AWS App Mesh & AWS Cloud Map ワークショップ

2021/06/24

シニアエバンジェリスト

亀田 治伸

**[はじめに]**

本シナリオのオリジナルは以下になります。本シナリオは当該シナリオを日本語化しオンラインイベント用に手順を減らし初心者でも実行可能なように簡素化したものです。

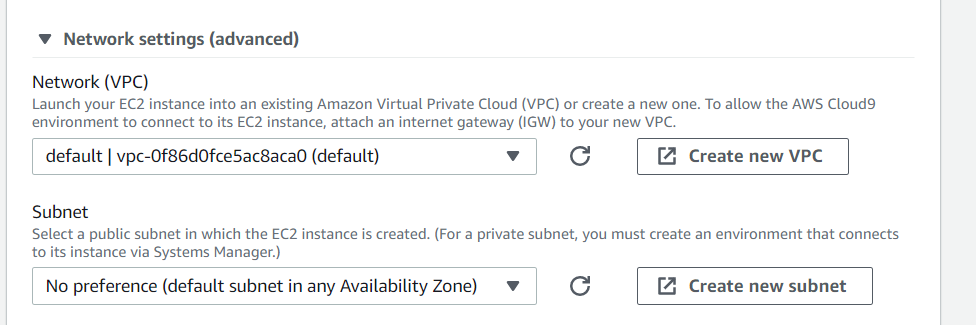
https://www.appmeshworkshop.com/

**必ず全ての作業はオレゴンリージョン**で行ってください。

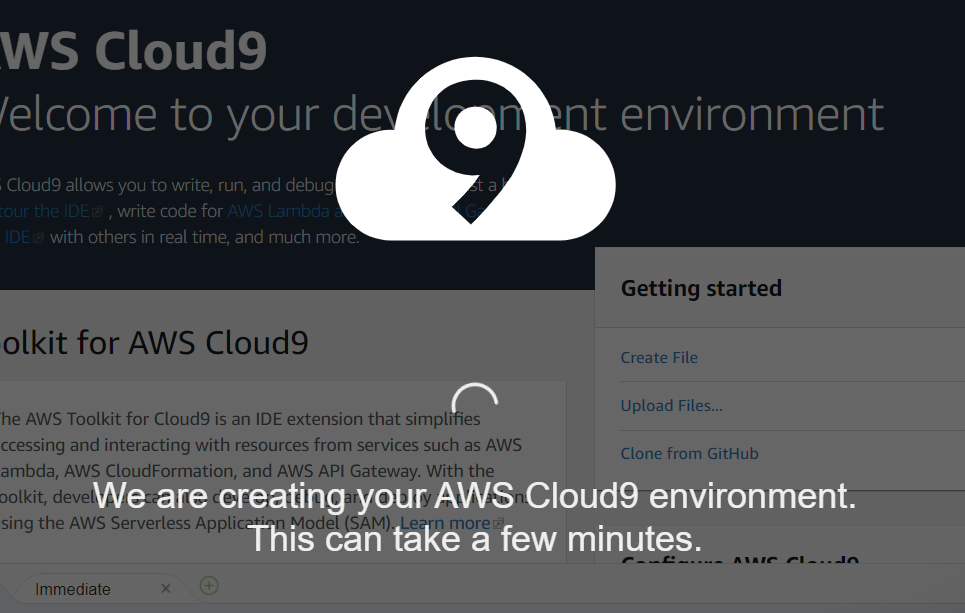
また、このワークショップでは大量のコマンドを実行しますが、全て[copycommands.txt]に入っています。以下の指示で「コマンド1番を実行します」などと記載されている場合、当該ファイルの中に入っているコマンドをそのままターミナルにコピーして実行することを意味します。

**[開発環境の構築]**

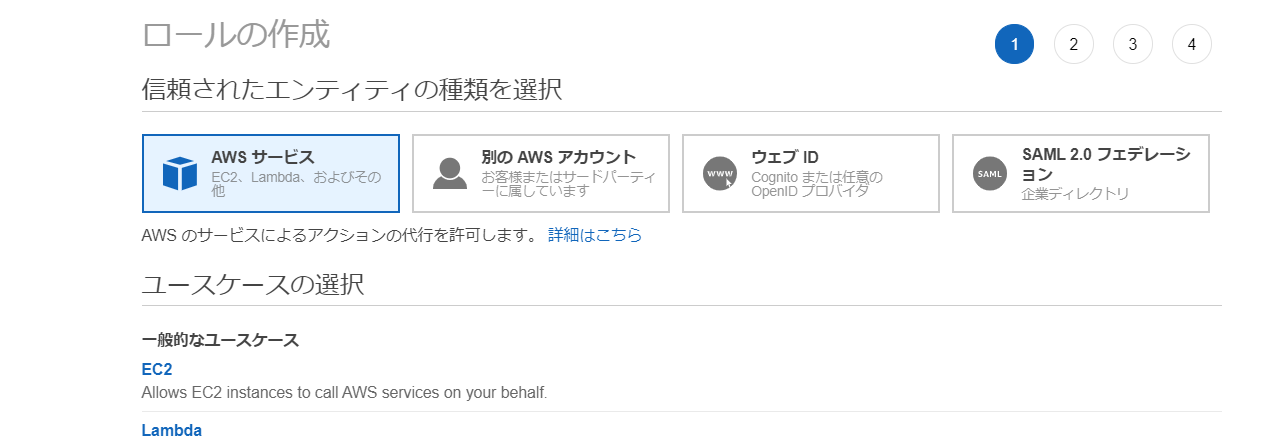
1. マネージメントコンソール、Cloud9の画面に遷移し、[Create environment]をおします
2. [Name]に適当な名前を付けて、[Next Step]をおします
3. [Instance Type]は[t3.small]を選んでください（デフォルトのt2.microではメモリが足りません。）
4. その他全てデフォルトのままで[Next step]をおします。デフォルトVPCがオレゴンリージョンに存在していない場合、起動したCloud9にブラウザでアクセスが行えないため、その場合、[Network settings]から、任意のVPC、任意のPublic Subnetを指定してください



