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

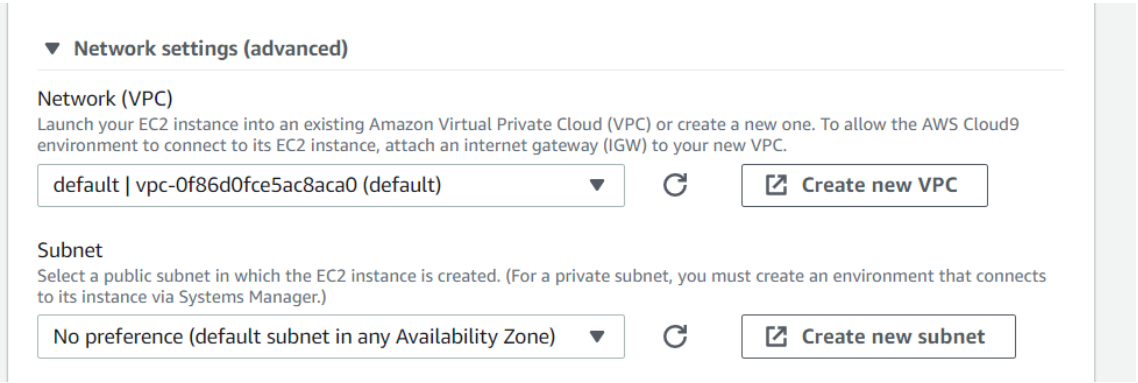
[はじめに]

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じっここう可能なように簡素化したものになります。
https://www.appmeshworkshop.com/

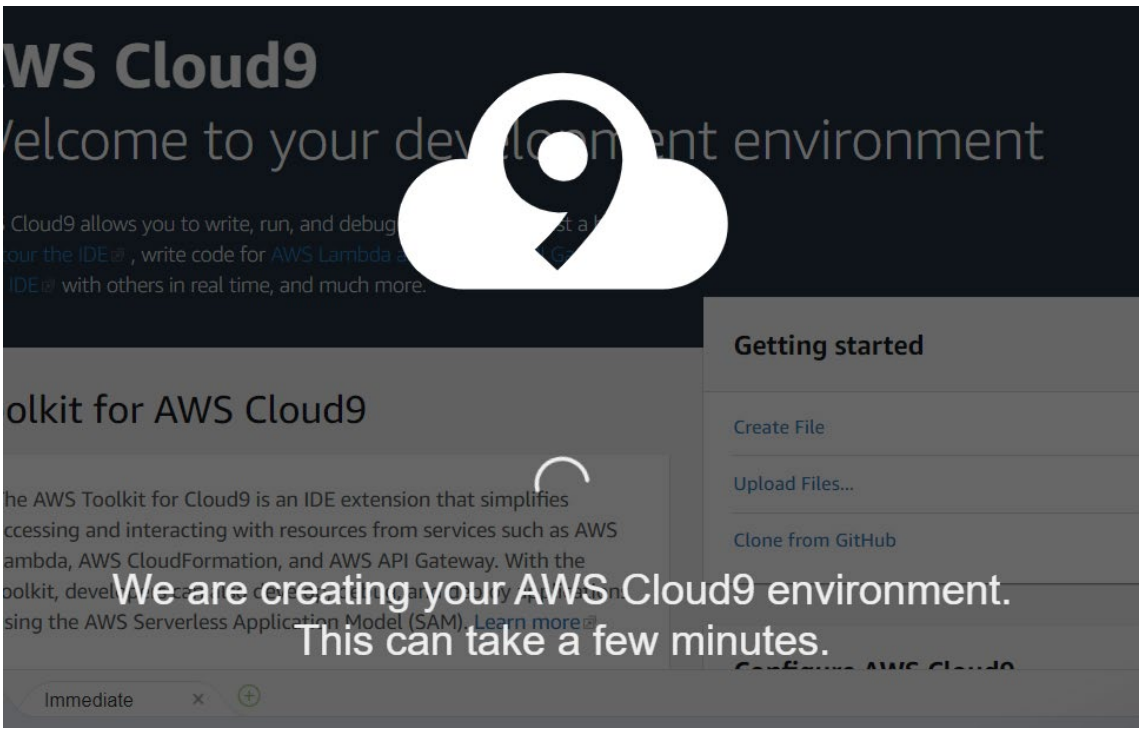
必ず全ての作業はオレゴンリージョンで行ってください。
また、このワークショップでは大量のコマンドを実行しますが、全て[copycommands.txt]入っています。以下の指示で「コマンド 1 番を実行します」などと記載されている場合、当該ファイルの中に入っているコマンドをそのままターミナルにコピーして実行することを意味します。

