-content.

sudo find /var/www/html -type f -exec chmod u=rw,g=rx,o=rx {} \; # Change all file
permissions within /var/www/html to user read/write, group read-only, and others
read/execute.

sudo find /var/www/html -type d -exec chmod u=rwx,g=rx,o=rx {} \; # Change /var/
www/html directory permissions to user read/write/execute, group read/execute, and
others read/execute.
```

Ubuntu サーバーの場合:

```
sudo groupadd web-content # Create a group named web-content.

sudo usermod -G web-content -a ubuntu # Add the user ubuntu (your default user for
this environment) to the group web-content.

sudo usermod -G web-content -a www-data # Add the user www-data (Apache HTTP
Server) to the group web-content.

sudo chown -R ubuntu:web-content /var/www/html # Change the owner of /var/www/html
and its files to user ubuntu and group web-content.

sudo find /var/www/html -type f -exec chmod u=rw,g=rx,o=rx {} \; # Change all file
permissions within /var/www/html to user read/write, group read-only, and others
read/execute.
```

```
sudo find /var/www/html -type d -exec chmod u=rwx,g=rx,o=rx {} \; # Change /var/
www/html directory permissions to user read/write/execute, group read/execute, and
others read/execute.
```

2. 次のコマンドを実行して、`index.php` という PHP ベースのウェブページを Apache HTTP Server のデフォルトウェブサイトルートフォルダ (`/var/www/html`) に作成します。

Amazon Linux の場合:

```
sudo touch /var/www/html/index.php && sudo chown -R ec2-user:web-content /var/www/
html/index.php && sudo chmod u=rw,g=rx,o=rx /var/www/html/index.php && sudo printf
'%s\n%s\n%s' '<?php' '  phpinfo();' '?>' >> /var/www/html/index.php
```

Amazon Linux の場合も、上記コマンドによって、ファイルの所有者は `ec2-user` に、ファイルのグループは `web-content` に、ファイルのアクセス許可は、読み取り/書き込み (ユーザー)、読み取り/実行 (グループ、その他) に変更されます。

Ubuntu サーバーの場合:

```
sudo touch /var/www/html/index.php && sudo chown -R ubuntu:web-content /var/www/
html/index.php && sudo chmod u=rw,g=rx,o=rx /var/www/html/index.php && sudo printf
'%s\n%s\n%s' '<?php' '  phpinfo();' '?>' >> /var/www/html/index.php
```

Ubuntu Server の場合も、上記コマンドによって、ファイルの所有者は `ubuntu` に、ファイルのグループは `web-content` に、ファイルのアクセス許可は、読み書き (ユーザー)、読み取り/実行 (グループ、その他) に変更されます。

上記コマンドが正常に実行されると、次の内容で `index.php` ファイルが作成されます。

```
<?php
  phpinfo();
?>
```

3. Amazon VPC のネットワーク ACL と、この EC2 環境に関連付けられている Amazon EC2 のセキュリティグループを設定して、ポート 80 経由の着信ウェブトラフィックで新しいウェブページを表示を有効にします。そのためには、次の 8 個のコマンドを一度に 1 つずつ次の順序で実行します。各コマンドの動作を理解するには、各コマンドの # の後に表示される情報をお読みください。

⚠ Important

次のコマンドを実行すると、この環境のセキュリティグループとネットワーク ACL に関連付けられているすべての EC2 環境と Amazon EC2 インスタンスに対するポート 80 経由の着信ウェブトラフィックを有効にします。これにより、これ以外の EC2 環境および Amazon EC2 インスタンスに対して、ポート 80 経由の着信ウェブトラフィックが予期せずにも有効になる可能性があります。

i Note

次の 2 番目から 4 番目のコマンドでは、セキュリティグループのポート 80 経由の着信ウェブトラフィックが有効になります。ポート 22 経由の着信 SSH トラフィックのみを許可するデフォルトのセキュリティグループがある場合は、最初のコマンドとそれに続く 2 番目から 4 番目のコマンドを実行する必要があります。ただし、カスタムセキュリティグループで、既にポート 80 経由の着信ウェブトラフィックが許可されている場合は、これらのコマンドの実行をスキップすることができます。

次の 5 番目から 8 番目のコマンドでは、ネットワーク ACL のポート 80 経由の着信ウェブトラフィックの許可が有効になります。デフォルトのネットワーク ACL があり、すべてのポート経由ですべての着信トラフィックがすでに許可されている場合は、これらのコマンドの実行をスキップすることができます。ただし、ポート 80 経由の着信ウェブトラフィックを許可しないカスタムネットワーク ACL を保有しているとします。その場合、最初のコマンドを実行してから、5 番目から 8 番目のコマンドを実行します。

```
MY_INSTANCE_ID=$(curl http://169.254.169.254/latest/meta-data/instance-id) # Get
the ID of the instance for the environment, and store it temporarily.

MY_SECURITY_GROUP_ID=$(aws ec2 describe-instances --instance-id $MY_INSTANCE_ID
--query 'Reservations[].Instances[0].SecurityGroups[0].GroupId' --output text)
# Get the ID of the security group associated with the instance, and store it
temporarily.

aws ec2 authorize-security-group-ingress --group-id $MY_SECURITY_GROUP_ID --
protocol tcp --cidr 0.0.0.0/0 --port 80 # Add an inbound rule to the security group
to allow all incoming IPv4-based traffic over port 80.
```

```
aws ec2 authorize-security-group-ingress --group-id $MY_SECURITY_GROUP_ID --ip-
permissions IpProtocol=tcp,Ipv6Ranges='[{CidrIpv6=::/0}]',FromPort=80,ToPort=80 #
Add an inbound rule to the security group to allow all incoming IPv6-based traffic
over port 80.
```

```
MY_SUBNET_ID=$(aws ec2 describe-instances --instance-id $MY_INSTANCE_ID --query
'Reservations[].Instances[0].SubnetId' --output text) # Get the ID of the subnet
associated with the instance, and store it temporarily.
```

```
MY_NETWORK_ACL_ID=$(aws ec2 describe-network-acls --filters
Name=association.subnet-id,Values=$MY_SUBNET_ID --query
'NetworkAcls[].Associations[0].NetworkACLId' --output text) # Get the ID of the
network ACL associated with the subnet, and store it temporarily.
```

```
aws ec2 create-network-acl-entry --network-acl-id $MY_NETWORK_ACL_ID --ingress --
protocol tcp --rule-action allow --rule-number 10000 --cidr-block 0.0.0.0/0 --port-
range From=80,To=80 # Add an inbound rule to the network ACL to allow all IPv4-
based traffic over port 80. Advanced users: change this suggested rule number as
desired.
```

```
aws ec2 create-network-acl-entry --network-acl-id $MY_NETWORK_ACL_ID --ingress --
protocol tcp --rule-action allow --rule-number 10100 --ipv6-cidr-block ::/0 --port-
range From=80,To=80 # Add an inbound rule to the network ACL to allow all IPv6-
based traffic over port 80. Advanced users: change this suggested rule number as
desired.
```

4. ウェブサーバーのルート内にある `index.php` ファイルへの URL を取得します。これを行うには、次のコマンドを実行し、新しいウェブブラウザタブまたは AWS Cloud9 IDE とは別のウェブブラウザを使用して、表示される URL に移動します。正常に実行された場合は、Apache HTTP Server、MySQL、PHP、およびその他の関連設定に関する情報がウェブページに表示されます。

```
MY_PUBLIC_IP=$(curl http://169.254.169.254/latest/meta-data/public-ipv4) && echo
http://$MY_PUBLIC_IP/index.php # Get the URL to the index.php file within the web
server root.
```

ステップ 4: クリーンアップする

この環境を引き続き使用するものの、ポート 80 経由の着信ウェブトラフィックを無効にしたいとします。その場合、次の 8 個のコマンドを 1 度にひとつずつ次の順序で実行し、環境に関連付けられているセキュリティグループおよびネットワーク ACL で先ほど設定した該当の着信トラフィック

ルールを削除します。各コマンドの動作を理解するには、各コマンドの # の後に表示される情報をお読みください。

Important

次のコマンドを実行すると、この環境のセキュリティグループとネットワーク ACL に関連付けられているすべての EC2 環境と Amazon EC2 インスタンスに対して、ポート 80 経由の着信 ウェブトラフィックが無効になります。これにより、これ以外の EC2 環境および Amazon EC2 インスタンスに対して、ポート 80 経由の着信ウェブトラフィックが予想せずに無効になる可能性があります。

Note

次の 5 番目から 8 番目のコマンドでは、既存のルールを削除して、ネットワーク ACL でポート 80 経由の着信ウェブトラフィックが許可されないようにします。デフォルトのネットワーク ACL があり、すべてのポート経由ですべての着信トラフィックがすでに許可されている場合は、これらのコマンドの実行をスキップすることができます。ただし、ポート 80 経由の着信ウェブトラフィックを許可する既存のルールを含むカスタムネットワーク ACL があり、それらのルールを削除したいとします。その場合、最初のコマンドとそれに続く 5 番目から 8 番目のコマンドを実行する必要があります。

```
MY_INSTANCE_ID=$(curl http://169.254.169.254/latest/meta-data/instance-id) # Get the ID of the instance for the environment, and store it temporarily.
```

```
MY_SECURITY_GROUP_ID=$(aws ec2 describe-instances --instance-id $MY_INSTANCE_ID --query 'Reservations[].Instances[0].SecurityGroups[0].GroupId' --output text) # Get the ID of the security group associated with the instance, and store it temporarily.
```

```
aws ec2 revoke-security-group-ingress --group-id $MY_SECURITY_GROUP_ID --protocol tcp --cidr 0.0.0.0/0 --port 80 # Delete the existing inbound rule from the security group to block all incoming IPv4-based traffic over port 80.
```

```
aws ec2 revoke-security-group-ingress --group-id $MY_SECURITY_GROUP_ID --ip-permissions IpProtocol=tcp,Ipv6Ranges='[{"CidrIpv6>:::/0}]',FromPort=80,ToPort=80 # Delete the existing inbound rule from the security group to block all incoming IPv6-based traffic over port 80.
```

```
MY_SUBNET_ID=$(aws ec2 describe-instances --instance-id $MY_INSTANCE_ID --query
'Reservations[].Instances[0].SubnetId' --output text) # Get the ID of the subnet
associated with the instance, and store it temporarily.

MY_NETWORK_ACL_ID=$(aws ec2 describe-network-acls --filters Name=association.subnet-
id,Values=$MY_SUBNET_ID --query 'NetworkAcls[].Associations[0].NetworkAclId' --output
text) # Get the ID of the network ACL associated with the subnet, and store it
temporarily.

aws ec2 delete-network-acl-entry --network-acl-id $MY_NETWORK_ACL_ID --ingress --rule-
number 10000 # Delete the existing inbound rule from the network ACL to block all IPv4-
based traffic over port 80. Advanced users: if you originally created this rule with a
different number, change this suggested rule number to match.

aws ec2 delete-network-acl-entry --network-acl-id $MY_NETWORK_ACL_ID --ingress --rule-
number 10100 # Delete the existing inbound rule from the network ACL to block all IPv6-
based traffic over port 80. Advanced users: if you originally created this rule with a
different number, change this suggested rule number to match.
```

この環境を使用し終わったら、今後、AWS アカウントに料金が請求されないように、その環境を削除します。手順については、[での環境の削除 AWS Cloud9](#) を参照してください。

の WordPress チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境内で WordPress をインストールして実行できます。WordPress は、配信ウェブコンテンツに広く使用されているオープンソースのコンテンツ管理システム (CMS) です。

Note

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。これには、Amazon Elastic Compute Cloud (Amazon EC2) などのサービスに対して発生する可能性のある料金が含まれます。詳細については、「[Amazon EC2 の料金](#)」を参照してください。

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。
- すべての最新ソフトウェアパッケージを含む最新の EC2 インスタンスを持っています。AWS Cloud9 IDE ターミナルウィンドウで、 オプションyum updateを使用して を実行し、確認を求めることなく更新-yをインストールできます。インストール前に更新を検査する場合は、このオプションを省略できます。

```
sudo yum update -y
```

インストールの概要

環境の EC2 インスタンスに WordPress をインストールするには、以下のステップを実行:

1. WordPress インストールに関する情報を格納するオープンソースのリレーショナルデータベースである MariaDB サーバーのインストールと設定
2. WordPress のインストールと設定。これには、wordpress.conf 設定ファイルの編集が含まれます。
3. WordPress サイトをホストする Apache サーバーの設定
4. Apache サーバーによってホストされている WordPress ウェブコンテンツのプレビュー

ステップ 1: MariaDB サーバーのインストールと設定

1. AWS Cloud9 IDE で、ウィンドウ、新しいターミナルを選択し、次のコマンドを入力して MariaDB サーバーのインストールをインストールして開始します。

```
sudo yum install -y mariadb-server
sudo systemctl start mariadb
```


- 次に、`mysql_secure_installation` スクリプトを実行して、MariaDB サーバーのインストールのセキュリティを向上させることができます。

スクリプトへの応答を指定するときは、最初の質問に対して、[入力] を押して、root パスワードを空白のままにします。Set root password? に対して `n` を押し、残りの各セキュリティオプションに対しては `y` を押します。

```
mysql_secure_installation
```

- 今、MariaDB クライアントを使用して WordPress 情報を保存するためデータベーステーブルを作成します。

(パスワードを求められた場合、入力を押します。)

```
sudo mysql -u root -p
MariaDB [(none)]> create database wp_test;
MariaDB [(none)]> grant all privileges on wp_test.* to root@localhost identified by
';'
```

- MariaDB クライアントからログアウトするには、`exit` コマンドを実行します。

ステップ 2: WordPress のインストールと設定

- IDE ターミナルウィンドウで、`environment` ディレクトリに移動して、ディレクトリ `config` および `wordpress` を作成します。次に、`touch` コマンドを実行して、`config` ディレクトリで `wordpress.conf` という名前のファイルを作成:

```
cd /home/ec2-user/environment
mkdir config wordpress
touch config/wordpress.conf
```

- IDE エディタまたは `vim` を使用して、`wordpress.conf` を更新して、Apache サーバーによる WordPress コンテンツの提供を許可するホスト構成情報に置き換え:

```
# Ensure that Apache listens on port 80
Listen 8080
<VirtualHost *:8080>
    DocumentRoot "/var/www/wordpress"
    ServerName www.example.org
    # Other directives here
```

```
</VirtualHost>
```

- 次に、次のコマンドを実行して必要なアーカイブファイルを取得し、WordPress をインストール:

```
cd /home/ec2-user/environment
wget https://wordpress.org/latest.tar.gz
tar xvf latest.tar.gz
```

- touch コマンドを実行して、environment/wordpress ディレクトリで wp-config.php という名前のファイルを作成:

```
touch wordpress/wp-config.php
```

- IDE エディタまたは vim を使用して wp-config.php を更新します。次に、サンプルデータを自分の設定に置き換えます。

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wp_test' );

/** MySQL database username */
define( 'DB_USER', 'wp_user' );

/** MySQL database password */
define( 'DB_PASSWORD', 'YourSecurePassword' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

define('FORCE_SSL', true);

if ( $_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https' ) $_SERVER['HTTPS'] = 'on';
```

ステップ 3: Apache HTTP サーバーの設定

1. AWS Cloud9 IDE ターミナルウィンドウで、Apache がインストールされていることを確認します。

```
httpd -v
```

Apache サーバー がインストールされていないのであれば、次のコマンドを実行:

```
sudo yum install -y httpd
```

2. /etc/httpd/conf.d ディレクトリに移動します。このディレクトリは Apache の仮想ホスト設定ファイルの場所です。次に、ln コマンドを実行して、以前に作成した wordpress.conf を現在の作業ディレクトリ (/etc/httpd/conf.d)にリンク:

```
cd /etc/httpd/conf.d
sudo ln -s /home/ec2-user/environment/config/wordpress.conf
```

3. さて、/var/www ディレクトリに移動します。ここは Apache サーバーのデフォルトのルートフォルダです。そして、ln コマンドを実行して、先ほど作成した wordpress ディレクトリを、現在の作業ディレクトリ (/var/www)とリンク:

```
cd /var/www
sudo ln -s /home/ec2-user/environment/wordpress
```

4. chmod コマンドを実行して、Apache サーバーによる wordpress サブディレクトリでのコンテンツ実行を許可:

```
sudo chmod +x /home/ec2-user/
```

5. Apache サーバーを再起動して、新しい設定の検出を許可:

```
sudo service httpd restart
```

ステップ 4: WordPress ウェブコンテンツのプレビュー

1. AWS Cloud9 IDE を使用して、ディレクトリ index.html に という名前の新しいファイルを作成します environment/wordpress。

2. HTML 形式のテキストを `index.html` に追加します。例えば、次のようになります。

```
<h1>Hello World!</h1>
```

3. [環境]ウィンドウで、`index.html` ファイルを選択してから、[プレビュー]、[実行中のアプリケーションのプレビュー] の順に選択します。

Hello World!メッセージを表示しているウェブページは、アプリケーションのプレビュータブに表示されます。ご希望のブラウザでウェブコンテンツを表示するには、Pop Out Into New Window (新しいウィンドウで開く)を選択してください。

`index.html` ファイルを削除し、アプリケーションのプレビュータブを更新すると、WordPress の設定ページが表示されます。

混在コンテンツのエラーを管理する

ウェブブラウザは、HTTPS スクリプトと HTTP スクリプトまたはコンテンツを同時にロードしている場合、WordPress サイトの混在コンテンツエラーを表示します。エラーメッセージの文言は、使用しているウェブブラウザによって異なりますが、サイトへの接続が安全でないか、完全に安全でないことが通知されます。また、ウェブブラウザは、混在コンテンツへのアクセスをブロックします。

Important

デフォルトで、AWS Cloud9 IDE アプリケーションのプレビュータブでアクセスするすべてのウェブページは自動的に HTTPS プロトコルを使用します。ページの URI が安全でない http プロトコルをとりあげている場合、https によって自動的に置き換えられます。また、手動で https を http に変更しても、安全でないコンテンツにアクセスすることはできません。

ウェブサイトに HTTPS 実装に関するガイダンスについては、[WordPress のドキュメント](#)を参照してください。

の Java チュートリアル AWS Cloud9

Important

2 GiB 以上のメモリを持つ EC2 インスタンスにバックアップされた AWS Cloud9 開発環境を使用している場合は、拡張 Java サポートを有効にすることをお勧めします。有効にすると、コード補完、エラーの linting、コンテキスト固有のアクション、ブレークポイントやステップニングなどのデバッグオプションを含む生産性向上機能が使用できます。

詳細については、「[Java 開発のサポートの強化](#)」を参照してください。

このチュートリアルでは、AWS Cloud9 開発環境でいくつかの Java コードを実行できます。

このチュートリアルに従って、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを構築して実行する](#)
- [ステップ 4: を使用するようにを設定する AWS SDK for Java](#)
- [ステップ 5: 環境で AWS 認証情報管理を設定する](#)
- [ステップ 6: AWS SDK コードを追加する](#)
- [ステップ 7: AWS SDK コードを構築して実行する](#)
- [ステップ 8: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示

を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。

- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは、一連の Java 開発ツール AWS Cloud9 を開発環境にインストールします。Oracle JDK や OpenJDK などの Java 開発ツールのセットが環境にインストール済みである場合は、[ステップ 2: コードを追加する](#)に進むことができます。このサンプルは、OpenJDK 8 で開発されました。次の手順を実行して、環境にインストールすることができます。

1. OpenJDK 8 が既にインストールされているかどうかを確認します。これを行うには、IDE AWS Cloud9 のターミナルセッションで、**-version** オプションを使用して Java ランナーのコマンドラインバージョンを実行します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。)

```
java -version
```

上記のコマンドの出力に基づいて、次のいずれかの操作を行います。

- java コマンドが見つからないことを出力が示している場合は、この手順のステップ 2 に進み、OpenJDK 8 をインストールします。
 - 出力に Java(TM)、Java Runtime Environment、Java SE、J2SE、または Java2 から始まる値が含まれている場合、OpenJDK がインストールされていないか、デフォルト Java 開発ツールセットとして設定されていません。この手順のステップ 2 に進んで OpenJDK 8 をインストールし、OpenJDK 8 の使用に切り替えます。
 - 出力に java version 1.8 および OpenJDK で始まる値がある場合は、[ステップ 2: コードを追加する](#)に進みます。このサンプルの OpenJDK 8 が正しくインストールされます。
 - 出力に java version 未満の 1.8 と OpenJDK で始まる値が含まれている場合、この手順のステップ 2 に進んで、インストールされた OpenJDK バージョンを OpenJDK 8 にアップグレードします。
2. 最新のセキュリティ更新およびバグ修正がインストールされていることを確認します。これを行うには、yum ツール (Amazon Linux の場合) または apt ツール (Ubuntu Server の場合) を **update** コマンドを実行します。

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

3. OpenJDK 8 をインストールします。これを行うには、yum ツール (Amazon Linux の場合) または apt ツール (Ubuntu Server の場合) を **install** コマンドを使用して OpenJDK 8 パッケージを実行します。

Amazon Linux の場合:

```
sudo yum -y install java-1.8.0-openjdk-devel
```

Ubuntu Server の場合:

```
sudo apt install -y openjdk-8-jdk
```

詳細については、OpenJDK ウェブサイトで「[How to download and install prebuilt OpenJDK packages](#)」を参照してください。

4. デフォルトの Java 開発ツールセットを OpenJDK 8 に切り替えるかアップグレードします。これを行うには、**--config** オプションを使用して **update-alternatives** コマンドを実行します。このコマンドを 2 回実行し、Java ランナーおよびコンパイラのコマンドラインバージョンに切り替えるかアップグレードします。

```
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

各プロンプトで、OpenJDK 8 の選択番号を入力します (java-1.8 を含む番号)。

5. Java ランナーおよびコンパイラのコマンドラインバージョンが OpenJDK 8 を使用していることを確認します。これを行うには、**-version** オプションを使用して Java ランナーおよびコンパイラのコマンドラインバージョンを実行します。

```
java -version
```

```
javac -version
```

OpenJDK 8 が適切にインストールされて設定されている場合、Java ランナーバージョンの出力には `openjdk version 1.8` で始まる値が含まれており、Java コンパイラバージョンの出力は値 `javac 1.8` から始まります。

ステップ 2: コードを追加する

AWS Cloud9 IDE で、次のコードでファイルを作成し、という名前でファイルを保存します `hello.java`。(メニューバーでファイルを作成するには、ファイル、New File (新しいファイル)を選択します。ファイルを保存するには、ファイル、保存を選択します。)

```
public class hello {

    public static void main(String []args) {
        System.out.println("Hello, World!");

        System.out.println("The sum of 2 and 3 is 5.");

        int sum = Integer.parseInt(args[0]) + Integer.parseInt(args[1]);

        System.out.format("The sum of %s and %s is %s.\n",
            args[0], args[1], Integer.toString(sum));
    }
}
```

ステップ 3: コードを構築して実行する

1. Java コンパイラのコマンドラインバージョンを使用して、`hello.java` ファイルを `hello.class` ファイルにコンパイルします。これを行うには、IDE AWS Cloud9 のターミナルを使用して、`hello.java` ファイルと同じディレクトリから、`hello.java` ファイルを指定して Java コンパイラを実行します。

```
javac hello.java
```

2. Java ランナーのコマンドラインバージョンを使用して `hello.class` ファイルを実行します。これを行うには、`hello.class` ファイルと同じディレクトリから、追加する 2 つの整数 (`hello` と `hello.java` など) を使用して、5 ファイルで宣言した 9 クラスの名前を指定します。


```
java hello 5 9
```

3. 出力を比較します。

```
Hello, World!  
The sum of 2 and 3 is 5.  
The sum of 5 and 9 is 14.
```

ステップ 4: を使用するように を設定する AWS SDK for Java

このサンプルを強化して、AWS SDK for Java を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、先ほど作成したバケットを削除できます。

このステップでは、[Apache Maven](#) または [Gradle](#) を環境にインストールします。Maven と Gradle は、Java プロジェクトで使用できる共通構築自動化システムです。Maven または Gradle をインストールしたら、新しい Java プロジェクトの生成に使用します。この新しいプロジェクトでは、AWS SDK for Java への参照を追加します。これにより AWS SDK for Java、Java コードから Amazon S3 などの AWS サービスとやり取りするのに便利です。

トピック

- [Maven を使用して設定する](#)
- [Gradle を使用して設定する](#)

Maven を使用して設定する

1. 環境に Maven をインストールします。Maven が既にインストールされているかどうかを確認するには、IDE AWS Cloud9 のターミナルを使用して、**-version** オプションを使用して Maven を実行します。

```
mvn -version
```

成功すると、出力に Maven のバージョン番号が表示されます。Maven が既にインストールされている場合、この手順のステップ 4 に進んで Maven を使用し、環境で新しい Java プロジェクトを生成します。

2. ターミナルを使用して次のコマンドを実行し、Maven をインストールします。

Amazon Linux の場合、以下のコマンドは、Maven が保存されているパッケージリポジトリに関する情報を取得し、この情報を使用して Maven をインストールします。

```
sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
sudo yum install -y apache-maven
```

上記のコマンドの詳細については、Fedora Project Wiki ウェブサイトの「[Extra Packages for Enterprise Linux \(EPEL\)](#)」を参照してください。

Ubuntu サーバーでは、代わりに以下のコマンドを実行します。

```
sudo apt install -y maven
```

3. **-version** オプションを指定して Maven を実行し、インストールされていることを確認します。

```
mvn -version
```

4. 新しい Java プロジェクトを生成するには Maven を使用します。これを行うには、ターミナルを使用して、Maven によってプロジェクトを生成するディレクトリから次のコマンドを実行します (環境のルートディレクトリなど)。

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

前述のコマンドは、環境でプロジェクトの次のディレクトリ構造を作成します。

```
my-app
|- src
|   |- main
|       |- java
|           |- com
|               |- mycompany
|                   |- app
|                       |- App.java
|- test
|   |- java
|       |- com
```

```
|           `-- mycompany
|               `-- app
|                   `-- AppTest.java
|-- pom.xml
```

上記のディレクトリ構図の詳細については、Apache Maven プロジェクトウェブサイトの「[Maven Quickstart Archetype](#)」と「[Introduction to the Standard Directory Layout](#)」を参照してください。

5. プロジェクトのプロジェクトオブジェクトモデル (POM) ファイルを変更します (POM ファイルは Maven プロジェクトの設定を定義します)。これを行うには、[Environment (環境)] ウィンドウで my-app/pom.xml ファイルを開きます。エディタで、ファイルの現在の内容を次のコードに置き換えて、pom.xml ファイルを保存します。

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.6.0</version>
        <configuration>
          <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
          </descriptorRefs>
          <archive>
            <manifest>
              <mainClass>com.mycompany.app.App</mainClass>
            </manifest>
          </archive>
        </configuration>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
```

```
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
</plugin>
</plugins>
</build>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk</artifactId>
    <version>1.11.330</version>
  </dependency>
</dependencies>
</project>
```

上記の POM ファイルには、以下のように宣言を指定するプロジェクト設定が含まれています。

- `artifactId` の `my-app` 設定は、プロジェクトのルートディレクトリを設定し、`groupId` の `com.mycompany.app` 設定は `com/mycompany/app` および `package` ファイルで `App.java` サブディレクトリ構造と `AppTest.java` 宣言を設定します。
- `artifactId` の `my-app` 設定に加えて、`packaging` の `jar` 設定、`version` の `1.0-SNAPSHOT` 設定、`descriptorRef` の `jar-with-dependencies` 設定は、出力 JAR ファイルの `my-app-1.0-SNAPSHOT-jar-with-dependencies.jar` の名前を設定します。
- `plugin` セクションは、構築されるすべての依存関係が含まれている単一の JAR を宣言します。
- `groupId` を `com.amazonaws` に設定し、`artifactId` に設定している `dependency` セクションには、AWS SDK for Java ライブラリファイル `aws-java-sdk` が含まれています。使用する AWS SDK for Java バージョンは、`version` 設定で宣言します。別のバージョンを使用するには、このバージョン番号を置き換えます。

[ステップ 5: 環境で AWS 認証情報管理を設定する](#) に進んでください。

Gradle を使用して設定する

1. 環境に Gradle をインストールします。Gradle が既にインストールされているかどうかを確認するには、IDE AWS Cloud9 のターミナルを使用して、**-version** オプションを使用して Gradle を実行します。

```
gradle -version
```

成功すると、出力に Gradle のバージョン番号が表示されます。Gradle が既にインストールされている場合、この手順のステップ 4 に進んで Gradle を使用し、環境で新しい Java プロジェクトを生成します。

2. ターミナルを使用して以下のコマンドを実行し、Gradle をインストールします。以下のコマンドでは、SDKMAN! ツールをインストールして実行し、次に SDKMAN! を使用して最新バージョンの Node.js をインストールします。

```
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install gradle
```

上記のコマンドの詳細については、SDKMAN! ウェブサイトの「[Installation](#)」と Gradle ウェブサイトの「[Install with a package manager](#)」を参照してください。

3. **-version** オプションを指定して Gradle を実行し、インストールされていることを確認します。

```
gradle -version
```

4. Gradle を使用して、環境に新しい Java プロジェクトを生成します。これを行うには、ターミナルを使用し、次のコマンドを実行してプロジェクトのディレクトリを作成した後、そのディレクトリに切り替えます。

```
mkdir my-app
cd my-app
```

5. 次のコマンドを実行して、Gradle が環境の my-app ディレクトリに新しい Java アプリケーションプロジェクトを生成するようにします。

```
gradle init --type java-application
```

前述のコマンドは、環境でプロジェクトの次のディレクトリ構造を作成します。

```
my-app
|- .gradle
|  `-- (various supporting project folders and files)
|- gradle
|  `-- (various supporting project folders and files)
|- src
|  |- main
|  |   `-- java
|  |       `-- App.java
|  `-- test
|       `-- java
|           `-- AppTest.java
|- build.gradle
|- gradlew
|- gradlew.bat
`-- settings.gradle
```

6. プロジェクトの `AppTest.java` を変更します (これを行わない場合、プロジェクトが正常にビルドまたは実行されない可能性があります)。これを行うには、[Environment (環境)] ウィンドウで `my-app/src/test/java/AppTest.java` ファイルを開きます。エディタで、ファイルの現在の内容を次のコードに置き換えて、`AppTest.java` ファイルを保存します。

```
import org.junit.Test;
import static org.junit.Assert.*;

public class AppTest {
    @Test public void testAppExists () {
        try {
            Class.forName("com.mycompany.app.App");
        } catch (ClassNotFoundException e) {
            fail("Should have a class named App.");
        }
    }
}
```

7. プロジェクトの `build.gradle` ファイルを変更します (`build.gradle` ファイルは Gradle プロジェクトの設定を定義します)。これを行うには、[Environment (環境)] ウィンドウで `my-app/build.gradle` ファイルを開きます。エディタで、ファイルの現在の内容を次のコードに置き換えて、`build.gradle` ファイルを保存します。

```
apply plugin: 'java'
apply plugin: 'application'

repositories {
    jcenter()
    mavenCentral()
}

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.3.RELEASE"
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.11.330'
    }
}

dependencies {
    compile 'com.amazonaws:aws-java-sdk-s3'
    testCompile group: 'junit', name: 'junit', version: '4.12'
}

run {
    if (project.hasProperty("appArgs")) {
        args Eval.me(appArgs)
    }
}

mainClassName = 'App'
```

上記の build.gradle ファイルには、以下のように宣言を指定するプロジェクト設定が含まれています。

- `io.spring.dependency-management` プラグインは、AWS SDK for Java Maven 部品表 (BOM) をインポートしてプロジェクトの AWS SDK for Java 依存関係を管理するために使用されます。は使用するバージョンを `classpath` 宣言します。別のバージョンを使用するには、このバージョン番号を置き換えます。
- `com.amazonaws:aws-java-sdk-s3` には、AWS SDK for Java ライブラリファイルの Amazon S3 の部分が含まれています。使用するバージョンは `mavenBom` で宣言します。別のバージョンを使用する場合、このバージョン番号を置き換えます。

ステップ 5: 環境で AWS 認証情報管理を設定する

を使用して AWS サービスを AWS SDK for Java 呼び出すたびに、呼び出しで一連の AWS 認証情報を指定する必要があります。これらの認証情報は、にその呼び出しを行うための適切なアクセス許可 AWS SDK for Java があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[の環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for Java デベロッパーガイドの「[開発用の AWS 認証情報とリージョンのセットアップ](#)」を参照してください。

ステップ 6: AWS SDK コードを追加する

このステップでは、Amazon S3 を操作してバケットを作成し、使用可能なバケットの一覧を表示した後、作成したバケットを削除するコードを追加します。

[Environment (環境)] ウィンドウから、Maven の場合は `my-app/src/main/java/com/mycompany/app/App.java` ファイルを開き、Gradle の場合は `my-app/src/main/java/App.java` ファイルを開きます。エディタで、ファイルの現在の内容を次のコードに置き換えて、`App.java` ファイルを保存します。

```
package com.mycompany.app;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.CreateBucketRequest;
```



```
import java.util.List;

public class App {

    private static AmazonS3 s3;

    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.format("Usage: <the bucket name> <the AWS Region to use>\n" +
                "Example: my-test-bucket us-east-2\n");
            return;
        }

        String bucket_name = args[0];
        String region = args[1];

        s3 = AmazonS3ClientBuilder.standard()
            .withCredentials(new ProfileCredentialsProvider())
            .withRegion(region)
            .build();

        // List current buckets.
        ListMyBuckets();

        // Create the bucket.
        if (s3.doesBucketExistV2(bucket_name)) {
            System.out.format("\nCannot create the bucket. \n" +
                "A bucket named '%s' already exists.", bucket_name);
            return;
        } else {
            try {
                System.out.format("\nCreating a new bucket named '%s'...\n\n",
bucket_name);
                s3.createBucket(new CreateBucketRequest(bucket_name, region));
            } catch (AmazonS3Exception e) {
                System.err.println(e.getErrorMessage());
            }
        }

        // Confirm that the bucket was created.
        ListMyBuckets();

        // Delete the bucket.
```

```
        try {
            System.out.format("\nDeleting the bucket named '%s'...\n\n", bucket_name);
            s3.deleteBucket(bucket_name);
        } catch (AmazonS3Exception e) {
            System.err.println(e.getMessage());
        }

        // Confirm that the bucket was deleted.
        ListMyBuckets();
    }

    private static void ListMyBuckets() {
        List<Bucket> buckets = s3.listBuckets();
        System.out.println("My buckets now are:");

        for (Bucket b : buckets) {
            System.out.println(b.getName());
        }
    }
}
```

ステップ 7: AWS SDK コードを構築して実行する

前のステップからコードを実行するには、ターミナルから以下のコマンドを実行します。これらのコマンドは、Maven または Gradle を使用してプロジェクトの実行可能な JAR ファイルを作成した後、Java ランナーを使用して JAR を実行します。JAR は、Amazon S3 で作成するバケットの名前 (など my-test-bucket) と、バケットを作成する AWS リージョンの ID (など) を入力として実行します us-east-2。

Maven の場合、以下のコマンドを実行します。

```
cd my-app
mvn package
java -cp target/my-app-1.0-SNAPSHOT-jar-with-dependencies.jar com.mycompany.app.App my-test-bucket us-east-2
```

Gradle の場合、以下のコマンドを実行します。

```
gradle build
```

```
gradle run -PappArgs="['my-test-bucket', 'us-east-2']"
```

結果を以下の出力と比較します。

```
My buckets now are:

Creating a new bucket named 'my-test-bucket'...

My buckets now are:

my-test-bucket

Deleting the bucket named 'my-test-bucket'...

My buckets now are:
```

ステップ 8: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の C++ チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境で C++ コードを実行できます。このコードは、[AWS SDK for C++](#) で提供されたリソースも使用できます。Amazon Web Services に接続するために使用できる、モジュール化されたクロスプラットフォームのオープンソースライブラリです。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1 : g ++ と必要な dev パッケージをインストールする](#)
- [ステップ 2 : CMake をインストールする](#)
- [ステップ 3: SDK for C++ の取得と構築](#)
- [ステップ 4 : C ++ および CMakeLists ファイルを作成する](#)

- [ステップ 5: C++ コードを構築および実行する](#)
- [ステップ 6: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ1：g++と必要な dev パッケージをインストールする

C++ アプリケーションを構築して実行するには、g++ など、[GNUコンパイラコレクション \(GCC\)](#)によって提供される C++コンパイラであるユーティリティが必要です。

また、ヘッダーファイル (-devパッケージ) を libcurl、libopenssl、libuuid、zlib、およびオプションで、libpulse Amazon Polly のサポートのため追加する必要があります。

開発ツールをインストールするプロセスは、Amazon Linux/Amazon Linux 2 インスタンスと Ubuntu インスタンスのどちらを使用しているかによって若干異なります。

Amazon Linux-based systems

AWS Cloud9 ターミナルで次のコマンドを実行して、が既にgccインストールされているかどうかを確認できます。

```
g++ --version
```

もし g++ がインストールされていない場合は、「開発ツール」と呼ばれるパッケージグループのパートを簡単にインストールできます。これらのツールは、yum groupinstallコマンドでインスタンスに追加:

```
sudo yum groupinstall "Development Tools"
```

g++ --version を再度実行して、コンパイラがインストールされていることを確認します。

次に、システムのパッケージマネージャを使用して、必要なライブラリのパッケージをインストールします。

```
sudo yum install libcurl-devel openssl-devel libuuid-devel pulseaudio-libs-devel
```

Ubuntu-based systems

AWS Cloud9 ターミナルで次のコマンドを実行して、 が既にgccインストールされているかどうかを確認できます。

```
g++ --version
```

gcc がインストールされていない場合は、次のコマンドを実行して Ubuntu ベースのシステムにインストールできます。

```
sudo apt update
sudo apt install build-essential
sudo apt-get install manpages-dev
```

g++ --version を再度実行して、コンパイラがインストールされていることを確認します。

次に、システムのパッケージマネージャを使用して、必要なライブラリのパッケージをインストールします。

```
sudo apt-get install libcurl4-openssl-dev libssl-dev uuid-dev zlib1g-dev libpulse-dev
```

ステップ2：CMakeをインストールする

cmake ツールをインストールする必要があります。このツールは、ソースコードから実行可能ファイルを構築するプロセスを自動化します。

1. IDE のターミナルウィンドウで、次のコマンドを実行して、必要なアーカイブを取得します。

```
wget https://cmake.org/files/v3.18/cmake-3.18.0.tar.gz
```

2. アーカイブからファイルを抽出し、解凍したファイルが含まれているディレクトリに移動します。

```
tar xzf cmake-3.18.0.tar.gz
cd cmake-3.18.0
```

3. 次に、ブートストラップスクリプトを実行し、次のコマンドを実行して cmake をインストールします。

```
./bootstrap
make
sudo make install
```

4. 次のコマンドを実行して、ツールがインストールされていることを確認します。

```
cmake --version
```

ステップ 3: SDK for C++ の取得と構築

AWS SDK for C++ を設定するには、ソースから直接 SDK を構築するか、パッケージマネージャーを使用してライブラリをダウンロードします。使用可能なオプションの詳細については、「[AWS SDK for C++ デベロッパーガイド](#)」の [AWS 「SDK for C++ の使用開始」](#) を参照してください。

このサンプルは git の使用を示し、SDK ソースコードと cmake のクローンを作成して、SDK for C++ を構築します。

1. ターミナルで次のコマンドを実行して、リモートリポジトリをクローンし、AWS Cloud9 環境のため、すべての git サブモジュールを再帰的に取得します。

```
git clone --recurse-submodules https://github.com/aws/aws-sdk-cpp
```

2. 新しいaws-sdk-cppディレクトリに移動し、AWS SDK for C++ を構築するためのサブディレクトリを作成し、それに移動します。

```
cd aws-sdk-cpp
mkdir sdk_build
```

```
cd sdk_build
```

3.

Note

時間を節約するため、このステップは、AWS SDK for C++の Amazon S3 部分のみを構築します。完全なSDKを構築したいのであれば、cmake コマンドから `-DBUILD_ONLY=s3` を省略します。

完全な SDK for C++ を構築すると、Amazon EC2 インスタンスまたは自分のサーバーで利用できるコンピューティングリソースに応じて、完了までに 1 時間を超えることがあります。

cmake を使って、次のコマンドを実行して `sdk_build` ディレクトリに SDK for C++ の Amazon S3 の部分を構築します。

```
cmake .. -DBUILD_ONLY=s3
```

4. では、`make install` コマンドを実行して、ビルドされた SDK にアクセスできるようにします。

```
sudo make install  
cd ..
```

ステップ4 : C ++および CMakeLists ファイルを作成する

このステップでは、C++ ファイルを作成し、プロジェクトのユーザーに Amazon S3 バケットと対話を許可します。

また、CMakeLists.txt ファイルを作成して、C ++ライブラリを構築するために、cmake で使用する指示を提供します。

1. AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、 という名前のファイルを環境のルート (/) `s3-demo.cpp` に保存します。

```
#include <iostream>  
#include <aws/core/Aws.h>  
#include <aws/s3/S3Client.h>  
#include <aws/s3/model/Bucket.h>  
#include <aws/s3/model/CreateBucketConfiguration.h>
```

```

#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>

// Look for a bucket among all currently available Amazon S3 buckets.
bool FindTheBucket(const Aws::S3::S3Client &s3Client,
                  const Aws::String &bucketName) {

    Aws::S3::Model::ListBucketsOutcome outcome = s3Client.ListBuckets();

    if (outcome.IsSuccess()) {

        std::cout << "Looking for a bucket named '" << bucketName << "'..."
                  << std::endl << std::endl;

        Aws::Vector<Aws::S3::Model::Bucket> bucket_list =
            outcome.GetResult().GetBuckets();

        for (Aws::S3::Model::Bucket const &bucket: bucket_list) {
            if (bucket.GetName() == bucketName) {
                std::cout << "Found the bucket." << std::endl << std::endl;

                return true;
            }
        }

        std::cout << "Could not find the bucket." << std::endl << std::endl;
    } else {
        std::cerr << "listBuckets error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

// Create an Amazon S3 bucket.
bool CreateTheBucket(const Aws::S3::S3Client &s3Client,
                    const Aws::String &bucketName,
                    const Aws::String &region) {

    std::cout << "Creating a bucket named '"
              << bucketName << "'..." << std::endl << std::endl;

    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

```



```

    if (region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
            region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome =
        s3Client.CreateBucket(request);

    if (outcome.IsSuccess()) {
        std::cout << "Bucket created." << std::endl << std::endl;
    } else {
        std::cerr << "createBucket error: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

// Delete an existing Amazon S3 bucket.
bool DeleteTheBucket(const Aws::S3::S3Client &s3Client,
                    const Aws::String &bucketName) {

    std::cout << "Deleting the bucket named '"
        << bucketName << "'..." << std::endl << std::endl;

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        s3Client.DeleteBucket(request);

    if (outcome.IsSuccess()) {
        std::cout << "Bucket deleted." << std::endl << std::endl;
    } else {
        std::cerr << "deleteBucket error: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

```
}

#ifndef EXCLUDE_MAIN_FUNCTION
// Create an S3 bucket and then delete it.
// Before and after creating the bucket, and again after deleting the bucket,
// try to determine whether that bucket still exists.
int main(int argc, char *argv[]) {

    if (argc < 3) {
        std::cout << "Usage: s3-demo <bucket name> <AWS Region>" << std::endl
                  << "Example: s3-demo my-bucket us-east-1" << std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::String bucketName = argv[1];
        Aws::String region = argv[2];

        Aws::Client::ClientConfiguration config;

        config.region = region;

        Aws::S3::S3Client s3Client(config);

        if (!FindTheBucket(s3Client, bucketName)) {
            return 1;
        }

        if (!CreateTheBucket(s3Client, bucketName, region)) {
            return 1;
        }

        if (!FindTheBucket(s3Client, bucketName)) {
            return 1;
        }

        if (!DeleteTheBucket(s3Client, bucketName)) {
            return 1;
        }

        if (!FindTheBucket(s3Client, bucketName)) {
            return 1;
        }
    }
}
```

```
    }  
}  
Aws::ShutdownAPI(options);  
  
    return 0;  
}  
#endif // EXCLUDE_MAIN_FUNCTION
```

2. このコンテンツを含んだ 2 番目のファイルを作成し、環境のルート (/) に CMakeLists.txt という名前をつけてファイルを保存します。このファイルを使用すると、コードを実行可能ファイルに組み込むことができます。

```
# A minimal CMakeLists.txt file for the AWS SDK for C++.  
  
# The minimum version of CMake that will work.  
cmake_minimum_required(VERSION 2.8)  
  
# The project name.  
project(s3-demo)  
  
# Locate the AWS SDK for C++ package.  
set(AWSSDK_ROOT_DIR, "/usr/local/")  
set(BUILD_SHARED_LIBS ON)  
find_package(AWSSDK REQUIRED COMPONENTS s3)  
  
# The executable name and its source files.  
add_executable(s3-demo s3-demo.cpp)  
  
# The libraries used by your executable.  
target_link_libraries(s3-demo ${AWSSDK_LINK_LIBRARIES})
```

ステップ 5: C++ コードを構築および実行する

1. s3-demo.cpp と CMakeLists.txt を保存した環境のルートディレクトリで、cmake を実行してプロジェクトを構築します。

```
cmake .  
make
```

- これで、コマンドラインからプログラムを実行できます。次のコマンドで、my-unique-bucket-name を Amazon S3 バケットの一意の名前に置き換え、必要であれば、us-east-1 を別の AWS リージョン (バケットを作成したい場所) の識別子に置き換えます。

```
./s3-demo my-unique-bucket-name us-east-1
```

プログラムが正常に実行された場合は、次のような出力が返されます。

```
Looking for a bucket named 'my-unique-bucket-name'...

Could not find the bucket.

Creating a bucket named 'my-unique-bucket-name'...

Bucket created.

Looking for a bucket named 'my-unique-bucket-name'...

Found the bucket.

Deleting the bucket named 'my-unique-bucket-name'...

Bucket deleted.

Looking for a bucket named 'my-unique-bucket-name'...