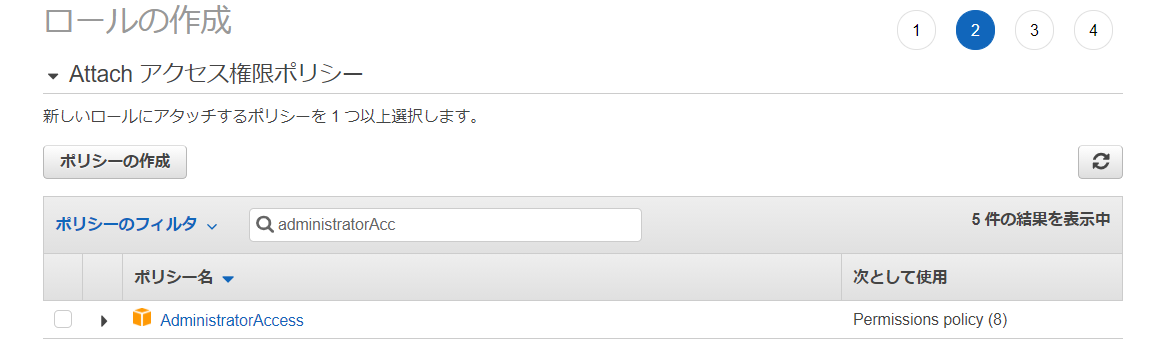
1. 最後の確認画面で[Create environment]をおし、数分間待つとターミナルにアクセスが可能となります



1. 待っている間にブラウザ別タブでAWSマネージメントコンソールを開きIAMの画面にいきます
2. 画面左ペインからロールをクリックし、[ロールの作成]をおします
3. 一般的なユースケースから[EC2]を選び、[次のステップ]をおします



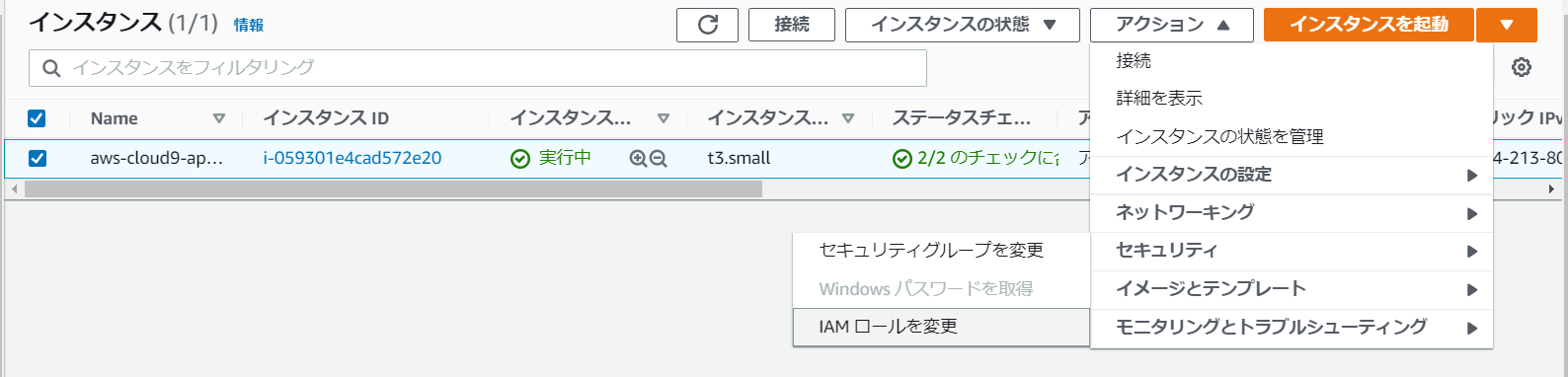
1. AdministratorAccessを指定し[次のステップ]をおします。次の画面はそのままさらに[次のステップ]をおします



