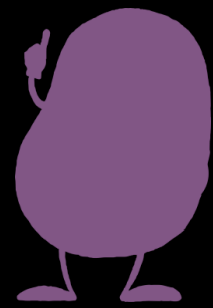
AWSとGitHubを使ってみよう勉強会

~ 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



updated:2023/9/10

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

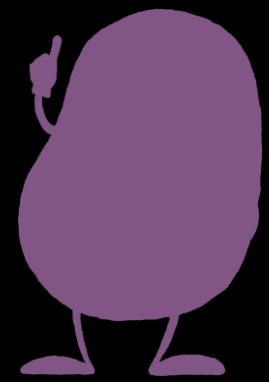
前回の課題の解説

・これでまでの課題でやってきた全体像

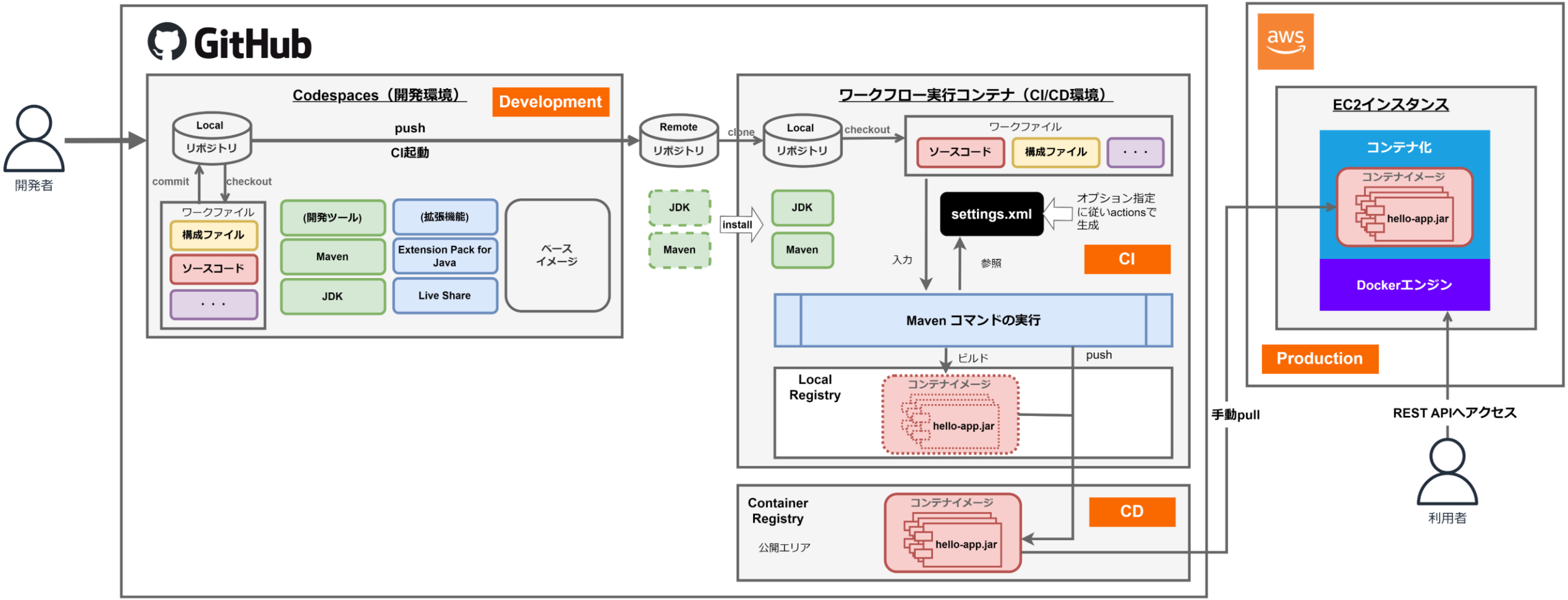
・AWSのユーザと権限

・AWSのネットワーク構成

・最後にEC2は



これでまでの課題でやってきた全体像 – 振り返り



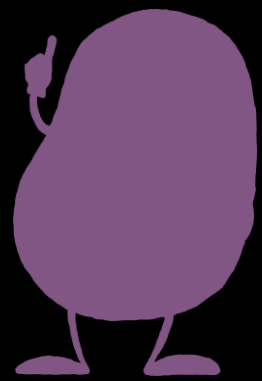
- ✓ 開発からCI/CD、プロダクションでの実行まですべてクラウド環境で実現！
- ✓ プロダクションへのデプロイは手動だがコミットからリポジトリの登録まではすべて自動！