Could not find the bucket.
```

ステップ 6: クリーンアップする

このサンプルの使用が終了した後、AWS アカウントへの継続的な課金を防ぐには、環境を削除します。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の Python チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境で Python コードを実行する方法を示します。

このチュートリアルに従うと、AWS アカウントに料金が発生する可能性があります。これには、Amazon Elastic Compute Cloud (Amazon EC2) や Amazon Simple Storage Service (Amazon S3)

などのサービスに対して発生する料金が含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: Python をインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを実行する](#)
- [ステップ 4: をインストールして設定する AWS SDK for Python \(Boto3\)](#)
- [ステップ 5: AWS SDK コードを追加する](#)
- [ステップ 6: AWS SDK コードを実行する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このチュートリアルを使用する前に、以下の前提条件を満たしていることを確認します。

- AWS Cloud9 EC2 開発環境がある

このチュートリアルでは、EC2 環境があり、環境が Amazon Linux または Ubuntu Server を実行している Amazon EC2 インスタンスに接続されているという前提を立てています。詳細については、「[EC2 環境を作成する](#)」を参照してください。

別のタイプの環境またはオペレーティングシステムを使用している場合は、このチュートリアルの手順の調整が必要になる場合があります。

- その環境の AWS Cloud9 IDE を開いた

環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: Python をインストールする

1. AWS Cloud9 IDE のターミナルセッションで、**python --version** コマンドを実行して Python が既にインストールされているかどうかを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の

順に選択します。) Python がインストール済みである場合は、[ステップ 2: コードを追加する](#)に進んでください。

2. **yum update** (Amazon Linux 用) または **apt update** (Ubuntu Server 用) コマンドを実行して、最新のセキュリティ更新プログラムおよびバグ修正がインストールされていることを確認します。

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

3. **install** コマンドを実行して Python をインストールします。

Amazon Linux の場合:

```
sudo yum -y install python3
```

Ubuntu Server の場合:

```
sudo apt-get install python3
```

ステップ 2: コードを追加する

AWS Cloud9 IDE で、次の内容のファイルを作成し、という名前でファイルを保存しますhello.py。(ファイルを作成するには、メニューバーでファイル、New File (新しいファイル)を選択します。 ファイルを保存するには、ファイル,保存を選択します。)

```
import sys

print('Hello, World!')

print('The sum of 2 and 3 is 5.')

sum = int(sys.argv[1]) + int(sys.argv[2])
```

```
print('The sum of {0} and {1} is {2}.'.format(sys.argv[1], sys.argv[2], sum))
```

ステップ 3: コードを実行する

1. AWS Cloud9 IDE のメニューバーで、Run、Run Configurations、New Run Configuration を選択します。
2. [新規] - 停止タブ、コマンドに、`hello.py 5 9` を入力します。コード内の 5 は `sys.argv[1]` を、9 は `sys.argv[2]` を表します。
3. [Run (実行)] を選択して、出力を比較します。

```
Hello, World!
The sum of 2 and 3 is 5.
The sum of 5 and 9 is 14.
```

4. デフォルトでは、はコードのランナー AWS Cloud9 を自動的に選択します。ランナーを変更するには、[Runner (ランナー)] を選択し、[Python 2] または [Python 3] を選択します。

Note

特定のバージョンの Python 用にカスタムランナーを作成できます。詳細については、[「ビルダーまたはランナーを作成する」](#)を参照してください。

ステップ 4: をインストールして設定する AWS SDK for Python (Boto3)

AWS SDK for Python (Boto3) を使用すると、Python コードを使用して Amazon S3 などの AWS サービスとやり取りできます。たとえば、SDK を使用して Amazon S3 バケットを作成し、利用可能なバケットを一覧表示して、作成したばかりのバケットを削除できます。

`pip` をインストールします。

AWS Cloud9 IDE で、`python -m pip --version` コマンドを実行して、`pip` が Python のアクティブバージョンに既にインストールされているかどうかを確認します。`pip` がインストール済みである場合は、次のセクションに進みます。

次のコマンドを実行して `pip` をインストールします。`sudo` の環境はユーザーの環境と異なるため、使用する Python のバージョンが現在のエイリアスバージョンと異なる場合は、バージョンを指定する必要があります。

```
curl -O https://bootstrap.pypa.io/get-pip.py # Get the install script.
sudo python3 get-pip.py                      # Install pip for Python 3.
python -m pip --version                      # Verify pip is installed.
rm get-pip.py                               # Delete the install script.
```

詳細については、pip ウェブサイトの「[インストール](#)」を参照してください。

のインストール AWS SDK for Python (Boto3)

をインストールしたらpip、 **pip install** コマンド AWS SDK for Python (Boto3) を実行して をインストールします。

```
sudo python3 -m pip install boto3 # Install boto3 for Python 3.
python -m pip show boto3         # Verify boto3 is installed for the current version
of Python.
```

詳細については、[の「クイックスタート AWS SDK for Python \(Boto3\)」](#) セクションの「インストール」を参照してください。

ステップ 2: 環境 で認証情報管理を設定する

を使用して AWS サービスを AWS SDK for Python (Boto3) 呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報により、呼び出しに必要なアクセス許可が SDK にあるかどうかを判断します。認証情報に必要なアクセス許可がない場合、呼び出しは失敗します。

認証情報を環境に保存するには、[の環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、『[』の「認証情報 AWS SDK for Python \(Boto3\)」](#)を参照してください。

ステップ 5: AWS SDK コードを追加する

Amazon S3 を使用してバケットを作成するコードを追加し、使用可能なバケットを一覧表示して、任意でバケットを削除します。

AWS Cloud9 IDE で、次の内容のファイルを作成し、 という名前でファイルを保存しますs3.py。

```
import sys
import boto3
from botocore.exceptions import ClientError
```



```
def list_my_buckets(s3_resource):
    print("Buckets:\n\t", *[b.name for b in s3_resource.buckets.all()], sep="\n\t")

def create_and_delete_my_bucket(s3_resource, bucket_name, keep_bucket):
    list_my_buckets(s3_resource)

    try:
        print("\nCreating new bucket:", bucket_name)
        bucket = s3_resource.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            },
        )
    except ClientError as e:
        print(
            f"Couldn't create a bucket for the demo. Here's why: "
            f"{e.response['Error']['Message']}"
        )
        raise

    bucket.wait_until_exists()
    list_my_buckets(s3_resource)

    if not keep_bucket:
        print("\nDeleting bucket:", bucket.name)
        bucket.delete()

        bucket.wait_until_not_exists()
        list_my_buckets(s3_resource)
    else:
        print("\nKeeping bucket:", bucket.name)

def main():
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket_name", help="The name of the bucket to create.")
    parser.add_argument("region", help="The region in which to create your bucket.")
    parser.add_argument(
        "--keep_bucket",
        help="Keeps the created bucket. When not "
```

```

        "specified, the bucket is deleted "
        "at the end of the demo.",
        action="store_true",
    )

    args = parser.parse_args()
    s3_resource = (
        boto3.resource("s3", region_name=args.region)
        if args.region
        else boto3.resource("s3")
    )
    try:
        create_and_delete_my_bucket(s3_resource, args.bucket_name, args.keep_bucket)
    except ClientError:
        print("Exiting the demo.")

if __name__ == "__main__":
    main()

```

ステップ 6: AWS SDK コードを実行する

1. メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
2. コマンドには、と入力します。ここで `s3.py my-test-bucket us-west-2`、`my-test-bucket` は作成するバケットの名前、`us-west-2` はバケットが作成される AWS リージョンの ID です。デフォルトでは、バケットはスクリプトが終了する前に削除されます。バケットを保持するには、コマンドに `--keep_bucket` を追加します。AWS リージョン IDs、の [「Amazon Simple Storage Service のエンドポイントとクォータ」](#) を参照してくださいAWS 全般のリファレンス。

Note

Amazon S3 バケット名は、アカウント AWS だけでなく AWS、全体で一意である必要があります。

3. [Run (実行)] を選択して、出力を比較します。

Buckets:

```
a-pre-existing-bucket
```

```
Creating new bucket: my-test-bucket
```

```
Buckets:
```

```
a-pre-existing-bucket
```

```
my-test-bucket
```

```
Deleting bucket: my-test-bucket
```

```
Buckets:
```

```
a-pre-existing-bucket
```

ステップ 7: クリーンアップする

このチュートリアルを完了した後、AWS アカウントへの継続的な課金を防ぐには、AWS Cloud9 環境を削除します。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の .NET チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境でいくつかの .NET コードを実行できます。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2。 \(オプション\): Lambda 関数用の .NET CLI 拡張をインストールする](#)
- [ステップ 3: .NET コンソールアプリケーションプロジェクトを作成する](#)
- [ステップ 4: コードを追加する](#)
- [ステップ 5: コードを構築および実行する](#)
- [ステップ 6: を使用する .NET コンソールアプリケーションプロジェクトを作成してセットアップする AWS SDK for .NET](#)

- [ステップ 7: AWS SDK コードを追加する](#)
- [ステップ 8: AWS SDK コードを構築して実行する](#)
- [ステップ 9: クリーンアップする。](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは、このサンプルの実行に必要な .NET SDK を環境にインストールします。

1. .NET SDK の最新バージョンが環境に既にインストールされているかどうかを確認します。これを行うには、IDE AWS Cloud9 のターミナルセッションで、`--version` オプションを使用して .NET Core コマンドラインインターフェイス (CLI) を実行します。

```
dotnet --version
```

.NET コマンドラインツールのバージョンが表示され、バージョンが 2.0 以上の場合、[ステップ 3: .NET コンソールアプリケーションプロジェクトを作成する](#) まで進みます。バージョンが 2.0 より古い場合、または `bash: dotnet: command not found` などのエラーが表示された場合は、.NET SDK のインストールを続行します。

2. Amazon Linux の場合、IDE AWS Cloud9 のターミナルセッションで次のコマンドを実行して、最新のセキュリティ更新プログラムとバグ修正がインストールされていることを確認し、.NET SDK に必要な `libunwind` パッケージをインストールします。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。)

```
sudo yum -y update
sudo yum -y install libunwind
```

Ubuntu Server の場合、IDE AWS Cloud9 のターミナルセッションで次のコマンドを実行して、最新のセキュリティ更新プログラムとバグ修正がインストールされていることを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。)

```
sudo apt -y update
```

3. 次のコマンドを実行して、.NET SDK インストーラスクリプトを環境にダウンロードします。

```
wget https://dot.net/v1/dotnet-install.sh
```

4. 次のコマンドを実行して、インストールスクリプトを現在のユーザーに実行可能にします。

```
sudo chmod u=rx dotnet-install.sh
```

5. 次のコマンドを実行して、.NET SDK をダウンロードし、インストールするインストーラスクリプトを実行します。

```
./dotnet-install.sh -c Current
```

6. PATH に .NET SDK を追加します。これを行うには、次のように、環境のシェルスクリプトファイル (たとえば、.bashrc ファイル) で、\$HOME/.dotnet サブディレクトリを環境の PATH 変数に追加します。

- a. .bashrc vi コマンドを使用して、ファイルを編集のために開きます。

```
vi ~/.bashrc
```

- b. Amazon Linux では、下向き矢印または j キーを使用して、export PATH で始まる行に移動します。

Ubuntu Server では、「G」と入力して、ファイルの最後の行に移動します。

- c. 右向き矢印または \$ キーを使用して、行の末尾に移動します。
- d. i キーを押して挿入モードに切り替えます (表示の末尾に -- INSERT --- と表示されます)。

- e. Amazon Linux の場合、`:$HOME/.dotnet` を入力することで、`$HOME/.dotnet` サブディレクトリを **PATH** 変数に追加します。必ずコロン文字 (:) を入力します。行は以下のようになります。

```
export PATH=$PATH:$HOME/.local/bin:$HOME/bin:$HOME/.dotnet
```

Ubuntu Server では、右矢印キーを押し、Enter を 2 回押してから、ファイルの末尾に次の行を単独で入力します。

```
export PATH=$HOME/.dotnet:$PATH
```

- f. ファイルを保存します。これを行うには、Esc キーを押して (`-- INSERT ---` が表示の末尾から消えます)、`:wq` と入力し (ファイルに書き込んで終了します)、Enter キーを押します。

7. `.bashrc` ファイルをソースに指定して、.NET SDK をロードします。

```
. ~/.bashrc
```

8. `--help` オプションで .NET CLI を実行することにより、.NET SDK がロードされることを確認します。

```
dotnet --help
```

成功した場合、.NET SDK のバージョン番号が、その他の使用状況情報と共に表示されます。

9. 環境に .NET SDK インストーラスクリプトを保持する必要がなくなった場合は、次の方法で削除できます。

```
rm dotnet-install.sh
```

ステップ 2。(オプション): Lambda 関数用の .NET CLI 拡張をインストールする

このチュートリアルでは必須ではありませんが、`Amazon.Lambda.Tools` パッケージもインストールする場合は、.NET CLI を使用して AWS Lambda 関数と AWS Serverless Application Model アプリケーションをデプロイできます。

1. 次のコマンドを実行して、このパッケージをインストールします。

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. 今、PATH および DOTNET_ROOT 環境変数を設定して、インストールされた Lambda ツールを指します。.bashrc ファイルを見つけるには export PATH セクションを開き、次のように表示されるように編集します (このファイルの編集の詳細については、ステップ 1 を参照してください)。

```
export PATH=$PATH:$HOME/.local/bin:$HOME/bin:$HOME/.dotnet:$HOME/.dotnet/tools
export DOTNET_ROOT=$HOME/.dotnet
```

ステップ 3: .NET コンソールアプリケーションプロジェクトを作成する

このステップでは、.NET を使用して hello という名前のプロジェクトを作成します。このプロジェクトには、.NET が IDE のターミナルからシンプルなアプリケーションを実行するのに必要なすべてのファイルが含まれています。アプリケーションのコードは C# で書かれています。

.NET コンソールアプリケーションプロジェクトを作成します。これを行うには、**new** コマンドを使用して .NET CLI を実行します。次のように、使用するコンソールアプリケーションオブジェクトテンプレートのタイプとプログラミング言語 (このサンプルでは C#) を指定します。

-n オプションは、プロジェクトが新しいディレクトリ hello に出力されることを示します。続いて、そのディレクトリに移動します。

```
dotnet new console -lang C# -n hello
cd hello
```

前述のコマンドは、いくつかのファイルを含む obj という名前のサブディレクトリを追加し、追加のスタンドアロンファイルを hello ディレクトリに追加します。次の 2 つのキーファイルに注意してください。

- hello/hello.csproj ファイルには、コンソールアプリケーションプロジェクトに関する情報が含まれています。
- hello/Program.cs ファイルには、実行するアプリケーションのコードが含まれています。

ステップ 4: コードを追加する

このステップでは、アプリケーションにコードを追加します。

IDE AWS Cloud9 の環境ウィンドウから、hello/Program.cs ファイルを開きます。

エディタで、ファイルの現在の内容を次のコードに置き換えて、Program.cs ファイルを保存します。

```
using System;

namespace hello
{
    class Program
    {
        static void Main(string[] args)
        {
            if (args.Length < 2) {
                Console.WriteLine("Please provide 2 numbers");
                return;
            }

            Console.WriteLine("Hello, World!");

            Console.WriteLine("The sum of 2 and 3 is 5.");

            int sum = Int32.Parse(args[0]) + Int32.Parse(args[1]);

            Console.WriteLine("The sum of {0} and {1} is {2}.",
                args[0], args[1], sum);
        }
    }
}
```

ステップ 5: コードを構築および実行する

このステップでは、プロジェクトとその依存関係を、実行可能なアプリケーションファイルを含む 1 つのバイナリファイルのセットに構築します。次にアプリケーションを実行します。

1. IDE では、次のように .NET のビルダーを作成します。

- a. メニューバーで、[Run (実行)]、[Build System (ビルドシステム)]、[New Build System (新しいビルドシステム)] の順に選択します。
- b. [My Builder.build] タブで、タブの内容を以下のコードに置き換えます。

```
{  
  "cmd" : ["dotnet", "build"],  
  "info" : "Building..."  
}
```

- c. [File (ファイル)]、[Save As (名前を付けて保存)] の順に選択します。
 - d. [Filename (ファイル名)] に「.NET.build」と入力します。
 - e. [Folder (フォルダ)] に「/.c9/builders」と入力します。
 - f. [保存] を選択します。
2. エディタに Program.cs ファイルの内容を表示した状態で、[実行]、[ビルドシステム]、[.NET] の順に選択します。次に、[Run (実行)]、[Build (ビルド)] を選択します。

このビルダは、bin という名前のサブディレクトリを追加し、Debug という名前のサブディレクトリを hello/obj サブディレクトリに追加します。次の 3 つのキーファイルに注意してください。

- この hello/bin/Debug/netcoreapp3.1/hello.dll ファイルは、実行可能なアプリケーションファイルです。
- hello/bin/Debug/netcoreapp3.1/hello.deps.json ファイルは、アプリケーションの依存関係をリストします。
- hello/bin/Debug/netcoreapp3.1/hello.runtimeconfig.json ファイルは、アプリケーションの共有ランタイムとそのバージョンを指定します。

 Note

フォルダ名 netcoreapp3.1 は、この例で使用されている .NET SDK のバージョンを反映しています。インストールしているバージョンによっては、フォルダ名に異なる番号が表示される場合があります。

3. 次のように、.NET のランナーを作成します。

- a. メニューバーで、[Run (実行)]、[Run With (次で実行)]、[New Runner (新しいランナー)] の順に選択します。
- b. [My Runner.run] タブで、タブの内容を以下のコードに置き換えます。

```
{  
  "cmd" : ["dotnet", "run", "$args"],  
  "working_dir": "$file_path",  
  "info" : "Running..."  
}
```

- c. [File (ファイル)]、[Save As (名前を付けて保存)] の順に選択します。
 - d. [Filename (ファイル名)] に「.NET.run」と入力します。
 - e. [Folder (フォルダ)] に「/.c9/runners」と入力します。
 - f. [保存] を選択します。
4. 追加する 2 つの整数を指定してアプリケーションを実行します (例: 5、9) を追加します。
 - a. エディタに Program.cs ファイルの内容を表示した状態で、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
 - b. [[新規] - アイドル] タブで、[ランナー: 自動] を選択し、[.NET] を選択します。
 - c. [Command (コマンド)] ボックスに、「hello 5 9」と入力します。
 - d. [Run] (実行) を選択します。

デフォルトでは、このランナーは .NET に hello/bin/Debug/netcoreapp3.1 ディレクトリで hello.dll ファイルを実行するよう指示します。

出力を以下と比較します。

```
Hello, World!  
The sum of 2 and 3 is 5.  
The sum of 5 and 9 is 14.
```

ステップ 6: を使用する .NET コンソールアプリケーションプロジェクトを作成してセットアップする AWS SDK for .NET

このサンプルを強化して、AWS SDK for .NET を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、先ほど作成したバケットを削除できます。

この新しいプロジェクトでは、AWS SDK for .NETへの参照を追加します。AWS SDK for .NET は、.NET コードから Amazon S3 などの AWS サービスとやり取りする便利な方法を提供します。次に、環境で AWS 認証情報管理を設定します。が AWS サービスとやり取りするには、これらの認証情報 AWS SDK for .NET が必要です。

プロジェクトを作成するには

1. .NET コンソールアプリケーションプロジェクトを作成します。これを行うには、使用するコンソールアプリケーションプロジェクトのテンプレートタイプとプログラミング言語を指定し、**new** コマンドを使用して .NET CLI を実行します。

-n オプションは、プロジェクトが新しいディレクトリ s3 に出力されることを示します。続いて、そのディレクトリに移動します。

```
dotnet new console -lang C# -n s3
cd s3
```

2. AWS SDK for .NETで Amazon S3 パッケージへのプロジェクトリファレンスを追加します。これを行うには、NuGet で Amazon S3 パッケージの名前を指定し、**add package** コマンドを使用して .NET Core を実行します。(NuGet では、.NET のパッケージを作成、ホスト、使用方法が定義され、それらのロールごとにツールが用意されています)。

```
dotnet add package AWSSDK.S3
```

Amazon S3 パッケージにプロジェクトリファレンスを追加すると、NuGet は AWS SDK for .NETの他の部分にもプロジェクトリファレンスを追加します。

Note

NuGet の他の AWS 関連パッケージの名前とバージョンについては、[NuGet ウェブサイトの「aws-sdk でタグ付けされた NuGet パッケージ」](#)を参照してください。NuGet

AWS 認証情報管理を設定するには

を使用して AWS サービスを AWS SDK for .NET 呼び出すたびに、呼び出しで一連の AWS 認証情報を指定する必要があります。これらの認証情報は、にその呼び出しを行うための適切なアクセス許可 AWS SDK for .NET があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

認証情報を環境に保存するには、[の環境 AWS のサービス からの呼び出し AWS Cloud9](#) の指示を実行してから、このトピックに戻ります。

詳細については、「AWS SDK for .NET デベロッパーガイド」の[AWS 「認証情報の設定」](#)を参照してください。

ステップ 7: AWS SDK コードを追加する

このステップでは、Amazon S3 を操作してバケットを作成し、作成したばかりのバケットを削除した後、利用できるバケットのリストを表示するコードを追加します。

IDE AWS Cloud9 の環境ウィンドウから、s3/Program.cs ファイルを開きます。エディタで、ファイルの現在の内容を次のコードに置き換えて、Program.cs ファイルを保存します。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace s3
{
    class Program
    {
        async static Task Main(string[] args)
        {
            if (args.Length < 2) {
                Console.WriteLine("Usage: <the bucket name> <the AWS Region to use>");
                Console.WriteLine("Example: my-test-bucket us-east-2");
                return;
            }

            if (args[1] != "us-east-2") {
                Console.WriteLine("Cannot continue. The only supported AWS Region ID is " +
                    "'us-east-2'.");
                return;
            }

            var bucketRegion = RegionEndpoint.USEast2;
            // Note: You could add more valid AWS Regions above as needed.

            using (var s3Client = new AmazonS3Client(bucketRegion)) {
```

```
var bucketName = args[0];

// Create the bucket.
try
{
    if (await AmazonS3Util.DoesS3BucketExistV2Async(s3Client, bucketName))
    {
        Console.WriteLine("Cannot continue. Cannot create bucket. \n" +
            "A bucket named '{0}' already exists.", bucketName);
        return;
    } else {
        Console.WriteLine("\nCreating the bucket named '{0}'...", bucketName);
        await s3Client.PutBucketAsync(bucketName);
    }
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Cannot continue. {0}", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Cannot continue. {0}", e.Message);
}

// Confirm that the bucket was created.
if (await AmazonS3Util.DoesS3BucketExistV2Async(s3Client, bucketName))
{
    Console.WriteLine("Created the bucket named '{0}'.", bucketName);
} else {
    Console.WriteLine("Did not create the bucket named '{0}'.", bucketName);
}

// Delete the bucket.
Console.WriteLine("\nDeleting the bucket named '{0}'...", bucketName);
await s3Client.DeleteBucketAsync(bucketName);

// Confirm that the bucket was deleted.
if (await AmazonS3Util.DoesS3BucketExistV2Async(s3Client, bucketName))
{
    Console.WriteLine("Did not delete the bucket named '{0}'.", bucketName);
} else {
    Console.WriteLine("Deleted the bucket named '{0}'.", bucketName);
};
```

```
// List current buckets.
Console.WriteLine("\nMy buckets now are:");
var response = await s3Client.ListBucketsAsync();

foreach (var bucket in response.Buckets)
{
    Console.WriteLine(bucket.BucketName);
}
}
}
}
```

ステップ 8: AWS SDK コードを構築して実行する

このステップでは、プロジェクトとその依存関係を、実行可能なアプリケーションファイルを含む 1 つのバイナリファイルのセットに構築します。次にアプリケーションを実行します。

1. プロジェクトをビルドします。これを行うには、エディタに `s3/Program.cs` ファイルの内容を表示した状態で、[Run (実行)]、[Build (ビルド)] の順に選択します。
2. 次のように、作成する Amazon S3 バケットの名前と、バケットを作成する AWS リージョンの ID (例: `my-test-bucket` と `us-east-2`) をがついたアプリケーションを実行します。
 - a. エディタに `s3/Program.cs` ファイルの内容を表示した状態で、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
 - b. [[新規] - アイドル] タブで、[ランナー: 自動] を選択し、[.NET] を選択します。
 - c. コマンドボックスに、アプリケーションの名前、作成する Amazon S3 バケットの名前、バケットを作成する AWS リージョンの ID (例:) を入力します `s3 my-test-bucket us-east-2`。
 - d. [Run] (実行) を選択します。

デフォルトでは、このランナーは .NET に `s3/bin/Debug/netcoreapp3.1` ディレクトリで `s3.dll` ファイルを実行するよう指示します。

結果を以下の出力と比較します。

```
Creating a new bucket named 'my-test-bucket'...
Created the bucket named 'my-test-bucket'.

Deleting the bucket named 'my-test-bucket'...
```

```
Deleted the bucket named 'my-test-bucket'.
```

```
My buckets now are:
```

ステップ 9: クリーンアップする。

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の Node.js チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境でいくつかの Node.js スクリプトを実行できます。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを実行する](#)
- [ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する](#)
- [ステップ 5: AWS SDK コードを追加する](#)
- [ステップ 6: AWS SDK コードを実行する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示

を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。

- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは Node.js をインストールして設定します。このサンプルを実行するために必要なものです。

1. AWS Cloud9 IDE のターミナルセッションで、**node --version** コマンドを実行して Node.js が既にインストールされているかどうかを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal] (新しいターミナル) の順に選択します。) 成功すると、出力に Node.js のバージョン番号が表示されます。Node.js がインストール済みである場合は、[ステップ 2: コードを追加する](#)に進んでください。
2. (Amazon Linux の場合) **yum update** または (Ubuntu Server の場合) **apt update** コマンドを実行して、最新のセキュリティ更新プログラムおよびバグ修正がインストールされていることを確認します。

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

3. Node.js をインストールするには、まず、次のコマンドを実行して Node Version Manager (nvm) をダウンロードします。(nvm は、Node.js のバージョンをインストールして管理するために役立つシンプルな Bash シェルスクリプトです。詳細については、GitHub ウェブサイトの「[Node Version Manager](#)」を参照してください。)

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
```

4. nvm の使用を開始するには、ターミナルセッションを閉じてもう一度起動するか、nvm をロードするコマンドを含む ~/.bashrc ファイルを入手します。


```
. ~/.bashrc
```

5. このコマンドを実行して、Amazon Linux 2、Amazon Linux 1、および Ubuntu 18.04 に Node.js 16 をインストールします。Amazon Linux 1 インスタンスと Ubuntu 18.04 インスタンスは v16 までの Node.js のみをサポートします。

```
nvm install 16
```

このコマンドを実行して、Amazon Linux 2023 および Ubuntu 22.04 に最新バージョンの Node.js をインストールします。

```
nvm install --lts && nvm alias default lts/*
```

Note

最新の AL2023 AWS Cloud9 image には Node.js 20 がインストールされ、最新の Amazon Linux 2 AWS Cloud9 image には Node.js 18 がインストールされています。Amazon Linux 2 に Node.js 18 AWS Cloud9 を手動でインストールする場合は、IDE AWS Cloud9 ターミナルで次のコマンドを実行します。

```
C9_NODE_INSTALL_DIR=~/.nvm/versions/node/v18.17.1
C9_NODE_URL=https://d3kgj69l4ph6w4.cloudfront.net/static/node-amazon/node-
v18.17.1-linux-x64.tar.gz
mkdir -p $C9_NODE_INSTALL_DIR
curl -fSsl $C9_NODE_URL | tar xz --strip-components=1 -C
"$C9_NODE_INSTALL_DIR"
nvm alias default v18.17.1
nvm use default
echo -e 'nvm use default' >> ~/.bash_profile
```

ステップ 2: コードを追加する

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、という名前でファイルを保存します `hello.js`。(メニューバーでファイルを作成するには、ファイル、New File (新しいファイル)を選択します。ファイルを保存するには、ファイル、保存を選択します。)

```
console.log('Hello, World!');

console.log('The sum of 2 and 3 is 5.');
```

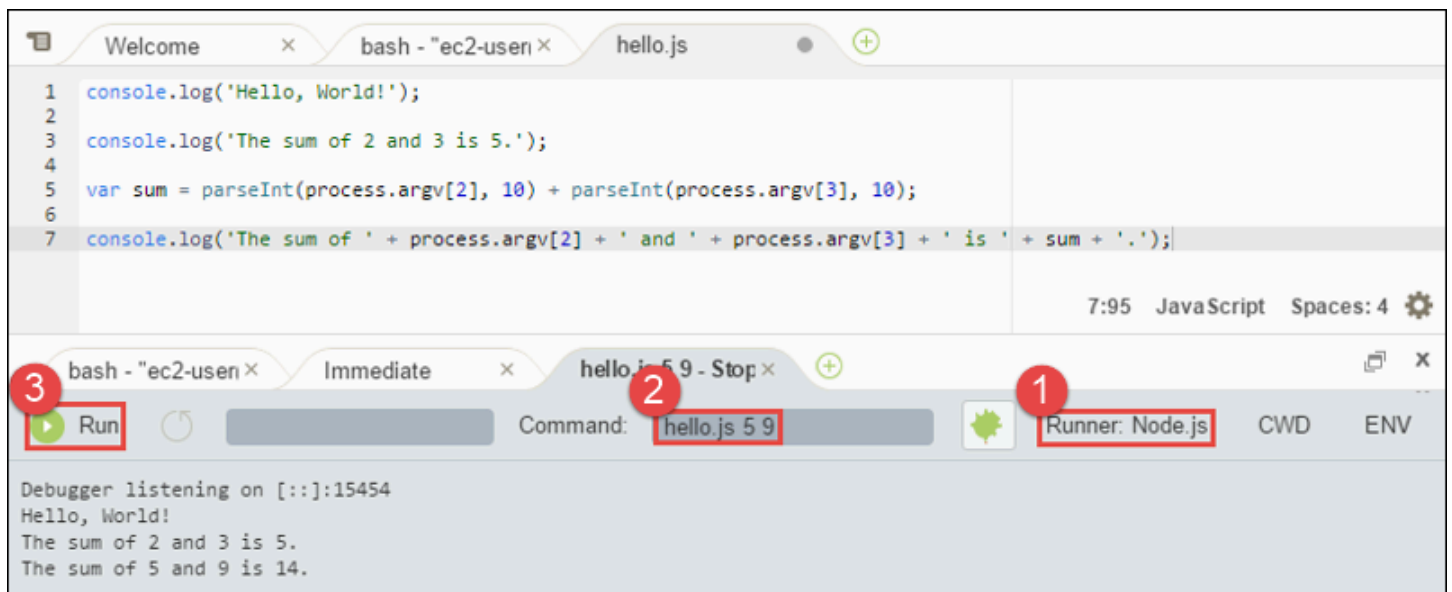
```
var sum = parseInt(process.argv[2], 10) + parseInt(process.argv[3], 10);

console.log('The sum of ' + process.argv[2] + ' and ' +
  process.argv[3] + ' is ' + sum + '.');
```

ステップ 3: コードを実行する

1. AWS Cloud9 IDE のメニューバーで、実行、設定の実行、新しい実行設定を選択します。
2. [[New] - Id] e ([新規] - アイドル) タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Node.js] を選択します。
3. [Command (コマンド)] に「hello.js 5 9」と入力します。このコードで、5 は process.argv[2] を表し、9 は process.argv[3] を表します (process.argv[0] はランタイム (node) の名前、process.argv[1] はファイル (hello.js) の名前です)。
4. [Run (実行)] ボタンを選択して、出力を比較します。

```
Hello, World!
The sum of 2 and 3 is 5.
The sum of 5 and 9 is 14.
```



ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する

で Node.js スクリプトを実行する場合 AWS Cloud9、AWS SDK for JavaScript バージョン 3 (V3) と古い AWS SDK for JavaScript バージョン 2 (V2) のいずれかを選択できます。V2 と同様に V3 を使用すると、Amazon Web Services を簡単に操作できますが、TypeScript で記述されており、モジュール化されたパッケージなど、頻繁にリクエストされる複数の機能が追加されています。

AWS SDK for JavaScript (V3)

このサンプルを強化して、Node.js で AWS SDK for JavaScript を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、作成したばかりのバケットを削除できます。

このステップでは、AWS SDK for JavaScript in Node.js の Amazon S3 サービスクライアントモジュールをインストールして設定しますが、そうすれば、JavaScript コードから、Amazon S3 AWS サービスを簡単に操作できるようになります。

他の AWS サービスを使用する場合は、個別にインストールする必要があります。AWS モジュールのインストールの詳細については、「[AWS デベロッパーガイド \(V3\)](#)」の「[」を参照してください](#)。Node.js と AWS SDK for JavaScript (V3) の使用を開始する方法については、SDK for JavaScript [開発者ガイド \(V3\)](#) の「[Node.js の使用を開始する](#)」を参照してください。AWS JavaScript

Node.js に AWS SDK for JavaScript をインストールしたら、環境で認証情報管理を設定する必要があります。Node.js の AWS SDK for JavaScript では、サービスとやり取り AWS するためにこれらの認証情報が必要です。

Node.js に AWS SDK for JavaScript をインストールするには

npm を使用して **install** コマンドを実行します。

```
npm install @aws-sdk/client-s3
```

詳細については、AWS SDK for JavaScript デベロッパーガイドの「[SDK for JavaScript のインストール](#)」を参照してください。

環境で認証情報管理を設定するには

Node.js で AWS SDK for JavaScript を使用して AWS サービスを呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は AWS SDK for JavaScript in Node.js にその呼び出しを行う適切な許可があるかどうかを判別します。認証情報に適切なアクセス権限がない場合、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[の環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for JavaScript デベロッパーガイドの「[Setting Credentials in Node.js \(Node.js に認証情報を設定\)](#)」を参照してください。

AWS SDK for JavaScript (V2)

このサンプルを強化して、Node.js で AWS SDK for JavaScript を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、作成したばかりのバケットを削除できます。

このステップでは、AWS SDK for JavaScript を Node.js にインストールして設定します。これにより、JavaScript コードから Amazon S3 などの AWS サービスとやり取りする便利な方法が提供されます。Node.js に AWS SDK for JavaScript をインストールしたら、環境で認証情報管理を設定する必要があります。Node.js の AWS SDK for JavaScript では、サービスとやり取り AWS するためにこれらの認証情報が必要です。

Node.js に AWS SDK for JavaScript をインストールするには

npm を使用して **install** コマンドを実行します。

```
npm install aws-sdk
```

詳細については、AWS SDK for JavaScript デベロッパーガイドの「[SDK for JavaScript のインストール](#)」を参照してください。

環境で認証情報管理を設定するには

Node.js で AWS SDK for JavaScript を使用して AWS サービスを呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は AWS SDK for JavaScript in Node.js にその呼び出しを行う適切な許可があるかどうかを判別します。認証情報に適切なアクセス権限がない場合、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[の環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for JavaScript デベロッパーガイドの「[Setting Credentials in Node.js \(Noda.jp に認証情報を設定\)](#)」を参照してください。

ステップ 5: AWS SDK コードを追加する

AWS SDK for JavaScript (V3)

このステップでは、今度は Amazon S3 を操作してバケットを作成し、利用できるバケットのリストを表示した後、作成したバケットを削除するコードをいくつか追加します。このコードは後で実行します。

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、 という名前でファイルを保存します `s3.js`。

```
import {
  CreateBucketCommand,
  DeleteBucketCommand,
  ListBucketsCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const wait = async (milliseconds) => {
  return new Promise((resolve) => setTimeout(resolve, milliseconds));
};

export const main = async () => {
  const client = new S3Client({});
  const now = Date.now();
  const BUCKET_NAME = `easy-bucket-${now.toString()}`;

  const createBucketCommand = new CreateBucketCommand({ Bucket: BUCKET_NAME });
  const listBucketsCommand = new ListBucketsCommand({});
  const deleteBucketCommand = new DeleteBucketCommand({ Bucket: BUCKET_NAME });

  try {
    console.log(`Creating bucket ${BUCKET_NAME}.`);
    await client.send(createBucketCommand);
    console.log(`${BUCKET_NAME} created`);

    await wait(2000);

    console.log(`Here are your buckets:`);
```

```
const { Buckets } = await client.send(listBucketsCommand);
Buckets.forEach((bucket) => {
  console.log(` • ${bucket.Name}`);
});

await wait(2000);

console.log(`Deleting bucket ${BUCKET_NAME}.`);
await client.send(deleteBucketCommand);
console.log(`${BUCKET_NAME} deleted`);
} catch (err) {
  console.error(err);
}
};

main();
```

AWS SDK for JavaScript (V2)

このステップでは、今度は Amazon S3 を操作してバケットを作成し、利用できるバケットのリストを表示した後、作成したバケットを削除するコードをいくつか追加します。このコードは後で実行します。

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、 という名前でファイルを保存します `s3.js`。

```
if (process.argv.length < 4) {
  console.log(
    "Usage: node s3.js <the bucket name> <the AWS Region to use>\n" +
    "Example: node s3.js my-test-bucket us-east-2"
  );
  process.exit(1);
}

var AWS = require("aws-sdk"); // To set the AWS credentials and region.
var async = require("async"); // To call AWS operations asynchronously.

AWS.config.update({
  region: region,
});
```

```
var s3 = new AWS.S3({ apiVersion: "2006-03-01" });
var bucket_name = process.argv[2];
var region = process.argv[3];

var create_bucket_params = {
  Bucket: bucket_name,
  CreateBucketConfiguration: {
    LocationConstraint: region,
  },
};

var delete_bucket_params = { Bucket: bucket_name };

// List all of your available buckets in this AWS Region.
function listMyBuckets(callback) {
  s3.listBuckets(function (err, data) {
    if (err) {
    } else {
      console.log("My buckets now are:\n");

      for (var i = 0; i < data.Buckets.length; i++) {
        console.log(data.Buckets[i].Name);
      }
    }

    callback(err);
  });
}

// Create a bucket in this AWS Region.
function createMyBucket(callback) {
  console.log("\nCreating a bucket named " + bucket_name + "...\\n");

  s3.createBucket(create_bucket_params, function (err, data) {
    if (err) {
      console.log(err.code + ": " + err.message);
    }

    callback(err);
  });
}

// Delete the bucket you just created.
function deleteMyBucket(callback) {
```

```
console.log("\nDeleting the bucket named " + bucket_name + "...\\n");

s3.deleteBucket(delete_bucket_params, function (err, data) {
  if (err) {
    console.log(err.code + ": " + err.message);
  }

  callback(err);
});

// Call the AWS operations in the following order.
async.series([
  listMyBuckets,
  createMyBucket,
  listMyBuckets,
  deleteMyBucket,
  listMyBuckets,
]);
```

ステップ 6: AWS SDK コードを実行する

1. npm を使用して **install** コマンドを実行すると、コードによる Amazon S3 オペレーションの非同期的に呼び出しが有効になります。

```
npm install async
```

2. AWS Cloud9 IDE のメニューバーで、実行、実行設定、新しい実行設定を選択します。
3. [[New] - Id] e ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Node.js] を選択します。
4. AWS SDK for JavaScript (V3) を使用している場合は、コマンドタイプに `s3.js`。AWS SDK for Javascript (v2) を使用している場合、コマンドタイプの場合 `s3.js my-test-bucket us-east-2`、`my-test-bucket`は作成して削除するバケットの名前、`us-east-2`はバケットを作成する AWS リージョンの ID です。他の ID については、[の「Amazon Simple Storage Service \(Amazon S3\)Amazon Web Services 全般のリファレンス」](#)を参照してください。

Note

Amazon S3 バケット名は、アカウント AWS だけでなく AWS、全体で一意である必要があります。

5. [実行] ボタンを選択して、出力を比較します。

```
My buckets now are:  
  
Creating a new bucket named 'my-test-bucket'...  
  
My buckets now are:  
  
my-test-bucket  
  
Deleting the bucket named 'my-test-bucket'...  
  
My buckets now are:
```

ステップ 7: クリーンアップする

このサンプルの使用が完了した後に AWS アカウントで継続的に課金されないようにするには、環境を削除する必要があります。手順については、[での環境の削除 AWS Cloud9](#) を参照してください。

の PHP チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境でいくつかの PHP スクリプトを実行できます。

このチュートリアルに従って、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)

- [ステップ 3: コードを実行する](#)
- [ステップ 4: をインストールして設定する AWS SDK for PHP](#)
- [ステップ 5: AWS SDK コードを追加する](#)
- [ステップ 6: AWS SDK コードを実行する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは PHP をインストールして設定します。このサンプルを実行するために必要なものです。

Note

次の手順では、PHP のみがインストールされます。Apache ウェブサーバーや MySQL データベースなどの関連ツールをインストールするには、Amazon EC2 ユーザーガイドの「[チュートリアル: Amazon Linux への LAMP ウェブサーバーのインストール](#)」を参照してください。

1. AWS Cloud9 IDE のターミナルセッションで、**php --version** コマンドを実行して PHP が既にインストールされているかどうかを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の

順に選択します。) 成功すると、出力に PHP のバージョン番号が表示されます。PHP がインストール済みである場合は、[ステップ 2: コードを追加する](#)に進みます。

2. **yum update** (Amazon Linux 用) または **apt update** (Ubuntu Server 用) コマンドを実行して、最新のセキュリティ更新プログラムおよびバグ修正がインストールされていることを確認します。

Amazon Linux 2 および Amazon Linux:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

3. **install** コマンドを実行して PHP をインストールします。

複数 Amazon Linux 2:

```
sudo amazon-linux-extras install -y php7.2
```

Amazon Linux の場合:

```
sudo yum -y install php72
```

Note

次のコマンドを使用して、Amazon Linux のバージョンを表示できます。

```
cat /etc/system-release
```

Ubuntu Server の場合:

```
sudo apt install -y php php-xml
```

詳細については、PHP ウェブサイトの「[インストールと設定](#)」を参照してください。

ステップ 2: コードを追加する

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、 という名前でファイルを保存します hello.php。(ファイルを作成するには、メニューバーでファイル、New File (新しいファイル) を選択します。 ファイルを保存するには、[ファイル、保存] を選択し、[ファイル名] に hello.php と入力してから [保存] を選択します。)

```
<?php
print('Hello, World!');

print("\nThe sum of 2 and 3 is 5.");

$sum = (int)$argv[1] + (int)$argv[2];

print("\nThe sum of $argv[1] and $argv[2] is $sum.");
?>
```

Note

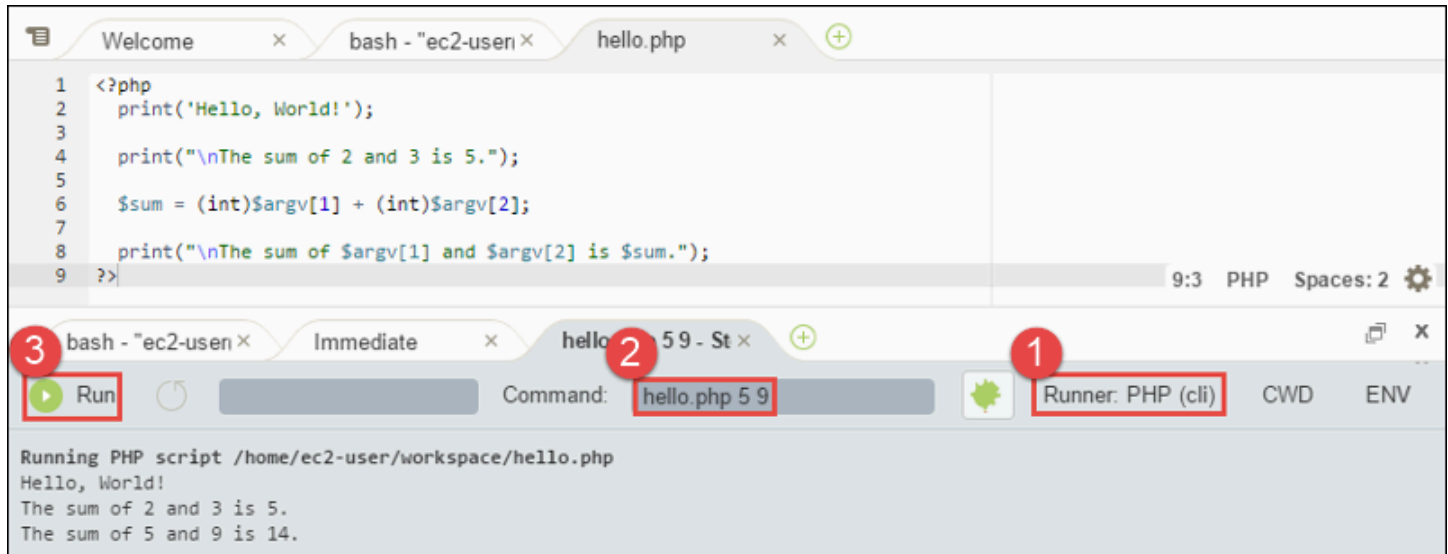
前述のコードは外部ファイルに依存していません。ただし、ファイルに他の PHP ファイルを含めたり要求したりし、それらのファイル AWS Cloud9 を使用して入力時にコード補完を行う場合は、「プロジェクト」、「PHP サポート」、「PHP コード補完設定を有効にする」を有効にしてから、それらのファイルへのパスを「プロジェクト」、「PHP サポート」、「PHP 完了インクルードパス」設定に追加します。(設定を表示および変更するには、メニューバーで、[AWS Cloud9]、[設定] の順に選択します。)

ステップ 3: コードを実行する

1. AWS Cloud9 IDE のメニューバーで、Run、Run Configurations、New Run Configuration を選択します。
2. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[PHP (cli)] を選択します。
3. [Command (コマンド)] に「hello.php 5 9」と入力します。このコードで、5 は \$argv[1] を表し、9 は \$argv[2] を表します (\$argv[0] はファイル名 (hello.php) です。)
4. [Run (実行)] ボタンを選択して、出力を比較します。

```
Hello, World!
```

```
The sum of 2 and 3 is 5.  
The sum of 5 and 9 is 14.
```



ステップ 4: をインストールして設定する AWS SDK for PHP

このサンプルを強化して、AWS SDK for PHP を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、先ほど作成したバケットを削除できます。

このステップでは、PHP コードから Amazon S3 などの AWS サービスとやり取りする便利な方法 AWS SDK for PHP を提供する をインストールして設定します。をインストールする前に AWS SDK for PHP、[Composer](#) をインストールする必要があります。をインストールしたら AWS SDK for PHP、環境で認証情報管理を設定する必要があります。が AWS サービスとやり取りするには、これらの認証情報 AWS SDK for PHP が必要です。

Composer をインストールするには

サイレント () とエラー表示 (**curl**) オプションを指定して **-s -S** コマンドを実行し、Composer インストーラを PHP アーカイブ (PHAR) ファイル (判りやすくするために **composer.phar** という名前にします) にパイピングします。

```
curl -sS https://getcomposer.org/installer | php
```

をインストールするには AWS SDK for PHP

Ubuntu Server では、AWS SDK for PHPをインストールするために Composer に必要な追加のパッケージをインストールします。

```
sudo apt install -y php-xml php-curl
```

Amazon Linux または Ubuntu Server では、php コマンドを使用して Composer インストーラを実行し、AWS SDK for PHPをインストールします。

```
php composer.phar require aws/aws-sdk-php
```

このコマンドは、複数のフォルダとファイルを環境に作成します。最初に使用するファイルは autoload.php です。これは 環境 の vendor フォルダにあります。

Note

インストール後に、Composer から追加の依存関係をインストールすることが勧められる可能性があります。これを行うには、次のようなコマンドを使用して、インストールする依存関係のリストを指定します。たとえば、次のコマンドでは、Composer に次の依存関係のリストをインストールするように指示します。

```
php composer.phar require psr/log ext-curl doctrine/cache aws/aws-php-sns-message-validator
```

詳細については、AWS SDK for PHP デベロッパーガイドの「[インストール](#)」を参照してください。

環境で認証情報管理を設定するには

を使用して AWS サービスを AWS SDK for PHP 呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は、にその呼び出しを行うための適切なアクセス許可 AWS SDK for PHP があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[の環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for PHP デベロッパーガイドにある[ベーシック用法](#)の「クライアントの作成」セクションを参照してください。

ステップ 5: AWS SDK コードを追加する

このステップでは、今度は Amazon S3 を操作してバケットを作成し、利用できるバケットのリストを表示した後、作成したバケットを削除するコードをいくつか追加します。このコードは後で実行します。

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、という名前でファイルを保存します `s3.php`。

```
<?php
require './vendor/autoload.php';

if ($argc < 4) {
    exit("Usage: php s3.php <the time zone> <the bucket name> <the AWS Region to use>
\n" .
        "Example: php s3.php America/Los_Angeles my-test-bucket us-east-2");
}

$timezone = $argv[1];
$bucketName = $argv[2];
$region = $argv[3];

date_default_timezone_set($timezone);

$s3 = new Aws\S3\S3Client([
    'region' => $region,
    'version' => '2006-03-01'
]);

# Lists all of your available buckets in this AWS Region.
function listMyBuckets($s3)
{
    print("\nMy buckets now are:\n");

    $promise = $s3->listBucketsAsync();

    $result = $promise->wait();

    foreach ($result['Buckets'] as $bucket) {
        print("\n");
    }
}
```

```
        print($bucket['Name']);
    }
}

listMyBuckets($s3);

# Create a new bucket.
print("\n\nCreating a new bucket named '$bucketName'...\n");

try {
    $promise = $s3->createBucketAsync([
        'Bucket' => $bucketName,
        'CreateBucketConfiguration' => [
            'LocationConstraint' => $region
        ]
    ]);

    $promise->wait();
} catch (Exception $e) {
    if ($e->getCode() == 'BucketAlreadyExists') {
        exit("\nCannot create the bucket. " .
            "A bucket with the name '$bucketName' already exists. Exiting.");
    }
}

listMyBuckets($s3);

# Delete the bucket you just created.
print("\n\nDeleting the bucket named '$bucketName'...\n");

$promise = $s3->deleteBucketAsync([
    'Bucket' => $bucketName
]);

$promise->wait();

listMyBuckets($s3);

?>
```


ステップ 6: AWS SDK コードを実行する

1. AWS Cloud9 IDE のメニューバーで、Run、Run Configurations、New Run Configuration を選択します。
2. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[PHP (cli)] を選択します。
3. [Command (コマンド)] に「s3.php America/Los_Angeles my-test-bucket us-east-2」と入力します。それぞれ以下を表します。

- America/Los_Angeles はデフォルトのタイムゾーン ID です。その他の ID については、PHP ウェブサイトの「[サポートされるタイムゾーンのリスト](#)」を参照してください。
- my-test-bucket は作成した後削除するバケットの名前です。

Note

Amazon S3 バケット名は、アカウント AWS だけでなく AWS、全体で一意である必要があります。

- us-east-2 は、バケットを作成する AWS リージョンの ID です。他の ID については、[の「Amazon Simple Storage Service \(Amazon S3\)Amazon Web Services 全般のリファレンス」](#)を参照してください。
4. [実行] ボタンを選択して、出力を比較します。

```
My buckets now are:
```

```
Creating a new bucket named 'my-test-bucket'...
```

```
My buckets now are:
```

```
my-test-bucket
```

```
Deleting the bucket named 'my-test-bucket'...
```

```
My buckets now are:
```

ステップ 7: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の PHP ランナーに関する問題のトラブルシューティング AWS Cloud9

PHP CLI ランナーで問題が発生した場合は、ランナーが PHP に設定され、デバッガーモードが有効になっていることを確認する必要があります。

AWS の SDK for Ruby AWS Cloud9

AWS SDK for Ruby AWS Cloud9 で を使用する方法については、[AWS SDK for Ruby AWS Cloud9](#) AWS デベロッパーガイドの SDK for Ruby で を使用するを参照してください。

Note

このチュートリアルに従うと、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

の Go チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境でいくつかの Go コードを実行できます。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを実行する](#)

- [ステップ 4: をインストールして設定する AWS SDK for Go](#)
- [ステップ 5: AWS SDK コードを追加する](#)
- [ステップ 6: AWS SDK コードを実行する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に開いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは Go をインストールして設定します。このサンプルを実行するために必要なものです。

1. AWS Cloud9 IDE のターミナルセッションで、**go version** コマンドを実行して Go が既にインストールされているかどうかを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。) 成功した場合、出力に Go のバージョン番号が含まれています。それ以外の場合は、エラーメッセージが出力されます。Go がインストール済みである場合は、[ステップ 2: コードを追加する](#) に進んでください。
2. (Amazon Linux) 用 **yum update** または (Ubuntu Server) 用 **apt update** コマンドを実行して、最新のセキュリティ更新プログラムおよびバグ修正がインストールされていることを確認します。

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

- Go をインストールするには、以下のコマンドを 1 つずつ実行します。

```
wget https://storage.googleapis.com/golang/go1.9.3.linux-amd64.tar.gz # Download
the Go installer.
sudo tar -C /usr/local -xzf ./go1.9.3.linux-amd64.tar.gz             # Install Go.
rm ./go1.9.3.linux-amd64.tar.gz                                       # Delete the
installer.
```

前述のコマンドは、このトピックが作成された時点の最新の安定バージョンの Go を前提にしています。詳細については、Go プログラミング言語ウェブサイトでの「[Downloads](#)」を参照してください。

- Go バイナリのパスを PATH 環境変数に追加します。次のようになります。
 - 編集のため、シェルスクリプトファイルのファイル (たとえば ~/.bashrc) を開きます。
 - 次のコード行の末尾に、次のように入力します。これにより、コードは次のようになります。

```
PATH=$PATH:/usr/local/go/bin
```

- ファイルを保存します。
- ~/.bashrc ファイルをソース化し、ターミナルが今参照した Go バイナリを見つけられるようにします。

```
. ~/.bashrc
```

- go version** コマンドを実行して、Go が正しくインストールおよび設定されていることを確認します。成功した場合、出力に Go のバージョン番号が含まれています。

ステップ 2: コードを追加する

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、という名前でファイルを保存します `hello.go`。(メニューバーでファイルを作成するには、ファイル、New File (新しいファイル) を選択します。ファイルを保存するには、ファイル、保存を選択します。)

```
package main

import (
    "fmt"
    "os"
    "strconv"
)

func main() {
    fmt.Printf("Hello, World!\n")

    fmt.Printf("The sum of 2 and 3 is 5.\n")

    first, _ := strconv.Atoi(os.Args[1])
    second, _ := strconv.Atoi(os.Args[2])
    sum := first + second

    fmt.Printf("The sum of %s and %s is %s.",
        os.Args[1], os.Args[2], strconv.Itoa(sum))
}
```

ステップ 3: コードを実行する

1. AWS Cloud9 IDE のメニューバーで、Run、Run Configurations、New Run Configuration を選択します。
2. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Go] を選択します。

Note

[Go] が利用できない場合は、Go 用のカスタムランナーを作成できます。

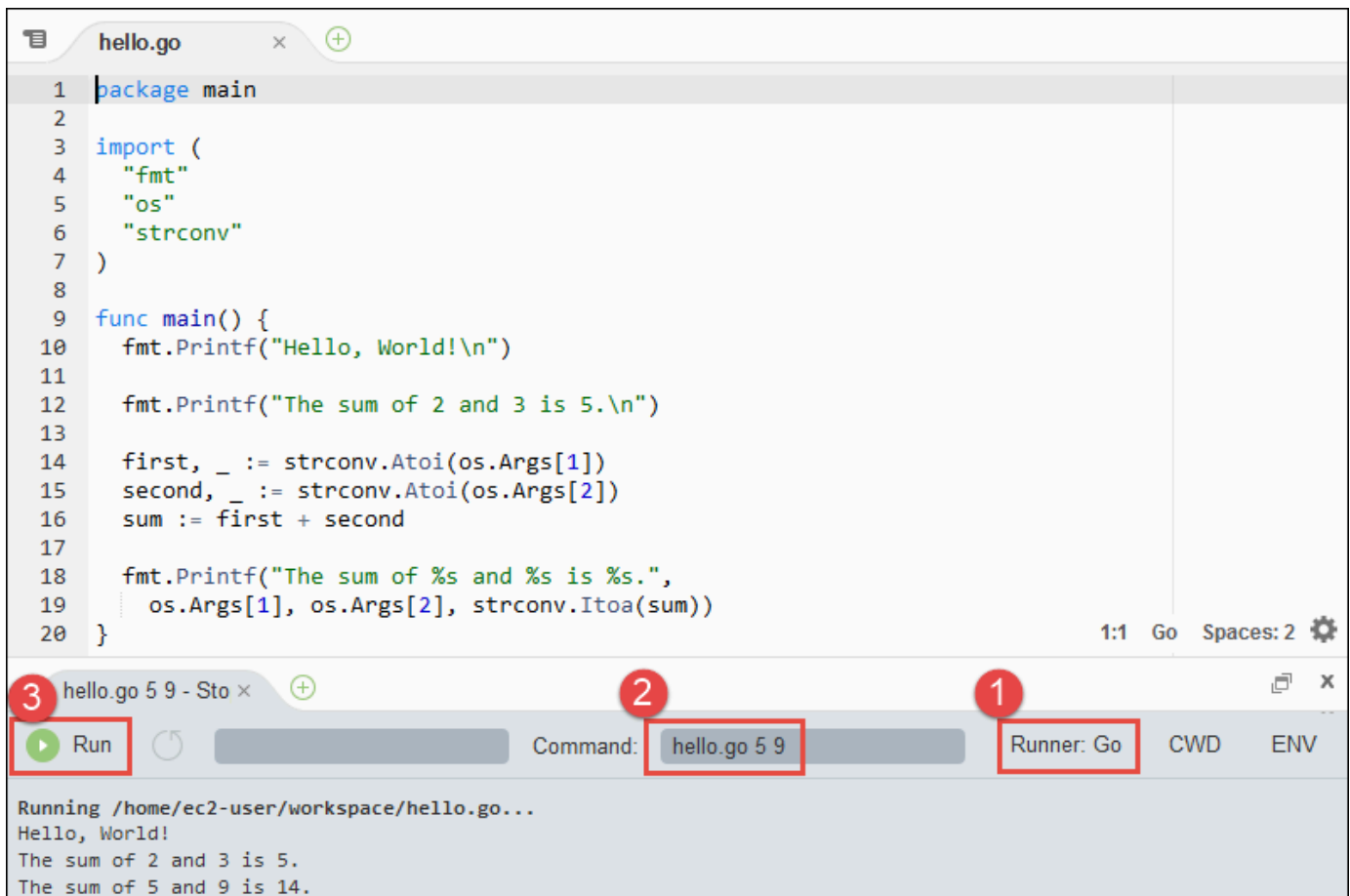
1. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[New Runner (新しいランナー)] を選択します。
2. [My Runner.run] タブで、タブの内容を以下のコードに置き換えます。

```
{
    "cmd" : ["go", "run", "$file", "$args"],
    "info" : "Running $project_path$file_name...",
    "selector" : "source.go"
```

}

3. メニューバーで [File (ファイル)]、[Save As (名前を付けて保存)] の順に選択し、ファイルを `Go.run` として `/.c9/runners` フォルダに保存します。
4. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Go] を選択します。
5. [hello.go] タブを選択し、アクティブにします。

3. [Command (コマンド)] に「`hello.go 5 9`」と入力します。コード内の 5 は `os.Args[1]` を、9 は `os.Args[2]` を表します。



4. [実行] ボタンを選択して、出力を比較します。

```

Hello, World!
The sum of 2 and 3 is 5.
The sum of 5 and 9 is 14.

```

ステップ 4: をインストールして設定する AWS SDK for Go

このサンプルを強化して、AWS SDK for Go を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、先ほど作成したバケットを削除できます。

このステップでは、Go コードから Amazon S3 などの AWS サービスとやり取りする便利な方法 AWS SDK for Go を提供する をインストールして設定します。AWS SDK for Go をインストールする前に、GOPATH 環境変数を設定する必要があります。AWS SDK for Go インストールして GOPATH 環境変数を設定したら、環境で認証情報管理をセットアップする必要があります。が サービスとやり取り AWS するには、これらの認証情報 AWS SDK for Go が必要です。

GOPATH 環境変数を設定するには

1. 編集する ~/.bashrc ファイルを開きます。
2. ファイルの最後の行の後に、次のコードを入力します。

```
GOPATH=~/.environment/go  
  
export GOPATH
```

3. ファイルを保存します。
4. ~/.bashrc ファイルをソース化し、ターミナルが今参照した GOPATH 環境変数を見つけられるようにします。

```
. ~/.bashrc
```

5. **echo \$GOPATH** コマンドを実行して、GOPATH 環境変数が正しく設定されたことを確認します。成功した場合、/home/ec2-user/environment/go または /home/ubuntu/environment/go が出力されます。

をインストールするには AWS SDK for Go

AWS SDK for Go ソースの場所を指定して、**go get** コマンドを実行します。

```
go get -u github.com/aws/aws-sdk-go/...
```

Go は、GOPATH 環境変数で指定された場所に AWS SDK for Go ソースをインストールします。これは環境内の go フォルダです。

環境で認証情報管理を設定するには

を使用して AWS サービスを AWS SDK for Go 呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は、にその呼び出しを行うための適切なアクセス許可 AWS SDK for Go があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[の環境 AWS のサービス からの呼び出し AWS Cloud9](#)の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for Go デベロッパーガイドの「[認証情報の指定](#)」を参照してください。

ステップ 5: AWS SDK コードを追加する

このステップでは、今度は Amazon S3 を操作してバケットを作成し、利用できるバケットのリストを表示した後、作成したバケットを削除するコードをいくつか追加します。このコードは後で実行します。

AWS Cloud9 IDE で、このコンテンツを含むファイルを作成し、 という名前でファイルを保存します s3.go。

```
package main

import (
    "fmt"
    "os"

    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func main() {

    if len(os.Args) < 3 {
        fmt.Printf("Usage: go run s3.go <the bucket name> <the AWS Region to use>\n" +
            "Example: go run s3.go my-test-bucket us-east-2\n")
        os.Exit(1)
    }

    sess := session.Must(session.NewSessionWithOptions(session.Options{
        SharedConfigState: session.SharedConfigEnable,
    }))
}
```



```
svc := s3.New(sess, &aws.Config{
    Region: aws.String(os.Args[2]),
})

listMyBuckets(svc)
createMyBucket(svc, os.Args[1], os.Args[2])
listMyBuckets(svc)
deleteMyBucket(svc, os.Args[1])
listMyBuckets(svc)
}

// List all of your available buckets in this AWS Region.
func listMyBuckets(svc *s3.S3) {
    result, err := svc.ListBuckets(nil)

    if err != nil {
        exitErrorf("Unable to list buckets, %v", err)
    }

    fmt.Println("My buckets now are:\n")

    for _, b := range result.Buckets {
        fmt.Printf(aws.StringValue(b.Name) + "\n")
    }

    fmt.Printf("\n")
}

// Create a bucket in this AWS Region.
func createMyBucket(svc *s3.S3, bucketName string, region string) {
    fmt.Printf("\nCreating a new bucket named '" + bucketName + "'...\n\n")

    _, err := svc.CreateBucket(&s3.CreateBucketInput{
        Bucket: aws.String(bucketName),
        CreateBucketConfiguration: &s3.CreateBucketConfiguration{
            LocationConstraint: aws.String(region),
        },
    })

    if err != nil {
        exitErrorf("Unable to create bucket, %v", err)
    }

    // Wait until bucket is created before finishing
```

```

    fmt.Printf("Waiting for bucket %q to be created...\n", bucketName)

    err = svc.WaitUntilBucketExists(&s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
}

// Delete the bucket you just created.
func deleteMyBucket(svc *s3.S3, bucketName string) {
    fmt.Printf("\nDeleting the bucket named '" + bucketName + "'...\n\n")

    _, err := svc.DeleteBucket(&s3.DeleteBucketInput{
        Bucket: aws.String(bucketName),
    })

    if err != nil {
        exitErrorf("Unable to delete bucket, %v", err)
    }

    // Wait until bucket is deleted before finishing
    fmt.Printf("Waiting for bucket %q to be deleted...\n", bucketName)

    err = svc.WaitUntilBucketNotExists(&s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
}

// If there's an error, display it.
func exitErrorf(msg string, args ...interface{}) {
    fmt.Fprintf(os.Stderr, msg+"\n", args...)
    os.Exit(1)
}

```

ステップ 6: AWS SDK コードを実行する

1. AWS Cloud9 IDE のメニューバーで、Run、Run Configurations、New Run Configuration を選択します。
2. [[New] - Idle ([新規] - アイドル)] タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Go] を選択します。
3. コマンドには、「」と入力します。ここで `s3.go YOUR_BUCKET_NAME THE_AWS_REGION`、`YOUR_BUCKET_NAME` は作成して削除するバケットの名前、`THE_AWS_REGION` はバ

ケットを作成する AWS リージョンの ID です。たとえば、米国東部 (オハイオ) リージョンの場合は、us-east-2 を使用します。他の ID については、[の「Amazon Simple Storage Service \(Amazon S3\)Amazon Web Services 全般のリファレンス」](#)を参照してください。

 Note

Amazon S3 バケット名は、アカウント AWS だけでなく AWS、全体で一意である必要があります。

4. [実行] ボタンを選択して、出力を比較します。

```
My buckets now are:

Creating a new bucket named 'my-test-bucket'...