1. 名前に、[AppMesh-Workshop-Admin]とつけ、[ロールの作成]をおします。

**注意：この名前を変更すると動作しません。後ほど利用するスクリプトにIAM名が固定で埋め込まれているためです。**

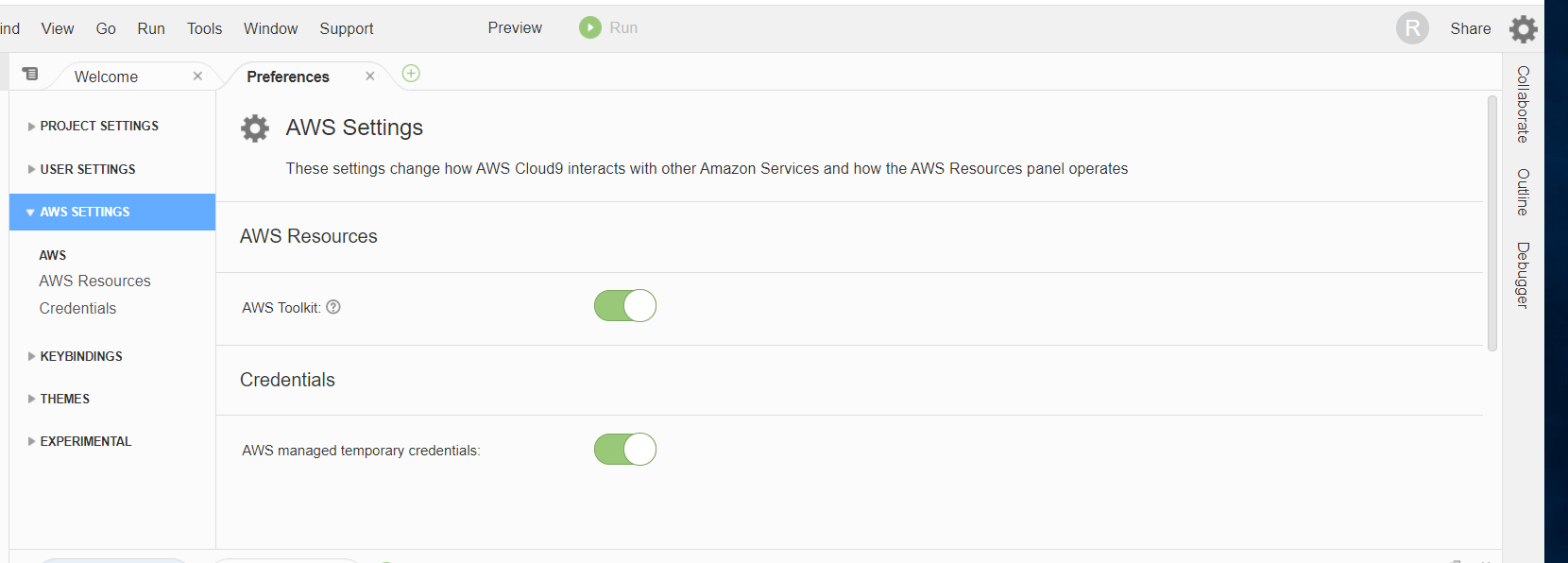
1. IAMロールが作成されたら、EC2の画面に遷移します
2. 画面左ペインからインスタンスをクリックし、Cloud9用EC2を特定します。（aws-cloud9から名前が始まっています）
3. [アクション]から[セキュリティ]→[IAMロールを変更]を選びます



