

AWSとGitHubを使ってみよう勉強会

～ 第5回 AWS EC2環境の構築 ～

株式会社 豆蔵
ビジネスソリューション事業部



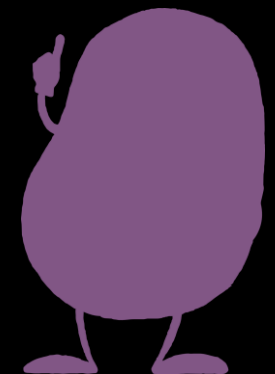
本日の内容

- 前回の課題の解説(25分)
- 次回までの課題の説明(20分)

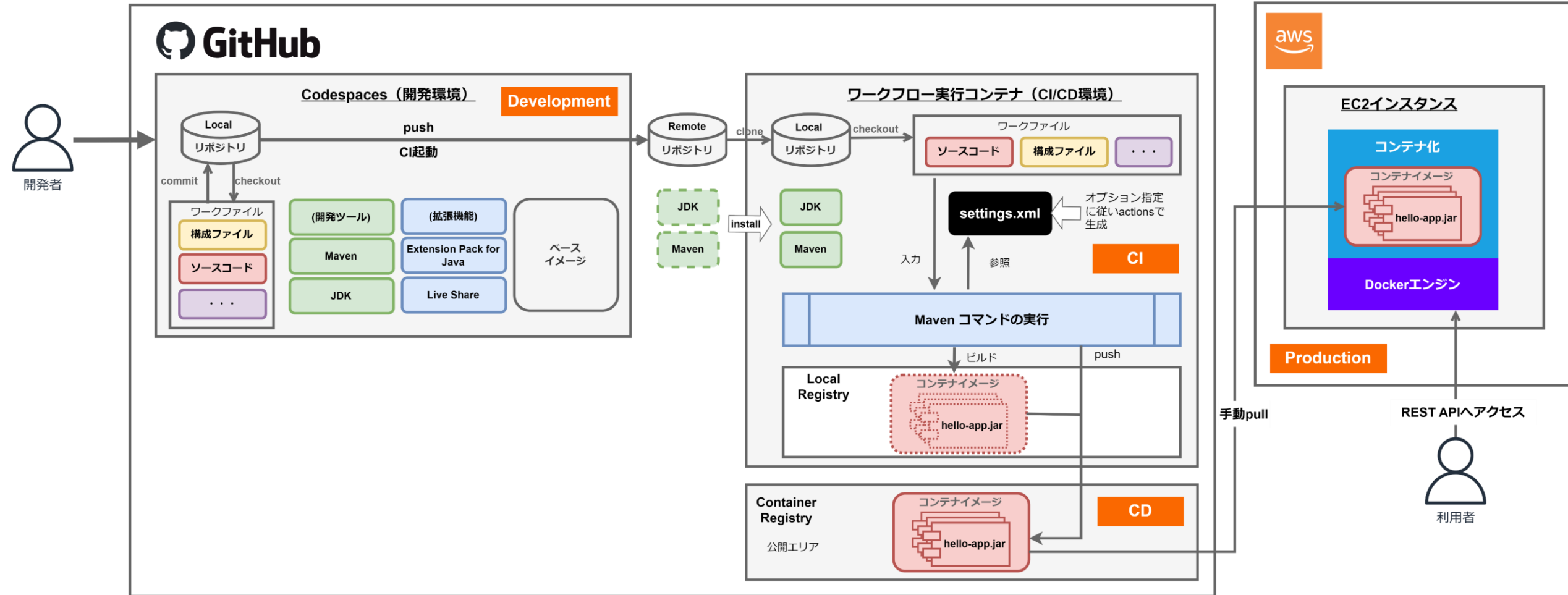
※前回の課題は手順どおりに実施するものなので「前回の課題の回答」はなしで前回の課題の解説を行います

前回の課題の解説

- ・これでまでの課題でやってきた全体像
- ・AWSのユーザと権限
- ・AWSのネットワーク構成
- ・最後にEC2は



これまでの課題でやってきた全体像 - 振り返り

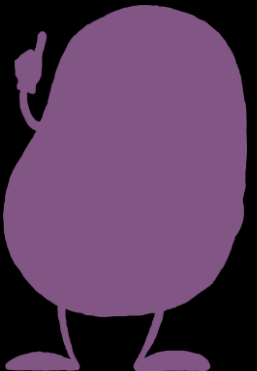


- ✓ 開発からCI/CD、プロダクションでの実行まですべてクラウド環境で実現！
- ✓ プロダクションへのデプロイは手動だがコミットからリポジトリの登録まではすべて自動！

前回の課題の解説

- ・これでまでの課題でやってきた全体像
- ・AWSのユーザと権限
- ・AWSのネットワーク構成
- ・最後にEC2は

AWSのユーザと権限、AWSのネットワーク構成は厳密性よりも分かりやすさを優先しかなり意識しているものもあります。したがって、100点満点で正確な説明をしている訳ではありませんのでその点は予めご承知おきください



ルートユーザとIAMユーザ

サインイン

☐ ルートユーザー

無制限アクセスを必要とするタスクを実行するアカウント所有者。 [詳細はこちら](#)

☒ IAM ユーザー

日常的なタスクを実行するアカウント内のユーザー。 [詳細はこちら](#)

アカウント ID (12 桁) またはアカウントエイリアス

次へ

最初に作ったのがルートユーザで次に作ったのIAMユーザ。
ログインから分かれてるけど、これってなんだ??

IAMユーザを作るときにはグループを指定したけど、これっているの??

許可を設定

既存のグループにユーザーを追加するか、新しいグループを作成します。グループを使用することは、職務機能別にユーザーの許可を管理するためのベストプラクティスの方法です。 [詳細はこちら](#)

許可のオプション

☒ ユーザーをグループに追加

ユーザーを既存のグループに追加するか、新しいグループを作成します。グループを使用して、職務別にユーザーの許可を管理することをお勧めします。

☐ 許可のコピー

既存のユーザーから、すべてのグループメンバーシップ、アタッチされた管理ポリシー、およびインラインポリシーをコピーします。

☐ ポリシーを直接アタッチする

ユーザーにマネージドポリシーを直接アタッチします。ベストプラクティスとして、代わりにグループにポリシーをアタッチすることをお勧めします。次に、ユーザーを適切なグループに追加します。

ユーザーグループ (1/1)

検索



グループを作成

< 1 > ⚙️

<input checked="" type="checkbox"/>	グループ名	ユーザー	アタッチされたポリシー	作成日
<input checked="" type="checkbox"/>	Administrators	1	AdministratorAccess	2021-03-07 (2 年前)

許可の境界を設定 - オプション

アタッチされたポリシーってなに??