MAMEZOU-TECH

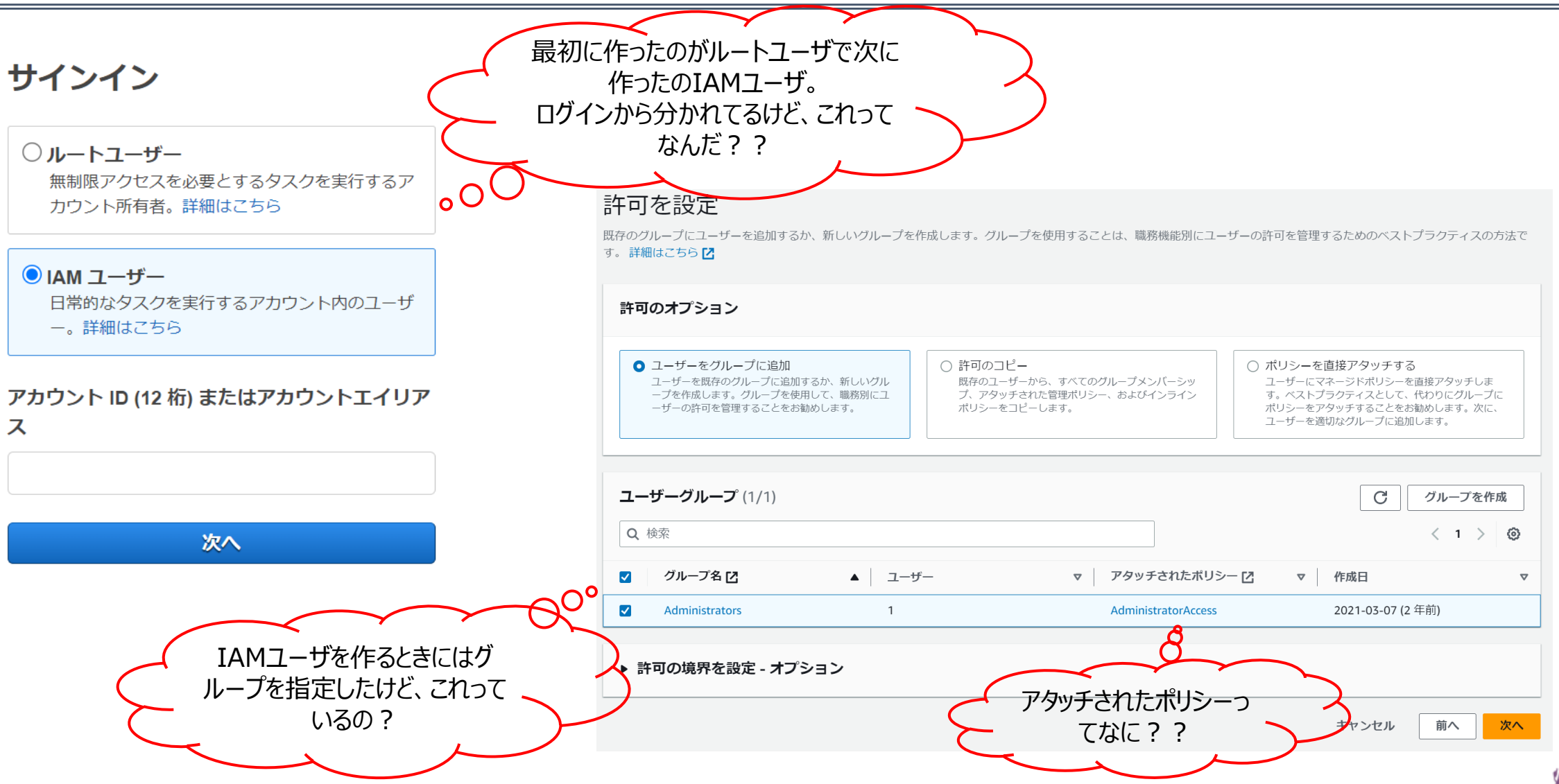
前回の課題の解説

・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訳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