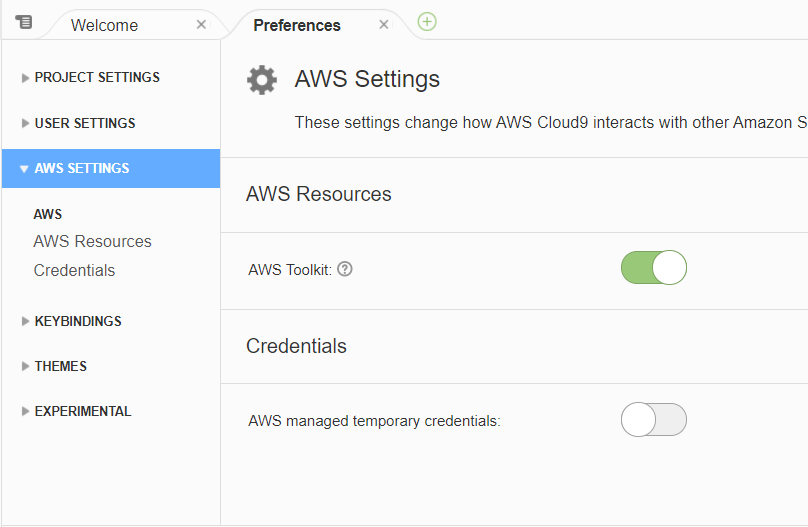
1. 先程作成したIAMロールを選び[保存]をおします



1. Cloud9の画面に戻るとターミナルが使えるようになっているはずです。
2. 画面右上の歯車マークをおし、[AWS SETTINGS]を選びます。



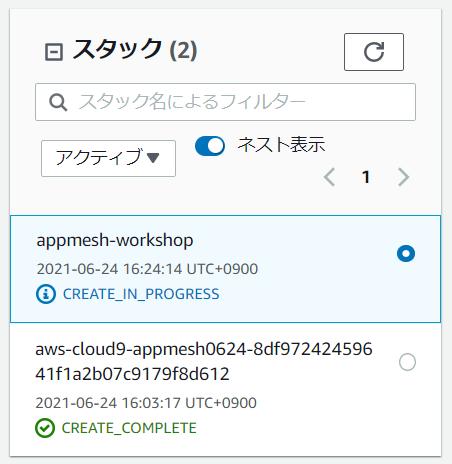
1. [Credentials]のチェックを外して、タブを閉じます



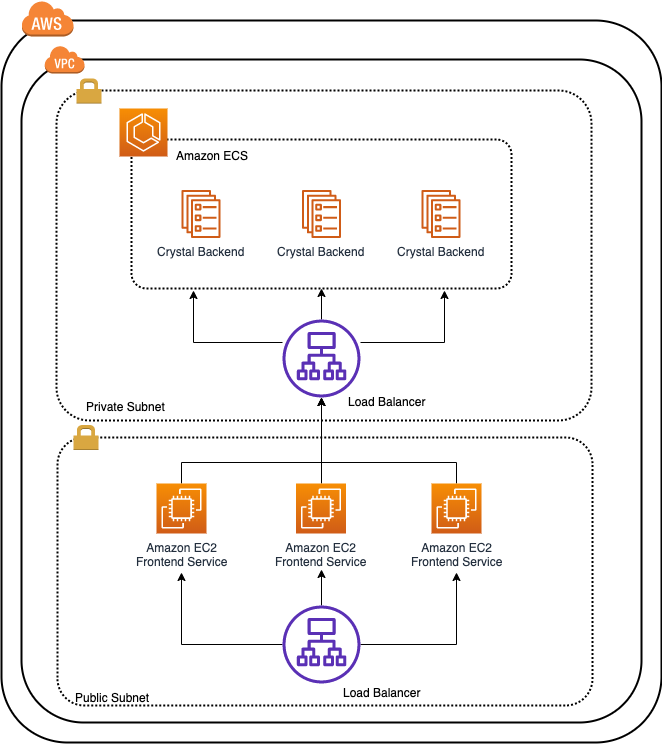
1. コマンド1,2,3を順番に実行します
2. kubectl, jq, gettext、およびAWS cliをインストールするため、コマンド4,5を順番に実行します
3. 環境構築に必要なファイルを、コマンド6を実行してダウンロードします

.c9/project.settings変更の確認画面が出た場合は[Accept]をクリックして下さい

1. CloudFormationのテンプレートを、コマンド7を実行してダウンロードします
2. 8を実行してCloudFormationスタックを起動します。CloudFormationのマネージメントコンソールで状況が確認可能です。Completeとなるまで待ちます。10分ほどで完了するはずです。イベントタブなどを見ながら何が作られているのかを確認してください



以下のような環境が構築されています。（実際はECS・EKSの混在環境となるのでより複雑です）初期構築される環境はApp Meshを使わずにELBがルーティングを行っています。この環境をApp Mesh及びCloud Mapをベースとしたネットワーク環境に作り替えていくことがこのワークショップの目的です。まだこの時点ではコンテナが起動していないため、アクセスができません。