My buckets now are:

my-test-bucket

Deleting the bucket named 'my-test-bucket'...

My buckets now are:
```

ステップ 7: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の TypeScript チュートリアル AWS Cloud9

このチュートリアルでは、AWS Cloud9 開発環境で TypeScript を使用方法を示します。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 や Amazon S3 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 料金表](#)」および「[Amazon S3 料金表](#)」を参照してください。

トピック

- [前提条件](#)

- [ステップ 1: 必要なツールをインストールする](#)
- [ステップ 2: コードを追加する](#)
- [ステップ 3: コードを実行する](#)
- [ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する](#)
- [ステップ 5: AWS SDK コードを追加する](#)
- [ステップ 6: AWS SDK コードを実行する](#)
- [ステップ 7: クリーンアップする](#)

前提条件

このサンプルを使用する前に、設定が次の要件を満たしていることを確認します。

- 既存の AWS Cloud9 EC2 開発環境が必要です。このサンプルは、Amazon Linux または Ubuntu Server を実行する Amazon EC2 インスタンスに接続された EC2 環境が既にあることを前提としています。別のタイプの環境またはオペレーティングシステムがある場合、このサンプルの指示を関連ツールを設定する必要がある場合があります。詳細については、「[での環境の作成 AWS Cloud9](#)」を参照してください。
- 既存の環境の AWS Cloud9 IDE が既に関いている。環境を開くと、はウェブブラウザでその環境の IDE AWS Cloud9 を開きます。詳細については、「[で環境を開く AWS Cloud9](#)」を参照してください。

ステップ 1: 必要なツールをインストールする

このステップでは、Node Package Manager(**npm**) を使用して TypeScript をインストールします。**npm** をインストールするには、Node Version Manager (**nvm**) を使用します。**nvm** がない場合は、このステップでインストールします。

1. IDE のターミナルセッションで、コマンドライン TypeScript AWS Cloud9 コンパイラを **--version** オプションで実行して TypeScript が既にインストールされているかどうかを確認します。(新しいターミナルセッションを開始するには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] の順に選択します。) 成功すると、出力に TypeScript のバージョン番号が表示されます。TypeScript がインストール済みである場合は、[ステップ 2: コードを追加する](#) に進みます。

```
tsc --version
```

2. **--version** オプションを指定して **npm** を実行することにより、**npm** が既にインストールされているかどうかを確認します。成功すると、出力に **npm** のバージョン番号が表示されます。**npm** がインストール済みである場合は、この手順のステップ 10 に進み、**npm** を使用して TypeScript をインストールします。

```
npm --version
```

3. **yum update** (Amazon Linux の場合) または **apt update** (Ubuntu Server の場合) コマンドを実行して、最新のセキュリティ更新プログラムおよびバグ修正がインストールされていることを確認します。

Amazon Linux の場合:

```
sudo yum -y update
```

Ubuntu Server の場合:

```
sudo apt update
```

4. **npm** をインストールするには、まず、次のコマンドを実行して Node Version Manager (**nvm**) をダウンロードします(**nvm** は、Node.js のバージョンをインストールして管理するために役立つシンプルな Bash シェルスクリプトです。詳細は、GitHub ウェブサイトの [Node Version Manager](#) を参照してください)。

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

5. **nvm** の使用を開始するには、ターミナルセッションを閉じてもう一度起動するか、`~/.bashrc` **nvm** をロードするコマンドを含む ファイルを入手してください。

```
. ~/.bashrc
```

6. **--version** オプションを指定して **nvm** を実行することにより、**nvm** がインストールされていることを確認します。

```
nvm --version
```

7. 最新のバージョン 16 の Node.js をインストールするには、**nvm** コマンドを実行します (**npm** は Node.js に含まれています)。

```
nvm install v16
```

8. Node.js がインストール済みであることを確認するために、**--version** オプションを指定してコマンドラインバージョンの Node.js を実行します。

```
node --version
```

9. **--version** オプションを指定して **npm** を実行することにより、**npm** がインストールされていることを確認します。

```
npm --version
```

10. **-g** オプションをで **npm** を実行して、TypeScript ファイルをインストールします。これにより、TypeScript がグローバルパッケージとして環境にインストールされます。

```
npm install -g typescript
```

11. **--version** オプションを指定してコマンドライン TypeScript コンパイラーを実行し、TypeScript がインストール済みであることを確認します。

```
tsc --version
```

ステップ 2: コードを追加する

1. AWS Cloud9 IDE で、という名前のファイルを作成します `hello.ts`。(メニューバーでファイルを作成するには、ファイル、New File (新しいファイル)を選択します。 ファイルを保存するには、ファイル、保存を選択します。)
2. IDE のターミナルで、`hello.ts` ファイルと同じディレクトリから **npm** を実行して `@types/node` ライブラリをインストールします。

```
npm install @types/node
```

これにより、`node_modules/@types/node` ファイルと同じディレクトリに `hello.ts` フォルダが追加されます。この新しいフォルダには、Node.js の型定義が入ります。これらの定義は、この手順の後半で `hello.ts` ファイルに追加する `console.log` プロパティと `process.argv` プロパティで使用するために TypeScript で必要になります。

3. 次のコードを `hello.ts` ファイルに追加します。

```
console.log('Hello, World!');

console.log('The sum of 2 and 3 is 5.');
```

```
const sum: number = parseInt(process.argv[2], 10) + parseInt(process.argv[3], 10);

console.log('The sum of ' + process.argv[2] + ' and ' +
  process.argv[3] + ' is ' + sum + '.');
```

ステップ 3: コードを実行する

1. ターミナルで、`hello.ts` ファイルと同じディレクトリから TypeScript コンパイラーを実行します。含める `hello.ts` ファイルと追加のライブラリを指定します。

```
tsc hello.ts --lib es6
```

TypeScript は、`hello.ts` ファイルと一連の ECMAScript 6 (ES6) ライブラリファイルを使用して、`hello.ts` ファイル内の TypeScript コードを、`hello.js` というファイル内の同等の JavaScript コードに変換します。

2. [Environment (環境)] ウィンドウで、`hello.js` ファイルを開きます。
3. メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
4. [[New] - Id] e ([新規] - アイドル) タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Node.js] を選択します。
5. [Command (コマンド)] に「`hello.js 5 9`」と入力します。このコードで、5 は `process.argv[2]` を表し、9 は `process.argv[3]` を表します (`process.argv[0]` はランタイム (node) の名前、`process.argv[1]` はファイル (`hello.js`) の名前です)。
6. [Run (実行)] を選択して、出力を比較します。完了したら、[Stop (停止)] を選択します。

```
Hello, World!
The sum of 2 and 3 is 5.
The sum of 5 and 9 is 14.
```



Note

IDE で新しい実行構成を作成する代わりに、ターミナルからコマンド **node hello.js 5 9** を実行してこのコードを実行することもできます。

ステップ 4: Node.js で AWS SDK for JavaScript をインストールして設定する

このサンプルを強化して、Node.js で AWS SDK for JavaScript を使用して Amazon S3 バケットを作成し、使用可能なバケットを一覧表示してから、先ほど作成したバケットを削除できます。

このステップでは、AWS SDK for JavaScript を Node.js にインストールして設定します。SDK は、JavaScript コードから Amazon S3 などの AWS サービスとやり取りする便利な方法を提供します。Node.js に AWS SDK for JavaScript をインストールしたら、環境で認証情報管理を設定する必要があります。SDK が AWS サービスとやり取りするには、これらの認証情報が必要です。

Node.js に AWS SDK for JavaScript をインストールするには

AWS Cloud9 IDE のターミナルセッションで、の `hello.js` ファイルと同じディレクトリから [ステップ 3: コードを実行する](#)、`npm` を実行して AWS SDK for JavaScript を Node.js にインストールします。

```
npm install aws-sdk
```


このコマンドは、[ステップ 3: コードを実行する](#) から `node_modules` フォルダにいくつかのフォルダを追加します。これらのフォルダには、Node.js の AWS SDK for JavaScript のソースコードと依存関係が含まれています。詳細については、AWS SDK for JavaScript デベロッパーガイドの「[SDK for JavaScript のインストール](#)」を参照してください。

環境で認証情報管理を設定するには

Node.js で AWS SDK for JavaScript を使用して AWS サービスを呼び出すたびに、呼び出しで一連の認証情報を指定する必要があります。これらの認証情報は、Node.js の AWS SDK for JavaScript にその呼び出しを行うための適切なアクセス許可があるかどうかを決定します。認証情報に適切なアクセス権限がない場合は、呼び出しは失敗します。

このステップでは、環境内に認証情報を保存します。これを行うには、[環境 AWS のサービスからの呼び出し AWS Cloud9](#) の手順を実行してから、このトピックに戻ります。

詳細については、AWS SDK for JavaScript デベロッパーガイドの「[Setting Credentials in Node.js \(Noda.jp に認証情報を設定\)](#)」を参照してください。

ステップ 5: AWS SDK コードを追加する

このステップでは、今度は Amazon S3 を操作してバケットを作成し、利用できるバケットのリストを表示した後、作成したバケットを削除するコードをいくつか追加します。このコードは後で実行します。

1. AWS Cloud9 IDE で、前のステップの `hello.js` ファイルと同じディレクトリに、という名前のファイルを作成します `s3.ts`。
2. AWS Cloud9 IDE のターミナルから、`s3.ts` ファイルと同じディレクトリで、コードを 2 **npm**回実行して TypeScript 用の非同期ライブラリをインストールし、JavaScript 用の非同期ライブラリを再度インストールすることで、Amazon S3 オペレーションを非同期的に呼び出すことができます。

```
npm install @types/async # For TypeScript.
npm install async        # For JavaScript.
```

3. 次のコードを `s3.ts` ファイルに追加します。

```
import * as async from 'async';
import * as AWS from 'aws-sdk';

if (process.argv.length < 4) {
```

```
console.log('Usage: node s3.js <the bucket name> <the AWS Region to use>\n' +
  'Example: node s3.js my-test-bucket us-east-2');
process.exit(1);
}

const AWS = require('aws-sdk'); // To set the AWS credentials and AWS Region.
const async = require('async'); // To call AWS operations asynchronously.

const s3: AWS.S3 = new AWS.S3({apiVersion: '2006-03-01'});
const bucket_name: string = process.argv[2];
const region: string = process.argv[3];

AWS.config.update({
  region: region
});

const create_bucket_params: any = {
  Bucket: bucket_name,
  CreateBucketConfiguration: {
    LocationConstraint: region
  }
};

const delete_bucket_params: any = {
  Bucket: bucket_name
};

// List all of your available buckets in this AWS Region.
function listMyBuckets(callback): void {
  s3.listBuckets(function(err, data) {
    if (err) {

    } else {
      console.log("My buckets now are:\n");

      for (let i: number = 0; i < data.Buckets.length; i++) {
        console.log(data.Buckets[i].Name);
      }
    }

    callback(err);
  });
}
```

```
// Create a bucket in this AWS Region.
function createMyBucket(callback): void {
    console.log("\nCreating a bucket named '" + bucket_name + "'...\n");

    s3.createBucket(create_bucket_params, function(err, data) {
        if (err) {
            console.log(err.code + ": " + err.message);
        }

        callback(err);
    });
}

// Delete the bucket you just created.
function deleteMyBucket(callback): void {
    console.log("\nDeleting the bucket named '" + bucket_name + "'...\n");

    s3.deleteBucket(delete_bucket_params, function(err, data) {
        if (err) {
            console.log(err.code + ": " + err.message);
        }

        callback(err);
    });
}

// Call the AWS operations in the following order.
async.series([
    listMyBuckets,
    createMyBucket,
    listMyBuckets,
    deleteMyBucket,
    listMyBuckets
]);
```

ステップ 6: AWS SDK コードを実行する

1. ターミナルで、s3.ts ファイルと同じディレクトリから TypeScript コンパイラーを実行します。含める s3.ts ファイルと追加のライブラリを指定します。

```
tsc s3.ts --lib es6
```

TypeScript は、`s3.ts` ファイル、Node.js の AWS SDK for JavaScript、非同期ライブラリ、および一連の ECMAScript 6 (ES6) ライブラリファイルを使用して、`s3.ts` ファイル内の TypeScript コードを という名前のファイル内の同等の JavaScript コードに変換します `s3.js`。

2. [Environment (環境)] ウィンドウで、`s3.js` ファイルを開きます。
3. メニューバーで、[Run (実行)]、[Run Configurations (実行設定)]、[New Run Configuration (新しい実行設定)] の順に選択します。
4. [[New] - Id] e ([新規] - アイドル) タブで、[Runner: Auto (ランナー: 自動)] を選択し、[Node.js] を選択します。
5. コマンドには、「」と入力します。ここで `s3.js YOUR_BUCKET_NAME THE_AWS_REGION`、`YOUR_BUCKET_NAME` は作成して削除するバケットの名前、`THE_AWS_REGION` はバケットを作成する AWS リージョンの ID です。たとえば、米国東部 (オハイオ) リージョンの場合は、`us-east-2` を使用します。他の ID については、[の「Amazon Simple Storage Service \(Amazon S3\)Amazon Web Services 全般のリファレンス」](#)を参照してください。

 Note

Amazon S3 バケット名は、アカウント AWS だけでなく AWS、全体で一意である必要があります。

6. [Run (実行)] を選択して、出力を比較します。完了したら、[Stop (停止)] を選択します。

```
My buckets now are:

Creating a new bucket named 'my-test-bucket'...

My buckets now are:

my-test-bucket

Deleting the bucket named 'my-test-bucket'...

My buckets now are:
```

ステップ 7: クリーンアップする

このサンプルの使用が完了したら、AWS アカウントへの継続的な課金を防ぐには、環境を削除する必要があります。手順については、「[での環境の削除 AWS Cloud9](#)」を参照してください。

の Docker チュートリアル AWS Cloud9

このチュートリアルでは、Amazon EC2 AWS Cloud9 の Amazon Linux インスタンス内で実行中の Docker コンテナに SSH 開発環境を接続する方法を示します。これにより、IDE AWS Cloud9 を使用して Docker コンテナ内のコードとファイルを操作し、そのコンテナでコマンドを実行できます。Docker の詳細については、Docker ウェブサイトの「[Docker とは](#)」を参照してください。

このチュートリアルに従い、このサンプルを作成すると、AWS アカウントに料金が発生する可能性があります。Amazon EC2 などのサービスに対して発生する可能性がある料金も含まれます。詳細については、「[Amazon EC2 の料金](#)」を参照してください。

トピック

- [前提条件](#)
- [ステップ 1: Docker のインストールと実行](#)
- [ステップ 2: イメージの構築](#)
- [ステップ 3: コンテナの実行](#)
- [ステップ 4: 環境の作成](#)
- [ステップ 5: コードの実行](#)
- [ステップ 6: クリーンアップする](#)

前提条件

- Amazon Linux または Ubuntu Server を実行している Amazon EC2 インスタンスを用意する必要があります。このサンプルは、AWS アカウントに Amazon Linux または Ubuntu Server を実行している Amazon EC2 インスタンスが既にあることを前提としています。Amazon EC2 インスタンスを起動するには、「[Linux 仮想マシンの起動](#)」を参照してください。ウィザードの「Amazon マシンイメージ (AMI) の選択」ページで、表示名が Amazon Linux AMI または Ubuntu Server で始まる AMI を選択します。
- Amazon EC2 インスタンスを Amazon VPC 内で実行する場合は、追加の要件があります。「[AWS Cloud9 開発環境の VPC 設定](#)」を参照してください。
- Amazon EC2 インスタンスには、少なくとも 8~16 GB の利用可能な空きディスク容量が必要です。このサンプルでは、3 GB 以上のサイズの Docker イメージを使用し、イメージ構築のためにさらに 3 GB 以上のディスク空き容量を使用することがあります。8 GB 以下の空き容量のディスクでこのサンプルの実行を試行すると、Docker イメージが構築されない、あるいは Docker コン

テナが実行されない結果となる場合があります。インスタンスの空きディスク容量を確認するには、インスタンスで **df -h** (「人間が読める形式のディスクファイルシステム情報」) などのコマンドを実行します。既存のインスタンスのディスクサイズを増やすには、Amazon EC2 ユーザーガイドの「[ボリュームの変更](#)」を参照します。

ステップ 1: Docker のインストールと実行

このステップでは、Docker が Amazon EC2 インスタンスにインストールされているかをチェックし、まだインストールされていない場合は Docker をインストールします。Docker をインストールしたら、インスタンスでこれを実行します。

1. **ssh** ユーティリティまたは PuTTY など、SSH クライアントを使って、実行中の Amazon EC2 インスタンスに接続します。これを行うには、「[Linux 仮想マシンを起動する](#)」の「ステップ 3: インスタンスに接続する」を参照してください。
2. Docker がインスタンスにインストールされているかを確認します。これを行うには、**docker** オプションを使用して **--version** コマンドをインスタンスで実行します。

```
docker --version
```

Docker がインストールされている場合、Docker バージョンおよびビルド番号が表示されます。この場合、この手順の先のステップ 5 に進みます。

3. Docker をインストールします。これを行うには、**yum** または **apt** コマンドを **install** アクションを指定して実行し、インストールする **docker** または **docker.io** パッケージを指定します。

Amazon Linux の場合:

```
sudo yum install -y docker
```

Ubuntu Server の場合:

```
sudo apt install -y docker.io
```

4. Docker がインストールされていることを確認します。これを行うには、**docker --version** コマンドを再度実行します。Docker バージョンおよびビルド番号が表示されます。
5. Docker を実行します。これを行うには、**docker** サービスおよび **start** アクションを指定して、**service** コマンドを実行します。

```
sudo service docker start
```

6. Docker が実行されていることを確認します。これを行うには、**docker** アクションで **info** コマンドを実行します。

```
sudo docker info
```

Docker が実行されている場合、Docker についての情報が表示されます。

ステップ 2: イメージの構築

このステップでは、Dockerfile を使用してインスタンスに Docker イメージを構築します。このサンプルでは、Node.js が含まれるイメージおよびサンプルチャートサーバーアプリケーションを使用します。

1. インスタンスで Dockerfile を作成します。これを行うには、SSH クライアントをインスタンスに接続した状態で、インスタンスの /tmp ディレクトリ内に Dockerfile という名前のファイルを作成します。例えば、次のように **touch** コマンドを実行します。

```
sudo touch /tmp/Dockerfile
```

2. Dockerfile ファイルに次の内容を追加します。

```
# Build a Docker image based on the Amazon Linux 2 Docker image.
FROM amazonlinux:2

# install common tools
RUN yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
RUN yum update -y
RUN yum install -y sudo bash curl wget git man-db nano vim bash-completion tmux
gcc gcc-c++ make tar

# Enable the Docker container to communicate with AWS Cloud9 by
# installing SSH.
RUN yum install -y openssh-server

# Ensure that Node.js is installed.
```

```
RUN yum install -y nodejs

# Create user and enable root access
RUN useradd --uid 1000 --shell /bin/bash -m --home-dir /home/ubuntu ubuntu && \
    sed -i 's/%wheel\s.*/%wheel ALL=NOPASSWD:ALL/' /etc/sudoers && \
    usermod -a -G wheel ubuntu

# Add the AWS Cloud9 SSH public key to the Docker container.
# This assumes a file named authorized_keys containing the
# AWS Cloud9 SSH public key already exists in the same
# directory as the Dockerfile.
RUN mkdir -p /home/ubuntu/.ssh
ADD ./authorized_keys /home/ubuntu/.ssh/authorized_keys
RUN chown -R ubuntu /home/ubuntu/.ssh /home/ubuntu/.ssh/authorized_keys && \
    chmod 700 /home/ubuntu/.ssh && \
    chmod 600 /home/ubuntu/.ssh/authorized_keys

# Update the password to a random one for the user ubuntu.
RUN echo "ubuntu:$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 32 | head -n 1)" \
    | chpasswd

# pre-install Cloud9 dependencies
USER ubuntu
RUN curl https://d2j6vhu5uywtq3.cloudfront.net/static/c9-install.sh | bash

USER root
# Start SSH in the Docker container.
CMD ssh-keygen -A && /usr/sbin/sshd -D
```

上のコンテンツを Dockerfile ファイルに追加するには、次のようにインスタンスで **vi** ユーティリティを使用できます。

- a. を使用して /tmp/Dockerfile、ファイル AWS Cloud9 を開いて編集します。

```
sudo vi /tmp/Dockerfile
```

- b. Dockerfile ファイルに前述のコンテンツを貼り付けます。方法がわからない場合は、SSH クライアントのドキュメントを参照してください。
- c. コマンドモードに切り替えます。これを行うには Esc キーを押します (ウィンドウの下部から -- INSERT -- が消えます)。

- d. `:wq` と入力します (/tmp/Dockerfile ファイルに書き込み、このファイルを保存し、`vi` を終了します)。次に、Enter を押します。

 Note

Docker イメージの頻繁に更新されるリストには、 からアクセスできます AWS CodeBuild。詳細については、AWS CodeBuild ユーザーガイドで「[CodeBuild が提供する Docker イメージ](#)」を参照してください。

3. インスタンスで、Docker コンテナが使用する AWS Cloud9 SSH パブリックキーを含むファイルを作成します。これを行うには、`touch` コマンドを実行して、Dockerfile ファイルと同じディレクトリ内に `authorized_keys` のような名前のファイルを作成します。

```
sudo touch /tmp/authorized_keys
```

4. SSH AWS Cloud9 パブリックキーを `authorized_keys` ファイルに追加します。SSH AWS Cloud9 パブリックキーを取得するには、次の手順を実行します。
 - a. <https://console.aws.amazon.com/cloud9/> で AWS Cloud9 コンソールを開きます。
 - b. AWS ナビゲーションバーの AWS リージョンセクターで、このトピックの後半で AWS Cloud9 開発環境を作成する AWS リージョンを選択します。
 - c. ようこそページが表示された場合は、新しい AWS Cloud9 環境 で環境の作成 を選択します。それ以外の場合は、[Create environment (環境の作成)] を選択します。
 - d. [Name environment (環境に名前を付ける)] ページで、[名前] には、環境の名前を入力します。(ここでは、名前は重要ではありません。後で別の名前を選択できます。)
 - e. [Next step] を選択します。
 - f. [Environment type (環境タイプ)] で [Connect and run in remote server (SSH) (リモートサーバーに接続して実行する)] を選択します。
 - g. [View public SSH key (パブリック SSH キーを表示)] を展開します。
 - h. [Copy key to clipboard (キーをクリップボードにコピー)] を選択します (これは [View public SSH key (パブリック SSH キーの表示)] と [Advanced settings (詳細設定)] の間になります)。
 - i. [キャンセル] を選択します。

- j. クリップボードのコンテンツを `authorized_keys` ファイルに貼り付け、このファイルを保存します。例えば、このステップで前に説明したように、**vi** ユーティリティを使用できます。
5. **docker** コマンドを **build** アクションを指定して実行し、タグ `cloud9-image:latest` をイメージに追加し、使用する Dockerfile ファイルへのパスを指定してイメージをビルドします。

```
sudo docker build -t cloud9-image:latest /tmp
```

成功すると、構築出力の最後の 2 行には `Successfully built` および `Successfully tagged` が表示されます。

Docker でイメージが正しく構築されたことを確かめるには、**docker** コマンドに `image ls` アクションを指定して実行します。

```
sudo docker image ls
```

成功した場合には、出力の `REPOSITORY` フィールドが `cloud9-image`、`TAG` フィールドが `latest` に設定されたエントリが表示されます。

6. Amazon EC2 インスタンスのパブリック IP アドレスをメモしておきます。これは [ステップ 4: 環境の作成](#) で必要になります。インスタンスのパブリック IP アドレスがわからない場合は、インスタンスで次のコマンドを実行してそれを取得します。

```
curl http://169.254.169.254/latest/meta-data/public-ipv4
```

ステップ 3: コンテナの実行

このステップでは、インスタンスで Docker コンテナを実行します。このコンテナは前のステップで構築したイメージに基づいています。

1. Docker コンテナを実行するには、インスタンスで **run** アクションと以下のオプションを指定して **docker** コマンドを実行します。

```
sudo docker run -d -it --expose 9090 -p 0.0.0.0:9090:22 --name cloud9 cloud9-image:latest
```

- `-d` は、コンテナを実行していたルートプロセス (このサンプルでは、SSH クライアント) を終了して、デタッチ済みモードでコンテナを実行します。
- `-it` は、割り当てられた疑似 TTY でコンテナを実行し、コンテナがアタッチされていない場合でも STDIN をオープンのままにします。
- `--expose` は、コンテナに対して特定のポート (このサンプルでは、ポート 9090) を利用可能にします。
- `-p` は、指定された IP アドレスとポートを介して特定のポートを Amazon EC2 インスタンスの内部で利用できるようにします。このサンプルでは、コンテナのポート 9090 は、Amazon EC2 インスタンスのポート 22 を介して内部からアクセスできます。
- `--name` は、このコンテナの人間が読み取れる名前です (このサンプルでは、cloud9)。
- `cloud9-image:latest` は、コンテナを実行するために使用するビルドイメージの人間が読み取れる名前です。

Docker がコンテナを正しく実行していることを確かめるには、**docker** コマンドに `container ls` アクションを指定して実行します。

```
sudo docker container ls
```

成功した場合には、出力の `IMAGE` フィールドが `cloud9-image:latest`、`NAMES` フィールドが `cloud9` に設定されたエントリが表示されます。

2. 実行中のコンテナにログインします。これを行うには、**exec** アクションと以下のオプションを指定して **docker** コマンドを実行します。

```
sudo docker exec -it cloud9 bash
```

- `-it` は、割り当てられた疑似 TTY でコンテナを実行し、コンテナがアタッチされていない場合でも STDIN をオープンのままにします。
- `cloud9` は、実行中のコンテナの人間が読み取れる名前です。
- `bash` は、実行中のコンテナで標準のシェルを起動します。

成功した場合、ターミナルプロンプトはログインしたユーザー名をコンテナとコンテナの ID に変更します。

Note

実行中のコンテナからログアウトする場合は、**exit** コマンドを実行します。ターミナルプロンプトはログインしたユーザー名をインスタンスとこのインスタンスのプライベート DNS に再度変更します。コンテナは実行中である必要があります。

- ログイン後に AWS Cloud9 開始する実行中のコンテナのディレクトリについては、そのアクセス許可を **rwxr-xr-x** に設定します。これは、所有者には読み取り/書き込み/実行のアクセス許可、グループには読み取り/実行のアクセス許可、その他には読み取り/実行のアクセス許可があることを意味します。例えば、ディレクトリのパスが `~` である場合、実行中のコンテナで **chmod** コマンドを次のように実行して、ディレクトリにこれらのアクセス許可を設定できます。

```
sudo chmod u=rwx,g=rx,o=rx ~
```

- [ステップ 4: 環境の作成](#) で必要となるため、Node.js バイナリを含む実行中のコンテナのディレクトリのパスをメモしておきます。このパスがわからない場合は、実行中のコンテナで次のコマンドを実行してこれを取得します。

```
which node
```

ステップ 4: 環境の作成

このステップでは、AWS Cloud9 を使用して SSH AWS Cloud9 開発環境を作成し、実行中の Docker コンテナに接続します。が環境 AWS Cloud9 を作成すると、IDE AWS Cloud9 が表示され、コンテナ内のファイルとコードの操作を開始できます。

AWS Cloud9 コンソールを使用して SSH AWS Cloud9 開発環境を作成します。CLI を使用して SSH 環境を作成することはできません。

前提条件

- まず、「[セットアップ AWS Cloud9](#)」のステップを完了していることを確認します。これにより、AWS Cloud9 コンソールにサインインして環境を作成できます。
- 環境 AWS Cloud9 に接続する既存のクラウドコンピューティングインスタンス (の Amazon EC2 インスタンスなど AWS アカウント) または独自のサーバーを特定します。

- 既存のインスタンスまたは独自のサーバーが、すべての [SSH ホスト要件](#) を満たしていることを確認します。これには、特定のバージョンの Python、Node.js、およびその他のコンポーネントをインストールしていること、ログイン後に AWS Cloud9 をスタートしたいディレクトリに具体的な許可を設定していること、関連した Amazon Virtual Private Cloud を設定していることなどが含まれます。

SSH 環境を作成する

1. 前述の前提条件を完了していることを確認してください。
2. SSH クライアントを使用して既存のインスタンスまたは独自のサーバーに接続します (まだ接続していない場合)。これにより、インスタンスまたはサーバーに必要な公開 SSH キー値を追加できます。詳細については、この手順で後ほど説明します。

Note

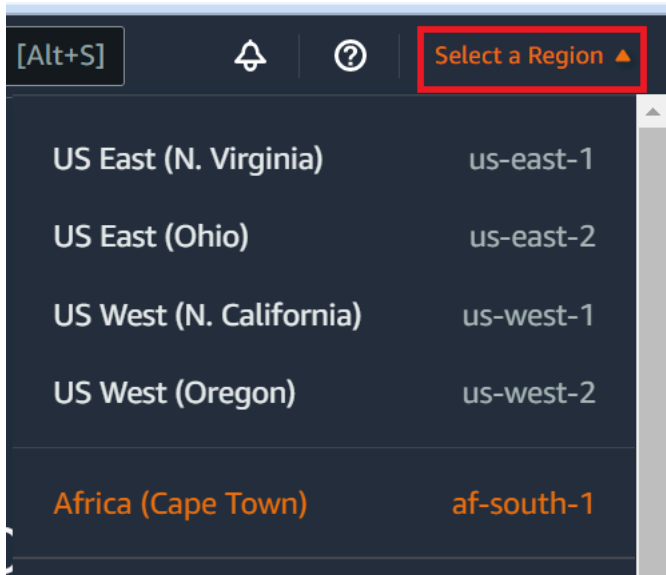
既存の AWS クラウド コンピューティングインスタンスに接続するには、次の 1 つ以上のリソースを参照してください。

- Amazon EC2 については、「Amazon EC2 ユーザーガイド」の「[Linux インスタンスに接続](#)」を参照してください。
- Amazon Lightsail については、Amazon Lightsail ドキュメントの「[Linux/UNIX ベースの Lightsail インスタンスへの接続](#)」を参照してください。
- については AWS Elastic Beanstalk、「AWS Elastic Beanstalk デベロッパーガイド」の「[サーバーインスタンスの一覧表示と接続](#)」を参照してください。
- については AWS OpsWorks、「AWS OpsWorks ユーザーガイド」の「[SSH を使用して Linux インスタンスにログインする](#)」を参照してください。
- その他の方法については AWS のサービス、その特定のサービスのドキュメントを参照してください。

独自のサーバーに接続するには、SSH を使用します。SSH は macOS および Linux オペレーティングシステムに既にインストールされています。Windows で SSH を使用してサーバーに接続するには、[PuTTY](#) をインストールする必要があります。

3. <https://console.aws.amazon.com/cloud9/> で AWS Cloud9 コンソールにサインインします。

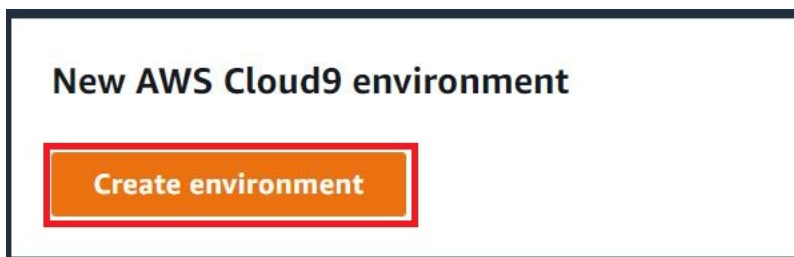
4. AWS Cloud9 コンソールにサインインしたら、上部のナビゲーションバーで AWS リージョンを選択して環境を作成します。利用可能な のリストについては AWS リージョン、 [AWS Cloud9](#) の「」を参照してくださいAWS 全般のリファレンス。



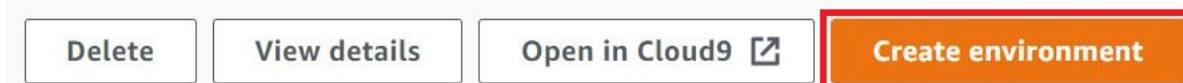
5. 今回、開発環境を初めて作成する場合は、ウェルカムページが表示されます。新しい AWS Cloud9 環境パネルで、環境の作成を選択します。

以前に開発環境を作成したことがある場合は、画面の左側にあるペインを展開することもできます。[Your environments] (自分の環境)、[Create environment] (環境の作成) の順に選択します。

ウェルカムページの場合:



または、[Your environments] (自分の環境) ページの場合:



6. [Create environment] (環境の作成) ページで、環境の名前を入力します。
7. [Description (説明)] に、環境に関する説明を入力します。このチュートリアルでは、This environment is for the AWS Cloud9 tutorial. を使用します。

8. [Environment type] (環境タイプ) で、以下のオプションから [Existing Compute] (既存のコンピューティング) を選択します。
- 新しい EC2 インスタンス – SSH 経由で に直接接続 AWS Cloud9 できる Amazon EC2 インスタンスを起動します。
 - 既存のコンピューティング – 開いているインバウンドポートを必要としない Amazon EC2 インスタンスを起動します。 は を介してインスタンス AWS Cloud9 に接続します [AWS Systems Manager](#)。
 - [Existing compute] (既存のコンピューティング) オプションを選択すると、Systems Manager がユーザーに代わって EC2 インスタンスとやり取りすることを許可するためのサービスロールと IAM インスタンスプロファイルが作成されます。両方の名前は、インターフェイスの下方にある [Systems Manager アクセス用のサービスロールとインスタンスプロファイル] セクションで確認できます。詳細については、「[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)」を参照してください。

 Warning

環境に EC2 インスタンスを作成すると、Amazon EC2 AWS アカウント の に料金が発生する可能性があります。Systems Manager を使用して EC2 インスタンスへの接続を管理する場合、追加コストはかかりません。

 Warning

AWS Cloud9 は SSH パブリックキーを使用してサーバーに安全に接続します。安全な接続を確立するには、以下の手順に従って公開キーを `~/.ssh/authorized_keys` ファイルに追加し、ログイン認証情報を入力します。[Copy key to clipboard] (キーをクリップボードにコピー) を選択して SSH キーをコピーするか、[View public SSH key] (SSH 公開キーを表示) を選択してキーを表示します。

9. [Existing compute] (既存のコンピューティング) パネルの [User] (ユーザー) に、この手順で先にインスタンスまたはサーバーに接続するときに使用したログイン名を入力します。例えば、AWS クラウド コンピューティングインスタンスの場合は、`ec2-user`、`ubuntu`、または `root` を入力します。

 Note

ログイン名をインスタンスやサーバーの管理者アクセス許可または管理者ユーザーに関連付けることをお勧めします。より具体的には、このログイン名がインスタンスやサーバーの Node.js インストールの所有者であることが推奨されます。これをチェックするには、インスタンスやサーバーのターミナルから、**ls -l \$(which node)** (または **nvm** を使用している場合は **ls -l \$(nvm which node)**) コマンドを実行します。このコマンドは、Node.js インストールの所有者名を表示します。また、インストールのアクセス許可、グループ名、場所も表示されます。

10. [Host] (ホスト) に、インスタンスまたはサーバーのパブリック IP アドレス (優先) またはホスト名を入力します。
11. ポートには、インスタンスまたはサーバーへの接続を試みる AWS Cloud9 ポートを入力します。または、デフォルトのポートをそのまま使用します。
12. [Additional details - optional] (その他の詳細 - オプション) を選択し、環境パス、node.js バイナリへのパス、SSH ジャンプホスト情報を表示します。
13. 環境パスには、AWS Cloud9 開始するインスタンスまたはサーバーのディレクトリへのパスを入力します。この手順の前提条件で、以前、これを確認しました。空白のままにすると、AWS Cloud9 はログイン後に通常インスタンスまたはサーバーを起動するディレクトリを使用します。これは通常、ホームまたはデフォルトのディレクトリです。
14. [Path to Node.js binary] (Node.js バイナリへのパス) にパス情報を入力し、インスタンスまたはサーバーの Node.js バイナリへのパスを指定します。パスを取得するには、インスタンスまたはサーバーでコマンド **which node** (**nvm** を使用している場合は **nvm which node**) を実行できます。例えば、パスは `/usr/bin/node` のようになります。空白のままにした場合、AWS Cloud9 は接続の試行時に Node.js バイナリの場所の推測を試みます。
15. [SSH jump host] (SSH ジャンプホスト) に、インスタンスまたはサーバーが使用するジャンプホストに関する情報を入力します。USER_NAME@HOSTNAME:PORT_NUMBER 形式を使用します (例: `ec2-user@ip-192-0-2-0:22`)。

ジャンプホストは、以下の要件を満たしている必要があります。

- SSH を使用してパブリックインターネット経由で到達可能とする必要があります。
- 特定のポートを経由したすべての IP アドレスからのインバウンドアクセスを許可する必要があります。

- 既存のインスタンスまたはサーバーの ~/.ssh/authorized_keys ファイルにコピーしたパブリック SSH キー値を、ジャンプホストの ~/.ssh/authorized_keys ファイルにもコピーする必要があります。
 - Netcat をインストールする必要があります。
16. 最大 50 個のタグを追加します。タグごとにキーと値を指定します。これを行うには、[Add new tag] (新しいタグを追加) を選択します。タグはリソースタグとして AWS Cloud9 環境にアタッチされ、CloudFormation スタック、Amazon EC2 インスタンス、Amazon EC2 セキュリティグループの基盤となるリソースに伝達されます。タグの詳細については、[IAM ユーザーガイド](#)の[AWS「リソースタグを使用したアクセスの制御」](#)およびこのガイドのタグに関する詳細[情報](#)を参照してください。

 Warning

これらのタグを作成後に更新した場合、変更は基になるリソースには反映されません。詳細については、[タグに関する詳細情報の「基礎となるリソースへのタグ更新の伝播」](#)を参照してください。

17. [Create] (作成) を選択して環境を作成すると、ホームページにリダイレクトされます。アカウントが正常に作成されると、AWS Cloud9 コンソールの上部に緑色のフラッシュバーが表示されます。新しい環境を選択し、[Open in Cloud9] (Cloud9 で開く) を選択して IDE を起動できます。

Delete

View details

 Open in Cloud9 

Create environment

アカウントの作成に失敗すると、AWS Cloud9 コンソールの上部に赤い点滅バーが表示されます。ウェブブラウザ、AWS アクセス許可、インスタンス、または関連するネットワークに問題があるため、アカウントの作成に失敗することがあります。アカウントが失敗する原因と考えられる問題の可能な解決方法については、「[AWS Cloud9 のトラブルシューティング](#)」セクションを参照してください。

 Note

環境でプロキシを使用してインターネットにアクセスしている場合は、依存関係をインストール AWS Cloud9 できるようにプロキシの詳細を に提供する必要があります。詳細については、「[依存関係をインストールできませんでした](#)」を参照してください。

ステップ 5: コードの実行

このステップでは、IDE AWS Cloud9 を使用して、実行中の Docker コンテナ内でサンプルアプリケーションを実行します。

1. 実行中のコンテナの IDE AWS Cloud9 が表示されたら、サンプルチャットサーバーを起動します。これを行うには、[Environment (環境)] ウィンドウで、サンプル workspace/server.js ファイルを右クリックし、[Run (実行)] を選択します。
2. サンプルアプリケーションをプレビューします。これを行うには、[Environment (環境)] ウィンドウで workspace/client/index.html ファイルを開きます。次に、メニューバーで [Tools (ツール)、[Preview (プレビュー)]、[Preview Running Application (実行中のアプリケーションのプレビュー)] の順に選択します。
3. アプリケーションのプレビュータブで、[Your Name (あなたの名前)] に名前を入力します。[Message (メッセージ)] にメッセージを入力します。次に、[Send (送信)] を選択します。チャットサーバーは、ユーザーの名前とメッセージをリストに追加します。

ステップ 6: クリーンアップする

このステップでは、環境を削除し、Amazon EC2 インスタンスから AWS Cloud9 および Docker サポートファイルを削除します。また、このサンプルの使用が終了した後に AWS アカウントへの継続的な料金が発生しないようにするには、Docker を実行している Amazon EC2 インスタンスを終了する必要があります。

ステップ 6.1: 環境を削除する

環境を削除するには、[での環境の削除 AWS Cloud9](#) を参照します。

ステップ 6.2: コンテナから AWS Cloud9 サポートファイルを削除する

環境を削除した後も、一部の AWS Cloud9 サポートファイルはコンテナに残ります。コンテナを引き続き使用したいが、これらのサポートファイルが不要になった場合は、ログイン後に開始 AWS Cloud9 するように指定したコンテナのディレクトリから .c9 フォルダを削除します。例えば、ディレクトリが ~ の場合、以下のように **-r** オプションを使用して **rm** コマンドを実行します。

```
sudo rm -r ~/.c9
```

ステップ 6.3: インスタンスから Docker サポートファイルを削除する

Docker コンテナ、Docker イメージ、および Amazon EC2 インスタンス上の Docker を保持したいと思わなくなりましたが、インスタンスは維持したい場合には、次のように Docker サポートファイルを削除できます。

1. インスタンスから Docker コンテナを削除します。これを行うには、インスタンスで **docker** コマンドを **stop** と **rm** の停止アクション、および人間が読み取れるコンテナの名前を指定して実行します。

```
sudo docker stop cloud9
sudo docker rm cloud9
```

2. インスタンスから Docker イメージを削除します。これを行うには、**docker** アクションおよびイメージタグを使用して **image rm** コマンドをインスタンスで実行します。

```
sudo docker image rm cloud9-image:latest
```

3. まだ残っているすべての追加 Docker ファイルを削除します。これを行うには、**docker** アクションを使用して **system prune** コマンドをインスタンスで実行します。

```
sudo docker system prune -a
```

4. Docker をアンインストールします。これを行うには、インスタンスで **yum** コマンドを **remove** アクションを指定して実行し、アンインストールする **docker** パッケージを指定します。

Amazon Linux の場合:

```
sudo yum -y remove docker
```

Ubuntu Server の場合:

```
sudo apt -y remove docker
```

また、前のステップで作成した Dockerfile ファイルおよび `authorized_keys` ファイルを削除することもできます。例えば、インスタンスで **rm** コマンドを実行します。