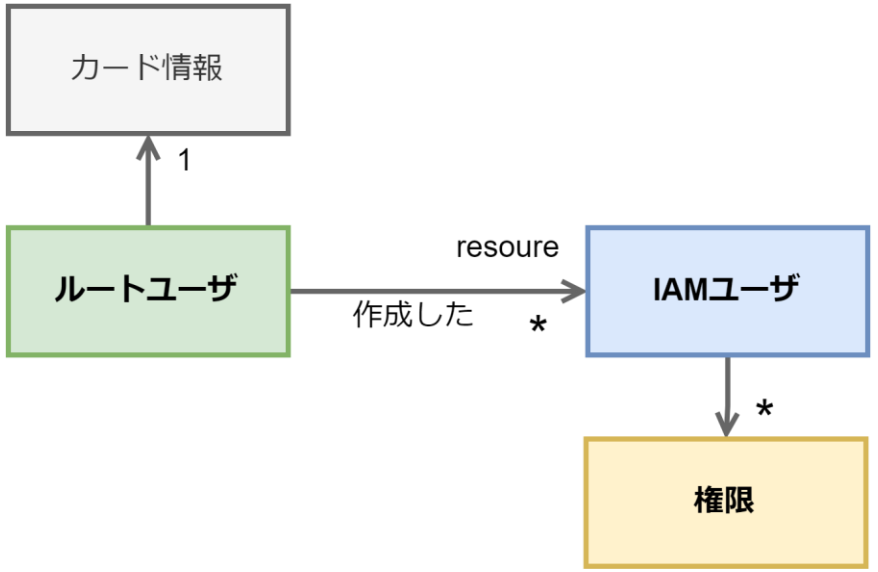
ルートユーザとIAMユーザ



MAMEZOU-TECH

ルートユーザとIAMユーザ

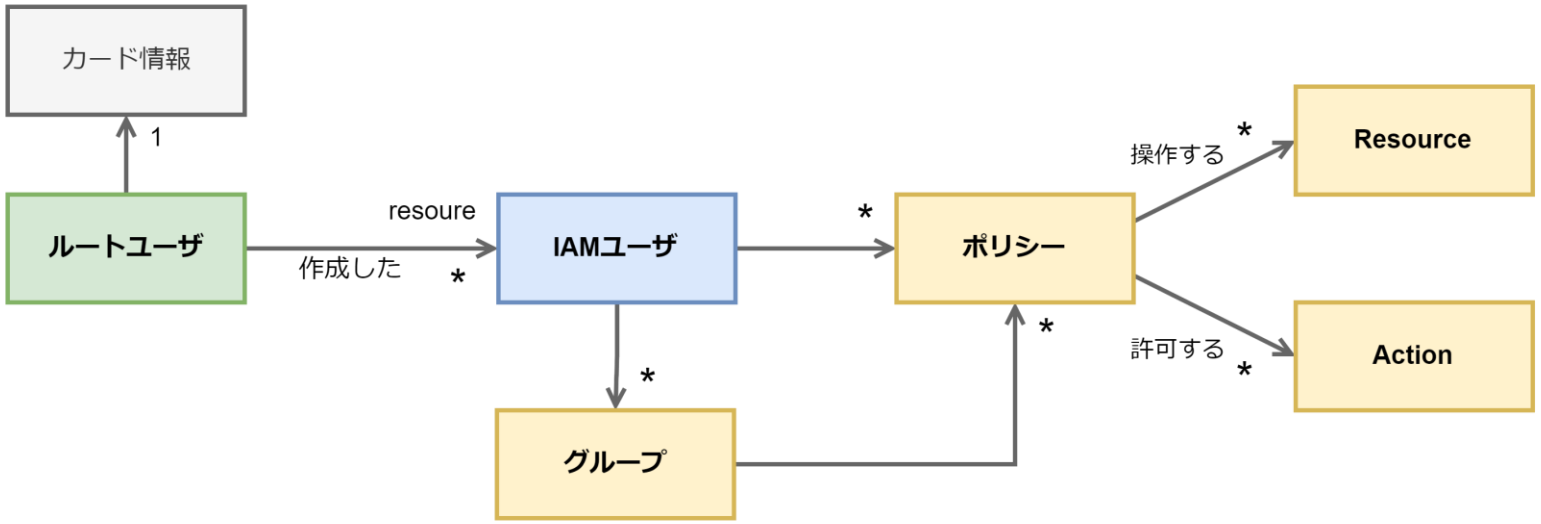
- ルートユーザ
  - AWSアカウント作成時に最初に用意されアカウント
  - 予めすべての権限が付与されてる。これを制限する方法はないので、非常に権限の強いユーザとなる
  - AWSからの請求につかうクレジットカード情報が紐づいている
- IAMとは
  - ID と AWS のサービスおよびリソースへのアクセスを一元管理するAWS独自の権限管理の仕組み
  - 日本では「アイアム」、英語圏では「アイエーエム」と呼ばれることが多い



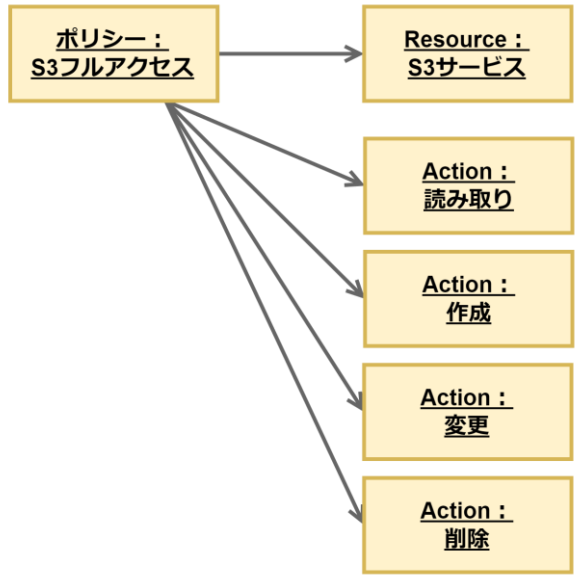
- IAMユーザとは
  - IAMで作成したユーザ。ルートユーザは最初からフルアクセス権が付ついているのに対して、IAMユーザで何をするには必要な権限を付与する必要がある
  - AWSのユーザの種類にはルートユーザとIAMユーザの2種類しかなく、IAMユーザはLinux等でよく言われる「ルートユーザ」に対する「一般ユーザ」の意味に等しい