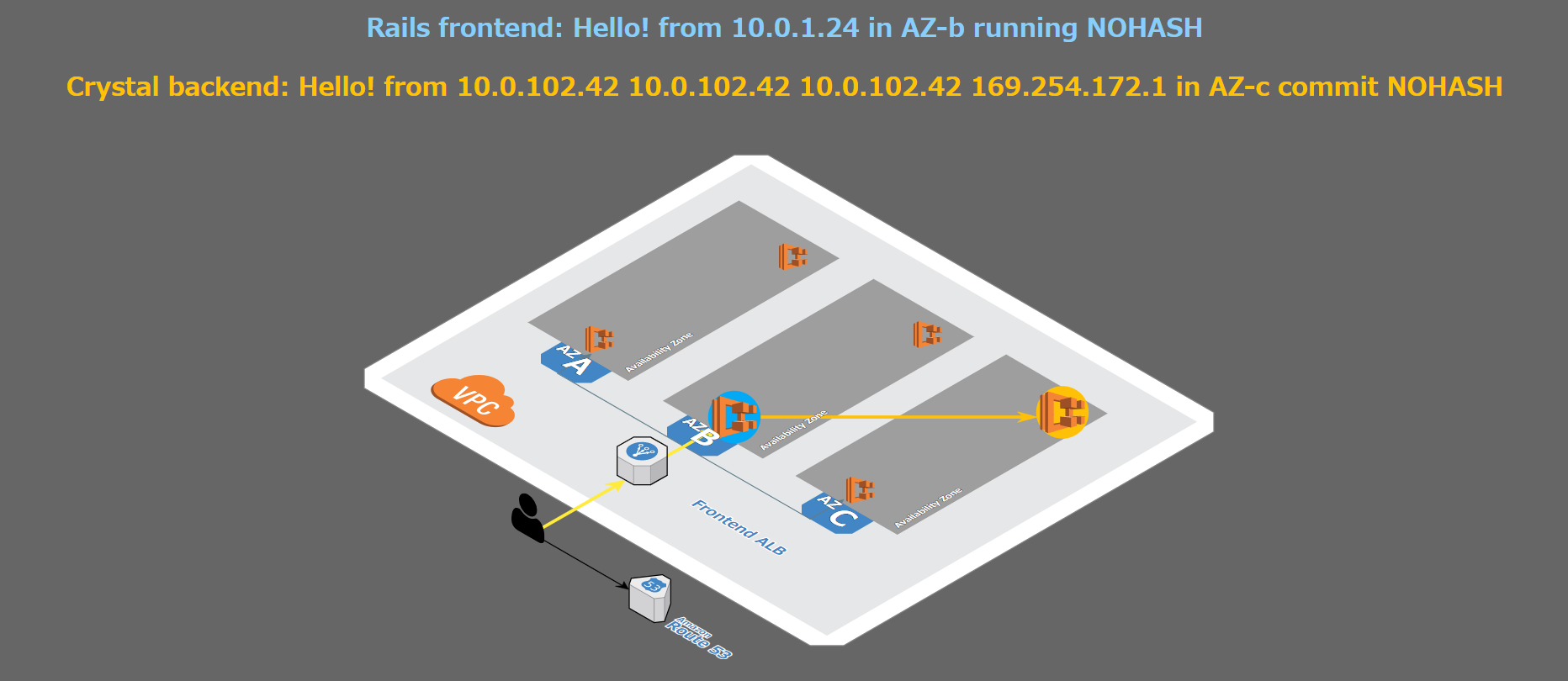


1. EC2にSSH ログインするための鍵情報を、コマンド9番を実行し入手します
2. コマンド10、11番を順番に実行し、Dockerイメージの作成、ECRレポジトリの作成とイメージのPush、ECSサービスの起動、EKSクラスターの起動をおこないます。

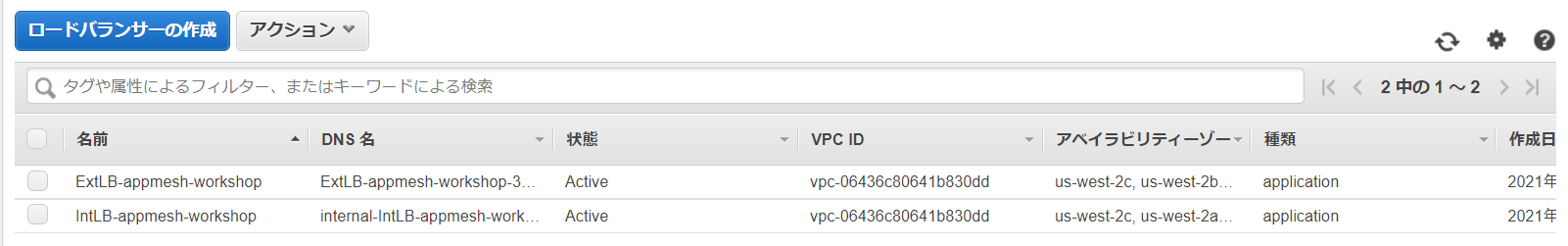
途中赤字で少しpythonのバージョンにかかわるアラートが上がりますが、動作に影響はないので安心してください。

さらに、CloudFormationスタックが起動しますので、マネージメントコンソールでステータスを確認しながらまちます。ステータスがCompleteになってもターミナルは戻ってきませんが、しばらく待っているともう一つスタックが起動されます。おおよそ25分程度で作業が完了します。

1. 完了したらコマンド12番を実行し表示されたURLにブラウザでアクセスしてみてください。Frontend, Backend両方とも動作しており、ラウンドロビンによりELBがルーティングしていることがわかります。