```
sudo rm /tmp/Dockerfile
sudo rm /tmp/authorized_keys
```

ステップ 6.4: インスタンスの終了

Amazon EC2 インスタンスを終了するには、「Amazon EC2 ユーザーガイド」の「[インスタンスの終了](#)」を参照します。

関連チュートリアル

- [AWS RoboMaker デベロッパーガイドのAWS RoboMaker の開始方法](#)。このチュートリアルでは、AWS Cloud9 を使用してサンプルロボットアプリケーションを変更、構築、バンドルします。

の高度なトピック AWS Cloud9

これらのトピックには、次の情報が含まれています。

- 高度な設定と意思決定に使用される情報。
- 特定のタスクに関連する情報で、をよりよく理解できます AWS Cloud9 が、そのタスクを完了するために重要ではありません。

トピック

- [の SSH 環境と比較した EC2 環境 AWS Cloud9](#)
- [AWS Cloud9 開発環境の VPC 設定](#)
- [SSH 環境ホスト要件](#)
- [SSH AWS Cloud9 環境用の AWS Cloud9 インストーラの使用](#)
- [のインバウンド SSH IP アドレス範囲 AWS Cloud9](#)
- [AWS Cloud9 EC2 開発環境の AMI コンテンツ](#)
- [のサービスにリンクされたロールの使用 AWS Cloud9](#)
- [を使用した AWS Cloud9 API コールのログ記録 AWS CloudTrail](#)
- [タグ](#)

の SSH 環境と比較した EC2 環境 AWS Cloud9

「[環境とコンピューティングリソースの概要](#)」で説明されているように、[環境を使用する際](#)、AWS Cloud9 環境を EC2 または SSH 環境としてセットアップできます。

次の表は、AWS Cloud9における EC2 環境と SSH 環境の使用の類似点と相違点の両方のハイライトを示しています。

EC2 環境	SSH 環境
AWS Cloud9 は、関連付けられた Amazon EC2 インスタンスを作成し、インスタンスのライフサイクルを管理します。これには、開始、停止、および終了オペレーションが含まれます。	既存のクラウドコンピューティングインスタンスまたは独自のサーバーを使用します。ユーザーにはライフサイクルを管理する責任があります。

EC2 環境	SSH 環境
インスタンスは Amazon Linux または Ubuntu Server で実行されます。	Linux を実行する任意のクラウドコンピューティングインスタンス、または Linux を実行する独自のサーバーを使用できます。
AWS Cloud9 は、 の使用を開始するインスタンスを自動的にセットアップします AWS Cloud9。	インスタンスまたは独自のサーバーを、 AWS Cloud9で操作できるように手動で設定する必要があります。
AWS Cloud9 はインスタンスに AWS Command Line Interface (AWS CLI) を自動的にセットアップします。	インスタンスまたは独自のサーバー AWS CLI で を使用する場合は、自分で設定する必要があります。
インスタンスは何百もの便利なパッケージにアクセスできます。いくつかの一般的なパッケージは既にインストールされ設定済みです。例えば、Git、Docker、Node.js、および Python です。	必要に応じて、一般的なタスクを完了する追加のパッケージをダウンロード、インストール、設定する必要があります。
たとえば定期的にシステムの更新を適用することで、インスタンスを保守します。	インスタンスまたは独自のサーバーを保守します。
環境を削除すると、 は関連付けられたインスタンス AWS Cloud9 を自動的に終了します。	環境を削除した時、インスタンスまたは独自のサーバーは残ります。

EC2 環境	SSH 環境
AWS マネージド一時認証情報は EC2 環境で使用できます。これらの認証情報を使用すると、いくつかの制限で、発信者のすべての AWS リソースのすべての AWS アクションを有効または無効にできます AWS アカウント。環境の Amazon EC2 インスタンスのインスタンスプロファイルを設定したり、IAM ユーザーなどの AWS エンティティの永続的 AWS なアクセス認証情報を保存したりする必要はありません。環境の Amazon EC2 インスタンスがプライベートサブネットで起動された場合、AWS マネージド一時認証情報を使用して、Amazon EC2 環境が AWS エンティティ、例えば IAM ユーザーに代わって AWS サービスにアクセスすることを許可することはできません。 AWS Toolkit、Git パネル、拡張 Java サポートを使用できます。	AWS マネージド一時認証情報は SSH 環境では使用できません。AWS Identity and Access Management を使用して、 と他の AWS Cloud9 および AWS のサービス リソースの両方を操作できるようにするアクセス許可を管理する必要があります。 AWS Toolkit、Git パネル、および拡張Javaサポートは使用できません。

AWS Cloud9 開発環境の VPC 設定

Amazon Virtual Private Cloud (Amazon VPC) に関連付けられているすべての AWS Cloud9 開発環境は、特定の VPC 要件を満たしている必要があります。これらの環境には、EC2 環境、および VPC 内で実行される AWS クラウド コンピューティングインスタンスに関連付けられた SSH 環境が含まれます。例えば、Amazon EC2 インスタンスおよび Amazon Lightsail インスタンスです。

の Amazon VPC 要件 AWS Cloud9

が AWS Cloud9 使用する Amazon VPC には、次の設定が必要です。以下の要件は熟知しており、互換性のある VPC を早急に作成したいという場合は、「[VPC と他の VPC リソースを作成する](#)」に進んでください。

次のチェックリストを使用して、以下のすべての要件を VPC が満たしていることを確認します。

- VPC は、AWS Cloud9 開発環境と同じ AWS アカウント と AWS リージョン にすることも、VPC は環境 AWS アカウント とは異なる の共有 VPC にすることもできます。ただし、VPC は 環境 AWS リージョン と同じ する必要があります。の Amazon VPCs「」を参照してくださいの [VPCsのリストを表示する AWS リージョン](#)。AWS リージョンの Amazon VPC を作成する手順の詳細については AWS Cloud9、「」を参照してください[VPC と他の VPC リソースを作成する](#)。共有 Amazon VPC の使用の詳細については、「Amazon VPC ユーザーガイド」の「[共有 VPC の使用](#)」を参照してください。
- VPC にはパブリックサブネットが必要です。トラフィックがインターネットゲートウェイにルーティングされる場合、サブネットはパブリックです。Amazon VPC のサブネットのリストについては、「[VPC のサブネットのリストを表示する](#)」を参照してください。
- 環境が SSH 経由で EC2 インスタンスに直接アクセスしている場合、インスタンスはパブリックサブネットでのみ起動できます。サブネットがパブリックかどうかの確認方法については、「[パブリックサブネットかどうかを確認する](#)」を参照してください。
- Systems Manager を使って [no-ingress Amazon EC2 インスタンス](#)にアクセスしている場合、インスタンスはパブリックサブネットまたはプライベートサブネットに起動できます。
- パブリックサブネットを使用している場合は、インターネットゲートウェイを VPC にアタッチします。これは、AWS Systems Manager Agentインスタンスの (SSM Agent) が Systems Manager に接続できるようにするためです。
- プライベートサブネットを使用している場合は、パブリックサブネットで NAT ゲートウェイをホストして、サブネットのインスタンスによるインターネットとの通信を許可します。インターネットゲートウェイの設定を表示または変更する方法の詳細については、「[インターネットゲートウェイの設定を表示または変更する](#)」を参照してください。
- パブリックサブネットには、最小限のルートセットを含むルートテーブルが必要です。サブネットにルートテーブルがあるかどうかを確認する方法については、「[サブネットにルートテーブルがあるかどうかを確認する](#)」を参照してください。ルートテーブルを作成する方法については、「[ルートテーブルの作成](#)」を参照してください。
- VPC (またはアーキテクチャに応じて AWS クラウド コンピューティングインスタンス) に関連付けられたセキュリティグループでは、インバウンドトラフィックとアウトバウンドトラフィックの最小セットを許可する必要があります。Amazon VPC のセキュリティグループのリストについては、「[VPC のセキュリティグループのリストを表示する](#)」を参照してください。Amazon VPC のセキュリティグループの作成の詳細については、「[VPC でのセキュリティグループの作成](#)」を参照してください。
- セキュリティレイヤーを追加するため、VPC にネットワーク ACL がある場合、ネットワーク ACL はインバウンドおよびアウトバウンドトラフィックの最小セットを許可する必要があります。Amazon VPC に 1 つ以上のネットワーク ACL があるかどうかを確認するには、「[VPC に 1](#)

[つ以上のネットワーク ACL があるかどうかを確認する](#)」を参照してください。ネットワーク ACL の作成については、「[ネットワーク ACL の作成](#)」を参照してください。

- 開発環境が [SSM を使用して EC2 インスタンスにアクセス](#) し、インスタンスが起動先のパブリックサブネットによってパブリック IP アドレスに割り当てられていることを確認します。そのためには、パブリックサブネットのパブリック IP アドレスの自動割り当てオプションを有効にし、[Yes] に設定する必要があります。サブネット設定ページ内で AWS Cloud9 環境を作成する前に、これをパブリックサブネットで有効にすることができます。パブリックサブネットで自動割り当て IP 設定を変更する手順については、「Amazon VPC ユーザーガイド」の「[サブネットのパブリック IPv4 アドレス属性を変更する](#)」を参照してください。パブリックサブネットとプライベートサブネットの設定の詳細については、「[サブネットをパブリックまたはプライベートとして設定する](#)」を参照してください。

Note

次の手順では、にサインイン AWS マネジメントコンソール し、管理者認証情報を使用して Amazon VPC コンソール (<https://console.aws.amazon.com/vpc>) または Amazon EC2 コンソール (<https://console.aws.amazon.com/ec2>) を開きます。

AWS CLI または を使用する場合は AWS CloudShell、 の管理者の認証情報 AWS CloudShell を使用して AWS CLI または を設定することをお勧めします AWS アカウント。これができない場合は、AWS アカウント 管理者に確認してください。

の VPCs のリストを表示する AWS リージョン

Amazon VPC コンソールを使用するには、AWS ナビゲーションバーで、環境 AWS リージョン AWS Cloud9 を作成する を選択します。次に、ナビゲーションペインで、[お客様の VPC] を選択します。

AWS CLI または を使用するには AWS CloudShell、Amazon EC2 **describe-vpcs** コマンドを次のように実行します。

```
aws ec2 describe-vpcs --output table --query 'Vpcs[*].VpcId' --region us-east-2
```

前述のコマンドで、 を環境 AWS リージョン AWS Cloud9 を作成する us-east-2 に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

出力には、VPC ID のリストが含まれます。

VPC のサブネットのリストを表示する

Amazon VPC コンソールを使用するため、ナビゲーションペインで [お客様の VPC] を選択します。[VPC ID] 列の VPC の ID をメモしておきます。次に、ナビゲーションペインで [サブネット] を選択し、[VPC] 列でその ID を含むサブネットを探します。

AWS CLI または を使用するには `aws-shell`、Amazon EC2 **describe-subnets** コマンドを次のように実行します。

```
aws ec2 describe-subnets --output table --query 'Subnets[*].[SubnetId,VpcId]' --region us-east-2
```

前述のコマンド `us-east-2` で、 をサブネット AWS リージョン を含む に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力で、VPC の ID と一致するサブネットを探します。

パブリックサブネットかどうかを確認する

Important

環境の EC2 インスタンスをプライベートサブネットに起動するとします。SSM サービスに接続できるように、そのインスタンスに対してアウトバウンドトラフィックが許可されていることを確認してください。プライベートサブネットの場合、アウトバウンドトラフィックは通常、ネットワークアドレス変換 (NAT) ゲートウェイまたは VPC エンドポイントを介して設定されます。(NAT ゲートウェイにはパブリックサブネットが必要です)。

SSM にアクセスするために NAT ゲートウェイではなく VPC エンドポイントを選択するとします。これらのパッチがインターネットアクセスに依存する場合、インスタンスの自動更新とセキュリティパッチが機能しないことがあります。[AWS Systems Manager Patch Manager](#) などの他のアプリケーションを使用して、環境に必要なソフトウェア更新を管理できます。AWS Cloud9 ソフトウェアは通常どおり更新されます。

Amazon VPC コンソールを使用するには、ナビゲーションペインで [サブネット] を選択します。AWS Cloud9 使用するサブネットの横にあるボックスを選択します。[ルートテーブル] タブで、[ターゲット] 列に [igw-] で始まるエントリがある場合、そのサブネットはパブリックです。

AWS CLI または を使用するには `aws-shell`、Amazon EC2 **`describe-route-tables`** コマンドを実行します。

```
aws ec2 describe-route-tables --output table --query 'RouteTables[*].Routes[*].
{GatewayIds:GatewayId}' --region us-east-2 --filters Name=association.subnet-
id,Values=subnet-12a3456b
```

前述のコマンドで、 をサブネット AWS リージョン を含む `us-east-2` に置き換え、 をサブネット ID `subnet-12a3456b` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (`'`) を二重引用符 (`"`) で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力に `igw-` で始まる結果が 1 つでもある場合は、そのサブネットがパブリックです。

出力に結果がない場合は、ルートテーブルがサブネットではなく、代わりに VPC に関連付けられている可能性があります。これを確認するには、次の例のように、サブネット自体の代わりに、サブネットに関連する VPC に対して Amazon EC2 **`describe-route-tables`** コマンドを実行します。

```
aws ec2 describe-route-tables --output table --query 'RouteTables[*].Routes[*].
{GatewayIds:GatewayId}' --region us-east-1 --filters Name=vpc-id,Values=vpc-1234ab56
```

前述のコマンドで、 を VPC AWS リージョン を含む `us-east-2` に置き換え、 を VPC ID `vpc-1234ab56` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (`'`) を二重引用符 (`"`) で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力に `igw-` で始まる結果が 1 つでもある場合は、その VPC にはインターネットゲートウェイが含まれています。

インターネットゲートウェイの設定を表示または変更する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [インターネットゲートウェイ] を選択します。インターネットゲートウェイの横にあるボックスを選択します。設定を確認するには、それぞれのタブをクリックします。タブの設定を変更するには、該当する場合は [編集] を選択し、画面の指示に従います。

AWS CLI または を使用して設定 `aws-shell` を表示するには、Amazon EC2 **`describe-internet-gateways`** コマンドを実行します。

```
aws ec2 describe-internet-gateways --output table --region us-east-2 --internet-gateway-id igw-1234ab5c
```

前述のコマンドで、 をインターネットゲートウェイ AWS リージョン を含む us-east-2 に置き換え、 をインターネットゲートウェイ ID igw-1234ab5c に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

インターネットゲートウェイを作成する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [インターネットゲートウェイ] を選択します。 [インターネットゲートウェイの作成] を選択し、画面の指示に従います。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **create-internet-gateway** コマンドを実行します。

```
aws ec2 create-internet-gateway --output text --query  
'InternetGateway.InternetGatewayId' --region us-east-2
```

前述のコマンドで、 を新しいインターネットゲートウェイ AWS リージョン を含む us-east-2 に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

出力には、新規のインターネットゲートウェイの ID が含まれています。

インターネットゲートウェイを VPC にアタッチする

Amazon VPC コンソールを使用するには、ナビゲーションペインで [インターネットゲートウェイ] を選択します。インターネットゲートウェイの横にあるボックスを選択します。必要に応じて [アクション]、 [VPC にアタッチ] の順に選択し、画面の指示に従います。

AWS CLI または を使用するにはaws-shell、次のように Amazon EC2 **attach-internet-gateway** コマンドを実行します。

```
aws ec2 attach-internet-gateway --region us-east-2 --internet-gateway-id igw-a1b2cdef  
--vpc-id vpc-1234ab56
```

前述のコマンドus-east-2で、 をインターネットゲートウェイ AWS リージョン を含む に置き換えます。igw-a1b2cdef をインターネットゲートウェイ ID に置き換えます。さらに、vpc-1234ab56 を VPC ID に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

サブネットにルートテーブルがあるかどうかを確認する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [サブネット] を選択します。AWS Cloud9 使用する VPC のパブリックサブネットの横にあるボックスを選択します。[ルートテーブル] タブで、[ルートテーブル] の値がある場合は、パブリックサブネットにルートテーブルがあります。

AWS CLI または `aws-shell` を使用するには `aws-shell`、Amazon EC2 **describe-route-tables** コマンドを実行します。

```
aws ec2 describe-route-tables --output table --query 'RouteTables[*].Associations[*].{RouteTableIds:RouteTableId}' --region us-east-2 --filters Name=association.subnet-id,Values=subnet-12a3456b
```

前述のコマンドで、`subnet-12a3456b` をパブリックサブネット AWS リージョン を含む `us-east-2` に置き換え、`subnet-12a3456b` をパブリックサブネット ID `subnet-12a3456b` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (`'`) を二重引用符 (`"`) で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力に値がある場合、パブリックサブネットには少なくとも 1 つのルートテーブルがあります。

出力に結果がない場合は、ルートテーブルがサブネットではなく、代わりに VPC に関連付けられている可能性があります。これを確認するには、次の例のように、サブネット自体の代わりに、サブネットに関連する VPC に対して Amazon EC2 **describe-route-tables** コマンドを実行します。

```
aws ec2 describe-route-tables --output table --query 'RouteTables[*].Associations[*].{RouteTableIds:RouteTableId}' --region us-east-2 --filters Name=vpc-id,Values=vpc-1234ab56
```

前述のコマンドで、`vpc-1234ab56` を VPC AWS リージョン を含む `us-east-2` に置き換え、`vpc-1234ab56` を VPC ID `vpc-1234ab56` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (`'`) を二重引用符 (`"`) で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力に少なくとも 1 つの結果がある場合、VPC には少なくとも 1 つのルートテーブルがあります。

サブネットをルートテーブルに添付する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ルートテーブル] を選択します。アタッチするルートテーブルの横にあるチェックボックスをオンにします。[サブネットの関

連付け] タブで [編集] を選択し、アタッチ先のサブネットの横にあるチェックボックスをオンにして、[保存] を選択します。

AWS CLI または を使用するには `aws-shell`、Amazon EC2 **associate-route-table** コマンドを次のように実行します。

```
aws ec2 associate-route-table --region us-east-2 --subnet-id subnet-12a3456b --route-table-id rtb-ab12cde3
```

前述のコマンドで、 をルートテーブル AWS リージョン を含む `us-east-2` に置き換えます。 `subnet-12a3456b` をサブネット ID に置き換えます。さらに、 `rtb-ab12cde3` をルートテーブル ID に置き換えます。 `aws-shell` で前述のコマンドを実行する場合は、 `aws` を無視します。

ルートテーブルの作成

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ルートテーブル] を選択します。 [ルートテーブルの作成] を選択し、画面の指示に従います。

AWS CLI または を使用するには `aws-shell`、Amazon EC2 **create-route-table** コマンドを次のように実行します。

```
aws ec2 create-route-table --output text --query 'RouteTable.RouteTableId' --region us-east-2 --vpc-id vpc-1234ab56
```

前述のコマンドで、 を新しいルートテーブル AWS リージョン を含む `us-east-2` に置き換え、 を VPC ID `vpc-1234ab56` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。 `aws-shell` で前述のコマンドを実行する場合は、 `aws` を無視します。

出力には、新しいルートテーブルの ID が含まれています。

ルートテーブルの設定を表示または変更する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ルートテーブル] を選択します。ルートテーブルの横にあるチェックボックスをオンにします。設定を確認するには、それぞれのタブをクリックします。タブの設定を変更するには、 [編集] を選択し、画面の指示に従います。

AWS CLI または を使用して設定 `aws-shell` を表示するには、次のように Amazon EC2 **describe-route-tables** コマンドを実行します。

```
aws ec2 describe-route-tables --output table --region us-east-2 --route-table-ids rtb-ab12cde3
```

前述のコマンドで、 をルートテーブル AWS リージョン を含む us-east-2 に置き換え、 をルートテーブル ID rtb-ab12cde3 に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

の推奨ルートテーブルの最小設定 AWS Cloud9

送信先	ターゲット	ステータス	伝播済み
CIDR-BLOCK	ローカル	アクティブ	いいえ
0.0.0.0/0	igw-INTERNET-GATEWAY-ID	アクティブ	いいえ

これらの設定で、*CIDR-BLOCK* は、サブネットの CIDR ブロックで、*igw-INTERNET-GATEWAY-ID* は、互換性のあるインターネットゲートウェイの ID です。

VPC のセキュリティグループのリストを表示する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [セキュリティグループ] を選択します。[セキュリティグループの検索] ボックスに、VPC ID または名前を入力し、Enter キーを押します。その VPC のセキュリティグループが検索結果のリストに表示されます。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **describe-security-groups** コマンドを実行します。

```
aws ec2 describe-security-groups --output table --query 'SecurityGroups[*].GroupId' --region us-east-2 --filters Name=vpc-id,Values=vpc-1234ab56
```

前述のコマンドで、 を VPC AWS リージョン を含む us-east-2 に置き換え、 を VPC ID vpc-1234ab56 に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

出力には、その VPC のセキュリティグループ ID のリストが含まれています。

AWS クラウド コンピューティングインスタンスのセキュリティグループのリストを表示する

Amazon EC2 コンソールを使用するには、ナビゲーションペインで [インスタンス] を展開し、[インスタンス] を選択します。インスタンスの一覧で、インスタンスの横のボックスを選択します。そのインスタンスのセキュリティグループが、[Security groups] の横の [Description] タブに表示されます。

AWS CLI または `aws-shell` を使用するには `aws-shell`、次のように Amazon EC2 **describe-security-groups** コマンドを実行します。

```
aws ec2 describe-instances --output table --query  
'Reservations[*].Instances[*].NetworkInterfaces[*].Groups[*].GroupId' --region us-  
east-2 --instance-ids i-12a3c456d789e0123
```

前述のコマンドで、 をインスタンス AWS リージョン を含む `us-east-2` に置き換え、 をインスタンス ID `i-12a3c456d789e0123` に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (`'`) を二重引用符 (`"`) で置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

出力には、そのインスタンスのセキュリティグループ ID のリストが含まれています。

VPC でセキュリティグループの設定を表示または変更する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [セキュリティグループ] を選択します。セキュリティグループの横にあるボックスを選択します。設定を確認するには、それぞれのタブをクリックします。タブの設定を変更するには、該当する場合は [編集] を選択し、画面の指示に従います。

AWS CLI または `aws-shell` を使用して設定 `aws-shell` を表示するには、次のように Amazon EC2 **describe-security-groups** コマンドを実行します。

```
aws ec2 describe-security-groups --output table --region us-east-2 --group-ids  
sg-12a3b456
```

前述のコマンドで、 をインスタンス AWS リージョン を含む `us-east-2` に置き換え、 をセキュリティグループ ID `sg-12a3b456` に置き換えます。`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

AWS クラウド コンピューティングインスタンスのセキュリティグループの設定を表示または変更する

Amazon EC2 コンソールを使用するには、ナビゲーションペインで [インスタンス] を展開し、[インスタンス] を選択します。インスタンスの一覧で、インスタンスの横のボックスを選択します。[説明] タブの [セキュリティグループ] で、セキュリティグループを選択します。各タブを確認します。タブの設定を変更するには、該当する場合は [編集] を選択し、画面の指示に従います。

AWS CLI または `aws` を使用して設定 `aws-shell` を表示するには、次のように Amazon EC2 **describe-security-groups** コマンドを実行します。

```
aws ec2 describe-security-groups --output table --region us-east-2 --group-ids sg-12a3b456
```

前述のコマンドで、`sg-12a3b456` をインスタンス AWS リージョン を含む `us-east-2` に置き換え、`aws-shell` で前述のコマンドを実行する場合は、`aws` を無視します。

のインバウンドトラフィックとアウトバウンドトラフィックの最小設定 AWS Cloud9

Important

インスタンスの IA セキュリティグループにはインバウンドルールがない場合があります。この場合、別のホストからインスタンスへの着信トラフィックは許可されません。no-ingress EC2 インスタンスの使用については、[「を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager」](#)を参照してください。

- インバウンド: ポート 22 で SSH を使用するすべての IP アドレス。ただし、これらの IP アドレスは、[「のインバウンド SSH IP アドレス範囲 AWS Cloud9」](#)が AWS Cloud9 使用する IP アドレスのみに制限できます。詳細については、[「のインバウンド SSH IP アドレス範囲 AWS Cloud9」](#)を参照してください。

Note

2018 年 7 月 31 日以降に作成された EC2 環境の場合、`aws-shell` はセキュリティグループ AWS Cloud9 を使用して、ポート 22 経由で SSH を使用してインバウンド IP アドレスを制限します。これらのインバウンド IP アドレスは、特に `aws-shell` が AWS Cloud9 使用するアドレスのみ

です。詳細については、「[のインバウンド SSH IP アドレス範囲 AWS Cloud9](#)」を参照してください。

- インバウンド (ネットワーク ACL のみ): EC2 環境 と、Amazon Linux または Ubuntu Server を実行して Amazon EC2 インスタンスと関連付けられた SSH 環境である場合、ポート 32768〜61000 経由で TCP を使用するすべての IP アドレス。詳細と、他の Amazon EC2 インスタンスタイプのポート範囲を含むについては、Amazon VPC ユーザーガイドの「[一時ポート](#)」を参照してください。
- アウトバウンド: 任意のプロトコルとポートを使用するすべてのトラフィックソース。

セキュリティグループレベルでこの動作を設定することができます。さらにセキュリティレベルを高めるために、ネットワーク ACL を使用することもできます。詳細については、Amazon VPC ユーザーガイドの「[セキュリティグループおよびネットワーク ACL の比較](#)」を参照してください。

たとえば、インバウンドおよびアウトバウンドのルールをセキュリティグループに追加するには、次のようにルールを設定します。

インバウンドルール

タイプ	プロトコル	ポート範囲	ソース
SSH (22)	TCP (6)	22	0.0.0.0 (ただし次の注記および のインバウンド SSH IP アドレス範囲 AWS Cloud9 を参照)。

Note

2018 年 7 月 31 日以降に作成された EC2 環境の場合、はポート 22 経由で SSH を使用してインバウンド IP アドレスを制限するインバウンドルール AWS Cloud9 を追加します。これにより、が AWS Cloud9 使用するアドレスのみに制限されます。詳細については、「[のインバウンド SSH IP アドレス範囲 AWS Cloud9](#)」を参照してください。

アウトバウンドルール

タイプ	プロトコル	ポート範囲	ソース
すべてのトラフィック	すべて	すべて	0.0.0.0/0

例えば、インバウンドおよびアウトバウンドのルールをネットワーク ACL に追加するよう選択した場合は、次のようにルールをセットアップします。

インバウンドルール

ルール番号	タイプ	プロトコル	ポート範囲	ソース	許可/拒否
100	SSH (22)	TCP (6)	22	0.0.0.0 (ただし のインバウンド SSH IP アドレス範囲 AWS Cloud9 を参照)。	許可
200	カスタム TCP ルール	TCP (6)	32768-61000 (Amazon Linux および Ubuntu Server インスタンスの場合。その他のインスタンスタイプについては、 一時ポート を参照してください)。	0.0.0.0/0	許可

ルール番号	タイプ	プロトコル	ポート範囲	ソース	許可/拒否
*	すべてのトラフィック	すべて	すべて	0.0.0.0/0	DENY

アウトバウンドルール

ルール番号	タイプ	プロトコル	ポート範囲	ソース	許可/拒否
100	すべてのトラフィック	すべて	すべて	0.0.0.0/0	許可
*	すべてのトラフィック	すべて	すべて	0.0.0.0/0	DENY

セキュリティグループとネットワーク ACL の詳細については、次のAmazon VPC ユーザーガイドを参照してください。

- [セキュリティ](#)
- [VPC のセキュリティグループ](#)
- [ネットワーク ACL](#)

VPC でのセキュリティグループの作成

Amazon VPC または Amazon EC2 コンソールを使用するには、次のいずれかのアクションを実行します。

- Amazon VPC コンソールのナビゲーションペインで、[セキュリティグループ] を選択します。[セキュリティグループの作成] を選択し、画面の指示に従います。
- Amazon EC2 コンソールのナビゲーションペインで [ネットワーク & セキュリティ] を展開し、[セキュリティグループ] を選択します。[セキュリティグループの作成] を選択し、画面の指示に従います。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **create-security-group** コマンドを次のように実行します。

```
aws ec2 create-security-group --region us-east-2 --vpc-id vpc-1234ab56
```

前述のコマンドで、 を VPC AWS リージョン を含む us-east-2 に置き換え、 を VPC ID vpc-1234ab56 に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

VPC に 1 つ以上のネットワーク ACL があるかどうかを確認する

Amazon VPC コンソールを使用するため、ナビゲーションペインで [お客様の VPC] を選択します。AWS Cloud9 使用する VPC の横にあるボックスを選択します。[概要] タブで、[ネットワーク ACL] の値がある場合は、VPC に少なくとも 1 つのネットワーク ACL があります。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **describe-network-acls** コマンドを実行します。

```
aws ec2 describe-network-acls --output table --query  
'NetworkAcls[*].Associations[*].NetworkAclId' --region us-east-2 --filters Name=vpc-  
id,Values=vpc-1234ab56
```

前述のコマンドで、 を VPC AWS リージョン を含む us-east-2 に置き換え、 を VPC ID vpc-1234ab56 に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

出力のリストに少なくとも 1 つのエントリがある場合、その VPC には少なくとも 1 つのネットワーク ACL があります。

VPC のネットワーク ACL のリストを表示する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ネットワーク ACL] を選択します。[ネットワーク ACL を検索] ボックスに、VPC ID または名前を入力し、[Enter] キーを押します。その VPC のネットワーク ACL が検索結果のリストに表示されます。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **describe-network-acls** コマンドを実行します。

```
aws ec2 describe-network-acls --output table --query  
'NetworkAcls[*].Associations[*].NetworkAclId' --region us-east-2 --filters Name=vpc-  
id,Values=vpc-1234ab56
```

前述のコマンドで、 を VPC AWS リージョン を含む us-east-2 に置き換え、 を VPC ID vpc-1234ab56 に置き換えます。上記のコマンドを Windows で実行するには、一重引用符 (') を二重引用符 (") で置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

出力には、その VPC のネットワーク ACL のリストが含まれています。

ネットワーク ACL の設定を表示または変更する

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ネットワーク ACL] を選択します。ネットワーク ACL の横にあるボックスを選択します。設定を確認するには、それぞれのタブをクリックします。タブの設定を変更するには、該当する場合は [編集] を選択し、画面の指示に従います。

AWS CLI または を使用して設定aws-shellを表示するには、Amazon EC2 **describe-network-acls** コマンドを実行します。

```
aws ec2 describe-network-acls --output table --region us-east-2 --network-acl-ids
acl-1234ab56
```

前述のコマンドで、 をネットワーク ACL AWS リージョン を含む us-east-2 に置き換え、 をネットワーク ACL ID acl-1234ab56 に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

ネットワーク ACL の作成

Amazon VPC コンソールを使用するには、ナビゲーションペインで [ネットワーク ACL] を選択します。[ネットワーク ACL の作成] を選択し、画面の指示に従います。

AWS CLI または を使用するにはaws-shell、Amazon EC2 **create-network-acl** コマンドを実行します。

```
aws ec2 create-network-acl --region us-east-2 --vpc-id vpc-1234ab56
```

前述のコマンドus-east-2で、 を、新しいネットワーク ACL をアタッチする VPC AWS リージョン を含む に置き換えます。さらに、vpc-1234ab56 を VPC ID に置き換えます。aws-shell で前述のコマンドを実行する場合は、aws を無視します。

VPC と他の VPC リソースを作成する

次の手順に従って、アプリケーションの実行に必要な VPC および追加の VPC リソースを作成します。VPC リソースには、サブネット、ルートテーブル、インターネットゲートウェイ、NAT ゲートウェイが含まれます。

コンソールを使用して VPC、サブネット、その他の VPC リソースを作成するには

1. Amazon VPC コンソールの <https://console.aws.amazon.com/vpc/> を開いてください。
2. VPC ダッシュボードで、[Create VPC (VPC を作成する)] を選択します。
3. [Resources to create] (作成するリソース) で、[VPC and more] (VPC など) を選択します。
4. VPC リソースの名前タグを作成するには、[名前タグの自動生成] を選択したままにします。VPC リソースに独自の名前タグを指定するには、これを選択解除します。
5. [IPv4 CIDR ブロック] に VPC の IPv4 アドレス範囲を入力する必要があります。の推奨 IPv4 範囲 AWS Cloud9 は です 10.0.0.0/16。
6. (オプション) IPv6 トラフィックをサポートするには、[IPv6 CIDR ブロック]、[Amazon が提供する IPv6 CIDR ブロック] の順に選択します。
7. [テナンシー] を選択します。このオプションは、VPC で起動する EC2 インスタンスを、他の AWS アカウント と共有しているハードウェアで実行するか、または自分専用のハードウェアで実行するかを定義します。VPC のテナンシーとして Default を選択すると、この VPC で起動した EC2 インスタンスは、インスタンスの起動時に指定したテナンシー属性を使用します。詳細については、「Amazon EC2 ユーザーガイド」の「[定義済みのパラメータを使用したインスタンスの起動](#)」を参照してください。

VPC のテナンシーで Dedicated を選択すると、インスタンスは常に、ユーザー専用のハードウェアで実行される、[専有インスタンス](#)として実行されます。AWS Outposts を使用している場合、Outpost にはプライベート接続が必要であり、Default テナンシーを使用する必要があります。

8. [アベイラビリティーゾーン (AZ) の数] では、本番環境用にサブネットを 2 つ以上の Availability Zones でプロビジョニングすることをお勧めします。サブネットの AZ を選択するには、[AZ のカスタマイズ] を展開します。それ以外の場合は、で AZs AWS を選択できます。
9. サブネットを設定するには、[パブリックサブネットの数] と [プライベートサブネットの数] の値を選択します。サブネットの IP アドレス範囲を選択するには、[サブネット CIDR ブロックをカスタマイズ] を展開します。それ以外の場合は、自分で AWS 選択します。
10. (オプション) プライベートサブネットのリソースが IPv4 経由でパブリックインターネットにアクセスする必要がある場合は、[NAT ゲートウェイ] で、NAT ゲートウェイを作成する AZ の数

を選択します。本番環境では、パブリックインターネットへのアクセスを必要とするリソースがある各 AZ に NAT ゲートウェイをデプロイすることをお勧めします。

11. (オプション) プライベートサブネット内のリソースが IPv6 経由でパブリックインターネットにアクセスする必要がある場合は、[Egress Only インターネットゲートウェイ] で、[はい] をクリックします。
12. (オプション) VPC から Amazon S3 に直接アクセスするには、[VPC エンドポイント]、[S3 ゲートウェイ] の順に選択します。これにより、Amazon S3 用のゲートウェイ VPC エンドポイントが作成されます。詳細については、「AWS PrivateLink ガイド」の「[ゲートウェイ VPC エンドポイント](#)」を参照してください。
13. (オプション) [DNS オプション] では、ドメイン名解決の両方のオプションがデフォルトで有効になっています。デフォルトの設定がニーズに合わない場合は、これらのオプションを非アクティブ化できます。
14. (オプション) VPC にタグを追加するには、[追加のタグ] を展開して、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
15. [プレビュー] ペインでは、設定した VPC リソース間の関係を視覚化できます。実線はリソース間の関係を表します。点線は、NAT ゲートウェイ、インターネットゲートウェイ、およびゲートウェイエンドポイントへのネットワークトラフィックを表します。VPC の作成後、[リソースマップ] タブを使用することで、VPC 内のリソースをこの形式でいつでも視覚化できます。
16. VPC の設定が完了したら、[VPC の作成] を選択します。

VPC のみを作成する

次の手順に従って Amazon VPC コンソールを使用し、追加の VPC リソースのない VPC を作成します。

コンソールを使用して追加の VPC リソースのない VPC を作成するには

1. Amazon VPC コンソールの <https://console.aws.amazon.com/vpc/> を開いてください。
2. VPC ダッシュボードで、[Create VPC (VPC を作成する)] を選択します。
3. [作成するリソース] で、[VPC のみ] を選択します。
4. (オプション) [名前タグ] に、使用する VPC の名前を入力します。これにより、Name というキーと指定した値を含むタグが作成されます。
5. [IPv4 CIDR block] で、次のいずれかを実行します。
 - [IPv4 CIDR 手動入力] を選択し、VPC の IPv4 アドレス範囲を入力します。の推奨 IPv4 範囲 AWS Cloud9 は です 10.0.0.0/16。

- [IPAM が割り当てられた IPv4 CIDR ブロック] を選択し、Amazon VPC IP Address Manager (IPAM) の IPv4 アドレスプールとネットマスクを選択します。CIDR ブロックのサイズは、IPAM プールの割り当てルールによって制限されます。IPAM は、AWS ワークロードの IP アドレスの計画、追跡、モニタリングに役立つ VPC 機能です。詳細については、「Amazon Virtual Private Cloud 管理者ガイド」の「[Amazon VPC とは](#)」を参照してください。

IPAM を使用して IP アドレスを管理する場合は、このオプションを選択することをお勧めします。このオプションを選択しないと、VPC に指定した CIDR ブロックが IPAM CIDR 割り当てと重複する可能性があります。

6. (オプション) デュアルスタック VPC を作成するには、VPC の IPv6 アドレス範囲を指定します。[IPv6 CIDR ブロック] で、次のいずれかを実行します。
 - [IPAM が割り当てられた IPv6 CIDR] を選択し、IPAM IPv6 アドレスプールを選択します。CIDR ブロックのサイズは、IPAM プールの割り当てルールによって制限されます。
 - Amazon の IPv6 アドレスプールの IPv6 CIDR ブロックをリクエストするには、[Amazon が提供する IPv6 CIDR ブロック] を選択します。Network Border Group で、[ガ IP アドレスを AWS アドバタイズするグループ](#)を選択します。Amazon では IPv6 CIDR ブロックサイズが /56 に固定されています。
 - Bring Your Own IPv6 Address (BYOIP) を使用して に持ち込んだ IPv6 CIDR ブロックを使用するには、私が所有する IPv6 CIDR を選択します。AWS <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-byoip.html>[Pool (プール)] で、IPv6 CIDR ブロックの割り当て元となる IPv6 アドレスプールを選択します。
7. (オプション) [テナンシー] を選択します。このオプションは、VPC で起動する EC2 インスタンスを、他の と共有されているハードウェアで実行するか、ユーザー専用のハードウェアで実行する AWS アカウント かを定義します。VPC のテナンシーとして Default を選択すると、この VPC で起動した EC2 インスタンスは、インスタンスの起動時に指定したテナンシー属性を使用します。詳細については、「Amazon EC2 ユーザーガイド」の「[定義済みのパラメータを使用したインスタンスの起動](#)」を参照してください。

VPC のテナンシーで Dedicated を選択すると、インスタンスは常に、ユーザー専用のハードウェアで実行される、[専有インスタンス](#)として実行されます。AWS Outposts を使用している場合、Outpost にはプライベート接続が必要であり、Default テナンシーを使用する必要があります。
8. (オプション) VPC にタグを追加するには、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
9. [Create VPC (VPC の作成)] を選択します。

10. VPC の作成後、サブネットを追加できます。

のサブネットを作成する AWS Cloud9

Amazon VPC コンソールを使用して、互換性のある VPC のサブネットを作成できます AWS Cloud9。EC2 インスタンスのプライベートサブネットとパブリックサブネットのどちらを作成できるかは、環境の接続方法によって異なります。

- SSH による直接アクセス：パブリックサブネットのみ
- Systems Manager からのアクセス: パブリックまたはプライベートサブネット

環境の EC2 をプライベートサブネットに起動するオプションが利用できるのは、[コンソール、コマンドライン、または CloudFormation](#) を使った「no-ingress」EC2 環境を作成する場合に限られます。

パブリックまたはプライベートのいずれかになりえる[サブネットを作成と同じステップ](#)に従ってください。サブネットがインターネットゲートウェイへのルートがあるルートテーブルに関連付けられている場合は、「パブリックサブネット」と呼ばれます。サブネットがインターネットゲートウェイへのルートがないルートテーブルに関連付けられている場合は、「プライベートサブネット」と呼ばれます。詳細については、[サブネットをパブリックまたはプライベートとして設定する](#)を参照してください。

前の手順に従って の VPC を作成した場合は AWS Cloud9、この手順に従う必要もありません。これは、[新しい VPC の作成] ウィザードによって自動的にサブネットが作成されるためです。

Important

- には、環境と同じ AWS リージョン に互換性のある VPC が既に存在している AWS アカウント 必要があります。詳細については、[の Amazon VPC 要件 AWS Cloud9](#)の VPC 要件を参照してください。
- この手順では、 にサインイン AWS マネジメントコンソール し、 の IAM 管理者の認証情報を使用して Amazon VPC コンソールを開くことをお勧めします AWS アカウント。これができない場合は、AWS アカウント 管理者に確認してください。
- 一部の組織では、独自のサブネットの作成が許可されていない場合があります。サブネットを作成できない場合は、AWS アカウント 管理者またはネットワーク管理者に確認してください。

サブネットを作成するには

1. Amazon VPC コンソールがまだ開いていない場合は、 にサインイン AWS マネジメントコンソール し、 <https://console.aws.amazon.com/vpc> で Amazon VPC コンソールを開きます。
2. ナビゲーションバーで、AWS リージョン が環境のリージョンと同じでない場合は、正しいリージョンを選択します。
3. [サブネット] ページが表示されていない場合は、ナビゲーションペインで [サブネット] を選択します。
4. [Create Subnet (サブネットの作成)] を選択します。
5. [サブネットの作成] ダイアログボックスの [名前タグ] にサブネットの名前を入力します。
6. [VPC] で、サブネットを関連付ける VPC を選択します。
7. アベイラビリティーゾーンでは、使用するサブネット AWS リージョン の 内のアベイラビリティーゾーンを選択するか、設定なしを選択して、 がアベイラビリティーゾーン AWS を選択できるようにします。
8. [IPv4 CIDR ブロック] に、使用するサブネットの IP アドレス範囲を CIDR 形式で入力します。この IP アドレスの範囲は、VPC の IP アドレスのサブネットである必要があります。

CIDR ブロックの詳細については、Amazon VPC ユーザーガイドの[VPC とサブネットのサイジング](#)を参照してください。 [3.1 Basic Concept and Prefix Notation](#) (RFC 4632) または [IPv4 CIDR ブロック](#) (Wikipedia) も参照してください。

サブネットの作成後、[パブリックサブネットまたはプライベートサブネットとして設定する](#)。

サブネットをパブリックまたはプライベートとして設定する

サブネットを作成した後、インターネットとの通信方法を指定して、サブネットをパブリックまたはプライベートにすることができます。

パブリックサブネットにはパブリック IP アドレスがあり、サブネットのインスタンスと他の AWS のサービスの通信を可能にするインターネットゲートウェイ (IGW)が添付されています。

プライベートサブネット内のインスタンスにはプライベート IP アドレスがあり、ネットワークアドレス変換 (NAT) ゲートウェイを使用して、サブネットとインターネットとその他の AWS のサービスの間でトラフィックを送受信します。NAT ゲートウェイはパブリックサブネットにホストする必要があります。

Public subnets

Note

環境のインスタンスがプライベートサブネットでも起動された場合でも、VPC は少なくとも 1 つのパブリックサブネットを備えている必要があります。これは、インスタンスとの間でトラフィックを転送する NAT ゲートウェイがパブリックサブネットにホストされている必要があるためです。

サブネットをパブリックとして設定するには、インターネットゲートウェイ (IGW) を添付し、その IGW へのルートを指定するようにルートテーブルを設定し、インバウンドトラフィックとアウトバウンドトラフィックを制御するため、セキュリティグループの設定を定義します。

これらのタスクの実行に関するガイダンスは [VPC と他の VPC リソースを作成する](#) で提供されています。

Important

開発環境が [SSM を使用して EC2 インスタンスにアクセス](#) し、インスタンスが起動先のパブリックサブネットによってパブリック IP アドレスに割り当てられていることを確認します。そのためには、パブリックサブネットのパブリック IP アドレスの自動割り当てオプションを有効にし、[Yes] に設定する必要があります。これは、サブネット設定ページ内に AWS Cloud9 環境を作成する前に、パブリックサブネットに有効にできます。パブリックサブネットに自動割り当て IP 設定を変更する手順については、「Amazon VPC ユーザーガイド」の「[サブネットのパブリック IPv4 アドレス属性を変更する](#)」を参照してください。パブリックサブネットとプライベートサブネットの詳細については、「[サブネットをパブリックまたはプライベートとして設定する](#)」を参照してください。

Private subnets

Systems Manager からアクセスする no-ingress インスタンスを作成する場合は、プライベートサブネットでも起動できます。プライベートサブネットにはパブリック IP アドレスはありません。そのため、リクエストには、プライベート IP アドレスをパブリックアドレスにマッピングし、応答には、パブリック IP アドレスをプライベートアドレスにマッピングする必要があります。

⚠ Warning

アカウントで NAT ゲートウェイを作成して使用するには料金がかかります。NAT ゲートウェイの時間単位の使用料金とデータ処理料金が適用されます。Amazon EC2 データ転送料金も適用されます。詳細については、「[Amazon VPC の料金](#)」を参照してください。

NAT ゲートウェイを作成して設定する前に、次の手順を行います。

- NAT ゲートウェイをホストするパブリック VPC サブネットを作成します。
- NAT ゲートウェイに割り当てることができる[Elastic IP アドレス](#)をプロビジョンします。
- プライベートサブネットの場合は、起動されたインスタンスがプライベート IP アドレスに割り当てられるように、[パブリック IPv4 アドレスの自動割り当てを有効にする] チェックボックスを解除します。詳細については、Amazon VPC ユーザーガイドの「[VPC の IP アドレス指定](#)」を参照してください。

このタスクのステップについては、Amazon VPC ユーザーガイドの「[NAT ゲートウェイの使用](#)」を参照してください。。

⚠ Important

現在、環境の EC2 インスタンスがプライベートサブネットで起動されている場合、[AWS マネージド一時認証情報](#)を使用して、EC2 環境が IAM ユーザーなどの AWS エンティティ AWS のサービスに代わってにアクセスすることを許可することはできません。

SSH 環境ホスト要件

環境を既存のクラウドコンピューティングインスタンスまたは独自のサーバーに接続する AWS Cloud9 ように に指示するには、AWS Cloud9 SSH 開発環境を作成します。ただし、SSH 環境を作成する前に、代わりに EC2 環境を作成することの利点を検討してみてください。

EC2 環境 を作成すると、AWS Cloud9 によって新しい環境が作成され、Amazon EC2 に対して新しいインスタンスの起動がリクエストされます。その後、新しく起動されたインスタンスが新しい環境に接続されます。EC2 環境の作成には次の利点があります。

- インスタンスが自動で起動される。EC2 環境を作成すると、は Amazon EC2 に新しいインスタンスの作成を同時に AWS Cloud9 リクエストします。SSH 環境では、既存のクラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) または独自のサーバーを自分で指定する必要があります。
- インスタンスが自動でシャットダウンされる。デフォルトでは、EC2 環境の IDE に接続されたすべてのウェブブラウザインスタンスが閉じられてから 30 分後に、AWS Cloud9 が自動的に EC2 環境をシャットダウンします。(この動作はいつでも変更できます)。これにより、追加料金が Amazon EC2 使用用 AWS アカウント に適用される可能性を減らすことができます。
- インスタンスが自動でクリーンアップされる。EC2 環境を削除すると、接続されている Amazon EC2 インスタンスも自動的に削除されます。これにより、Amazon EC2 の使用 AWS アカウント に対して に適用される追加料金の可能性を減らすこともできます。クラウドコンピューティングインスタンスに接続されている SSH 環境 では、インスタンスを自分で削除する必要があります。
- AWS マネージド一時認証情報。EC2 環境では、発信者のすべての AWS リソースのすべての AWS アクションを簡単にオンまたはオフにできます AWS アカウント (いくつかの制限があります)。環境の Amazon EC2 インスタンスのインスタンスプロファイルを設定したり、エンティティ (IAM ユーザーなど) の AWS 永続的 AWS なアクセス認証情報を保存したりする必要はありません。

詳細については、「[AWS マネージド一時認証情報](#)」を参照してください。

- AWS Toolkit と Git パネル。ビジュアルソースコントロール AWS のサービス を操作して使用するためのこれらのツールは、Amazon EC2 AWS Cloud9 インスタンスで作成された環境でのみ使用できます。

代わりに EC2 環境を作成したい場合は、「[EC2 環境を作成する](#)」を参照してください。それ以外の場合は、SSH 環境の作成に関する情報を読み続けてください。

SSH 環境を作成するタイミングと方法

以下の要件がある場合は、EC2 環境ではなく SSH 環境を作成する必要があります。

要件	手順
AWS クラウド コンピューティングインスタンスの使用 AWS アカウント に対して に追加料金が発生しないようにします。したがって、代わりに AWS の外部にある既存のクラウドコ	1. インスタンスまたはサーバーが、このトピックで後述する 要件 を満たしていることを確認します。

要件	手順
コンピューティングインスタンスまたは独自のサーバー AWS Cloud9 に接続することにしました。	2. 用の SSH 環境を作成 AWS Cloud9 して、インスタンスまたはサーバーをそこに接続します。
環境の作成時に新しいインスタンス AWS Cloud9 を起動する AWS アカウント のではなく、で既存の AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) を使用します。	1. インスタンスが、このトピックで後述する要件を満たしていることを確認します。 2. 用の SSH 環境を作成 AWS Cloud9 して、インスタンスをそこに接続します。
AWS Cloud9 現在 EC2 環境でサポートされていない Amazon EC2 インスタンスタイプ (例:) を使用します R4。	1. 目的のインスタンスタイプに基づいて Amazon EC2 インスタンスを起動します。または、目的のインスタンスタイプ AWS アカウント を実行する 内の既存のインスタンスを特定します。 2. インスタンスが、このトピックで後述する要件を満たしていることを確認します。 3. 用の SSH 環境を作成 AWS Cloud9 して、インスタンスをそこに接続します。
Amazon Linux または Ubuntu サーバー以外の Amazon マシンイメージ (AMI) に基づいた Amazon EC2 インスタンスを使用したいと考えています。	1. 目的の AMI に基づいて Amazon EC2 インスタンスを起動します。または、目的の AMI AWS アカウント に基づいて 内の既存のインスタンスを特定します。 2. インスタンスが、このトピックで後述する要件を満たしていることを確認します。 3. 用の SSH 環境を作成 AWS Cloud9 して、インスタンスをそこに接続します。
複数の環境を単一の既存のクラウドコンピューティングインスタンスまたは独自のサーバーに接続したいと考えています。	1. インスタンスまたはサーバーが、このトピックで後述する要件を満たしていることを確認します。 2. インスタンスまたはサーバー AWS Cloud9 を接続する環境ごとに SSH 環境を作成します。

Note

Amazon EC2 インスタンスを起動すると、Amazon EC2の AWS アカウント に対して料金が発生する場合があります。詳細については、[Amazon EC2 の料金表](#)を参照してください。

SSH ホスト要件

既存のクラウドコンピューティングインスタンスまたは独自のサーバーは、が SSH 環境に接続 AWS Cloud9 するための以下の要件を満たしている必要があります。

- Linux を実行する必要があります (Windows AWS Cloud9 はサポートされていません)。
- Arm ベースのアーキテクチャを使用することはできません。(Arm プロセッサを中心に構築されたシステムのサポートは検討中です)。
- SSH を使用してパブリックインターネット経由で到達可能である必要があります。Virtual Private Cloud (VPC) またはバーチャルプライベートネットワーク (VPN) 経由でのみ到達可能である場合、その VPC または VPN はパブリックインターネットにアクセスできる必要があります。
- ホストが Amazon Virtual Private AWS Cloud (Amazon VPC) の一部である既存のクラウドコンピューティングインスタンスである場合は、追加の要件があります。Amazon Virtual Private Cloud 詳細については、「[Amazon VPC の設定](#)」を参照してください。
- Python3 インストールpip3時に、 をインストールしてデフォルトPythonバージョンとして設定する必要があります AWS Cloud9。バージョンをチェックするには、既存のインスタンスまたはサーバーのターミナルから、コマンド **python --version** を実行します。インスタンスまたはサーバーに Python をインストールするには、以下のリソースのいずれかを参照してください。
 - 「Python サンプル」の「[ステップ 1: 必要なツールをインストールする](#)」。
 - Python ウェブサイトから [Python をダウンロード](#)します。

Note

既存の AWS クラウド コンピューティングインスタンスに接続して要件を満たすには、次の 1 つ以上のリソースを参照してください。

- Amazon EC2 については、「Amazon EC2 ユーザーガイド」の「[Linux インスタンスに接続](#)」を参照してください。
- Amazon Lightsail については、Amazon Lightsail ドキュメントの「[Linux/UNIX ベースの Lightsail インスタンスへの接続](#)」を参照してください。

- については AWS Elastic Beanstalk、[「AWS Elastic Beanstalk デベロッパーガイド」の「サーバーインスタンスの一覧表示と接続」](#)を参照してください。
- については AWS OpsWorks、[「AWS OpsWorks ユーザーガイド」の「SSH を使用して Linux インスタンスにログインする」](#)を参照してください。
- その他の方法については AWS のサービス、サービスの[ドキュメント](#)を参照してください。

独自のサーバーに接続して要件を確認し、これらの要件を満たすには、「SSH コマンドを使用してサーバーに接続」(macOS/Linux)、または「PuTTY を使用してサーバーに接続」(Windows) などの語句を使用してインターネットを検索してください。

- 次のコマンドを実行して、必要なすべてのパッケージをインストールします。

Amazon Linux の場合:

```
sudo yum install -y make glibc-devel gcc gcc-c++
```

Ubuntu Server の場合:

```
sudo apt install build-essential
```

- Node.js がインストールされている必要があります。ホストのオペレーティングシステムでサポートされている最新の Node.js バージョンをインストールすることをお勧めします。

Warning

AWS Cloud9 でサポートされていない Node.js バージョンを使用すると、SSH 環境の作成時にインストールの問題が発生する可能性があります AWS Cloud9。

バージョンをチェックするには、既存のインスタンスのターミナルまたはサーバーから、コマンド **node --version** を実行します。インスタンスまたはサーバーに Node.js をインストールするには、次のいずれかを参照してください。

- Node.js サンプルの [ステップ 1: 必要なツールをインストールする](#)。
- Node.js ウェブサイトの [「Installing Node.js via package manager」](#)。
- GitHub の [「Node Version Manager」](#)。

- 既存のインスタンスやサーバーでログイン後に AWS Cloud9 の起動元とするディレクトリパスには、`rwxr-xr-x` に設定されたアクセス許可が必要です。つまり、[Configure settings] (設定の構成) ページで [User] (ユーザー) に対して [環境の作成ウィザード](#) で指定したログイン名に対応する所有者への読み取り/書き込み/実行のアクセス許可、この所有者が所属するグループへの読み取り/実行のアクセス許可、およびその他への読み取り/実行のアクセス許可が必要です。

例えば、ディレクトリのパスが `~` である場合 (`~` は [Configure settings] (設定の構成) ページで [User] (ユーザー) に指定したログイン名のホームディレクトリを表します)、次のコマンドと続く指示を使用し、インスタンスまたはサーバーで `chmod` コマンドを実行して、このような許可をディレクトリに設定できます。

