SRA の技術情報誌
GS LetterNeo
Vol.155　2021 年 6 月

ECS 簡単デプロイツール
AWS Copilot CLI

山本 悠介

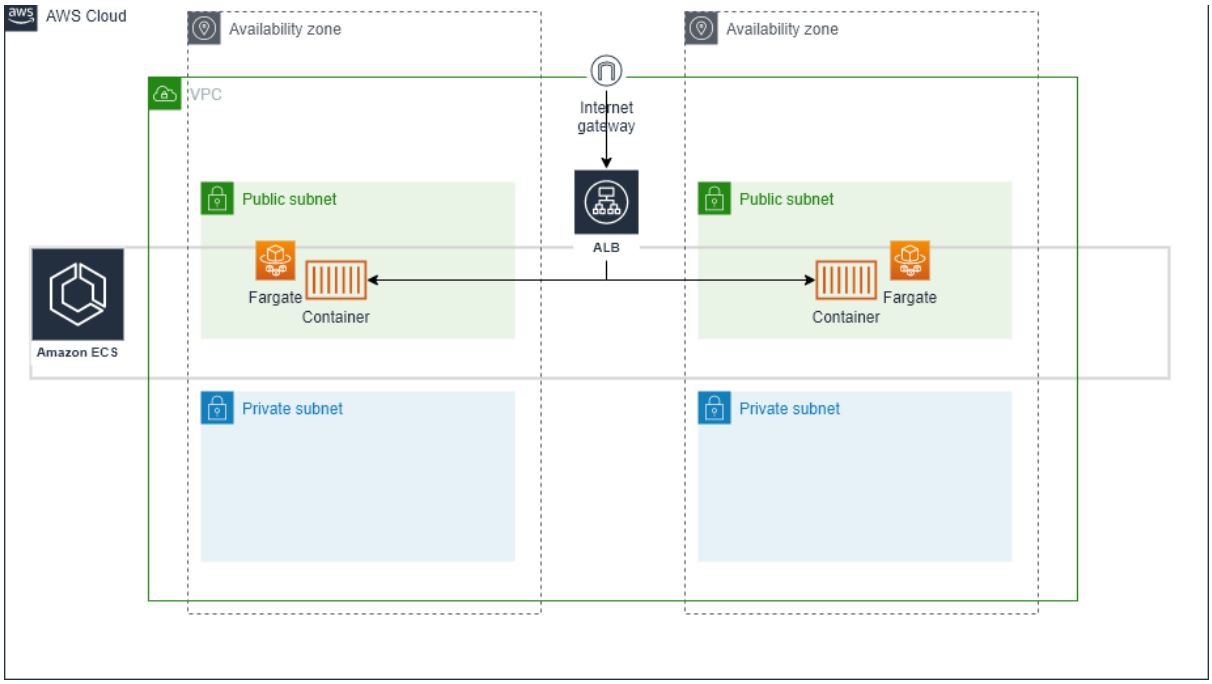
DX 技術本部

はじめに

AWS Copilot CLI は Amazon ECS CLI の後継にあたるもので、コンテナアプリケーションのビルド、リリース、運用を簡単に実現出来るツールになります。

本稿では、AWS Copliot CLI を利用し ECS クラスターを作成し、コンテナアプリケーションを動作させます。その簡単さが伝わればと思います。

作成する構成



シングルリージョン、マルチ AZ 構成でパブリックサブネット×2、プライベートサブネット×2 の一般的な構成の上で ECS クラスターを構築し、コンテナを Service として起動させます。

AWS Copilot CLI をインストールする

AWS Copilot CLI は Linux、macOS、Windows システムをサポートしていますが、今回は Linux システムにて進めます。

```
  $ sudo curl -Lo /usr/local/bin/copilot
https://github.com/aws/copilot-cli/releases/latest/download/copilot-
linux \
     && sudo chmod +x /usr/local/bin/copilot \
    && copilot -help
```

AWS 認証情報を設定する

`aws configure` を実行して AWS の認証情報を設定します。

```
$ aws configure
```

デプロイするアプリケーションを準備する

Nginx サーバをデモサービスとしてデプロイしてみます。利用する Dockerfile を記載します。

Dockerfile 内容)

```
FROM nginx:alpine
EXPOSE 80
COPY index.html /usr/share/nginx/html
```

表示する index.html を準備します。

index.html 内容例)

```
<html>
<head><title>copilot test</title></head>
<body>copilot test</body>
</html>
```

適当なディレクトリ(例は nginx)に配置します。

```
nginx
    Dockerfile
    index.html
```

アプリケーションをセットアップする

AWS Copilot には Application という Service と Environment、Pipeline を取りまとめる概念が存在します。上記で作成したディレクトリ内にて `copilot init` を実行します。

```
$ copilot init
```