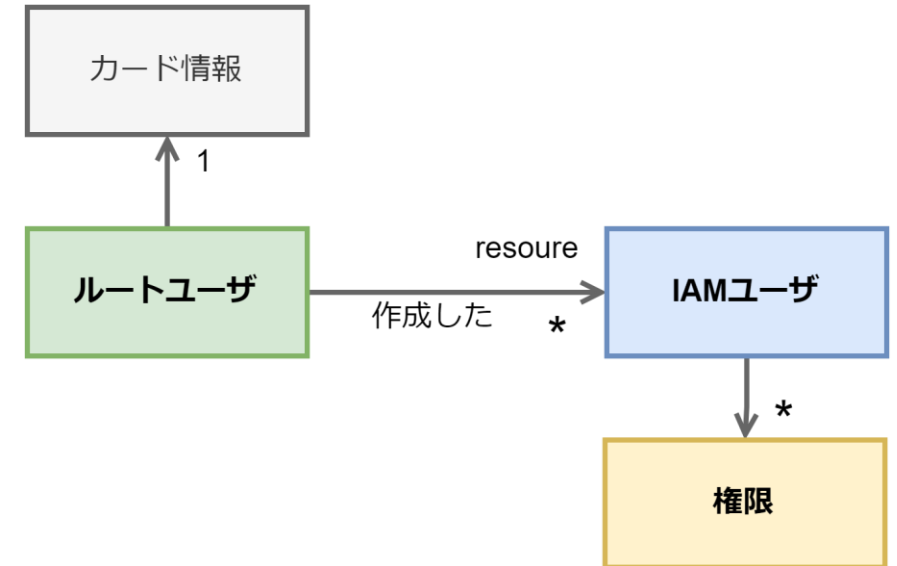
キャンセル

前へ

次へ

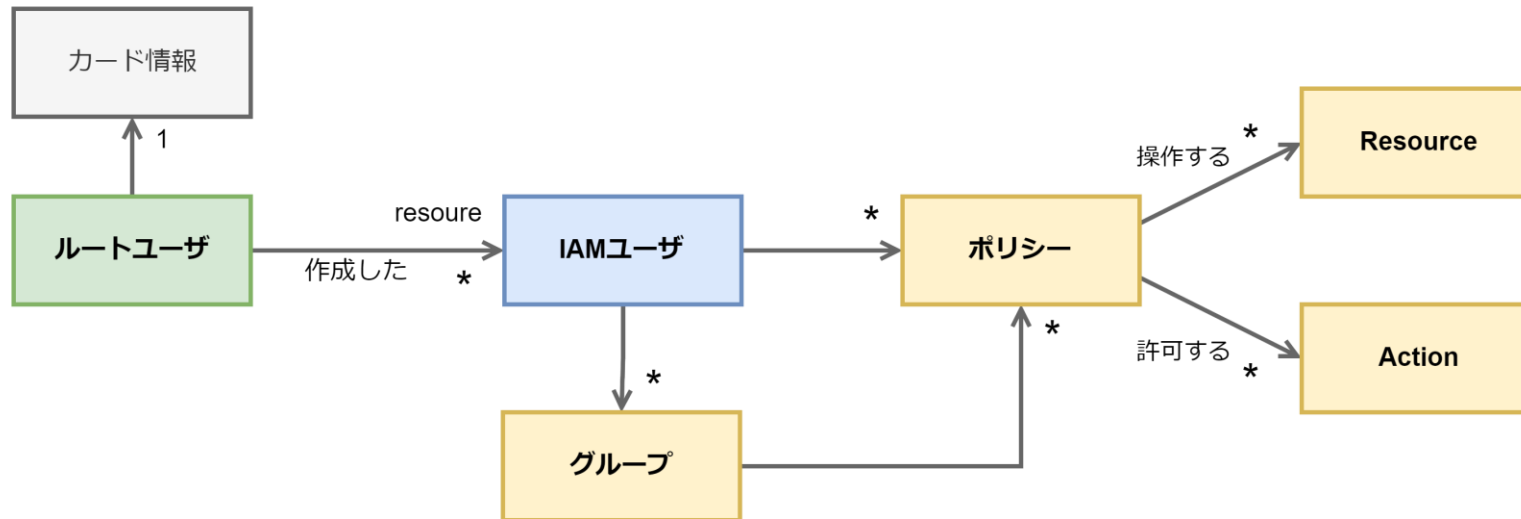
ルートユーザとIAMユーザ

- ルートユーザ
 - AWSアカウント作成時に最初に用意されアカウント
 - 予めすべての権限が付与されてる。これを制限する方法はないので、非常に権限の強いユーザとなる
 - AWSからの請求につかうクレジットカード情報が紐づいている
- IAMとは
 - ID と AWS のサービスおよびリソースへのアクセスを一元管理するAWS独自の権限管理の仕組み
 - 日本では「アイアム」、英語圏では「アイエーエム」と呼ばれることが多い
- IAMユーザとは
 - IAMで作成したユーザ。ルートユーザは最初からフルアクセス権が付いているのに対して、IAMユーザで何をするには必要な権限を付与する必要がある
 - AWSのユーザの種類にはルートユーザとIAMユーザの2種類しかなく、IAMユーザはLinux等によく言われる「ルートユーザ」に対する「一般ユーザ」の意味に等しい

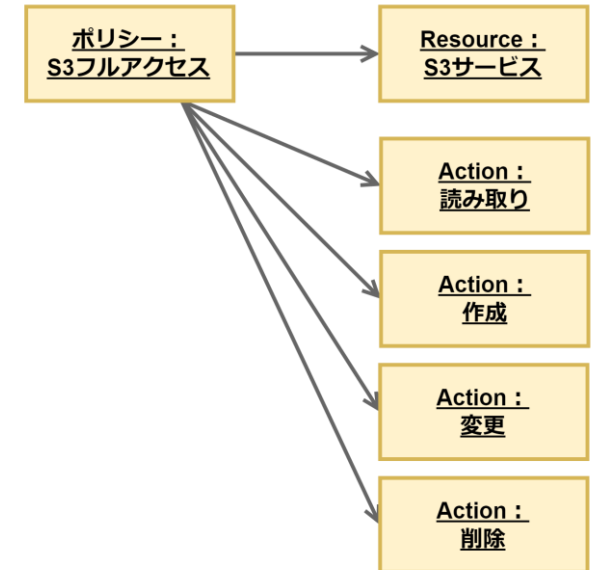


権限 – グループとポリシー

AWSの権限構造



S3のポリシー定義の例



• ポリシー

- なんの「Resource」をどう「Action」できるかを定義したもので、一般的に言われる「権限」に相当するもの
- ストレージサービスのS3に対するフルアクセスを表すポリシーの実体イメージとしては次のようになる

• グループ

- AWSではポリシーをまとめるための手段としてグループが用意されている
- グループに所属しているIAMユーザはグループを経由してそのグループに付与されているすべてのポリシーが付与される

• ポリシーの割り当て

- ポリシーはIAMユーザに対して直接割り当てることも、グループを経由して割り当てることもできる

再度見てみる - ルートユーザとIAMユーザ

サインイン

☐ ルートユーザー

無制限アクセスを必要とするタスクを実行するアカウント所有者。 [詳細はこちら](#)

☒ IAM ユーザー

日常的なタスクを実行するアカウント内のユーザー。 [詳細はこちら](#)

アカウント ID (12 桁) またはアカウントエイリアス

次へ

ルートユーザとIAMユーザは別物なんだね！
IAMユーザってヘンテコな名前だけど「一般ユーザ」ということか

グループはユーザをグルーピングする目的の他にも権限を割り当てる目的もあるんだネ！

許可を設定

既存のグループにユーザーを追加するか、新しいグループを作成します。グループを使用することは、職務機能別にユーザーの許可を管理するためのベストプラクティスの方法です。 [詳細はこちら](#)