```
sudo chmod u=rwx,g=rx,o=rx ~
```

- 既存のインスタンスまたはサーバーに [AWS Cloud9 インストーラをダウンロードして実行](#) します。
- 必要に応じて、SSH 経由のインバウンドトラフィックを が AWS Cloud9 使用する IP アドレスのみに制限できます。これを行うには、IP 範囲へのインバウンド SSH トラフィックを、「[のインバウンド SSH IP アドレス範囲 AWS Cloud9](#)」で説明されているように設定します。

インスタンスまたはサーバーが上記の要件を満たしていることを確認したら、 が接続 AWS Cloud9 するための [SSH 環境を作成](#) します。

SSH AWS Cloud9 環境用の AWS Cloud9 インストーラの使用

SSH AWS Cloud9 開発環境を作成する前に、環境に接続するクラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) または独自のサーバーが [SSH ホスト要件を満た](#) している必要があります。これらの要件の 1 つは、インスタンスまたはサーバーで AWS Cloud9 インストーラをダウンロードして実行する必要があることです。Installer AWS Cloud9 は、インスタンスまたはサーバーが が AWS Cloud9 サポートするオペレーティングシステムプラットフォームとアーキテクチャで実行されているかどうかを確認する Linux シェルスクリプトです。このチェックが成功すると、スクリプトはインスタンスまたはサーバー上 AWS Cloud9 に存在する必要があるコンポーネントとその依存関係のインストールを試みます。

このトピックでは、このインストーラスクリプトをターゲットのインスタンスやサーバーにダウンロードして実行する方法について説明します。

- [AWS Cloud9 インストーラをダウンロードして実行する](#)
- [AWS Cloud9 インストーラのトラブルシューティング](#)

AWS Cloud9 インストーラをダウンロードして実行する

1. 環境に接続したいクラウドコンピューティングインスタンスまたは独自のサーバーが [SSH ホスト要件](#)を満たしていることを確認します。これには、特定のバージョンの Python および Node.js をインストール済みであること、ログイン後に AWS Cloud9 をスタートしたいディレクトリに特定の許可を設定していること、関連する Amazon Virtual Private Cloud をセットアップしていることなどが含まれます。
2. インスタンスまたはサーバーに接続しているときに、そのインスタンスまたはサーバーで以下のいずれかのコマンドを実行します。いずれかのコマンドを実行する前に gcc をインストールする必要があります。

```
curl -L https://d3kgj69l4ph6w4.cloudfront.net/static/c9-install-2.0.0.sh | bash
wget -O - https://d3kgj69l4ph6w4.cloudfront.net/static/c9-install-2.0.0.sh | bash
```

3. エラーなしで [Done (完了)] メッセージが表示された場合は、[SSH 環境を作成](#)できます。

エラーメッセージが表示された場合は、次のセクションのトラブルシューティング情報を参照してください。

AWS Cloud9 インストーラのトラブルシューティング

このセクションでは、AWS Cloud9 一般的な問題、考えられる原因、インストーラエラーのトラブルシューティングに推奨される解決策について説明します。

該当する問題が以下に示されていない場合や、追加のヘルプが必要な場合は、[AWS Cloud9 ディスカッションフォーラム](#)を参照してください。(このフォーラムに参加すると、サインインが必要になる AWS 場合があります)。当社に直接[お問い合わせ](#)いただくこともできます。

- [-bash: wget: コマンドが見つかりません](#)
- [エラー: 続行するには make をインストールしてください](#)
- [エラー: 続行するには gcc をインストールしてください](#)
- [設定: エラー: カースが見つかりません](#)

-bash: wget: コマンドが見つかりません

問題: インストーラスクリプトを実行すると、-bash: wget: command not found というメッセージが表示されます。

考えられる原因: **wget** ユーティリティがインスタンスまたはサーバーにインストールされていません。

推奨される解決策: 代わりに **curl** ユーティリティを使用してインストーラスクリプトをインスタンスまたはサーバーで実行します。

エラー: 続行するには **make** をインストールしてください

問題: インストーラスクリプトを実行すると、**Error: please install make to proceed** というメッセージが表示されます。

考えられる原因: **make** ユーティリティがインスタンスまたはサーバーにインストールされていません。

推奨される解決策: 代わりに **make** ユーティリティを使用して、インストーラスクリプトをインスタンスまたはサーバーで再度実行します。

make ユーティリティをインストールするには、次のようなコマンドの 1 つをインスタンスまたはサーバーで実行します。

- Amazon EC2 で実行されている Amazon Linux、Amazon Linux 2、および Red Hat Enterprise Linux (RHEL) の場合: **sudo yum -y groupinstall "Development Tools"**
- Amazon EC2 で実行されている Ubuntu Server の場合: **sudo apt install -y build-essential**
- SUSE の場合: **sudo zypper install -y make**

エラー: 続行するには **gcc** をインストールしてください

問題: インストーラスクリプトを実行すると、**Error: please install gcc to proceed** というメッセージが表示されます。

考えられる原因: **gcc** ユーティリティがインスタンスまたはサーバーにインストールされていません。

推奨される解決策: 代わりに **gcc** ユーティリティを使用して、インストーラスクリプトをインスタンスまたはサーバーで再度実行します。

gcc ユーティリティをインストールするには、次のようなコマンドの 1 つをインスタンスまたはサーバーで実行します。

- Amazon EC2 で実行されている Amazon Linux、Amazon Linux 2、および Red Hat Enterprise Linux (RHEL) の場合: **sudo yum -y groupinstall "Development Tools"**
- Amazon EC2 で実行されている Ubuntu Server の場合: **sudo apt install -y build-essential**
- SUSE の場合: **sudo zypper install -y gcc**
- 他のオペレーティングシステムについては、「[GCC のインストール](#)」を参照してください。

設定: エラー: カースが見つかりません

問題: インストーラスクリプトを実行すると、`configure: error: curses not found` というメッセージが表示されます。

考えられる原因: **ncurses** ターミナルコントロールライブラリがインスタンスまたはサーバーにインストールされていません。

推奨される解決策: **ncurses** ターミナルコントロールライブラリ (および一部のオペレーティングシステムでは **glibc-static** ライブラリ) をインストールし、インストーラスクリプトをインスタンスまたはサーバーで再度実行します。

ncurses ターミナルコントロールライブラリ (および一部のオペレーティングシステムでは **glibc-static** ライブラリ) をインストールするには、次のコマンドをインスタンスまたはサーバーで実行します。

- Amazon EC2 で実行されている Amazon Linux、Amazon Linux 2、および Red Hat Enterprise Linux (RHEL) の場合: **sudo yum -y install ncurses-devel**
- SUSE の場合: **sudo zypper install -y ncurses-devel** および **sudo zypper install -y glibc-static**

のインバウンド SSH IP アドレス範囲 AWS Cloud9

受信トラフィックは、AWS Cloud9 が SSH 経由で Amazon VPC の AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) またはネットワーク内の独自のサーバーに接続するために使用する IP アドレス範囲のみに制限できます。

Note

受信トラフィックは、AWS Cloud9 が SSH 経由で接続するために使用する IP アドレス範囲のみに制限できます。2018 年 7 月 31 日以降に作成した EC2 環境の場合、このトピックはスキップできます。これは、が、その環境のインバウンド SSH トラフィックを、このトピックで後述する IP アドレスのみ AWS Cloud9 に自動的に制限するためです。は、環境の Amazon EC2 インスタンスに関連付けられているセキュリティグループにルールを自動的に追加することでこれ AWS Cloud9 を行います。このルールは、ポート 22 経由のインバウンド SSH トラフィックを、関連付けられた AWS リージョンの IP アドレスのみに制限します。ネットワーク内の独自のサーバーについては、このトピックで後述する手順に従う必要があります。

ほとんどの AWS リージョンの IP アドレス範囲は、「」の[AWS「IP アドレス範囲」](#)で説明されているように、ip-ranges.json ファイルにありますAWS 全般のリファレンス。

Note

ip-ranges.json ファイルに現在含まれていないアジアパシフィック (香港)、欧州 (ミラノ)、中東 (バーレーン) リージョンの IP アドレスについては、[以下](#)を参照してください。

ip-ranges.json ファイル内の IP 範囲を見つけるには:

- Windows の場合は、を使用して AWS Tools for Windows PowerShell次のコマンドを実行します。

```
Get-AWSPublicIpAddressRange -ServiceKey CLOUD9
```

- Linux の場合は、[ip-ranges.json](#) ファイルをダウンロードします。次に、jq などのツールで、以下のコマンドを使用してファイルに対してクエリを実行できます。

```
jq '.prefixes[] | select(.service=="CLOUD9")' < ip-ranges.json
```

これらの IP 範囲は変更される場合があります。変更されるごとに、AmazonIpSpaceChanged トピックの受信者に通知が送信されます。これらの通知を取得するには、「AWS 全般のリファレンス」の「[AWS の IP アドレス範囲の通知](#)」を参照してください。

AWS クラウドコンピューティングインスタンスを使用する環境を設定するときにこれらの IP アドレス範囲を使用するには、「」を参照してください[AWS Cloud9 開発環境の VPC 設定](#)。また、Amazon Linux または Ubuntu Server で実行している Amazon EC2 インスタンスと関連づけられた EC2 環境または SSH 環境の着信トラフィックを制限を選択する場合は、少なくとも、ポート 32768〜61000 経由で TCP を使用するすべての IP アドレスも必ず許可してください。詳細と、他の AWS クラウドコンピューティングインスタンスタイプのポート範囲については、Amazon VPC ユーザーガイドの[一時ポート](#)を参照してください。

独自のネットワークを使用する SSH 環境を設定するときにこれらの IP アドレス範囲を使用するには、ネットワークのドキュメンテーションを参照するか、ネットワーク管理者に問い合わせてください。

ip-ranges.json にない IP アドレス

AWS Cloud9 現在、次の AWS リージョンの IP アドレス範囲は ip-ranges.json ファイルで提供されていません。アジアパシフィック (香港)、欧州 (ミラノ)、中東 (バーレーン)。次の表に、それらのリージョンの IP 範囲を示します。

Note

各リージョンには、AWS Cloud9 コントロールプレーン (情報ルーティング) サービスとデータプレーン (情報処理) サービスをサポートする 2 つの IP アドレス範囲があります。

AWS リージョン	コード	IP 範囲 (CIDR 表記)
アジアパシフィック (香港)	ap-east-1	18.163.201.96/27
		18.163.139.32/27
ヨーロッパ (ミラノ)	eu-south-1	15.161.135.64/27
		15.161.135.96/27
中東 (バーレーン)	me-south-1	15.185.141.160/27
		15.185.91.32/27

AWS Cloud9 EC2 開発環境の AMI コンテンツ

AWS Cloud9 が EC2 環境に使用する Amazon マシンイメージ (AMI) に関する詳細を取得するには、次の情報を使用します。

Important

環境の Amazon EC2 インスタンスが Amazon Linux 2023 AMI または Amazon Linux 2 AMI テンプレートに基づいている場合、セキュリティ更新は起動直後にインスタンスにインストールされます。さらに、セキュリティパッチは 1 時間ごとにインスタンスに自動的に適用されます。これらの更新はバックグラウンドプロセスによって適用され、インスタンスの使用には影響しません。

Ubuntu EC2 環境の場合、セキュリティ更新プログラムは起動直後にインスタンスにインストールされます。次に、unattended-upgrades パッケージは、利用可能な更新プログラムを毎日自動的にインストールします。

トピック

- [Amazon Linux 2023/Amazon Linux 2](#)
- [Ubuntu Server](#)

Amazon Linux 2023/Amazon Linux 2

Important

[コンソールを使用して Amazon EC2 環境を作成](#) するときは、[Amazon Linux 2023] オプションの選択をお勧めします。Amazon Linux 2023 AMI は、安全で安定した、高パフォーマンスなランタイム環境を提供するだけでなく、2024 年までの長期サポートも含まれています。

Amazon Linux インスタンスのバージョンを表示するには、接続された環境の AWS Cloud9 IDE または コマンドや PuTTY などの SSH ユーティリティから次の ssh コマンドを実行します。

```
cat /etc/system-release
```

Amazon Linux インスタンスにインストールされているパッケージのリストを表示するには、次のコマンドを 1 つ以上実行します。

インストールされているすべてのパッケージを単一のリストとして表示するには:

```
sudo yum list installed
```

指定したテキストがパッケージ名に含まれている、インストールされているパッケージのリストを表示するには:

```
sudo yum list installed | grep YOUR_SEARCH_TERM
```

前述のコマンドで、YOUR_SEARCH_TERM をパッケージ名の一部に置き換えます。たとえば、sql が名前に含まれている、インストールされているすべてのパッケージのリストを表示するには:

```
sudo yum list installed | grep sql
```

インストールされているすべてのパッケージリストを 1 度に 1 ページずつ表示するには:

```
sudo yum list installed | less
```

表示されたページをスクロールするには:

- 行を下に移動するには、**j** を押します。
- 行を上に移すには、**k** を押します。
- ページを下に移動するには、**Ctrl-F** を押します。
- ページを上に移すには、**Ctrl-B** を押します。
- 終了するには、**q** を押します。

Note

Amazon Linux 2 では、Extras Library を使用してアプリケーションおよびソフトウェア更新をインスタンスにインストールできます。このようなソフトウェア更新は、トピックと呼ばれます。詳細については、Amazon EC2 ユーザーガイドの [Extras Library \(Amazon Linux 2\)](#) を参照してください。

追加のオプションについては、`man yum` コマンドを実行してください。以下のリソースも参照してください。

- Amazon Linux 2023: [AMI ページ](#)。
- Amazon Linux: [Amazon Linux AMI 2018.03 パッケージ](#)。

Ubuntu Server

Ubuntu Server インスタンスのバージョンを表示するには、接続された環境の AWS Cloud9 IDE から、または ssh コマンドや PuTTY などの SSH ユーティリティから次のコマンドを実行します。

```
lsb_release -a
```

[説明] フィールドの横にバージョンが表示されます。

Ubuntu Server にインストールされているパッケージのリストを表示するには、以下の 1 つ以上のコマンドを実行します。

インストールされているすべてのパッケージを単一のリストとして表示するには:

```
sudo apt list --installed
```

指定したテキストがパッケージ名に含まれている、インストールされているパッケージのリストを表示するには:

```
sudo apt list --installed | grep YOUR_SEARCH_TERM
```

前述のコマンドで、YOUR_SEARCH_TERM をパッケージ名の一部に置き換えます。たとえば、sql が名前に含まれている、インストールされているすべてのパッケージのリストを表示するには:

```
sudo apt list --installed grep sql
```

インストールされているすべてのパッケージリストを 1 度に 1 ページずつ表示するには:

```
sudo apt list --installed | less
```

表示されたページをスクロールするには:

- 行を下に移動するには、**j** を押します。

- 行を上に移動するには、**k** を押します。
- ページを下に移動するには、**Ctrl-F** を押します。
- ページを上に移動するには、**Ctrl-B** を押します。
- 終了するには、**q** を押します。

追加のオプションについては、`man apt` コマンドを実行してください。Ubuntu ウェブサイトの「[Ubuntu パッケージ検索](#)」も参照してください。

のサービスにリンクされたロールの使用 AWS Cloud9

AWS Cloud9 は AWS Identity and Access Management (IAM) [サービスにリンクされたロール](#)を使用します。サービスにリンクされたロールは、直接リンクされた一意のタイプの IAM ロールです AWS Cloud9。サービスリンクロールは、AWS Cloud9 による事前定義済みのロールであり、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可を備えています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を追加する必要がないため、の設定 AWS Cloud9 が簡単になります。は、サービスにリンクされたロールのアクセス許可 AWS Cloud9 を定義し、のみがそのロールを引き受け AWS Cloud9 ることができます。定義されたアクセス許可には、信頼ポリシーとアクセス許可ポリシーが含まれ、そのアクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

ロールを削除するには、まず関連リソースを削除します。これにより、AWS Cloud9 リソースへのアクセス許可が誤って削除されないため、リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスについては、「[IAM と連携するAWS サービス](#)」を参照して、サービスにリンクされたロール列がはいになっているサービスを見つけてください。そのサービスに関するサービスにリンクされたロールのドキュメントを表示するには、リンクが設定されている Yes] (はい) を選択します。

- [AWS Cloud9のサービスにリンクされたロールのアクセス許可](#)
- [のサービスにリンクされたロールの作成 AWS Cloud9](#)
- [のサービスにリンクされたロールの編集 AWS Cloud9](#)
- [のサービスにリンクされたロールの削除 AWS Cloud9](#)
- [AWS Cloud9 サービスにリンクされたロールでサポートされているリージョン](#)

のサービスにリンクされたロールのアクセス許可 AWS Cloud9

AWS Cloud9 は、AWSServiceRoleForAWSCloud9 という名前のサービスにリンクされたロールを使用します。このサービスにリンクされたロールは、ロールを引き受ける上で `cloud9.amazonaws.com` サービスを信頼します。

このサービスにリンクされたロールのアクセス許可ポリシーは `AWSCloud9ServiceRolePolicy` という名前で、は指定されたリソースのポリシーにリストされているアクションを完了 AWS Cloud9 できます。

Important

License Manager を使用していて、`unable to access your environment` エラーが発生する場合は、古いサービスにリンクされたロールを License Manager をサポートするバージョンに置き換える必要があります。古いロールを削除するだけで置き換えることができます。その後、更新されたロールが自動的に作成されます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RunInstances",
        "ec2:CreateSecurityGroup",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResources"
      ],
      "Resource": "*"
    },
  ],
  {
```

```

    "Effect": "Allow",
    "Action": [
        "ec2:TerminateInstances",
        "ec2:DeleteSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:DeleteStack"
    ],
    "Resource": "arn:aws:cloudformation:*:*:stack/aws-cloud9-*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateTags"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringLike": {
            "aws:RequestTag/Name": "aws-cloud9-*"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "ec2:ResourceTag/aws:cloudformation:stack-name": "aws-cloud9-*"
        }
    }
},
{

```

```

    "Effect": "Allow",
    "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
    ],
    "Resource": [
        "arn:aws:license-manager:*:*:license-configuration:*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfiles",
        "iam:GetInstanceProfile"
    ],
    "Resource": [
        "arn:aws:iam:*:*:instance-profile/cloud9/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam:*:*:role/service-role/AWSCloud9SSMAccessRole"
    ],
    "Condition": {
        "StringLike": {
            "iam:PassedToService": "ec2.amazonaws.com"
        }
    }
}
]
}

```

が IAM エンティティ (ユーザー、グループ、ロールなど) に代わってサービスにリンクされたロールを作成 AWS Cloud9 できるようにするには、アクセス許可を設定する必要があります。

AWS Cloud9 が AWSServiceRoleForAWSCloud9 サービスにリンクされたロールを作成できるようにするには、サービスにリンクされたロールを作成 AWS Cloud9 する必要がある IAM エンティティのアクセス許可ポリシーに次のステートメントを追加します。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "iam:AWSServiceName": "cloud9.amazonaws.com"
    }
  }
}
```

または、AWS マネージドポリシー `AWSCloud9User` または `AWSCloud9Administrator` を IAM エンティティに追加できます。

`AWSServiceRoleForAWSCloud9` サービスにリンクされたロールの削除を IAM エンティティに許可するには、サービスリンクされたロールを削除する必要がある IAM エンティティの許可ポリシーに次のステートメントを追加します。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "iam:AWSServiceName": "cloud9.amazonaws.com"
    }
  }
}
```

のサービスにリンクされたロールの作成 AWS Cloud9

サービスにリンクされたロールを手動で作成する必要はありません。AWS Cloud9 開発環境を作成すると、 によってサービスにリンクされたロール `AWSCloud9` が作成されます。

のサービスにリンクされたロールの編集 AWS Cloud9

で AWSServiceRoleForAWSCloud9 サービスにリンクされたロールを編集することはできません AWS Cloud9。たとえば、サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、「IAM ユーザーガイド」の「[サービスリンクロールの編集](#)」を参照してください。

のサービスにリンクされたロールの削除 AWS Cloud9

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングまたはメンテナンスされることがなくなります。

IAM でのサービスにリンクされたロールの削除

IAM を使ってサービスにリンクされたロールを削除する前に、そのロールで使用されている任意の AWS Cloud9 リソースを削除する必要があります。AWS Cloud9 リソースを削除するには、「[環境の削除](#)」を参照してください。

IAM コンソールを使って、AWSServiceRoleForAWSCloud9 サービスにリンクされたロールを削除することができます。詳細については、「[IAM ユーザーガイド](#)」の「サービスリンクロールの削除」を参照してください。

AWS Cloud9 サービスにリンクされたロールでサポートされているリージョン

AWS Cloud9 は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートします。詳細については、「Amazon Web Services 全般のリファレンス」の「[AWS Cloud9](#)」を参照してください。

を使用した AWS Cloud9 API コールのログ記録 AWS CloudTrail

AWS Cloud9 は CloudTrail と統合されています。CloudTrail は、ユーザー、ロール、またはによって実行されたアクションを記録するサービス AWS のサービス です AWS Cloud9。CloudTrail は、のすべての API コールをイベント AWS Cloud9 としてキャプチャします。キャプチャされた呼び出しには、AWS Cloud9 コンソールからの呼び出しと、AWS Cloud9 APIs へのコード呼び出しが含まれます。証跡を作成する場合は、イベントを含む Amazon Simple Storage Service (Amazon S3) バケットへの CloudTrail イベントの継続的な配信を有効にすることができます AWS Cloud9。

証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、リクエストの実行元の IP アドレス AWS Cloud9、リクエストの実行者、リクエストの実行日時などの詳細を確認できます。

CloudTrail の詳細については、「[AWS CloudTrail ユーザーガイド](#)」を参照してください。

AWS Cloud9 CloudTrail の情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。でアクティビティが発生すると AWS Cloud9、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、[CloudTrail イベント履歴でのイベントの表示](#)を参照してください。

のイベントなど AWS アカウント、 のイベントの継続的な記録については AWS Cloud9、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- [証跡の作成のための概要](#)
- [CloudTrail がサポートするサービスと統合](#)
- [CloudTrail 用 Amazon SNS 通知の構成](#)
- [複数のリージョンから CloudTrail ログファイルを受け取る](#) および [複数のアカウントから CloudTrail ログファイルを受け取る](#)

AWS Cloud9 は、CloudTrail ログファイルのイベントとして次のアクションのログ記録をサポートします。

- CreateEnvironmentEC2
- CreateEnvironmentSSH
- CreateEnvironmentMembership
- DeleteEnvironment
- DeleteEnvironmentMembership
- DescribeEnvironmentMemberships

- DescribeEnvironments
- DescribeEnvironmentStatus
- ListEnvironments
- ListTagsForResource
- TagResource
- UntagResource
- UpdateEnvironment
- UpdateEnvironmentMembership

 Note

の一部の CloudTrail イベント AWS Cloud9 は、パブリック API オペレーションによって発生しません。代わりに、ユーザー認証とマネージド一時認証情報に影響を与える内部更新によって、次のイベントが開始されます。

- DisableManagedCredentialsByCollaborator
- EnvironmentTokenSuccessfullyCreated
- ManagedCredentialsUpdatedOnEnvironment

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management IAM ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「[CloudTrail userIdentity 要素](#)」を参照してください。

AWS Cloud9 ログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルには、ログエントリが 1 つ以上あります。イベントは、あら

ゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、およびリクエストパラメータに関する情報が含まれています。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

- [CreateEnvironmentEC2](#)
- [CreateEnvironmentSSH](#)
- [CreateEnvironmentMembership](#)
- [DeleteEnvironment](#)
- [DeleteEnvironmentMembership](#)
- [DescribeEnvironmentMemberships](#)
- [DescribeEnvironments](#)
- [DescribeEnvironmentStatus](#)
- [ListEnvironments](#)
- [ListTagsForResource](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateEnvironment](#)
- [UpdateEnvironmentMembership](#)

CreateEnvironmentEC2

以下の例は、CreateEnvironmentEC2 アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
```

```

        "creationDate": "2019-01-14T11:29:47Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2019-01-14T11:33:27Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "CreateEnvironmentEC2",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "instanceType": "t2.small",
    "subnetId": "subnet-1d4a9eEX",
    "description": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "dryRun": true,
    "automaticStopTimeMinutes": 30,
    "name": "my-test-environment",
    "clientRequestToken": "cloud9-console-f8e37272-e541-435d-a567-5c684EXAMPLE"
  },
  "responseElements": null,
  "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
  "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]
}

```

CreateEnvironmentSSH

以下の例は、CreateEnvironmentSSH アクションを示す CloudTrail ログエントリです。

```

{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",

```

```

    "userName": "MyUser",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-01-14T11:29:47Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2019-01-14T11:33:27Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "CreateEnvironmentSSH",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "host": "198.51.100.0",
    "port": 22,
    "name": "my-ssh-environment",
    "description": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "clientRequestToken": "cloud9-console-b015a0e9-469e-43e3-be90-6f432EXAMPLE",
    "loginName": "ec2-user"
  },
  "responseElements": {
    "environmentId": "5c39cc4a85d74a8bbb6e23ed6EXAMPLE"
  },
  "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
  "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]
}
```

CreateEnvironmentMembership

以下の例は、CreateEnvironmentMembership アクションを示す CloudTrail ログエントリです。

```

{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
```

```

    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/MyUser",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "MyUser",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-01-14T11:29:47Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2019-01-14T11:33:27Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "CreateEnvironmentMembership",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
    "userArn": "arn:aws:iam::111122223333:user/MyUser",
    "permissions": "read-write"
  },
  "responseElements": {
    "membership": {
      "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
      "permissions": "read-write",
      "userId": "AIDACKCEVSQ6C2EXAMPLE",
      "userArn": "arn:aws:iam::111122223333:user/MyUser"
    }
  },
  "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
  "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]
}

```

DeleteEnvironment

以下の例は、DeleteEnvironment アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2019-01-14T11:29:47Z"
          }
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    {
      "eventTime": "2019-01-14T11:33:27Z",
      "eventSource": "cloud9.amazonaws.com",
      "eventName": "DeleteEnvironment",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE"
      },
      "responseElements": null,
      "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
      "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```

DeleteEnvironmentMembership

以下の例は、DeleteEnvironmentMembership アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2019-01-14T11:29:47Z"
          }
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    {
      "eventTime": "2019-01-14T11:33:27Z",
      "eventSource": "cloud9.amazonaws.com",
      "eventName": "DeleteEnvironmentMembership",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
        "userArn": "arn:aws:iam::111122223333:user/MyUser",
      },
      "responseElements": null,
      "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
      "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```


DescribeEnvironmentMemberships

以下の例は、DescribeEnvironmentMemberships アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2019-01-14T11:29:47Z"
          }
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    {
      "eventTime": "2019-01-14T11:33:27Z",
      "eventSource": "cloud9.amazonaws.com",
      "eventName": "DescribeEnvironmentMemberships",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "nextToken": "NEXT_TOKEN_EXAMPLE",
        "permissions": [ "owner" ],
        "maxResults": 15
      },
      "responseElements": null,
      "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
      "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```

```
}
```

DescribeEnvironments

以下の例は、DescribeEnvironments アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2019-01-14T11:29:47Z"
          }
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    {
      "eventTime": "2019-01-14T11:33:27Z",
      "eventSource": "cloud9.amazonaws.com",
      "eventName": "DescribeEnvironments",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "environmentIds": [
          "2f5ff70a640f49398f67e3bdeb811ab2"
        ]
      },
      "responseElements": null,
      "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
      "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```

```
]
}
```

DescribeEnvironmentStatus

以下の例は、DescribeEnvironmentStatus アクションを示す CloudTrail ログエントリです。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:sts::123456789012:myuser_role",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:sts::123456789012:myuser_role",
        "accountId": "123456789012",
        "userName": "barshane_role"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-03-12T15:10:54Z"
      }
    }
  },
  "eventTime": "2021-03-12T15:13:31Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "DescribeEnvironmentStatus",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "XX.XX.XXX.XX",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.951
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.282-b08
java/1.8.0_282 vendor/Oracle_Corporation",
  "requestParameters": {
    "environmentId": "31ea8a12746a4221b7d8e07d9ef6ee21"
  },
  "responseElements": null,
  "requestID": "68b163fb-aa88-4f40-bafd-4a18bf24cbd5",
```

```
"eventID": "c0fc52a9-7331-4ad0-a8ee-157995dfb5e6",
"readOnly": true,
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "123456789012"
}
```

ListEnvironments

以下の例は、ListEnvironments アクションを示す CloudTrail ログエントリです。

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "MyUser",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2019-01-14T11:29:47Z"
          }
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    {
      "eventTime": "2019-01-14T11:33:27Z",
      "eventSource": "cloud9.amazonaws.com",
      "eventName": "ListEnvironments",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "signin.amazonaws.com",
      "requestParameters": {
        "nextToken": "NEXT_TOKEN_EXAMPLE",
        "maxResults": 15
      },
      "responseElements": null,
      "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
    }
  ]
}
```

```

    "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }
]
}
```

ListTagsForResource

以下の例は、ListTagsForResource アクションを示す CloudTrail ログエントリです。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:sts::123456789012:myuser_role",
    "accountId": "123456789012",
    "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "123456789012:myuser_role",
        "accountId": "123456789012",
        "userName": "barshane_role"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-03-23T16:41:51Z"
      }
    }
  },
  "eventTime": "2021-03-23T16:42:58Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "ListTagsForResource",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "XX.XX.XXX.XX",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.976
Linux/4.9.230-0.1.ac.224.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.282-b08
java/1.8.0_282 vendor/Oracle_Corporation cfg/retry-mode/legacy",
```

```

    "requestParameters": {
      "resourceARN": "arn:aws:cloud9:us-east-1:123456789012:environment:3XXXXXXXXX6a4221b7d8e07d9ef6ee21"
    },
    "responseElements": {
      "tags": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "requestID": "5750a344-8462-4020-82f9-f1d500a75162",
    "eventID": "188d572d-9a14-4082-b98b-0389964c7c30",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
  }
}

```

TagResource

以下の例は、TagResource アクションを示す CloudTrail ログエントリです。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:sts::123456789012:myuser_role",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/myuser_role",
        "accountId": "123456789012",
        "userName": "MyUser"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-03-23T15:03:57Z"
      }
    }
  },
  "eventSource": "iam.amazonaws.com",
  "eventName": "TagResource",
  "eventTime": "2021-03-23T15:03:57Z",
  "eventSeverity": "INFO",
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012",
  "sourceIPAddress": "192.168.1.1",
  "userAgent": "aws-cli/2.0.0 Python/3.7.7 Linux/4.18.20-1.el8.x86_64 botocore/2.0.0"
}

```

```

    "eventTime": "2021-03-23T15:08:16Z",
    "eventSource": "cloud9.amazonaws.com",
    "eventName": "TagResource",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "54.XXX.XXX.XXX",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.976
Linux/4.9.230-0.1.ac.224.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.282-b08
java/1.8.0_282 vendor/Oracle_Corporation cfg/retry-mode/legacy",
    "requestParameters": {
        "resourceARN": "arn:aws:cloud9:us-
east-1:123456789012:environment:3XXXXXXXXX6a4221b7d8e07d9ef6ee21",
        "tags": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "responseElements": null,
    "requestID": "658e9d70-91c2-41b8-9a69-c6b4cc6a9456",
    "eventID": "022b2893-73d1-44cb-be6f-d3faa68e83b1",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}

```

UntagResource

以下の例は、UntagResource アクションを示す CloudTrail ログエントリです。

```

{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:sts::123456789012/MyUser",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:MyUser",
                "accountId": "123456789012",
                "userName": "MyUser"
            }
        }
    },

```

```

        "webIdFederationData": {},
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2021-03-23T15:58:36Z"
        }
    },
    "eventTime": "2021-03-23T16:05:08Z",
    "eventSource": "cloud9.amazonaws.com",
    "eventName": "UntagResource",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.XX.XX.XXX",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.976
Linux/4.9.230-0.1.ac.224.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.282-b08
java/1.8.0_282 vendor/Oracle_Corporation cfg/retry-mode/legacy",
    "requestParameters": {
        "resourceARN": "arn:aws:cloud9:us-
east-1:123456789012:environment:3XXXXXXXXX6a4221b7d8e07d9ef6ee21",
        "tagKeys": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "responseElements": null,
    "requestID": "0eadaef3-dc0a-4cd7-85f6-135b8529f75f",
    "eventID": "41f2f2e2-4b17-43d4-96fc-9857981ca1de",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}

```

UpdateEnvironment

以下の例は、UpdateEnvironment アクションを示す CloudTrail ログエントリです。

```

{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",

```



```

    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "MyUser",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-01-14T11:29:47Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2019-01-14T11:33:27Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "UpdateEnvironment",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
    "description": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "name": "my-test-environment-renamed"
  },
  "responseElements": null,
  "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
  "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]
}

```

UpdateEnvironmentMembership

以下の例は、UpdateEnvironmentMembership アクションを示す CloudTrail ログエントリです。

```

{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/MyUser",
        "accountId": "111122223333",

```

```

    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "MyUser",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-01-14T11:29:47Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2019-01-14T11:33:27Z",
  "eventSource": "cloud9.amazonaws.com",
  "eventName": "UpdateEnvironmentMembership",
  "awsRegion": "us-west-2",
  "sourceIpAddress": "192.0.2.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
    "userArn": "arn:aws:iam::111122223333:user/MyUser",
    "permissions": "read-only"
  },
  "responseElements": {
    "membership": {
      "environmentId": "2f5ff70a640f49398f67e3bdeEXAMPLE",
      "permissions": "read-only",
      "userId": "AIDACKCEVSQ6C2EXAMPLE",
      "userArn": "arn:aws:iam::111122223333:user/MyUser"
    }
  },
  "requestID": "f0e629fb-fd37-49bd-b2cc-e9822EXAMPLE",
  "eventID": "8a906445-1b2a-47e9-8d7c-5b242EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]]

```

タグ

タグは、AWS リソースにア AWS タッチするラベルまたは属性です。各タグは、キーと、それとペアになる値で構成されます。[「IAM ユーザーガイド」](#)の AWS Cloud9 「リソース [タグを使用したアクセスの制御](#)」で説明されているように、[タグを使用して AWS](#) リソースへのアクセスを制御できま

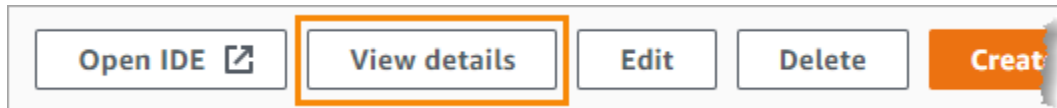
す。タグは、「[ユーザー定義のコスト配分タグ](#)」で説明されているように、請求情報の管理にも役立ちます。

[AWS Cloud9 EC2 開発環境を作成する](#)と、AWS Cloud9 には環境を管理するために必要な特定のシステムタグが含まれます。システムタグは「aws:」で始まります。その作成プロセス中に、独自のリソースタグを追加することもできます。

環境を作成したら、環境に添付されているタグの表示、環境への新しいリソースタグの追加、以前に追加したタグの変更または削除を行うことができます。AWS Cloud9 環境に最大 50 個のユーザー定義タグをアタッチできます。

次の方法の 1 つまたは複数を使用して、タグを表示または更新します。

- [AWS Cloud9 コンソール](#)で、目的の環境を選択し、[View Details (詳細を表示)] を選択します。



- 、[list-tags-for-resource](#)、[tag-resource](#)の AWS Cloud9 CLI コマンドを使用します [untag-resource](#)。
- 次の AWS Cloud9 API アクションを使用します: [ListTagsForResource](#)、[TagResource](#)、および [UntagResource](#)。

Warning

上記の方法 AWS Cloud9 を使用して 用に作成または更新したタグは、基盤となるリソースに自動的に伝達されません。これを行う方法については、次のセクション「[基礎となるリソースへのタグ更新の伝播](#)」を参照してください。

基礎となるリソースへのタグ更新の伝播

AWS Cloud9 CLI コマンドまたは API アクションを使用して AWS Cloud9 環境にアタッチされているタグを追加、変更、または削除する場合、それらの変更は CloudFormation スタック、Amazon EC2 インスタンス、Amazon EC2 セキュリティグループなどの基盤となるリソースに自動的に伝達されません。これらの変更は、手動で反映する必要があります。

次の手順をより使いやすくするために、対象の環境の環境 ID を取得できます。これをしたい場合は、以下の手順に従います。

1. [AWS Cloud9 コンソール](#)で、目的の環境を選択し、[View Details (詳細を表示)] を選択します。
2. [Environment ARN (環境 ARN)] プロパティを探し、環境 ID を記録します。環境 ID は、環境 ARN の一部であり、「environment:」の後ろにあります。

タグの使用目的に応じて、次の 1 つ以上の場所にタグの更新を伝達する必要があります。

CloudFormation スタックへのタグ更新の伝播

Note

タグを AWS CloudFormation スタックに更新すると、それらの更新はスタックに関連付けられている Amazon EC2 インスタンスと Amazon EC2 セキュリティグループに自動的に伝播されます。

1. [CloudFormation コンソール](#)に移動します。
2. 関心のある AWS Cloud9 環境に対応するスタックを見つけて選択します。環境 ID を記録した場合は、その環境 ID を使用して環境をフィルタリングできます。
3. [スタック情報] タブの [タグ] セクションで、タグのリストを確認します。
4. タグを更新する必要がある場合は、ページ上部にある [更新] を選択し、指示に従います。詳細については、「[AWS CloudFormation ユーザーガイド](#)」の「[スタックの直接更新](#)」を参照してください。

[describe-stacks](#) および [update-stack](#) CLI コマンドを使用してタグを更新することもできます。

Amazon EC2 インスタンスへのタグ更新の伝播

1. [Amazon EC2 インスタンス](#) コンソールに移動します。
2. 関心のある AWS Cloud9 環境に対応する Amazon EC2 インスタンスを見つけて選択します。以前に環境 ID を記録した場合は、それを使用して環境をフィルタリングできます。
3. [タグ] タブで、必要に応じてタグを表示および更新します。

また、[describe-tags](#)、[create-tags](#)、および [delete-tags](#) CLI コマンドを使用してタグを更新することもできます。

Amazon EC2 セキュリティグループへのタグ更新の伝播

1. [Amazon EC2 セキュリティグループ](#) コンソールに移動します。
2. 関心のある AWS Cloud9 環境に対応するセキュリティグループを見つけて選択します。以前に環境 ID を記録した場合は、それを使用して環境をフィルタリングできます。
3. [タグ] タブを開き、必要に応じてタグを表示および更新します。

また、[describe-tags](#)、[create-tags](#)、および [delete-tags](#) CLI コマンドを使用してタグを更新することもできます。

のセキュリティ AWS Cloud9

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。お客様は AWS、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。セキュリティは、AWS お客様とお客様の間の責任共有です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ – AWS クラウドで提供されているすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。における当社のセキュリティ責任は最優先事項であり AWS、当社のセキュリティの有効性は、[AWS コンプライアンスプログラムの一環としてサードパーティーの監査者によって定期的にテストおよび検証されています](#)。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、データの機密性、組織の要件、適用される法律や規制などのその他の要因によって決まります。

AWS Cloud9 は、サポートする特定の AWS サービスを通じて[責任共有モデル](#)に従います。AWS サービスセキュリティ情報については、[AWS 「サービスセキュリティドキュメント」ページ](#)と[AWS、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービス](#)を参照してください。

以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成する AWS Cloud9 ようにを設定する方法を示します。

トピック

- [でのデータ保護 AWS Cloud9](#)
- [の Identity and Access Management AWS Cloud9](#)
- [でのログ記録とモニタリング AWS Cloud9](#)
- [のコンプライアンス検証 AWS Cloud9](#)
- [の耐障害性 AWS Cloud9](#)
- [のインフラストラクチャセキュリティ AWS Cloud9](#)
- [ソフトウェアの更新プログラムと修正プログラム](#)
- [のセキュリティのベストプラクティス AWS Cloud9](#)

でのデータ保護 AWS Cloud9

責任 AWS [共有モデル](#)、でのデータ保護に適用されます AWS Cloud9。このモデルで説明されているように、AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、[データプライバシーに関するよくある質問](#)を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「[AWS 責任共有モデルおよび GDPR](#)」のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント、AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします：

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の[CloudTrail 証跡の使用](#)を参照してください。
- AWS 暗号化ソリューションと、内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-3](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS CLI AWS Cloud9 または他の AWS のサービス を操作する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

データ暗号化

データ暗号化とは、と AWS アカウントの間 AWS Cloud9 を移動するとき、および AWS Cloud9 設定ストアや AWS クラウドコンピューティングインスタンスに保存されている保管中のデータを保護することです。

これに関連して AWS Cloud9、次のタイプのデータでは、暗号化による保護が必要になる場合があります。

コンテンツとデータ

ユーザーが操作、収集、保存する情報。このタイプのデータの例を次に示します。

- コードファイル
- 添付された EC2 環境または SSH 環境の設定、アプリケーション、およびデータ

AWS Cloud9 メタデータ

AWS Cloud9 を操作、収集、保存するデータ。このタイプのデータの例を次に示します。

- タブの状態、開いているファイル、IDE 優先などの IDE 設定
- AWS Cloud9 環境名や説明などの開発環境メタデータ
- AWS Cloud9 サービス API、コンソールログ
- HTTP リクエストなどのサービスログ

AWS Cloud9 は、データプレーンサービスを通じてコンテンツとデータの一部も送信します。これには、ファイル、ターミナル入力、出力テキスト、一部の IDE コマンド (ファイルの保存など) が含まれます。

保管中の暗号化

保存時の暗号化とは、保存中にデータを暗号化することで、不正なアクセスからデータを保護することです。コードファイル、パッケージ、依存関係などの AWS Cloud9 環境に保存された顧客データは、常に顧客のリソースに保存されます。顧客が Amazon EC2 環境を使用している場合、データは AWS アカウントに存在する関連する Amazon Elastic Block Store (Amazon EBS) ボリュームに保存されます。お客様が SSH 環境を使用している場合、データは Linux サーバーのローカルストレージに保存されます。

AWS Cloud9 開発環境用に Amazon EC2 インスタンスが作成されると、暗号化されていない Amazon EBS ボリュームが作成され、そのインスタンスにアタッチされます。データを暗号化するには、暗号化された EBS ボリュームを作成して EC2 インスタンスにアタッチする必要があります。AWS Cloud9 およびアタッチされた Amazon EBS ボリュームは、デフォルトでリージョン固有の設定である Amazon EBS のデフォルトの暗号化をサポートしています。詳細については、AWS Elastic Compute Cloud ユーザーガイドの「[デフォルトでの暗号化](#)」を参照してください。

環境名、環境のメンバー、IDE 設定など、AWS Cloud9 開発環境に関するメタデータは、お客様のリソースではなく AWSによって保存されます。環境の説明や IDE 設定など、お客様固有の情報は暗号化されます。

転送中の暗号化

転送中の暗号化とは、通信エンドポイント間の移動中にデータが傍受されるのを防ぐことです。顧客のクライアントと AWS Cloud9 サービス間で送信されるすべてのデータは、HTTPS、WSS、および暗号化された SSH によって暗号化されます。

- HTTPS – お客様のウェブブラウザと AWS Cloud9 サービス間の安全なリクエストを確保します。AWS Cloud9 また、は、お客様のブラウザから HTTPS 経由で送信された Amazon CloudFront からアセットをロードします。
- WSS (WebSocket Secure) –お客様のウェブブラウザと AWS Cloud9 サービス間の WebSocket を介した安全な双方向通信を可能にします。
- 暗号化された SSH (Secure Shell): クライアントのウェブブラウザと AWS Cloud9 サービス間の安全なデータ転送を有効にします。

HTTPS、WSS、および SSH プロトコルの使用は、でサポートされているブラウザを使用する によって異なります AWS Cloud9。「[でサポートされているブラウザ AWS Cloud9](#)」を参照してください。

Note

暗号化プロトコルは、AWS Cloud9でデフォルトで実装されています。お客様は、転送中の暗号化設定を変更することはできません。

キー管理

AWS Key Management Service (AWS KMS) は AWS KMS keys、顧客のデータを暗号化するために使用される暗号化キーを作成および制御するためのマネージドサービスです。は、顧客に代わってデータを暗号化するための暗号化キー AWS Cloud9 を生成および管理します。

ネットワーク間のトラフィックのプライバシー

SSH 環境は、オンプレミスのお客様所有のコンピューティングとストレージに接続します。暗号化された SSH、HTTPS、および WSS 接続は、サービスと SSH 環境間のデータ転送をサポートします。

特定の VPCs とサブネット内で起動するように AWS Cloud9 EC2 開発環境 (Amazon EC2 インスタンスによってバックアップ) を設定できます。Amazon Virtual Private Cloud 設定の詳細については、[AWS Cloud9 開発環境の VPC 設定](#) を参照してください。

の Identity and Access Management AWS Cloud9

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS Cloud9 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで利用できる AWS のサービス です。

トピック

- [オーディエンス](#)
- [アイデンティティを使用した認証](#)
- [ポリシーを使用したアクセスの管理](#)
- [が IAM と AWS Cloud9 連携する方法](#)
- [のアイデンティティベースのポリシーの例 AWS Cloud9](#)
- [AWS Cloud9 ID とアクセスのトラブルシューティング](#)
- [AWS Cloud9 が IAM リソースとオペレーションと連携する方法](#)
- [AWS の マネージドポリシー AWS Cloud9](#)
- [のカスタマー管理ポリシーの作成 AWS Cloud9](#)
- [AWS Cloud9 アクセス許可リファレンス](#)
- [AWS マネージド一時認証情報](#)

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「[AWS Cloud9 ID とアクセスのトラブルシューティング](#)」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「[が IAM と AWS Cloud9 連携する方法](#)」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「[のアイデンティティベースのポリシーの例 AWS Cloud9](#)」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーテッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「[AWS アカウントにサインインする方法](#)」を参照してください。

プログラムによるアクセスの場合、は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「[API リクエストに対するAWS 署名バージョン 4](#)」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーション ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「[IAM アイデンティティセンターとは](#)」を参照してください。

IAM ユーザーとグループ

[IAM ユーザー](#)は、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「[ID プロバイダーとのフェデレーションを使用して にアクセスすることを人間のユーザーに要求する AWS](#)」を参照してください。

[IAM グループ](#)は、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「[IAM ユーザーに関するユースケース](#)」を参照してください。

IAM ロール

[IAM ロール](#)は、特定のアクセス許可を持つアイデンティティであり、一時的な認証情報を提供します。ユーザーから [IAM ロール \(コンソール\) に切り替えるか、または API オペレーションを呼び出すことで、ロール](#)を引き受けることができます。AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「[ロールを引き受けるための各種方法](#)」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイドの [IAM でのクロスアカウントリソースアクセス](#) を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられている場合のアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「[JSON ポリシー概要](#)」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイドの [カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する](#) を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「[管理ポリシーとインラインポリシーのいずれかを選択する](#)」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、[プリンシパルを指定する](#)必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「[IAM エンティティのアクセス許可境界](#)」を参照してください。

- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「[サービスコントロールポリシー](#)」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「[リソースコントロールポリシー \(RCP\)](#)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「[セッションポリシー](#)」を参照してください。

複数のポリシータイプ

1つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「[ポリシー評価ロジック](#)」を参照してください。

が IAM と AWS Cloud9 連携する方法

IAM を使用して へのアクセスを管理する前に AWS Cloud9、使用できる IAM 機能を確認してください AWS Cloud9。

で使用できる IAM 機能 AWS Cloud9

IAM 機能	AWS Cloud9 サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし

IAM 機能	AWS Cloud9 サポート
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
転送アクセスセッション (FAS)	あり
サービスロール	あり
サービスリンクロール	はい

AWS Cloud9 およびその他の AWS のサービスがほとんどの IAM 機能とどのように連携するかの概要については、「IAM ユーザーガイド」の[AWS 「IAM と連携する のサービス」](#)を参照してください。

アイデンティティベースのポリシー AWS Cloud9

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「[カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する](#)」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用するすべての要素について学ぶには、「IAM ユーザーガイド」の「[IAM JSON ポリシーの要素のリファレンス](#)」を参照してください。

アイデンティティベースのポリシーの例 AWS Cloud9

AWS Cloud9 アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいの[アイデンティティベースのポリシーの例 AWS Cloud9](#)。

内のリソースベースのポリシー AWS Cloud9

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、[プリンシパルを指定する](#)必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、またはを含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドの[IAM でのクロスアカウントリソースアクセス](#)を参照してください。

AWS Cloud9 はリソースベースのポリシーをサポートしていませんが、AWS Cloud9 API と IDE AWS Cloud9 を介して AWS Cloud9 環境メンバーの AWS Cloud9 環境リソースのアクセス許可を制御できます。

のポリシーアクション AWS Cloud9

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの Action 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS Cloud9 アクションのリストを確認するには、「サービス認可リファレンス」の「[で定義されるアクション AWS Cloud9](#)」を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス AWS Cloud9 を使用します。

```
account
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
```



```
"account:action1",  
"account:action2"  
]
```

AWS Cloud9 アイデンティティベースのポリシーの例を表示するには、「」を参照してください[のアイデンティティベースのポリシーの例 AWS Cloud9](#)。

のポリシーリソース AWS Cloud9

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、[Amazon リソースネーム \(ARN\)](#) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

AWS Cloud9 リソースタイプとその ARNs [「で定義されるリソース AWS Cloud9」](#) を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「[AWS Cloud9で定義されるアクション](#)」を参照してください。

AWS Cloud9 アイデンティティベースのポリシーの例を表示するには、「」を参照してください[のアイデンティティベースのポリシーの例 AWS Cloud9](#)。

のポリシー条件キー AWS Cloud9

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの[条件演算子](#)を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」の[AWS 「グローバル条件コンテキストキー」](#)を参照してください。

AWS Cloud9 条件キーのリストを確認するには、「サービス認可リファレンス」の「[の条件キー AWS Cloud9](#)」を参照してください。条件キーを使用できるアクションとリソースについては、「[で定義されるアクション AWS Cloud9](#)」を参照してください。

AWS Cloud9 アイデンティティベースのポリシーの例を表示するには、「」を参照してください。[のアイデンティティベースのポリシーの例 AWS Cloud9](#)。

ACLs AWS Cloud9

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

を使用した ABAC AWS Cloud9

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの[条件要素](#)でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「[ABAC 認可でアクセス許可を定義する](#)」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「[属性ベースのアクセスコントロール \(ABAC\) を使用する](#)」を参照してください。

での一時的な認証情報の使用 AWS Cloud9

一時的な認証情報のサポート: あり

一時的な認証情報は、AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスウィッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「[IAM の一時的な認証情報](#)」および「[AWS のサービスと IAM との連携](#)」を参照してください。

の転送アクセスセッション AWS Cloud9

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。

のサービスロール AWS Cloud9

サービスロールのサポート: あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの [AWS のサービスに許可を委任するロールを作成する](#)を参照してください。

Warning

サービスロールのアクセス許可を変更すると、AWS Cloud9 機能が破損する可能性があります。AWS Cloud9 が指示する場合にのみ、サービスロールを編集します。

のサービスにリンクされたロール AWS Cloud9

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービ

スにリンクされたロールは に表示され AWS アカウント、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「[IAM と提携するAWS のサービス](#)」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

のアイデンティティベースのポリシーの例 AWS Cloud9

デフォルトでは、ユーザーおよびロールには、AWS Cloud9 リソースを作成または変更する権限はありません。IAM 管理者は、リソースに必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーを作成する \(コンソール\)](#)」を参照してください。

各リソースタイプの ARN の形式など AWS Cloud9、で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「[のアクション、リソース、および条件キー AWS Cloud9](#)」を参照してください。ARNs

トピック

- [ポリシーに関するベストプラクティス](#)
- [AWS Cloud9 コンソールの使用](#)
- [自分の権限の表示をユーザーに許可する](#)

ポリシーに関するベストプラクティス

ID ベースのポリシーは、誰かがアカウント内の AWS Cloud9 リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS

管理ポリシーを使用します。これらはで使えます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの [AWS マネージドポリシー](#) または [ジョブ機能のAWS マネージドポリシー](#) を参照してください。

- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイドの [IAM でのポリシーとアクセス許可](#) を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションがなどの特定のを通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON ポリシー要素:条件](#) を参照してください。
- IAM アクセスアナライザーを使用して IAM ポリシーを検証し、安全で機能的な権限を確保する – IAM アクセスアナライザーは、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイドの [IAM Access Analyzer でポリシーを検証する](#) を参照してください。
- 多要素認証 (MFA) を要求する – IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイドの [MFA を使用した安全な API アクセス](#) を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイドの [IAM でのセキュリティのベストプラクティス](#) を参照してください。

AWS Cloud9 コンソールの使用

AWS Cloud9 コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、の AWS Cloud9 リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみ呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが AWS Cloud9 引き続きコンソールを使用できるようにするには、エンティティに AWS Cloud9 *ConsoleAccess* または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「[ユーザーへのアクセス許可の追加](#)」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
  "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": "*"    
  }  
]  
}
```