EC2のロードバランサーの画面を見ると2つのELB（外向けようと、フロントエンドとバックエンドの間用）ができていることがわかります。

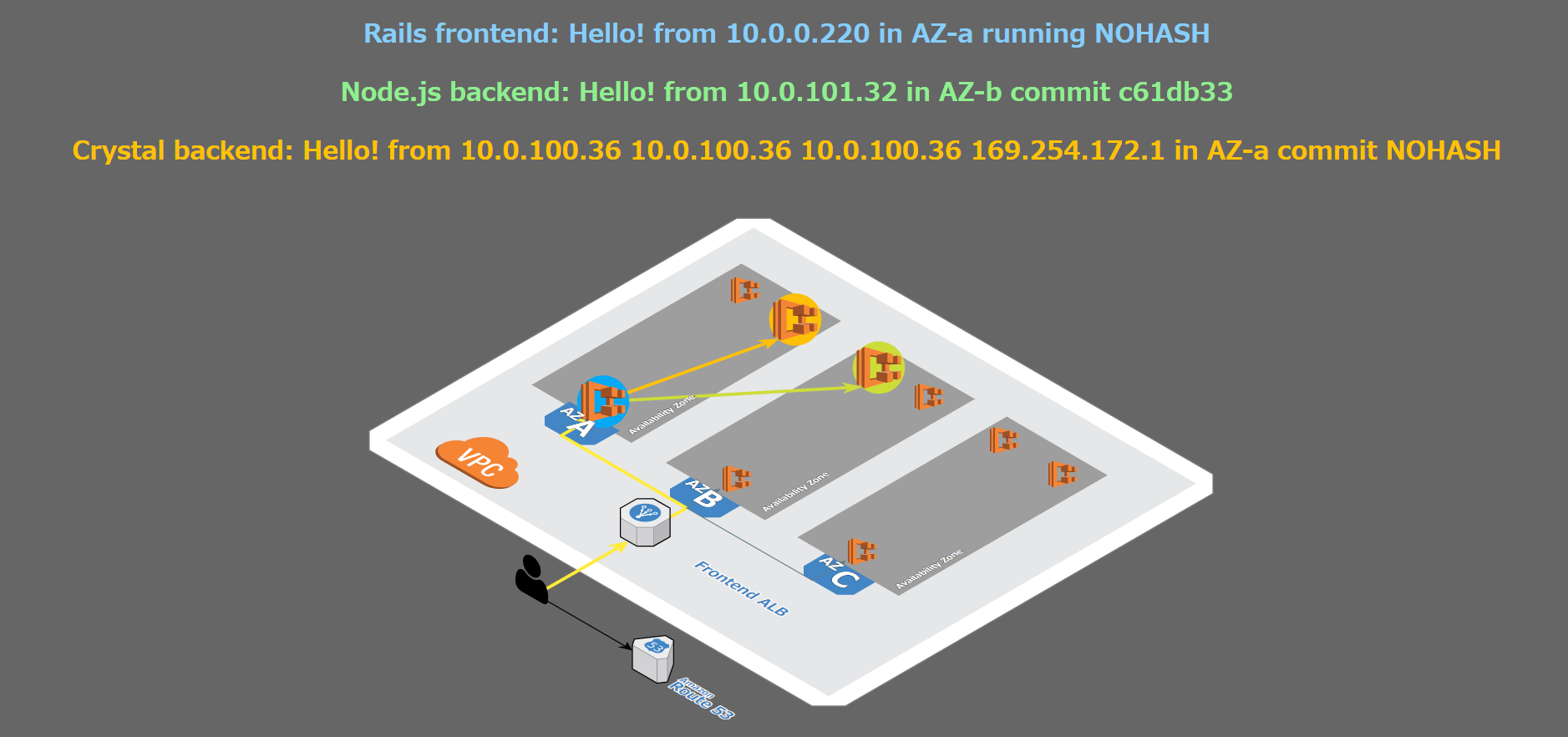


また、App Meshのマネージメントコンソールを確認するとまだ現時点で何もできていないことがわかります。

1. バックエンドのEKS環境にNode.jsサービスを起動していきます。コマンド13番を実行してください

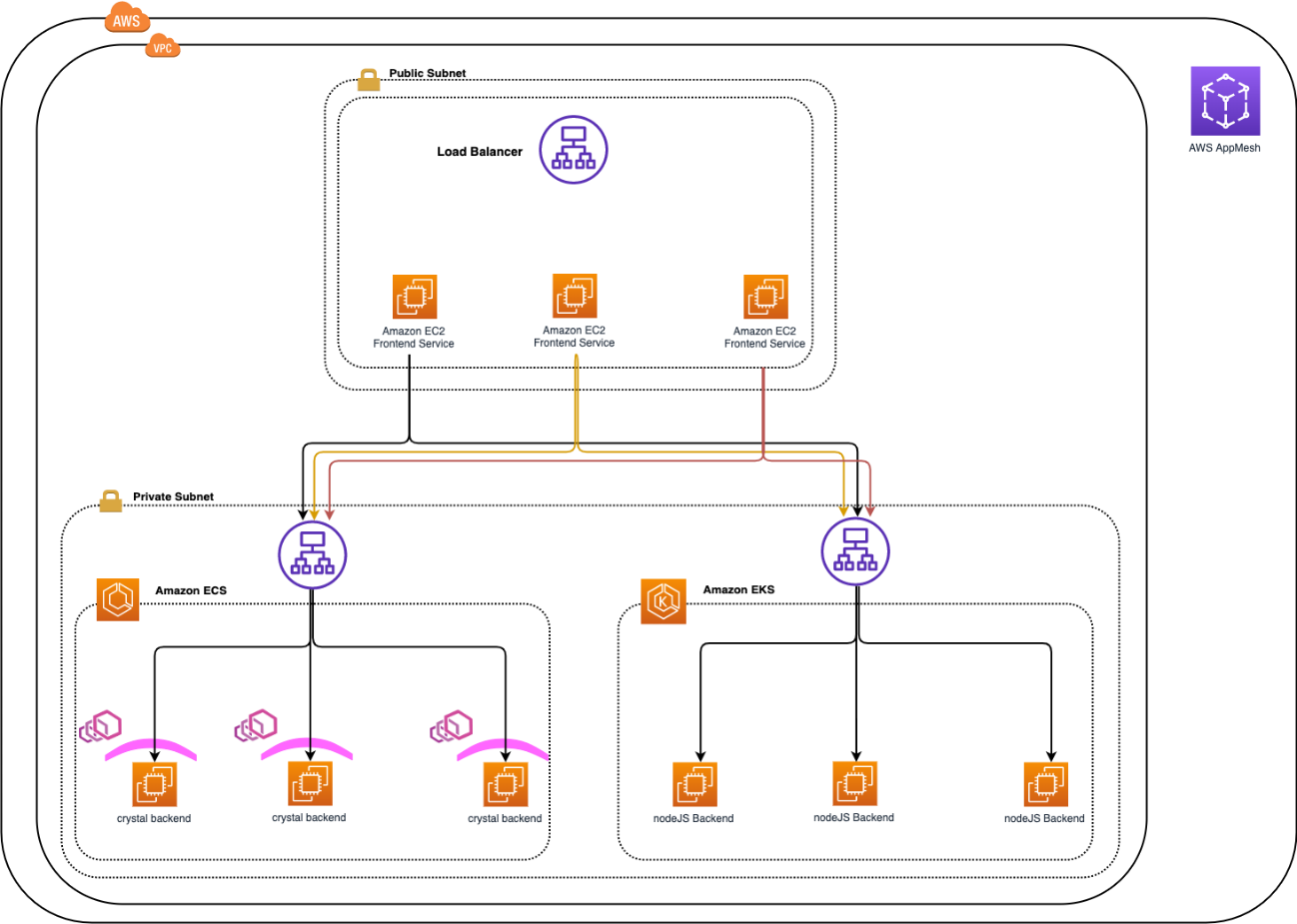
この手順でエラーとなる場合、IAMロールの名前が誤っています。最初からやり直してください。k8sの認証は独自です。Cloud9が起動時に引き継ぐIAMクレデンシャルは一時的なものであり時間とともに変更します。たとえ管理権限を保有していたとしても、起動時とその後の設定変更時の権限が時間とともに異なるため、エラーとなります。したがって前述のクレデンシャルの作業をしていないとエラーとなります。また、13番のスクリプトにはIAMロール名が埋め込まれています。名前だけの問題であればスクリプトの修正で作業は続行可能です。