許可のオプション

☒ ユーザーをグループに追加
ユーザーを既存のグループに追加するか、新しいグループを作成します。グループを使用して、職務別にユーザーの許可を管理することをお勧めします。

☐ 許可のコピー
既存のユーザーから、すべてのグループメンバーシップ、アタッチされた管理ポリシー、およびインラインポリシーをコピーします。

☐ ポリシーを直接アタッチする
ユーザーにマネージドポリシーを直接アタッチします。ベストプラクティスとして、代わりにグループにポリシーをアタッチすることをお勧めします。次に、ユーザーを適切なグループに追加します。

ユーザーグループ (1/1)

検索

<input checked="" type="checkbox"/>	グループ名	ユーザー	アタッチされたポリシー	作成日
<input checked="" type="checkbox"/>	Administrators	1	AdministratorAccess	2021-03-07 (2 年前)

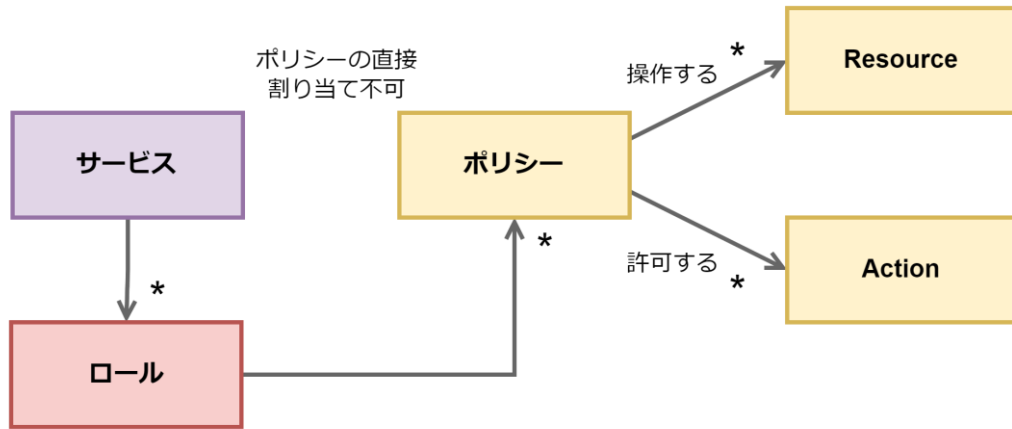
許可の境界を設定 - オプション

これはグループを経由して割り当てられる権限なんだネ

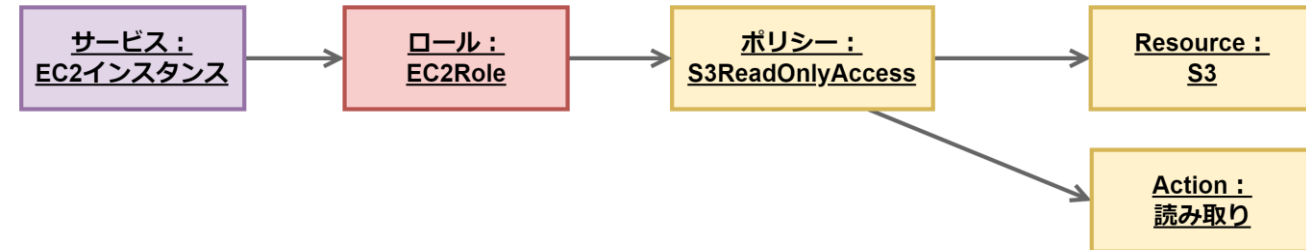
キャンセル 前へ 次へ

AWSには「ロール」もある – ややこしい・・・

ロールの場合の権限構造



ロールを使ったポリシーの設定例



• サービス

- EC2やS3などのAWSのサービス。もっと平たく言えば実行プログラム
- 例えばあるEC2インスタンスからS3を参照する際に該当のEC2インスタンスはS3に対する操作権限が必要となるため、サービスにもポリシー（権限）が必要となる

• ロール

- サービスに割り当てるポリシーをグルーピングするもの
- 一般的に「ロール」というとユーザに割り当てるイメージが強いが、AWSではユーザに割り当てるのがグループで、サービスに割り当てるものがロールとなる。ロールをユーザに割り当てることはできない。
- （ほんとに最低のネーミングで初めましての人のほとんどが混乱する）

• ポリシーの割り当て

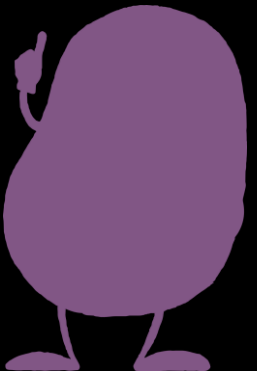
- サービスにポリシーを直接割り当てることはできない。よって、必ずロールを作成して付与する必要がある

まとめ - AWSのユーザと権限

- AWSの権限周りは難しく感じるが実はネーミングがヘンなだけ
 - ① **IAMユーザ**は要は自分で権限を設定する**一般ユーザ**
 - ② **ポリシー**は要は**権限**
 - ③ **グループ**はポリシーをまとめて**ユーザ**に割り当てるもの
 - ④ **ロール**はユーザではなく**サービス**にポリシーをまとめて割り当てるもの
 - ⑤ ユーザにはポリシーを直接割り当てられるが、サービスには割り当てることができない
 - ⑥ なので、サービスにポリシーを割り当てるには必ずロールを経由する必要がある
- これがイメージできればどんなサービスを使うことになってもバッチリ（個人の感想です）

前回の課題の解説

- ・これでまでの課題でやってきた全体像
- ・AWSのユーザと権限
- ・AWSのネットワーク構成
- ・最後にEC2は



AWSのネットワーク構成

課題は1つのEC2インスタンスしか使っていませんが、構成が分かりやすくなるようここでは複数の要素を使った例にしています



インターネットゲートウェイ



アドレス変換テーブル

グローバル	プライベート
52.192.212.7	172.31.32.15
52.195.212.18	172.31.16.7

アベイラリティゾーン
ap-northeast-1a

アベイラリティゾーン
ap-northeast-1c

VPC
172.31.0.0/16

ルートテーブル

送信先	ターゲット
172.31.0.0/16	local
0.0.0.0/0	Internet G/W

参照

参照

Public subnet
172.31.32.0/20

Security group



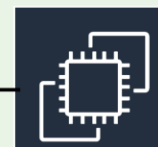
172.31.32.15
(52.195.212.7)

ルーティング

インバウンドルール

protocol	ポート	送信元
TCP	7001	0.0.0.0/0
TCP	22(SSH)	0.0.0.0/0

Public subnet
172.31.16.0/20



172.31.16.7
(52.195.212.18)

ルーティング

■ VPC

- AWS上のプライベートな仮想ネットワーク環境
- 実体はCIDRで区切られたプライベートアドレス空間

■ インターネットゲートウェイ(IGW)