実行すると、

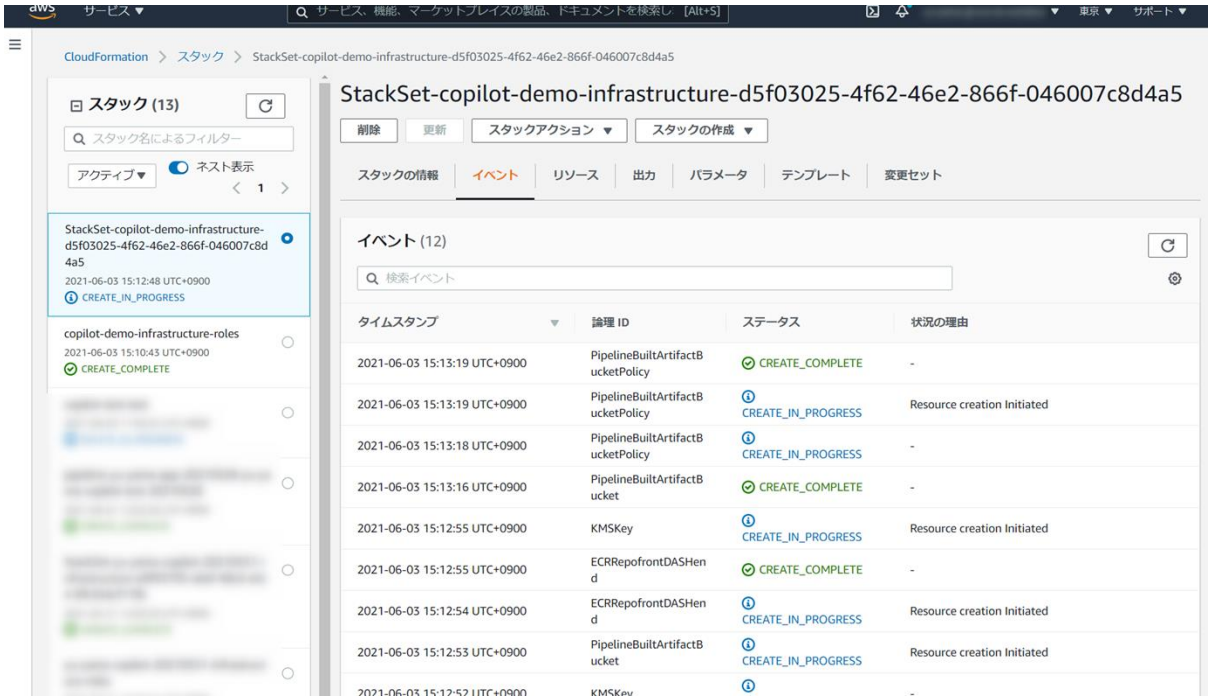
Application の名前：適当につけます。今回は copilot-demo とします
ワークロードタイプ：今回は Load Balanced Web Service を選択します
Service name：ECS ないに起動する Service 名。今回は front-end とします
image：./Dockerfile を選択します
port：公開するポート。今回は 80 とします

などそれぞれ聞かれるので、回答して進めます。回答し終わると、AWS Copilot がそれぞれ AWS リソースの作成を開始するので暫し待ちます。

テスト環境にデプロイする

アプリケーションのセットアップ終了後、test 環境へのデプロイ許可を問われるので、`y` を 選択し、デプロイを行います。

リソースの作成状況は CloudFormation でも確認できます。暫し待ちましょう。



デプロイが終了すると、エンドポイントが表示されます。

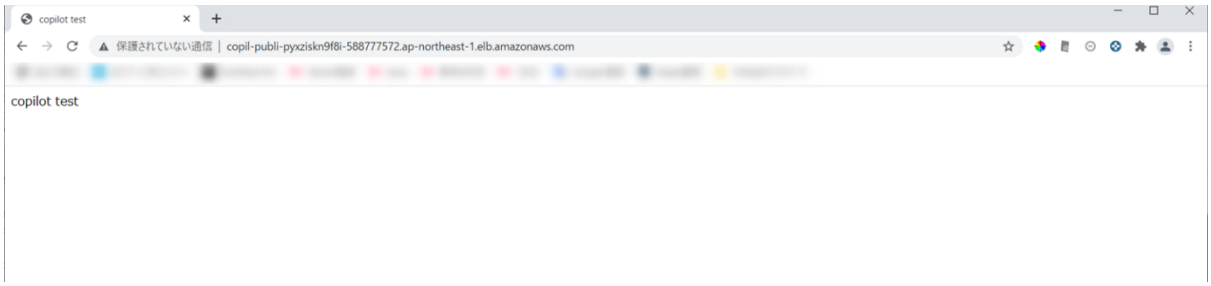
3.112.14.117 - ec2-user@ip-172-31-36-181:~/nginx VT