[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



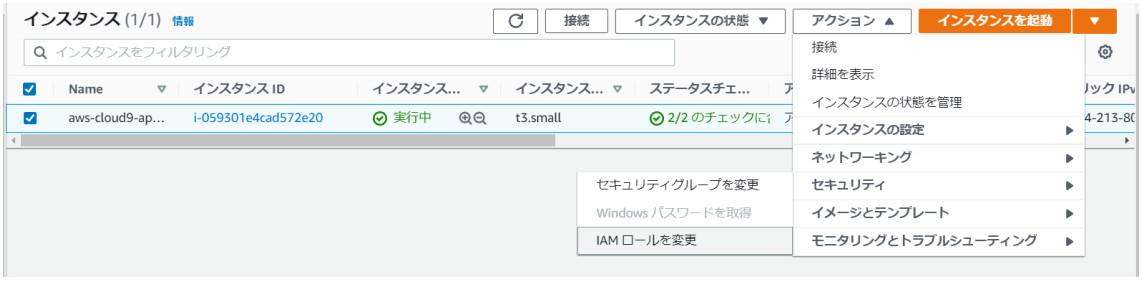
5. 最後の確認画面で[Create environment]をおし、数分間待つとターミナルにアクセスが可能となります



6. 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
7. 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
8. 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

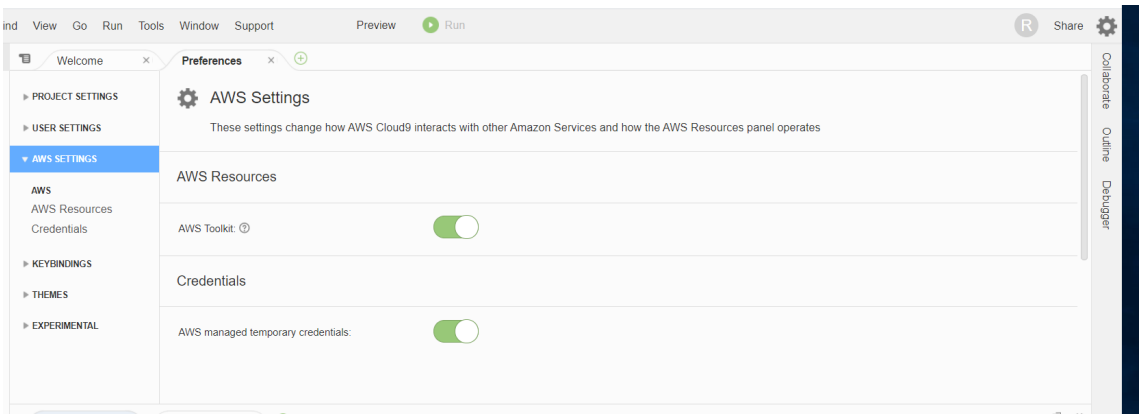
9. AdministratorAccess を指定し[次のステップ]をおします。次の画面はそのままさらに[次のステップ]をおします

10. 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。
11. IAM ロールが作成されたら、EC2 の画面に遷移します
12. 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)
13. [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