AWS Cloud9 ID とアクセスのトラブルシューティング

次の情報は、 および IAM の使用時に発生する可能性がある一般的な問題の診断 AWS Cloud9 と修正に役立ちます。

トピック

- [でアクションを実行する権限がありません AWS Cloud9](#)
- [iam:PassRole を実行する権限がありません](#)
- [自分の 以外のユーザーに自分の AWS Cloud9 リソース AWS アカウント へのアクセスを許可したい](#)

でアクションを実行する権限がありません AWS Cloud9

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な *awes:GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
awes:GetWidget on resource: my-example-widget
```

この場合、*awes:GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS Cloud9にロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS Cloud9でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに自分の AWS Cloud9 リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS Cloud9 をサポートしているかどうかを確認するには、「」を参照してください [が IAM と AWS Cloud9 連携する方法](#)。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの [「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」](#) を参照してください。

- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「[サードパーティー AWS アカウント が所有する へのアクセスを提供する](#)」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの [外部で認証されたユーザー \(ID フェデレーション\) へのアクセスの許可](#) を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイドの [IAM でのクロスアカウントのリソースへのアクセス](#) を参照してください。

AWS Cloud9 が IAM リソースとオペレーションと連携する方法

AWS Identity and Access Management は、AWS Cloud9 開発環境と他の AWS のサービス およびリソースの両方を操作できるアクセス許可を管理するために使用されます。

AWS Cloud9 リソースとオペレーション

では AWS Cloud9、プライマリリソースは AWS Cloud9 開発環境です。ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。以下の表に環境 ARN を示します。詳細については、Amazon Web Services 全般のリファレンスの「[Amazon リソースネーム \(ARN\) と AWS サービス ネームスペース](#)」を参照してください。

リソースタイプ	ARN 形式
環境	<code>arn:aws:cloud9: <i>REGION_ID</i> :<i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i></code>
特定 AWS リージョンの特定アカウントが所有するすべての環境	<code>arn:aws:cloud9: <i>REGION_ID</i> :<i>ACCOUNT_ID</i> :environment:*</code>
特定のリージョンの特定アカウントが所有するすべての環境	<code>arn:aws:cloud9: <i>REGION_ID</i> :<i>ACCOUNT_ID</i> :*</code>
アカウントとリージョンに関係なく、すべての AWS Cloud9 リソース	<code>arn:aws:cloud9:*</code>

たとえば、以下の要領で Amazon リソースネーム (ARN) を使用して、ステートメント内で特定の環境を指定することができます。

```
"Resource": "arn:aws:cloud9:us-east-2:123456789012:environment:70d899206236474f9590d93b7c41dfEX"
```

すべてのリソースを指定するには、Resource エlement 内でワイルドカード文字 (*) を使用します。

```
"Resource": "*"
```

単一のステートメントに複数のリソースを指定するには、コンマで Amazon リソースネーム (ARN) を区切ります。

```
"Resource": [  
  "arn:aws:cloud9:us-east-2:123456789012:environment:70d899206236474f9590d93b7c41dfEX",  
  "arn:aws:cloud9:us-east-2:123456789012:environment:81e900317347585a0601e04c8d52eaEX"  
]
```

AWS Cloud9 には、AWS Cloud9 リソースを操作するための一連のオペレーションが用意されています。リストについては、「[AWS Cloud9 アクセス許可リファレンス](#)」を参照してください。

リソース所有権についての理解

AWS アカウント アカウントは、リソースを作成したユーザーに関係なく、アカウントで作成されたリソースを所有します。

以下のユースケースとシナリオについて検討します。

- のルートアカウントの認証情報を使用して AWS Cloud9 開発環境 AWS アカウント を作成するとします。これは可能ですがお勧めしません。この場合、AWS アカウント は環境の所有者です。
- で IAM ユーザーを作成し AWS アカウント、そのユーザーに環境を作成するアクセス許可を付与するとします。これで、ユーザーは環境を作成できます。ただし、ユーザーが属 AWS アカウントする は引き続き環境を所有します。
- 環境を作成するアクセス許可 AWS アカウント を持つに IAM ロールを作成するとします。これで、ロールを引き受けることのできるすべてのユーザーが環境を作成できます。ロールが属する AWS アカウントが環境を所有しているとします。

Note

1 つ以上の AWS Cloud9 環境の ARN 所有者であるユーザーアカウントを削除すると、これらの環境には所有者がいません。このシナリオの回避策は、AWS Cloud9 SDK を使用して、`CreateEnvironmentMembership` アクションと `EnvironmentMember` データ型を使用して読み取りおよび書き込み権限を持つ別の IAM ユーザーを追加することです。この IAM ユーザーを追加したら、環境ファイルを新しい AWS Cloud9 環境にコピーし、この所有者を ARN 所有者にすることができます。このアクションの詳細については、「[CreateEnvironmentMembership](#)」を参照してください。このデータ型の詳細については、「AWS Cloud9 API リファレンスガイド」の「[EnvironmentMember](#)」を参照してください。

リソースへのアクセスの管理

許可ポリシーでは、誰がどのリソースにアクセスできるかを記述します。

Note

このセクションでは、AWS Cloud9での IAM の使用について説明します。ここでは、IAM サービスに関する詳細情報を提供しません。完全な IAM ドキュメンテーションについては、IAM ユーザーガイドの「[IAM とは](#)」を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの [IAM JSON ポリシーリファレンス](#) を参照してください。

IAM アイデンティティに添付されているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースにアタッチされたポリシーは、リソースベースのポリシーと呼ばれます。は、アイデンティティベースのポリシーとリソースベースのポリシーの両方 AWS Cloud9 をサポートします。

以下の各 API アクションでは、これらの API アクションを呼び出したい IAM アイデンティティに添付する必要があるのは IAM ポリシーのみです。

- `CreateEnvironmentEC2`
- `DescribeEnvironments`

次の API アクションではリソースベースのポリシーが必要です。IAM ポリシーは必要ありませんが、これらの API アクションを呼び出す IAM ID にアタッチされている場合は IAM ポリシー AWS

Cloud9 を使用します。リソースベースのポリシーは、目的の AWS Cloud9 リソースに適用する必要があります。

- `CreateEnvironmentMembership`
- `DeleteEnvironment`
- `DeleteEnvironmentMembership`
- `DescribeEnvironmentMemberships`
- `DescribeEnvironmentStatus`
- `UpdateEnvironment`
- `UpdateEnvironmentMembership`

これらの各 API アクションの詳細については、AWS Cloud9 API リファレンスを参照してください。

リソースベースのポリシーを AWS Cloud9 リソースに直接アタッチすることはできません。代わりに、環境メンバーを追加、変更、更新、または削除するときに、適切なリソースベースのポリシーを AWS Cloud9 リソースに AWS Cloud9 アタッチします。

AWS Cloud9 リソースに対してアクションを実行するアクセス許可をユーザーに付与するには、ユーザーが属する IAM グループにアクセス許可ポリシーをアタッチします。AWS Cloud9 可能な限り、の AWS 管理 (事前定義) ポリシーをアタッチすることをお勧めします。AWS 管理ポリシーには、環境、環境ユーザー、および環境への読み取り専用アクセスのみを持つユーザーの完全な管理など、一般的な使用シナリオとユーザータイプに対する事前定義されたアクセス許可のセットが含まれています。の AWS 管理ポリシーのリストについては AWS Cloud9、「」を参照してください [AWS の マネージドポリシー AWS Cloud9](#)。

より詳細な使用シナリオと一意のユーザータイプについては、独自のカスターマネージドポリシーを作成してアタッチすることができます。「[の追加のセットアップオプション AWS Cloud9](#)」および「[のカスタマー管理ポリシーの作成 AWS Cloud9](#)」を参照してください。

IAM ポリシー (AWS マネージドまたはカスターマネージド) を IAM ID にアタッチするには、IAM ユーザーガイドの「[IAM ポリシーのアタッチ \(コンソール\)](#)」を参照してください。

API オペレーションのセッション許可

AWS CLI または AWS API を使用してロールまたはフェデレーティッドユーザーの一時セッションをプログラムで作成する場合、セッションポリシーをパラメータとして渡して、ロールセッションの

範囲を拡張できます。つまり、効果的なセッションの許可は、[ロールのアイデンティティベースのポリシーとセッションポリシーの共通部分](#)です。

セッション中にリソースへのアクセス要求が行われた場合、適用可能な Deny ステートメントがないものの、セッションポリシーに適用可能な Allow ステートメントもない場合、ポリシー評価の結果は[暗黙的な拒否](#)となります。(詳細については、IAM ユーザーガイドの「[アカウント内でリクエストが許可されるか拒否されるかの判別](#)」を参照してください)。

ただし、リソースベースのポリシーを必要とする AWS Cloud9 API オペレーション (上記を参照) では、リソースポリシーPrincipalでとして指定されている場合、を呼び出す IAM エンティティにアクセス許可が付与されます。この明示的なアクセス許可は、セッションポリシーの暗黙的な拒否よりも優先されるため、セッションは AWS Cloud9 API オペレーションを正常に呼び出すことができます。

AWS の マネージドポリシー AWS Cloud9

AWS 管理ポリシーは、によって作成および管理されるスタンドアロンポリシーです AWS。AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有の[カスタム管理ポリシー](#)を定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使えるようになったときに、AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: AWSCloud9Administrator

IAM アイデンティティに AWSCloud9Administrator ポリシーをアタッチできます。

このポリシーは、**###**アクセスを提供する管理者アクセス許可を付与します AWS Cloud9。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- AWS Cloud9 – その中のすべての AWS Cloud9 アクション AWS アカウント。
- Amazon EC2 – 複数の Amazon VPC とサブネットリソースに関する情報を取得します AWS アカウント。
- IAM – で IAM ユーザーに関する情報を取得し AWS アカウント、 AWS アカウント 必要に応じて AWS Cloud9 サービスにリンクされたロールを で作成します。
- Systems Manager — StartSession を呼び出して、セッションマネージャーセッションのインスタンスへの接続を開始する許可をユーザーに付与。この許可は、Systems Manager を介して EC2 インスタンスと通信する環境を開くユーザーにとって必要です。詳細については、[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)を参照してください。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSCloud9Administrator](#)」を参照してください。

AWS マネージドポリシー: AWSCloud9User

IAM アイデンティティに AWSCloud9User ポリシーをアタッチできます。

このポリシーは、#### に AWS Cloud9 開発環境の作成と、所有環境の管理許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- AWS Cloud9 – 環境に関する情報を作成および取得し、環境のユーザー設定を取得および変更します。
- Amazon EC2 – 複数の Amazon VPC とサブネットリソースに関する情報を取得します AWS アカウント。
- IAM – で IAM ユーザーに関する情報を取得し AWS アカウント、 AWS アカウント 必要に応じて AWS Cloud9 サービスにリンクされたロールを で作成します。
- Systems Manager — StartSession を呼び出して、セッションマネージャーセッションのインスタンスへの接続を開始する許可をユーザーに付与。この許可は、Systems Manager を介して EC2 インスタンスと通信する環境を開くユーザーにとって必要です。詳細については、[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)を参照してください。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSCloud9User](#)」を参照してください。

AWS マネージドポリシー: AWSCloud9EnvironmentMember

IAM アイデンティティに AWSCloud9EnvironmentMember ポリシーをアタッチできます。

このポリシーは、AWS Cloud9 共有環境に参加する機能を提供する#####アクセス許可を付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- AWS Cloud9 – 環境に関する情報を取得し、環境のユーザー設定を取得および変更します。
- IAM – で IAM ユーザーに関する情報を取得し AWS アカウント、AWS アカウント 必要に応じて AWS Cloud9 サービスにリンクされたロールを で作成します。
- Systems Manager — StartSession を呼び出して、セッションマネージャーセッションのインスタンスへの接続を開始する許可をユーザーに付与。この許可は、Systems Manager を介して EC2 インスタンスと通信する環境を開くユーザーにとって必要です。詳細については、[を使用した no-ingress EC2 インスタンスへのアクセス AWS Systems Manager](#)を参照してください。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSCloud9EnvironmentMember](#)」を参照してください。

AWS 管理ポリシー: AWSCloud9ServiceRolePolicy

サービスにリンクされたロール AWSServiceRoleForAWSCloud9 は、このポリシーを使用して、AWS Cloud9 環境が Amazon EC2 および CloudFormation リソースとやり取りできるようにします。

アクセス許可の詳細

AWSCloud9ServiceRolePolicy は、AWSServiceRoleForAWSCloud9 に、 が開発環境の作成と実行に必要な AWS のサービス (Amazon EC2 と CloudFormation) と AWS Cloud9 やり取りするために必要なアクセス許可を付与します。

AWS Cloud9 は、サービスにリンクされたロールのアクセス許可を定義し、のみがそのロールを引き受け AWS Cloud9 することができます。定義される許可は信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

がサービスにリンクされたロール AWS Cloud9 を使用する方法の詳細については、「」を参照してください[のサービスにリンクされたロールの使用 AWS Cloud9](#)。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の[AWSCloud9ServiceRolePolicy](#)」を参照してください。

AWS Cloud9 AWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始 AWS Cloud9 してからの AWS の管理ポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、AWS Cloud9 ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AWSCloud9Administrator ポリシーに新しいアクションが追加されました。	ssm:OpenDataChannel アクションが AWSCloud9 Administrator ポリシーに追加されました。このアクションは、SSM セッションのデータチャネルを開くアクセス許可をユーザーに付与します。	2025 年 9 月 25 日
AWSCloud9User ポリシーに新しいアクションが追加されました。	ssm:OpenDataChannel アクションが AWSCloud9 User ポリシーに追加されました。このアクションは、SSM セッションのデータチャネルを開くアクセス許可をユーザーに付与します。	2025 年 9 月 25 日
AWSCloud9EnvironmentMember ポリシーに新しいアクションが追加されました。	ssm:OpenDataChannel アクションが AWSCloud9 EnvironmentMember ポリシーに追加されました。このアクションは、SSM セッションのデータチャネルを開くアクセス許可をユーザーに付与します。	2025 年 9 月 25 日
AWSCloud9 User、AWSCloud9Administr	AWSCloud9 User、AWSCloud9Administr	2023 年 10 月 12 日

変更	説明	日付
ator、AWSCloud9EnvironmentMember の各ポリシーに新しいアクションを追加しました。	ator、AWSCloud9EnvironmentMember の各ポリシーに ssm:GetConnectionStatus アクションを追加しました。このアクションにより、SSM 接続ステータスを確認するアクセス許可がユーザーに付与されます。cloud9:ValidateEnvironmentName API は非推奨となり、AWSCloud9User ポリシーから削除されました。	
AWSCloud9User ポリシーと AWSCloud9Administrator ポリシーに API を追加しました。	AWSCloud9User ポリシーと AWSCloud9Administrator ポリシーに新しい 2 つの API として ec2:DescribeInstanceTypeOfferings と ec2:DescribeRouteTables を追加しました。これらの API の目的は、 が、お客様が AWS Cloud9 環境を作成するときに選択したインスタンスタイプをデフォルトのサブネットがサポートしていること AWS Cloud9 を検証することです。	2023 年 8 月 2 日

変更	説明	日付
AWSCloud9ServiceRolePolicy の更新	AWSCloud9ServiceRolePolicy が更新され、License Manager ライセンス設定によって管理される Amazon EC2 インスタンスを開始および停止 AWS Cloud9 できるようになりました。	2022 年 1 月 12 日
AWS Cloud9 が変更の追跡を開始しました	AWS Cloud9 は、AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 3 月 15 日

のカスタマー管理ポリシーの作成 AWS Cloud9

アクセスコントロール要件を満たしている AWS 管理ポリシーがない場合は、独自のカスタマー管理ポリシーを作成してアタッチできます。

カスタマーマネージドポリシーを作成するには、IAM ユーザーガイドの「[IAM ポリシーの作成 \(コンソール\)](#)」を参照。

トピック

- [ポリシー要素の指定: 効果、プリンシパル、アクション、リソース](#)
- [お客様のマネージドポリシーの例](#)

ポリシー要素の指定: 効果、プリンシパル、アクション、リソース

サービスは、AWS Cloud9 リソースごとに一連の API オペレーションを定義します。これらの API オペレーションのアクセス許可を付与するために、はポリシーで指定できる一連のアクション AWS Cloud9 を定義します。

以下は、基本的なポリシーの要素です。

- **Effect** – ユーザーがアクションをリクエストする際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に付与 (許可) していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。これは、別のポリ

シーがアクセスを許可している場合でも、ユーザーのリソースへのアクセスを禁止するために行うことができます。

- **Principal** – アイデンティティベースのポリシー (IAM ポリシー) で、ポリシーが添付されているユーザーが暗黙のプリンシパルとなります。リソースベースのポリシーでは、権限を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。
- **Resource** – Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。
- **Action** – アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。たとえば、cloud9:CreateEnvironmentEC2 許可は、CreateEnvironmentEC2 オペレーションを実行する許可をユーザーに与えます。

IAM ポリシーの構文と記述の詳細については、IAM ユーザーガイドの [IAM JSON ポリシーのリファレンス](#)を参照してください。

すべての AWS Cloud9 API アクションとそれらが適用されるリソースを示す表については、「」を参照してください[AWS Cloud9 アクセス許可リファレンス](#)。

お客様のマネージドポリシーの例

このセクションでは、AWS Cloud9 アクションの許可を付与するポリシー例を示しています。以下例にあげたIAM ポリシーを選択して、IAM アイデンティティへの AWS Cloud9 アクセスを許可するか、明示的に拒否できます。

IAM アイデンティティにカスタマー管理ポリシーを作成またはアタッチするには、「IAM ユーザーガイド」の「[IAM ポリシーの作成 \(コンソール\)](#)」および「[IAM ポリシーのアタッチ \(コンソール\)](#)」を参照してください。

Note

次の例では、米国東部 (オハイオ) リージョン (us-east-2)、架空の AWS アカウント ID ()、架空の AWS Cloud9 開発環境 ID (123456789012) を使用しています81e900317347585a0601e04c8d52eaEX。

トピック

- [環境に関する情報を取得](#)
- [EC2 環境を作成](#)

- [特定の Amazon EC2 インスタンスタイプを持つ EC2 環境を作成](#)
- [特定の Amazon VPC サブネットに EC2 環境を作成](#)
- [特定の環境名がついた EC2 環境を作成](#)
- [SSH 環境のみを作成](#)
- [環境を更新、または環境の更新を禁止](#)
- [環境メンバーのリストを取得](#)
- [特定のユーザーのみと環境を共有](#)
- [環境の共有を禁止](#)
- [環境メンバーの設定を変更、または変更を禁止](#)
- [環境メンバーを削除、または削除を禁止](#)
- [環境を削除、または削除を禁止](#)
- [SSM 環境作成用のカスタム IAM ポリシー](#)

環境に関する情報を取得

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントで環境に関する情報を取得するのを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:DescribeEnvironments",
      "Resource": "*"
    }
  ]
}
```

Note

前述のアクセス許可は、AWS 管理ポリシー `AWSCloud9Administrator` および `AWSCloud9User` にすでに含まれています。

EC2 環境を作成

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウントに AWS Cloud9 EC2 開発環境を作成することを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:CreateEnvironmentEC2",
      "Resource": "*"
    }
  ]
}
```

Note

前述のアクセス許可は、AWS 管理ポリシー `AWSCloud9Administrator` および `AWSCloud9User` にすでに含まれています。

特定の Amazon EC2 インスタンスタイプを持つ EC2 環境を作成

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウントに AWS Cloud9 EC2 開発環境を作成することを許可します。ただし、EC2 環境は指定されたクラスの Amazon EC2 インスタンスタイプのみを使用できます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:CreateEnvironmentEC2",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "cloud9:InstanceType": "t3.*"
        }
      }
    }
  ]
}
```

 Note

AWS 管理ポリシー `AWSCloud9Administrator` または `AWSCloud9User` が IAM エンティティに既にアタッチされている場合、その AWS 管理ポリシーは前述の IAM ポリシーステートメントの動作を上書きします。これは、これらの AWS 管理ポリシーがより寛容であるためです。

特定の Amazon VPC サブネットに EC2 環境を作成

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウントに AWS Cloud9 EC2 開発環境を作成することを許可します。ただし、EC2 環境は指定した Amazon VPC サブネットのみを使用できます。

JSON

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

```

    "Action": "cloud9:CreateEnvironmentEC2",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "cloud9:SubnetId": [
          "subnet-12345678",
          "subnet-23456789"
        ]
      }
    }
  }
]
}

```

Note

AWS 管理ポリシー `AWSCloud9Administrator` または `AWSCloud9User` が IAM エンティティに既にアタッチされている場合、その AWS 管理ポリシーは前述の IAM ポリシーステートメントの動作を上書きします。これは、これらの AWS 管理ポリシーがより寛容であるためです。

特定の環境名がついた EC2 環境を作成

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウントに AWS Cloud9 EC2 開発環境を作成することを許可します。ただし、EC2 環境は指定された名前のみを使用できます。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:CreateEnvironmentEC2",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloud9:EnvironmentName": "my-demo-environment"
        }
      }
    }
  ]
}

```

```

    }
  }
}
]
}

```

Note

AWS 管理ポリシー `AWSCloud9Administrator` または `AWSCloud9User` が IAM エンティティに既にアタッチされている場合、その AWS 管理ポリシーは前述の IAM ポリシーステートメントの動作を上書きします。これは、これらの AWS 管理ポリシーがより寛容であるためです。

SSH 環境のみを作成

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウントに AWS Cloud9 SSH 開発環境を作成することを許可します。ただし、エンティティは AWS Cloud9 EC2 開発環境を作成できません。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:CreateEnvironmentSSH",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "cloud9:CreateEnvironmentEC2",
      "Resource": "*"
    }
  ]
}

```


環境を更新、または環境の更新を禁止

次の IAM エンティティにアタッチされた IAM ポリシーステートメントの例では、そのエンティティがアカウント内の任意の AWS Cloud9 開発環境に関する情報を変更することを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:UpdateEnvironment",
      "Resource": "*"
    }
  ]
}
```

Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれています AWSCloud9Administrator。

次の IAM エンティティに添付された IAM ポリシーステートメント例では、そのエンティティが環境に関する情報を指定された Amazon リソースネーム (ARN) で変更することを明示的に禁止します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "cloud9:UpdateEnvironment",
      "Resource": "arn:aws:cloud9:us-east-2:123456789012:environment:81e900317347585a0601e04c8d52eaEX"
    }
  ]
}
```

```
}
```

環境メンバーのリストを取得

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントで環境のメンバーリストを取得するのを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:DescribeEnvironmentMemberships",
      "Resource": "*"
    }
  ]
}
```

Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれています。AWS Cloud9 Administrator。また、前述のアクセス許可は、AWS 管理ポリシー の同等のアクセス許可よりも許容されます。AWS Cloud9 User。

特定のユーザーのみと環境を共有

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、エンティティが指定したユーザーのみのアカウントにおける環境の共有を許可します。

JSON

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

```

```
"Effect": "Allow",
"Action": [
  "cloud9:CreateEnvironmentMembership"
],
"Resource": "*",
"Condition": {
  "StringEquals": {
    "cloud9:UserArn": "arn:aws:iam::123456789012:user/MyDemoUser"
  }
}
}
```

Note

AWS 管理ポリシー `AWSCloud9Administrator` または `AWSCloud9User` が既に IAM エンティティにアタッチされている場合、それらの AWS 管理ポリシーは前述の IAM ポリシーステートメントの動作を上書きします。これは、これらの AWS 管理ポリシーがより寛容であるためです。

環境の共有を禁止

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントの環境の共有を禁止します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cloud9:CreateEnvironmentMembership",
        "cloud9:UpdateEnvironmentMembership"
      ],
      "Resource": "*"
    }
  ]
}
```

```
]
}
```

環境メンバーの設定を変更、または変更を禁止

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントの環境においてメンバーの設定の変更を許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:UpdateEnvironmentMembership",
      "Resource": "*"
    }
  ]
}
```

Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれています AWSCloud9Administrator。

次の IAM エンティティに添付された IAM ポリシーステートメント例では、そのエンティティが環境におけるメンバーの設定を指定された Amazon リソースネーム (ARN) を使って変更することを明示的に禁止します。

JSON

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

```
"Action": "cloud9:UpdateEnvironmentMembership",
"Resource": "arn:aws:cloud9:us-
east-2:123456789012:environment:81e900317347585a0601e04c8d52eaEX"
}
]
}
```

環境メンバーを削除、または削除を禁止

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントで環境のメンバーの削除を許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloud9:DeleteEnvironmentMembership",
      "Resource": "*"
    }
  ]
}
```

 Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれていません `AWSCloud9Administrator`。

次の IAM エンティティに添付された IAM ポリシーステートメント例では、そのエンティティが指定された Amazon リソースネーム (ARN) で環境のメンバーの削除を明示的に禁止します。

JSON

```
{
  "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Effect": "Deny",  
    "Action": "cloud9:DeleteEnvironmentMembership",  
    "Resource": "arn:aws:cloud9:us-east-2:123456789012:environment:81e900317347585a0601e04c8d52eaEX"  
  }  
]
```

環境を削除、または削除を禁止

次の IAM エンティティに添付された IAM ポリシーステートメントの例では、そのエンティティが自分のアカウントで環境を削除するのを許可します。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "cloud9:DeleteEnvironment",  
      "Resource": "*"   
    }  
  ]  
}
```

Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれています `AWSCloud9Administrator`。

次の IAM エンティティに添付された IAM ポリシーステートメント例では、そのエンティティが指定された Amazon リソースネーム (ARN) で環境を削除するのを明示的に禁止します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "cloud9:DeleteEnvironment",
      "Resource": "arn:aws:cloud9:us-east-2:123456789012:environment:81e900317347585a0601e04c8d52eaEX"
    }
  ]
}
```

SSM 環境作成用のカスタム IAM ポリシー

現在、AWSCloud9Administrator ポリシーまたは AWSCloud9User ポリシーがアタッチされた SSM 環境を作成する際に、アクセス許可の問題が発生しています。次の IAM ポリシーステートメントの例では、IAM エンティティにアタッチすると、ユーザーは AWS マネージドポリシー AWSCloud9Administrator または をアタッチして使用できます AWSCloud9User。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloud9:UpdateUserSettings",
        "cloud9:GetUserSettings",
        "iam:GetUser",
        "iam:ListUsers",
        "iam:ListRoles",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeRouteTables"
      ],
      "Resource": "*"
    }
  ],
}
```

```
{
  "Effect": "Allow",
  "Action": [
    "cloud9:CreateEnvironmentEC2",
    "cloud9:CreateEnvironmentSSH"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "cloud9:OwnerArn": "true"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "cloud9:GetUserPublicKey"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "cloud9:UserArn": "true"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "cloud9:DescribeEnvironmentMemberships"
  ],
  "Resource": [
    "*"
  ],
  "Condition": {
    "Null": {
      "cloud9:UserArn": "true",
      "cloud9:EnvironmentId": "true"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ]
}
```



```

    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "cloud9.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": "ssm:StartSession",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
        "StringLike": {
            "ssm:resourceTag/aws:cloud9:environment": "*"
        },
        "StringEquals": {
            "aws:CalledViaFirst": "cloud9.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ssm:StartSession"
    ],
    "Resource": [
        "arn:aws:ssm:*:*:document/*"
    ]
},
{
    "Effect": "Allow",
    "Action": ["iam:ListInstanceProfilesForRole", "iam:CreateRole"],
    "Resource": ["arn:aws:iam:*:*:role/service-role/
AWSCloud9SSMAccessRole"]
},
{
    "Effect": "Allow",
    "Action": ["iam:AttachRolePolicy"],
    "Resource": ["arn:aws:iam:*:*:role/service-role/
AWSCloud9SSMAccessRole"],
    "Condition": {
        "StringEquals": {

```

```

        "iam:PolicyARN": "arn:aws:iam::aws:policy/
AWSCloud9SSMInstanceProfile"
    }
},
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::*:role/service-role/
AWSCloud9SSMAccessRole",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "ec2.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateInstanceProfile",
        "iam:AddRoleToInstanceProfile"
    ],
    "Resource": [
        "arn:aws:iam::*:instance-profile/cloud9/
AWSCloud9SSMInstanceProfile"
    ]
}
]
}

```

AWS Cloud9 アクセス許可リファレンス

AWS Cloud9 ポリシーで AWS ワイド条件キーを使用して条件を表現できます。リストについては、IAM ユーザーガイドの [IAM JSON ポリシーエレメント：条件](#) を参照してください。

アクションは、ポリシーの Action フィールドで指定します。アクションを指定するには、API オペレーション名 ("Action": "cloud9:DescribeEnvironments" など) の前に cloud9: プレフィックスを使用します。単一のステートメントに複数のアクションを指定するには、コンマで区切ります (例えば、"Action": ["cloud9:UpdateEnvironment", "cloud9:DeleteEnvironment"])。

ワイルドカード文字の使用

ポリシーの Resource フィールドでリソース値として、ワイルドカード文字 (*) を使用して、または使用せずに ARN を指定します。ワイルドカードを使用して複数のアクションまたはリソースを指定することができます。たとえば、cloud9:*はすべての AWS Cloud9 アクションを指定し、で始まるすべての AWS Cloud9 アクションcloud9:Describe*を指定しますDescribe。

次の例では、IAM エンティティが、アカウント内の環境の環境および環境メンバーシップに関する情報を取得するのを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloud9:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

上記のアクセス許可は、AWS 管理ポリシー にすでに含まれてい
ますAWSCloud9Administrator。また、前述のアクセス許可は、AWS 管理ポリシー の同等
のアクセス許可よりも許容されますAWSCloud9User。

AWS Cloud9 API オペレーションとアクションに必要なアクセス許可

Note

アクセスコントロールをセットアップし、IAM アイデンティティ (アイデンティティベースのポリシー) にアタッチできる許可ポリシーを作成する際に、以下の表をリファレンスとして使用します。

[Public API operations](#) 表は、SDK と AWS Command Line Interface を使用してお客様が呼び出せる API オペレーションをリストアップします。

[Permission-only API operations](#) は、顧客コードまたは AWS Command Line Interface によって直接呼び出せない API オペレーションをリストアップします。ただし、IAM ユーザーには、コンソールを使用して AWS Cloud9 アクションを実行する時に、こういったオペレーションに対して許可をとる必要があります。

公開 API オペレーション

AWS Cloud9 オペレーション	必要な許可 (API アクション)	[リソース]
CreateEnvironmentEC2	cloud9:CreateEnvironmentEC2 AWS Cloud9 EC2 開発環境を作成するために必要です。	*
CreateEnvironmentMembership	cloud9:CreateEnvironmentMembership 環境にメンバーを追加するために必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
DeleteEnvironment	cloud9:DeleteEnvironment 環境を削除するために必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
DeleteEnvironmentMembership	cloud9:DeleteEnvironmentMembership 環境からメンバーを削除するために必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
DescribeEnvironmentMemberships	cloud9:DescribeEnvironmentMemberships	*

AWS Cloud9 オペレーション	必要な許可 (API アクション)	[リソース]
	環境のメンバーのリストを取得するのに必要です。	
DescribeEnvironments	cloud9:DescribeEnvironments 環境に関する情報を取得するのに必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
DescribeEnvironmentStatus	cloud9:DescribeEnvironmentStatus 環境のステータスに関する情報を取得するのに必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
UpdateEnvironment	cloud9:UpdateEnvironment 環境の設定を更新するのに必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>
UpdateEnvironmentMembership	cloud9:UpdateEnvironmentMembership 環境のメンバーの設定を更新するのに必要です。	arn:aws:cloud9: <i>REGION_ID</i> : <i>ACCOUNT_ID</i> :environment: <i>ENVIRONMENT_ID</i>

許可のみの API オペレーション

AWS Cloud9 オペレーション	説明	コンソールドキュメンテーション
ActivateEC2Remote	cloud9:ActivateEC2Remote	で環境を開く AWS Cloud9

AWS Cloud9 オペレーション	説明	コンソールドキュメンテーション
	IDE が接続する Amazon EC2 AWS Cloud9 インスタンスを起動します。	
CreateEnvironmentSSH	cloud9:CreateEnvironmentSSH SSH AWS Cloud9 開発環境を作成します。	SSH 環境を作成する
CreateEnvironmentToken	cloud9:CreateEnvironmentToken AWS Cloud9 IDE とユーザーの環境間の接続を許可する認証トークンを作成します。	EC2 環境を作成する
DescribeEC2Remote	cloud9:DescribeEC2Remote ホスト、ユーザー、ポートなど、EC2 開発環境への接続に関する詳細を取得します。	EC2 環境を作成する
DescribeSSHRemote	cloud9:DescribeSSHRemote ホスト、ユーザー、ポートなど、SSH 開発環境への接続に関する詳細を取得します。	SSH 環境を作成する
GetEnvironmentConfig	cloud9:GetEnvironmentConfig AWS Cloud9 IDE を初期化するために使用される設定情報を取得します。	IDE AWS Cloud9 の使用

AWS Cloud9 オペレーション	説明	コンソールドキュメンテーション
GetEnvironmentSettings	cloud9:GetEnvironmentSettings 指定された開発環境の AWS Cloud9 IDE 設定を取得します。	IDE AWS Cloud9 の使用
GetMembershipSettings	cloud9:GetMembershipSettings 指定された環境メンバーの AWS Cloud9 IDE 設定を取得します。	での共有環境の使用 AWS Cloud9
GetUserPublicKey	cloud9:GetUserPublicKey ユーザーのパブリック SSH キーを取得します。これは、SSH 開発環境への接続 AWS Cloud9 に よって使用されます。	SSH 環境を作成する
GetUserSettings	cloud9:GetUserSettings 指定されたユーザーの AWS Cloud9 IDE 設定を取得します。	IDE AWS Cloud9 の使用

AWS Cloud9 オペレーション	説明	コンソールドキュメンテーション
ModifyTemporaryCredentialsOnEnvironmentEC2	cloud9:ModifyTemporaryCredentialsOnEnvironmentEC2 AWS Cloud9 統合開発環境 (IDE) で使用される Amazon EC2 インスタンスに AWS マネージド一時認証情報を設定します。	AWS マネージド一時認証情報
UpdateEnvironmentSettings	cloud9:UpdateEnvironmentSettings 指定された開発環境の AWS Cloud9 IDE 設定を更新します。	IDE AWS Cloud9 の使用
UpdateMembershipSettings	cloud9:UpdateMembershipSettings 指定された環境メンバーの AWS Cloud9 IDE 設定を更新します。	での共有環境の使用 AWS Cloud9
UpdateSSHRemote	cloud9:UpdateSSHRemote ホスト、ユーザー、ポートなど、SSH 開発環境への接続の詳細を更新します。	SSH 環境を作成する

AWS Cloud9 オペレーション	説明	コンソールドキュメンテーション
UpdateUserSettings	cloud9:UpdateUserSettings 指定されたユーザーの AWS Cloud9 IDE 設定を更新します。	IDE AWS Cloud9 の使用
GetMigrationExperiences	cloud9:GetMigrationExperiences から CodeCatalyst AWS Cloud9 への移行エクスペリエンスを取得するアクセス許可を AWS Cloud9 ユーザーに付与します。	

AWS マネージド一時認証情報

AWS マネージド一時認証情報がサポートするアクションのリストを検索する場合は、「」に進みます [AWS マネージド一時認証情報でサポートされるアクション](#)。

AWS Cloud9 EC2 開発環境の場合、AWS Cloud9 は環境内で一時的な AWS アクセス認証情報を利用できるようにします。これらは AWS マネージド一時認証情報と呼ばれます。これには次の利点があります。

- AWS 環境内の任意の場所にエンティティ (IAM ユーザーなど) の永続的な AWS アクセス認証情報を保存する必要はありません。これによって、ユーザーが認識して承認することなく、環境メンバがこれらの認証情報にアクセスできなくなります。
- 環境に接続する Amazon EC2 インスタンスにインスタンスプロファイルを手動で設定、管理、またはアタッチする必要はありません。インスタンスプロファイルは、一時的な AWS アクセス認証情報を管理するもう 1 つのアプローチです。

- AWS Cloud9 は一時的な認証情報を継続的に更新するため、1 つの認証情報セットを期間限定でのみ使用できます。これは AWS セキュリティのベストプラクティスです。詳細については、「[AWS マネージド一時認証情報の作成と更新](#)」を参照してください。
- AWS Cloud9 では、一時的な認証情報を使用して環境から AWS アクションとリソースにアクセスする方法に関する追加の制限を設けています。これは AWS セキュリティのベストプラクティスでもあります。

Important

現在、環境の EC2 インスタンスがプライベートサブネットで起動されている場合、AWS マネージド一時認証情報を使用して EC2 環境が AWS エンティティ (IAM ユーザーなど) に代わって AWS サービスにアクセスすることを許可することはできません。プライベートサブネットに EC2 インスタンスを起動できる時期に関する詳細については、「[のサブネットを作成する AWS Cloud9](#)」を参照してください。

Note

AWS マネージド一時認証情報を使用する場合は、インラインポリシーの代わりに AWS マネージドポリシーを使用することを検討してください。

EC2 環境が AWS エンティティ (IAM ユーザーなど) AWS のサービス に代わって にアクセスしようとするたびに、AWS マネージド一時認証情報がどのように機能するかを次に示します。

1. AWS Cloud9 は、呼び出し元の AWS エンティティ (IAM ユーザーなど) に、リクエストされたリソースに対してリクエストされたアクションを実行するアクセス許可があるかどうかをチェックします AWS。許可がない場合、または明示的に拒否されている場合、リクエストは失敗します。
 2. AWS Cloud9 は、AWS マネージド一時認証情報をチェックして、そのアクセス許可が、リクエストされたリソースに対してリクエストされたアクションを許可するかどうかを確認します AWS。許可がない場合、または明示的に拒否されている場合、リクエストは失敗します。AWS マネージド一時認証情報がサポートするアクセス許可のリストについては、「」を参照してください [AWS マネージド一時認証情報でサポートされるアクション](#)。
- AWS エンティティと AWS マネージド一時認証情報の両方が、リクエストされたリソースに対してリクエストされたアクションを許可している場合、リクエストは成功します。

- AWS エンティティまたは AWS マネージド一時認証情報のいずれかが、リクエストされたリソースに対してリクエストされたアクションを明示的に拒否または明示的に許可しなかった場合、リクエストは失敗します。つまり、呼び出し元の AWS エンティティに正しいアクセス許可がある場合でも、AWS Cloud9 が明示的に許可しない場合、リクエストは失敗します。同様に、AWS Cloud9 が特定のリソースに対して特定のアクションを実行することを許可する場合、AWS エンティティが明示的に許可していない場合、リクエストは失敗します。

EC2 環境の所有者は、次のように、その環境の AWS マネージド一時認証情報をいつでもオンまたはオフにできます。

1. 環境を開いた状態で、IDE AWS Cloud9 のメニューバーで AWS Cloud9、設定 を選択します。
2. [優先] タブのナビゲーションペインで、[AWS 設定]、[認証情報] を選択します。
3. AWS マネージド一時認証情報を使用して、AWS マネージド一時認証情報をオンまたはオフにします。

Note

AWS Cloud9 API オペレーションを呼び出し [UpdateEnvironment](#)、`managedCredentialsAction` パラメータに値を割り当てることで、AWS マネージド一時認証情報を有効または無効にすることもできます。この API オペレーションは、AWS SDKs や などの標準 AWS ツールを使用してリクエストできます AWS CLI。

AWS マネージド一時認証情報をオフにすると、リクエストを行った AWS エンティティに関係なく AWS のサービス、環境は にアクセスできません。ただし、環境の AWS マネージド一時認証情報を有効にできない、または有効にしたくない場合でも、環境にアクセスする必要があるとします AWS のサービス。その場合、次の代替方法を検討してください。

- Amazon EC2 インスタンス にインスタンスプロファイルを添付して、環境に接続します。手順については、「[インスタンスプロファイルを作成して使用し一時認証情報を管理する](#)」を参照してください。
- 特別な環境変数を設定したり、`aws configure` コマンドを実行したりするなどして、永続的な AWS アクセス認証情報を環境に保存します。手順については、「[環境に永続的アクセス認証情報を作成して保存する](#)」を参照してください。

上記の方法は、EC2 環境の AWS マネージド一時認証情報によって許可 (または拒否) されるすべての許可より優先されます。

AWS マネージド一時認証情報でサポートされるアクション

AWS Cloud9 EC2 開発環境の場合、AWS マネージド一時認証情報は、発信者のすべての AWS リソースに対するすべての AWS アクションを許可しますが AWS アカウント、以下の制限があります。

- では AWS Cloud9、次のアクションのみが許可されます。

- `cloud9:CreateEnvironmentEC2`
- `cloud9:CreateEnvironmentSSH`
- `cloud9:DescribeEnvironmentMemberships`
- `cloud9:DescribeEnvironments`
- `cloud9:DescribeEnvironmentStatus`
- `cloud9:UpdateEnvironment`

- IAM には、以下のアクションのみが許可されています。

- `iam:AttachRolePolicy`
- `iam:ChangePassword`
- `iam:CreatePolicy`
- `iam:CreatePolicyVersion`
- `iam:CreateRole`
- `iam:CreateServiceLinkedRole`
- `iam>DeletePolicy`
- `iam>DeletePolicyVersion`
- `iam>DeleteRole`
- `iam>DeleteRolePolicy`
- `iam>DeleteSSHPublicKey`
- `iam:DetachRolePolicy`
- `iam:GetInstanceProfile`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetRole`

- iam:GetRolePolicy
 - iam:GetSSHPublicKey
 - iam:GetUser
 - iam:List*
 - iam:PassRole
 - iam:PutRolePolicy
 - iam:SetDefaultPolicyVersion
 - iam:UpdateAssumeRolePolicy
 - iam:UpdateRoleDescription
 - iam:UpdateSSHPublicKey
 - iam:UploadSSHPublicKey
- ロールとやりとりするすべての IAM アクションは、Cloud9- で始まるロール名に対してのみ許可されます。ただし、iam:PassRole はすべてのロール名で使使します。
- AWS Security Token Service (AWS STS) では、次のアクションのみが許可されます。
- sts:GetCallerIdentity
 - sts:DecodeAuthorizationMessage
- サポートされているすべての AWS アクションは、環境の IP アドレスに制限されます。これは AWS セキュリティのベストプラクティスです。

AWS Cloud9 が EC2 環境にアクセスするために必要なアクションまたはリソースをサポートしていない場合、または EC2 環境で AWS マネージド一時認証情報がオフになっていて再びオンにできない場合は、次の代替方法を検討してください。

- EC2 環境に接続する Amazon EC2 インスタンスにインスタンスプロファイルを添付します。手順については、「[インスタンスプロファイルを作成・使用しマネージド一時認証情報を管理する](#)」を参照してください。
- 特別な環境変数を設定するか、aws configure コマンドを実行するなどして、永続的な AWS アクセス認証情報を EC2 環境に保存します。手順については、「[環境に永続的アクセス認証情報を作成して保存する](#)」を参照してください。

上記の方法は、EC2 環境の AWS マネージド一時認証情報によって許可 (または拒否) されるすべての許可より優先されます。

AWS マネージド一時認証情報の作成と更新

AWS Cloud9 EC2 開発環境の場合、AWS マネージド一時認証情報は環境を初めて開いたときに作成されます。

AWS マネージド一時認証情報は、次のいずれかの条件の下で更新されます。

- 一定の時間が経過するたび。現在、これは 5 分ごとです。
- 環境の IDE を表示するウェブブラウザタブを再ロードするたび。
- 環境の `~/.aws/credentials` ファイルにリストされているタイムスタンプに達したとき。
- AWS マネージド一時認証情報) 設定がオフに設定されている場合、それをオンに戻すたび。(この設定を表示または変更するには、IDE のメニューバーにおけるAWS Cloud9優先を選択します。[優先] タブでは、ナビゲーションペインで、AWS 設定、認証情報を選択します。)
- セキュリティのため、AWS マネージド一時認証情報は 15 分後に自動的に期限切れになります。認証情報を更新するには、環境所有者が IDE を通じて AWS Cloud9 環境を実行します。環境所有者のロールに関する詳細については、[AWS マネージド一時認証情報へのアクセスの制御](#) を参照してください。

AWS マネージド一時認証情報へのアクセスの制御

AWS マネージド一時認証情報を持つ共同作業者は、AWS Cloud9 を使用して他の とやり取りできます AWS のサービス。信頼できる共同作業者だけが AWS マネージド一時認証情報を与えるため、環境所有者以外の者を新しいメンバーを追加した場合、これらの認証情報は無効になります。認証情報は、`~/.aws/credentials` ファイルを削除すると無効になります。

Important

AWS マネージド一時認証情報も 15 分ごとに自動的に期限切れになります。共同作業者が引き続き使用できるように認証情報を更新するには、環境所有者が IDE を介して AWS Cloud9 環境に接続されている必要があります。

AWS マネージド一時認証情報を他のメンバーと共有できるように再有効化できるのは、環境所有者のみです。環境所有者が IDE を開くと、AWS マネージド一時認証情報が無効になっていることを確認するダイアログボックスが表示されます。環境所有者は、すべてのメンバーの資格情報を再度有効にするか、すべてのメンバーの認証情報を無効にできます。

⚠ Warning

ベストセキュリティプラクティスに従うため、環境に最後に追加されたユーザーのアイデンティティが確実にわからない場合は、マネージド一時認証情報が無効のままになります。[[Collaborate \(コラボレーション\)](#)] ウィンドウで読み取り/書き込み許可がおりたメンバーの一覧をチェックできます。

でのログ記録とモニタリング AWS Cloud9

CloudTrail によるアクティビティのモニタリング

AWS Cloud9 は、ユーザー AWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています AWS Cloud9。CloudTrail は、 のすべての API コールをイベント AWS Cloud9 としてキャプチャします。キャプチャされた呼び出しには、AWS Cloud9 コンソールからの呼び出しと API への AWS Cloud9 コード呼び出しが含まれます。APIs

証跡を作成する場合は、イベントを含む Amazon Simple Storage Service (Amazon S3) バケットへの CloudTrail イベントの継続的な配信を有効にすることができます AWS Cloud9。

追跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail で収集した情報を使用して、AWS Cloud9に対して行ったリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については、「[を使用した AWS Cloud9 API コールのログ記録 AWS CloudTrail](#)」を参照してください。

EC2 環境のパフォーマンスのモニタリング

AWS Cloud9 EC2 開発環境を使用している場合は、関連する Amazon EC2 インスタンスの信頼性、可用性、パフォーマンスをモニタリングできます。インスタンスのステータスのモニタリングを使って、インスタンスによるアプリケーションの実行が妨げられるような問題をAmazon EC2 が検出したかどうかをすばやく判断できます。

詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EC2 のモニタリング](#)」を参照してください。

のコンプライアンス検証 AWS Cloud9

サードパーティーの監査者は、複数の AWS コンプライアンスプログラムの一環として、AWS サービスのセキュリティとコンプライアンスを評価します。

AWS Cloud9 は、以下のコンプライアンスプログラムの対象です。

SOC

AWS システムおよび組織統制 (SOC) レポートは、が主要なコンプライアンス統制と目的 AWS を達成する方法を示す独立したサードパーティー審査レポートです。

サービス	SDK	SOC 1、2、3
AWS Cloud9	cloud9	✓

PCI

Payment Card Industry Data Security Standard (PCI DSS) は、PCI Security Standards Council によって管理される機密情報のセキュリティ標準であり、American Express、Discover Financial Services、JCB International、MasterCard Worldwide、Visa Inc により創設されました。

サービス	SDK	PCI
AWS Cloud9	cloud9	✓

FedRAMP

Federal Risk and Authorization Management Program (FedRAMP) は米国政府全体のプログラムであり、クラウドの製品やサービスに対するセキュリティ評価、認可、および継続的なモニタリングに関する標準アプローチを提供しています。

FedRAMP による評価および認可を受けているサービスのステータスは、次のとおりです。

- 第三者評価機関 (3PAO) の評価: このサービスは現在、第三者評価機関による評価を受けています。
- 共同承認委員会 (JAB) による審査: このサービスは、現在、JAB による審査を受けているところです。

サービス	SDK	FedRAMP Moderate (East/West)	FedRAMP High (GovCloud)
AWS Cloud9	cloud9	JAB による審査	該当なし

DoD CC SRG

米国防総省 (DoD) クラウドコンピューティングセキュリティ要求事項ガイド (SRG) には、クラウドサービスプロバイダー (CSP) が DoD の暫定認可を取得して DoD ユーザーへのサービス提供を可能にする、標準化された評価と認可プロセスが規定されています。

DoD CC SRG の評価および承認を受けているサービスのステータスは、次のとおりです。

- 第三者評価機関 (3PAO) の評価: このサービスは現在、第三者評価機関による評価を受けています。
- 共同承認委員会 (JAB) による審査: このサービスは、現在、JAB による審査を受けているところです。
- アメリカ国防情報システム局 (DISA) による審査: このサービスは、現在、DISA による審査を受けているところです。

サービス	SDK	DoD CC SRG IL2 (East/West)	DoD CC SRG IL2 (GovCloud)	DoD CC SRG IL4 (GovCloud)	DoD CC SRG IL5 (GovCloud)	DoD CC SRG IL6 (AWS シークレットリージョン)
AWS Cloud9	cloud9	JAB による審査	該当なし	該当なし	該当なし	該当なし

HIPAA BAA

1996 年の医療保険の相互運用性と説明責任に関する法令 (HIPAA) は、患者の同意や認識なく機密性の高い患者の健康情報が開示されないようにするための国家基準の作成を義務付けた連邦法です。

AWS は、HIPAA の対象となる対象エンティティとそのビジネスアソシエイトが、保護された医療情報 (PHI) を安全に処理、保存、送信できるようにします。さらに、2013 年 7 月現在、はそのようなお客様に標準化されたビジネスアソシエイト補遺 (BAA) AWS を提供しています。

サービス	SDK	HIPAA BAA
AWS Cloud9	cloud9	✓

IRAP

オーストラリア政府のお客様は、情報セキュリティ登録評価プログラム (IRAP) を使用して、適切な制御が行われていることを検証し、オーストラリアサイバーセキュリティセンター (ACSC) が作成したオーストラリア政府情報セキュリティマニユアル (ISM) の要件に対応する適切な責任モデルを決定することができます。

サービス	名前空間*	IRAP による保護
AWS Cloud9	cloud9	✓

*名前空間は、AWS 環境全体のサービスを識別するのに役立ちます。たとえば、IAM ポリシーを作成するときは、Amazon リソースネーム (ARNs) と read AWS CloudTrail ログを使用します。

C5

Cloud Computing Compliance Controls Catalog (C5) は、ドイツ連邦情報セキュリティ局 (BSI) がドイツで導入したドイツ政府支援の証明スキームで、ドイツ政府の「クラウドプロバイダーに対するセキュリティに関するレコメンデーション」内でクラウドサービスを使用するときに、組織が一般的なサイバー攻撃に対する運用上のセキュリティを実証できるようにします。

サービス	SDK	C5
AWS Cloud9	cloud9	✓

FINMA

FINMA はスイスの独立した金融市場の規制機関です。アマゾン ウェブ サービス (AWS) は、FINMA ISAE 3000 タイプ 2 レポートを完了しました。