File Edit Setup Control Window KanjiCode Help

```
- Creating the infrastructure for stack copilot-demo-test-front-end              [create complete]
[289.3s]
  - Service discovery for your services to communicate within the VPC            [create complete]
[2.7s]
  - Update your environment's shared resources                                   [update complete]
[124.1s]
    - A security group for your load balancer allowing HTTP and HTTPS traffic    [create complete]
[7.3s]
    - An Application Load Balancer to distribute public traffic to your services [create complete]
[94.0s]
  - An IAM Role for the Fargate agent to make AWS API calls on your behalf       [create complete]
[21.4s]
  - A CloudWatch log group to hold your service logs                             [create complete]
[2.7s]
- Creating the infrastructure for stack copilot-demo-test-front-end              [create complete]
[289.3s]
  - Service discovery for your services to communicate within the VPC            [create complete]
[2.7s]
  - Update your environment's shared resources                                   [update complete]
[124.1s]
    - A security group for your load balancer allowing HTTP and HTTPS traffic    [create complete]
[7.3s]
    - An Application Load Balancer to distribute public traffic to your services [create complete]
[94.0s]
  - An IAM Role for the Fargate agent to make AWS API calls on your behalf       [create complete]
[21.4s]
  - A CloudWatch log group to hold your service logs                             [create complete]
[2.7s]
- Creating the infrastructure for stack copilot-demo-test-front-end              [create complete]
[289.3s]
  - Service discovery for your services to communicate within the VPC            [create complete]
[2.7s]
  - Update your environment's shared resources                                   [update complete]
[124.1s]
    - A security group for your load balancer allowing HTTP and HTTPS traffic    [create complete]
[7.3s]
    - An Application Load Balancer to distribute public traffic to your services [create complete]
[94.0s]
  - An IAM Role for the Fargate agent to make AWS API calls on your behalf       [create complete]
[21.4s]
  - A CloudWatch log group to hold your service logs                             [create complete]
[2.7s]
  - An ECS service to run and maintain your tasks in the environment cluster     [create complete]
[78.4s]
    Deployments

               Revision  Rollout      Desired  Running  Failed  Pending

      PRIMARY  1         [completed]  1        1        0       0

  - A target group to connect the load balancer to your service                  [create complete]
[2.7s]
  - An ECS task definition to group your containers and run them on ECS          [create complete]
[4.9s]
  - An IAM role to control permissions for the containers in your tasks          [create complete]
[22.2s]
? Deployed front-end, you can access it at http://copil-Publi-PYXZISKN9F8I-588777572.ap-northeast-1.elb.amazonaws.com.
[ec2-user@ip-172-31-36-181 nginx]$
```