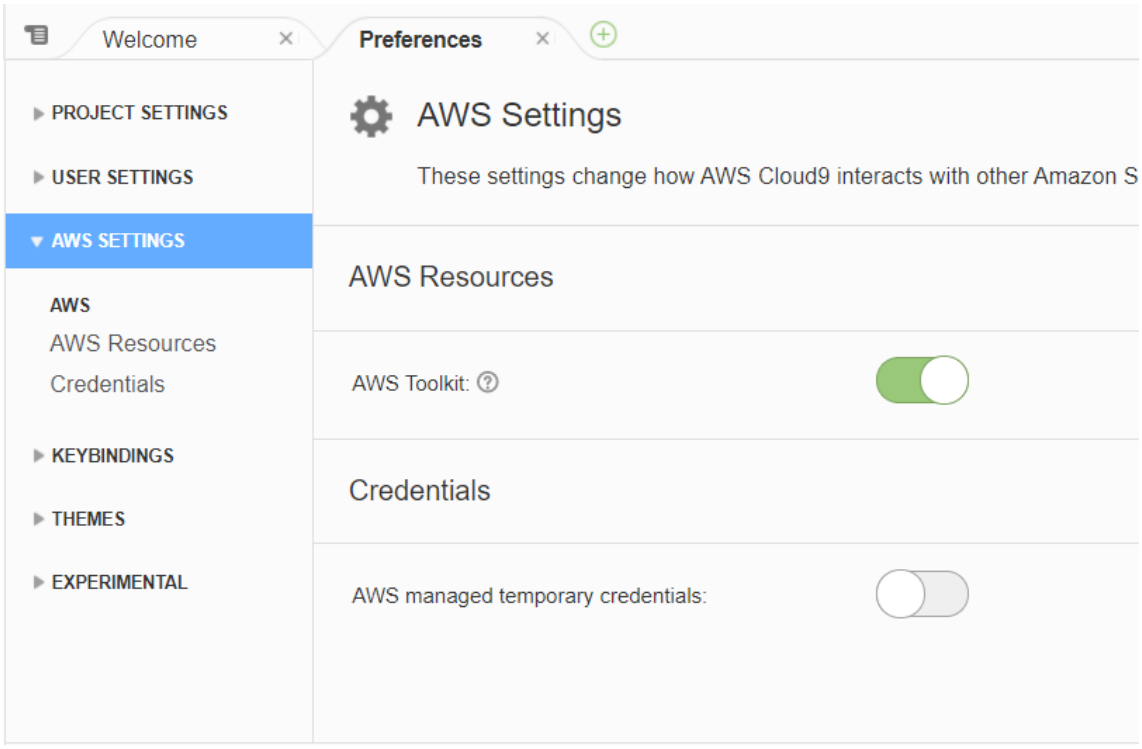


14. 先程作成した IAM ロールを選び[保存]をおします

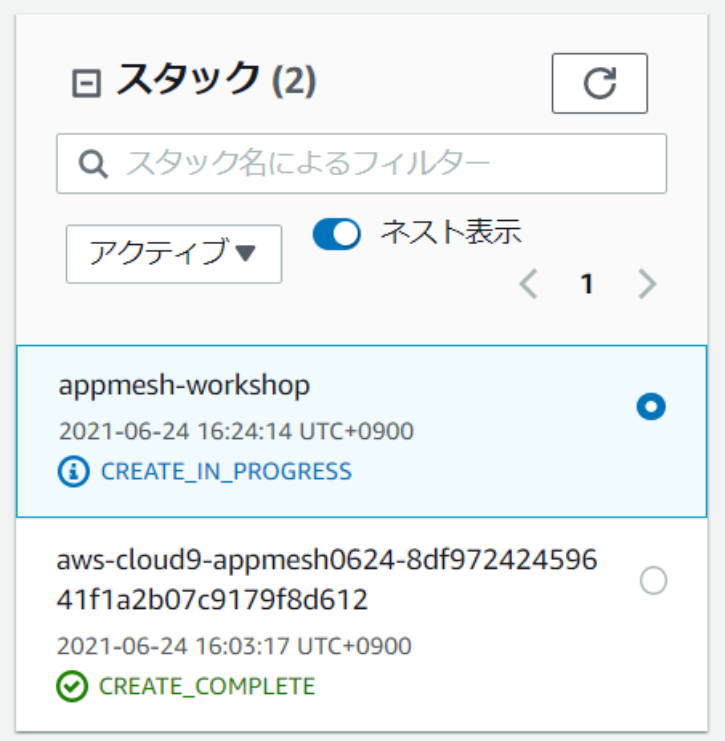
15. Cloud9 の画面に戻るとターミナルが使えるようになっているはずです。
16. 画面右上の歯車マークをおし、[AWS SETTINGS]を選びます。