- インターネットとVPCを中継するもの
- グローバルアドレス空間のインターネットとプライベートアドレス空間のVPCを繋ぐもので実体はNAT
- IGWに繋がっているサービスに割り当てられたグローバルIPとプライベートIPを1対1で紐づける。これによりIGWに繋がっているEC2などのサービスは外部からグローバルIPでアクセス可能となる
- よって、プライベートアドレス空間内に存在するサービスが直接グローバルIPアドレスのインタフェースを持つ訳ではない

■ セキュリティグループ

- EC2などのサービスはいずれかのセキュリティグループに紐づけられており、そのセキュリティグループを通してアクセスされる。
- このセキュリティグループは実質的にfirewallの役割となる。よって、外部からのアクセスを許可するにはセキュリティグループでポートを開ける必要がある

■ ルートテーブル

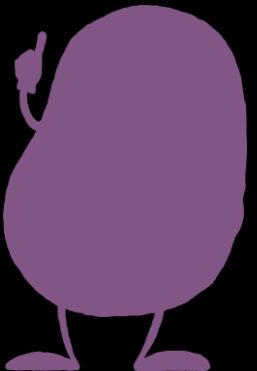
- サブネットには必ず1つのルートテーブルが紐づけられている
- サブネット内のルーティングは紐づけられているルートテーブルを参照して決定される

■ サブネット

- VPCをさらにCIDRで分割したアドレス空間
- publicとprivateの2種類があり違いは内部のサービスをIGWに直接つながるかになる
- IGWに直接繋がられる場合、NATの対象となるため、直接外部とアクセスが可能となる

前回の課題の解説

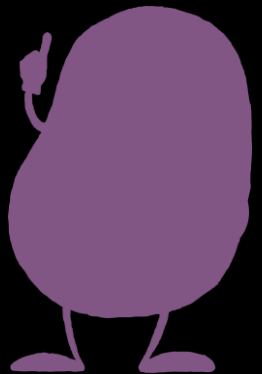
- ・これでまでの課題でやってきた全体像
- ・AWSのユーザと権限
- ・AWSのネットワーク構成
- ・最後にEC2は



最後にEC2は..

- 環境を構築して分かったかと思いますが、EC2自体はただのオンプレのサーバとなんら変わりません
- クラウドならではのAWSならではのことはここまでで説明したユーザと権限、ネットワーク構成が基本となります
- ここまでの内容が理解できれば、ECS Fargetaなど他のサービスもすんなり使うことができるかと思います
- 重要な内容なので、ネットや書籍などで復習してみましょう

次回までの課題の説明



次回までの課題

- テーマ
 - GitHub Packagesで公開したコンテナイメージを今度はAWSのECS Fargateで動かす
 - コンテナ化で環境変数は重要な役割を果たすのでそれを使ってみる
 - 今回はちょっと難しいかもしれませんが最後なので頑張りましょう！
- お題
 - 環境変数の設定でサンプルアプリの動作を変えられるようにする
 - 改変したアプリをEC2で動かして確認してみる
 - ECS Fargateにデプロイしてサンプルアプリを動かしてみる
- ゴール
 - REST APIで返す値を環境変数で指定できること
 - サンプルアプリがECS Fargateで動作すること

次回で補足しますがECS Fargateがどのようなものか知り場合は検索すると沢山出てきますので、そちらを参照ください

課題の実施手順

Step1. REST APIで返す値を環境変数で指定できるようにサンプルアプリを修正する

Step2. コンテナをビルドしEC2からpullして動作を確認する

Step3. ECS Fargateにデプロイして動作を確認する

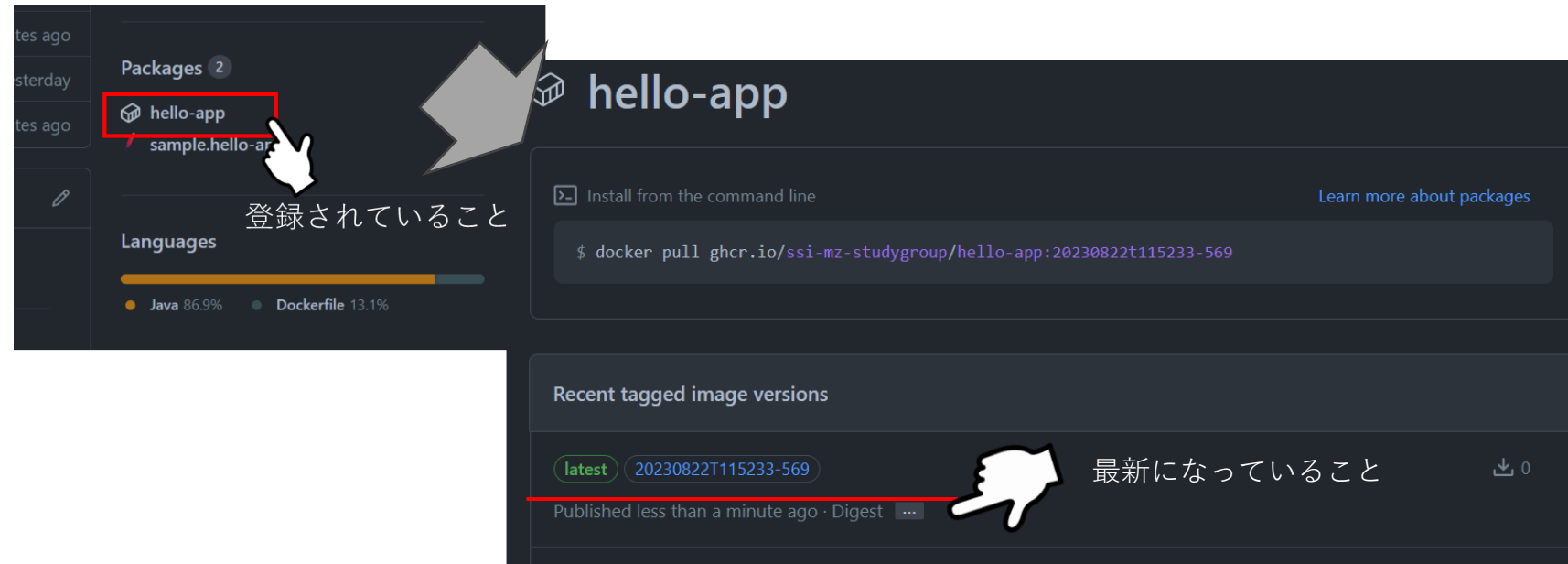
Step4. 今回の課題

Step1. REST APIで返す値を環境変数で指定できるようにサンプルアプリを修正する

- 変更内容
 - 環境変数CONFIG_VALの設定が
 - ある場合は、その設定値をREST APIで返す
 - ない場合は、デフォルト値として"Hello"
- 変更方法
 - MicroProfile Configを使って設定ファイルの設定値を環境変数で上書きできるようにする
 - MicroProfile Configについては以下を参照
 - <https://developer.mamezou-tech.com/msa/mp/cntrn06-mp-config/>
- ご自身でやってもらう方がいいと思いますがMicroProfileは目的外なので、ひな形リポジトリに回答例をアップしています
 - src/main/java/sample/HelloResource.java(変更)
 - src/main/resources/META-INF/microprofile-config.properties(追加)
 - src/test/java/sample/HelloResourceTest(変更)