1. 前述の手順で名前空間ができましたので、コマンド14、15番を順に実行し、サービスを起動し、Route53にゾーンを登録し名前解決を出来るようにします。
2. 再度ELBにアクセスするとnode.js環境が新たに加わっていることがわかります

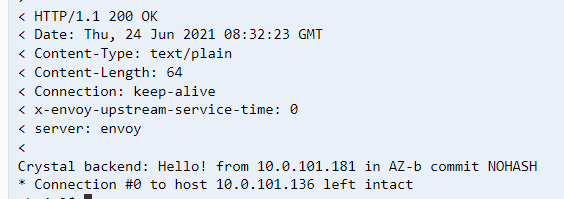


**[App Meshの実装]**

ここから本題のApp Meshの実装にはいります。以下の図のようにECSの環境にApp Meshを実装していきます。先程作成したNode.js環境やFrontエンド環境へのApp Meshの実装は時間の関係上割愛しますが、興味がある方はオリジナルの手順を見てください。



1. コマンド16番を実行し、メッシュ環境を作成します
2. ここから作成されたメッシュ環境に、仮想サービスと仮想ノードを作成していきます。仮想サービスは、メッシュ内の仮想ノードによって提供される実際のサービスを抽象化したものです。仮想ノードは、EC2 Auto Scaling Group、Amazon ECSサービス、Kubernetesデプロイメントなど、特定のタスクグループへの論理的ポインタとして機能します。コマンド17番を実行します
3. 仮想ノードができましたので、次に仮想サービスを作ります。コマンド18番を実行します
4. ここからEnvoyベースのside car proxy環境を作っていきます。App MeshはEnvoyをベースとしてネットワークの機能を提供しますが、その環境はside car proxy という、既存コンテナ環境（アプリ環境）の横で動作する独立したコンテナです。まず、side carを起動させるためにタスク定義を行います。コマンド19番を実行してください
5. サービスを更新しタスクを認識させます。コマンド20,21番順に実行してください。これにより、各コンテナへのアクセスがApp Mesh経由になります。
6. 確認のためにコマンド22番を実行しEC2へアクセスします。
7. コマンド23番を実行します。このコマンドの実行により、フロントエンドのEC2からバックエンドのApp Mesh配下のコンテナに通信を行います。以下のように戻ってきた値を見るとヘッダー部分に[envoy]とセットされており、App Mesh経由で通信されていることがわかります



1. [exit]とタイプしセッションを停止します

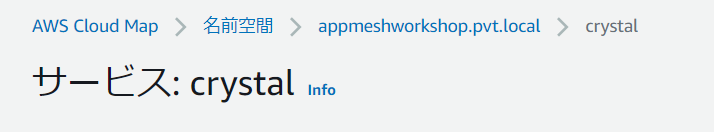
**[Cloud Mapの実行]**

今までの手順でApp Meshの実装が行えました。さらに、Node.js環境、フロントエンド環境への実装はオリジナルの手順書にありますので、是非挑戦してみてください。

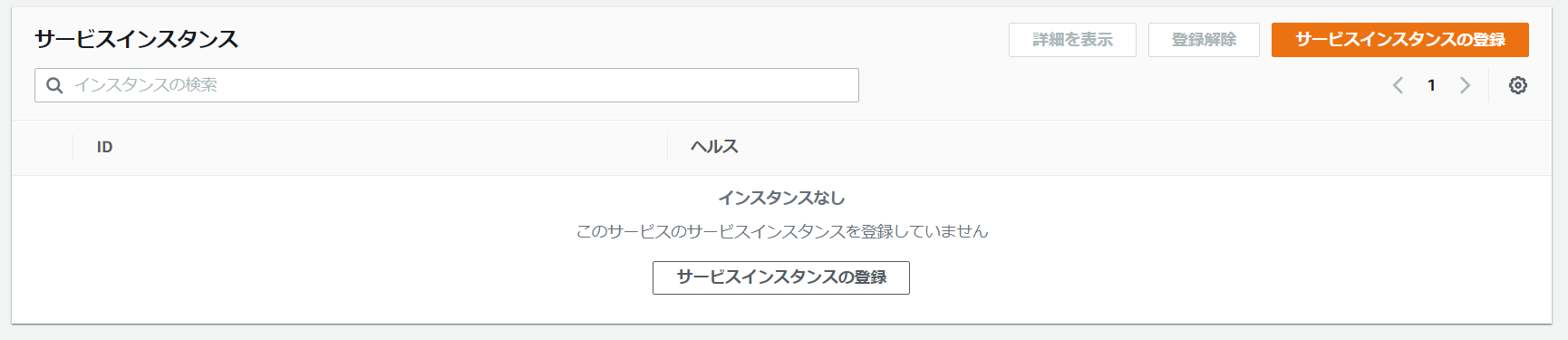
ここからの手順ではCloud Mapを用いて、フロントエンドからバックエンドへの通信に用いられているELBをApp Mesh + Cloud Map置き換えていきます。これにより、ELBが不要となります。作業開始前に、ELBが現在3つあることを確認しておいてください。