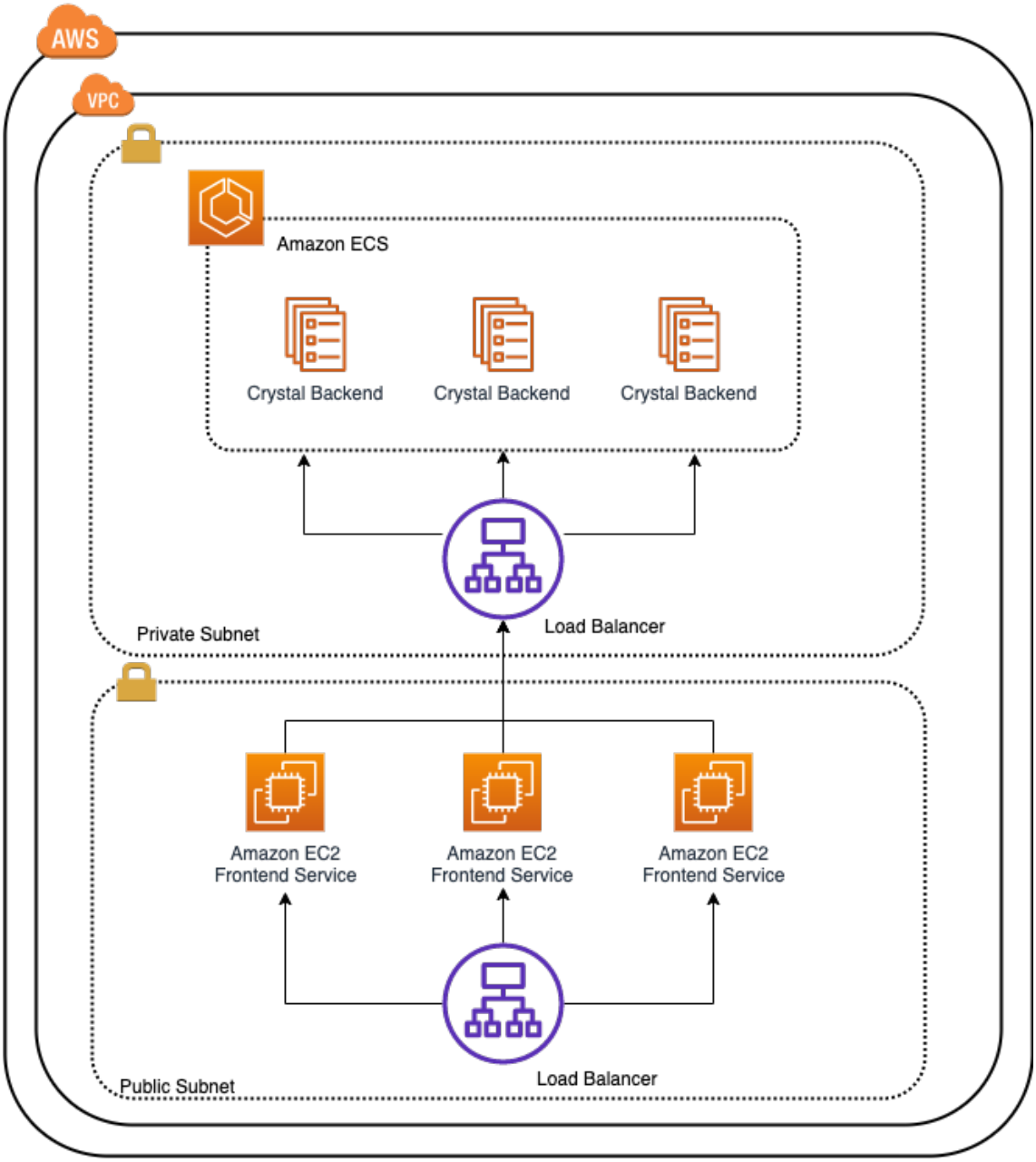
17. [Credentials]のチェックを外して、タブを閉じます



18. コマンド 1,2,3 を順番に実行します
19. 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
20. 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
21. 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
22. 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