Step2. コンテナをビルドしEC2からpullして動作を確認する(1/2)

- Step1で修正した内容をリポジトリにコミットする
- 3回目の「GitHub Packagesにコンテナイメージがデプロイする」の課題で作成したワークフロー (image-publish.yml)を使って、コンテナイメージをビルドしてGitHub Packages Container Registryに登録する



- イメージのビルドが上手くいっていることを確認できたらEC2へ

Step2. コンテナをビルドしEC2からpullして動作を確認する(2/2)

- EC2が停止している場合は起動しSSHでEC2に接続する
- 4回目資料のp.38-39を参考にビルドしたコンテナを起動し、REST APIを呼び出す
 - 環境変数は設定していないので、デフォルトの“Hello”が返ってくる
- コンテナを終了させ、次は環境変数を指定(-eオプション)してコンテナを起動し、設定ファイルの値を上書きする。起動したらREST APIを呼び出す
 - REST APIのレスポンスに環境変数で指定した値が返ってくればOK！
- コンテナを終了し、EC2は使わないので停止しておく

Step3. ECS Fargateにデプロイして動作を確認する

- ECS Fargateのデプロイでこれからやること
 - ① 事前準備
 - ・ タスク実行ロールの作成（CloudWatchへの書き込みの許可）
 - ② クラスターの作成
 - ③ タスク定義の作成
 - ・ タスク定義の作成
 - ・ ロググループの作成
 - ④ サービスの作成

Step3-①. 事前準備(1/4)

- ・ タスク実行ロールの作成（CloudWatchへの書き込みの許可）

IAMメニューから行う

The image shows a two-part screenshot of the AWS IAM console. The top part shows the search results for 'IAM' in the AWS Service Catalog. The bottom part shows the 'IAM > ロール' (IAM > Roles) page with a list of roles and a 'ロールを作成' (Create Role) button.

①IAMを入力して検索

②クリック

③ロールをクリック

④作成をクリック

Identity and Access Management (IAM)

検索 IAM の検索

ダッシュボード

▼ アクセス管理

- ユーザーグループ
- ユーザー
- ロール**
- ポリシー

IAM > ロール

ロール (20) 情報

IAM ロールは、短期間有効な認証情報を持つアクセス権を持つアカウント作成できるアイデンティティです。信頼するエンティティにロールを委任することもできます。

検索

ロール名	信頼された
<input type="checkbox"/> aws-ec2-spot-fleet-tagging-role	AWS のサー...
<input type="checkbox"/> aws-identity-providers-federation-github-actions	ID プロバイダー: arn:aws:iam::089528693478:oidc-provide
<input type="checkbox"/> AWSServiceRoleForAmazonElasticFileSystem	AWS のサービス: elasticfilesystem (サービスにリンクされ

Step3-①. 事前準備(2/4)

・タスク実行ロールの作成（CloudWatchへの書き込みの許可）

ステップ 1
信頼されたエンティティを選択

ステップ 2
許可を追加

ステップ 3
名前、確認、および作成

信頼されたエンティティを選択 [情報](#)

信頼されたエンティティタイプ



AWS のサービス

EC2、Lambda、その他の AWS サービスが、このアカウントでアクションを実行することを許可します。



AWS アカウント

お客様またはサードパーティーに属する他の AWS アカウントのエンティティが、このアカウントでアクションを実行することを許可します。



ウェブアイデンティティ

指定された外部ウェブアイデンティティプロバイダーによってフェデレーションされたユーザーが、このロールを引き受け、このアカウントでアクションを実行することを許可します。



SAML 2.0 フェデレーション

会社のディレクトリから SAML 2.0 を使用してフェデレーションされたユーザーが、このアカウントでアクションを実行することを許可します。



カスタム信頼ポリシー

カスタム信頼ポリシーを作成して、他のユーザーがこのアカウントでアクションを実行できるようにします。

①これを選択

ユースケース

EC2、Lambda、その他の AWS のサービスがこのアカウントでアクションを実行することを許可します。

一般的なユースケース



EC2

Allows EC2 instances to call AWS services on your behalf.



Lambda

Allows Lambda functions to call AWS services on your behalf.

他の AWS のサービスのユースケース:

Elastic Container Service



Elastic Container Service

Allows ECS to create and manage AWS resources on your behalf.



Elastic Container Service Autoscale

Allows Auto Scaling to access and update ECS services.



Elastic Container Service Task

Allows ECS tasks to call AWS services on your behalf.



EC2 Role for Elastic Container Service

Allows EC2 instances in an ECS cluster to access ECS.

③ Elastic Container Service Taskを選択

④次へ

キャンセル

次へ

Step3-①. 事前準備(3/4)

・ タスク実行ロールの作成（CloudWatchへの書き込みの許可）

ステップ 1
信頼されたエンティティを選択

ステップ 2
許可を追加

ステップ 3
名前、確認、および作成

許可を追加 情報

許可ポリシー (875) 情報

新しいロールにアタッチする 1 つ以上のポリシーを選択します。

検索: ECS

① ECSを入力してEnterキーで絞り込み実行

ステップ 1
信頼されたエンティティを選択

ステップ 2
許可を追加

ステップ 3
名前、確認、および作成

ECSが付くポリシーに絞り込まれる

許可を追加 情報

許可ポリシー (選択済み 1/875) 情報

新しいロールにアタッチする 1 つ以上のポリシーを選択します。

検索: ポリシーをプロパティまたはポリシー名でフィルタし、Enterキーを押します。 7一致 < 1 > ⚙

② AmazonECSTaskExecutionRolePolicyをチェック

	ポリシー名	タイプ	説明
<input type="checkbox"/>	ecsFargateE...	カスタ...	ecsExecTask in Fargate
<input type="checkbox"/>	AmazonEC...	AWS 管理	Provides administrative access to Amazon ECS r...
<input checked="" type="checkbox"/>	AmazonECSTaskExecutionRolePolicy	AWS 管理	Provides access to other AWS service resources...
<input type="checkbox"/>	AWSCodeDeployRoleForECS	AWS 管理	Provides CodeDeploy service wide access to per...
<input type="checkbox"/>	AWSCodeDeployRoleForECSLimited	AWS 管理	Provides CodeDeploy service limited access to p...
<input type="checkbox"/>	AWSElasticBeanstalkRoleECS	AWS 管理	(Elastic Beanstalk operations role) Allows a multi...
<input type="checkbox"/>	AWSFaultInjectionSimulatorECSAcc...	AWS 管理	This policy grants the Fault Injectio

③次へ

許可の境界を設定 - オプション 情報

許可の境界を設定して、このロールに付与される許可の最大数を制御します。これは一般的な設定ではありませんが、許可の管理を他のユ...

