

ECS 簡単デプロイツール AWS Copilot CLI

山本 悠介

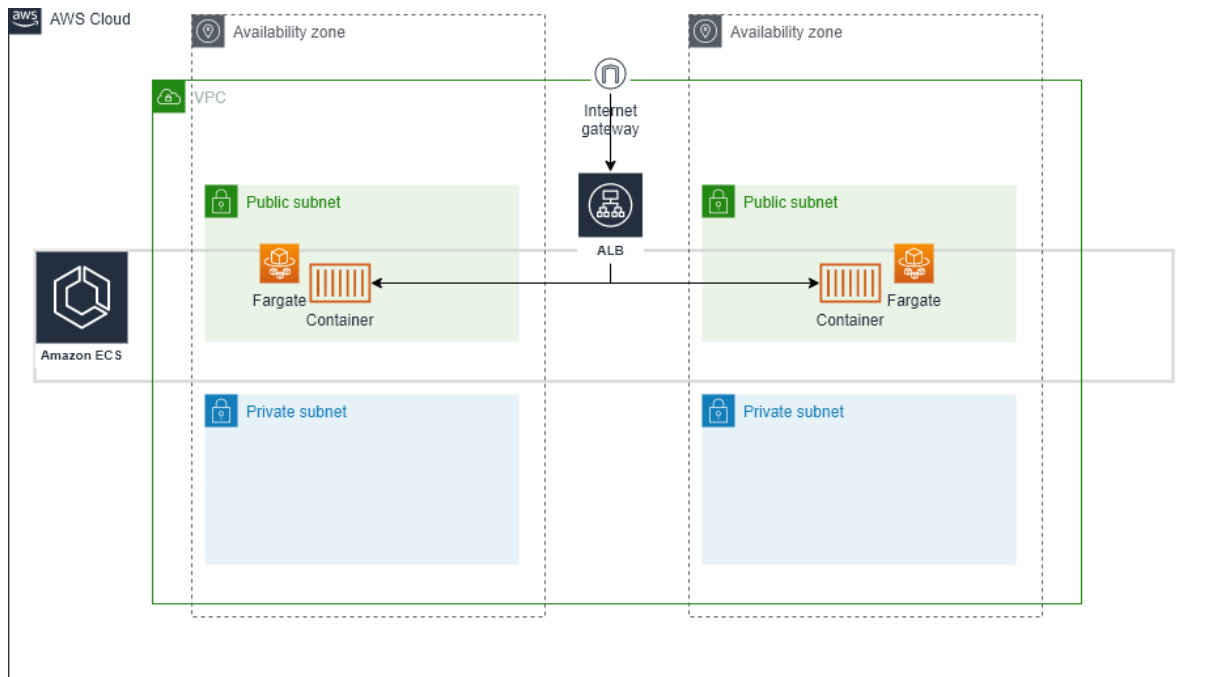
DX 技術本部

🚧 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 Copilot 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 잠시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。 잠시待ちましょう。

The screenshot shows the AWS CloudFormation console for a StackSet. The main panel displays the 'イベント (12)' (Events) tab for the StackSet 'StackSet-copilot-demo-infrastructure-d5f03025-4f62-46e2-866f-046007c8d4a5'. The events table shows the following data:

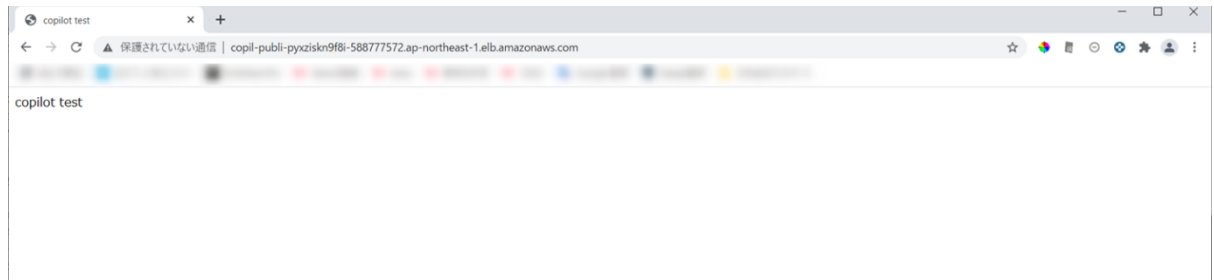
タイムスタンプ	論理 ID	ステータス	状況の理由
2021-06-03 15:13:19 UTC+0900	PipelineBuildArtifactBucketPolicy	CREATE_COMPLETE	-
2021-06-03 15:13:19 UTC+0900	PipelineBuildArtifactBucketPolicy	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-03 15:13:18 UTC+0900	PipelineBuildArtifactBucketPolicy	CREATE_IN_PROGRESS	-
2021-06-03 15:13:16 UTC+0900	PipelineBuildArtifactBucket	CREATE_COMPLETE	-
2021-06-03 15:12:55 UTC+0900	KMSKey	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-03 15:12:55 UTC+0900	ECRReprofrontDASHend	CREATE_COMPLETE	-
2021-06-03 15:12:54 UTC+0900	ECRReprofrontDASHend	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-03 15:12:53 UTC+0900	PipelineBuildArtifactBucket	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-03 15:12:52 UTC+0900	KMSKey	CREATE_IN_PROGRESS	-

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]
[4.8s]
- 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-PYZISK9F8I-58877572.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。

The screenshot displays the AWS Management Console interface. On the left, the navigation menu includes Amazon Container Services, Amazon ECS (Clusters, Task definitions), Amazon EKS (Clusters), and Amazon ECR (Repositories, Registries, Public gallery). The main content area is divided into two sections:

ECR > リポジトリ

The 'Private' tab is selected, showing a list of private repositories. A red box highlights the first repository:

リポジトリ名	URI	作成時刻	タグのイミュータビリティ	プッシュ時にスキャン	暗号化タイプ
copilot-demo/front-end	1.dkr.ecr.ap-northeast-1.amazonaws.com/copilot-demo/front-end	2021年6月03日 15:12:54	無効	無効	AES-256

クラスター > copilot-demo-test-Cluster-qlmYf7R8LT6

The cluster details page shows the following information:

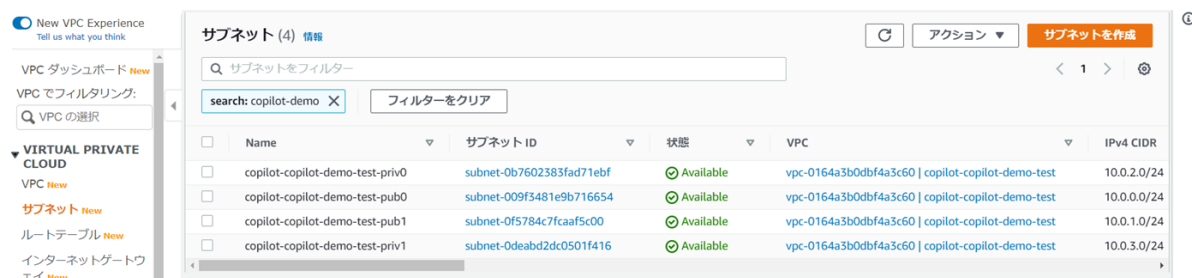
- クラスター ARN: arn:aws:ecs:ap-northeast-1:758094741038:cluster/copilot-demo-test-Cluster-qlmYf7R8LT6
- ステータス: ACTIVE
- 登録済みコンテナインスタンス: 0
- 保留中のタスクの数: 0 個の Fargate, 0 個の EC2, 0 個の External
- 実行中のタスクの数: 1 個の Fargate, 0 個の EC2, 0 個の External
- アクティブサービス数: 1 個の Fargate, 0 個の EC2, 0 個の External
- ドレーニングサービス数: 0 個の Fargate, 0 個の EC2, 0 個の External

The 'タスク' (Tasks) tab is selected, showing a table of tasks. A red arrow points to the first task:

サービス名	ステータス	サービスタイプ	タスク定義	必要なタスク	実行中のタスク	起動タイプ	プラットフォーム
copilot-demo-test-front-end-Service-Y7PGWuLyf6m6	ACTIVE	REPLICA	copilot-dem...	1	1	FARGATE	LATEST(1...

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できます。

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。

**夢を。Yawaraka Innovation**
やわらかいのべーしょん