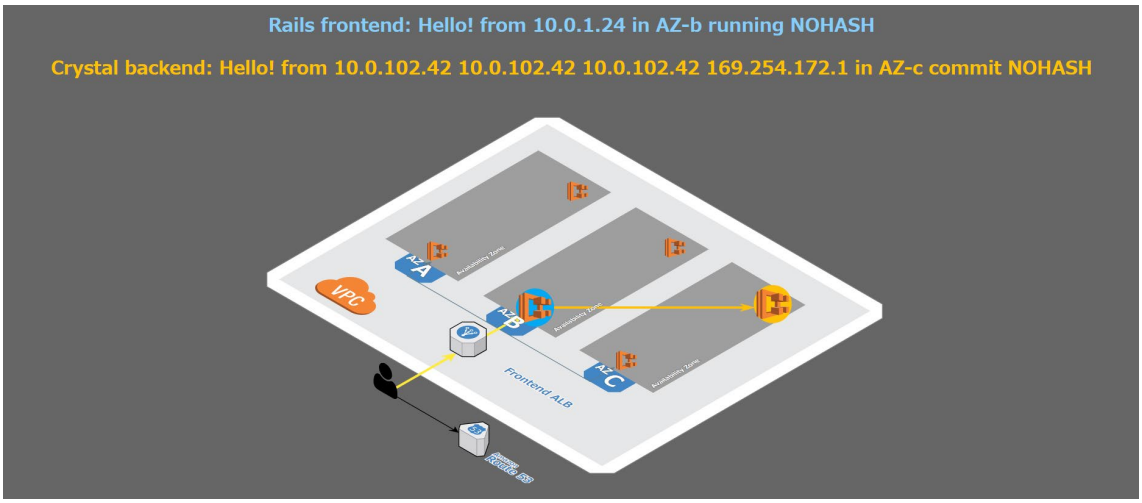
以下のような環境が構築されています。（実際は ECS・EKS の混在環境となるのでより複雑です）初期構築される環境は App Mesh を使わずに ELB がルーティングを行っています。この環境を App Mesh 及び Cloud Map をベースとしたネットワーク環境に作り替えていくことがこのワークショップの目的です。まだこの時点ではコンテナが起動していないため、アクセスができません。



23. 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
24. 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。