キャンセル

前へ

次へ

Step3-①. 事前準備(3/4)

・タスク実行ロールの作成（CloudWatchへの書き込みの許可）

ステップ 1: 信頼されたエンティティを選択する

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "",  
6       "Effect": "Allow",  
7       "Principal": {  
8         "Service": [  
9           "ecs-tasks.amazonaws.com"  
10        ]  
11      },  
12      "Action": "sts:AssumeRole"  
13    }  
14  ]  
15 }
```

編集



①キャプチャは見切れているが、この上にロール名を入力するところがあるのでロール名を入力する。例はecsTaskExecutionRole2としている（※Role2の2は荻原さんの環境の都合なので本来はなくてOK）

ステップ 2: 許可を追加する

許可ポリシーの概要			
ポリシー名	タイプ	次としてアタッチ:	
AmazonECSTaskExecutionRolePolicy	AWS 管理	許可ポリシー	

編集

ここで設定したロールは「③タスク定義の作成」で利用する

タグ

タグを追加 - オプション 情報

タグは AWS リソースに追加できるキーと値のペアで、リソースの特定、整理、検索に役立ちます。

リソースに関連付けられたタグはありません。

タグを追加

最大 50 個のタグを追加できます。

②確認画面なので「ロールを作成」を押して完了



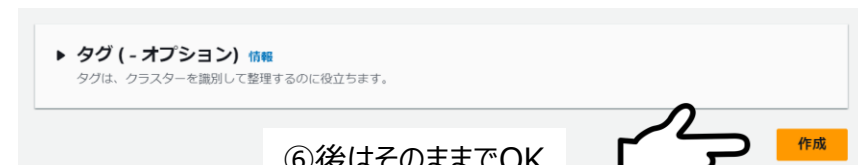
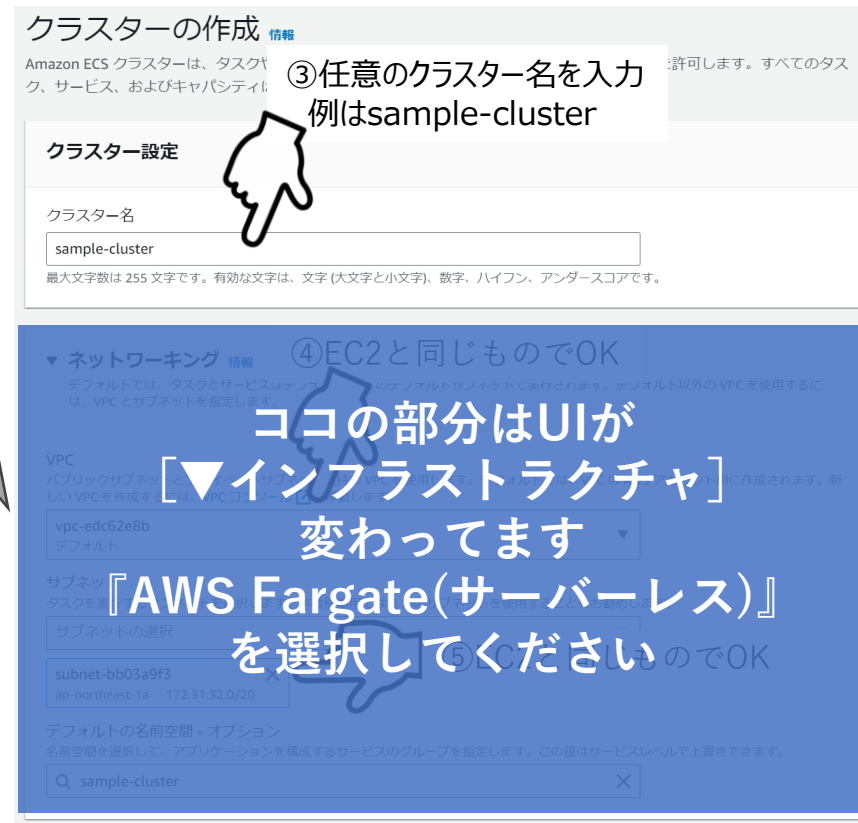
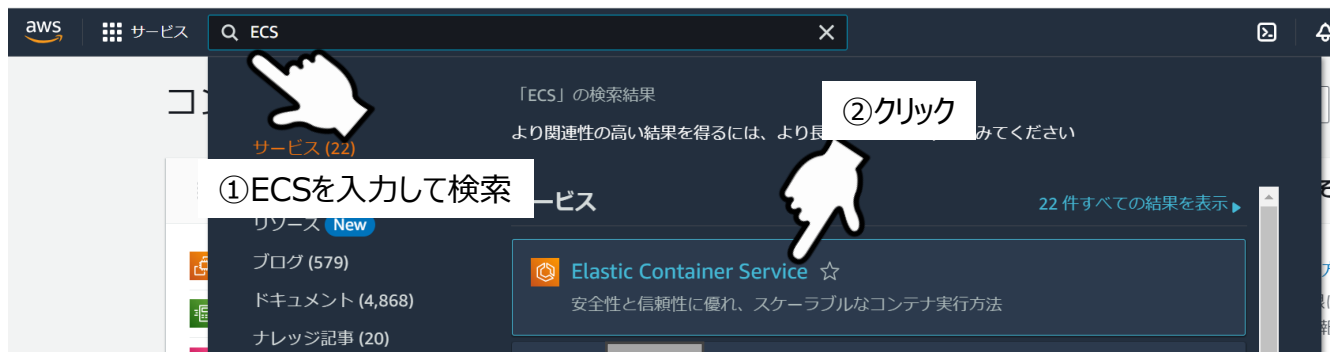
キャンセル

前へ

ロールを作成

Step3-②. クラスターの作成(1/2)

ECSメニューから行う



Step3-②. クラスターの作成(2/2)

🔄 クラスター sample-cluster の作成が進行中です。 CloudFormation で表示

Amazon Elastic Container Service > クラスター

クラスター (1) 情報

🔍 クラスターの検索

クラスター	サービス	タスク	登録済みコンテナインスタンス	CloudWatch モニタリ...	キャパシ...
rms-service-cluster	3	0 件が保留中 3 件が実...	0	🕒 デフォルト	デフォル

運が悪いと数分待つので待っていると

🟢 クラスター sample-cluster は正常に作成されました。

Amazon Elastic Container Service > クラスター

クラスター (2) 情報

🔍 クラスターの検索

クラスター	サービス	タスク	登録済みコンテナインスタンス	CloudWatch モニタリ...	キャパシ...
rms-service-cluster	3	0 件が保留中 3 件が実...	0	🕒 デフォルト	デフォル
sample-cluster	0	実行中のタスクなし	0	🕒 デフォルト	デフォル

これで作成完了

Step3-③. タスク定義の作成(1/2)

クラスター

名前空間 [新規](#)

タスク定義

アカウント設定 [新規](#)



AWSCLIをインストール

①タスク定義をクリック



②新しいタスク定義の作成をクリック

Step3-③. タスク定義の作成(2/2)

新しいタスク定義の作成 [情報](#)

タスク定義の設定

タスク定義ファミリー [情報](#)
一意のタスク定義ファミリー名を指定します。

最大 255 文字の英字 (大文字と小文字)、数字、ハイフン、アンダースコアを使用できます。

▼ **インフラストラクチャの要件**
Specify the infrastructure requirements for the task definition.

起動タイプ [情報](#)
起動タイプを選択すると、タスク定義パラメータが変更されます。

☒ **AWS Fargate**
コンテナのサーバーレスコンピューティング。

☐ **Amazon EC2 インスタンス**
Amazon EC2 インスタンスを使用したセルフマネージドインフラストラクチャ。

OS、アーキテクチャ、ネットワークモード
ネットワークモードはタスクに使用され、選択したコンピューティングタイプによって異なります。

オペレーティングシステム/アーキテクチャ [情報](#)

ネットワークモード [情報](#)

タスクサイズ [情報](#)
タスク用に予約する CPU とメモリの量を指定します。

CPU メモリ

▼ **条件付きのタスクロール**

タスクロール [情報](#)
タスク IAM ロールは、タスクのコンテナが AWS のサービスリソースにリクエストを行うことを許可します。IAM コンソール からタスク IAM ロールを作成できます。

タスク実行ロール [情報](#)
タスク実行 IAM ロールは、コンテナエージェントがユーザーに代わって AWS API リクエストを行うために使用されます。まだタスク実行 IAM ロールを作成していない場合は、当社で作成させていただきます。

①任意のタスク名を入力
例はsample-app-task2

②Fargateのみがチェックされていること

③.5vCPUと1GBを選択

④タスクロールはなし「-」

⑤タスク実行ロールは3-①で作成した
ロールを設定する

⑥任意の名前を入力
例はsample-app

⑦イメージURLを入力
GitHub Packagesのイメージ名を入力
タグはlatestでよい

コンテナ 1 [情報](#) 必須コンテナ 削除

名前 イメージ URI 必須コンテナ

プライベートレジストリ [情報](#)
Secrets Manager に認証情報を保存し、その認証情報を使用してプライベートレジストリのイメージを参照します。

☒ **プライベートレジ**

ポートマッピング [情報](#)
ポートマッピングは、コンテナがホストのポートにアクセスしてトラフィックを送受信できるようにします。ポートマッピング設定を変更すると、関連するサービスに障害を与えます。

コンテナポート プロトコル ポート名 アプリケーションプロトコル 削除

[他のポートマッピングの追加](#)

⑧コンテナポートに7001を指定

⑨展開して確認

▼ **ログ記録 - オプション**

サイドカーの CPU とメモリの割り当て
タスク定義に既存のサイドカーがない場合、サイドカーを自動的に追加するログ記録のオプションがあります。AWS は、選択したオプションに基づいて CPU とメモリの調整に関するレコメンデーションを提供しています。

AWS Fargate で実行されるタスクには、ログ収集を使用することをお勧めします。ログ収集の詳細はこちら。

☒ **ログ収集の使用** [情報](#)
デフォルト設定を使用して、コンテナログをログ記録の送信先に送信するようにタスクを設定します。Amazon CloudWatch の料金に関する情報を参照してください。

キー	値のタイプ	値