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

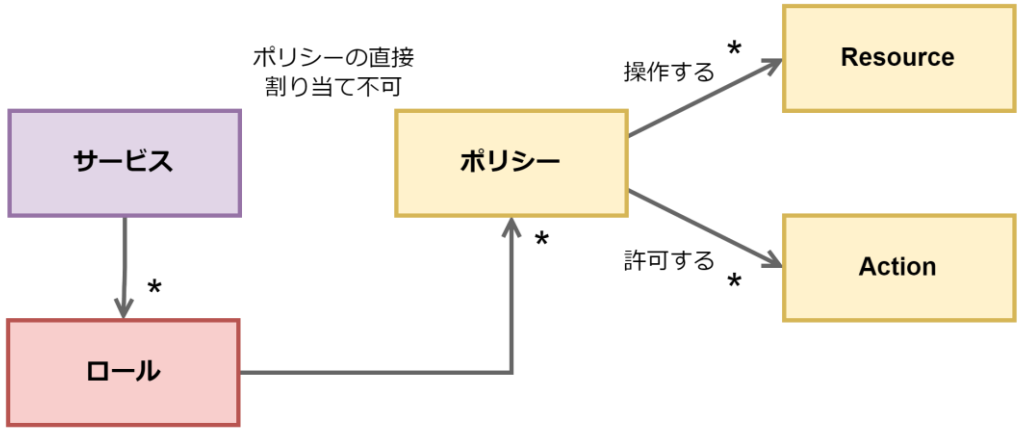
ルートユーザとIAMユーザは別物なんだね！
IAMユーザってヘンテコな名前だけど「一般ユーザ」ということか