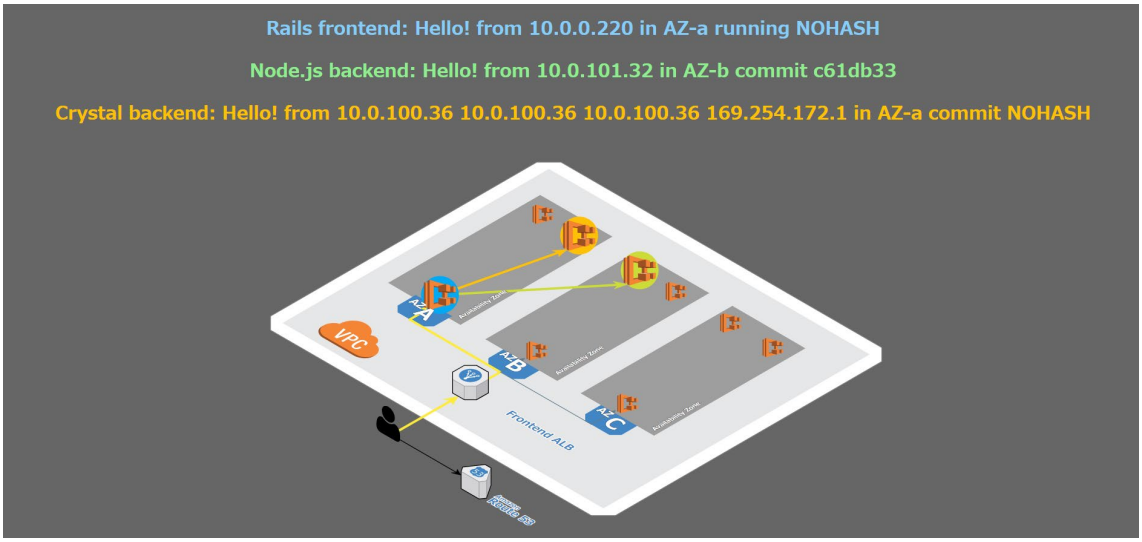
25. 完了したらコマンド 12 番を実行し表示された URL にブラウザでアクセスしてみてください。Frontend, Backend 両方とも動作しており、ラウンドロビンにより ELB がルーティングしていることがわかります。



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。

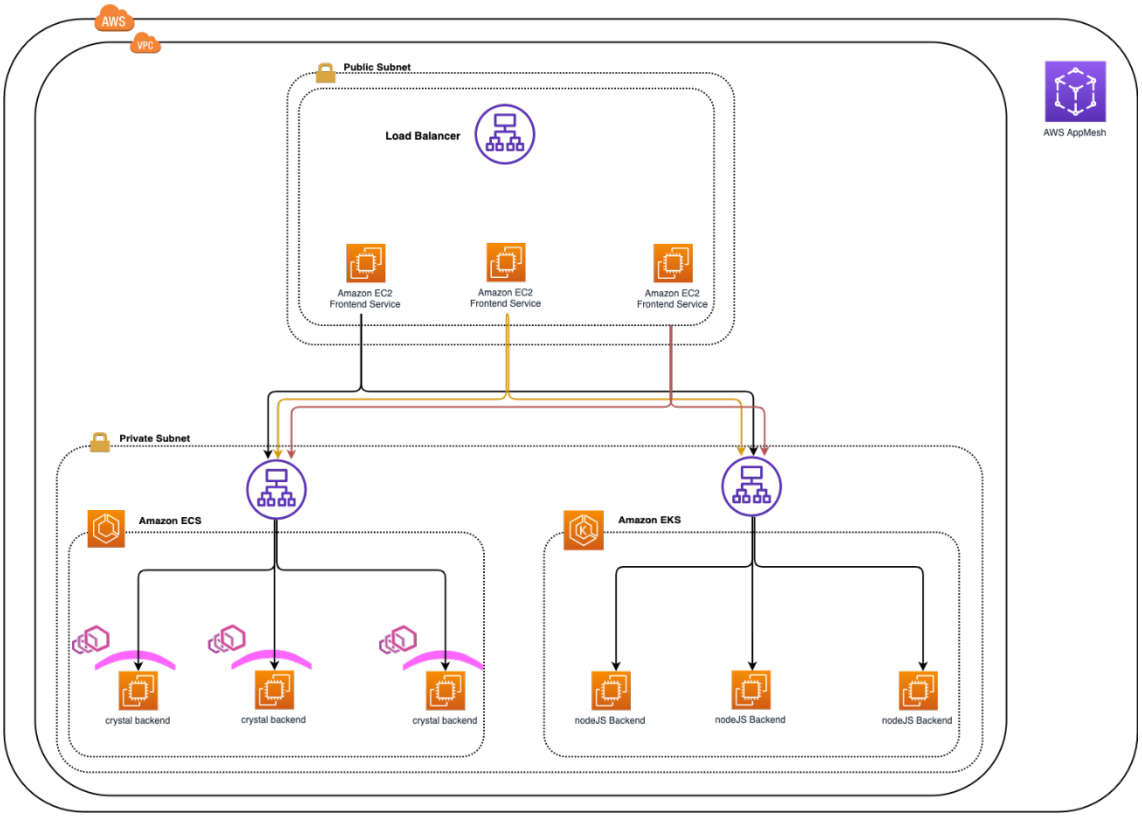
また、App Mesh のマネージメントコンソールを確認するとまだ現時点で何もできていないことがわかります。