awslogs-group	<input type="text" value="値"/>	<input type="text" value="/ecs/sample-app-task2"/>
awslogs-region	<input type="text" value="値"/>	<input type="text" value="ap-northeast-1"/>
awslogs-stream-prefix	<input type="text" value="値"/>	<input type="text" value="ecs"/>
awslogs-create-group	<input type="text" value="値"/>	<input type="text" value="true"/> 削除

[パラメータを追加](#)

⑩同じような感じになっていることを確認

⑪あとはデフォルトのままでよいので
ここまでの入力と確認ができれば
「作成」ボタンをクリックしてタスク定義の作成は完了！

Step3-④. サービスの作成(1/6)

クラスター

名前空間 [新規](#)

タスク定義

アカウント設定 [新規](#)

①クラスターをクリック

[AWS Copilot をインストール](#)

[Amazon ECR](#)

リポジトリ

[Amazon Elastic Container Service](#) > クラスター

クラスター (2) 情報

Q クラスターの検索

②Step3-②で作成したクラスターをクリック

クラスター

[rms-service-cluster](#) 3

[sample-cluster](#) 0

0 件が

実行中のタスクが

クラスター

名前空間 [新規](#)

タスク定義

アカウント設定 [新規](#)

[AWS Copilot をインストール](#)

[Amazon ECR](#)

リポジトリ

[AWS Batch](#)

[AWS Proton](#)

[ドキュメント](#)

[製品の検出](#)

[サブスクリプション](#)

[Amazon Elastic Container Service](#) > [クラスター](#) > [sample-cluster](#) > サービス

sample-cluster

クラスターの概要

ARN
sample-cluster

ステータス
アクティブ

CloudWatch モニタリング
デフォルト

登録済みコンテナインスタンス
-

サービス

タスク

ドレイン中

保留中

アクティブ

実行中

④作成をクリック

サービス

タスク

インフラストラクチャ

メトリクス

スケジュールされたタスク

タグ

サービス (0) 情報

サービスをフィルター

すべての起動タイプ

すべてのサービスタイプ

③サービスタグをクリック

サービスがありません

表示するサービスがありません。

作成

Step3-④. サービスの作成(2/6)

Amazon Elastic Container Service > クラスター > sample-cluster > サービスの作成

作成 情報

環境

AWS Fargate

既存のクラスター

既存のクラスターを選択します。新しいクラスターを作成するには、[クラスター](#) にアクセスしてください。

sample-cluster

▼ コンピューティング設定 (アドバンスド)

コンピューティングオプション 情報

コンピューティングタイプ間でタスクを分散させるには、適切なコンピューティングタイプを選択します。

☐ キャパシティープロバイダー戦略
1つ以上のキャパシティープロバイダーにタスクを分散する起動戦略を指定します。

☒ 起動タイプ
キャパシティープロバイダー戦略を使用せずに、タスクを直接起動します。

起動タイプ 情報

マネージドキャパシティープロバイダーを選択します。External Instance Profile (EIP) を使用してクラスターに登録されます。

FARGATE

プラットフォームのバージョンを選択します。最新のバージョンを選択します。

LATEST

①こっちが選択されていること

②FARGATEが選択されていること

③LATESTが選択されていること

デプロイ設定

④こっちが選択されていること

アプリケーションタイプ 情報

実行するアプリケーションのタイプを指定します。

☒ サービス

停止して再起動できる長時間のコンピューティング作業を処理するタスクグループを起動します。例としては、ウェブアプリケーションが挙げられます。

☐ タスク

実行および終了するスタンドアロンタスクを起動します。例としては、バッチジョブが挙げられます。

タスク定義

既存のタスク定義を選択します。

⑤Step3-③で作成したタスク定義を選択

☐ リビジョンの手動指定

選択したタスク定義ファミリーに100個のリビジョンを選択する代わりに、リビジョンを手動で入力します。

ファミリー

sample-app-task2

リビジョン

1 (最新)

サービス名

このサービスに一意の名前を割り当てます。

sample-app-service

⑥任意のサービス名を入力
例はsamample-app-service

サービスタイプ 情報

サービススケジューラが従うサービスタイプを指定します。

☒ レプリカ

クラスター全体に必要な数のタスクを配置して維持します。

☐ デモン

各コンテナインスタンシアを1つ配置します。

⑦必要なタスクに1を入力

必要なタスク

起動するタスクの数を指定します。

1

▶ デプロイオプション

▶ デプロイ不具合の検出 情報

Step3-④. サービスの作成(3/6)

▼ ネットワーキング

VPC [情報](#)

使用する Virtual Private Cloud を選択します。

vpc-edc62e8b
デフォルト

① クラスターで選択したものと同一のもの
つまり、EC2と同じVPCを選択

サブネット

タスクスケジューラが配置するために考慮する VPC サブネットを選択します。

サブネットの選択

② EC2と同じサブネットを選択

subnet-bb03a9f3
ap-northeast-1a 172.31.32.0/20

セキュリティグループ [情報](#)

既存のセキュリティグループを選択するか、新しいセキュリティグループを作成します。

- ☒ 既存のセキュリティグループを使用
- ☐ 新しいセキュリティグループの作成

③ 既存を選択

セキュリティグループ名

既存の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選択

④ EC2と同じセキュリティグループを選択

sg-0e43a398391ee41be
launch-wizard-4

パブリック IP [情報](#)

タスクの Elastic Network Interface にパブリック IP を自動的に割り当てるかどうかを選択します。

- ☒ オンになっています

⑤ オンになっていること

▶ ロードバランシング - オプション

▶ サービスの Auto Scaling - オプション

CloudWatch アラームに応じてサービスの必要数を指定範囲内で調整します。アプリケーションのニーズに合わせていつでも Service Auto Scaling の設定を変更できます。

▶ タグ (- オプション) [情報](#)