サービス	SDK	FINMA
AWS Cloud9	cloud9	✓

GSMA

GSM 協会は、世界中のモバイルネットワーク事業者の利益を代表する業界団体です。アマゾン ウェブ サービス (AWS) の欧州 (パリ) および米国東部 (オハイオ) リージョンは、現在、GSM 協会 (GSMA) によってセキュリティ認定スキームサブスクリプション管理 (SAS-SM) のデータセンター運用管理 (DCOM) の範囲において認定を受けています。このように GSMA の要件に合致していることは、クラウドサービスプロバイダーに対する高い期待事項を満たすという AWS の継続的なコミットメントを示しています。

サービス	米国東部 (オハイオ)	欧州 (パリ)
AWS Cloud9	✓	✓

PiTuKri

AWS PiTuKri 要件との整合性は、フィンランド運輸通信局である Traficom によって設定されたクラウドサービスプロバイダーに対する高い期待を満たすという当社の継続的なコミットメントを示しています。

サービス	SDK	PiTuKri
AWS Cloud9	cloud9	✓

AWS のサービス が特定のコンプライアンスプログラムの対象であるかどうかを確認するには、[AWS のサービス「コンプライアンスプログラムによる対象範囲内」](#)の「コンプライアンス」を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS「コンプライアンスプログラム」](#)を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、[「Downloading Reports in AWS Artifact」](#)を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービスは、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、[AWS 「セキュリティドキュメント」](#)を参照してください。

の耐障害性 AWS Cloud9

AWS グローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心に構築されています。AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、[AWS 「グローバルインフラストラクチャ」](#)を参照してください。

グローバル AWS インフラストラクチャに加えて、はデータの耐障害性とバックアップのニーズをサポートする特定の機能 AWS Cloud9 をサポートしています。

- クラウド AWS Cloud9 内のアセット (ドキュメント AWS CodeCommit、ソースコード、バイナリファイルなど) をプライベートに保存および管理するために使用できる Amazon Web Services がホストするバージョン管理サービスと統合します。詳細については、「AWS CodeCommit ユーザーガイド」の「[AWS Cloud9 との統合 AWS CodeCommit](#)」を参照してください。
- AWS Cloud9 開発環境で Git バージョン管理システムを使用して、リモート GitHub リポジトリにファイルとデータをバックアップします。詳細については、「[Git パネルを使用したビジュアルソース管理](#)」を参照してください。

のインフラストラクチャセキュリティ AWS Cloud9

マネージドサービスである AWS Cloud9 は、AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスとがインフラストラクチャ AWS を保護する方法については、[AWS 「クラウドセキュリティ」](#)を参照してください。インフラストラクチャセキュリティの

ベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「[Infrastructure Protection](#)」を参照してください。

AWS が公開した API コールを使用して、ネットワーク AWS Cloud9 経由で にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

Note

デフォルトでは、AWS Cloud9 EC2 開発環境はインスタンスのシステムパッケージのセキュリティパッチを自動的にインストールします。

ソフトウェアの更新プログラムと修正プログラム

AWS Cloud9 開発環境は、クラウドコンピューティングリソース上で実行されます。クラウドコンピューティングリソースは、EC2 環境用 Amazon EC2 インスタンスまたは SSH 環境用の独自のクラウドコンピューティングリソースとすることができます。これらの両方のオプションの詳細については、「[環境とコンピューティングリソース](#)」セクションを参照してください。

AWS Cloud9 EC2 環境は、環境の起動後にオペレーティングシステムのセキュリティパッチと更新を自動的にインストールします。AWS Cloud9 環境には、`g` IDE 機能を機能およびサポート AWS Cloud9 するために必要なソフトウェアパッケージも含まれています。これらのパッケージには、環境がロードされると自動的にパッチが適用されます。

AWS Cloud9 EC2 環境には、いくつかのプリインストールされた AWS 開発ツールも含まれています。これらのツールの更新は、AWS Cloud9 ユーザーまたはその管理者が手動でインストールする必要があります。これらのツールの更新の詳細については、以下のセクションを参照してください。

- 「AWS Command Line Interface ユーザーガイド」の「[AWS CLIの最新バージョンをインストールまたは更新する](#)」。
- 「AWS Serverless Application Model デベロッパーガイド」の「[CLI AWS SAM バージョンの管理](#)」。

- 「AWS Cloud Development Kit (AWS CDK) デベロッパーガイド」の「[AWS CDKをインストールする](#)」。

のセキュリティのベストプラクティス AWS Cloud9

以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションに相当するものではありません。これらのベストプラクティスはユーザーの環境に適切ではないか、十分ではない場合があるため、絶対的な解決策ではなく、役に立つ情報としてお考えください。

のセキュリティのベストプラクティス AWS Cloud9

- コードをバージョン管理システムに安全に保存します。例: [AWS CodeCommit](#)。
- AWS Cloud9 EC2 開発環境では、[Amazon Elastic Block Store](#) 暗号化ボリュームを設定して使用します。
- EC2 環境の場合は、[タグ](#)を使用して、AWS Cloud9 リソースへのアクセスをコントロールします。
- 共有 AWS Cloud9 開発環境については、[???](#)ベストプラクティスに従ってください。

トラブルシューティング AWS Cloud9

以下の情報を使用して、の問題を特定して対処します AWS Cloud9。

該当する問題が以下に示されていない場合や、追加のヘルプが必要な場合は、[AWS Cloud9 デイスクッションフォーラム](#)を参照してください。このフォーラムにアクセスするには、サインインが必要になることがあります。当社に直接[お問い合わせ](#)いただくこともできます。

トピック

- [Installer \(インストーラ\)](#)
- [AWS Cloud9 環境](#)
- [Amazon EC2](#)
- [その他の AWS サービス](#)
- [アプリケーションプレビュー](#)
- [パフォーマンス](#)
- [サードパーティのアプリケーションとサービス](#)

Installer (インストーラ)

次のセクションでは、AWS Cloud9 インストーラーに関連する問題のトラブルシューティングについて概説します。

AWS Cloud9 インストーラがハングまたは失敗する

問題: [AWS Cloud9 インストーラをダウンロードして実行する](#)と、1 つ以上のエラーが発生し、インストーラスクリプトに が表示されませんDone。

原因: AWS Cloud9 インストーラで復旧できないエラーが 1 つ以上発生し、その結果失敗します。

解決策: 詳細については、「[AWS Cloud9 インストーラのトラブルシューティング](#)」を参照してください。一般的な問題、考えられる原因、および推奨される解決策を参照してください。

AWS Cloud9 「Package Cloud9 IDE 1」と表示してもインストーラが終了しない

Issue: AWS Cloud9 SSH 開発環境を作成するプロセスの一環として、既存の Amazon EC2 インスタンスまたは独自のサーバーにインストールされます。[AWS Cloud9 インストーラ] ダイアログボッ

クスに「Package Cloud9 IDE 1」というメッセージが表示されて、インストールが停止します。
[キャンセル] を選択すると、「インストールに失敗しました」というメッセージが表示されます。
このエラーは、お客様の SSH ホストに AWS Cloud9 パッケージをインストールできない場合に発生します。

原因: SSH ホストでは、Node.js がインストールされている必要があります。ホストのオペレーティングシステムでサポートされている最新の Node.js バージョンをインストールすることをお勧めします。がサポート AWS Cloud9 していないバージョンの がホストNode.jsにある場合、インストールエラーが発生する可能性があります。

推奨される解決策: AWS Cloud9 がサポートする Node.js のバージョンを SSH ホストにインストールします。

依存関係をインストールできませんでした

問題: AWS Cloud9 依存関係をダウンロードするにはインターネットアクセスが必要です。

考えられる原因:

- AWS Cloud9 環境がプロキシを使用してインターネットにアクセスしている場合、依存関係をインストールするにはプロキシの詳細 AWS Cloud9 が必要です。プロキシの詳細を に提供しなかった場合 AWS Cloud9は、このエラーが表示されます。
- 別の原因としては、環境がアウトバウンドトラフィックを許可していないことが考えられます。

推奨される解決策:

- プロキシの詳細を に提供するには AWS Cloud9、環境 ~/.bashrc ファイルに次のコードを追加します。

```
export http_proxy=[proxy url for http]
export https_proxy=[proxy url for https]
#Certificate Authority used by your proxy
export NODE_EXTRA_CA_CERTS=[path_to_pem_certificate]
```

例えば、HTTP プロキシ URL が https://172.31.26.80:3128 で、HTTP プロキシ URL が https://172.31.26.80:3129 である場合、次の行を ~/.bashrc ファイルに追加し、NODE_EXTRA_CA_CERTS を PEM 形式で認証機関のパスに設定します。この変数の詳細については、「https://nodejs.org/api/cli.html#node_extra_ca_certsfile」を参照してください。


```
export http_proxy=http://172.31.26.80:3128
export https_proxy=https://172.31.26.80:3129
export NODE_EXTRA_CA_CERTS=[path_to_pem_certificate]
```

- no-ingress Amazon EC2 インスタンスを使用している場合は、Amazon S3 の Amazon VPC エンドポイントが設定されていることを確認する必要があります。詳細については、「[Amazon S3 のダウンロード依存関係に対応する Amazon VPC エンドポイントの設定](#)」を参照してください。

SSH 環境エラー: 「pty.js をインストールするには Python バージョン 3 が必要です」

問題: SSH AWS Cloud9 開発環境を開くと、IDE AWS Cloud9 のターミナルに「pty.js のインストールには Python バージョン 3 が必要です」で始まるメッセージが表示されます。

原因: SSH 環境が意図したとおりに動作するには、Python バージョン 3 がインストールされている必要があります。

解決策: 環境に Python バージョン 3 をインストールします。バージョンを確認するには、サーバーのターミナルからコマンド **python --version** を実行します。サーバーに Python 3 をインストールする方法については、次のいずれかを参照してください。

- Python サンプルの [ステップ 1: Python をインストールする](#)。
- Python のウェブサイトから [Python をダウンロード](#)します。

AWS Cloud9 環境

次のセクションでは、AWS Cloud9 環境に関連する問題のトラブルシューティングについて概説します。

環境の作成エラー: 「EC2 インスタンスを作成することができません ...」

問題: AWS Cloud9 開発環境を作成しようとする、「アカウントの検証とアクティベーション中にアカウントに EC2 インスタンスを作成できません」というメッセージが表示されます。

原因: AWS は現在、 を検証してアクティブ化しています AWS アカウント。アクティベーションが完了するまで (最大 24 時間かかる場合があります)、この環境やその他の環境を作成することはできません。

解決策: 後で環境をもう一度作成してみてください。24 時間経ってもこのメッセージが届く場合は、[サポート](#)にお問い合わせください。さらに、重要な留意事項として、環境を作成する試行が失敗しても、AWS CloudFormation はアカウントに関連スタックを作成します。これらのスタックは、アカウントのスタック作成クォータにカウントされます。スタック作成クォータの消費を避けるために、これらの失敗したスタックを削除できます。詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS CloudFormation コンソールのスタックを削除](#)」を参照してください。

環境作成エラー: 「sts:AssumeRole を実行する権限がありません」

問題: 新しい環境を作成しようとすると、エラー「sts:AssumeRole を実行する権限がありません」が表示され、環境が作成されません。

考えられる原因: AWS Cloud9 サービスにリンクされたロールが に存在しません AWS アカウント。

推奨される解決策: で AWS Cloud9 サービスにリンクされたロールを作成します AWS アカウント。これを行うには、次のコマンドを AWS Command Line Interface (AWS CLI) または AWS CloudShell で実行できます。

```
aws iam create-service-linked-role --aws-service-name cloud9.amazonaws.com # For the
AWS CLI.
iam create-service-linked-role --aws-service-name cloud9.amazonaws.com      # For the
aws-shell.
```

これができない場合は、AWS アカウント 管理者に確認してください。

このコマンドを実行した後、もう一度環境の作成を試します。

フェデレーティッドアイデンティティで環境を作成できない

問題: AWS フェデレーティッド ID を使用して AWS Cloud9 開発環境を作成しようとすると、アクセスエラーメッセージが表示され、環境は作成されません。

原因: サービスにリンクされたロール AWS Cloud9 を使用します。サービスにリンクされたロールは、iam:CreateServiceLinkedRole コールを使用してアカウントで初めて環境を作成したときに作成されます。ただし、フェデレーティッドユーザーは IAM API を呼び出すことができません。詳細については、AWS Security Token Service API リファレンスの「[GetBucketEncryption](#)」を参照してください。

解決策: IAM コンソールで AWS Cloud9 、または AWS Command Line Interface (AWS CLI) でこのコマンドを実行して、 のサービスにリンクされたロールを作成するように AWS アカウント 管理者に依頼します。

```
aws iam create-service-linked-role --aws-service-name cloud9.amazonaws.com
```

または、AWSシェルの使用して次のコマンドを実行します。

```
iam create-service-linked-role --aws-service-name cloud9.amazonaws.com
```

詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの使用」を参照してください。 <https://docs.aws.amazon.com/IAM/latest/UserGuide/using-service-linked-roles.html>

コンソールエラー: 「ユーザーにはリソースでアクションを実行する権限がありません」

問題: AWS Cloud9 コンソールを使用して AWS Cloud9 開発環境を作成または管理しようとする、 「User arn:aws:iam::123456789012:user/MyUser is not authorized to perform cloud9:action on resource arn:aws:cloud9:us-east-2:123456789012:environment:12a34567b8cd9012345ef67abcd890e1」 のようなフレーズを含むエラーが表示されます。

- arn:aws:iam::123456789012:user/MyUser は、リクエストするユーザーの Amazon リソースネーム (ARN) です。
- action は、ユーザーがリクエストしたオペレーションの名前です。
- arn:aws:cloud9:us-east-2:123456789012:environment:12a34567b8cd9012345ef67abcd890e1 は、ユーザーがオペレーションを実行するためにリクエストした環境の ARN です。

原因: AWS Cloud9 コンソールにサインインしたユーザーに、アクションを実行するための正しい AWS アクセス許可がありません。

解決策: ユーザーが正しい AWS アクセス許可を持っていることを確認して、もう一度アクションを実行してみます。詳細については次を参照してください:

- チーム設定の [ステップ 2: AWS Cloud9 グループにアクセス許可を追加する](#)
- エンタープライズセットアップの [ステップ 6. 組織内のグループとユーザーが 使用できるようにする AWS Cloud9](#)
- 共有環境を使用するの [環境メンバーのアクセスロールについて](#)

環境に接続できない

問題: ユーザーは環境に接続できず、接続段階で行き詰まっています。

原因: ~/ .ssh/authorized_keys ファイルのアクセス許可を変更したり、そのファイルから AWS Cloud9 キーを削除したり、ファイルを完全に削除したりすると、この問題が発生する可能性があります。

解決策: このファイルを削除しないでください。削除した場合は、環境を再作成して、既存の環境の [EBS ボリューム](#) を新しい EC2 環境にアタッチする必要があります。これにより、失われたデータが取得されます。アクセス許可が足りない場合は、ファイルに Read-Write アクセス許可があることを確認してください。これにより、SSH デーモンがファイルを読み取れるようにします。

環境を開くことができない

問題: 環境を開こうとすると、IDE が 5 分以上表示されません。

考えられる原因:

- AWS Cloud9 コンソールにサインインしている IAM ユーザーには、環境を開くために必要な AWS アクセス許可がありません。
- 環境が AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) に関連付けられている場合、次の可能性があります。
 - インスタンスに関連付けられている VPC が正しく設定されていません AWS Cloud9。
 - がインスタンスに接続しようとしているときに、インスタンスが状態間で移行しているか AWS Cloud9、自動ステータスチェックに失敗しています。
- 環境が SSH 環境の場合、関連付けられたクラウドコンピューティングインスタンスまたは独自のサーバーが、AWS Cloud9 がアクセスできるように正しく設定されていません。

推奨される解決策:

- AWS Cloud9 コンソールにサインインしている IAM ユーザーに、環境を開くために必要な AWS アクセス許可があることを確認します。その後、もう一度環境を開いてみてください。詳細については、以下を参照するか、AWS アカウント 管理者にお問い合わせください。
 - チームセットアップの [ステップ 2: AWS Cloud9 グループにアクセス許可を追加する](#)
 - 認証とアクセスコントロールの [AWS の マネージドポリシー AWS Cloud9](#)
 - アドバンスドチーム設定の [を使用するチームのカスタマー管理ポリシーの例 AWS Cloud9](#)

- 認証とアクセスコントロールの [お客様のマネージドポリシーの例](#)
- 「IAM ユーザーガイド」の「[IAM ユーザーのアクセス許可の変更](#)」を参照してください。
- 「IAM ユーザーガイド」の「[IAM ポリシーのトラブルシューティング](#)」

サインインした IAM ユーザーがまだ環境を開くことができない場合は、サインアウトしてから、アカウントの AWS アカウント ルートユーザーまたは管理者ユーザーとしてサインインし直してみてください。その後、もう一度環境を開いてみてください。この方法で環境を開くことができない場合は、IAM ユーザーのアクセス許可に問題がある可能性が最も大です。

- 環境が AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) に関連付けられている場合は、次の操作を行います。
- インスタンスに関連付けられている VPC が正しい設定になっていることを確認し AWS Cloud9、もう一度環境を開いてください。詳細については、「[の Amazon VPC 要件 AWS Cloud9](#)」を参照してください。

AWS クラウドコンピューティングインスタンスに関連付けられている VPC が の正しい設定に設定 AWS Cloud9 されていても環境を開くことができない場合、インスタンスのセキュリティグループが へのアクセスを妨げている可能性があります AWS Cloud9。トラブルシューティングの手法としてのみ、セキュリティグループをチェックして、少なくともポート 22 経由のインバウンド SSH トラフィックがすべての IP アドレス (Anywhere または 0.0.0.0/0) について許可されていることを確認してください。指示については、「Amazon EC2 ユーザーガイド」<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html#describing-security-group>の「[セキュリティグループについて説明する](#)」および「[セキュリティグループルールを更新する](#)」を参照してください。

追加の VPC トラブルシューティングの手順については、関連する 5 分間の動画「[AWS ナレッジセンターの動画: VPC 内のインスタンスに接続できない場合の確認点とは?](#)」を YouTube でご覧ください。

 Warning

トラブルシューティングが完了したら、インバウンドルールを適切なアドレス範囲に設定してください。詳細については、「[the section called “インバウンド SSH IP アドレスの範囲”](#)」を参照してください。

- インスタンスを再起動して、インスタンスが動作していること、およびすべてのシステムチェックをパスしていることを確認してから、もう一度環境を開いてみます。詳細について

は、Amazon EC2 ユーザーガイドで「[インスタンスの再起動](#)」と「[ステータスチェックの表示](#)」を参照してください。

- 環境が SSH 環境の場合は、環境に関連付けられているクラウドコンピューティングインスタンスまたは独自のサーバーが正しく設定され、AWS Cloud9 が環境にアクセスできることを確認してください。その後、もう一度環境を開いてみてください。詳細については、「[SSH 環境ホスト要件](#)」を参照してください。

AWS Cloud9 環境を開くことができません：「この環境には現在共同作業者がアクセスできません。マネージド一時認証情報の削除が完了するまでお待ちください。または、この環境の所有者にお問い合わせしてください。」

問題: 環境所有者ではないユーザーが新しい共同作業者を環境に追加すると、AWS マネージド一時認証情報は無効になります。~/aws/credentials ファイルを削除すると、認証情報は無効になります。~/aws/credentials ファイルの削除中、新しい共同作業者は AWS Cloud9 環境にアクセスできません。

原因: AWS マネージド一時認証情報の削除中に環境へのアクセスができないのは、セキュリティ対策です。これにより、環境所有者は、信頼できる共同作業者がマネージド認証情報にアクセスできることを確認できます。共同作業者のリストが有効であることが確認できたら、環境所有者はマネージド認証情報を再度有効にして共有することができます。詳細については、「[AWS マネージド一時認証情報へのアクセスの制御](#)」を参照してください。

推奨される解決策: ~/aws/credentials ファイルが完全に削除されるまで待つから、AWS Cloud9 環境をもう一度開いてください。認証情報待機時間の有効期限は最大 15 分です。または、マネージド一時認証情報を再度有効または無効にするよう、環境所有者に依頼します。認証情報が再び有効または無効になると、共同作業者はすぐに環境にアクセスできます。マネージド認証情報の状態を ENABLED または DISABLED に切り替えることで、環境所有者は、認証情報が中間状態に残らないようにします。中間状態では、共同作業者が環境にアクセスできない可能性があります。

 Note

環境所有者と共同作業者が同じ AWS アカウントに属しているとします。この場合、共同作業者は、コンソールの [Your environments] (自分の環境) ページで環境カードを確認し、連絡先の環境所有者を特定できます。環境所有者も、環境の詳細ページにアがっています。

環境の削除エラー: 「1 つ以上の環境を削除できませんでした」

問題: AWS Cloud9 コンソールで 1 つ以上の環境を削除しようとする、と、「1 つ以上の環境が削除に失敗した」というメッセージが表示され、少なくとも 1 つの環境は削除されません。

考えられる原因: AWS CloudFormation 1 つ以上の環境の削除に問題がある可能性があります。環境の作成と削除 CloudFormation は AWS Cloud9 に依存します。

推奨される解決策: CloudFormation を使用して、削除されていない各環境を削除してみてください。

1. <https://console.aws.amazon.com/cloudformation> で CloudFormation コンソールを開きます。
2. AWS ナビゲーションバーで、環境 AWS リージョン の を選択します。
3. CloudFormation スタックのリストで、スタック名に削除されていない環境名が含まれ、ステータスが DELETE_FAILED であるエントリを選択します。たとえば、環境名が **my-demo-environment** の場合は、aws-cloud9-my-demo-environment を含む名前が始まるスタックを選択します。(環境名の横にあるボックスまたはオプションを選択してください。環境名自体は選択しないでください)。
4. [アクション]、[スタックの削除] の順に選択します。
5. プロンプトが表示されたら、[はい、削除します] を選択します。

スタックの削除処理には数分かかる場合があります。

スタックがリストから消えた場合、環境は削除されています。

数分たってもスタックに [DELETE_FAILED] と表示される場合、環境はまだ削除されていません。この場合、障害が発生した各スタックのリソースを手動で削除できます。

Note

失敗したスタックのリソースを手動で削除しても、スタック自体は から削除されません AWS アカウント。

これらのリソースを手動で削除するには、次の手順に従います。 CloudFormation コンソールで、失敗したスタックを選択し、リソースセクションを選択します。このリストの各リソース AWS の のコンソールに移動し、そのコンソールを使用してリソースを削除します。

IDE AWS Cloud9 での環境のタイムアウト時間の変更

問題: ユーザーは Amazon EC2 環境のタイムアウト時間を更新したいと考えています。

原因: デフォルトのタイムアウト時間は 30 分です。これは一部のユーザーには短すぎるかもしれません。

推奨される解決策:

1. 設定する環境を開きます。
2. [AWS Cloud9 IDE] のメニューバーで、[AWS Cloud9]、[設定] の順に選択します。
3. [設定] ウィンドウで、[Amazon EC2 インスタンス] セクションまでスクロールします。
4. 使用可能なリストからタイムアウト値を選択し、更新します。

AWS Cloud9 環境に十分なディスク容量がないため、AWS Toolkit で SAM アプリケーションをローカルで実行中にエラーが発生しました

問題: Toolkit を使用して AWS SAM テンプレートで定義されたアプリケーションの AWS SAM CLI コマンドを実行すると、エラーが発生します。

考えられる原因: AWS Toolkit を使用してサーバーレスアプリケーションをローカルで実行およびデバッグすると、は Docker イメージ AWS SAM を使用します。これらのイメージは、デプロイを計画している Lambda 環境をエミュレートするランタイム環境とビルドツールを構築します。

ただし、環境に十分なディスク容量がない場合、これらの機能を提供する Docker イメージは構築できず、ローカル SAM アプリケーションを実行できません。この問題が発生した場合は、[Output] (出力) タブに次のようなエラーが表示されることがあります、

```
Error: Could not find amazon/aws-sam-cli-emulation-image-python3.7:rapid-1.18.1 image locally and failed to pull it from docker.
```

このエラーは、Python ランタイムを使用して構築された SAM アプリケーションに関連します。アプリケーション用に選択したランタイムに応じて、若干異なるメッセージが表示される場合があります。

推奨される解決策: Docker イメージを構築できるように、環境内のディスク領域を解放します。IDE の端末で次のコマンドを実行して、未使用の Docker イメージを削除します。


```
docker image prune -a
```

ディスク容量の制限により SAM CLI コマンドで問題が繰り返して発生する場合は、開発環境に切り替えて別の[インスタンスタイプ](#)を使用します。

[\(先頭に戻ります\)](#)

古いバージョンの Microsoft Edge ブラウザを使用して IDE をロードすることはできません

問題: Microsoft Edge ウェブブラウザを使用して IDE AWS Cloud9 をロードしようとすると、HTTP403: FORBIDDEN エラーが返されます。

考えられる原因: IDE AWS Cloud9 は、特定の古いバージョンの [Microsoft Edge](#) をサポートしていません。

推奨される解決策: ブラウザを更新するには、Microsoft Edge ツールバーの省略記号 (...) ボタンをクリックします。メニューで、[Settings] (設定) を選択して [About Microsoft Edge] (について) を選択します。アップデートが必要な場合は、自動的にダウンロードされ、インストールされます。

[\(先頭に戻ります\)](#)

IDE File Explorer でサブフォルダ構造 /home/ec2-user/environment/home/ec2-user/environment AWS Cloud9 を作成することはできません。

問題: IDE File Explorer でサブフォルダ構造 /home/ec2-user/environment/home/ec2-user/environment AWS Cloud9 を作成すると、このディレクトリを開くことができないことを示すエラーメッセージが表示されます。

考えられる原因: 現在、IDE のファイルシステムを使用して、同じ名前のフォルダ内にサブフォルダ構造 /home/ec2-user/environment AWS Cloud9 を作成することはできません。AWS Cloud9 IDE File Explorer からこのディレクトリ内のファイルにはアクセスできませんが、コマンドラインを使用してアクセスできます。この問題の影響を受けるのは /home/ec2-user/environment/home/ec2-user/environment のファイルパスだけです。/test/home/ec2-user/environment や /home/ec2-user/environment/test などのファイルパスは機能します。これは既知の問題であり、IDE File Explorer AWS Cloud9 にのみ影響します。

推奨される解決策: 別のファイル名と構造を使用してください。

[\(先頭に戻ります\)](#)

CodeCatalyst の IDE の File Explorer 内にサブフォルダ構造 /projects/projects AWS Cloud9 を作成することはできません。

問題: CodeCatalyst の AWS Cloud9 IDE File Explorer でサブフォルダ構造 /projects/projects を作成すると、このディレクトリを開くことができないことを示すエラーメッセージが表示されます。

考えられる原因: 現在、CodeCatalyst の AWS Cloud9 IDE の File Explorer を使用して、同じ名前のフォルダ内にサブフォルダ構造/プロジェクトを作成することはできません。AWS Cloud9 IDE File Explorer からこのディレクトリ内のファイルにはアクセスできませんが、コマンドラインを使用してアクセスできます。この問題は /projects/projects のファイルパスにのみ影響します。/test/projects や /projects/test などのファイルパスは正常に機能します。これは既知の問題で、AWS Cloud9 IDE ファイルエクスプローラーにのみ影響します。

推奨される解決策: 別のファイル名と構造を使用してください。

[\(先頭に戻ります\)](#)

tmux セッションエラー AWS Cloud9 のため、でターミナルウィンドウとやり取りできない

問題: で新しいターミナルウィンドウを起動しようとする AWS Cloud9、予想されるコマンドラインインターフェイスは使用できません。コマンドプロンプトがないため、テキストを入力できません。tmux: need UTF-8 locale (LC_CTYPE) や invalid LC_ALL, LC_CTYPE or LANG などのエラーメッセージが返されます。

考えられる原因: tmux エラーが原因でターミナルが応答しない可能性があります。は [tmux](#) ユーティリティ AWS Cloud9 を使用します。これにより、ページがリロードされたり、開発環境に再接続しても、ターミナルに表示される情報は保持されます。

tmux セッションでは、ターミナルウィンドウの表示内容はクライアントによって処理されます。クライアントは、複数のセッションを管理できるサーバーと通信します。サーバーとクライアントは、tmp フォルダにあるソケットを介して通信します。開発環境に tmp フォルダがないか、過度に制限的なアクセス許可が適用されている場合、tmux セッションは実行できません。この場合、IDE のターミナルウィンドウは応答しなくなります。

推奨される解決策: tmux エラーによってターミナルウィンドウと対話できない場合は、別の方法を使用して適切な許可を持つ tmp フォルダを作成する必要があります。これにより、tmux セッションを実行できます。1 つの解決策は、.bash_profile または .bashrc ファイルの LC_CTYPE をエクスポートすることです。もう 1 つの推奨される解決策は、AWS Systems Manager を使用して

ホスト管理設定をセットアップすることです。これにより、Amazon EC2 コンソールから該当するインスタンスにアクセスできるようになります。

ホスト管理の設定

1. まず、AWS Cloud9 コンソールで、環境のインスタンスの名前を見つけます。これを行うには、[Your environments] (自分の環境) ページで該当するパネルを選択し、[View details] (詳細を表示) を選択します。[環境の詳細] ページで [インスタンスに移動] を選択します。Amazon EC2 コンソールで、アクセスする必要があるインスタンスの名前を確認します。
2. AWS Systems Manager コンソールに移動し、ナビゲーションペインでクイックセットアップを選択します。
3. [クイックセットアップ] ページで [作成] を選択します。
4. [設定タイプ] では、[ホスト管理] に移動し、[登録] を選択します。
5. [ホスト管理構成オプションのカスタマイズ] の [ターゲット] セクションで、[手動] を選択します。
6. アクセスする EC2 インスタンスを選択してから、[作成] を選択します。

インスタンスに接続してコマンドを実行する

Note

次の手順は、新しい EC2 コンソール用です。

1. Amazon EC2 コンソールのナビゲーションペインで、[インスタンス] を選択し、接続するインスタンスを選択します。
2. 接続 を選択します。

[Connect] (接続) がアクティブ化されていない場合は、最初にインスタンスを起動する必要があります。
3. [Connect to your instance] (インスタンスへ接続) ペインの [Connection method] (接続方法) で、[Session Manager]、[Connect] (接続) の順に選択します。
4. 表示されたターミナルセッションウィンドウで、次のコマンドを入力します。これらのコマンドは、tmux ソケットを使用するための適切なアクセス許可を持つ tmp フォルダを作成します。

```
sudo mkdir /tmp
```

```
sudo chmod 777 /tmp
sudo rmdir /tmp/tmux-*
```

[\(先頭に戻ります\)](#)

Amazon EC2

次のセクションでは、Amazon EC2 に関連する問題のトラブルシューティングについて概説します。

Amazon EC2 インスタンスが自動的に更新されない

問題: 最近のシステム更新は、AWS Cloud9 開発環境に接続する Amazon EC2 インスタンスに自動的に適用されません。

原因: 事前の情報や承認なく自動的に最近の更新を適用すると、コードやAmazon EC2 インスタンスで予期しない動作が発生する可能性があります。

推奨される解決策:

「Amazon EC2 ユーザーガイド」の「[インスタンスソフトウェアの更新](#)」にある指示に従って、定期的に Amazon EC2 インスタンスにシステム更新を適用します。

インスタンスでコマンドを実行するには、インスタンスに接続されている環境から IDE AWS Cloud9 のターミナルセッションを使用できます。

また、ssh や PuTTY などの SSH リモートアクセスユーティリティを使用してインスタンスに接続できます。これを行うには、ローカルコンピュータから、ssh-keygen または PuTTYgen などの SSH キーペア作成ユーティリティを使用します。インスタンスに接続されている環境の AWS Cloud9 IDE を使用して、生成されたパブリックキーをインスタンスに保存します。次に、生成済みのプライベートキーと共に SSH リモートアクセスユーティリティを使用して、インスタンスにアクセスします。詳細については、ユーティリティのドキュメントを参照してください。

AWS CLI または AWS シェルエラー: EC2 環境の「リクエストに含まれるセキュリティトークンが無効です」

問題: AWS Command Line Interface (AWS CLI) または AWS-shell を使用して EC2 環境の IDE AWS Cloud9 でコマンドを実行しようとする、「リクエストに含まれるセキュリティトークンが無効です」というエラーが表示されます。

原因: AWS マネージド一時認証情報があり、以下のいずれかが発生すれば、セキュリティトークンが無効になる可能性があります。

- AWS マネージド一時認証情報では許可されていないコマンドを実行しようとした。許可されたコマンドの一覧については、「[AWS マネージド一時認証情報でサポートされるアクション](#)」を参照してください。
- AWS マネージド一時認証情報は 15 分後に自動的に期限切れになります。
- 環境所有者以外のユーザーが新しいメンバーを追加したため、共有環境の AWS マネージド一時認証情報が非アクティブ化されました。

推奨される解決策:

- AWS マネージド一時認証情報で許可されているコマンドのみを実行します。AWS マネージド一時認証情報で許可されていないコマンドを実行する必要がある場合は、環境の AWS CLI または AWS シェルを永続的な認証情報のセットで設定します。これにより、この制限がなくなります。手順については、「[環境に永続的アクセス認証情報を作成して保存する](#)」を参照してください。
- 非アクティブ化または期限切れの認証情報については、が環境内の一時的な認証情報 AWS Cloud9 を更新できるように、環境所有者が環境を開いていることを確認してください。詳細については、「[AWS マネージド一時認証情報へのアクセスの制御](#)」を参照してください。

VPC の IP アドレスを Docker が使用しているため、EC2 環境に接続できません

問題: EC2 環境で、IPv4 クラスレスドメイン間ルーティング (CIDR) ブロック 172.17.0.0/16 を使用する Amazon VPC で EC2 インスタンスを起動した場合、その環境を開こうとすると、接続が停止することがあります。

原因: Docker は、同じブリッジネットワークに接続されているコンテナが通信できるようにするブリッジネットワークと呼ばれるリンクレイヤーデバイスを使用します。は、コンテナ通信にデフォルトのブリッジを使用するコンテナ AWS Cloud9 を作成します。デフォルトのブリッジは、コンテナのネットワークに通常 172.17.0.0/16 サブネットを使用します。

環境のインスタンスの VPC サブネットが、Docker で既に使用しているのと同じアドレス範囲を使用している場合、IP アドレスの競合が発生する可能性があります。したがって、AWS Cloud9 がインスタンスに接続しようとする、その接続はゲートウェイルートテーブルによって Docker ブリッジにルーティングされます。これにより、は開発環境をバックアップする EC2 インスタンスに接続 AWS Cloud9 できなくなります。

推奨される解決策: Amazon VPC と Docker が同じ IPv4 CIDR アドレスブロックを使用することで発生する IP アドレスの競合を解決するには、EC2 環境をバックアップするインスタンス用に新しい VPC を設定します。この新しい VPC では、172.17.0.0/16 とは異なる CIDR ブロックを設定します (既存の VPC またはサブネットの IP アドレスの範囲を変更することはできません)。

設定情報については、「Amazon VPC ユーザーガイド」の「[VPC とサブネットサイジング](#)」を参照してください。

IDE File Explorer でサブフォルダ構造 `/home/ec2-user/environment/home/ec2-user/environment` AWS Cloud9 を作成することはできません。

問題: IDE File Explorer でサブフォルダ構造 `/home/ec2-user/environment/home/ec2-user/environment` AWS Cloud9 を作成すると、このディレクトリを開くことができないことを示すエラーメッセージが表示されます。

考えられる原因: 現在、IDE のファイルシステムを使用して、同じ名前のフォルダ内にサブフォルダ構造 `/home/ec2-user/environment` AWS Cloud9 を作成することはできません。AWS Cloud9 IDE File Explorer からこのディレクトリ内のファイルにはアクセスできませんが、コマンドラインを使用してアクセスできます。この問題の影響を受けるのは `/home/ec2-user/environment/home/ec2-user/environment` のファイルパスだけです。`/test/home/ec2-user/environment` や `/home/ec2-user/environment/test` などのファイルパスは機能します。これは既知の問題であり、IDE File Explorer AWS Cloud9 にのみ影響します。

推奨される解決策: 別のファイル名と構造を使用してください。

AWS License Manager ライセンス設定が Amazon EC2 インスタンスに関連付けられている場合、コンソール AWS Cloud9 から起動できない

問題: コンソールから AWS Cloud9 EC2 環境を起動しようとする、エラーメッセージ `unable to access your environment` が返されます。

考えられる原因: AWS License Manager 全体でソフトウェアベンダーライセンスの管理を合理化します AWS クラウド。License Manager を設定するときは、エンタープライズ契約の条項に基づくライセンスルールのセットであるライセンス設定を作成します。これらのライセンス設定は、Amazon マシンイメージ (AMI) や などのメカニズムにアタッチできます CloudFormation。これらのメカニズムのいずれかを使用して、EC2 インスタンスを起動できます。

`AWSServiceRoleForAWSCloud9` サービスにリンクされたロール (SLR) の古いバージョンの `AWSCloud9ServiceRolePolicy` には現在、`license-configuration` リソース条件が含まれてい

ません。このため、AWS Cloud9 はインスタンスを起動および停止することはできません。そのため、AWS Cloud9 は Amazon EC2 インスタンスへのアクセスを拒否され、エラーが返されます。

推奨される解決策: 既存の AWS Cloud9 環境にアクセスして License Manager を使用できない場合は、古い `AWSCloud9ServiceRolePolicy` サービスにリンクされたロールを、`ガ`インスタンス `license-configuration` に適用されたときに EC2 アクションを明示的に許可する [SLR のバージョン](#) に置き換えます。古いロールを削除するだけで置き換えることができます。その後、更新されたロールが自動的に作成されます。

EC2 環境で一部のコマンドやスクリプトを実行できない

問題: AWS Cloud9 EC2 開発環境を開いた後、一部のタイプのパッケージをインストールしたり、`yum` やなどのコマンドを実行したり `apt`、他の Linux オペレーティングシステムで通常機能するコマンドを含むスクリプトを実行したりすることはできません。

原因: `ガ` `EC2` インスタンスは、Amazon Linux (Red Hat Enterprise Linux (RHEL) に基づく) または Ubuntu Server のいずれかに依存します。AWS Cloud9

解決策: EC2 環境用に IDE でパッケージをインストールしたり管理したり、コマンドやスクリプトを実行したりする場合は、その環境のインスタンスに応じて、RHEL (Amazon Linux用) または Ubuntu Server と互換性があることを確認してください。

を使用して EC2 環境を作成するときに「インスタンスプロファイル `AWSCloud9SSMInstanceProfile` がアカウントに存在しません」と報告するエラーメッセージ CloudFormation

問題: [AWS::Cloud9::EnvironmentEC2](#) CloudFormation resource を使用して EC2 環境を作成すると、ユーザーはインスタンスプロファイル `AWSCloud9SSMInstanceProfile` がアカウントに存在しないというエラーメッセージを受け取ります。

原因: `no-ingress` EC2 環境を作成するときは、サービスロール `AWSCloud9SSMAccessRole` およびインスタンスプロファイル `AWSCloud9SSMInstanceProfile` を作成する必要があります。これらの IAM リソースにより、Systems Manager による開発環境をバックアップする EC2 インスタンスを管理が有効になります。

コンソールで `no-ingress` 環境を作成する場合、`AWSCloud9SSMAccessRole` および `AWSCloud9SSMInstanceProfile` は自動的に作成されます。ただし、CloudFormation または を使用して最初の非進入環境 AWS CLI を作成する場合は、これらの IAM リソースを手動で作成する必要があります。

推奨される解決策: CloudFormation テンプレートの編集と IAM アクセス許可の更新については、「」を参照してください。 [CloudFormation を使用して非進入 EC2 環境を作成する](#)

を使用して EC2 環境を作成するときに「リソースperform:
ssm:StartSessionで」と報告されるエラーメッセージ CloudFormation

問題: [AWS::Cloud9::EnvironmentEC2](#) CloudFormation リソースを使用して EC2 環境を作成するAccessDeniedException場合、ユーザーは を受け取り、「リソースssm:StartSessionで を実行する権限がありません」と通知されます。

原因: ユーザーには no-ingress インスタンス用Systems Manager を活用する EC2 環境の設定パートとして必要なStartSession API を呼び出す許可がありません。

推奨される解決策: CloudFormation テンプレートの編集と IAM アクセス許可の更新については、「」を参照してください[CloudFormation を使用して非進入 EC2 環境を作成する](#)。

を使用して EC2 環境を作成するときに「実行する:
iam:GetInstanceProfile on resource: instance
profile**AWSCloud9SSMInstanceProfile**」という承認がないと報告する
エラーメッセージ AWS CLI

問題: [AWS CLI](#) を使用して EC2 環境を作成するときに、ユーザーは AccessDeniedException を受け取り、AWS Cloud9 環境には「リソース (インスタンスプロファイル AWSCloud9SSMInstanceProfile) に対して iam:GetInstanceProfile を実行する権限がありません」と通知されます。

原因: AWS Cloud9 Ingress インスタンスに Systems Manager を使用する EC2 環境の設定の一部として必要な StartSession API を呼び出すアクセス許可を付与します。

推奨される解決策: 必要なAWSCloud9SSMAccessRoleサービスロールと AWSCloud9SSMInstanceProfile を AWS Cloud9 環境に追加する方法については、「」を参照してください[を使用した Systems Manager のインスタンスプロファイルの管理 AWS CLI](#)。

デフォルトの暗号化が Amazon EBS ボリュームに適用されている場合に、環境の作成障害がでる

問題: Failed to create environments. The development environment '[environment-ID]' failed to create Amazon EC2 環境を作成しようとすると、エラーが返されます。

考えられる原因: AWS Cloud9 IDE がデフォルトで暗号化されている Amazon EBS ボリュームを使用している場合、AWS Identity and Access Management のサービスにリンクされたロール AWS Cloud9 は、これらの EBS ボリューム AWS KMS keys の にアクセスする必要があります。アクセスが提供されていない場合、IDE AWS Cloud9 が起動に失敗し、問題のデバッグが困難になる可能性があります。

推奨される解決策: アクセスを提供するには、Amazon EBS AWS Cloud9ボリュームで使用されるカスタマーマネージドキーにAWSServiceRoleForAWSCloud9、 のサービスにリンクされたロールを追加します。

このタスクの詳細については、「[規範ガイダンスパターンAWS Cloud9](#)」の「[デフォルトの暗号化で Amazon EBS ボリュームを使用する を作成する](#)」を参照してください。AWS

EC2-Classic アカウントの VPC エラー: 「環境にアクセスできません」

問題: EC2-Classic は、Amazon EC2 のオリジナルリリースで導入されました。2013 年 12 月 4 日より前にセットアップ AWS アカウント された を使用する場合、AWS Cloud9 EC2 開発環境の作成時に Amazon VPC とサブネットを設定しないと、このエラーが発生する可能性があります。

デフォルトの VPC 設定を受け入れると、Amazon EC2 インスタンスは EC2-Classic ネットワーク内で起動されます。インスタンスはデフォルト VPC のサブネットで起動されません。環境の作成に失敗すると、次のメッセージが表示されます。

環境エラー

環境にアクセスできません

環境の作成はエラーで失敗しました: 次のリソースの作成に失敗しました: [インスタンス]。ユーザーからロールバックがリクエストされました。

EC2 インスタンスがデフォルト VPC に存在しないためにエラーが発生したことを確認できます。を使用して CloudFormation 、開発環境のスタックイベント履歴を表示します。

1. CloudFormation コンソールを開きます。詳細については、「[CloudFormation コンソールにログイン](#)」を参照してください。
2. CloudFormation コンソールで、スタックを選択します。
3. [スタック] ページで、作成に失敗した開発環境の名前を選択します。
4. [Stack details] (スタックの詳細) ページで [Events] (イベント) タブを選択し、次のエントリをチェックします。

ステータス: CREATE_FAILED

ステータスの理由: AssociatePublicIpAddress パラメータは、VPC の起動でのみサポートされています。 [...]

原因: AWS Cloud9 開発環境は、特定の VPC 要件を満たす Amazon VPC に関連付ける必要があります。EC2-Classic が有効なアカウントの場合、[EC2 環境の作成](#) 時にデフォルトのネットワーク設定を受け入れると、必要な EC2 インスタンスが VPC 内で起動されません。代わりに、インスタンスは EC2-Classic ネットワーク内で起動されます。

推奨される解決策: EC2-Classic アカウントでは、[EC2 環境の作成](#)時に VPC とサブネットを選択する必要があります。[設定の構成] ページの [ネットワーク設定 (アドバンスド)] セクションで、EC2 インスタンスを起動できる VPC とサブネットを選択します。

その他の AWS サービス

次のセクションでは、他の AWS サービスに関連する問題のトラブルシューティングについて概説します。

CodeCatalyst の IDE の File Explorer 内にサブフォルダ構造 /projects/projects AWS Cloud9 を作成することはできません。

問題: CodeCatalyst の AWS Cloud9 IDE File Explorer でサブフォルダ構造 /projects/projects を作成すると、このディレクトリを開くことができないことを示すエラーメッセージが表示されます。

考えられる原因: 現在、CodeCatalyst の AWS Cloud9 IDE の File Explorer を使用して、同じ名前のフォルダ内にサブフォルダ構造/プロジェクトを作成することはできません。AWS Cloud9 IDE File Explorer からこのディレクトリ内のファイルにはアクセスできませんが、コマンドラインを使用してアクセスできます。この問題は /projects/projects のファイルパスにのみ影響します。/test/projects

や `/projects/test` などのファイルパスは正常に機能します。これは既知の問題で、AWS Cloud9 IDE ファイルエクスプローラーにのみ影響します。

推奨される解決策：別のファイル名と構造を使用してください。

IDE の外部で実行中のアプリケーションを表示できない

問題: IDE 外部のウェブブラウザタブで実行中のアプリケーションを表示しようとする、そのウェブブラウザタブがエラーを表示するか、空白になります。

考えられる原因:

- アプリケーションが IDE で実行していません。
- アプリケーションが `127.0.0.1` または `localhost` の IP で実行しています。
- アプリケーションは AWS Cloud9 EC2 開発環境で実行されています。さらに、Amazon EC2 インスタンスに関連付けられているセキュリティグループの 1 つ以上が、アプリケーションで必要とするプロトコル、ポート、または IP アドレスを介したインバウンドトラフィックを許可していません。
- アプリケーションは、AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) AWS Cloud9 の SSH 開発環境で実行されています。さらに、対応するインスタンスに関連付けられている仮想プライベートクラウド (VPC) のサブネットのネットワーク ACL が、アプリケーションで必要とするプロトコル、ポート、または IP アドレスを介したインバウンドトラフィックを許可していません。
- URL に誤りがある。
- インスタンスのパブリック IP アドレスではなく、アプリケーションプレビュータブの URL がリクエストされている。
- `127.0.0.1` または `localhost` の IP を含むアドレスに移動しようとしています。これらの IP は、環境のリソースではなく、ローカルコンピュータのリソースにアクセスしようとしています。
- インスタンスのパブリック IP アドレスが変更されています。
- ウェブリクエストが、アプリケーションで必要とするプロトコル、ポート、または IP アドレス経由のトラフィックをブロックする仮想プライベートネットワーク (VPN) から送信されています。
- アプリケーションが SSH 環境で実行しています。ただし、サーバーまたは関連付けられたネットワークが、アプリケーションで必要とするプロトコル、ポート、または IP アドレスを介したインバウンドトラフィックを許可していません。

推奨される解決策:

- IDE でアプリケーションが実行されていることを確認します。
- アプリケーションが 127.0.0.1 または localhost の IP を使用して実行されていないことを確認します。Node.js および Python の例については、「[アプリケーションの実行](#)」を参照してください。
- アプリケーションが AWS クラウドコンピューティングインスタンス (Amazon EC2 インスタンスなど) で実行されているとします。この場合、対応するインスタンスに関連付けられているすべてのセキュリティグループが、アプリケーションで必要とするプロトコル、ポート、および IP アドレスを介したインバウンドトラフィックを許可していることを確認します。指示については、実行中のアプリケーションをインターネット上で共有するの [ステップ 2: インスタンスのセキュリティグループを設定する](#) を参照してください。「Amazon VPC ユーザーガイド」の「[VPC のセキュリティグループ](#)」を参照してください。
- アプリケーションが AWS クラウドコンピューティングインスタンスで実行されているとします。さらに、対応するインスタンスに関連付けられた VPC のサブネットにネットワーク ACL が存在するとします。この場合、ネットワーク ACL が、アプリケーションで必要とするプロトコル、ポート、および IP アドレスを介したインバウンドトラフィックを許可していることを確認します。指示については、実行中のアプリケーションをインターネット上で共有するの [ステップ 3: インスタンスのサブネットをセットアップする](#) を参照してください。また、「Amazon VPC ユーザーガイド」の「[ネットワーク ACL](#)」を参照してください。
- プロトコル (およびポートを指定する必要がある場合はポート) を含むリクエストしている URL が正しいことを確認します。指示については、実行中のアプリケーションをインターネット上で共有するの [ステップ 4: 実行中のアプリケーションの URL を共有する](#) を参照してください。
- 形式 `https://12a34567b8cd9012345ef67abcd890e1.vfs.cloud9.us-east-2.amazonaws.com/` (12a34567b8cd9012345ef67abcd890e1 は環境 AWS Cloud9 に割り当てる ID、は環境の AWS リージョンの us-east-2 ID) の URL をリクエストすることはお勧めしません。この URL を使用するには、環境の IDE が開いており、アプリケーションが同じウェブブラウザで実行されている必要があります。
- 127.0.0.1 または localhost の IP を含むアドレスに移動しようとしているとします。代わりに、実行中のアプリケーションの正しい非ローカルアドレスに移動してみてください。詳細については、「[実行中のアプリケーションをインターネット経由で共有する](#)」を参照してください。
- アプリケーションが AWS クラウドコンピューティングインスタンスで実行されているとします。インスタンスのパブリック IP アドレスが変更されているかどうかを確認します。インスタンスのパブリック IP アドレスは、インスタンスが再起動されると変更される可能性があります。この IP アドレスの変更を防ぐには、Elastic IP アドレスを割り当てて、それを実行中のインスタンスに割り当てます。指示については、実行中のアプリケーションをインターネット上で共有するの [ステップ 4: 実行中のアプリケーションの URL を共有する](#) を参照してください。

- ウェブリクエストが VPN から送信されている場合は、その VPN でアプリケーションで必要とするプロトコル、ポート、および IP アドレス経由のトラフィックが許可されていることを確認します。VPN を変更できない場合は、ネットワーク管理者にお問い合わせください。または、可能であれば、別のネットワークからウェブリクエストを行います。
- アプリケーションが独自のサーバーの SSH 環境で実行しているとします。サーバーおよび関連付けられているネットワークが、アプリケーションで必要とするプロトコル、ポート、または IP アドレスを介したインバウンドトラフィックを許可していることを確認します。サーバーまたは関連付けられているネットワークを変更できない場合は、サーバーまたはネットワークの管理者にお問い合わせください。
- `curl` コマンドに URL を続けて実行し、環境のターミナルからアプリケーションの実行を試みます。このコマンドでエラーメッセージが表示される場合は、関連しない他の問題がある可能性があります。AWS Cloud9。

AWS Toolkit の実行中にエラーが発生しました：「環境が inodes を使い果たしています。「fs.inotify.max_user_watches」の制限を増やしてください。」

問題: AWS Toolkit が使用するファイル監視ユーティリティが、監視できるファイルの現在の制限またはクォータに近づいています。

Cause: AWS Toolkit は、ファイルとディレクトリへの変更をモニタリングするファイル監視ユーティリティを使用します。ユーティリティが監視できるファイル数が現在のクォータに近づくと、警告メッセージが表示されます。

推奨される解決策: ファイルウォッチャーで処理できるファイルの最大数を増やすには、次の操作を行います。

1. ターミナルセッションをスタートするには、メニューバーで、[Window (ウィンドウ)]、[New Terminal (新しいターミナル)] を選択します。
2. 次のコマンドを入力します。

```
sudo bash -c 'echo "fs.inotify.max_user_watches=524288" >> /etc/sysctl.conf' &&  
sudo sysctl -p
```