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！

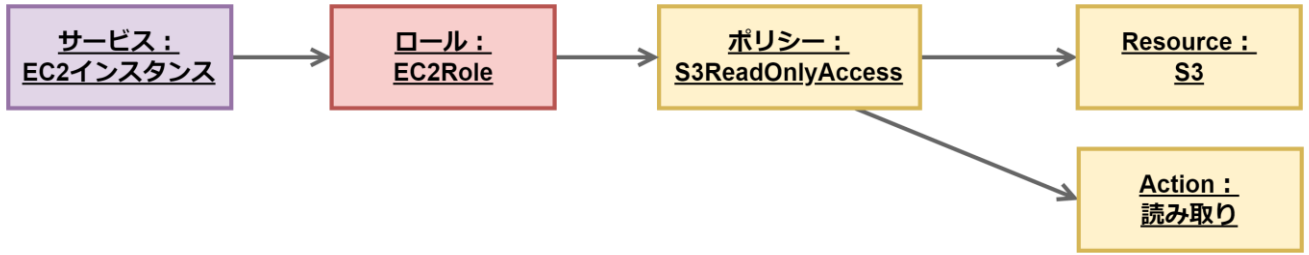
これはグループを経由して割り当てられる権限なんだネ

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



ロールを使ったのポリシーの設定例



- 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

MAMEZOU-TECH

まとめ - AWSのユーザと権限

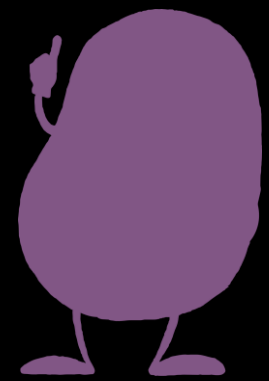
- AWSの権限周りは難しく感じるが実はネーミングがヘンなだけ

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）

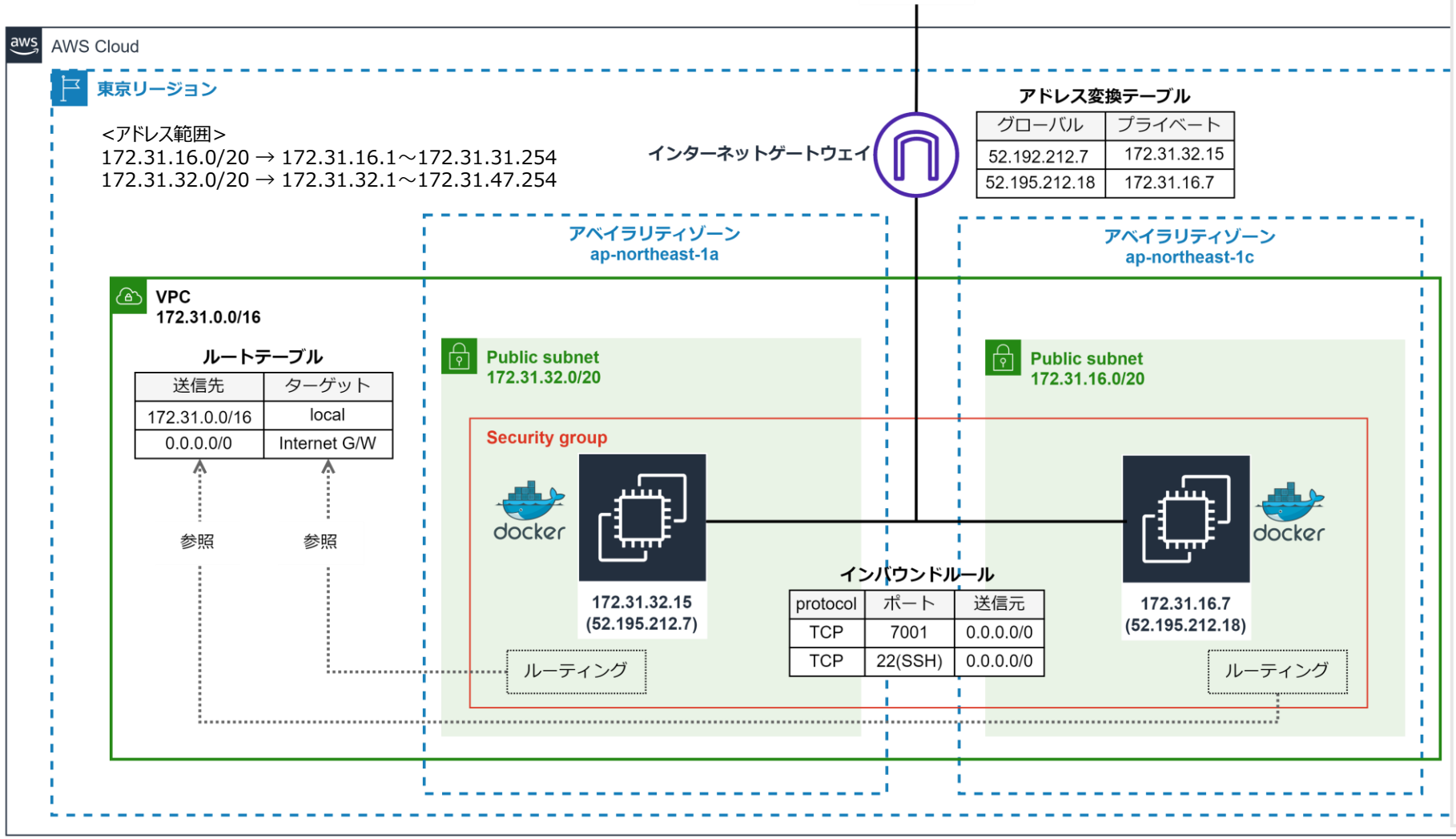
前回の課題の解説

・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



■ 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

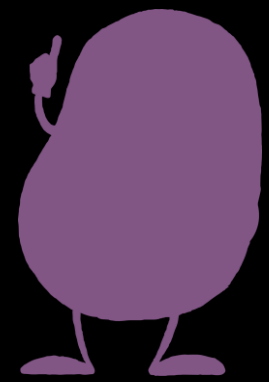
■ セキュリティグループ
- EC2などのサービスはいずれかのセキュリティグループに紐づけられており、そのセキュリティグループを通してアクセスされる。
- このセキュリティグループは実質的にfirewallの役割となる。よって、外部からのアクセスを許可するにはセキュリティグループでポートを開ける必要がある

■ 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に直接繋げられる場合、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

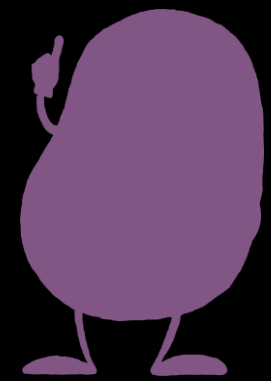
・これでまでの課題でやってきた全体像
・AWSのユーザと権限
・AWSのネットワーク構成
・最後にEC2は



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
- クラウドならではや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 Packges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 Fagate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

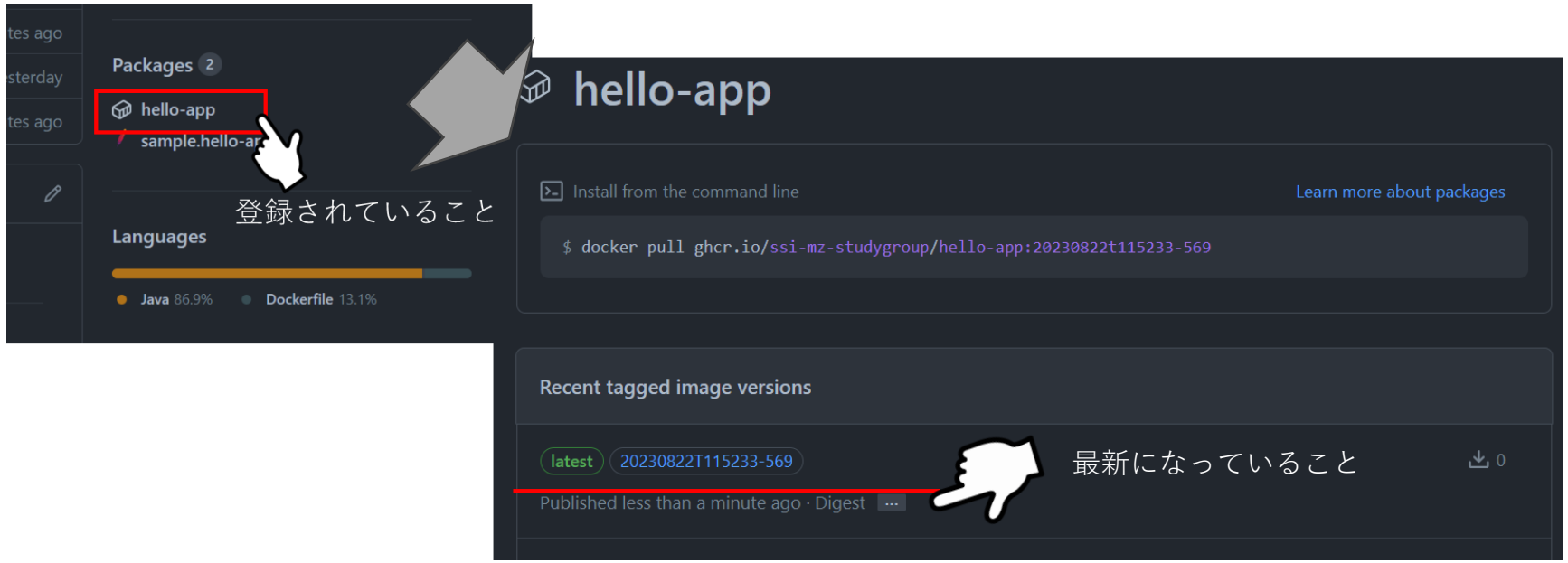
- 変更内容
  - 環境変数CONFIG_VALの設定が
    - ある場合は、その設定値をREST APIで返す
    - ない場合は、デフォルト値として“Hello"