タグは、リソースを識別して整理するのに役立ちます。

キャンセル

作成

⑥ 後はそのままで作成を実行

Step3-④. サービスの作成(4/6)

sample-app-service のデプロイが進行中です。これには数分かかります。 CloudFormation を表示

Amazon Elastic Container Service > クラスター > sample-cluster > サービス

sample-cluster

クラスターの概要

ARN sample-cluster	ステータス アクティブ	CloudWatch モニタリング デフォルト	登録済みコンテナインスタンス -
サービス ドレイン中	アクティブ	タスク 保留中	実行中 -

進行中と出ていてもタグをクリックしたり、最新に更新を押していると↓のようなサービスの内容が出てくる

サービス | タスク | インフラストラクチャ | メトリクス | スケジュールされたタスク | タグ

サービス (1) 情報

値でサービスをフィルター 全ての起動タイプ 全てのサービスタイプ

サービス名	ステータス	ARN	サービスタイプ	デプロイとタスク	前回のデプロイ	タスク	リビジョン
sample-app-service	アクティブ	arn:aws:ecs...	REPLICA	1/1 件の実...	進行中	sample-a...	1

リンクをクリックでサービスの詳細へ

Step3-④. サービスの作成(5/6)

サービスの詳細

Amazon Elastic Container Service > クラスター > sample-cluster > サービス > sample-app-service > イベント

sample-app-service 情報

サービスを更新 サービスを削除

正常性とメトリクス タスク ログ デプロイ イベント 設定 ネットワーキング タグ

イベント (1)

値でイベントをフィルター

開始時刻 ▼ | メッセージ ▼ | イベント ID ▼

2023年8月22日 22:15 (UTC+9:00) service sample-app-service has started 1 tasks; task [744e8d74c30049e19ef35dbaa8c421aa](#).

744e8d74c30049e19ef35dbaa8c421aa

設定 ログ ネットワーキング タグ

タスク

ARN 744e8d74c30049e19ef35dbaa8c421aa 前回のステータス 実行中

外部からはこのアドレスでアクセスする 7.333Z 2023-08-22 15:15:00.686Z

アプリのログはここからCloudWatch経由で見れる

設定

オペレーティングシステム/アーキテクチャ Linux/X86_64	キャパシティプロバイダー -	ENI ID eni-0069581f956b39c99	パブリック IP 43.207.122.142 オープンアドレス
CPU メモリ .5 vCPU 1 GB	起動タイプ FARGATE	ネットワークモード awsipc	プライベート IP 172.31.40.137
プラットフォームのバージョン 1.4.0	タスク定義: 修正 sample-app-task2:1	サブネット ID subnet-bb03a9f3	MAC アドレス 06:87:38:32:de:d1

上手いかないときはここ（イベントログ）を確認

タスクの詳細画面へ

Step3-④. サービスの作成(6/6)

正常起動の確認

[Amazon Elastic Container Service](#) > [クラスター](#) > [sample-cluster](#) > サービス

sample-cluster

クラスターの概要

ARN
sample-cluster

ステータス
アクティブ

CloudWatch モニタリング
デフォルト

サービス

タスク

ドレイン中
-

アクティブ
1

保留中
-

起動できたら前のページ確認したアドレスで
REST APIが呼び出せるか確認してみる。
正常に応答が返ってくればOK!

サービス | タスク | インフラストラクチャ | メトリクス | スケジュールされたタスク | タグ

サービス (1) 情報

必要なタスク分（つまり1つ）が起動して緑状態になったらOK！

値でサービスをフィルター

すべての起動タイプ

すべてのサービスタイプ

<input type="checkbox"/>	サービス名	ステータス	ARN	サービスタイプ	デプロイとタスク
<input type="checkbox"/>	sample-app-service	アクティブ	arn:aws:ecs...	REPLICA	1/1 件の実...

タスクの停止

重要

ECS Fargateには無料枠は付いていません
ECS FargateはEC2と同様にタスクが起動している時間で課金されます
ですので、使い終わったらタスクを停止するようにしてください。課金が停止します
タスクの停止は「必要なタスク」を0にしてサービスを更新だけです

sample-cluster

クラスタの概要

ARN sample-cluster	ステータス アクティブ	CloudWatch モニタリング デフォルト	登録済みコンテナインスタンス -
サービス ドレイン中 -	アクティブ 1	タスク 保留中 -	

②更新をクリック

サービス | タスク | インフラストラクチャ | メトリクス | スケジュールされたタスク | タグ

①対象のサービスをチェック

サービス (1/1) 情報

サービス名	ステータス	ARN	サービスタイプ	デプロイとタスク
sample-app-service	アクティブ	arn:aws:ecs...	REPLICA	1/1 件の実行中

sample-app-service を更新 情報

デプロイ

①ここをチェック

☒ 新しいデプロイの強制

タスク定義
既存のタスク定義を選択します。新しいタスク定義を作成するには、[タスク定義](#) にアクセスしてください。

☐ リビジョンの手動指定
選択したタスク定義ファミリーに最新の 100 個のリビジョンを選択する代わりに、リビジョンを手動で入力します。

ファミリー
sample-app-task2

リビジョン
2 (最新)

サービスタイプ
REPLICA

必要なタスク
起動するタスクの数を指定します

②0にする

0

最小実行タスク % 情報 最大実行タスク % 情報

③この状態で更新を実行する
実行後は0/0件が実行中になり、タスクが起動していないことを確認する

Step4. 今回の課題

- Step2のEC2でやったようにFargateでも環境変数を設定し、環境変数に従った、応答が返るようにしましょう
- ヒント
 - どのコンテナをどのように動かすかを定義してるのはタスク定義
 - タスク定義は世代で管理される
 - 新しいタスク定義で動作させるのはサービスで参照するタスク定義のリビジョンを更新する
 - サービスの実体はコントロールプレーン（コンテナオーケストレーター）でサービスに指定された状態を保とうとする
 - タスクは起動するたびに割り当てられるグローバルIPが変わるので、REST APIを呼び出す際は都度、IPアドレスを確認すること

次回の予定

次回は最終回だよ



次回の予定

- 今回の課題の説明
 - 補足説明と質疑応答