

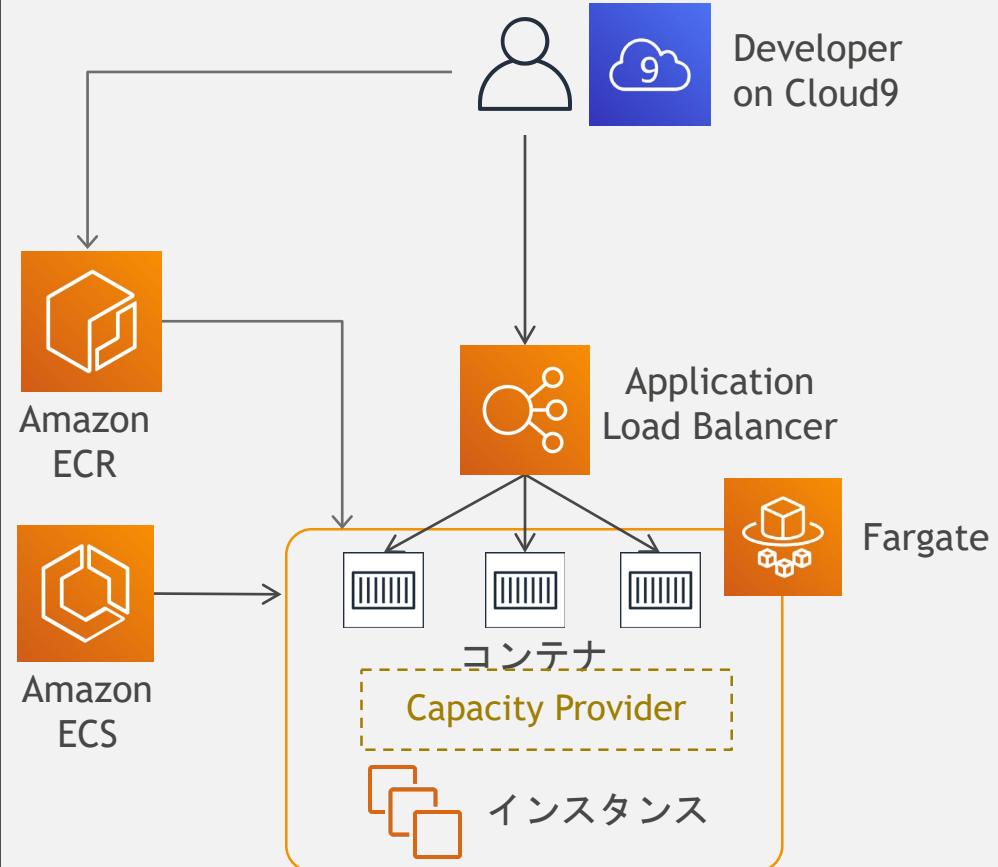
ECS Fargate

アマゾンウェブサービスジャパン株式会社

ハンズオン

AWSでの Docker 管理環境の構築方法

- AWS Fargate を使ったコンテナ運用

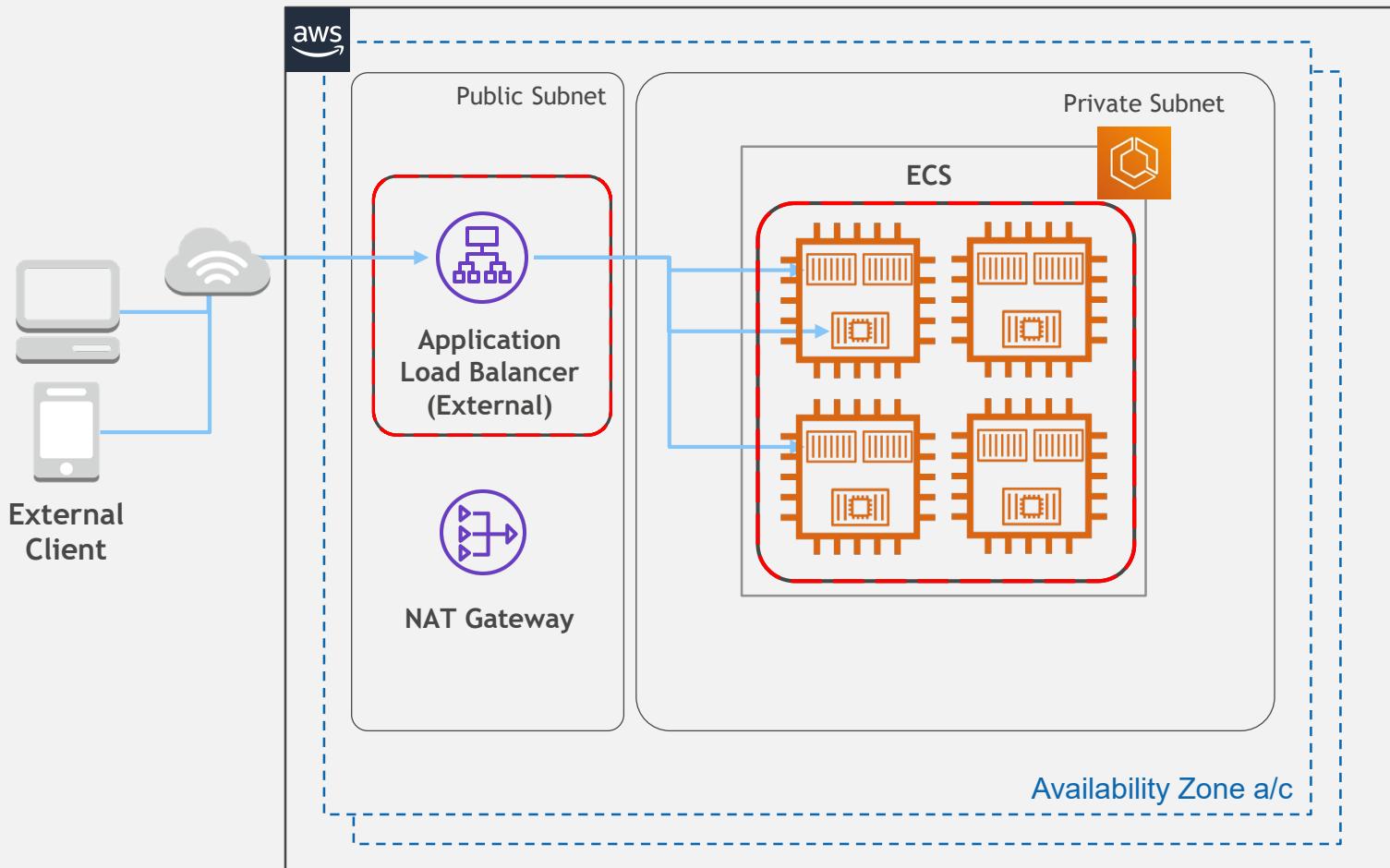


ハンズオンの手順

1. VPC作成
2. ALB作成
3. ECSクラスター作成
4. Dockerアプリ構築
 - ECRリポジトリ作成
 - Cloud9環境の構築
5. ECSタスク定義・サービス作成

VPC作成

本ハンズオンで構築する構成図

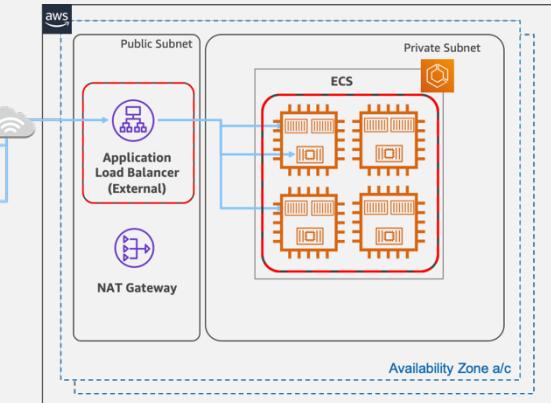


ハンズオンはすべて
東京リージョン
で実施します

CloudFormationの実行手順の説明 (1)

本日のハンズオンに必要となるネットワークを構築する。

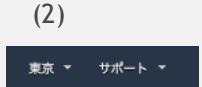
- 主題の ECS/Cloud9 に集中するため、今回は CloudFormation で以下を自動構築する。
 - VPC [10.1.0.0/16]
 - Internet Gateway & EIP
 - NAT Gateway & EIP * 2
 - Subnet
 - Public Subnet [10.1.0.0/24, 10.1.2.0/24]
 - Private Subnet [10.1.1.0/24, 10.1.3.0/24]
 - Route Table



CloudFormationの実行手順の説明 (2)

GUI の場合

1. マネージメントコンソールを開く
2. 東京リージョンを選択する
3. Services -> CloudFormation を開く
4. 「スタックの作成」を押す
5. テンプレートソースでS3 URL を入力し、「次へ」を押す
<https://s3-ap-northeast-1.amazonaws.com/tarohiro-temp/20181002-ecs-cloud9-cicd-handson/assets/cloudformation.yml>
6. スタックの名前を "vpc-for-handson-cicd-xx" と入力
7. 他は入力せず「次へ」>「次へ」>「スタックの作成」



CLI の場合

```
$ aws cloudformation create-stack \
--region ap-northeast-1 \
--stack-name vpc-for-handson-cicd-xx \
--template-body https://s3-ap-northeast-1.amazonaws.com/tarohiro-temp/20181002-ecs-cloud9-cicd-handson/assets/cloudformation.yml
```

(2)	(3)	(4)	(5)	(6)
東京	サービス	リソースグループ	テンプレートの指定	スタックの名前
サポート	CloudFormation	CloudFormation テンプレートによるリソ	テンプレートソース Amazon S3 URL https://s3-ap-northeast-1.amazonaws.com/tarohiro-temp/20181002-ecs-cloud9-cicd-handson/ass	スタックの名前 vpc-for-handson-cicd-00



作成されたVPC IDの確認

サービス>VPC

The screenshot shows the AWS VPC console interface. A large orange callout box at the top right contains the text "cicd-XXで検索". A red circle labeled "1" highlights the "VPC" tab under "Virtual Private Cloud". A search bar labeled "2" contains the text "cicd-99". A red box labeled "3" surrounds the row for "handson-vpc99". A red circle labeled "5" points to the "Name" column of the selected row. A large orange callout box on the right contains the instructions: "Name欄をhandson-vpcXXに変えてください。検索されたVPC IDをメモしてください。". At the bottom, another orange callout box contains the text "自身のスタック名か確認(*-cicd-XX)".

1

2 cicd-XXで検索

3

5

Name欄をhandson-vpcXXに変えてください。
検索されたVPC IDをメモしてください。

自身のスタック名か確認(*-cicd-XX)

VPC ダッシュボード
VPC でフィルタリング:
VPC の選択

Virtual Private Cloud

VPC

サブネット

VPC ID

状態

IPv4 CIDR

available 1

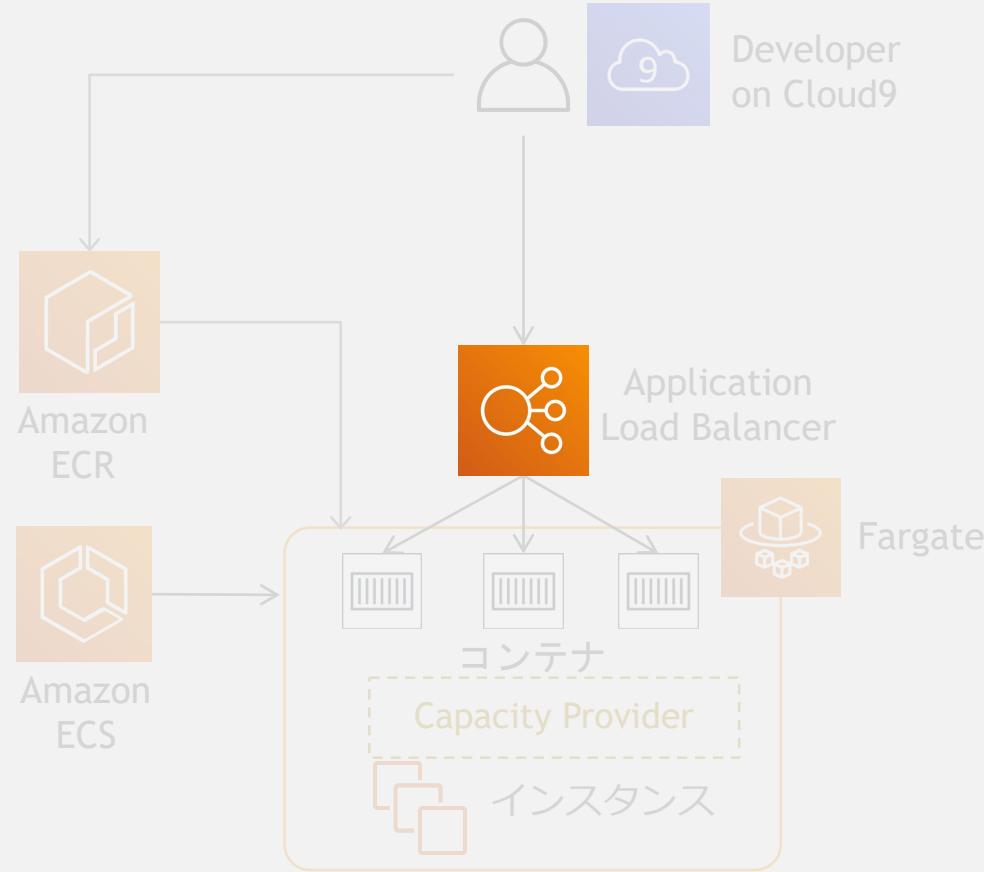
説明 CIDR ブロック フローログ タグ

タグの追加/編集

キー	値
Name	handson-vpc
aws:cloudformation:logical-id	VPC
aws:cloudformation:stack-id	arn:aws:cloudformation:ap-nor
aws:cloudformation:stack-name	vpc-handson-cicd-99

ALB作成

ここまで作る



ALB用セキュリティグループ作成(1)

サービス>VPC

左側メニュー:

- New VPC Experience Tell us what you think
- VPC ダッシュボード New
- VPC でフィルタリング:
 - VPC の選択
- VIRTUAL PRIVATE CLOUD
- セキュリティ
 - ネットワーク ACL
 - セキュリティグループ New

セキュリティグループを作成ボタン (赤枠):

セキュリティグループ名: **ALB security group XX**

説明: **開発者への SSH アクセスを許可**

VPC: **vpc-aa1644cf (HandsOn)**

セキュリティグループを作成ボタン (赤枠):

ALB用セキュリティグループ作成(2)

セキュリティグループ (1/4) 情報

検索 セキュリティグループをフィルタリング

作成したALB用SGを選択

Name	セキュリティグループ名	セキュリティグループID	VPC ID	説明	所有者
<input checked="" type="checkbox"/> -	sg-009362d242258dde7	ALB security group 20...	vpc-05d18edc2f7e073da	20200914 fargate	294963776963
<input type="checkbox"/> -	sg-0f21ba7a7fb702fe6	default	vpc-05d18edc2f7e073da	default VPC security gr...	294963776963
<input type="checkbox"/> -	sg-0fe2d7e462b342688	launch-wizard-1	vpc-5e977a3a	launch-wizard-1 create...	294963776963
<input type="checkbox"/> -	sg-69c1420e	default	vpc-5e977a3a	default VPC security gr...	294963776963

sg-009362d242258dde7 - ALB security group 20200914

詳細 インバウンドルール アウトバウンドルール タグ

インバウンドルール

HTTPとカスタムTCP(8080)を
Anywhere(0.0.0.0/0)に開放し、
Save Rules

インバウンドルール 情報

タイプ フロトコル ポート範囲 ソース

HTTP	TCP	80	任意の... 0.0.0.0/ */0
カスタム TCP	TCP	8080	任意の... 0.0.0.0/ */0

説明 - オプション 情報

インバウンドルールを編集

ルールを保存

aws

ALB(Application Load Balancer)作成

Services > EC2 > Load Balancer

The screenshot shows the AWS EC2 Load Balancer creation interface. On the left, a sidebar lists services: EC2 Dashboard (New), Events (New), Tags, Limits, Instances, Images, Elastic Block Store, Networks & Security, and Load Balancing. The 'Load Balancing' section is expanded, and 'Load Balancer' is selected, both highlighted with red boxes. The main area shows a search bar and a table for creating a load balancer. A red box highlights the 'Create' button in the top right of the table header. A red arrow points from the 'Load Balancer' sidebar entry to this button. Another red arrow points from the 'Create' button to the 'Application Load Balancer' row in the table. An orange callout box on the right says 'Application Load Balancer を選択'. The table rows are:

Load Balancer Type	Protocol	Action
Application Load Balancer	HTTP, HTTPS	作成
Network Load Balancer	TCP, TLS, UDP	作成
Classic Load Balancer	以前の世代 HTTP, HTTPS, および TCP	作成

Application Load Balancer を選択

Application Load Balancer

HTTP, HTTPS

作成

非常に柔軟性の高い機能セットが必要な場合は、Application Load Balancer を選択します。Application Load Balancer はリエクストレーブルで動作し、マイクロサービスとコラボを含む、アプリケーションアーキテクチャを対象とした高度なルーティングおよび可視性機能を提供します。

詳細はごちら >

Network Load Balancer

TCP, TLS, UDP

作成

非常に高いパフォーマンス、大規模な TLS のオフロード、証明書のデプロイの一元管理、UDP のサポート、およびアプリケーションの静的 IP アドレスが必要な場合は、Network Load Balancer を選択します。Network Load Balancer は接続レベルで動作し、非常に低いレイテンシを維持しながら、1 秒あたり数百万のリクエストを確実に処理することができます。

詳細はごちら >

Classic Load Balancer

以前の世代
HTTP, HTTPS, および TCP

作成

EC2-Classic ネットワークで既存のアプリケーションを実行している場合は、Classic Load Balancer を選択します。

詳細はごちら >

Load Balancer の選択

基本設定

1. ロードバランサーの設定 2. セキュリティ設定の構成 3. セキュリティグループの設定 4. ルーティングの設定 5. ターゲット

手順 1: ロードバランサーの設定

基本的な設定

ロードバランサーを設定するには、名前を指定し、スキームを選択して、1つ以上のリスナーを指定し、ネットワークを選択します。デフォルトのネットワークはインターネット接続ロードバランサーです。

名前 (i) (オレンジ色で強調)

スキーム (i) インターネット向け (オレンジ色で強調)

IP アドレスタイプ (i)

Nameは php-sample-xx

Internet-facingを選択

リスナー

リスナーとは、設定したプロトコルとポートを使用して接続リクエストをチェックするプロセスです。

ロードバランサーのプロトコル (オレンジ色で強調)

ロードバランサーのポート (オレンジ色で強調)

リスナーの追加

Listenerはデフォルト

アベイラビリティーゾーン

ロードバランサーのアベイラビリティーゾーンを指定します。ロードバランサーは、これらのアベイラビリティーゾーンにのみトラフィックを送ります。ロードバランサーの可用性を高めるには、2つ以上のアベイラビリティーゾーンからサブネットを指定する必要があります。

VPC (i) (オレンジ色で強調)

アベイラビリティーゾーン ap-northeast-1a (オレンジ色で強調)

ap-northeast-1c (オレンジ色で強調)

IPv4 アドレス (i) AWS によって割り当て済み

IPv4 アドレス (i) AWS によって割り当て済み

handsonで作成したvpcを選択



ここに注意

サブネットにはap-northeast-1aの
Public subnetとap-northeast-1cの
Public subnet 2を選択

キャンセル 次の手順: セキュリティ設定の構成



セキュリティ設定(スキップ)

手順 2: セキュリティ設定の構成

▲ ロードバランサーのセキュリティを向上させましょう。ロードバランサーは、いずれのセキュアリスナーも使用していません。

ロードバランサーへのトラフィックを保護する必要がある場合は、フロントエンド接続に HTTPS プロトコルをお使いください。最初のステップに戻り、[基本的な設定](#) セクションでセキュアなリスナーを追加/設定することができます。または現在の設定のまま続行することもできます。

キャンセル

戻る

次の手順: セキュリティグループの設定



セキュリティグループ設定

手順 3: セキュリティグループの設定

セキュリティグループは、ロードバランサーへのトラフィックを制御するファイアウォールのルールセットです。このページで、特定のトラフィックに対してロードバランサーへの到達を許可するルールを追加できます。最初に、新しいセキュリティグループを作成するか、既存のセキュリティグループから選択するかを決定します。

セキュリティグループの割り当て:
 新しいセキュリティグループを作成する
 既存のセキュリティグループを選択する

フィルタ [VPC セキュリティグループ]

セキュリティグループ ID	名前	説明	アクション
<input checked="" type="checkbox"/> sg-011ce5af87d7cd9a	ALB Security Group 20200908	test	コピーして新規作成
<input checked="" type="checkbox"/> sg-0bae0db904d8743ce	default	default VPC security group	コピーして新規作成



ここに注意

先程作成したALB用のSGと
defaultのSGを **両方** チェック

キャンセル 戻る 次の手順: ルーティングの設定



ルーティング設定

手順 4: ルーティングの設定

ロードバランサーは、指定するプロトコルとポートを使用してこのターゲットグループのターゲットにリクエストをルーティングし、これらのヘルスチェック設定を使用してターゲットでヘルスチェックを実行します。各ターゲットグループは1つのロードバランサーのみに関連付けることができることに注意してください。

ターゲットグループ

ターゲットグループ [①](#) 新しいターゲットグループ

名前 [①](#) dummy-20200908

dummy-xx

ターゲットの種類
 インスタンス
 IP
 Lambda 関数

プロトコル [①](#) HTTP

ポート [①](#) 80

適当な名前を指定
(このターゲットグループは
あとの手順で削除します)

ヘルスチェック

プロトコル [①](#) HTTP

パス [①](#) /

▶ ヘルスチェックの詳細設定

キャンセル

戻る

次の手順: ターゲットの登録



ターゲットトレジスター(スキップ)

手順 5: ターゲットの登録

ターゲットグループにターゲットを登録します。有効にしたアベイラビリティゾーンでターゲットを登録する場合、登録処理が完了し、ターゲットが最初のヘルスチェックに合格するとすぐに、ロードバランサーはターゲットへのリクエストのルーティングを開始します。

登録済みターゲット

インスタンスを登録解除するには、1つ以上の登録インスタンスを選択し、[削除] をクリックします。

削除

<input type="checkbox"/>	インスタンス	名前	ポート	状態	セキュリティグループ	ゾーン	▼
利用可能なインスタンスがありません。							

インスタンス

追加のインスタンスを登録するには、1つ以上の実行中のインスタンスを選択し、ポートを指定して、[追加] をクリックします。デフォルトのポートは、ターゲットグループに対して指定されたポートです。指定されたポートすでにインスタンスが登録されている場合は、別のポートを指定する必要があります。

登録済みに追加 ポート 80

検索



<input type="checkbox"/>	インスタンス	名前	状態	セキュリティグルー	ゾーン	サブネット ID	サブネット CIDR	▼
利用可能なインスタンスがありません。								

利用可能なインスタンスがありません。

キャンセル 戻る 次の手順: 確認



ALB作成

手順 6: 確認

操作する前にロードバランサーの詳細を確認してください。

ロードバランサー

名前: php-sample20200908
スケーム: internet-facing
リスナー: ポート:80 - プロトコル:HTTP
IPアドレスタイプ: ipv4
VPC: vpc-09c1a290b9126f23a (hands-on-vpc20200908)
サブネット: subnet-044aa0c915c2d372e (Public subnet), subnet-0883df31ef63c16ae (Public subnet 2)
タグ

セキュリティグループ

セキュリティグループ: sg-0bae0db904d8743ce, sg-011ce5af87d7cd9a

ルーティング

ターゲットグループ: 新しいターゲットグループ
ターゲットグループ名: dummy-20200908
ポート: 80
ターゲットの種類: instance
プロトコル: HTTP
ヘルスチェックプロトコル: HTTP
バス:
ヘルスチェックポート: traffic port
正常のしきい値: 5

作成

VPC ID	アベイラビリティゾーン	種類	作成日
vpc-09c1a290b9126f23a	ap-northeast-1a, ap-nor...	application	2020年9月

ロードバランサー: php-sample20200908

説明 リスナー モニタリング 統合サービス タグ

基本的な設定

名前: php-sample20200908
ARN: arn:aws:elasticloadbalancing:ap-northeast-1:294963776963:loadbalancer/app/php-sample20200908/71260db5811188d0
DNS 名: **php-sample20200908-1148614587.ap-northeast-1.elb.amazonaws.com**
状態: provisioning

アプリケーションのエンドポイントとなるDNS名をメモしておく(DNS名横のアイコンをクリック)



ALBリスナー削除

リスナーおよびターゲットはECSサービス構成時に再度作成しますので、ALB作成時に自動構成されたものは削除します

The screenshot shows the AWS CloudFormation console with the following details:

- Table Headers:** 名前, DNS名, 状態, VPC
- Table Data:** php-sample20200908 (dnsName: php-sample20200908-11486..., status: provisioning)
- Context:** ロードバランサー: php-sample20200908
- Tabs:** 説明, **リスナー**, モニタリング, 統合サービス, タグ
- Buttons:** リスナーの追加, **編集**, **削除**
- Form Fields:** リスナーID (checkbox checked), セキュリティポリシー (HTTP : 80, 該当なし), SSL 証明書 (該当なし), ルール (デフォルト: 転送先 dummy-20200908, ルールの表示/編集)
- Modal:** リスナーの削除 (確認メッセージ: これらのリスナーを削除してよろしいですか? HTTP : 80)
 - キャンセル
 - はい、削除する**

ターゲットグループ削除

リスナーおよびターゲットはECSサービス構成時に
再度作成しますので、ALB作成時に
自動構成されたものは削除します

The screenshot shows the AWS EC2 Target Groups page. On the left, a sidebar lists various EC2 services. The 'Target Groups' section is highlighted with a red box and has a red arrow pointing to it from the top-left. In the main content area, a table titled 'ターゲットグループ (1/1)' displays one target group row:

名前	ARN	ポート	プロトコル	ターゲットの種類
<input checked="" type="checkbox"/> dummy-20200908	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/dummy-20200908/7890123456789012	80	HTTP	インスタンス

A red box highlights the target group name 'dummy-20200908'. A red arrow points from this box to a larger red box on the right side of the screen, which contains a confirmation dialog box. The dialog box is titled 'Delete target group?' and contains the following text:

このアクションは、元に戻すことができません。
ターゲットグループを削除すると、グループは削除されます。ターゲットグループに登録されている個々のリソースは、このアクションの結果として削除されません。

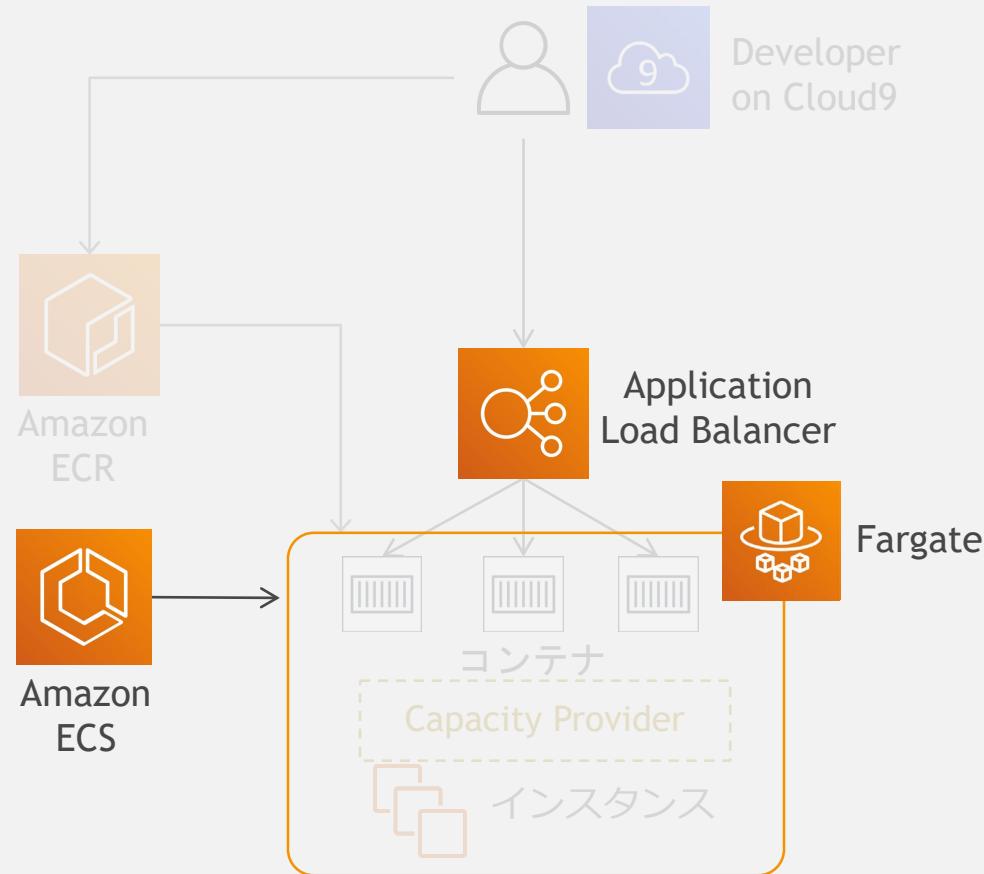
このターゲットグループを削除してよろしいですか?

- dummy-20200908

At the bottom of the dialog box, there are two buttons: 'キャンセル' (Cancel) and a highlighted orange button labeled 'はい、削除します' (Yes, Delete).

ECSクラスター作成

ここまで作る



Fargateクラスター作成(1)

ECS
コンソール

Services > ECS > クラスター

Amazon ECS

クラスター

タスク定義

Account Settings

Amazon EKS

Clusters

Amazon ECR

リポジトリ

AWS Marketplace

ソフトウェアを探す

サブスクリプション

クラスター

Amazon ECS クラスターは、タスクリクエットのクラスターを受け取ります。クラスター

詳細については、[ECS のドキュメント](#)を参照

クラスターの作成

今すぐ始める

表示 リスト カード

test20180728 >

CloudWatch
デフォルト

FARGATE

0

サービス

EC2

クラスターテンプレートの選択

クラスターの作成を簡略化するために、次のクラスターテンプレートが利用できます。後でその他の設定や統合を追加することができます。

ネットワーキングのみ

作成するリソース:

クラスター

VPC (オプション)

サブネット (オプション)

AWS Fargate を使用

EC2 Linux + ネットワーキング

作成するリソース:

クラスター

VPC

サブネット

Linux AMI を持つ Auto Scaling グループ

EC2 Windows + ネットワーキング

作成するリソース:

クラスター

VPC

サブネット

Windows AMI を持つ Auto Scaling グループ

次のステップ

Fargateクラスター作成(2)

クラスターの設定

クラスター名* ⓘ

ネットワーキング

クラスターで使用する新しいVPC を作成します。VPC は、Fargate タスクなどの AWS オブジェクトによって取得される AWS クラウドの分離された部分です。

VPC の作成 このクラスター用の新しいVPC を作成する

クラスター名を指定
(fargate-cluster)

作成

Fargateクラスターの確認

起動ステータス

コンテナインスタンスが起動中です。実行状態でアクセス可能になるまでに数分かかることがあります。新しいコンテナインスタンスの使用時間はすぐに開始され、インスタンス

The screenshot shows the AWS ECS Cluster Details page for the cluster 'fargate-clustre'. A red box highlights the 'Cluster Details' section, and a red arrow points from the 'Cluster Details' button on the left to this section.

Cluster Details:

- Cluster ARN: arn:aws:ecs:ap-northeast-1:294963776963:cluster/fargate-clustre
- 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
- 実行中のタスクの数: 0 個の Fargate、0 個の EC2
- アクティブサービス数: 0 個の Fargate、0 個の EC2
- ドレーニングサービス数: 0 個の Fargate、0 個の EC2

Action Buttons: クラスターの更新, クラスターの削除

ECS Cluster Status: ECSステータス - 1/1 完了 fargate-clustre

ECS Cluster Status Summary: ECS クラスター
ECSクラスターfargate-clustreが正常に作成されました

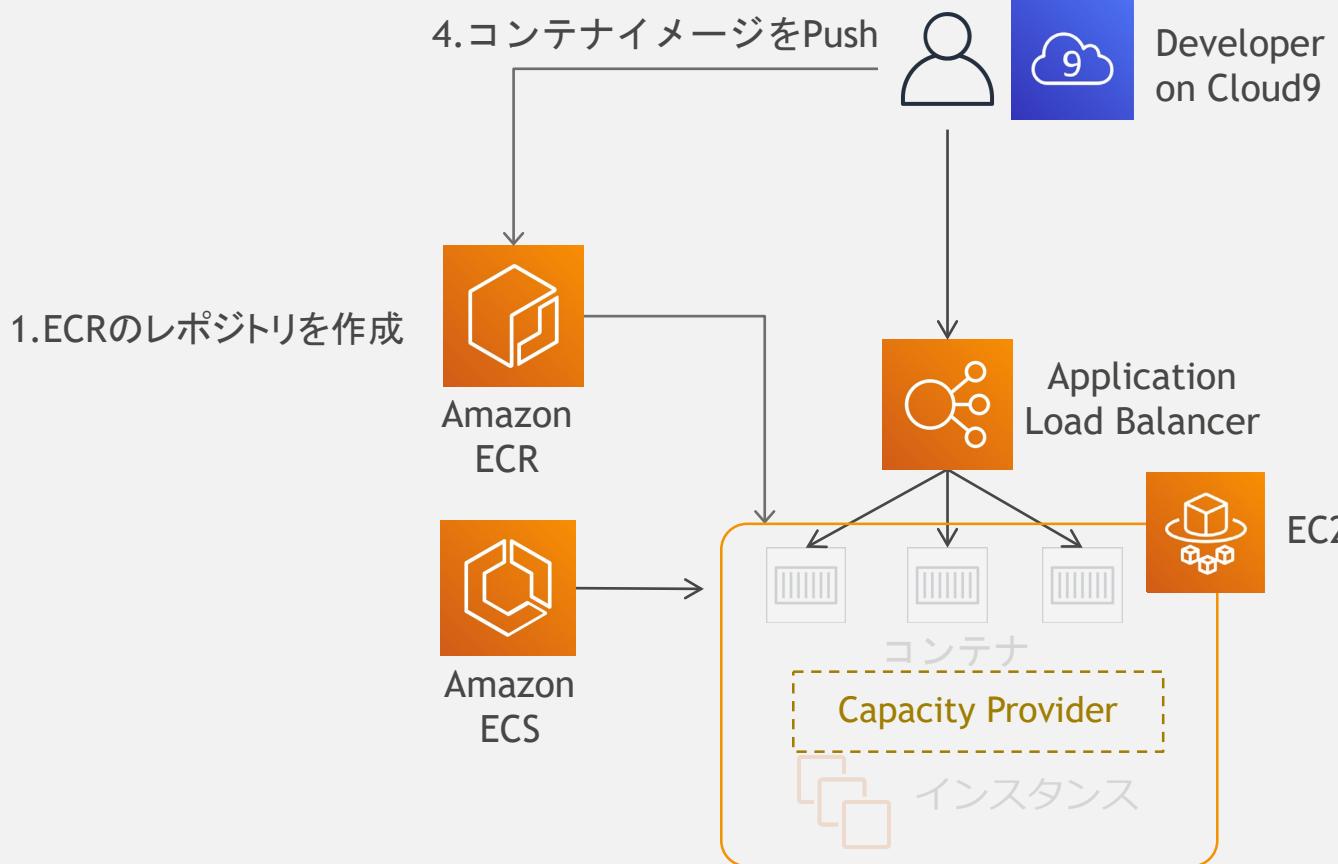
Filter Options: このページのフィルター, 起動タイプ: ALL, サービスタイプ: ALL

サービス名	ステータス	サービスタイ... タスク定義	必要なタスク... 実行中のタス...	起動タイプ	プラットフォ...
結果がありません					

Cloud9でのDockerイメージ開発 ECRへのPush

ここまで作る

2.Cloud9の環境構築
3.コンテナイメージ作成



ECRリポジトリ作成

Services > ECR へ

ECR
コンソール

コンピュート

Amazon Elastic Container Registry

コンテナイメージを簡単に保存、管理、およびデプロイする

Amazon Elastic Container Registry (ECR) は完全マネージド型のコンテナレジストリで、開発者がコンテナイメージを簡単に保存、管理、デプロイできるようにします。

リポジトリの作成

使用方法

ECRリポジトリ作成

リポジトリの設定

リポジトリ名

.dkr.ecr.ap-northeast-
1.amazonaws.com/

php-sample-01

php-sample-xx

リポジトリ名には名前空間を含めることができます (例: namespace/repo-name)。

タグのイミュータビリティ

タグのイミュータビリティを有効にして、同じタグを使用した後続イメージのプッシュでイメージタグが上書きされないようにします。タグのイミュータビリティを無効にして、イメージタグを上書きできるようにします。

無効にする

プッシュ時にスキャン

プッシュ時にスキャンを有効にすると、各イメージがリポジトリにプッシュされた後に自動的にスキャンされます。無効にした場合、スキャン結果を取得するには、各イメージのスキャンを手動で開始する必要があります。

無効にする

任意のリポジトリ名を指定

キャンセル

リポジトリを作成

ECRリポジトリ作成

リポジトリ (1)

リポジトリ名 URI 作成時刻

php-sample-01 .dkr.ecr.ap-northeast-1.an

リポジトリを検索

ブッシュコマンドの表示 削除 編集 リポジトリを作成

php-sample-20200908 のブッシュコマンド

macOS / Linux Windows

AWS CLI および Docker の最新バージョンがインストールされていることを確認します。詳細については、Amazon ECR の開始方法 [を参照してください。](#)

次の手順を使用して、リポジトリに対してイメージを認証し、ブッシュします。Amazon ECR 認証情報ヘルパーなどの追加のレジストリ認証方法については、レジストリの認証 [を参照してください。](#)

1. 認証トークンを取得し、レジストリに対して Docker クライアントを認証します。
AWS CLI を使用します。

```
aws ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-stdin
```

注意: AWS CLI の使用中にエラーが発生した場合は、最新バージョンの AWS CLI と Docker がインストールされていることを確認してください。

2. 以下のコマンドを使用して、Docker イメージを構築します。一から Docker ファイルを構築する方法については、「[こちらをクリック](#)」の手順を参照してください。既にイメージが構築されている場合は、このステップをスキップします。

```
docker build -t php-sample-20200908 .
```

これらのコマンドをこのあとの手順で利用する（後で指示があります）

AWS Cloud9環境の構築

Services > Cloud9 へ

The screenshot shows the AWS Cloud9 environment creation interface. At the top left, there's a search bar with 'cloud9' typed in. To its right is a '役に立つヒント' (Helpful Hint) section about AWS Budgets. A large orange callout box with the text '[Create environment] をクリック' (Click [Create environment]) points to a prominent orange button labeled 'Create environment' in the main central area. This central area contains the AWS Cloud9 logo and the tagline 'a cloud IDE for writing, running, and debugging code'. Below this is a descriptive paragraph about AWS Cloud9's benefits. On the left side, there's a sidebar with navigation links for various AWS services: 'コンピューティング' (Computing) including EC2, Lightsail, Elastic Container Service, EKS, Lambda, Batch, and Elastic Beanstalk; 'ストレージ' (Storage) including S3, EFS, Glacier, and Storage Gateway; and 'データベース' (Database) including RDS, DynamoDB, ElastiCache, Neptune, and Amazon Redshift. On the right side, there's a 'Getting started' section with several links: 'Before you start' (2 min read), 'Create a environment' (3 min read), 'Working with environments' (15 min read), 'Working with the IDE' (10 min read), and 'Working with AWS Lambda' (5 min read). The bottom of the page has a note about using the AWS Cloud9 dashboard to switch between environments.

AWS サービス

cloud9

Cloud9
コードの記述、実行、デバッグのためのクラウド IDE

コンピューティング

- EC2
- Lightsail
- Elastic Container Service
- EKS
- Lambda
- Batch
- Elastic Beanstalk

ストレージ

- S3
- EFS
- Glacier
- Storage Gateway

データベース

- RDS
- DynamoDB
- ElastiCache
- Neptune
- Amazon Redshift

Developer Tools

AWS Cloud9

a cloud IDE for writing, running, and debugging code

AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

How it works

Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.

Using the AWS Cloud9 dashboard, you can create and switch between many environments.

役に立つヒント

コストの管理

AWS Budgets を使用して、AWS のコスト、使用量、および予約をモニタリングします。今すぐ開始

[Create environment] をクリック

New AWS Cloud9 environment

Create environment

Getting started

- Before you start 2 min read
- Create a environment 3 min read
- Working with environments 15 min read
- Working with the IDE 10 min read
- Working with AWS Lambda 5 min read

Cloud9



AWS Cloud9環境の構築

AWS Cloud9 > Environments > Create environment

Step 1
Name environment

Step 2
Configure settings

Step 3
Review

Name environment

“hanson-xx”を入力

Environment name and description

Name

This name needs to be unique per user. You can update it at any time in your environment settings.

hanson-yourname

Limit: 60 characters

Description - Optional

This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

Write a short description for your environment

[Next step]をクリック

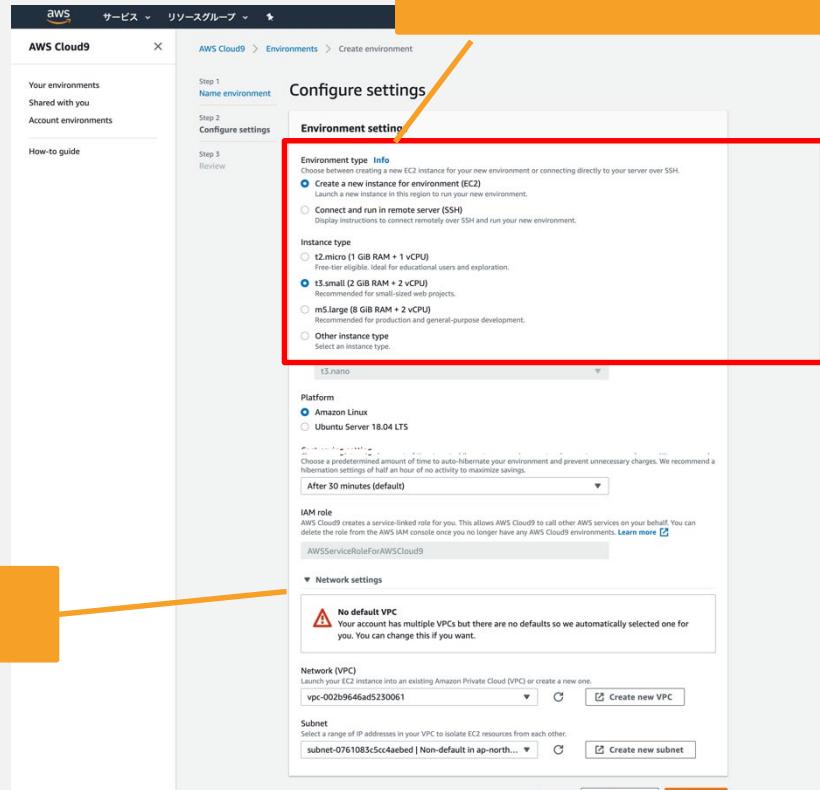
Limit: 200 characters

Cancel

Next step

手順の次ページで[Network settings]を設定

※ docker build 時などでメモリが不足することがあるので、t3.small 以上が望ましい。



AWS Cloud9環境の構築

ハンズオンで作成したVPCを選択
(Services -> VPC で確認)

Network settings (advanced)

Network (VPC)
vpc-t [▼] Create new VPC

Subnet
Select a range of IP addresses in your VPC to isolate EC2 resources from each other.
is-east-1a [▼] Create new subnet

ここに注意
Public subnetのいずれかを選択
(Services -> VPC -> Subnetで確認)

[Next step]

Review

Environment name and settings

Name: hansdon-yourname
Description: No description provided
Environment type: EC2
Instance type: t2.micro
Subnet: subnet-[...]
Cost-saving: After 30 min
IAM role: AWSServiceRoleForAWSCloud9 (generated)

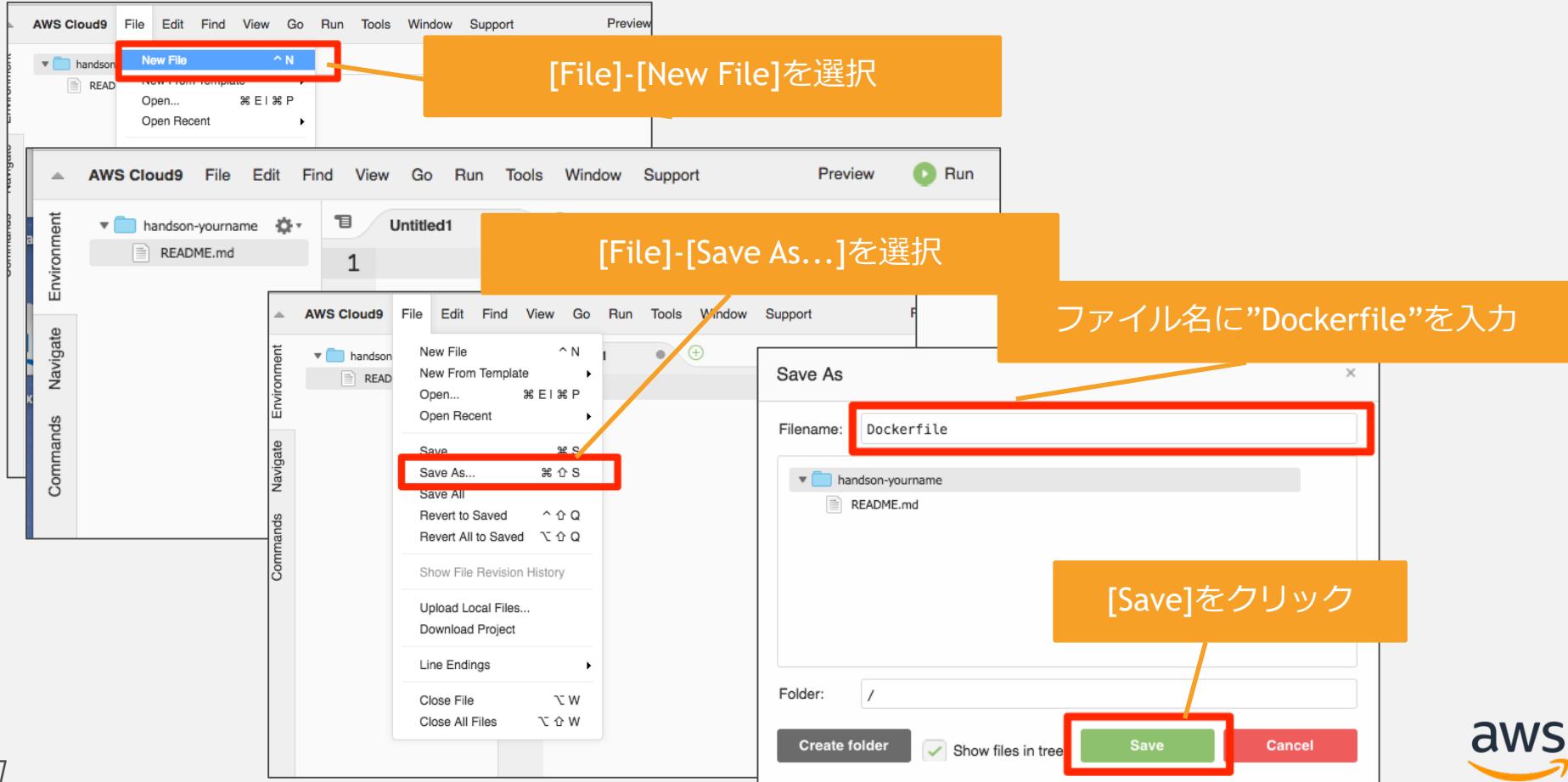
[Create environment] をクリック

We recommend the following best practices for using your AWS Cloud9 environment

- Use source control and backup your environment frequently. AWS Cloud9 does not perform automatic backups.
- Perform regular updates of software on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
- Turn on AWS CloudTrail in your AWS account to track activity in your environment. Learn more
- Only share your environment with trusted users. Sharing your environment may put your AWS access credentials at risk. Learn more

[Create environment]

Dockerfileの作成



Dockerfile作成(PHPサンプルアプリ)

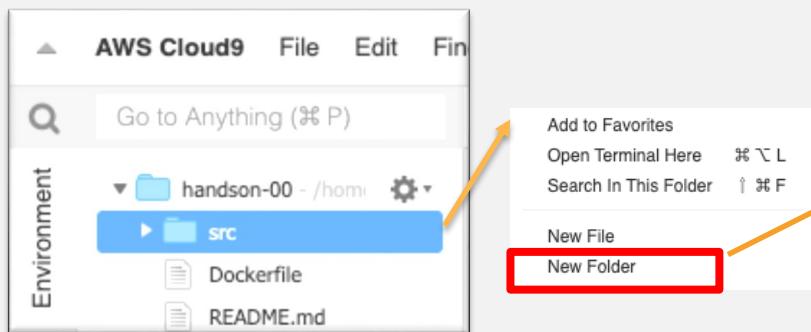
1. 前ページで作成したDockerfileに以下の内容を入力して保存

```
FROM php:7.0-apache  
COPY src/ /var/www/html/
```

※ ハンズオン資料の中に上記のコードを置いてるので、
そちらをコピペして頂いて構いません

これにより、Docker Hubの公式イメージからPHP/apacheのDockerイメージが取得されます
https://hub.docker.com/_/php/

2. srcフォルダーを作成



これにより、ローカルのsrcディレクトリ配下がコンテナ内の/var/www/htmlディレクトリにコピーされます

このペインで、マウスを左クリックし、メニューからNew Folderを選択、srcフォルダーを作成する

Dockerfile作成(PHPサンプルアプリ)

3. srcフォルダー内にindex.phpファイルを作成

srcフォルダーを選択し、File>New Fileで作成、File>Save As...で保存

```
<html>
  <head>
    <title>PHP Sample</title>
  </head>
  <body>
    <?php echo gethostname(); ?>
  </body>
</html>
```

ホスト名を表示するPHPアプリ
(余裕のある方は好きなコードを書いてください)

※ ハンズオン資料の中に上記のコードを置いてるので、
そちらをコピペして頂いて構いません

Dockerfile作成(PHPサンプルアプリ) - 例

AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run Share

handson-tarohiro Dockerfile index.php

Dockerfile

index.php

README.md

FROM php:7.0-apache
COPY src/ /var/www/html/

<html>
<head>
<title>PHP Sample</title>
</head>
<body>
<?php echo gethostname(); ?>
</body>
</html>

ここが「x」ではなく「・」ドットになっている場合は、保存されていません。File>Saveで保存してください。

2:25 Dockerfile Spaces: 4 5:9 PHP Spaces: 2

Dockerイメージ作成 ~ コンテナ起動

Cloud9ターミナルで作業

1. Dockerイメージ作成

```
$ docker build -t php-sample-xx . #メモしておいたコマンド("xx"の後はスペースとドット)  
$ docker images  
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE  
php-sample      latest    bc47e3ede49a    3 minutes ago   390 MB
```

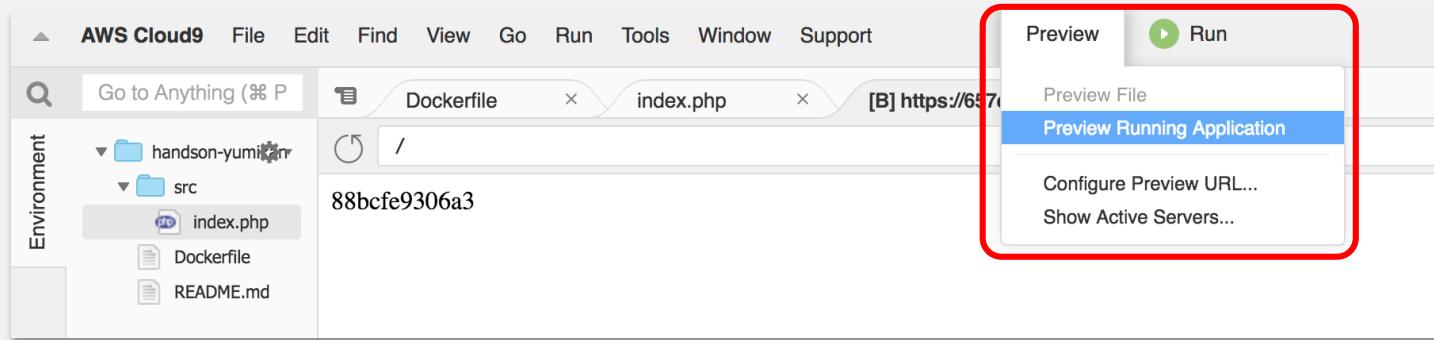
メモを忘れた方は、サービス>ECR、一覧から自身のリポジトリを選択、画面右上の「プッシュコマンドの表示」を押すとコマンドが表示されます。

2. Dockerコンテナ起動

```
$ docker run --rm -p 8080:80 -d php-sample-xx:latest #ホストPort8080でコンテナport80にアクセス  
$ docker ps  
$ CONTAINER ID  IMAGE      ~ ~ ~ PORTS      NAMES  
67776a2787dd  php-sample:latest  ~ ~ ~ 0.0.0.0:8080->80/tcp  modest_wozniak
```

Dockerイメージ作成 ~ コンテナ動作確認と停止

- [Preview]メニューの[Preview Running Application]を実行しPHPのページが表示されることを確認



- Dockerコンテナ停止

```
$ docker stop 67776a2787dd #`docker ps`で表示されたコンテナIDを指定
```

ECRへのDockerイメージPush

ブラウザの別タブでECR > リポジトリ > [レポジトリ名]を開き、「プッシュコマンドの表示」を押すと、下記のコマンドを確認できる。それらをCloud9上で実行

1. ECRへのログイン

```
$ aws ecr get-login --no-include-email --region ap-northeast-1 | docker login --username AWS --password-stdin  
xxxxxx.dkr.ecr.ap-northeast-1.amazonaws.com  
...  
Login Succeeded # ログインに成功することを確認する
```

2. タグ付け

```
$ docker tag php-sample-xx:latest XXXXXXXXXX.dkr.ecr.ap-northeast-1.amazonaws.com/php-sample-xx:latest
```

3. ECRへのイメージPush

```
$ docker push XXXXXXXXXX.dkr.ecr.ap-northeast-1.amazonaws.com/php-sample-xx:latest  
...  
latest: digest: sha256:.... size: 3450 # イメージ Push に成功することを確認する
```



ECRコンソールでのイメージ確認

Services > ECR > Repositories

ECS
コンソール

ECR > リポジトリ

リポジトリ (1)

リポジトリ名 URI 作成時刻 タグ

php-sample-01	.dkr.ecr.ap-northeast-1.amazonaws.com/php-sample-01	20/02/03 11:15:10	有効にする 無効にする
---------------	---	-------------------	-------------

このあとのタスク定義で利用するためリポジトリURIをメモしておく

ECR > リポジトリ > php-sample-01

php-sample-01

イメージ (1)

Image タグ イメージの URI ブッシュされた日時 ダイジェスト サイズ (MB) ステータスをスキャン 脆弱性

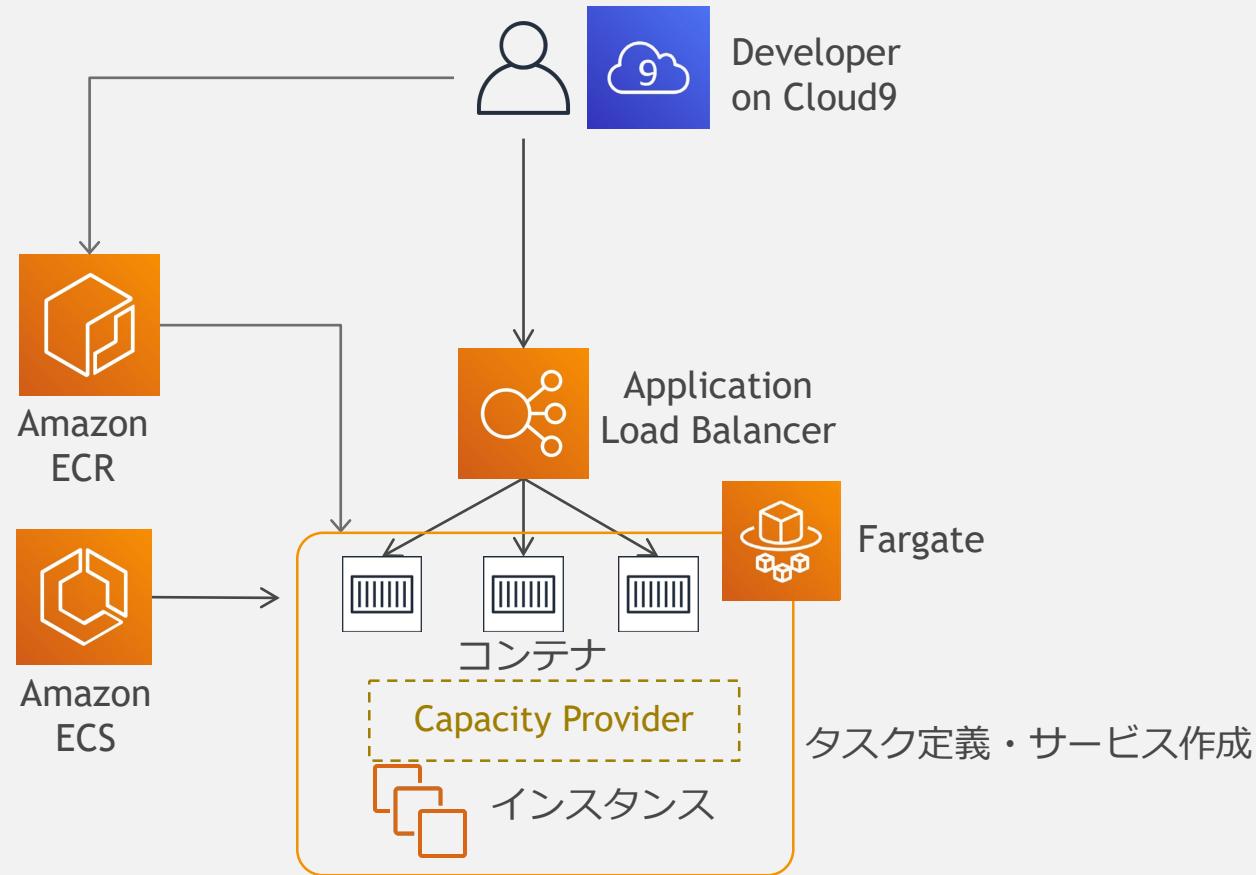
<input type="checkbox"/> latest	.dkr.ecr.ap-northeast-1.amazonaws.com/php-sample-01:latest	20/02/03 11:15:05	SHA-256: 70...L64	177.73	最新	-
---------------------------------	--	-------------------	-------------------	--------	----	---

ブッシュコマンドの表示 削除 スキャン < 1 > ⌂

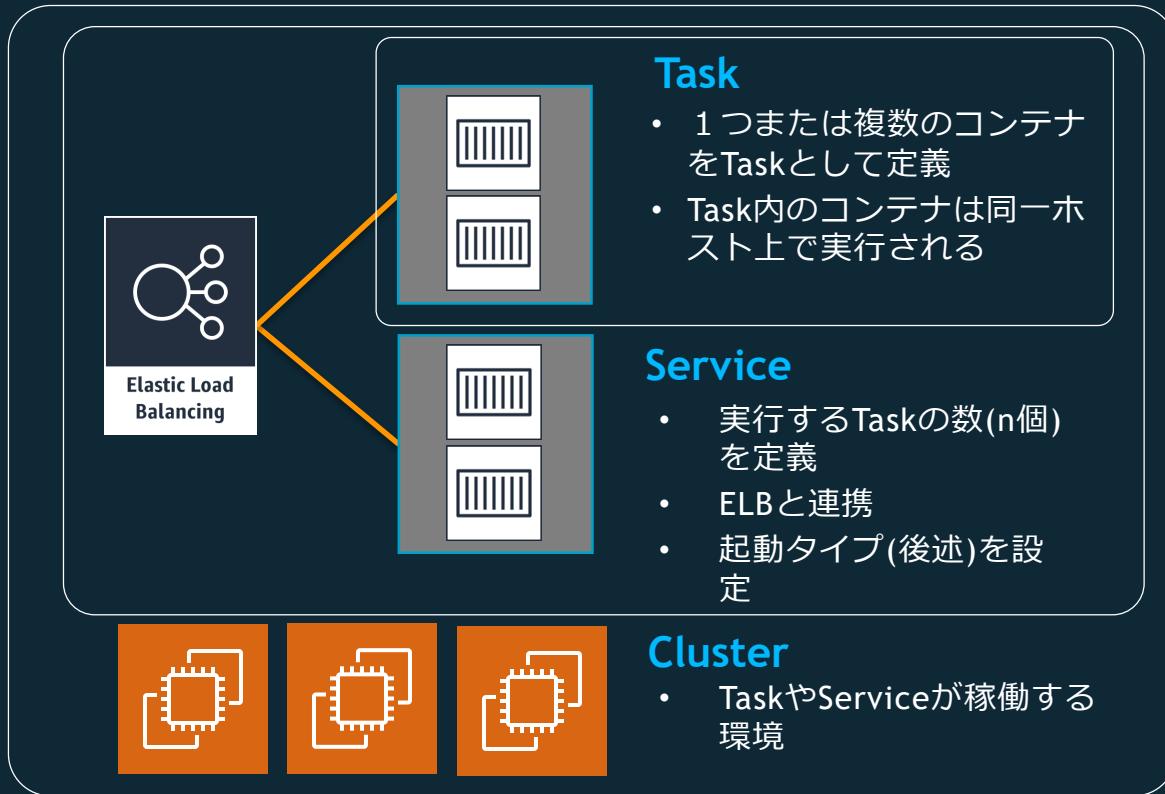
Latestのタグがついたイメージがあることを確認

ECSタスク定義・サービス作成

ここまで作る



Task Definition, Task, Service, Clusterの関係

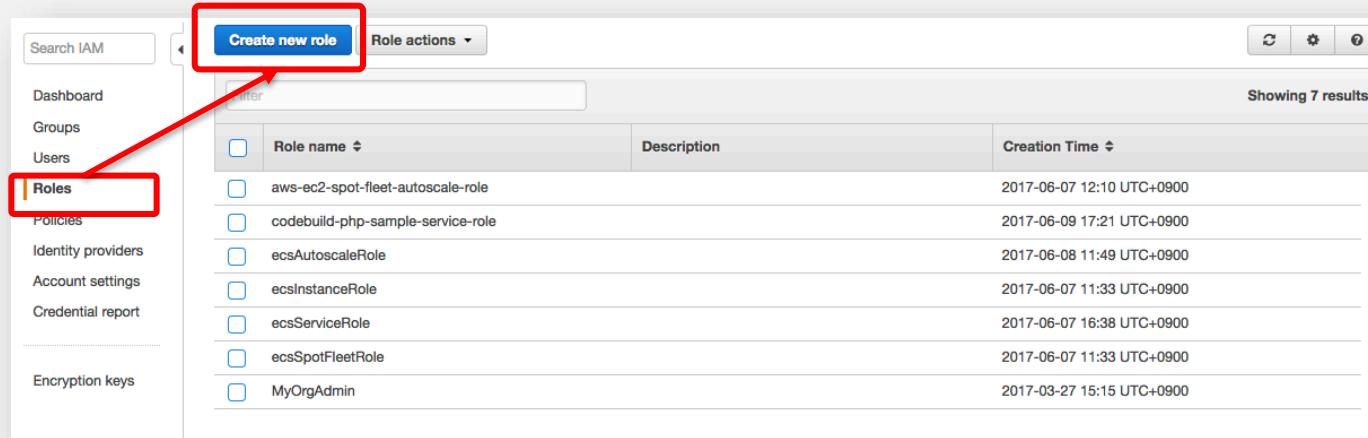


IAMロール設定(1)

コンソール

CodeDeployが利用するIAMロールを作成します。

Services -> IAM ^



The screenshot shows the AWS IAM Roles management interface. On the left, a sidebar lists navigation options: Dashboard, Groups, Users, Roles (which is highlighted with a red box), Policies, Identity providers, Account settings, Credential report, and Encryption keys. At the top center, there's a search bar and a 'Create new role' button, also highlighted with a red box. Below the search bar, there are filters for 'Role name' and 'Description'. A message at the top right says 'Showing 7 results'. The main table lists seven roles with their creation times:

Role name	Description	Creation Time
aws-ec2-spot-fleet-autoscale-role		2017-06-07 12:10 UTC+0900
codebuild-php-sample-service-role		2017-06-09 17:21 UTC+0900
ecsAutoscaleRole		2017-06-08 11:49 UTC+0900
ecsInstanceRole		2017-06-07 11:33 UTC+0900
ecsServiceRole		2017-06-07 16:38 UTC+0900
ecsSpotFleetRole		2017-06-07 11:33 UTC+0900
MyOrgAdmin		2017-03-27 15:15 UTC+0900



IAMロール設定(2)

Create role

Select type of trusted entity

1 2 3 4

AWS service EC2, Lambda and others
Another AWS account Belonging to you or 3rd party
Web Identity Cognito or any OpenID provider
SAML 2.0 federation Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	Comprehend	ElastiCache	Lambda	SMS
AWS Backup	Config	Elastic Beanstalk	Lex	SNS
AWS Support	Connect	Elastic Container Service	License Manager	SWF
Amplify	DMS	Elastic Transcoder	Machine Learning	SageMaker
AppSync	Data Lifecycle Manager	Elastic Load Balancing	Macie	Security Hub
Application Auto Scaling	Data Pipeline	Forecast	MediaConvert	Service Catalog
Application Discovery Service	DataSync	Glue	OpsWorks	Step Functions
Batch	DeepLens	Greengrass	Personalize	Storage Gateway
CloudFormation	Directory Service	GuardDuty	RAM	Textract
CloudHSM	DynamoDB	Inspector	RDS	Transfer
CloudTrail	EC2	IoT	Redshift	Trusted Advisor
CloudWatch Application Insights	EC2 - Fleet	IoT Things Graph	Rekognition	VPC
CloudWatch Events	EC2 Auto Scaling	KMS	RoboMaker	WorkLink
CodeBuild	EKS	Kinesis	S3	WorkMail
	EMR			

CodeDeploy

Select your use case

CodeDeploy
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

CodeDeploy - ECS
Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

CodeDeploy for Lambda
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.

* Required

Cancel

Next: Permissions

Roleタイプで
AWS Service >
CodeDeploy >
CodeDeploy - ECS を選択してから
Next: Permissionを選択



IAMロール設定(3)

Create role

Attached permissions policies

The type of role that you selected requires the following policy.

Policy name	Used as	Description
AWSCodeDeployRoleForECS	Permissions policy (1)	Provides CodeDeploy service wide access to ...

* Required

AWSCodeDeployRoleForECS
ポリシーが表示されていることを確認しNext: Tags

Cancel Previous Next: Tags

IAMロール設定(4)

Create role

Add tags (optional)

IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. [Learn more](#)

Key	Value (optional)	Remove
Add new key		

You can add 50 more tags.

1 2 3 4

Cancel Previous Next: Review

Role name: CodeDeployRoleforECS

Role description: Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

Trusted entities: AWS service: codedeploy.amazonaws.com

Policies: AWSCodeDeployRoleForECS

Permissions boundary: Permissions boundary is not set

No tags were added.

* Required

1 2 3 4

Cancel Previous Create role

□ ロール名はCodeDeployRoleforECSにしてCreate role

aws

Fargate用タスク定義作成(1)

Services -> ECS ^

ECS
コンソール

Amazon ECS

Clusters Task Definitions Repositories

Task Definitions

Task definitions specify the container information for your application, such as how many containers are part of your task, what resources they will use, how they are linked together, and which host ports they will use. [Learn more](#)

Create new Task Definition Create new revision Actions Last updated on June 7, 2017 3:42:29 PM (0m ago) [Edit](#) [Help](#)

Status: **ACTIVE** INACTIVE Filter in this page Task Definition

Select launch type compatibility

Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

FARGATE  Price based on task size Requires network mode awsvpc AWS-managed infrastructure, no Amazon EC2 instances to manage

EC2  Price based on resource usage Multiple network modes available Self-managed infrastructure using Amazon EC2 instances

*Required Cancel Next step

タスク定義作成(2)

タスクとコンテナの定義の設定

タスク定義では、タスクに含めるコンテナとコンテナ間のやり取りの方法を指定します。また、コンテナに使用するデータボリュームを指定することもできます。[詳細はどちら](#)

タスク定義名: ⓘ

互換性が必要: FARGATE

タスクロール: なし ⓘ
権限付与された AWS サービスへの API リクエストを行うためにタスクで使用できるオプションの IAM ロール。Amazon Elastic Container Service タスクロールは IAM コンソールで作成します。 ⓘ

ネットワークモード: awsvpc ⓘ
デフォルトを選択すると、ECS は Docker のデフォルトネットワーカード (Linux on Bridge と Windows NAT) を使用してコンテナを起動します。デフォルトは Windows で唯一サポートされるモードです。

タスクの実行 IAM ロール

このロールは、お客様に代わってコンテナイメージをpushし、コンテナログを Amazon CloudWatch に発行するタスクに必要です。また `ecsTaskExecutionRole` を持っていない場合は、お客様のためにそれを作成することができます。

タスク実行ロール: ⓘ

タスクサイズ

タスクサイズにより、タスクの固定サイズを指定できます。Fargate 起動タイプを使用したタスクにはタスクサイズが必須で、EC2 起動タイプではオプションです。タスクサイズが既定されている場合、コンテナレベルのメモリ設定はオプションです。タスクサイズは Windows コンテナではサポートされません。

コンテナメモリ (GB): ⓘ
0.25 vCPU の有効なメモリ範囲: 0.5GB - 2GB。

タスク CPU (vCPU): ⓘ
0.5 GB メモリの有効な CPU: 0.25 vCPU

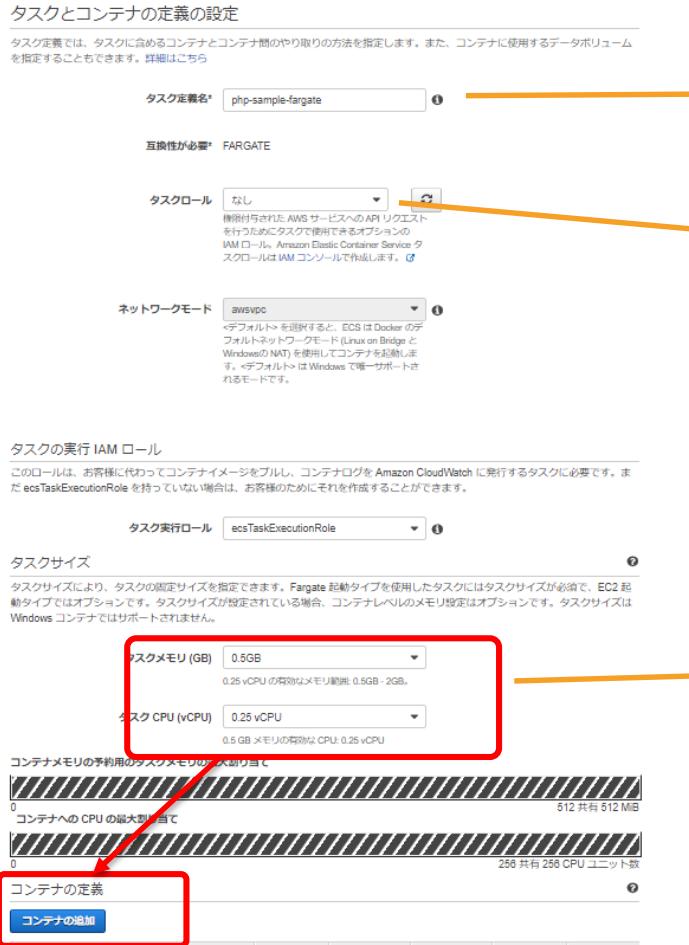
コンテナメモリの予約用のシステムメモリの割り当て

0 コンテナへの CPU の最大割り当て

0 256 共有 256 CPU ユニット数

コンテナの定義 ⓘ

コンテナの追加



タスク名を指定
(php-sample-fargate)

Task Role はなし/None

タスクに割り当てるメモリ/CPUを
0.5GB/0.25vCPUに指定



タスク定義作成(3)

▼ スタンダード

コンテナ名 *

イメージ * image:tag

プライベートレジストリの認証*

メモリ制限 (MiB) 128
+ ハード制限の追加

ポートマッピング コンテナポート プロトコル
+ ポートマッピングの追加

コンテナ名を指定(後続の手順の関係上今回はphp-sample-fargateに固定)

利用するDockerイメージを<ECRのリポジトリURI>:<タグ>の形式で指定(今回のタグはlatest)
リポジトリは以前の手順と同じものを指定する

コンテナに割り当てられるMemoryのリミットを定義
今回はSoftLimitで128MBを指定

コンテナportを80に設定

タスク定義作成(4)

環境

CPU ユニット数

GPU

基本 i

エントリポイント カンマ区切り: sh,-c i

コマンド カンマ区切り: echo,hello world i

作業ディレクトリ /usr/app i

環境変数

各コンテナに対してのCPU予約を定義
今回は1/4Core予約(1024 CPU units = 1Coreなので256を指定)

キャンセル

追加



タスク定義作成(5)

Container Definitions

Container N...	Image	Hard/Soft m...	CPU Units	GPU	Essential	
php-sam...	33509371799...	--/128	256		true	

Add container

Service Integration

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications. [Learn more](#)

Enable App Mesh integration

Proxy Configuration

The configuration details for the App Mesh proxy. [Learn More](#)

Enable Proxy Configuration

Volumes

Add volume

Configure via JSON

Tags

Key	Value
Add key	Add value

*Required

Cancel Previous Create



Fargateでのサービス作成(1)

タスク定義 > php-sample-fargate > 1

タスク定義: php-sample-fargate:1

タスク定義の詳細情報を表示します。タスク定義を変更するには、新しいリビジョンを作成するか、既存のリビジョンを更新するか、またはタスクの実行を開始するか選択してください。

新しいリビジョンの作成

アクション ▾

- タスクの実行
- サービスの作成**
- サービスの更新
- 登録解除

定義名: php

タスクロール: なし



Fargateでのサービス作成(2)

サービスの設定

サービスでは、クラスターで実行して維持するタスク定義のコピー数を指定できます。オプションで Elastic Load Balancing ロードバランサーを使用して、受信トラフィックをサービス内のコンテナに分散させることができます。Amazon ECS はタスクの数を維持し、ロードバランサーを使用してタスクのスケジュールを調整します。オプションで Service Auto Scaling を使用して、サービス内のタスクの数を調整することもできます。

起動タイプ FARGATE EC2

キャパシティープロバイダー戦略への切り替え

タスク定義 ファミリー

php-sample-fargate

リビジョン

1

プラットフォームのバージョン

LATEST

クラスター

fargate-cluster

サービス名

php-sample-fargate

サービスタイプ*

REPLICAS

タスクの数

1

最小ヘルス率

100

最大率

200

Launch TypeはFARGATEを選択

先ほど作成したタスク定義とクラスターを指定 (Fargate-cluster)

Service name は php-sample-fargate

初期起動タスク数を指定
今回は1タスクを起動する

Fargateでのサービス作成(3)

デプロイメント

サービスのデプロイメントオプションを選択してください。

デプロイメントタイプ*

ローリングアップデート ⓘ

Blue/Green デプロイメント (AWS CodeDeploy を使用) ⓘ

これは AWS CodeDeploy をサービスのデプロイメントコントローラーとして設定します。CodeDeploy アプリケーションとデプロイメントグループは、サービス用に自動的に作成されます。default settings サービスの作成後にローリングアップデートのデプロイメントタイプに変更するには、サービスを再作成して「ローリングアップデート」のデプロイメントタイプを選択する必要があります。

Deployment configuration*

CodeDeployDefault.ECSAllAtOnce

The deployment configuration specifies how traffic is shifted to the updated Amazon ECS task set. [Learn more](#)

CodeDeploy のサービスロール*

CodeDeployRoleforECS

サービスが承認済み AWS サービスへの API リクエストを行うために使用する IAM ロール。IAM コンソールで CodeDeploy のサービスロールを作成します。詳細は[こちら](#)

ⓘ タグ付けするには、新しい ARN とリソース ID の形式をオプトインする必要があります。
IAM ユーザーまたはロールが新しい ARN 形式をオプトインしていません。新しい形式をオプトインしてこの機能を使用します。オプトイン設定を管理します。 [➡](#)

*必須

キャンセル

次のステップ

Deployment TypeはBlue/green deploymentを選択

Service Roleは先ほど作成した CodeDeployRoleforECSを選択



Fargateでのサービス作成(4)

ネットワーク構成

VPC とセキュリティグループ

VPC とセキュリティグループは、タスク定義のネットワークモードが awsvpc であるときに設定可能です。

クラスター VPC*



サブネット*



セキュリティグループ*

php-sa-1134

編集

パブリック IP の自動割り当て

ENABLED

作成したVPCを選択
(10.1.0.0/16)

★よくある間違いポイント
VPC内のPrivateSubnetを2つ選択

★よくある間違いポイント
DefaultのSecurityGroupを選択(編集をおして変更)

Auto-assign public IPを
DISABLEDに変更



Fargateでのサービス作成(5)

ロードバランシング

Elastic Load Balancing ロードバランサーはサービス内で実行中のタスク間に受信トラフィックを分散させます。既存のロードバランサーを選択するか、Amazon EC2 コンソールでロードバランサーを作成してください。

ロードバランサーの種類*

Application Load Balancer

コンテナで動的ホストポートマッピングを有効にします(コンテインインスタンスごとに複数のタスクを許可)。ルールベースのルーティングとパスを使用することで、1つのロードバランサーの同じリスナーポートを複数のサービスで共有できます。

Network Load Balancer

Network Load Balancer は、Open Systems Interconnection (OSI) モデルの第 4 層で機能します。ロードバランサーが、リクエストを受信した後、フローハッシュルーティングアルゴリズムを使用して、デフォルトルールのターゲットグループからターゲットを選択します。

サービス用の IAM ロールの選択

awsvpc ネットワークモードを使用するタスク定義は、自動的に作成される AWSServiceRoleForECS サービスリンクロールを使用します。[詳細はこちら](#)

ロードバランサー名

Application Load Balancer を選択

作成したALB名を選択

ロードバランサーに追加のボタンを押す

リスナーポートは create new 80番を選択

テストリスナーは create new 8080番を選択

ロードバランサー用のコンテナ

php-sample-fargate : 80

プロダクションリスナーポート* 新規作成

プロダクションリスナープロトコル*

テストリスナー トラフィックをルーティングする前に、オプションのテストリスナーを使用して新しいアプリケーションリビジョンをテストします。

テストリスナーポート* 新規作成

テストリスナープロトコル*



Fargateでのサービス作成(6)

Additional configuration

To facilitate blue/green deployments with AWS CodeDeploy, you need two target groups. Each target group binds to a separate task set in the deployment. [Learn more](#)

Target group 1 name* tg-fargat-php-sample- ?

Target group 1 protocol* ?

Target type* ip ?

Path pattern* Path pattern: The first path pattern for a listener is the default path (/), which accepts all traffic that does not match another rule. You can later add additional patterns and priority values to this listener for other services.

Health check path* Additional health check options can be configured in the ELB console after you create your service.

Target group 2 name* tg-fargat-php-sample- ?

Target group 2 protocol* ?

Target type* ip ?

Path pattern* Path pattern: The first path pattern for a listener is the default path (/), which accepts all traffic that does not match another rule. You can later add additional patterns and priority values to this listener for other services.

Health check path* Additional health check options can be configured in the ELB console after you create your service.

今回は/index.phpというページを開くため、このページをヘルスチェックパスに設定

上記同様、/index.phpをヘルスチェックパスに設定



Fargateでのサービス作成(7)



Fargateでのサービス作成(8)

The screenshot shows the configuration of an automatic task scaling policy for an ECS service. The policy is named 'TargetTrackingPolicy' and uses the 'ECSServiceAverageCPUUtilization' metric. It has a target value of 10, a scale-out cool-down period of 60 seconds, and a scale-in cool-down period of 60 seconds. The minimum and maximum task counts are both set to 1.

Left panel (Service Auto Scaling 用 IAM ロール):

- タスクの最小数: 1
- タスクの必要数: 1
- タスク的最大数: 4

右 panel (自動タスクスケーリングポリシー):

- スケーリングポリシータイプ: ターゲットの追跡 (checked)
- ポリシー名*: TargetTrackingPolicy
- ECS サービスマトリクス: ECSServiceAverageCPUUtilization
- ターゲット値*: 10
- スケールアウトクールダウン時間: 60
- スケールインクールダウン時間: 60
- スケールインの無効化:

Bottom right corner:

- *必須
- キャンセル
- 戻る
- 次のステップ (Next Step) - highlighted with a red box

Annotations (orange boxes):

- タスクの最小数は 1
- タスクの必要数は 1
- タスクの最大数は 4
- ポリシー名は TargetTrackingPolicy
- ECSServiceAverageCPUUtilization を選択
- ターゲット値: 10
スケールアウトクールダウン時間 : 60
スケールインクールダウン時間 : 60

Fargateでのサービス作成(9)

ロードバランサー名 php-sample-01

ロードバランサ用のコンテナ php-sample-fargate

コンテナポート 80

プロダクションリスナーポート 80

ターゲットグループ 1 の名前 tg-fargate-php-sample-fargate-1

プロダクションリスナーのバスバ ターン /

プロダクションリスナーのヘルス チェックバス /index.php

ターゲットグループ 2 の名前 tg-fargate-php-sample-fargate-2

テストリスナーのバスパターン /

テストリスナーのヘルスチェック バス /index.php

Auto Scaling (オプション)

タスクの最小数 1

タスク的最大数 4

キャンセル 戻る サービスの作成

タスク Auto Scaling の作成

IAM Auto Scaling ロール: <create_new>

✓ 作成された IAM Auto Scaling ロール
IAM Auto Scaling ロールを作成しました。ポリシーのアタッチを待っています。表示: arn:aws:iam::██:role/ecsAutoscaleRole

IAM Auto Scaling ポリシー:

✓ IAM Auto Scaling ポリシーがアタッチされました
IAM Auto Scaling ポリシー ロールにアタッチ: arn:aws:iam::██:role/ecsAutoscaleRole。ポリシーの表示: AmazonEC2ContainerServiceAutoscaleRole

スケーラブルなターゲットの登録: service/fargate-cluster/php-sample-fargate

✓ スケーラブルなターゲットが登録されました。
スケーラブルなターゲット リソース ID に登録されています: service/fargate-cluster/php-sample-fargate

スケーリングポリシーの作成: TargetTrackingPolicy

✓ 作成されたターゲットの追跡ポリシー
ターゲットの追跡ポリシー作成済み: arn:aws:autoscaling:ap-northeast-1:██:scalingPolicy:86d466a0-bacc-4415-8f9b-a9dad517d6ad:resource/ecs/service/fargate-cluster/php-sample-fargate:policyName/TargetTrackingPolicy

戻る サービスの表示



Fargateでのサービス作成(10)

Clusters > fargate-cluster > Service: php-sample-fargate

Service : php-sample-fargate

Update

Delete

Cluster fargate-cluster

Desired count 1

Status ACTIVE

Pending count 1

Task definition php-sample-fargate

Running count 0

Service type REPLICA

Launch type FARGATE

Platform version LATEST(1.2.0)

Service role aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS

Details Tasks Events Auto Scaling Deployments Metrics Logs

Task status: Running Stopped

Filter in this page

< 1-1 > Page size 50

Task	Task Definition	Last status	Desired status	Group	Launch type	Platform version
------	-----------------	-------------	----------------	-------	-------------	------------------

34f8bb39-8986-489...	php-sample-fargate	PROVISIONING	RUNNING	service:php-sample-...	FARGATE	1.2.0
----------------------	--------------------	--------------	---------	------------------------	---------	-------

サービス作成が完了し、タスクが
1つ立ち上ることが確認

Fargateでのサービス作成(11)

Services -> EC2へ

コンソール
EC2

The screenshot shows the AWS EC2 Target Groups page. On the left, a sidebar lists various services: EC2 Dashboard, Events, Tags, Reports, Limits, Instances (selected), Instances, Launch Templates, Spot Requests, Reserved Instances, Dedicated Hosts, Scheduled Instances, Images, Elastic Block Store, Network & Security, Load Balancing (selected), Load Balancers, Target Groups (selected), Auto Scaling, Launch Configurations, Auto Scaling Groups, Systems Manager Services, and Run Command.

The main content area displays a table of registered targets for the target group "ecs-fargat-php-sample-fargate".

Name	Port	Protocol	Target type	VPC ID	Monitoring
ecs-fargat-php-sample-fargate	80	HTTP	ip	vpc-06f24252b9c8d0967	Enabled
ecs-hands-on-php-sample	80	HTTP	instance	vpc-06f24252b9c8d0967	Disabled

Below this, another table shows the "Registered targets" for the same target group:

IP address	Port	Availability Zone	Status
10.1.1.171	80	us-east-1a	healthy ⓘ

A callout box highlights the "Status" column in the second table, stating: "ALBのターゲットグループ配下にFargateモードで起動したタスク(に割り当てられたENI)がアタッチされ、StatusがHealthyになっていることを確認".

Red arrows point from the sidebar's "Target Groups" selection to the "Targets" tab in the top navigation and to the "Status" column in the "Registered targets" table.

サービス作成(12)



後片付け

作成したリソース削除

1. ECRレポジトリ

- php-sample-**xx**

2. ECSタスク定義

- 以下のタスク定義中のrevisionを deregisterする
- php-sample-fargate

3. ECSクラスタ削除

- サービスとタスクを削除した上でクラスタ削除

4. ALB

- php-sample-**xx**

9. ALBターゲットグループ

- tg-fargate-php-sample-fargate-1
- tg-fargate-php-sample-fargate-2

10. Cloud9環境

11. Security Group

- ALBで作成したSG (defaultは消せない)

12. CloudWatch logs

- ロググループ

13. CloudFormation

- VPCを作成したスタック

14. 各種IAMロール、ポリシー

- “ecs” を含むもの（ロールのみ）
- “sample” を含むもの（ロールとポリシー）

Thank you!