- 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

MAMEZOU-TECH

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）

MAMEZOU-TECH

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）

MAMEZOU-TECH

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　編集

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ecs-tasks.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

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　編集

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

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。

キャンセル　前へ　ロールを作成

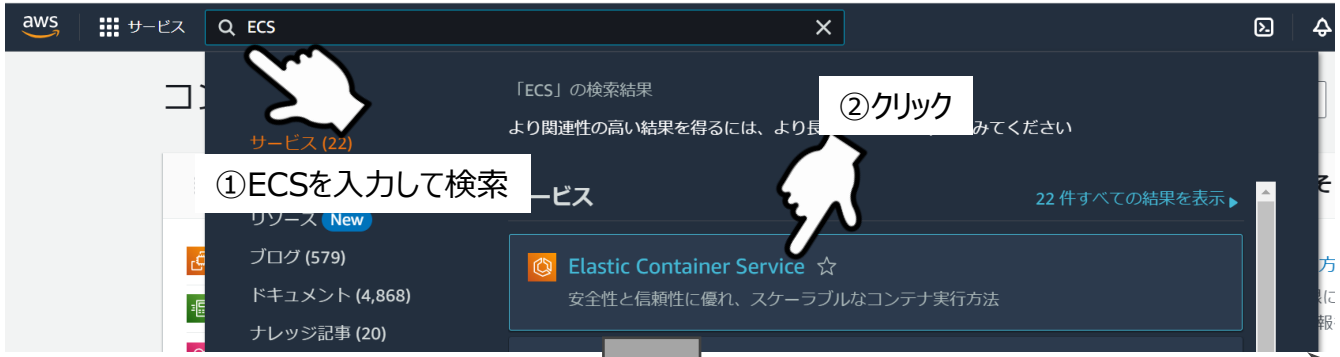
ここで設定したロールは「③タスク定義の作成」で利用する

②確認画面なので「ロールを作成」を押して完了

MAMEZOU-TECH

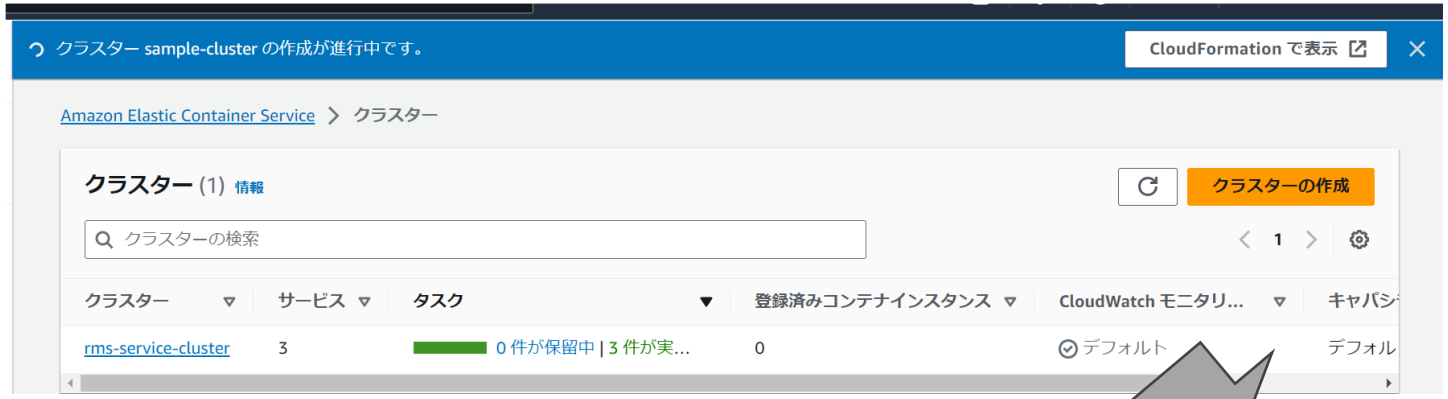
Step3-②. クラスターの作成(1/2)

ECSメニューから行う



MAMEZOU-TECH

Step3-②. クラスターの作成(2/2)

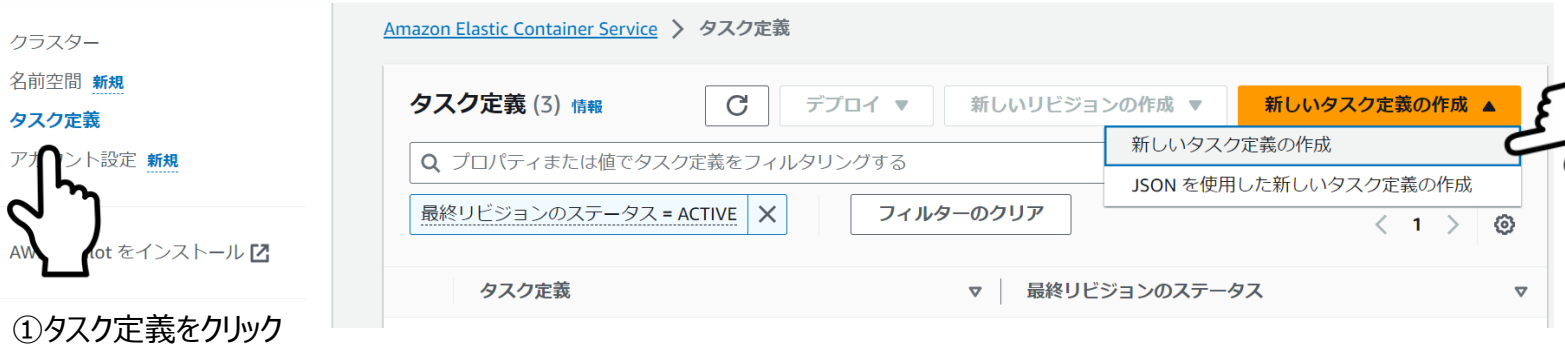


運が悪いと数分待つので待っていると

これで作成完了

MAMEZOU-TECH

Step3-③. タスク定義の作成(1/2)



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

MAMEZOU-TECH

Step3-③. タスク定義の作成(2/2)

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

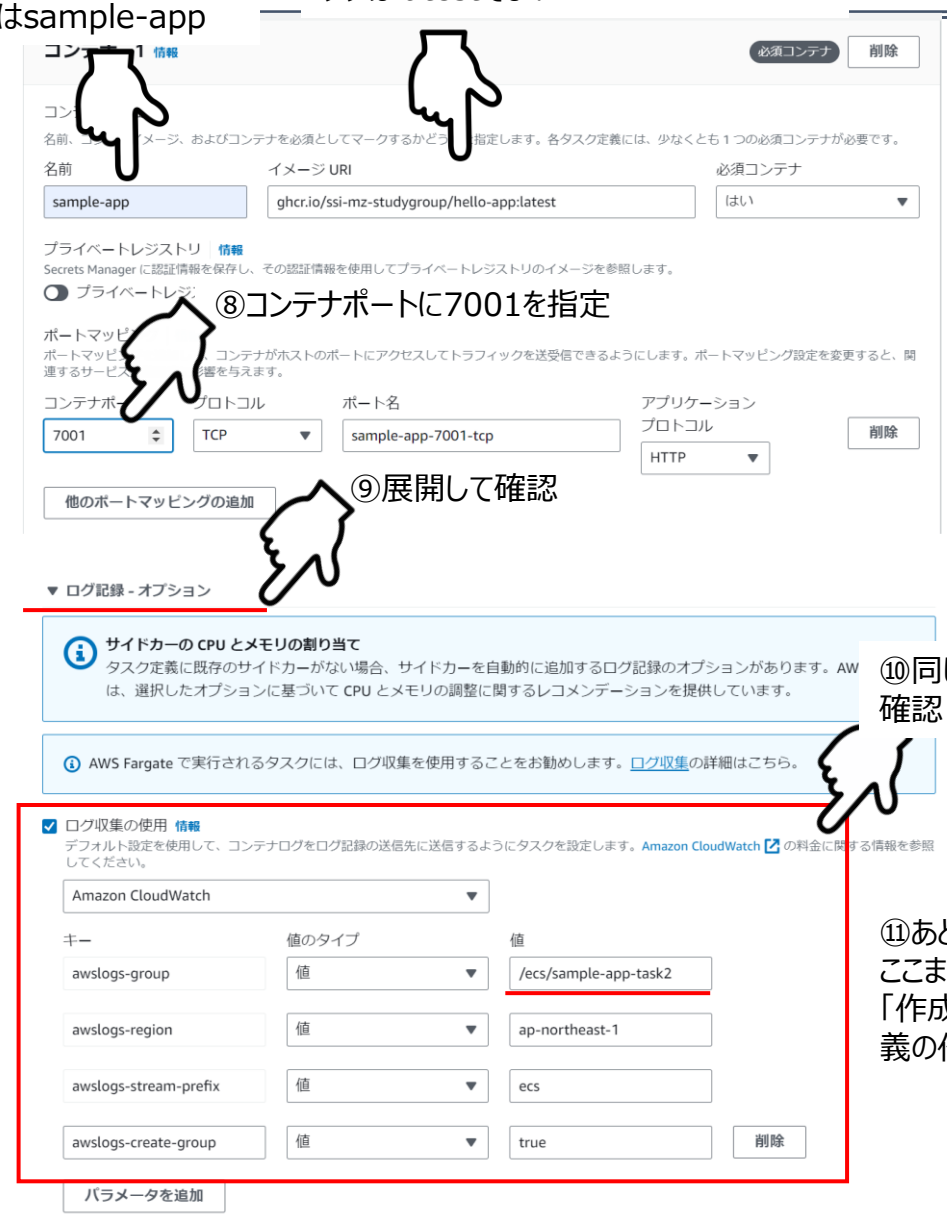
③.5vCPUと1GBを選択

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



⑧コンテナポートに7001を指定

⑨展開して確認

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たら「作成」ボタンをクリックしてタスク定義の作成は完了！

MAMEZOU-TECH

Step3-④. サービスの作成(1/6)

Step3-④. サービスの作成(2/6)

MAMEZOU-TECH

Step3-④. サービスの作成(3/6)

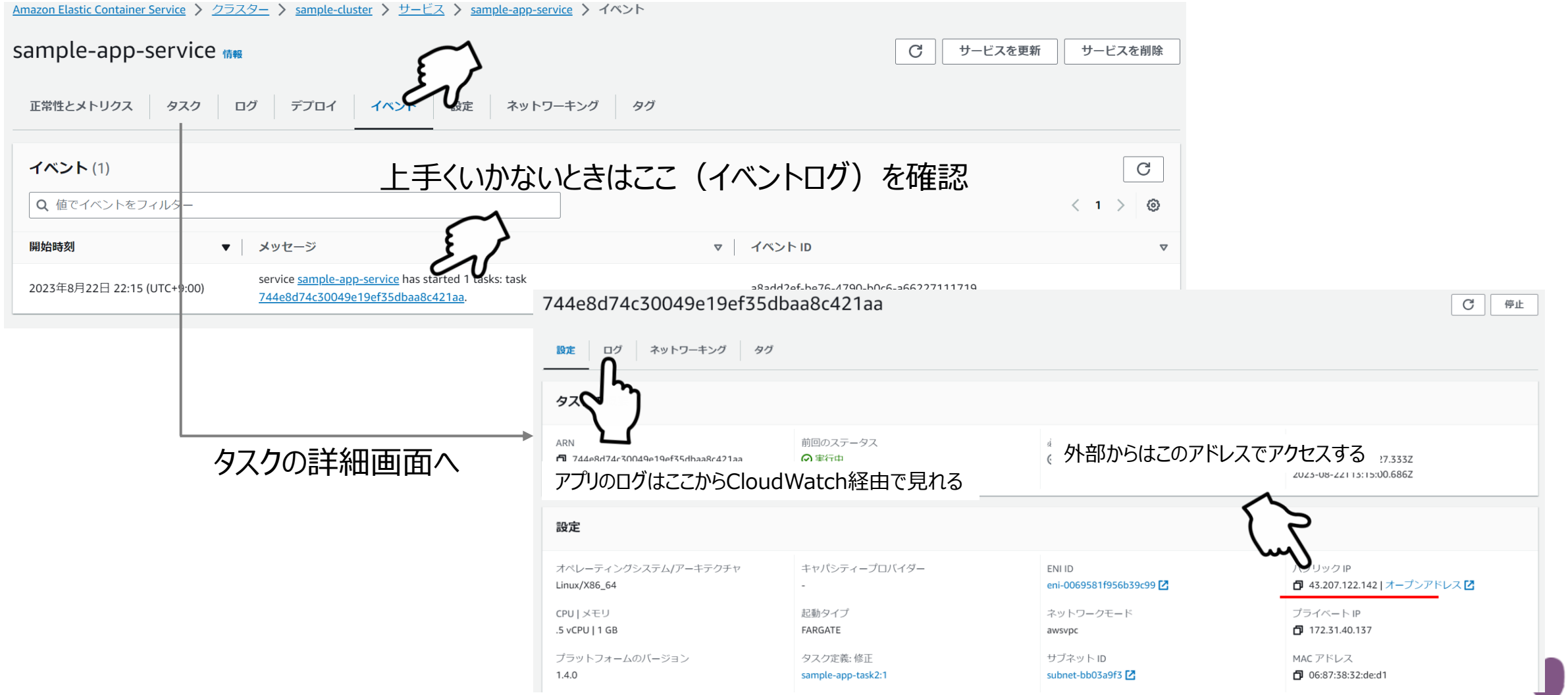
MAMEZOU-TECH

Step3-④. サービスの作成(4/6)

MAMEZOU-TECH

Step3-④. サービスの作成(5/6)

サービスの詳細



MAMEZOU-TECH

Step3-④. サービスの作成(6/6)

正常起動の確認

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!

MAMEZOU-TECH

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するだけです

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、タスクが起動してないことを確認する

MAMEZOU-TECH

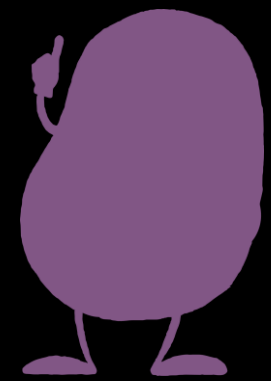
Step4. 今回の課題

- 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