1. コマンド24番を実行しCloud Map用名前空間を作成します
2. 続いで作成された名前空間にサービスを作成します。コマンド25番を実行してください。

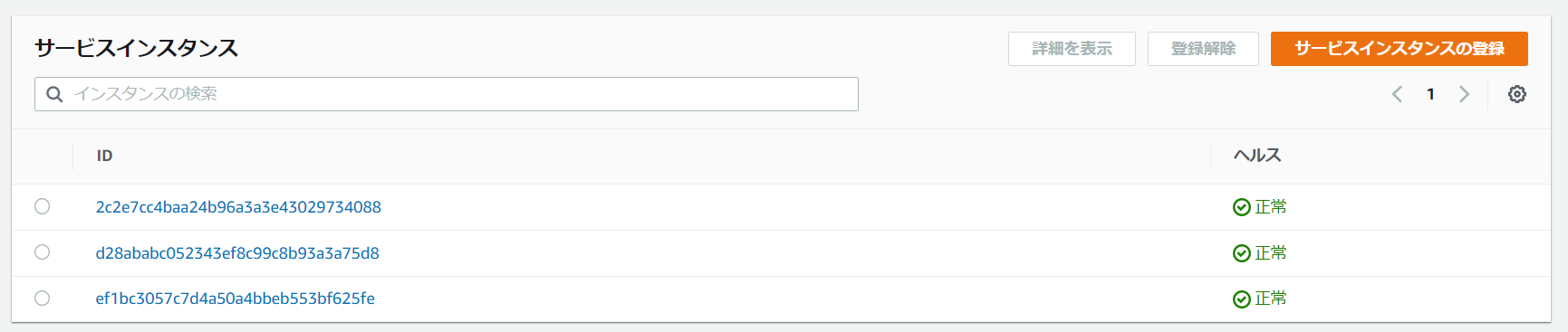
Cloud Mapのマネージメントコンソールにアクセスすると名前空間が一つできています。名前空間に紐つくサービスを見てみると、



まだ何もインスタンスが登録されていないことがわかります



1. 先程App Meshを実装した環境に対して、追加で仮想ノードを作成し、DNSの代わりにCloud Mapが名前解決を行うことを宣言します。コマンド26番を実行します。
2. コマンド27,28番を順番に実行し、新たな仮想ルーターを作成し、リクエストの向け先をこちらの新しい仮想ノードに行う準備をします。（この時点ではまだリクエストの向け先は新しい環境とならず、後ほど切り替えのコマンドを実行します。その前準備の設定を行っています）
3. コマンド29番を実行し、サービスをこの新しい仮想ルーターを使うように設定変更します
4. この仮想ノードと連携させるための新しいECSタスク定義、サービスを作成し、インスタンスをこのサービスレジストリを使うように設定します。コマンド30,31,32番を順番に実行します
5. コマンド33番を実行し、設定が反映されるのをまちます。インスタンスが自動で登録されているのがわかります。以下のようにCloud Mapのマネージメントコンソールで正しくインスタンスが登録されているのがわかります。



1. 手順40番で設定した内容を、コマンド34番を実行し書き換えます。これにより仮想ルーターの内容が書き換わり新しい仮想ノードにトラフィックの一部が向きます（ELBと新環境でA/Bテストを行っているイメージです）
2. ブラウザで先程のELBのURLにアクセスして正しく表示されることを確認してください
3. いよいよ、100％Cloud Map環境に切り替えるため、仮想ルーターの設定を書き換えるため、コマンド35番を実行します
4. コマンド36,37番を順番に実行し、旧ECSサービス環境とELBを削除したのち、DNS情報を書き換えます
5. ブラウザから引き続き同じようにアクセスできることを確認してください
6. EC2のロードバランサー画面で、手順書25番にあった内向け用ロードバランサーが無いのに動作していることがわかります。

おつかれさまでした！

是非フルの手順にも挑戦してみて下さい。

削除方法は以下です。

Cloud Map

サービスインスタンス

サービス

名前空間

R53

CNAMEレコード

Aレコード

ホストゾーン

ECS

サービス

タスク定義

ECR

レポジトリ

EC2

ロードバランサー

CFn　（上4つは1個づつ実行。同時実行しないこと）

eksctl-appmesh-workshop-addon-iamserviceaccount-appmesh-system-appmesh-controller

eksctl-appmesh-workshop-nodegroup-appmesh-workshop-ng

eksctl-appmesh-workshop-cluster

appmesh-workshop

aws-cloud9-xxxx

IAMロール

App Mesh

仮想サービス

ルート（仮想ルーターの下）

仮想ルーター

仮想ノード

メッシュ