Lambda ローカル関数実行エラー: SAM Local をインストールできない

問題: IDE AWS Cloud9 で AWS Lambda 関数のローカルバージョンを実行しようとする、ダイアログボックスが表示されます。ダイアログボックスには、AWS Cloud9 で SAM Local のインストールに問題があることが示されます。では、ローカルバージョンの AWS Lambda 関数を IDE で実行するために SAM Local AWS Cloud9 が必要です。SAM Local をインストールするまで、IDE でローカルバージョンの Lambda 関数を実行することはできません。

Cause: AWS Cloud9 環境内の想定されるパスに SAM Local が見つかりません~/.c9/bin/sam。これは SAM Local がまだインストールされていないか、インストールされていても AWS Cloud9 がその場所に見つけれないことが原因です。

推奨される解決策: AWS Cloud9 が SAM Local のインストールを完了するのを待つか、自分でインストールできます。

AWS Cloud9 が SAM Local のインストールを試みている様子を確認するには、メニューバーで Window, Installer を選択します。

SAM Local を自分でインストールするには、「[デベロッパーガイド](#)」の「[Linux AWS での SAM CLI のインストール](#)」の手順に従います。AWS Serverless Application Model

AWS Control Tower AWS Cloud9を使用して Amazon EC2 環境を作成しようとする、「環境の作成がエラーで失敗しました: 次のフック (複数可) が失敗しました:[ControlTower::Guard::Hook]」。

問題: AWS Cloud9 および AWS Control Tower プロアクティブコントロール CT.EC2.PR.8 との互換性の問題があります。このコントロールが有効な場合、AWS Cloud9で EC2 環境を作成することはできません。

原因: AWS Control Tower は、AssociatePublicIpAddress パラメータが AWS CloudFormation テンプレートにあることを想定しています。現在、このパラメータは追加できません。

推奨される解決策: AWS Control Tower コンソールから controlCT.EC2.PR.8 を無効にし、環境を再作成します AWS Cloud9。

デフォルトの暗号化が Amazon EBS ボリュームに適用されている場合に、環境の作成障害がでる

問題: Failed to create environments. The development environment '[environment-ID]' failed to create Amazon EC2 環境を作成しようとすると、エラーが返されます。

考えられる原因: AWS Cloud9 IDE がデフォルトで暗号化されている Amazon EBS ボリュームを使用している場合、AWS Identity and Access Management のサービスにリンクされたロール AWS Cloud9 は、これらの EBS ボリューム AWS KMS keys の にアクセスする必要があります。アクセスが提供されていない場合、IDE AWS Cloud9 が起動に失敗し、問題のデバッグが困難になる可能性があります。

推奨される解決策: アクセスを提供するには、Amazon EBS AWS Cloud9ボリュームで使用されるカスタマーマネージドキーにAWSServiceRoleForAWSCloud9、 のサービスにリンクされたロールを追加します。

このタスクの詳細については、「[規範ガイダンスパターンAWS Cloud9](#)」の「[デフォルトの暗号化で Amazon EBS ボリュームを使用する を作成する](#)」を参照してください。AWS

[\(先頭に戻ります\)](#)

AWS License Manager ライセンス設定が Amazon EC2 インスタンスに関連付けられている場合、コンソール AWS Cloud9 から起動できない

問題: コンソールから AWS Cloud9 EC2 環境を起動しようとすると、エラーメッセージunable to access your environmentが返されます。

考えられる原因: AWS License Manager 全体でソフトウェアベンダーライセンスの管理を合理化します AWS クラウド。License Manager を設定するときは、エンタープライズ契約の条項に基づくライセンスルールのセットであるライセンス設定を作成します。これらのライセンス設定は、Amazon マシンイメージ (AMI) や などのメカニズムにアタッチできます CloudFormation。これらのメカニズムのいずれかを使用して、EC2 インスタンスを起動できます。

AWSServiceRoleForAWSCloud9 サービスにリンクされたロール (SLR) の古いバージョンの AWSCloud9ServiceRolePolicy には現在、license-configuration リソース条件が含まれていません。このため、AWS Cloud9 はインスタンスを起動および停止することはできません。そのため、AWS Cloud9 は Amazon EC2 インスタンスへのアクセスを拒否され、エラーが返されます。

推奨される解決策: 既存の AWS Cloud9 環境にアクセスして License Manager を使用できない場合は、古い AWSCloud9ServiceRolePolicy サービスにリンクされたロールを、`ガインスタンスlicense-configuration`に適用されたときに EC2 アクションを明示的に許可する [SLR のバージョン](#)に置き換えます。古いロールを削除するだけで置き換えることができます。その後、更新されたロールが自動的に作成されます。

[\(先頭に戻ります\)](#)

アプリケーションプレビュー

次のセクションでは、アプリケーションプレビューに関連する問題のトラブルシューティングについて概説します。

環境を再ロードした後、アプリケーションプレビューを最新の情報に更新する必要がある

問題: アプリケーションプレビュータブを表示する環境を再ロードした後、タブにアプリケーションプレビューが表示されません。

原因: 無限ループを実行できるコードをユーザーが書くことがあります。または、コードで大量のメモリが使用されるため、アプリケーションプレビューの実行中に AWS Cloud9 IDE が一時停止または停止する可能性があります。これを防ぐには、環境が再ロードされるたびにアプリケーションプレビュータブを再ロード AWS Cloud9 しません。

解決策: アプリケーションプレビュータブを表示する環境を再ロードした後で、アプリケーションプレビューを表示するには、タブの [Click to load the page (クリックしてページをロード)] ボタンを選択します。

アプリケーションプレビューまたはファイルプレビュー通知: 「サードパーティーの Cookie が無効になっています」

問題: [アプリケーション](#)または[ファイルを](#)プレビューしようとする、次のメッセージと通知が表示されます。「ブラウザでサードパーティーの Cookie が無効になっているため、プレビュー機能が無効になっています。」

原因: IDE を開くためにサードパーティーの Cookie AWS Cloud9 は必要ありません。ただし、アプリケーションプレビュー機能またはファイルプレビュー機能を使用するには、サードパーティーの Cookie を有効にする必要があります。

解決方法: ウェブブラウザでサードパーティーの Cookie を有効にし、IDE を再読み込みしてから、もう一度プレビューを開いてみてください。

- Apple Safari: Apple サポートウェブサイトで「[Safari で Cookie と Web サイトのデータを管理する](#)」を参照してください。
- Google Chrome: Google Chrome ヘルプウェブサイトで「[Chrome で Cookie の削除、有効化、管理を行う](#)」の「Cookie の設定を変更する」を参照してください。
- Internet Explorer: Microsoft サポートウェブサイトで「Cookie の削除と管理を行う」の「[Cookie のブロックまたは許可](#)」を参照してください。
- Microsoft Edge: Microsoft サポートウェブサイトで「[サードパーティの Cookie をブロックする](#)」を参照してください。
- Mozilla Firefox: Mozilla サポートウェブサイトで「[Enable and disable cookies that websites use to track your preferences](#) (設定を追跡するためウェブサイトで使用する Cookie を有効または無効にする)」の「Accept third party cookies (サードパーティーの Cookie を受け入れる)」を参照してください。
- 他のウェブブラウザの場合: そのウェブブラウザのドキュメントを参照してください。

お使いのウェブブラウザでこの精度が許容される場合は、AWS Cloud9に対してのみサードパーティーの Cookie を有効にできます。そのためには、AWS Cloud9を使用するサポート対象のドメインに応じて、次のドメインを指定します。

AWS リージョン	ドメイン
米国東部(バージニア北部)	*.vfs.cloud9.us-east-1.amazonaws.com vfs.cloud9.us-east-1.amazonaws.com
米国東部 (オハイオ)	*.vfs.cloud9.us-east-2.amazonaws.com vfs.cloud9.us-east-2.amazonaws.com
米国西部 (北カリフォルニア)	*.vfs.cloud9.us-west-1.amazonaws.com

AWS リージョン	ドメイン
	<code>vfs.cloud9.us-west-1.amazonaws.com</code>
米国西部 (オレゴン)	<code>*.vfs.cloud9.us-west-2.amazonaws.com</code> <code>vfs.cloud9.us-west-2.amazonaws.com</code>
アフリカ (ケープタウン)	<code>*.vfs.cloud9.af-south-1.amazonaws.com</code> <code>vfs.cloud9.af-south-1.amazonaws.com</code>
アジアパシフィック (香港)	<code>*.vfs.cloud9.ap-east-1.amazonaws.com</code> <code>vfs.cloud9.ap-east-1.amazonaws.com</code>
アジアパシフィック (ムンバイ)	<code>*.vfs.cloud9.ap-south-1.amazonaws.com</code> <code>vfs.cloud9.ap-south-1.amazonaws.com</code>
アジアパシフィック (大阪)	<code>*.vfs.cloud9.ap-northeast-3.amazonaws.com</code> <code>vfs.cloud9.ap-northeast-3.amazonaws.com</code>
アジアパシフィック (ソウル)	<code>*.vfs.cloud9.ap-northeast-2.amazonaws.com</code> <code>vfs.cloud9.ap-northeast-2.amazonaws.com</code>

AWS リージョン	ドメイン
アジアパシフィック (シンガポール)	<code>*.vfs.cloud9.ap-southeast-1.amazonaws.com</code> <code>vfs.cloud9.ap-southeast-1.amazonaws.com</code>
アジアパシフィック (シドニー)	<code>*.vfs.cloud9.ap-southeast-2.amazonaws.com</code> <code>vfs.cloud9.ap-southeast-2.amazonaws.com</code>
アジアパシフィック (東京)	<code>*.vfs.cloud9.ap-northeast-1.amazonaws.com</code> <code>vfs.cloud9.ap-northeast-1.amazonaws.com</code>
カナダ (中部)	<code>*.vfs.cloud9.ca-central-1.amazonaws.com</code> <code>vfs.cloud9.ca-central-1.amazonaws.com</code>
欧州 (フランクフルト)	<code>*.vfs.cloud9.eu-central-1.amazonaws.com</code> <code>vfs.cloud9.eu-central-1.amazonaws.com</code>
欧州 (アイルランド)	<code>*.vfs.cloud9.eu-west-1.amazonaws.com</code> <code>vfs.cloud9.eu-west-1.amazonaws.com</code>

AWS リージョン	ドメイン
欧州 (ロンドン)	<code>*.vfs.cloud9.eu-west-2.amazonaws.com</code> <code>vfs.cloud9.eu-west-2.amazonaws.com</code>
欧州 (ミラノ)	<code>*.vfs.cloud9.eu-south-1.amazonaws.com</code> <code>vfs.cloud9.eu-south-1.amazonaws.com</code>
ヨーロッパ (パリ)	<code>*.vfs.cloud9.eu-west-3.amazonaws.com</code> <code>vfs.cloud9.eu-west-3.amazonaws.com</code>
ヨーロッパ (ストックホルム)	<code>*.vfs.cloud9.eu-north-1.amazonaws.com</code> <code>vfs.cloud9.eu-north-1.amazonaws.com</code>
中東 (バーレーン)	<code>*.vfs.cloud9.me-south-1.amazonaws.com</code> <code>vfs.cloud9.me-south-1.amazonaws.com</code>
南米 (サンパウロ)	<code>*.vfs.cloud9.sa-east-1.amazonaws.com</code> <code>vfs.cloud9.sa-east-1.amazonaws.com</code>

アプリケーションプレビュータブにエラーが表示される、または空白になる

問題: IDEのメニューバーで、[プレビュー、実行中のアプリケーションのプレビュー)] または [ツール、プレビュー、実行中のアプリケーションのプレビュー)] を選択して IDEのプレビュータブでアプリケーションを表示しようとする、タブにエラーが表示されるか、タブが空白になります。

考えられる原因:

- アプリケーションが IDE で実行していません。
- アプリケーションが HTTP を使用して実行していません。
- アプリケーションが複数のポート経由で実行されている。
- アプリケーションが 8080、8081、または 8082 以外のポート経由で実行されている。
- アプリケーションが 127.0.0.1、localhost、または 0.0.0.0 以外の IP で実行されている。
- ポート (8080、8081、または 8082) がプレビュータブの URL に指定していません。
- ネットワークがポート 8080、8081、または 8082 へのインバウンドトラフィックをブロックしています。
- 127.0.0.1、localhost、または 0.0.0.0 の IP を含むアドレスに移動しようとしています。デフォルトでは、IDE AWS Cloud9 はローカルコンピュータに移動しようとします。環境に接続されているインスタンスや独自のサーバーに移動しようとはしません。

推奨される解決策:

- IDE でアプリケーションが実行されていることを確認します。
- HTTP を使用してアプリケーションが実行されていることを確認します。Node.js および Python の例については、「[アプリケーションの実行](#)」を参照してください。
- アプリケーションが 1 つのポート経由のみで実行されていることを確認します。Node.js および Python の例については、「[アプリケーションの実行](#)」を参照してください。
- アプリケーションがポート 8080、8081、または 8082 経由で実行されていることを確認します。Node.js および Python の例については、「[アプリケーションの実行](#)」を参照してください。
- アプリケーションが IP 127.0.0.1、localhost、または 0.0.0.0 を使用して実行されていることを確認します。Node.js および Python の例については、「[アプリケーションの実行](#)」を参照してください。

- プレビュータブの URL に :8080、:8081、または :8082 を追加します。
- ネットワークがポート 8080、8081、または 8082 経由のインバウンドトラフィックを許可していることを確認します。ネットワークを変更できない場合は、ネットワーク管理者にお問い合わせください。
- 127.0.0.1、localhost、または 0.0.0.0 の IP を含むアドレスに移動しようとしている場合は、代わりに `https://12a34567b8cd9012345ef67abcd890e1.vfs.cloud9.us-east-2.amazonaws.com/` のアドレスに移動してみてください。このアドレスの場合、12a34567b8cd9012345ef67abcd890e1 は AWS Cloud9 が環境に割り当てる ID です。us-east-2 は環境の AWS リージョンの ID です。IDE の外部で、このアドレスに移動することもできます。ただし、これが可能なのは、環境の IDE が開いていて、アプリケーションが同じウェブブラウザで実行している場合のみです。
- 上記の条件がすべて満たされていることを確認した後で、アプリケーションを停止してから再度起動してみてください。
- アプリケーションを停止してから再度起動した場合は、メニューバーで再度 [Preview (プレビュー)]、[Preview Running Application (実行中のアプリケーションのプレビュー)] または [Tools (ツール)]、[Preview (プレビュー)]、[Preview Running Application (実行中のアプリケーションのプレビュー)] を選択します。対応するアプリケーションプレビュータブが既に表示されている場合は、そのタブで [Refresh (最新の情報に更新)] ボタン (円状の矢印) を選択します。

サイトへの接続が安全でないため、IDE で ウェブコンテンツをプレビューできません

問題: AWS Cloud9 EC2 環境でホストされている WordPress サイトなどのウェブコンテンツにアクセスしようとすると、IDE プレビューウィンドウには表示されません。

考えられる原因: デフォルトでは、IDE のアプリケーションプレビュータブでアクセスするすべてのウェブページは自動的に HTTPS AWS Cloud9 プロトコルを使用します。ページの URI が安全でない http プロトコルを備えている場合、https に自動的に置き換えられます。また、手動で https を http に戻す変更を行っても、安全でないコンテンツにアクセスすることはできません。

推奨される解決策: IDE でプレビューしようとしている ウェブサイトから安全ではない HTTP スクリプトまたはコンテンツを削除します。HTTPS の実装に関するガイダンスについては、ウェブサーバーまたはコンテンツ管理システムの指示に従ってください。

ファイルをプレビューすると 499 エラーが返される

問題: AWS Cloud9 IDE を使用して、src 属性を含み、属性が type に設定されている `<script>` 要素を含むファイルをプレビューしようとする module、499 エラーが発生し、スクリプトが期待どおりに実行されません。

原因: IDE AWS Cloud9 のファイルプレビューフェッチリクエストでは、認証のためにウェブブラウザから Cookie を送信する必要があります。デフォルトでは、ウェブブラウザは、通常のスクリプトリクエストに対して Cookie を送信します。crossorigin 属性を追加しない限り、モジュールスクリプトリクエストに対しては Cookie を送信しません。

解決策: crossorigin 属性を `<script>` 要素に追加します。例えば、`<script type="module" src="index.js" crossorigin></script>`。次に、変更したファイルを保存し、もう一度プレビューを試みます。

パフォーマンス

次のセクションでは、パフォーマンスに関連する問題のトラブルシューティングについて概説します。

AWS Cloud9 IDE の長時間のフリーズ

問題: 起動時および更新時に、IDE AWS Cloud9 ターミナルが長時間フリーズし、使用できなくなります。

原因: ご使用の環境に大量のファイルがあり、AWS Cloud9 のファイル監視モジュールによって再帰的に監視されている可能性があります。

推奨される解決策: ファイル監視の深さを減らし (最小値は 1)、大きいフォルダや、ソースコードに関係のないフォルダ (ビルド出力/アーティファクト、サードパーティパッケージ) を、無視するパターンに追加することを確認できます。これを行うには、[設定] > [ユーザー設定] > [ファイル監視] に移動します。これにより、AWS Toolkit の CodeLenses が正しく動作しないことに注意してください。

別の解決策としては、検索するファイルの最大数を減らして、ソースコードに関係のない大きなファイルやフォルダを無視することを確認します。これを行うには、[設定] > [プロジェクト設定] > [フォルダを指定して検索] に移動します。これにより、無視したフォルダはファイル検索に表示されなくなることに注意してください。

コンソールの警告: 「最小限のコード補完エンジンに切り替えます...」

問題: AWS Cloud9 コンソールで作業する場合 (IDE を開くときや IDE のウェブページを更新するときなど)、「1 つ以上のセッションまたは共同作業者がこの環境でアクティブです。メモリを節約するために最小限のコード補完エンジンに切り替えます。」というメッセージが表示される。このメッセージと相関して、コード補完の動作が遅いか断続的になっている可能性がある。

原因: コード補完エンジンを実行すると、環境からメモリと CPU サイクルが消費されます。さらに、共同作業者および追加のセッションごとに個別のコード補完エンジンが必要です。特に t2.nano やなどの小さなインスタンスサイズで大量のリソースを使用しないように t2.micro、は最小限のコード補完エンジン AWS Cloud9 に切り替えます。

推奨される解決策: 長期間にわたって頻繁に共同作業を行う場合は、EC2 環境の作成時により大きい Amazon EC2 インスタンスを選択します。または、SSH 環境をより容量の大きいインスタンスに接続します。

Note

より大きな Amazon EC2 インスタンスを選択すると、AWS アカウント に追加料金が発生する可能性があります。詳細については、[Amazon EC2 の料金表](#)を参照してください。

IDE による警告: 「この環境のメモリが不足しています」または「この環境の CPU ロードが高くなっています」

問題: IDE の実行中、「この環境のメモリが不足しています」または「この環境の CPU ロードが高くなっています」という内容のメッセージが表示される。

原因: IDE に、遅延やハングアップしないで実行を続けるのに必要なコンピューティングリソースがない可能性があります。

推奨される解決策:

- 実行中のプロセスを 1 つ以上停止し、使用可能なメモリを解放してください。これを行うには、メニューバーにある環境用 IDE で、[ツール、プロセスリスト)] を選択します。停止するプロセスごとに、プロセスを選択して [Force Kill (強制終了)] を選択します。
- 環境で swap ファイルを作成します。スワップファイルは、オペレーティングシステムが仮想メモリとして使用できる環境内のファイルです。

環境で現在スワップメモリを使用しているかどうかを確認するには、環境のターミナルセッションで **top** コマンドを実行します。スワップメモリが使用されている場合、出力にゼロ以外の Swap メモリ統計が表示されます (たとえば、Swap: 499996k total, 1280k used, 498716 free, 110672k cached)。リアルタイムのメモリ情報の表示をやめるには、Ctrl + C を押します。

スワップファイルを作成するには、環境で次のようなコマンドを実行します。

```
sudo fallocate --length 512MB /var/swapfile && sudo chmod 600 /var/swapfile && sudo  
mkswap /var/swapfile && echo '/var/swapfile swap swap defaults 0 0' | sudo tee -a /  
etc/fstab > /dev/null
```

前述のコマンドは以下の処理を実行します。

1. /var ディレクトリに swapfile という名前で 512 MB のファイルを作成します。
2. swapfile ファイルのアクセス権限を所有者のみの読み取り/書き込みに変更します。
3. swapfile ファイルをスワップファイルとして設定します。
4. /etc/fstab file に情報を書き込みます。これにより、システムをリブートするたびに、このスワップファイルを使用できます。

前述のコマンドを実行したら、このスワップファイルをすぐに使用可能にするために、次のコマンドを実行します。

```
sudo swapon /var/swapfile
```

- 環境をコンピューティングリソースの多いインスタンスまたはサーバーに移動するか、サイズを変更します。Amazon EC2 インスタンスを移動またはサイズ変更するには、「[Amazon EBS AWS Cloud9 ポリウムからの IDE の移動](#)」を参照してください。他のインスタンスまたはサーバータイプについては、インスタンスまたはサーバーのドキュメントを参照してください。

IDE AWS Cloud9 にファイルをアップロードできない

問題: ユーザーは IDE AWS Cloud9 に大きなファイルをアップロードできません。アップロードは失敗しています。

原因: AWS Cloud9 IDE AWS Cloud9 へのアップロード速度をスロットリングし、その結果、ファイルのアップロードリクエストがタイムアウトします。

推奨される解決策: ファイルを Amazon S3 にアップロードし、Amazon S3 を使用して IDE の CLI AWS Cloud9 を使用して環境にファイルをダウンロードすることをお勧めします。Amazon S3 にファイルをアップロードする方法の詳細については、「Amazon S3 ユーザーガイド」の「[オブジェクトのアップロード](#)」を参照してください。

IDE AWS Cloud9 のダウンロード速度が遅い

問題: IDE AWS Cloud9 からファイルをダウンロードしようとすると、ダウンロード速度が低下します。

原因: IDE からローカルファイルシステムにファイルをダウンロードする場合、転送速度は 0.1 メガバイト/秒に制限されます。

推奨される解決策: ファイル転送の速度を上げるには、IDE の CLI AWS Cloud9 を使用して Amazon S3 にファイルをアップロードし、Amazon S3 を使用してそこからファイルをダウンロードします。

サイトへの接続が安全でないため、IDE で ウェブコンテンツをプレビューできません

問題: AWS Cloud9 EC2 環境でホストされている WordPress サイトなどのウェブコンテンツにアクセスしようとすると、IDE プレビューウィンドウには表示されません。

考えられる原因: デフォルトでは、IDE のアプリケーションプレビュータブでアクセスするすべてのウェブページは自動的に HTTPS AWS Cloud9 プロトコルを使用します。ページの URI が安全でない http プロトコルを備えている場合、https に自動的に置き換えられます。また、手動で https を http に戻す変更を行っても、安全でないコンテンツにアクセスすることはできません。

推奨される解決策: IDE でプレビューしようとしている ウェブサイトから安全ではない HTTP スクリプトまたはコンテンツを削除します。HTTPS の実装に関するガイダンスについては、ウェブサーバーまたはコンテンツ管理システムの指示に従ってください。

[\(先頭に戻ります\)](#)

サードパーティのアプリケーションとサービス

次のセクションでは、サードパーティのアプリケーションとサービスに関連する問題のトラブルシューティングについて概説します。

tmux セッションエラー AWS Cloud9 のため、でターミナルウィンドウとやり取りできない

問題: で新しいターミナルウィンドウを起動しようとする AWS Cloud9、予想されるコマンドラインインターフェイスが使用できなくなります。コマンドプロンプトがないため、テキストを入力できません。tmux: need UTF-8 locale (LC_CTYPE) や invalid LC_ALL, LC_CTYPE or LANG などのエラーメッセージが返されます。

考えられる原因: tmux エラーが原因でターミナルが応答しない可能性があります。は [tmux](#) ユーティリティ AWS Cloud9 を使用します。これにより、ページがリロードされたり、開発環境に再接続しても、ターミナルに表示される情報は保持されます。

tmux セッションでは、ターミナルウィンドウの表示内容はクライアントによって処理されます。クライアントは、複数のセッションを管理できるサーバーと通信します。サーバーとクライアントは、tmp フォルダにあるソケットを介して通信します。開発環境に tmp フォルダがないか、過度に制限的なアクセス許可が適用されている場合、tmux セッションは実行できません。この場合、IDE のターミナルウィンドウは応答しなくなります。

推奨される解決策: tmux エラーによってターミナルウィンドウと対話できない場合は、別の方法を使用して適切な許可を持つ tmp フォルダを作成する必要があります。これにより、tmux セッションを実行できます。1 つの解決策は、.bash_profile または .bashrc ファイルの LC_CTYPE をエクスポートすることです。もう 1 つの推奨される解決策は、AWS Systems Manager を使用してホスト管理設定をセットアップすることです。これにより、Amazon EC2 コンソールから該当するインスタンスにアクセスできるようになります。

ホスト管理の設定

1. まず、AWS Cloud9 コンソールで、環境のインスタンスの名前を見つけます。これを行うには、[Your environments] (自分の環境) ページで該当するパネルを選択し、[View details] (詳細を表示) を選択します。[環境の詳細] ページで [インスタンスに移動] を選択します。Amazon EC2 コンソールで、アクセスする必要があるインスタンスの名前を確認します。
2. AWS Systems Manager コンソールに移動し、ナビゲーションペインでクイックセットアップを選択します。
3. [クイックセットアップ] ページで [作成] を選択します。
4. [設定タイプ] では、[ホスト管理] に移動し、[登録] を選択します。
5. [ホスト管理構成オプションのカスタマイズ] の [ターゲット] セクションで、[手動] を選択します。

6. アクセスする EC2 インスタンスを選択してから、[作成] を選択します。

インスタンスに接続してコマンドを実行する

Note

次の手順は、新しい EC2 コンソール用です。

1. Amazon EC2 コンソールのナビゲーションペインで、[インスタンス] を選択し、接続するインスタンスを選択します。
2. 接続 を選択します。

[Connect] (接続) がアクティブ化されていない場合は、最初にインスタンスを起動する必要があります。

3. [Connect to your instance] (インスタンスへ接続) ペインの [Connection method] (接続方法) で、[Session Manager]、[Connect] (接続) の順に選択します。
4. 表示されたターミナルセッションウィンドウで、次のコマンドを入力します。これらのコマンドは、tmux ソケットを使用するための適切なアクセス許可を持つ tmp フォルダを作成します。

```
sudo mkdir /tmp
sudo chmod 777 /tmp
sudo rmdir /tmp/tmux-*
```

古いバージョンの Microsoft Edge ブラウザを使用して IDE をロードすることはできません

問題: Microsoft Edge ウェブブラウザを使用して IDE AWS Cloud9 をロードしようとすると、HTTP403: FORBIDDEN エラーが返されます。

考えられる原因: AWS Cloud9 IDE は、特定の古いバージョンの をサポートしていません Microsoft Edge。

推奨される解決策: ブラウザを更新するには、Microsoft Edge ツールバーの省略記号 (...) ボタンをクリックします。メニューで、[Settings] (設定) を選択して [About Microsoft Edge] (について) を選択します。アップデートが必要な場合は、自動的にダウンロードされ、インストールされます。

C++ プロジェクトのデバッグ時に、gdb を伴うエラー

問題: IDE で C++ プロジェクトをデバッグしようとする、gdb デバッガに報告されるエラー

考えられる原因: AWS Cloud9 環境が特定の EC2 インスタンスタイプ (t3.small や など) を使用しているとします m5.large。この場合、IDE の組み込みランナーを使用して C++ プロジェクトを実行およびデバッグしようとする、デバッグエラーが発生することがあります。このエラーは、環境にプリインストールされた gdb (GNU プロジェクトデバッガ) は、特定のプロセッサプラットフォームでは動作しません。次のエラーコードが表示される可能性があります。

```
GDB server terminated with code 1
```

推奨される解決策: gdb が特定のプロセッサプラットフォームをサポートしていないという問題は、バージョン 3.0 以降で修正されました。古いバージョンのデバッガをアンインストールし、新しいバージョンの gdb にアップグレードします。

1. AWS Cloud9 ターミナルで次のコマンドを実行して、デバッガーの既存のバージョンを削除します。

```
sudo yum -y remove gdb
```

2. gdb のアーカイブを取得して展開し、次のコマンドを実行して、展開したファイルが含まれているディレクトリに移動します。

```
wget "http://ftp.gnu.org/gnu/gdb/gdb-8.3.tar.gz"
tar xzf gdb-8.3.tar.gz
cd gdb-8.3
```

3. 次のコマンドを実行してデバッガーを構築します。これを行うには、次のテキストをコピーして単一のブロックとして貼り付け、Return を押して make を実行します。

```
./configure --prefix=/usr \
            --with-system-readline \
            --with-python=/usr/bin/python3 &&
make
```

4. デバッガーをインストールします。

```
sudo make -C gdb install
```

5. 更新したバージョンのデバッガーがインストールされたことを確認します。

```
gdb --version
```

での PHP ランナーの問題 AWS Cloud9

問題: ユーザーは PHP CLI ランナーターミナルで出力を表示できません。

原因: CLI ランナーを PHP に設定し、デバッガーモードを有効にする必要があります。

推奨される解決策: CLI ランナーを PHP に設定し、デバッガーモードが有効になっていることを確認します。

Node.js に関連する GLIBC エラー

問題: ユーザーが Node.js を実行できず、GLIBC エラーが発生します。エラーメッセージの例を以下に示します。

```
node: /lib64/libm.so.6: version `GLIBC_2.27' not found (required by node)
node: /lib64/libc.so.6: version `GLIBC_2.28' not found (required by node)
```

原因: 使用しているインスタンスに関連する Node.js バージョンに問題がある可能性があります。

推奨ソリューション: Node.js を AWS Cloud9にインストールする方法について、「[ステップ 1: 必要なツールをインストールする](#)」セクションを参照してください。

でサポートされているブラウザ AWS Cloud9

次の表に、サポートされているブラウザを示します AWS Cloud9。

ブラウザ	バージョン
Google Chrome	最新の 3 つのバージョン
Mozilla Firefox	最新の 3 つのバージョン
Microsoft Edge	最新の 3 つのバージョン
MacOS 版 Apple Safari	最新の 2 つのバージョン

Warning

IDE で Mozilla Firefox AWS Cloud9 を優先ブラウザとして使用している場合は、AWS Cloud9 ウェブビューと AWS ツールキットがブラウザで正しく動作するのを防ぐサードパーティーの Cookie 設定があります。この問題の回避策として、下の画像に示すように、ブラウザ設定の「プライバシーとセキュリティ」セクションで [Cookies] をブロックしていないことを確認する必要があります。

General

Home

Search

Privacy & Security

More from Mozilla

Browser Privacy

Enhanced Tracking Protection

Trackers follow you around online to collect information about your browsing habits and interests. Firefox blocks many of these trackers and other malicious scripts. [Learn more](#)

[Manage Exceptions...](#)

☐ **Standard**
Balanced for protection and performance. Pages will load normally.

☐ **Strict**
Stronger protection, but may cause some sites or content to break.

☒ **Custom**
Choose which trackers and scripts to block.

☐ Cookies

☒ Tracking content [Only in Private Windows](#)

☒ Cryptominers

☒ Fingerprints

① You will need to reload your tabs to apply these [Reload All Tabs](#)

Extensions & Themes

のサービスクォータ AWS Cloud9

次の表に、AWS Cloud9 および関連 AWS サービスのクォータを示します。

- [AWS Cloud9 サービスクォータ](#)
- [関連する AWS サービスクォータ](#)

AWS Cloud9 クォータ

次の表は、AWS アカウントの AWS Cloud9 のデフォルトのクォータを示しています。特に明記されていない限り、制限はリージョンごとに存在します。マネジメントコンソールまたは CLI AWS を使用して引き上げを AWS リクエストできます。クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。

引き上げはすぐには許可されないので、引き上げが有効になるまでに 2~3 日かかる場合があります。

リソース	デフォルトの制限	調整可能
AWS Cloud9 EC2 開発環境の最大数	- ユーザーごとに 100 - アカウントあたり 200	はい
SSH 環境の最大数	- ユーザーごとに 100 - アカウントあたり 200	はい
環境内のメンバーの最大数	メンバーのデフォルト最大数は、その環境のインスタンスのメモリを 60 MB で割ったものと等しくなります。たとえば、メモリが 1 GiB のインスタンスのメンバー数は、最大 17 人です (1 GiB を 60 MB で割って切り下げる)。 がインスタンスのメモリを判断 AWS Cloud9 できない場	いいえ ¹

リソース	デフォルトの制限	調整可能
	合、そのインスタンスに関連付けられた環境ごとにデフォルトで最大 8 ユーザーになります。 環境のメンバーの絶対最大数は 25 です。	
編集可能な最大ファイルサイズ	8 MB	いいえ

¹ デフォルト最大メンバー数を増やすように[環境を移行](#)することができます。ただし、環境のメンバーの絶対最大数は変わらず 25 です。

AWS Cloud9 IDE ダウンロードクォータ

AWS Cloud9 IDE からローカルファイルシステムにファイルをダウンロードする場合、転送速度は 0.1 メガバイト/秒に制限されます。ファイルの転送速度を上げるには、IDE AWS Cloud9 の CLI を使用して Amazon S3 にファイルをアップロードし、Amazon S3 を使用してそこからファイルをダウンロードします。

関連する AWS サービスクォータ

Amazon Elastic Block Store (Amazon EBS) ボリュームの最大数	5,000 詳細については、Amazon Web Services 全般のリファレンスの Amazon Elastic Block Store エンドポイントとクォータ を参照してください。
CloudFormation スタックの最大数	200 詳細については、「AWS CloudFormation ユーザーガイド」の「 CloudFormation クォータを理解する 」を参照してください。

Amazon EC2 のクォータ

Amazon Web Services 全般のリファレンス の [Amazon EC2 エンドポイントとクォータ](#) を参照してください。

AWS Cloud9 ユーザーガイドのドキュメント履歴

このトピックでは、AWS Cloud9 ユーザーガイドに対する重要な変更点の一覧を示します。このドキュメントの更新に関する通知については、[RSS フィード](#)にサブスクライブできます。

最新の更新

次の表は、2019 年 3 月以降のAWS Cloud9 ユーザーガイドに対する重要な変更点の一覧です。

変更	説明	日付
AWS Cloud9 は新規顧客には利用できなくなりました	AWS Cloud9 は新規顧客には利用できなくなりました。 AWS Cloud9 の既存のお客様は、通常どおりサービスを引き続き使用できます。 詳細はこちら	2024 年 7 月 25 日
Amazon Linux 2023 のサポート AWS Cloud9 が追加されました。	AWS Cloud9 で Amazon Linux 2023 がサポートされるようになりました。	2023 年 12 月 15 日
Node.js チュートリアルを更新	Amazon Linux 2 と Node.js 18 のサポートに関して Node.js チュートリアルを更新しました。	2023 年 10 月 23 日
Amazon VPC ダッシュボードを使用した Amazon VPC の作成に関するセクションを更新	Amazon VPC ダッシュボードを使用した Amazon VPC の作成に関するセクションを更新しました。	2023 年 7 月 27 日
Amazon EventBridge スキーマの使用に関するセクション	AWS Toolkit for を使用した Amazon EventBridge スキーマの操作に関するセクションを追加しました AWS Cloud9。	2022 年 12 月 15 日

CodeCatalyst セクションを追加	新しい Amazon CodeCatalyst サービスに関するセクションを追加しました。	2022 年 12 月 2 日
AWS IoT 追加されたコンテンツ	の使用に関するセクション AWS IoT を追加しました。	2022 年 11 月 1 日
IDE 用 Amazon ECS AWS Cloud9 サービスの概要	IDE でアクセスできる Amazon ECS AWS Cloud9 サービスの特徴と機能の概要とチュートリアルを追加しました。	2022 年 10 月 20 日
AWS Cloud9 統合開発環境 (IDE) AWS CDK での の操作	AWS Cloud9 統合開発環境 (IDE) AWS CDK での の操作に関するセクションを追加しました。	2022 年 10 月 5 日
Amazon ECR のコンテンツを追加しました	AWS Amazon ECR の使用に関するセクションを追加しました。	2022 年 10 月 4 日
コンプライアンス検証	AWS Cloud9 対象範囲内のコンプライアンスプログラムのリストを更新しました。	2022 年 3 月 4 日
Java の拡張サポート	Java 使用時の開発エクスペリエンスを向上させるための追加の言語サポート。生産性に関する主な機能には、コード補完、エラーの linting、コンテキスト固有のアクション、ブレークポイントやステップイングなどのデバッグオプションが含まれます。	2022 年 1 月 18 日

AWSServiceRoleForAWS Cloud9 の更新	License Manager を使用する EC2 インスタンスをサポートするように、サービスにリンクされたロールが更新されました。	2022 年 1 月 12 日
Step Functions ドキュメントのサポート	Step Functions を使用してステートマシンを作成、編集、実行する方法について説明するコンテンツが追加されました。	2021 年 12 月 20 日
AWS Systems Manager ドキュメントのサポート	Systems Manager オートメーションドキュメントについて説明するコンテンツが追加されました。	2021 年 12 月 20 日
Amazon Elastic Container Service Exec のユーザーガイドを作成しました	これは Amazon ECS Exec の概要です。	2021 年 12 月 13 日
IDE AWS IoT AWS Cloud9 サービスのユーザーガイドを作成	このユーザーガイドでは、AWS Cloud9 IDE AWS IoT のサービスの使用を開始する方法について説明します。	2021 年 11 月 22 日
AWS リソースのサポート	リソースおよび関連ドキュメントを表示するためのインターフェイスオプションと、リソースタイプにアクセスするためのサポートが追加されました。	2021 年 11 月 5 日
IDE 用 Amazon ECR AWS Cloud9 サービスの概要	IDE でアクセスできる Amazon ECR AWS Cloud9 サービスの特徴と機能の概要とチュートリアルを追加しました。	2021 年 10 月 14 日

[App Runner サポート](#)

AWS Toolkit AWS App Runner にのサポートを追加しました。 2021 年 9 月 30 日

[AWS Cloud9 アフリカ \(ケープタウン\) およびアジアパシフィック \(大阪\) リージョンでも利用可能](#)

AWS Cloud9 は、アフリカ (ケープタウン) とアジアパシフィック (大阪) のリージョンでも利用可能になりました。これらのリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。 2021 年 9 月 1 日

[AWS Toolkit の CloudWatch Logs と Amazon S3](#)

AWS Toolkit for に CloudWatch Logs のサポートを追加しました AWS Cloud9。Amazon S3 バケットへの現在のファイルのアップロードを許可する新しい特徴。 2021 年 7 月 16 日

[Amazon S3 の VPC エンドポイント](#)

Amazon S3 の VPC エンドポイントにサポートを追加して、依存関係のダウンロードを許可できるようになりました。 2021 年 4 月 22 日

[Git パネルから利用可能なビジュアルソースコントロール](#)

デベロッパーとして、Git パネルを使用して、ユーザーインターフェイスで Git コマンドを実行します。 2021 年 2 月 1 日

[プライベートサブネットへの環境インスタンスの起動](#)

Systems Manager 経由でアクセスする EC2 インスタンスのサポートが追加され、プライベートサブネットに起動されます。

2021 年 1 月 21 日

[AWS Toolkit の統合](#)

AWS Explorer ウィンドウから AWS Toolkit AWS のサービスを使用してナビゲートおよび操作できるようになりました。

2020 年 12 月 11 日

[CloudFormation および no-ingress EC2 環境](#)

CloudFormation テンプレートを使用して no-ingress EC2 環境の作成に関するドキュメンテーションが拡張されました。

2020 年 10 月 29 日

[Amazon Linux 2 ベースの EC2 環境](#)

コンソールで EC2 環境を作成するときに、EC2 インスタンスの Amazon Linux 2 AMI を選択できます。

2020 年 10 月 7 日

[Systems Manager を使った no-ingress EC2 インスタンス](#)

を使用してプライベート EC2 インスタンスにアクセスするためのサポートを追加しました AWS Systems Manager。

2020 年 8 月 12 日

[AWS Serverless アプリケーションのローカルデバッグの強化](#)

AWS Serverless Applications の新しいローカルデバッグ機能のサポートが追加されました。

2020 年 7 月 30 日

[AWS Cloud9 欧州 \(ミラノ\) リージョンでも利用可能](#)

AWS Cloud9 が欧州 (ミラノ) リージョンでも利用可能になりました。このリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。

2020 年 7 月 29 日

[Amazon EBS 暗号化](#)

AWS Cloud9 開発環境で使用する EC2 インスタンスの Amazon EBS ボリュームを暗号化する方法を説明するセクション。

2020 年 7 月 3 日

[にリージョンのサポートを追加 AWS Cloud9](#)

AWS Cloud9 は、米国西部 (北カリフォルニア)、アジアパシフィック (香港)、欧州 (パリ)、中東 (バーレーン)、南米 (サンパウロ) の各リージョンでも利用可能になりました。これらのリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。

2020 年 5 月 7 日

[セキュリティ](#)

AWS Cloud9 ユーザーガイドにセキュリティに関する章を追加しました。

2020 年 4 月 30 日

[\[Tags \(タグ\)\]](#)

タグを使用すると、AWS Cloud9 リソースへのアクセスを制御したり、請求情報を管理したりできます。

2020 年 1 月 22 日

[にリージョンのサポートを追加 AWS Cloud9](#)

AWS Cloud9 は、アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シドニー)、カナダ (中部)、欧州 (ロンドン)、欧州 (ストックホルム) の各リージョンでも利用可能になりました。これらのリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#) の「」を参照してください Amazon Web Services 全般のリファレンス。

2019 年 12 月 18 日

[更新済み: トラブルシューティング、環境を開くことができない](#)

IDE を開くためにサードパーティーの Cookie は不要になりました。

2019 年 11 月 6 日

[追加: トラブルシューティング、サードパーティーの Cookie が無効になる](#)

IDE を開くのにサードパーティーの Cookie は不要になりました。ただし、アプリケーションプレビュー機能またはファイルプレビュー機能には必要です。これに関する情報は、「トラブルシューティング」トピックで参照できます。

2019 年 11 月 6 日

[ドキュメントの構成](#)

特に初めてのユーザーのために、ナビゲーションしやすいよう、ユーザーガイドの構成が変更されました。

2019 年 8 月 15 日

[AWS Cloud9 欧州 \(フランクフルト\) リージョンでも利用可能](#)

AWS Cloud9 が欧州 (フランクフルト) リージョンでも利用可能になりました。このリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。

2019 年 5 月 15 日

[LAMP サンプルを追加](#)

LAMP (Linux、Apache HTTP Server、MySQL、PHP) AWS Cloud9 で を使用する方法を示す新しいサンプルを追加しました。詳細については、「[AWS Cloud9の LAMP サンプル](#)」を参照してください。

2019 年 5 月 10 日

[WordPress サンプルを追加](#)

WordPress AWS Cloud9 で を使用する方法を示す新しいサンプルを追加しました。詳細については、「[AWS Cloud9の WordPress サンプル](#)」を参照してください。

2019 年 4 月 19 日

[AWS Cloud9 アジアパシフィック \(東京\) リージョンでも利用可能](#)

AWS Cloud9 がアジアパシフィック (東京) リージョンでも利用可能になりました。このリージョンと他の AWS リージョンに関連付けられたサービスエンドポイントとサービスクォータの詳細については、[AWS Cloud9](#)の「」を参照してくださいAmazon Web Services 全般のリファレンス。

2019 年 4 月 4 日

[EC2 環境での Ubuntu Server のサポートに関する情報の追加](#)

AWS Cloud9 コンソールを使用して Ubuntu Server に接続する AWS Cloud9 EC2 開発環境を作成する手順が追加されました。詳細については、[EC2 環境を作成する](#) を参照してください。

2019 年 4 月 2 日

現在、コードを使用して Ubuntu Server に接続する AWS Cloud9 EC2 開発環境を作成することはできません。たとえば、AWS CLI、SDK CloudFormation、Tools for Windows PowerShell、または AWS Cloud9 API を使用します。AWS SDKs これらの方法のサポートは、将来予定されています。

以前の更新

次の表に、2019 年 4 月以前の AWS Cloud9 ユーザーガイドの重要な変更点を示します。

変更	説明	変更日
学生、教員、および企業に関する使用開始手順の追加	の使用を開始する手順が拡張され、学生、教育者、企業向けの手順が追加され AWS Cloud9 ました。詳細については、「 セットアップ AWS Cloud9 」を参照してください。	2019 年 2 月 7 日
AWS CloudTrail サポートが追加されました	AWS CloudTrail が をサポートするようになりました AWS Cloud9。詳細については、「 を使用した AWS Cloud9 API コールのログ記録 AWS CloudTrail 」を参照してください。	2019 年 1 月 21 日
共有 VPC サポートの追加	AWS Cloud9 が Amazon VPCsの共有 VPC をサポートするようになりました。詳細については、「 の Amazon VPC 要件 AWS Cloud9 」を参照してください。	2018 年 12 月 7 日
AWS RoboMaker 統合が追加されました	AWS Cloud9 では、AWS RoboMaker がサポートされるようになりました。RoboMaker は、インテリジェントなロボットアプリケーションを大規模に簡単に開発、テスト、デプロイできるサービスです。詳細については、「 AWS RoboMaker デベロップャーガイド 」のRoboMaker の開始方法」	2018 年 11 月 26 日

変更	説明	変更日
	と「 での開発 AWS Cloud9 」を参照してください。	
言語プロジェクトに対する追加の生産性向上機能に関する情報の追加	AWS Cloud9 IDE では、言語プロジェクトのコンテキストで一部の言語の生産性向上機能が追加されました。詳細については、「 TypeScript のサポートと機能の強化 」を参照してください。	2018 年 10 月 2 日
Go (移動)] ウィンドウの追加、[Navigate (移動)] ウィンドウと [コマンド] ウィンドウの削除	Go ウィンドウは、2018 AWS Cloud9 年 10 月 2 日以降に作成された環境の IDE に追加されました。この新しいウィンドウは、[Navigate (移動)] ウィンドウと [Commands (コマンド)] ウィンドウ (両方とも 2018 年 10 月 2 日以降に作成された環境に対して、IDE から削除されました) に取って代わります。詳細については、「 IDE のツアー 」の ステップ 10: [Go (移動)] ウィンドウ を参照してください。	2018 年 10 月 2 日
AWS CDK サンプルを追加しました	AWS Cloud Development Kit () AWS Cloud9 で を使用する方法を示す新しいサンプルを追加しましたAWS CDK。詳細については、「 AWS CDK のチュートリアル AWS Cloud9 」を参照してください。	2018 年 8 月 30 日

変更	説明	変更日
EC2 環境に自動的に追加される SSH IP アドレスの制限に関する情報の追加	2018 AWS Cloud9 年 7 月 31 日以降に作成された ForEC2 開発環境では、受信 SSH トラフィックを AWS Cloud9 が SSH 経由で接続するために使用する IP アドレス範囲のみに自動的に制限する AWS Cloud9 ようになりました。詳細については、「 のインバウンド SSH IP アドレス範囲 AWS Cloud9 」を参照してください。	2018 年 7 月 31 日
Docker サンプルの追加	Docker AWS Cloud9 で を使用する方法を示す新しいサンプルを追加しました。詳細については、「 の Docker チュートリアル AWS Cloud9 」を参照してください。	2018 年 6 月 19 日
Java、.NET Core、および TypeScript に関するサンプルの追加	Java、.NET Core、TypeScript AWS Cloud9 で を使用する方法を示す新しいサンプルを追加しました。詳細については、「 の Java チュートリアル AWS Cloud9 」、「 の .NET チュートリアル AWS Cloud9 」、および「 の TypeScript チュートリアル AWS Cloud9 」を参照してください。	2018 年 5 月 29 日

変更	説明	変更日
サポートされているブラウザリストの追加	でサポートされているブラウザに関する情報を追加しました AWS Cloud9。詳細については、「 でサポートされているブラウザ AWS Cloud9 」を参照してください。	2018 年 5 月 23 日
SSH IP トラフィック制限情報の追加	が AWS Cloud9 SSH 経由でホストに接続するために使用する IP アドレス範囲のみに受信トラフィックを制限する方法に関する情報を追加しました。詳細については、「 のインバウンド SSH IP アドレス範囲 AWS Cloud9 」を参照してください。	2018 年 4 月 19 日
アプリケーションのプレビューと実行中のアプリケーションの共有に関するトラブルシューティングの追加	アプリケーションのプレビューと実行中のアプリケーションの共有に関する新しいトラブルシューティングを追加しました。詳細については、「 アプリケーションプレビュータブにエラーが表示される、または空白になる 」および「 IDE の外部で実行中のアプリケーションを表示できない 」を参照してください。	2018 年 4 月 19 日

変更	説明	変更日
[File Revision History (ファイルリビジョン履歴)] 情報の追加	IDE の [File Revision History (ファイルリビジョン履歴)] ペインの使用方法についての情報を追加しました。詳細については、「 IDE AWS Cloud9 でのファイルリビジョンの操作 」を参照してください。	2018 年 4 月 19 日
環境を開く場合のトラブルシューティングの追加	AWS Cloud9 開発環境を開くための新しいトラブルシューティングを追加しました。詳細については、「 環境を開くことができない 」を参照してください。	2018 年 3 月 19 日
AWS Cloud9 Installer に追加されたトラブルシューティング	AWS Cloud9 インストーラの新しいトラブルシューティングを追加しました。詳細については、「 AWS Cloud9 インストーラがハングまたは失敗する 」を参照してください。	2018 年 3 月 19 日
AWS CodePipeline 追加された情報	AWS Cloud9 で を使用する方法に関する情報を追加しました AWS CodePipeline。詳細については、「 IDE AWS CodePipeline での AWS Cloud9 の使用 」を参照してください。	2018 年 2 月 13 日

変更	説明	変更日
AWS CloudShell 追加された情報	AWS Cloud9 で を使用する方 法に関する情報を追加しまし た AWS CloudShell。詳細に ついては、「 AWS CLI および の aws-shell チュートリアル AWS Cloud9 」を参照してく ださい。	2018 年 1 月 19 日
GitHub のドキュメント入手可 能の追加	このガイドが GitHub で利用 可能になりました。GitHub を使用して、フィードバック の送信およびこのガイドの 内容に対する変更リクエスト の送信もできます。詳細につ いては、ガイドのナビゲー ションバーの [Edit on GitHub (GitHub で編集)] アイコンを 選択するか、GitHub ウェブサ イトで awsdocs/aws-cloud9- user-guide リポジトリを参照 してください。	2018 年 1 月 10 日
Kindle 形式の入手	このガイドが Amazon Kindle 形式でご利用いただけるよう になりました。詳細について は、ガイドのナビゲーシ ョンバーにある [Kindle を開く] アイコンを選択してください。	2018 年 1 月 2 日

変更	説明	変更日
Amazon Lightsail の情報を追加しました	AWS Cloud9 で を使用する方 法に関する情報を追加しまし たAmazon Lightsail。詳細につ いては、「 IDE AWS Cloud9 でのAmazon Lightsailインスタ ンスの操作 」を参照してくだ さい。	2017 年 19 月 12 日
の環境設定の説明を追加しま した AWS	AWS Cloud9 開発環境の特定 の AWS 設定の説明を追加し ました。詳細については、 「 AWS Cloud9 IDE での AWS プロジェクトおよびユーザー 設定の操作 」を参照してくだ さい。	2017 年 12 月 7 日
AWS アカウントのルートユー ザー用の開始手順とチーム用 の高度なセットアップ手順を 追加	AWS アカウントルートユー ザー AWS Cloud9 で を使用 するためのセットアップ手 順を追加しました。チーム AWS Cloud9 で を使用するた めの高度なセットアップ手順 を追加しました。詳細につい ては、「 セットアップ AWS Cloud9 」を参照してくださ い。	2017 年 5 月 12 日
環境要件に関する説明の拡大	Amazon EC2 インスタンスま たは独自のサーバーが SSH AWS Cloud9 開発環境に接続 するための要件のカバレッジ を拡大しました。詳細につい ては、「 SSH 環境ホスト要 件 」を参照してください。	2017 年 12 月 4 日

変更	説明	変更日
初回のドキュメントリリース	これは、『AWS Cloud9 ユーザーガイド』の初回リリースです。	2017 年 11 月 30 日