26. 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。
27. 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。
28. 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。



29. コマンド 16 番を実行し、メッシュ環境を作成します

30. 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
31. 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
32. 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
33. 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。
34. 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。
35. 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

```
< HTTP/1.1 200 OK
< Date: Thu, 24 Jun 2021 08:32:23 GMT
< Content-Type: text/plain
< Content-Length: 64
< Connection: keep-alive
< x-envoy-upstream-service-time: 0
< server: envoy
<
Crystal backend: Hello! from 10.0.101.181 in AZ-b commit NOHASH
* Connection #0 to host 10.0.101.136 left intact
```

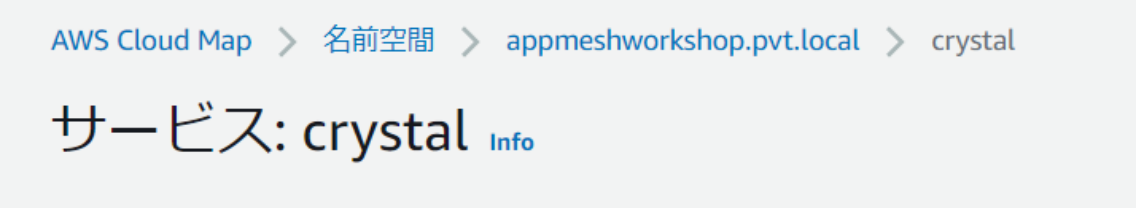
36. [exit]とタイプしセッションを停止します

[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。

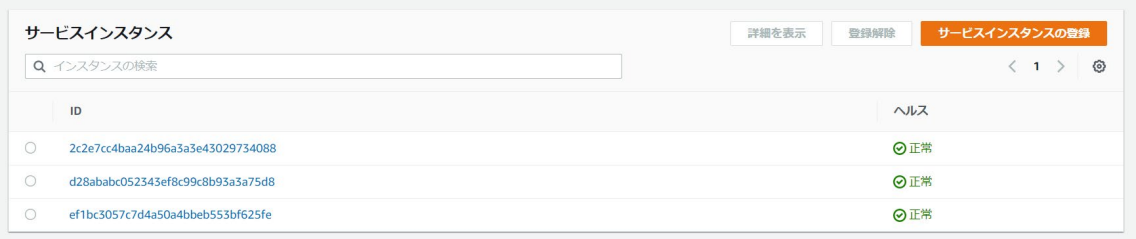
37. コマンド 24 番を実行し Cloud Map 用名前空間を作成します

38. 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

39. 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。
40. 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)
41. 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
42. 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
43. 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。



44. 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）
45. 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
46. 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
47. 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
48. 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
49. 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