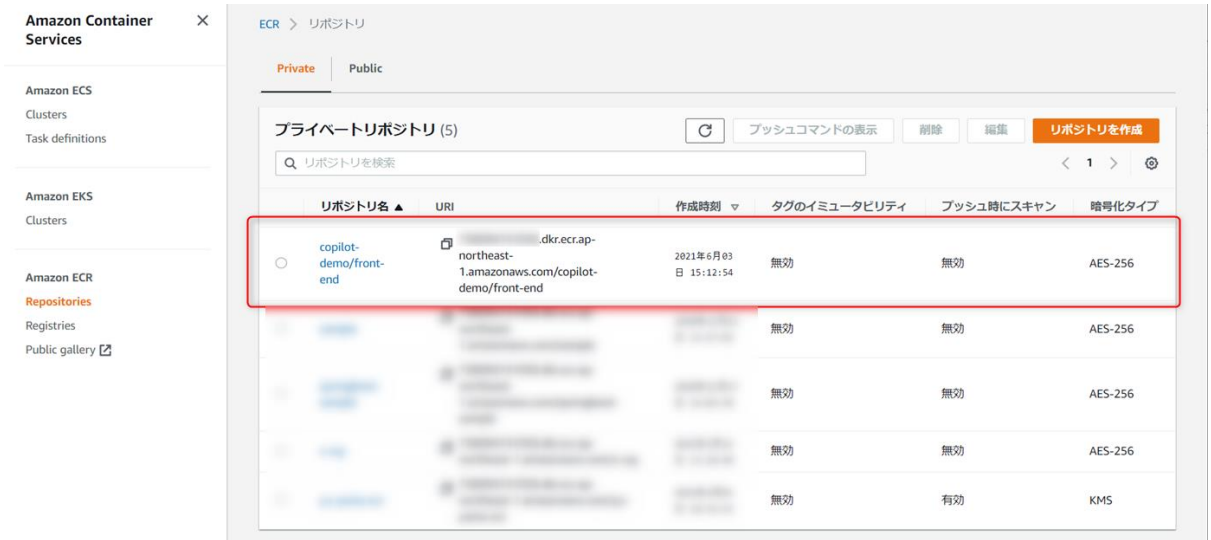
動作確認

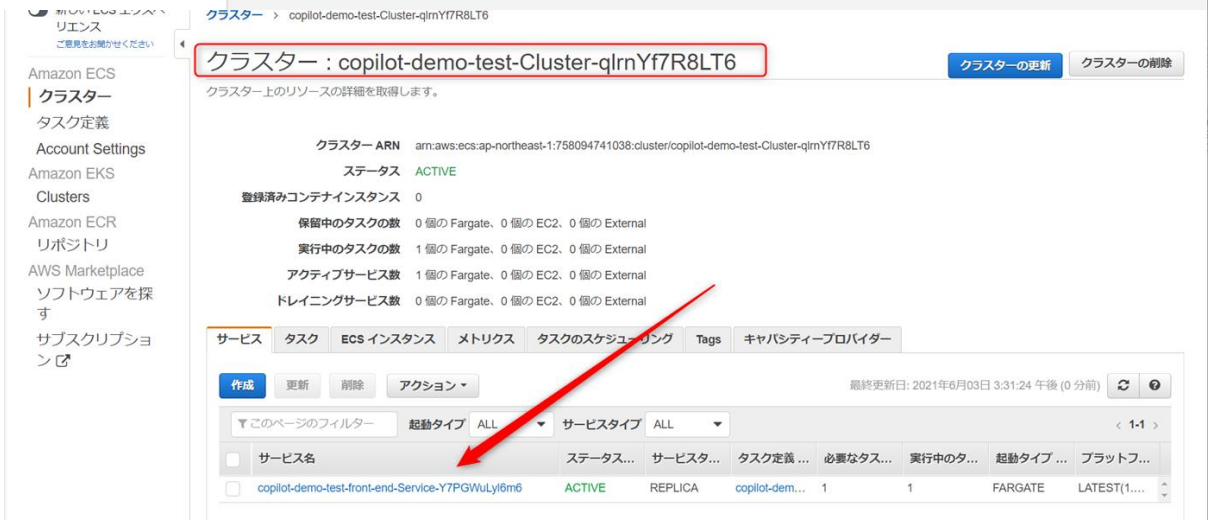
アクセスしてみると．．．



表示されています！

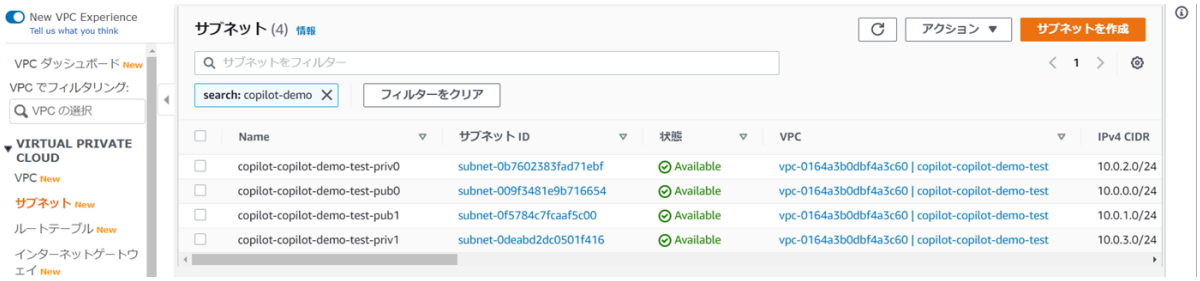
ECR や ECS クラスター、Service が作成されているのも確認できます。





パブリックサブネット×2、プライベートサブネット×2 が作成されていることも確認できます。

copilot-{アプリケーション名}-{環境名}-pub0 の様に名前が付くようですね。



後片付け

`copilot app delete`を実行して作成したリソースを全て削除します。

```
$ copilot app delete
```

まとめ

以上、AWS Copilot CLI にてコンテナアプリケーションのビルド、リリースを行いました。簡単さが伝わりましたでしょうか。

今回はワークロードとして Load Balanced Web Service を選びましたが、内部通信用の Backend Service、バッチなどに使用できる Scheduled job なども用意されています。

既存のサブネットを指定してコンテナをデプロイすることも可能です。

CI/CD パイプライン(CodePipeline)もコマンドで作成することが出来ます。

AWS のアカウントさえあればすぐ試せるので興味がある方はやってみてください！

参考 URL

AWS Copilot CLI - AWS Copilot CLI https://aws.github.io/copilot-cli/

Amazon ECS アプリケーションのデプロイによる AWS Copilot の開始方法 - Amazon Elastic Container Service https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/getting-tarted-aws-copilot-cli.html

GSLetterNeo Vol.155
2021 年 6 月 20 日発行
発行者　株式会社 SRA 先端技術研究所

編集者　土屋正人　熊澤努　方 学芬
バックナンバー http://www.sra.co.jp/gsletter
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん