



# チーム開発演習 シェアードシヨツプ 通販システム デプロイ手順書

# 目次

1	はじめに	3
1-1	作業完了目標	3
1-2	作業全体の流れ	3
1-3	関連用語の説明	4
2	必要なデータの準備	5
2-1	デプロイ用の SQL 文の準備	5
2-2	デプロイ用のソースコード、画像ファイルなどの準備	5
2-3	AWS 環境設定に必要な情報と環境の準備	5
3	AWS サービスでの環境構築	6
3-1	環境構築の全体像	6
3-2	環境構築作業の流れ	7
3-3	環境構築作業の実施	8
4	Web アプリケーションの準備	40
4-1	Web アプリケーションの準備作業の流れ	40
4-2	Web アプリケーションの準備作業の実施	40
5	Web アプリケーションのデプロイ	48
5-1	Web アプリケーションのデプロイ作業の流れ	48
5-2	Web アプリケーションのデプロイ作業の実施	48
6	成果報告会当日の作業	60

# 1 はじめに

本資料では、開発演習の成果物をクラウド環境(AWS)にデプロイする手順について説明します。

AWS 環境で公開した Web アプリケーションは、成果報告会でのデモンストレーションに使用します。成果報告会前にデプロイを完了させておきましょう。

なお、バグ修正まで完了したソースコードのみをデプロイしてください。バグ修正が完了しなかった場合は、デプロイはせずにローカル環境でデモンストレーションを実施してください。

## 1-1 作業完了目標

以降の手順にしたがって実施した作業が完了すると、AWS サービスを利用して Web アプリケーション(ショッピングサイト)を公開することができます。



## 1-2 作業全体の流れ

デプロイ作業は以下の流れで行います。

1. 実行環境に必要なデータを用意する
2. AWS を利用して、Web アプリケーションの実行環境を用意する
3. Web アプリケーションを準備する
4. 実行環境へデプロイする
5. 実行環境にブラウザで接続し、動作を確認する

## 1 - 3 関連用語の説明

### JAR ファイルとは

JAR ファイル(Java Archive ファイル)とは、ソースファイル(.java)をコンパイルして作成されたクラスファイル(.class)、設定ファイル、および HTML、画像などの静的ファイルを 1 つにまとめて圧縮したファイルのことです。拡張子は「.jar」です。

また、同じ用途で WAR ファイルという圧縮ファイルを作成することも開発現場ではあります。WAR ファイルのファイル拡張子は「.war」です。

### tar ファイルとは

tar ファイルとは、複数のファイルを 1 つにまとめて扱うアーカイブ・ファイル形式の 1 つで、Linux 環境でよく利用される形式です。拡張子は「.tar」です。

### ビルドとは

JAR ファイルなどの圧縮ファイルを作成する操作をビルド(build)と言います。また、ビルドして作成されたファイルのことを総称してビルドファイルと呼びます。

### デプロイとは

ビルドファイルをサーバに配置して、サーバ内でビルドファイル中のプログラムを実行可能な状態にすることをデプロイ(deploy)と呼びます。

## 2 必要なデータの準備

### 2-1 デプロイ用の SQL 文の準備

デモンストレーションを行う際に必要となるテーブル、シーケンスを作成し、必要なデータを登録するための SQL 文を準備してください。なお、AWS 環境では、「ユーザ作成(CREATE USER 文)」と「権限設定(GRANT 文)」、「PDB への切り替え(ALTER SESSION 文)」は不要です。

SQL 文は、テキストエディタで編集し「AWS 環境用 SQL 文.txt」というファイル名で保存しておきましょう。この「AWS 環境用 SQL 文.txt」が AWS 環境で行うデモンストレーション用の SQL です。なお、このデモンストレーション用のデータは各自 AWS 環境用に準備したい任意のデータ SQL を作成します。チームで作成したいデータが特になければ、ば開発時に使用したサンプルデータ SQL を用意してください。

### 2-2 デプロイ用のソースコード、画像ファイルなどの準備

ソースコード、画像ファイルについて下記の準備ができていることを確認しましょう。

- ・ 開発演習の動作検証が終了しバグ修正が完了していること
- ・ コントローラクラス内のメソッド戻り値やテンプレートファイルのパスが相対パス指定であること
  - ※パスの指定が異なるとローカル環境で正常に動作していても AWS 環境では正常に動作しません。
  - 注意 1: コントローラクラス内のメソッド戻り値の先頭に / を記述していないこと
    - NG-記述例 `return "/client/item/list";`
    - OK-記述例 `return "client/item/list";`
  - コントローラクラス内のメソッド戻り値でリダイレクト先が相対パス指定であること
    - NG-記述例 `return "redirect:http://localhost:55000/shared_shop/";`
    - OK-記述例 `return "redirect:/";`
  - テンプレートファイル内のパスが相対パス指定であること
    - NG-記述例 `<a href="http://localhost:55000/shared_shop/">トップ画面</a>`
    - OK-記述例 `<a href="/shared_shop/">トップ画面</a>`
    - OK-記述例 `<a th:href="@{/}">トップ画面</a>`
- ・ データベースに登録する画像ファイル名と同じ名前のファイルを Eclipse のプロジェクト名直下の images フォルダに格納していること

### 2-3 AWS 環境設定に必要な情報と環境の準備

AWS 環境設定のために、下記の準備ができていることを確認しましょう。

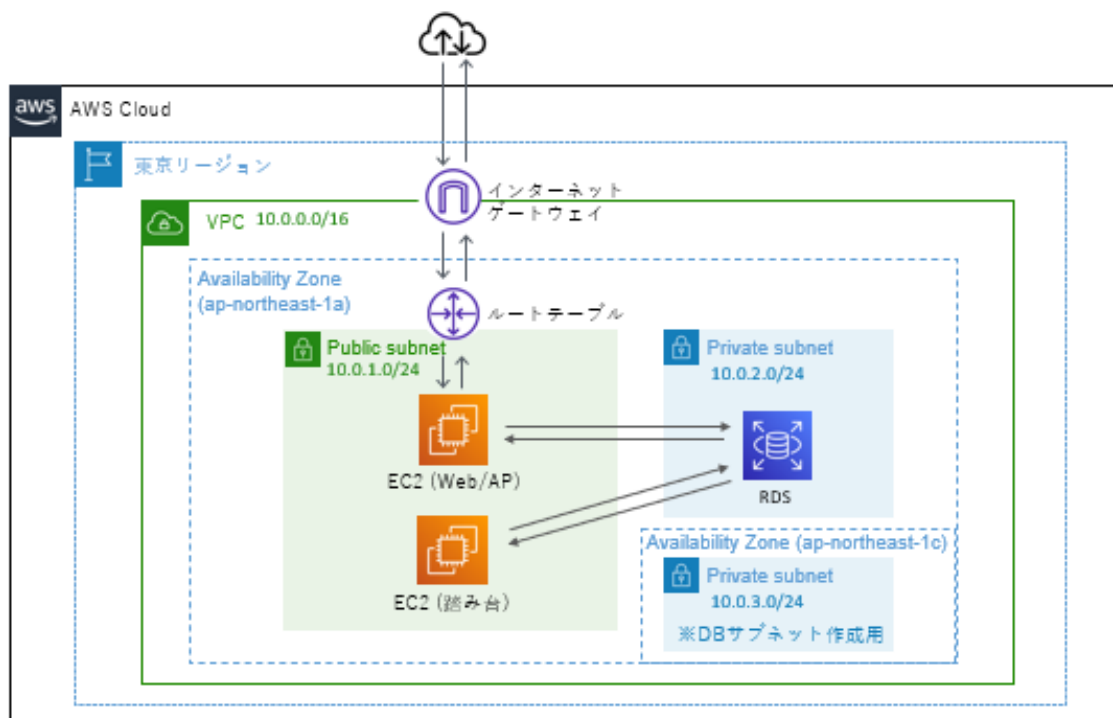
- ・ 安心サンドボックスへの接続アカウントとパスワードを準備できていること
- ・ 安心サンドボックスから AWS マネジメントコンソールへサインインを確認できていること
- ・ TeraTerm が作業用 PC にインストールされていること
- ・ SQL Developer が作業用 PC にインストールされていること
- ・ 以下のファイルを LMS よりダウンロードできていること
  - `oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm`
  - `oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm`
  - `jdk-16.0.2_linux-x64_bin.rpm`
- ・ AWS 環境用 SQL 文.txt が用意できていること
- ・ AWS 環境設定中に必要な情報をメモできる状態にしていること
  - テキストエディタを起動しているなど、各自のやりやすい方法で準備してください。

## 3 AWS サービスでの環境構築

以降の作業は、安心サンドボックス(<https://sandbox.shared.jp/auth/signIn>)を利用して AWS マネジメントコンソールへサインインしていることを前提としています。

### 3-1 環境構築の全体像

以下の構成図に従って各 AWS サービスを構築して Web アプリケーションの動作環境を構築します。



※上図はイメージです。IP アドレスなどが実際とは異なります。

### 3-2 環境構築作業の流れ

AWS サービスでの環境構築作業は下記の流れで行います。

1. 「ネットワーク環境」 VPC を作成する
2. 「サーバ環境」 EC2 インスタンス(Web/AP)を作成する
3. 「サーバ環境」 EC2 インスタンス(踏み台)を作成する
4. 「データベース環境」 RDS を作成する
5. EC2 インスタンス(Web/AP)と RDS を接続する
6. EC2 インスタンス(踏み台)と RDS を接続する

なお、構築作業時に入力する設定値は最低限の動作を確保するための値を採用しています。

### 3-3 環境構築作業の実施

#### 作業1.VPC(ネットワーク)を作成する

##### 【作業の目的】

パブリックサブネットはインターネットにつながったサブネットワークです。パブリックサブネットは、VPC と呼ばれるネットワーク上のサブネットワークとして作成します。

このサブネットで WEB アプリケーションをデプロイ(配置)した EC2 インスタンスを起動すると、インターネット経由で WEB アプリケーションを利用することができるようになります。

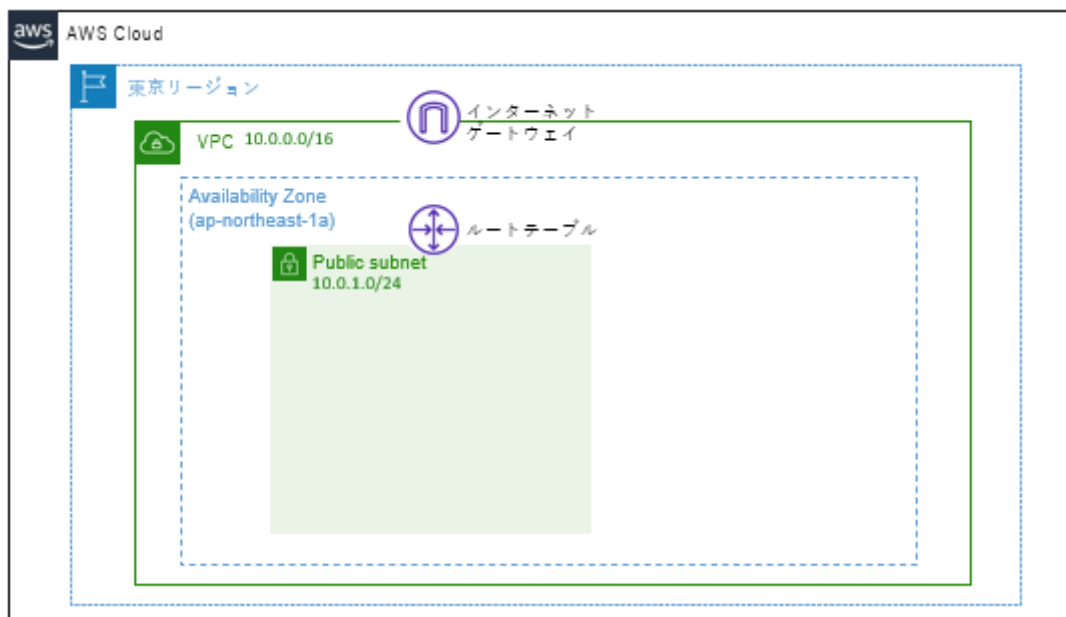
また、このサブネットで踏み台用の EC2 インスタンスを起動すると、データベースを管理・操作することができます。

2022 年 6 月の AWS のユーザインターフェイス刷新により、VPC を作成すると、自動でサブネット、ルートテーブル、ネットワーク接続(igw)も作成されます。

##### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ VPC(ネットワーク)の作成
- ・ サブネット、ルートテーブル、ネットワーク接続(igw)の設定の確認





### 【作業手順】

手順1. マネジメントコンソールでリージョンが [東京]であることを確認する

手順2. サービス一覧から[VPC]をクリックする

見つからない場合、[検索]欄に「VPC」と入力してサービスを探しましょう。

手順3. [VPC を作成] をクリックして VPC の作成画面へ遷移する

手順4. VPC の作成画面で、以下の通りに設定を行って [VPC を作成] をクリックする

- 作成するリソース: VPC など
- 自動生成: チェックする
- 名前タグ: [YYYYMMDDDeployHandsOnVPC] (YYYYMMDDは本日の日付)
- IPv4 CIDR ブロック: [10.0.0.0/16]

### VPC の設定

**作成するリソース** 情報  
VPC リソースのみ、または VPC と他のネットワークリソースを作成します。

☐ VPC のみ

☒ VPC など

**名前タグの自動生成** 情報  
Name タグの値を入力します。この値は、VPC 内のすべてのリソースの Name タグを自動生成するのに使用されます。

☒ 自動生成

20220621DeployHandsOnVPC

**IPv4 CIDR ブロック** 情報  
CIDR 表記を使用して VPC の開始 IP とサイズを決定します。

10.0.0.0/1665,536 IPs

**IPv6 CIDR ブロック** 情報

☒ IPv6 CIDR ブロックなし

☐ Amazon 提供の IPv6 CIDR ブロック

**テナンシー** 情報

デフォルト▼

手順5.「VPC エンドポイント」は「なし」を選択する  
それ以外の設定は全て初期値(デフォルト)のままにしてください。

**アベイラビリティーゾーン (AZ) 情報**  
サブネットをプロビジョニングする AZ の数を選択します。可用性を高めるには、少なくとも2つの AZ をお勧めします。

1	2	3
---	---	---

▶ AZ のカスタマイズ

**パブリックサブネットの数 情報**  
VPC に追加するパブリックサブネットの数。インターネット経由でパブリックにアクセス可能にする必要があるウェブアプリケーションには、パブリックサブネットを使用します。

0	2
---	---

**プライベートサブネットの数 情報**  
VPC に追加するプライベートサブネットの数。プライベートサブネットを使用して、パブリックアクセスを必要としないバックエンドリソースを保護します。

0	2	4
---	---	---

▶ サブネット CIDR ブロックをカスタマイズ

**NAT ゲートウェイ (\$) 情報**  
NAT ゲートウェイを作成するアベイラビリティーゾーン (AZ) の数を選択します。NAT ゲートウェイごとに料金が発生することに注意してください。

なし	1 AZ 内	AZ ごとに 1
----	--------	----------

**VPC エンドポイント 情報**  
エンドポイントは、VPC から S3 に直接アクセスすることで、NAT ゲートウェイの料金を削減し、セキュリティを向上させるのに役立ちます。デフォルトでは、フルアクセスポリシーが使用されます。このポリシーはいつでもカスタマイズできます。

なし	S3 ゲートウェイ
----	-----------

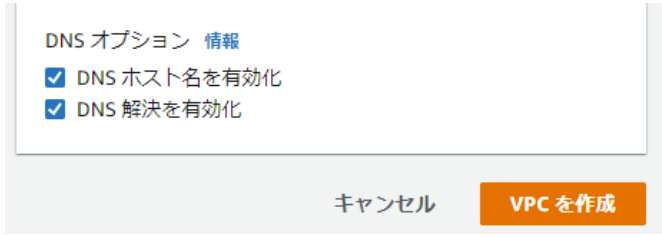
**DNS オプション 情報**  
☒ DNS ホスト名を有効化  
☒ DNS 解決を有効化

手順6.右側のプレビューで作成される VPC、サブネット、ルートテーブル、ネットワーク接続(igw)を確認する  
※以降の手順で利用します。それぞれの名前をメモしておくことをお勧めします。



- ・ VPC : 1 つ作成される
- ・ サブネット : アベイラビリティゾーン 1a と 1c にそれぞれ 2 つ作成される
- ・ ルートテーブル : 3 つのルートテーブルが作成される
- ・ ネットワーク接続(igw) : 1 つのネットワーク接続が作成される

手順7.設定内容を確認し[VPC を作成]をクリックする



ここまでで、VPC(ネットワーク)の作成は完了です。

## 作業2.EC2 インスタンス(Web/AP)を作成する

### 【作業の目的】

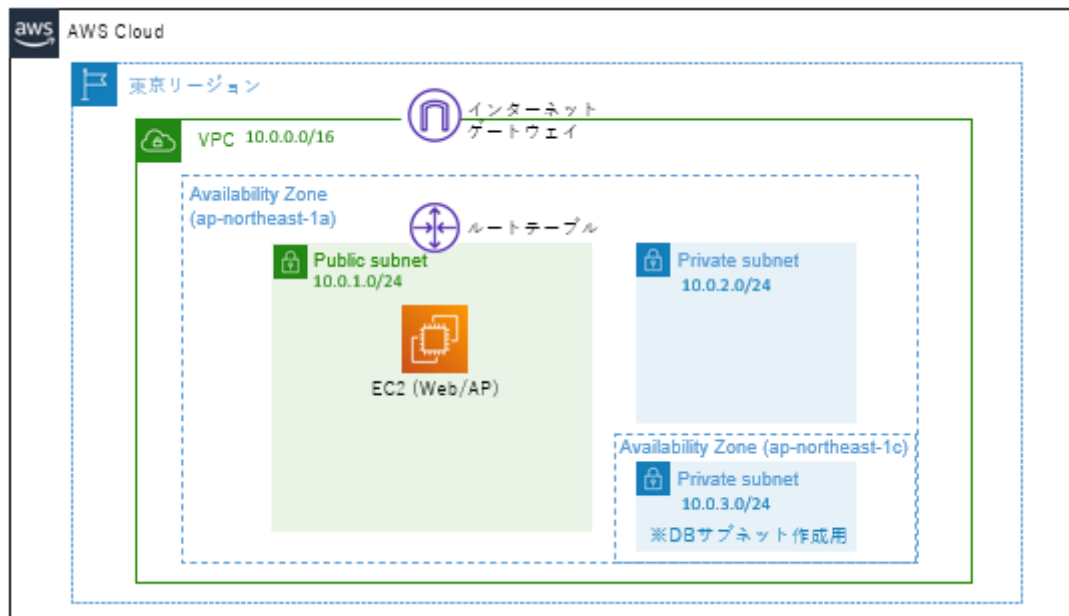
WEB アプリケーションをデプロイ(配置)し、実行するための EC2 インスタンスを作成します。

この EC2 インスタンスは、インターネット経由で WEB アプリケーションを利用できるようにするため、パブリックサブネットに作成します。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ ネットワーク(VPC)のパブリックサブネットに、EC2 インスタンスを作成
- ・ 作成する EC2 インスタンスへのアクセス制御を行うためのセキュリティグループを作成
- ・ 作成する EC2 インスタンスへの接続に必要な鍵(キーペア)を作成



### 【作業手順】

手順1.サービスの一覧から[EC2]をクリックする

見つからない場合、[検索]欄に「EC2」と入力してサービスを探しましょう。

手順2.[インスタンスを起動]をクリックする



## 手順3.EC2 インスタンスの作成ウィザードで以下の通りに設定する

以下の設定を行きましょう。下記に記載していない項目は、初期値(デフォルト)のままとします。

- 名前: **YYYYMMDD**DeployHandsOnEC2
- クイックスタート: Amazon Linux
- キーペア
  - ※「新しいキーペアの作成」を押下し、下記の項目を入力して「キーペアを作成」を押下
  - キーペア名: **YYYYMMDD**DeployHandsOnKey
  - キーペアのタイプ: 「RSA」
  - プライベートキーファイル形式: 「.pem」
- VPC: **YYYYMMDD**DeployHandsOnVPC-vpc
- サブネット: **YYYYMMDD**DeployHandsOnVPC-subnet-public2-ap-northeast-1c
  - ※パブリックサブネット
- パブリック IP の自動割り当て: 有効化
- ファイアウォール(セキュリティグループ): **YYYYMMDD**DeployHandsOnWebSecurityGroup
- セキュリティグループルール:
  - タイプ: ssh
  - ソースタイプ: 自分の IP
- ストレージ: 1x [ **20** ] GiB **gp2**

### インスタンスを起動 情報

Amazon EC2 では、AWS クラウドで実行される仮想マシン (インスタンス) を作成できます。以下の簡単なステップに従ってすばやく開始できます。

#### 名前とタグ 情報

名前

20220621DeployHandsOnEC2

さらにタグを追加

#### ▼ アプリケーションおよび OS イメージ (Amazon マシンイメージ) 情報

AMI は、インスタンスの起動に必要なソフトウェア設定 (オペレーティングシステム、アプリケーションサーバー、アプリケーション) を含むテンプレートです。お探しのものが以下に表示されない場合は、AMI を検索または参照してください。

最新

クイックスタート

Amazon Linux

aws

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

その他の AMI を開

覧する

AWS、Marketplace、

コミュニティからの

AMI を含む

Amazon マシンイメージ (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type

ami-0b7546e839d7ace12 (64 ビット (x86)) / ami-004332b441f90509b (64 ビット (Arm))

仮想化: hvm    ENA 有効: true    ルートデバイスタイプ: ebs

無料利用枠の対象 ▼

ネットワーク設定を編集する際には右上の[編集]ボタンをクリックしてください。

▼ インスタンスタイプ 情報

インスタンスタイプ

t2.micro  
ファミリー: t2 1 vCPU 1 GiB メモリ  
オンデマンド Linux 料金: 0.0152 USD 1 時間あたり  
オンデマンド Windows 料金: 0.0198 USD 1 時間あたり

無料利用枠の対象

インスタンスタイプを比較

▼ キーペア (ログイン) 情報

キーペアを使用してインスタンスに安全に接続できます。インスタンスを起動する前に、選択したキーペアにアクセスできることを確認してください。

キーペア名 - 必須

20220621DeployHandsOnKey

新しいキーペアの作成

▼ ネットワーク設定

VPC - 必須 情報

vpc-05be51b8cc9142a4d (20220621DeployHandsOnVPC-vpc)

10.0.0.0/16

サブネット 情報

subnet-08e55bb3e4cfa8b71

20220621DeployHandsOnVPC-subnet-public2-ap-northeast-1c

VPC: vpc-05be51b8cc9142a4d 所有者: 768792468715  
アベイラビリティゾーン: ap-northeast-1c 利用可能な IP アドレス: 4090

パブリック IP の自動割り当て 情報

有効化

ここをクリックして新しいキーペアを作成してください。

ファイアウォール (セキュリティグループ) 情報

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ 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

セキュリティグループ名 - 必須

20220621DeployHandsOnWebSecurityGroup

このセキュリティグループはすべてのネットワークインターフェイスに追加されます。セキュリティグループが作成された後で名前を編集することはできません。最大長は 255 文字です。有効な文字は a~z、A~Z、0~9、スペース、および \_、/、()、@、!、+、&、[]、\$。

説明 - 必須 情報

launch-wizard created 2022-06-21T02:53:39.532Z

インバウンドセキュリティグループのルール

▼ セキュリティグループルール 1 (TCP, 22, 1 [REDACTED])

タイプ 情報

ssh

プロトコル 情報

TCP

ポート範囲 情報

22

ソースタイプ 情報

自分の IP

ソース 情報

Q CIDR、プレフィックスリスト

[REDACTED]

説明 - optional 情報

例: 管理者のデスクトップの SSH

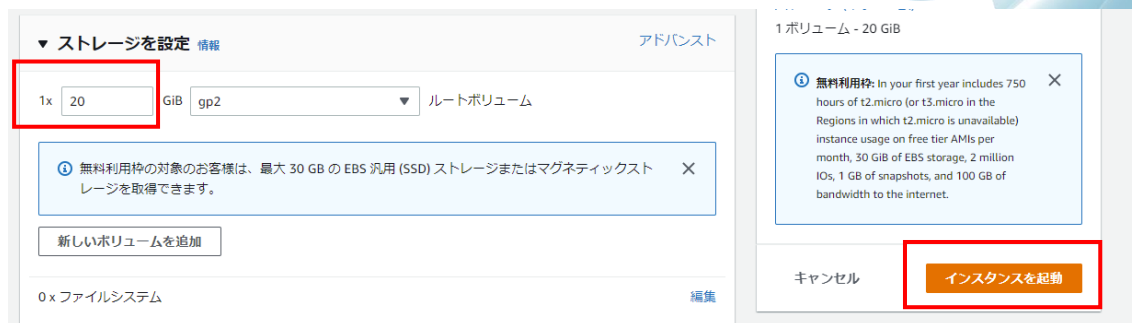
Add security group rule

▶ 高度なネットワーク設定

黒塗りの部分には、自分の PC の IP アドレスが表示されます。

15 / 60

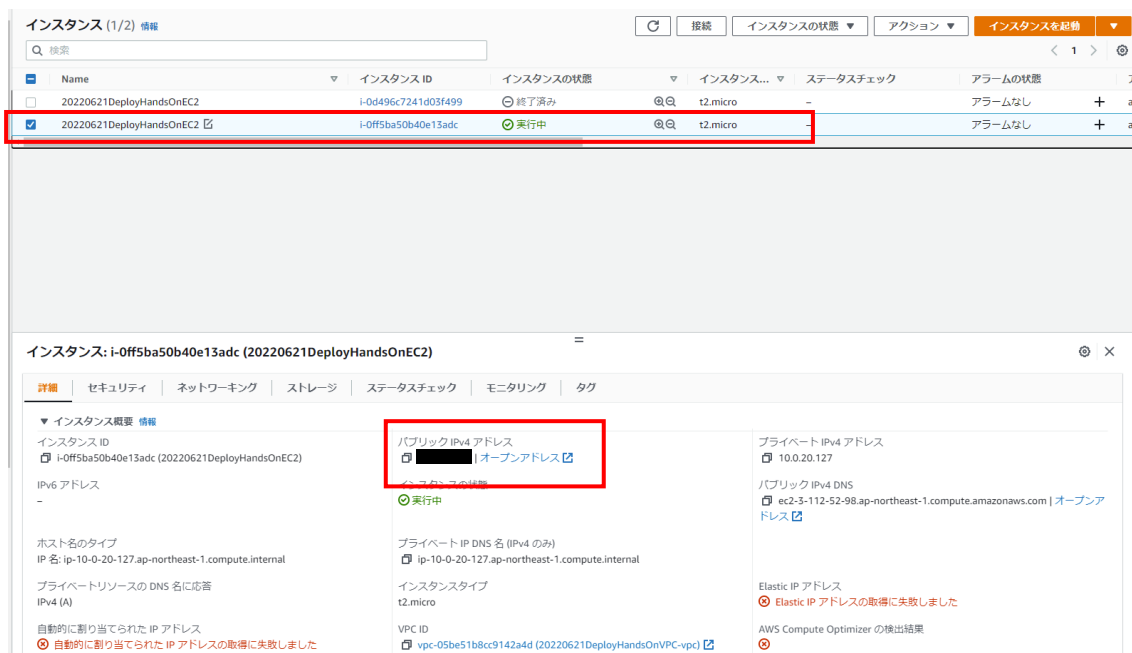
手順4. 設定内容を確認して「インスタンスを起動」をクリックする



手順5. 作成した EC2 インスタンスのパブリック IP アドレスを確認する



インスタンスの一覧から作成した EC2 インスタンスをチェックすると、画面下部にインスタンスの詳細情報が表示されます。その中に[パブリック IPv4 アドレス]が表示されています。

※パブリック IP アドレスは、EC2 インスタンスへの接続や Web アプリケーションへの接続時に使用しますので、メモしておいてください。この手順書ではセキュリティの都合上、見えないようにしています。



Name	インスタンス ID	インスタンスの状態	インスタンス...	ステータスチェック	アラームの状態	アクション
20220621DeployHandsOnEC2	i-0d496c7241d03f499	終了済み	t2.micro	-	アラームなし	+
20220621DeployHandsOnEC2	i-0ff5ba50b40e13adc	実行中	t2.micro	-	アラームなし	+

インスタンス: i-0ff5ba50b40e13adc (20220621DeployHandsOnEC2)	
<b>インスタンス概要</b> インスタンス ID i-0ff5ba50b40e13adc (20220621DeployHandsOnEC2) IPv6 アドレス - ホスト名のタイプ IP 名: ip-10-0-20-127.ap-northeast-1.compute.internal プライベートリソースの DNS 名に IPv4 (A) 自動的に割り当てられた IP アドレス 自動的に割り当てられた IP アドレスの取得に失敗しました	<b>パブリック IPv4 アドレス</b>  <span style="background-color: black; color: black;">XXXXXXXXXX</span> <a href="#">オープンアドレス</a> <b>インスタンスの状態</b>  実行中 <b>プライベート IP DNS 名 (IPv4 のみ)</b> IP-10-0-20-127.ap-northeast-1.compute.internal <b>インスタンスタイプ</b> t2.micro <b>VPC ID</b> vpc-05be51b8c9142a4d (20220621DeployHandsOnVPC-vpc)
	<b>プライベート IPv4 アドレス</b> 10.0.20.127 <b>パブリック IPv4 DNS</b> ec2-3-112-52-98.ap-northeast-1.compute.amazonaws.com <a href="#">オープンアドレス</a> <b>Elastic IP アドレス</b> Elastic IP アドレスの取得に失敗しました <b>AWS Compute Optimizer の検出結果</b>

ここまでで、EC2 インスタンス(Web/AP)の作成は完了です。



## 作業3.EC2 インスタンス(踏み台)を作成する

### 【作業の目的】

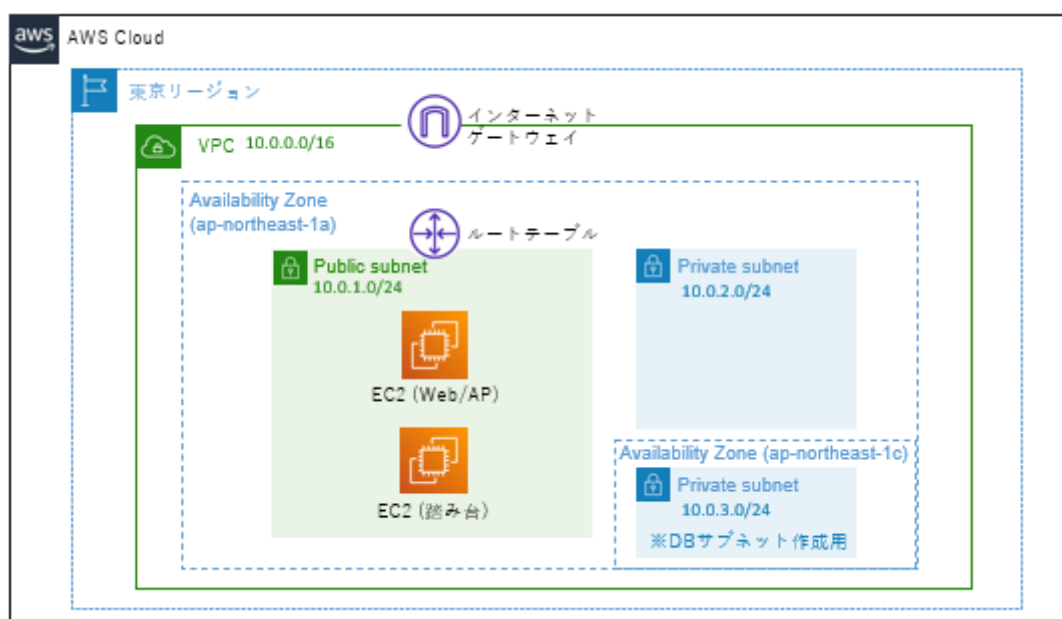
RDS に接続し、RDS のデータベースの管理や操作を行うための EC2 インスタンスを、パブリックサブネットに作成します。

RDS はプライベートサブネット内にあるため、PC からインターネット経由で直接接続することはできません。そのため、一旦 PC からインターネット経由で EC2 インスタンスに接続し、EC2 インスタンスから RDS に間接的に接続するという方法を取ります。このように目標の RDS に接続するために、この EC2 インスタンスは「踏み台」として使われます。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ ネットワーク(VPC)のパブリックサブネットに、EC2 インスタンスを作成
- ・ 作成する EC2 インスタンスへのアクセス制御を行うためのセキュリティグループを作成



## 【作業手順】

手順1. サービスの一覧から [EC2] をクリックする

手順2. [インスタンスの起動] をクリックする

手順3. EC2 インスタンスの作成ウィザードで以下の通りに設定する

- 名前: **YYYYMMDD**DeployHandsOnBastionInstance
- キーペア: **YYYYMMDD**DeployHandsOnKey  
※前回のステップの「EC2 インスタンスの作成(Web/AP)」で作成した既存のキーペア
- VPC: **YYYYMMDD**DeployHandsOnVPC-vpc
- サブネット: **YYYYMMDD**DeployHandsOnVPC-subnet-public2-ap-northeast-1c  
※パブリックサブネット
- パブリック IP の自動割り当て: 有効化
- ファイアウォール(セキュリティグループ):
- **YYYYMMDD**DeployHandsOnBastionSecurityGroup
- セキュリティグループルール:
  - タイプ: ssh
  - ソースタイプ: 自分の IP
- ストレージ: 1x [ **8** ] GiB **gp2**

## インスタンスを起動 情報

Amazon EC2 では、AWS クラウドで実行される仮想マシン (インスタンス) を作成できます。以下の簡単なステップに従ってすばやく開始できます。

### 名前とタグ 情報

名前

20220621DeployHandsOnBastionInstance

[さらにタグを追加](#)

### ▼ アプリケーションおよび OS イメージ (Amazon マシンイメージ) 情報

AMI は、インスタンスの起動に必要なソフトウェア設定 (オペレーティングシステム、アプリケーションサーバー、アプリケーション) を含むテンプレートです。お探しのものが以下に表示されない場合は、AMI を検索または参照してください。

🔍 何千ものアプリケーションイメージと OS イメージを含むカタログ全体を検索します。

最新

クイックスタート

Amazon  
Linux



Ubuntu



Windows



Red Hat



SUSE Linux



[その他の AMI を閲覧する](#)

AWS、Marketplace、コミュニティからの AMI を含む

Amazon マシンイメージ (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type  
ami-0b7546e839d7ace12 (64 ビット (x86)) / ami-004332b441f90509b (64 ビット (Arm))  
仮想化: hvm ENA 有効: true ルートデバイスタイプ: ebs

無料利用枠の対象 ▼

19 / 60



▼ ストレージを設定 情報 アドバンスト

1x 8 GiB gp2 ルートボリューム

無料利用枠の対象のお客様は、最大 30 GB の EBS 汎用 (SSD) ストレージまたはマグネティックストレージを取得できます。

新しいボリュームを追加

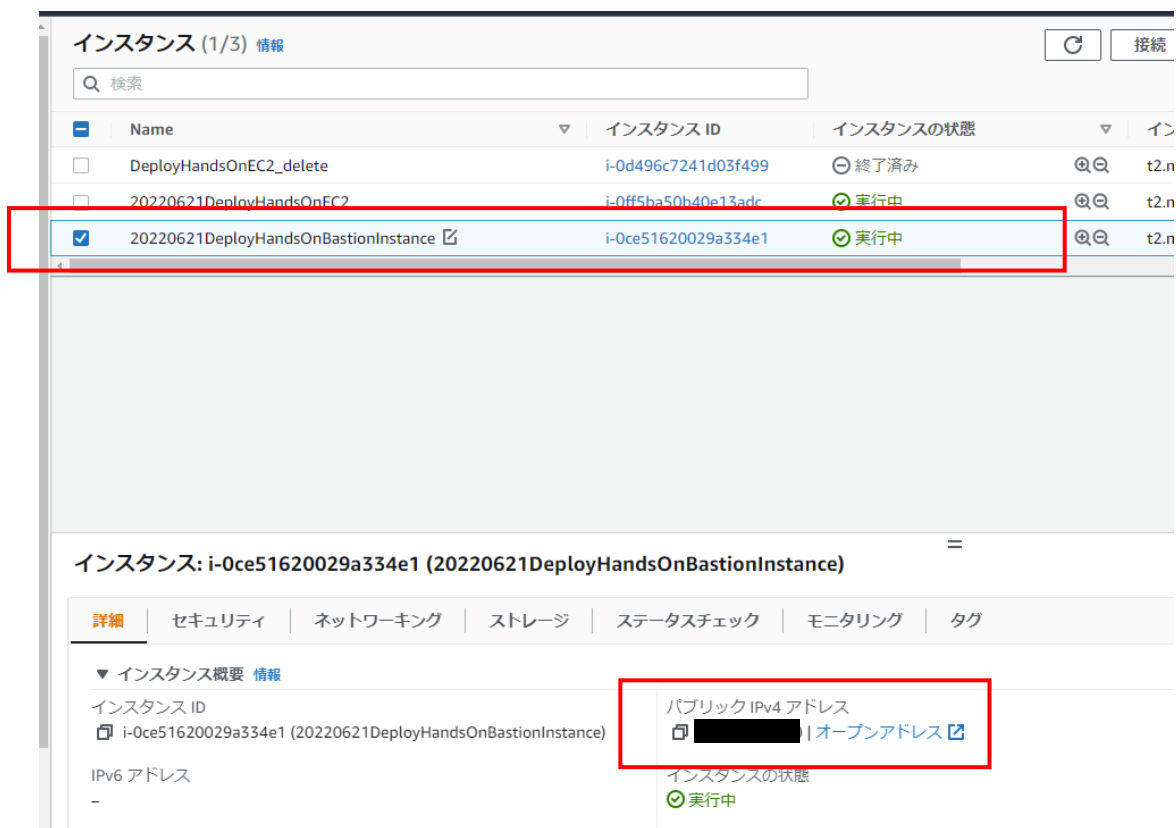
0 x ファイルシステム 編集

無料利用枠: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

キャンセル インスタンスを起動

手順4.作成した EC2 インスタンスのパブリック IP アドレスを確認する  
インスタンスの一覧から作成した EC2 インスタンスをチェックすると、画面下部にインスタンスの詳細情報が表示されます。その中に[パブリック IPv4 アドレス]が表示されています。

※パブリック IP アドレスは、EC2 インスタンスへの接続や RDS への接続時に使用しますので、メモしておいてください。



インスタンス (1/3) 情報

検索

Name	インスタンス ID	インスタンスの状態	イン
DeployHandsOnEC2_delete	i-0d496c7241d03f499	終了済み	t2.n
20220621DeployHandsOnEC2	i-0ff5ba50b40e13adr	実行中	t2.n
20220621DeployHandsOnBastionInstance	i-0ce51620029a334e1	実行中	t2.n

インスタンス: i-0ce51620029a334e1 (20220621DeployHandsOnBastionInstance)

詳細 セキュリティ ネットワーキング ストレージ ステータスチェック モニタリング タグ

▼ インスタンス概要 情報

インスタンス ID  
i-0ce51620029a334e1 (20220621DeployHandsOnBastionInstance)

パブリック IPv4 アドレス  
[Redacted] | [オープンアドレス](#)

IPv6 アドレス  
-

インスタンスの状態  
実行中

手順5.セキュリティグループ名と ID を確認する  
次の作業で EC2 インスタンス設定時に作成したセキュリティグループを使用します。  
セキュリティグループ一覧画面で、Web/AP 用のセキュリティグループ名と ID、踏み台用のセキュリティグループ名と ID をそれぞれ識別できるようにメモしておきましょう。

ここまでで、EC2 インスタンス(踏み台)の作成は完了です。

## 作業4 .RDS を作成する

### 【作業の目的】

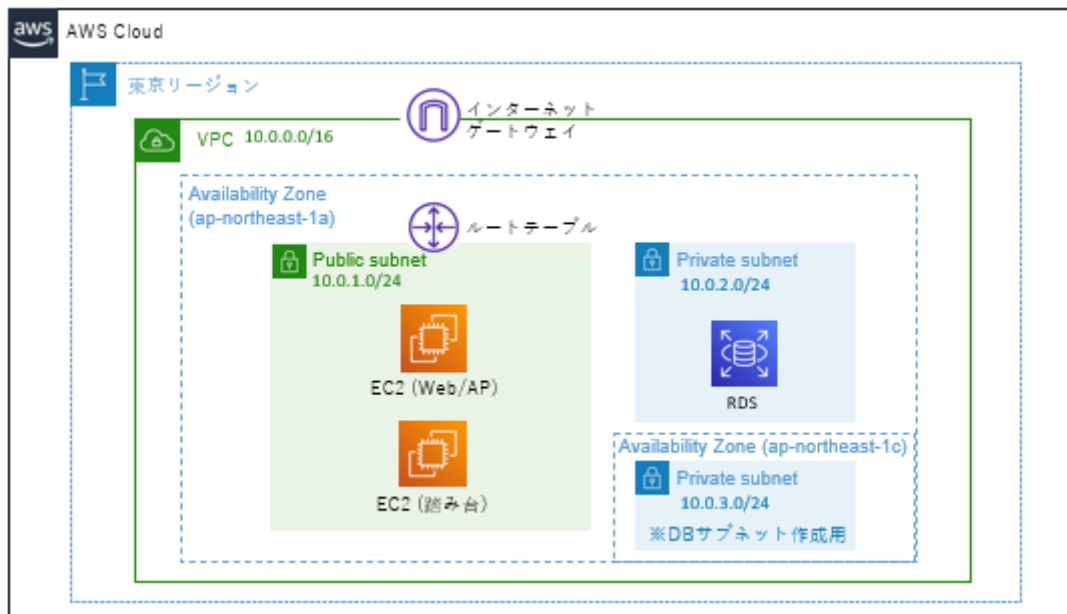
WEB アプリケーションで使用するデータベースを利用するための RDS インスタンスを、プライベートサブネットに作成します。

作成した RDS インスタンスには、WEB アプリケーションを実行する EC2 インスタンスと、データベースを管理・操作するための踏み台用 EC2 インスタンスから接続できるように設定します。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ RDS インスタンスを配置する DB サブネットグループを作成
  - ネットワーク(VPC)に作成した2つのプライベートサブネットを使用する
- ・ RDS へのアクセス制御を行うため、セキュリティグループを作成
- ・ EC2 インスタンス(WEB/AP)から RDS に接続する設定を追加
  - ・ EC2 インスタンス(踏み台)から RDS に接続する設定を追加
- ・ 作成した DB サブネットグループのプライベートサブネットに、RDS インスタンスを作成



### 【作業手順】

手順1.RDS ダッシュボードから [サブネットグループ] をクリックする  
DB インスタンスを配置するためのサブネットグループの作成を行います。

手順2.[DB サブネットグループの作成]をクリックする



手順3.DB サブネットグループの作成画面で以下の情報を入力する

- 名前:[deployhandsonsubnetgroup]
- 説明:[DB Subnet Group for Deploy Hands On]
- VPC:[YYYYMMDDDeployHandsOnVPC-vpc]

**サブネットグループの詳細**

名前  
サブネットグループの作成後に名前を変更することはできません。  
  
1～255 文字にする必要があります。英数字、スペース、ハイフン、アンダースコア、ピリオドを使用できます。

説明

VPC  
DB サブネットグループに使用するサブネットに対応する VPC 識別子を選択します。サブネットグループが作成された後、別の VPC 識別子を選択することはできません。

手順4. [サブネットを追加] 欄で以下の設定を行い [作成] をクリックする

- アベイラビリティーゾーン:[ap-northeast-1a] [ap-northeast-1c]
- サブネット:
  - ap-northeast-1c のパブリックサブネット
  - ap-northeast-1a のパブリックサブネット

### サブネットを追加

アベイラビリティーゾーン  
追加するサブネットを含まないアベイラビリティーゾーンを選択します。

アベイラビリティーゾーンを選択

ap-northeast-1a X ap-northeast-1c X

サブネット  
追加するサブネットを選択します。リストには、選択したアベイラビリティーゾーンのサブネットが含まれます。

サブネットを選択

subnet-08e55bb3e4cfa8b71 (10.0.16.0/20) X  
subnet-005f2e20c08121a2d (10.0.0.0/20) X

#### 選択したサブネット (2)

アベイラビリティーゾーン	サブネット ID	CIDR ブロック
ap-northeast-1c	subnet-08e55bb3e4cfa8b71	10.0.16.0/20
ap-northeast-1a	subnet-005f2e20c08121a2d	10.0.0.0/20

キャンセル 作成

手順5. EC2 のメニュー一覧から [セキュリティグループ] をクリックする  
RDS に設定するセキュリティグループを作成するための作業です。

手順6. [セキュリティグループの作成] をクリックする

手順7.以下の通りに設定を行い [セキュリティグループを作成] をクリックする

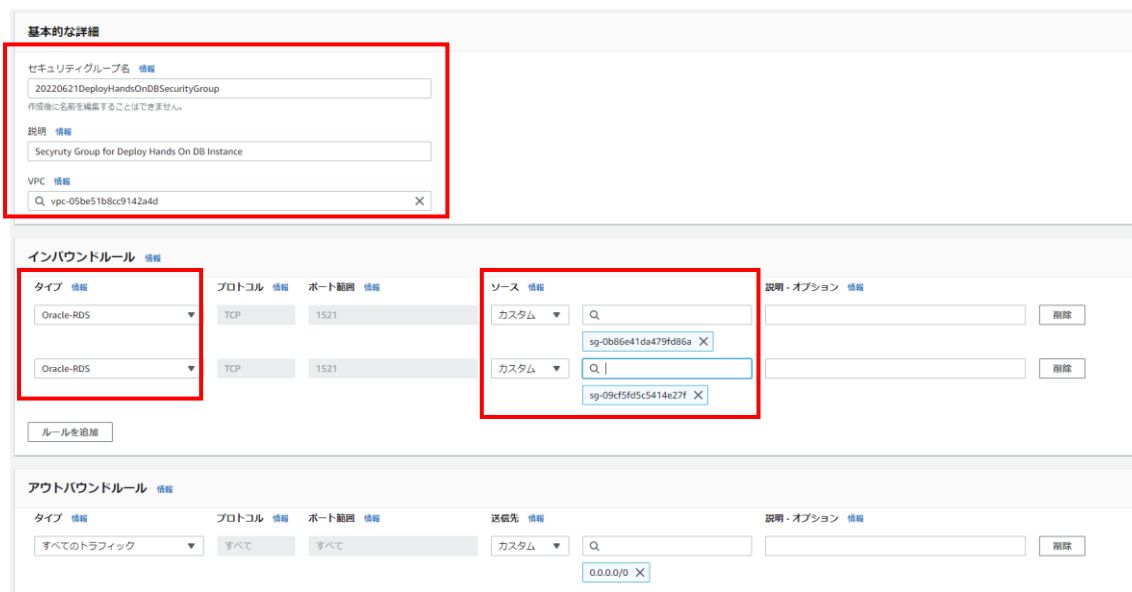
- 基本的な詳細
  - セキュリティグループ名:[YYYYMMDDDeployHandsOnDBSecurityGroup]  
(YYYYMMDDは本日の日付)
  - 説明:[Security Group for Deploy Hands On DB Instance]
  - VPC:[YYYYMMDDDeployHandsOnVPC-vpc] ※作成済みの VPC
- インバウンドルール
 

※EC2 インスタンス(WEB/AP)と EC2 インスタンス(踏み台)から、RDS に接続することを許可するルールです。

  - ① インバウンドルール 1
    - タイプ 1:[Oracle-RDS]
    - リソースタイプ 1:[カスタム]
    - ソース 1:[YYYYMMDDHandsOnWebSecurityGroup のセキュリティグループ ID]
  - ② インバウンドルール 2
 

※[ルールを追加]をクリックし、インバウンドルール 2 を設定します。

    - タイプ 2:[Oracle-RDS]
    - リソースタイプ 2:[カスタム]
    - ソース 2:[YYYYMMDDHandsOnBastionSecurityGroup のセキュリティグループ ID]



**基本的な詳細**

セキュリティグループ名 情報  
 20220621DeployHandsOnDBSecurityGroup  
作成後に名前を編集することはできません。

説明 情報  
 Security Group for Deploy Hands On DB Instance

VPC 情報  
 vpc-05be51b8cc9142a4d

---

**インバウンドルール** 情報

タイプ <small>情報</small>	プロトコル <small>情報</small>	ポート範囲 <small>情報</small>	ソース <small>情報</small>	説明・オプション <small>情報</small>
Oracle-RDS	TCP	1521	カスタム sg-0b86e41da479fd86a	
Oracle-RDS	TCP	1521	カスタム sg-09cfd5d5c5414e27f	

ルールを追加

---

**アウトバウンドルール** 情報

タイプ <small>情報</small>	プロトコル <small>情報</small>	ポート範囲 <small>情報</small>	送信先 <small>情報</small>	説明・オプション <small>情報</small>
すべてのトラフィック	すべて	すべて	カスタム 0.0.0.0/0	



手順8.RDS ダッシュボードを開いて [データベースの作成] をクリックする



The screenshot shows the Amazon RDS console interface. On the left is a navigation menu with options like 'Dashboard', 'Database', 'Query Editor', etc. The main content area has a top section for 'Amazon Aurora' with a 'Create Database' button. Below that is a 'Resources' section. The bottom section is titled 'データベースの作成' (Create Database) and contains a 'Create Database' button, which is highlighted with an orange box and the label 'こちらです' (Here).

手順9.データベースの作成画面で以下の通りに設定を行う

- データベース作成方法を選択
  - データベースの作成方法:[標準作成]
- エンジンのオプション
  - エンジンのタイプ:[Oracle]
  - Database management type : [Amazon RDS]
  - エディション:[Oracle Standard Edition Two]
  - ライセンス:[license-included]
  - バージョン:[Oracle 19.0.0.0.ru-20YY-MM.rur-20YY-MM.rX]

※無い場合は一番新しいバージョンを選択
- テンプレート
  - テンプレート:[開発/テスト]
- 設定
  - DB インスタンス識別子:[DeployHandsOnOracleInstance **YYYYMMDD**]  
(**YYYYMMDD**は本日の日付)
  - マスターユーザー名:[oracleadmin]
  - パスワードの自動生成:[無効(チェックを外す)]
  - マスターパスワード:MasterPass
  - マスターパスワードの確認:MasterPass
- DB インスタンスサイズ
  - DB インスタンスクラス:[バースト可能クラス(t クラスを含む)] [db,t3,small]

- ストレージ
  - ストレージタイプ:[汎用(SSD)]
  - ストレージ割り当て:[20]
  - ストレージの自動スケーリング:[無効](チェックを外す)
- 可用性と耐久性
  - マルチ AZ 配置:[スタンバイインスタンスを作成しないでください]
- 接続
  - コンピューティングリソース : [EC2 コンピューティングリソースに接続しない]
  - ネットワークタイプ:[IPv4]
  - VPC:[YYYYMMDDDeployHandsOnVPC] ※作成済みの VPC
  - DB サブネットグループ:[deployhandsonsubnetgroup] ※作成済みのグループ
  - パブリックアクセス:[なし]
  - VPC セキュリティグループ:[既存の選択]
  - 既存の VPC セキュリティグループ:[YMMDDDeployHandsOnDBSecurityGroup]
- ※default は削除する
- アベイラビリティゾーン:[ap-northeast-1c]
- データベース認証
  - データベース認証オプション:[パスワード認証]
- モニタリング
  - Performance Insights をオンにする:[無効](チェックを外す)
- 追加設定
  - 最初のデータベース名:[oracledb]
  - DB パラメータグループ:[default.oracle-se2-19](初期値)
  - オプショングループ:[default.oracle-se2-19](初期値)
  - 文字セット:[AL32UTF8](初期値)
  - 自動バックアップの有効化:[無効](チェックを外す)
  - 暗号を有効化:[無効](チェックを外す)
  - アラートログ:[無効]
  - 監査ログ:[無効]
  - リスナーログ:[無効]
  - Oracle Management Agent ログ:[無効]
  - トレースログ:[無効]
  - マイナーバージョン自動アップグレードの有効化:[無効]
  - メンテナンスウィンドウ:[設定なし]
  - 削除保護の有効化:[無効]

手順 1 0.設定内容を確認し、[データベースの作成] をクリックする

手順 1 1.RDS インスタンスを起動する

※数分～数十分かかることがあります。

ここまでで RDS の作成は完了です。

## 作業 5 .EC2 インスタンス(踏み台)と RDS を接続する

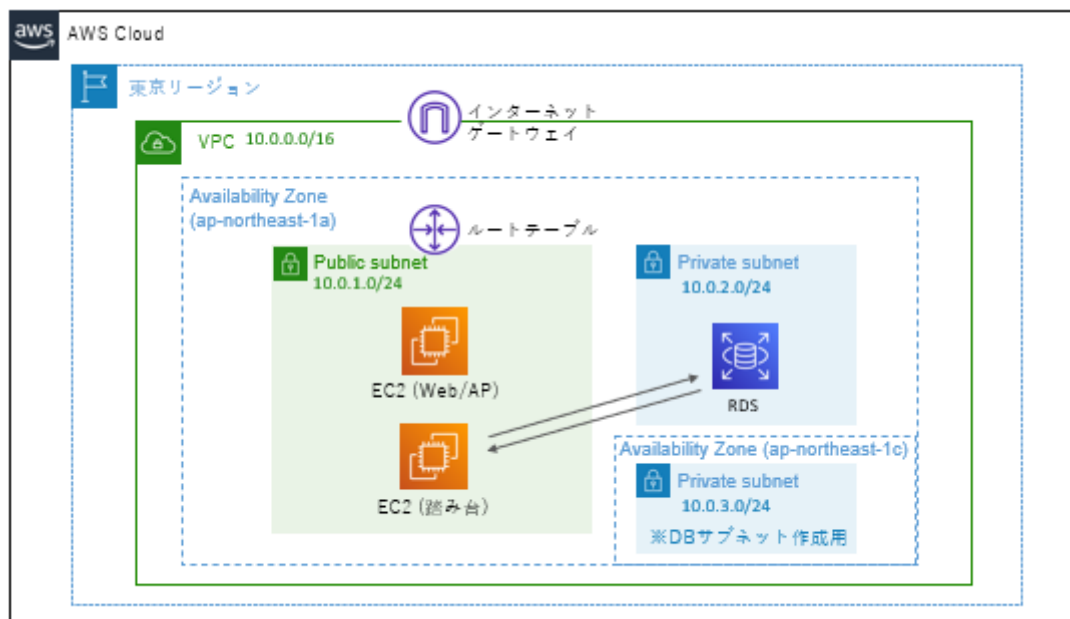
### 【作業の目的】

SQL Developer から EC2 インスタンス(踏み台)を経由して、RDS へ接続します。  
これにより SQL Developer を使用して、RDS の Oracle データベースが操作できるようになります。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ EC2 インスタンス(踏み台用)を経由して RDS へ接続できるように、SQL Developer を設定  
※SQL Developer の SSH 接続機能とポート転送機能を使用します。
- ・ SQL Developer から EC2 インスタンス(踏み台用)を経由して RDS への接続を確認



### 【作業手順】

手順 1 .RDS ダッシュボードを開く

手順 2 .作成した DB インスタンスのステータスが [利用可能] になっていることを確認する



手順3.データベース名のリンクをクリックする

手順4.画面下部の[接続とセキュリティ] タブから [エンドポイント] の欄に記載されている URL をメモする

※URL とは、下記赤枠内の「エンドポイント」の文字の下にある「～amazonaws.com」の部分までのすべての文字を指します。下図ではセキュリティの都合上一部を黒塗りしています。



**deployhandsonoracleinstance20220621**

**概要**

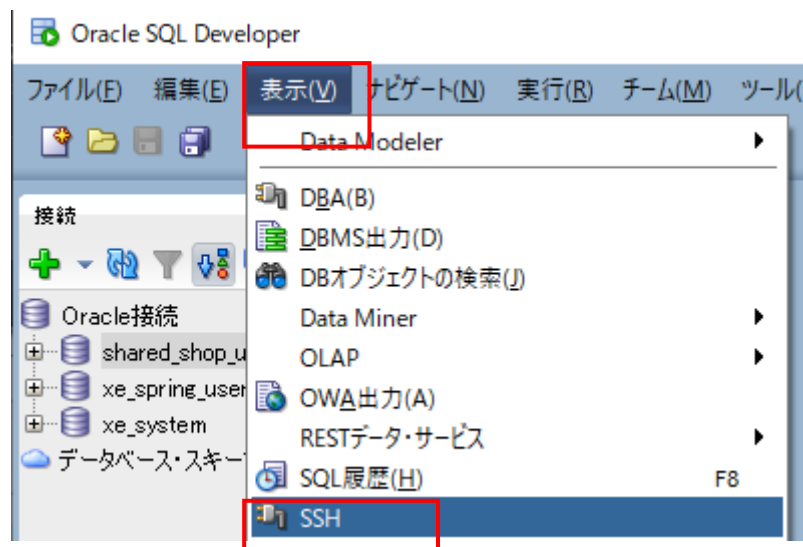
DB 識別子 deployhandsonoracleinstance20220621	CPU 2.84%
ロール インスタンス	現在のアクティビティ 1 接続

**接続とセキュリティ**

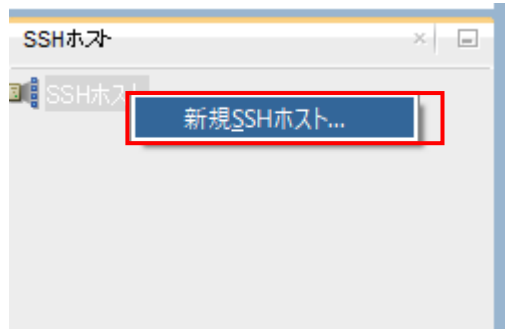
<b>エンドポイントとポート</b> <b>エンドポイント</b> deployhandsonoracleinstance20220621. [REDACTED] [REDACTED] ap-northeast-1.rds.amazonaws.com <b>ポート</b> 1521	<b>ネットワーク</b> アベイラビリティゾーン ap-northeast-1c VPC 20220621DeployHandsOnVPC-vpc (vpc-05be51b8cc9142a4d)
--	--

手順5.SQL Developer を起動する

手順6.SQLDeveloper の画面上部にある [表示] から [SSH] をクリックする



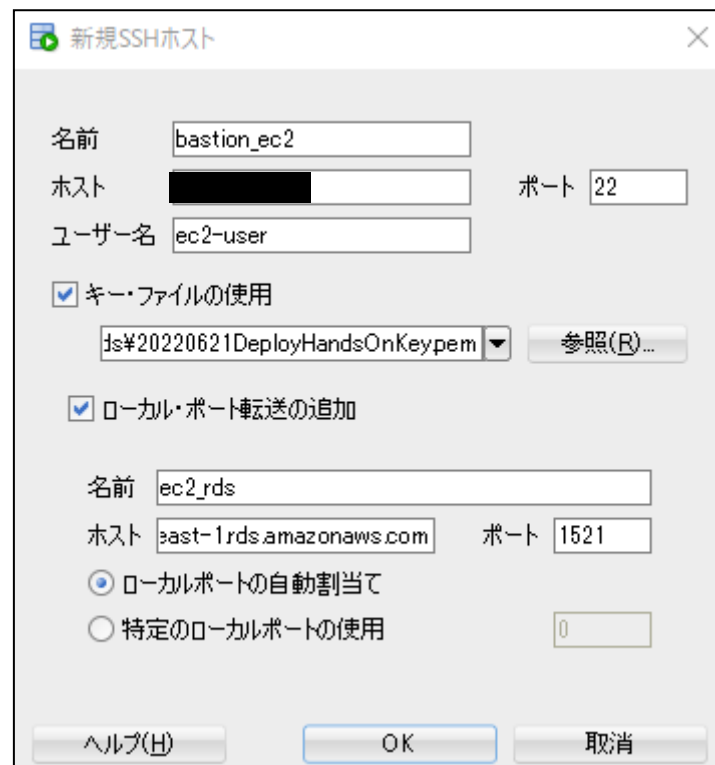
手順7. SQL Developer の画面左下の SSH ホストを右クリックして、[新規 SSH ホスト] をクリックする



手順8. 以下のように設定を行い [OK] をクリックする

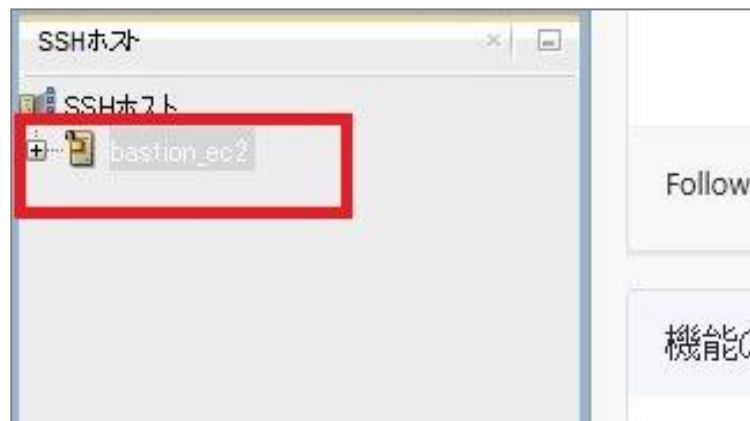
※SQL Developer から EC2 インスタンス(踏み台用)を経由して RDS へ接続を行うための設定です。

- 名前:[bastion\_ec2]
  - ホスト:[EC2 インスタンス(踏み台用)のパブリック IP アドレス]
  - ポート:[22] ※ssh のポート
  - ユーザー名:[ec2-user]
  - キー・ファイルの使用:[有効] [YYYYMMDDDeployHandsOnKey]
- ※EC2 インスタンスの作成時に指定したキーペアファイルを[参照]ボタンから選択
- ローカル:ポート転送の追加:[有効]※チェックする
  - 名前:[ec2\_rds]
  - ホスト:[RDS のエンドポイント]



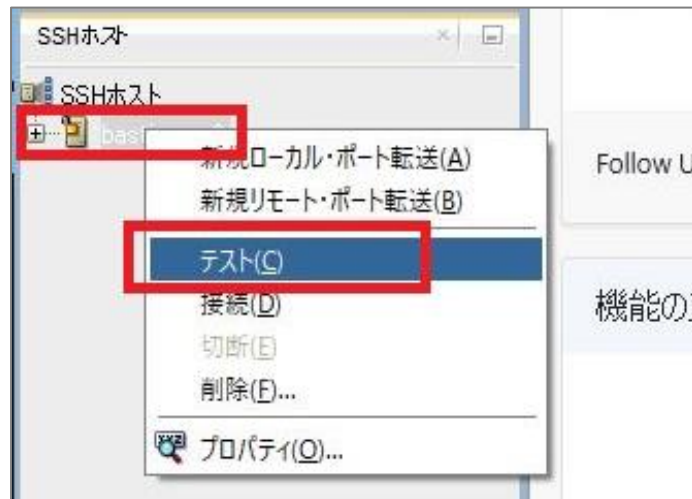
手順9.SQLDeveloper で作成した SSH ホストを確認する

SSH ホストの作成が完了すると下記画像の通り、[SSH ホスト]に作成した SSH ホストが表示されます。



手順10.RDS への接続確認をする

RDS への接続をテストするために SSH ホスト名[bastion\_ec2] を右クリックしてから [テスト] をクリックして [接続テストに成功しました] と表示されることを確認します。※少し時間がかかります。



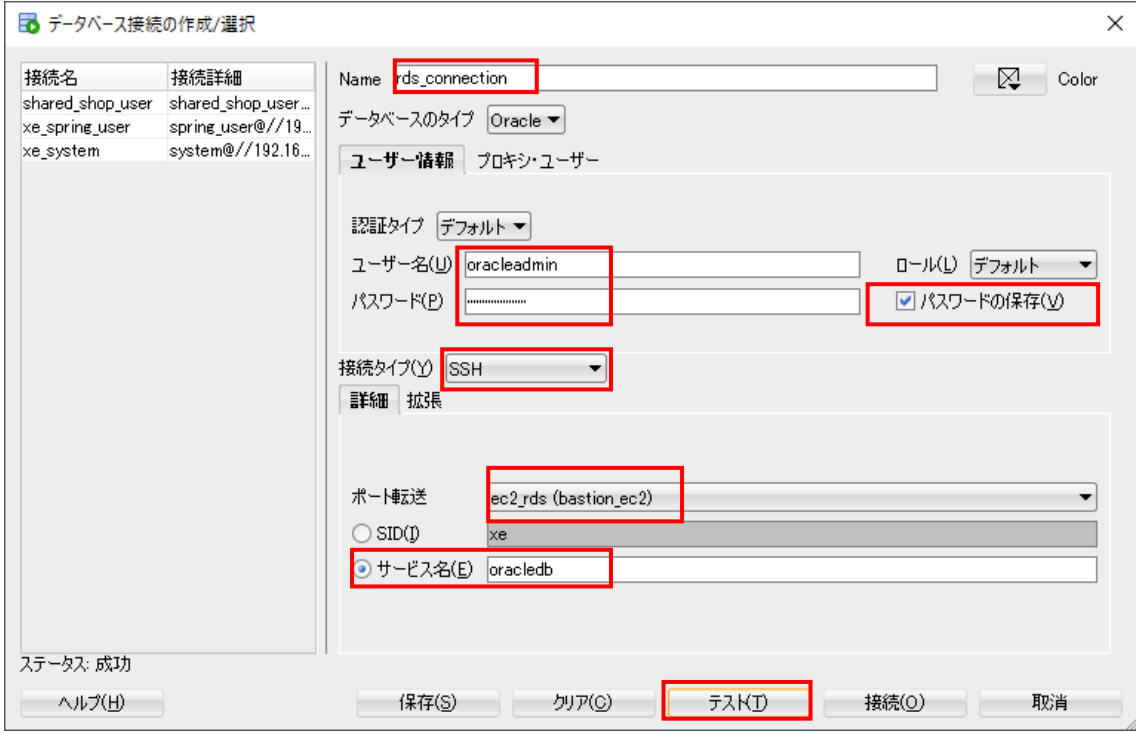
手順11.RDS への接続を確立する

SSH ホスト名[bastion\_ec2]を右クリックし、[接続]します。その後、[bastion\_ec2]の横の+マークをクリックして表示される名前を右クリックして、[接続]します。

## 手順 1 2.RDS 内のデータベースへの接続を作成する

SQL Developer の画面左上にある [接続] ウィンドウの下にある [+] マークをクリックして以下の通りに設定をします。

- 名前:[rds\_connection]
- ユーザ名:[oracleadmin](RDS 作成時に設定した値)
- パスワード:[MasterPass](RDS 作成時に設定したマスターパスワード)
- パスワードの保存:チェックする
- 接続タイプ:[SSH]
- ポート転送:[bastion\_ec2]
- サービス名:[oracledb] (RDS 作成時に設定した値)



データベース接続の作成/選択

接続名	接続詳細
shared_shop_user	shared_shop_user...
xe_spring_user	spring_user@//19...
xe_system	system@//192.16...

Name: **rds\_connection**

データベースのタイプ: Oracle

ユーザー情報 | プロキシ・ユーザー

認証タイプ: デフォルト

ユーザー名(U): **oracleadmin**

パスワード(P): **MasterPass**

パスワードの保存(Y): ☒

接続タイプ(Y): **SSH**

詳細 | 拡張

ポート転送: **ec2\_rds (bastion\_ec2)**

SID(I): xe

サービス名(E): **oracledb**

ステータス: 成功

ヘルプ(H) | 保存(S) | クリア(C) | **テスト(T)** | 接続(O) | 取消

## 手順 1 3.RDS 内のデータベースへの接続を確認する

設定内容を入力後、[テスト]をクリックし、左下に「成功」と表示されれば、正しく設定できています。

## 手順 1 4.RDS 内のデータベースへの接続を保存する

[保存]をクリックして設定を保存し、[接続]をクリックして DB に接続します。これで、SQL Developer から RDS 内のデータベース(Oracle)を操作できるようになります。

ここまでで、EC2 インスタンス(踏み台)と RDS の接続作業は完了です。

再度 SQL Developer から RDS へ接続する場合は、「手順 11.RDS への接続を確立する」を行った後、データベースへ接続する必要があります。



## 作業 6 .EC2 インスタンス(WEB/AP)と RDS を接続する

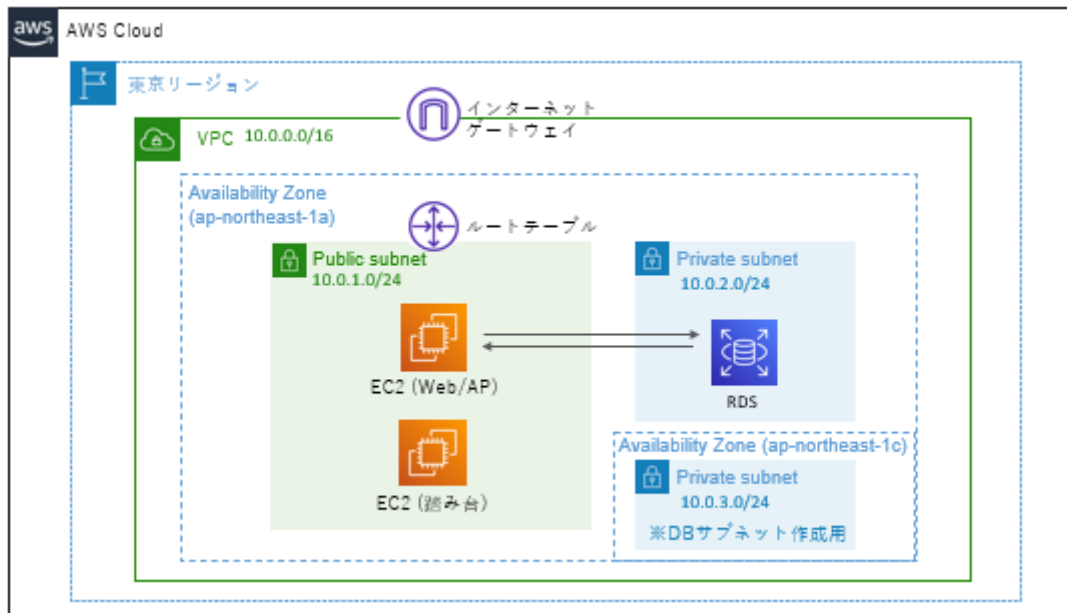
### 【作業の目的】

WEB アプリケーションを実行する EC2 インスタンス(WEB/AP)から、RDS へアクセスできるように接続設定を行います。  
これにより WEB アプリケーションからデータベースが利用できるようになります。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- TeraTerm を使って、EC2 インスタンス(WEB/AP)に接続
- 環境構築に必要なファイルを、EC2 インスタンス(WEB/AP)に転送
- EC2 インスタンス(WEB/AP)に Java の環境をインストール  
※WEB アプリケーションで使います。
- Oracle データベースを操作するためプログラム(sqlplus)を EC2 インスタンス(WEB/AP)にインストール
- sqlplus で、EC2 インスタンス(WEB/AP)から RDS 内のデータベースに接続できることを確認



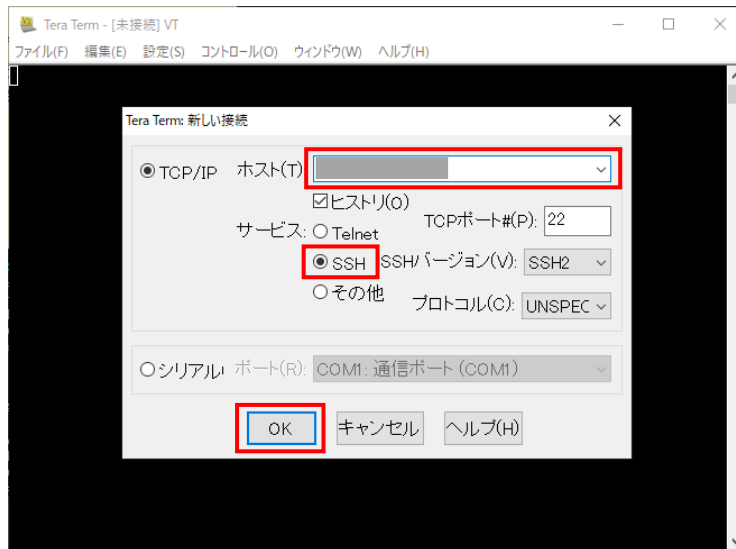


【作業手順】

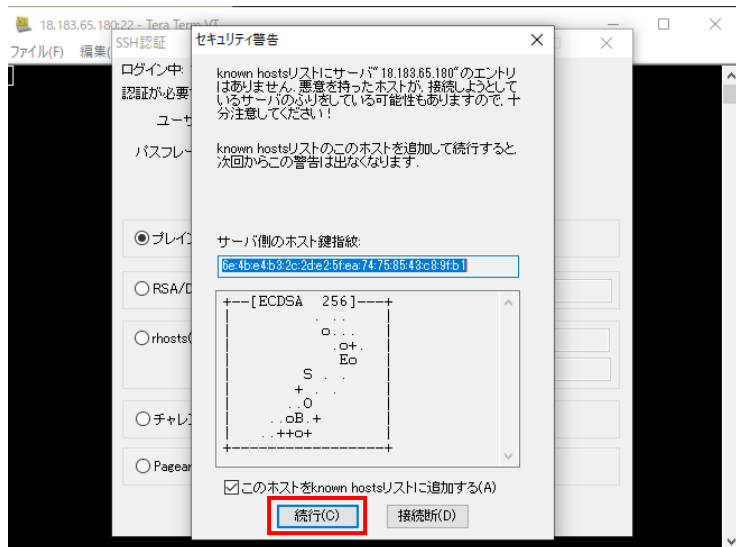
手順1.TeraTerm で新しい接続を作る

TeraTerm から EC2(WEB/AP)へ SSH 接続するための設定をします。

Teraterm を起動し、ホストに EC2 インスタンス(WEB/AP)のパブリック IP を指定し、SSH を選択後、[OK]ボタンを押します。

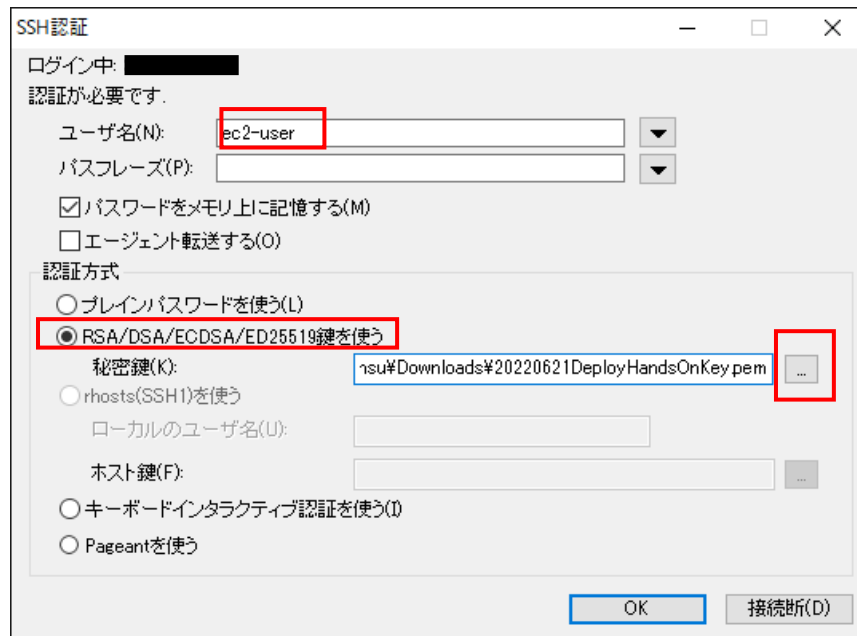


手順2.セキュリティ警告が表示されたら、[続行]ボタンを押す



手順3.SSH 認証画面で、ユーザ名と認証方式を指定

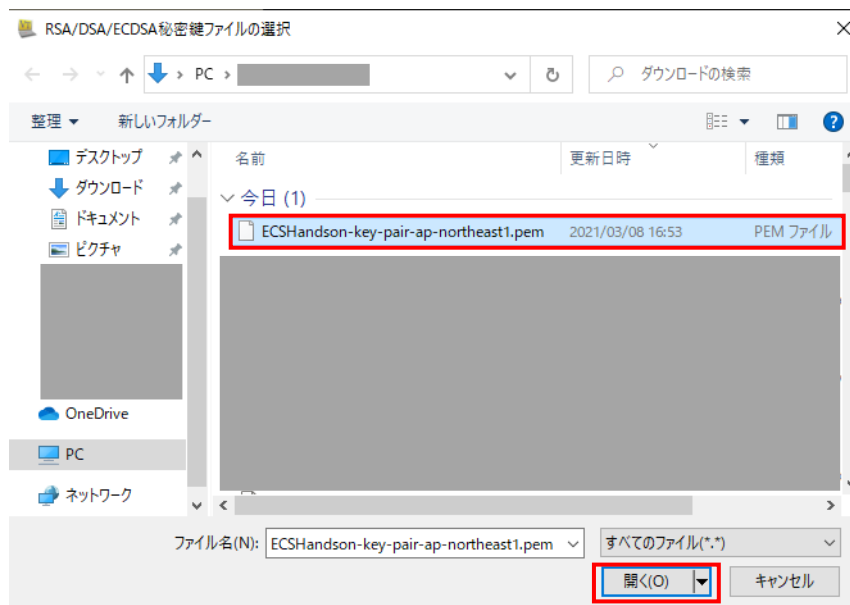
SSH 認証画面でユーザ名に「ec2-user」を入力し、[RSA/DSA 鍵を使う]を選択した後、[秘密鍵]の参照ボタンを押します。



手順4.秘密鍵ファイルを選択する

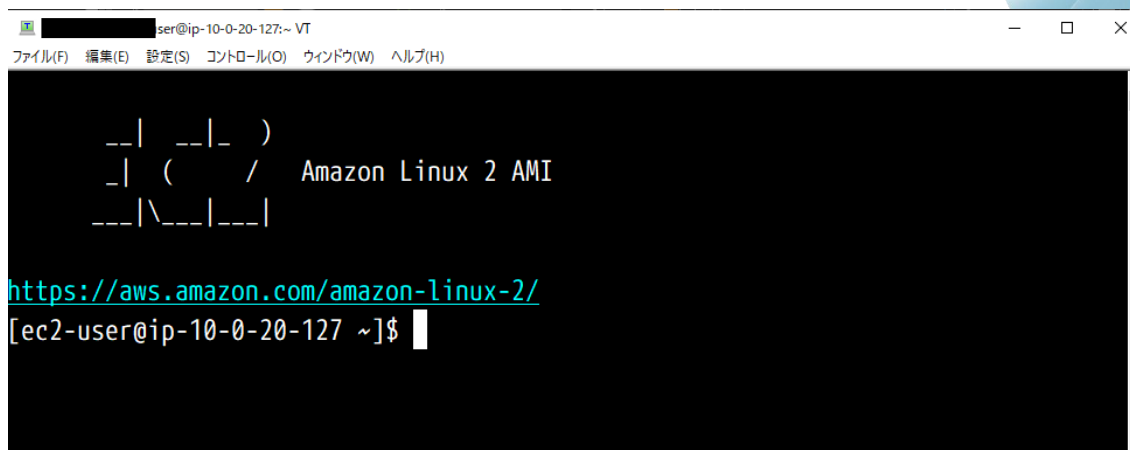
[秘密鍵ファイルの選択]画面でダウンロードしたキーペアファイルを選択し、[開く]ボタンを押す

※【注意】ファイル選択用のリストから「秘密鍵のファイル」または「すべてのファイル(\*.\*)」を選択します。



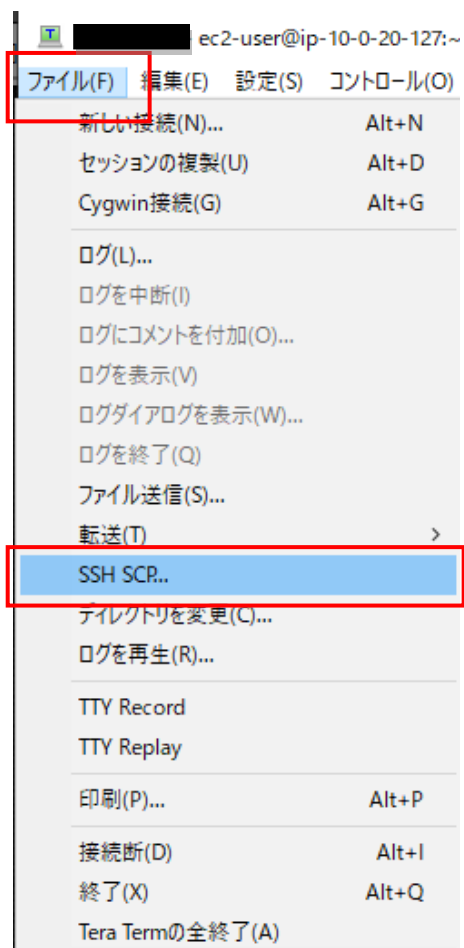
手順5.秘密鍵ファイルの選択後、[OK]ボタンを押す

EC2 インスタンスにログインできると、下記のような画面が表示されます。



手順6.TeraTerm 画面左上の [ファイル] から、[SSH SCP] を選択する

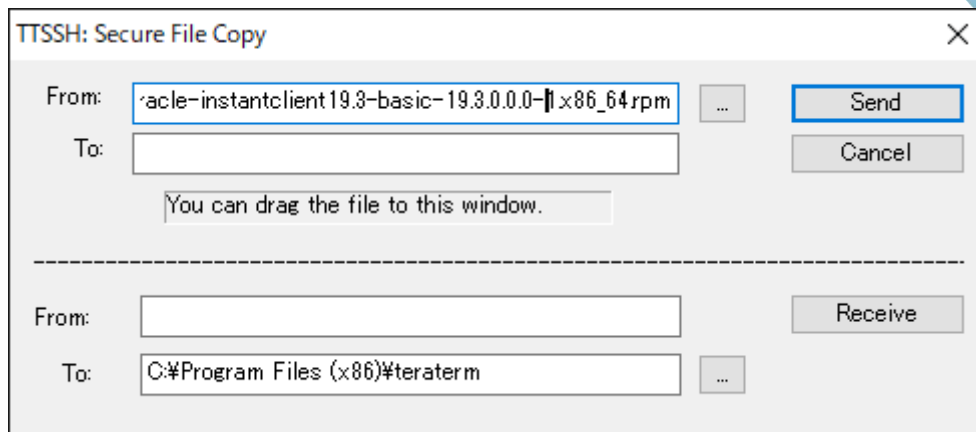
※EC2 インスタンスへファイルを転送するための作業です。



手順7.アップロードするファイルを選択して転送する

Secure File Copy のウィンドウが開いたら [from] の右側の [...] をクリックして、事前にダウンロードした以下のファイルをクリックして [Send] をクリックします。

- oracle-instantclient19.3-basic-19.3.0.0.0-1.x86\_64.rpm



手順8.同じ手順で以下の2ファイルも転送する

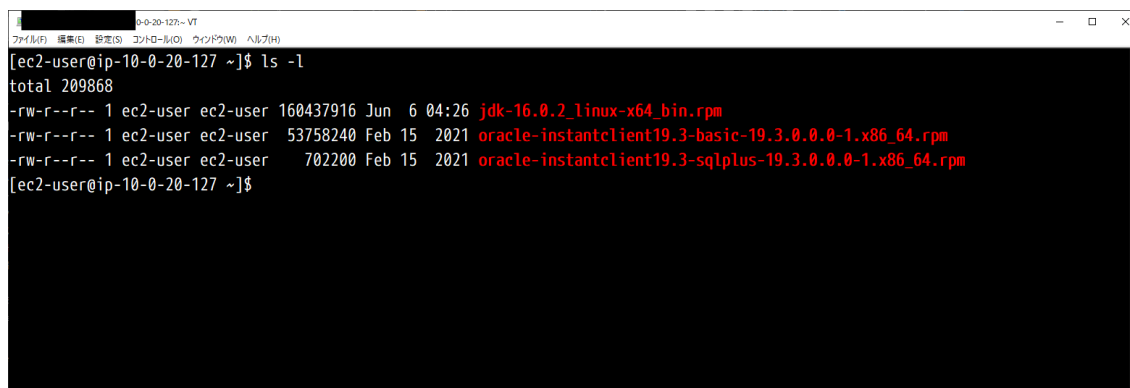
- oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86\_64.rpm
- jdk-16.0.2\_linux-x64\_bin.rpm

手順9.ファイルが転送できたことを確認する

ファイルが転送できたことを確認するために以下のコマンドを実行し、下の画面のように表示されることを確認します。

```
ls -l
```

※ディレクトリ上にあるファイルのリストを表示するコマンドです。



### 手順1 0 Java のバージョンを設定する

EC2 インスタンス(WEB/AP)のソフトウェアを最新版にした後、Java 環境のインストール、Java のバージョン設定を行うために、以下のコマンドを順番に実行します。

※ Is this ok [y/d/N]: と表示された場合には [y] を入力して実行します。

```
sudo yum update -y
```

```
sudo yum -y localinstall jdk-16.0.2_linux-x64_bin.rpm
```

```
sudo alternatives --config java
```

[alternatives] コマンドは、複数の JDK をインストールしている場合にバージョンを切り替えるためのコマンドです。実行後に以下のように表示が行われるため [1] を入力して実行します。

```
ec2-user@ip-10-0-20-127: ~
File(F) Edit(E) Settings(S) Control(O) Window(W) Help(H)
Installed:
  jdk-16.0.2.x86_64 2000:16.0.2-ga

Complete!
[ec2-user@ip-10-0-20-127 ~]$ sudo alternatives --config java

There is 1 program that provides 'java'.

  Selection    Command
-----
*+ 1          /usr/java/jdk-16.0.2/bin/java

Enter to keep the current selection[+], or type selection number: 1
[ec2-user@ip-10-0-20-127 ~]$
```

最後に Java のバージョンを確認する以下のコマンドを実行して、次の画像の通りに表示がされれば Java のバージョン設定は完了です。

```
java -version
```

```
[ec2-user@ip-10-0-20-127 ~]$ java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)
[ec2-user@ip-10-0-20-127 ~]$
```

## 手順1 1.Oracle の環境をインストールする

Oracle DB インスタンスへの接続に必要なソフトウェアをインストールするために、以下の2つのコマンドを実行し、インストールが100%完了することを確認してください。

```
sudo rpm -Uvh oracle-instantclient19.3-basic-19.3.0.0-1.x86_64.rpm
```

```
[ec2-user@ip-10-0-20-127 ~]$ sudo rpm -Uvh oracle-instantclient19.3-basic-19.3.0.0-1.x86_64.rpm
Preparing... ##### [100%]
Updating / installing...
 1:oracle-instantclient19.3-basic-19.3.0.0-1.x86_64.rpm ##### [100%]
```

```
sudo rpm -Uvh oracle-instantclient19.3-sqlplus-19.3.0.0-1.x86_64.rpm
```

```
[ec2-user@ip-10-0-20-127 ~]$ sudo rpm -Uvh oracle-instantclient19.3-sqlplus-19.3.0.0-1.x86_64.rpm
Preparing... ##### [100%]
Updating / installing...
 1:oracle-instantclient19.3-sqlplus-19.3.0.0-1.x86_64.rpm ##### [100%]
```

## 手順1 2.RDS へ接続するためのユーザ名とパスワードを準備する

下記の文字列をテキストエディタにコピーして、事前にメモした RDS のエンドポイントを含めた文字列を用意します。

```
oracledb/MasterPass@ (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=RDS のエンドポイント) (PORT=1521)) (CONNECT_DATA=(SID=DB 名)))
```

- MasterPass:[RDS 接続時に設定したパスワード]
- RDS のエンドポイント:[RDS ダッシュボードに表示されているエンドポイント URL]
- DB 名:[oracledb](RDS の作成時に設定した値)

## 記述例

```
oracledb/MasterPass@ (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=xxx.cepr1tu9me6g.ap-northeast-1.rds.amazonaws.com) (PORT=1521)) (CONNECT_DATA=(SID=oracledb)))
```

## 手順1 3.RDS へ接続を確認する

以下のコマンドを実行します。

```
sqlplus64 [手順12 で準備したユーザ名とパスワードを含む文字列]
```

※[]は不要です。

下記のように「SQL>」と表示され、SQL の入力可能な状態になれば RDS への接続は成功です。

```
[ec2-user@ip-10-0-20-127 ~]$ sqlplus64 oracledb/MasterPass@ (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=xxx.cepr1tu9me6g.ap-northeast-1.rds.amazonaws.com) (PORT=1521)) (CONNECT_DATA=(SID=oracledb)))

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Jun 21 05:06:19 2022
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Last Successful login time: Tue Jun 21 2022 04:43:32 +00:00

Connected to:
Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 - Production
Version 19.14.0.0.0

SQL>
```

手順 1 4 .sqlplus を終了する

[exit] と入力すると、sqlplus を終了できます。

ここまでで、EC2 インスタンス(WEB/AP)と RDS の接続設定は完了です。

## 4 Web アプリケーションの準備

### 4-1 Web アプリケーションの準備作業の流れ

Web アプリケーションの準備作業は下記の流れで行います。

1. 設定ファイル(pom.xml、application.properties)を編集する
2. プロジェクトをビルドし JAR ファイルを生成する
3. 画像ファイルを tar ファイル形式でエクスポートする

準備作業では、AWS サービスでの環境の設定値を利用します。  
作業は、Eclipse を使って作業をすることを前提としています。

### 4-2 Web アプリケーションの準備作業の実施

作業 1 .設定ファイル(pom.xml、application.properties)を編集する

#### 【作業の目的】

Web アプリケーションを AWS に構築した環境で動作させるための設定を行います。

#### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ JAR ファイルを使って実行できるように pom.xml を編集し保存
- ・ AWS 環境のデータベースが利用できるように application.prierties を編集し保存



### 【作業手順】

手順1.Eclipse を起動する

手順2.Eclipse のデプロイ対象プロジェクトにある pom.xml を開く

手順3.pom.xml を編集する

pom.xml 中の<project>タグ直下に子要素として<packaging>タグを追記し、保存します。

編集前

```
<project . . . >
  ~中略~
<description>Demo project for Spring Boot</description>
<properties>
<java.version>16</java.version>
</properties>
  ~中略~
</project>
```

編集後(下線部が追記箇所)

```
<project . . . >
  ~中略~
<description>Demo project for Spring Boot</description>
<packaging>jar</packaging>
<properties>
<java.version>16</java.version>
</properties>
  ~中略~
</project>
```

<packaging>タグは、ビルドするファイルのパッケージング・タイプ(ファイル形式)を指定するためのタグです。  
JAR ファイルをビルドしたい場合、開始タグと終了タグの間に「jar」と記入します。

#### 手順4. 「application.properties」を編集する

##### 編集前

```
spring.profiles.active=local

#AWS へのデプロイ準備作業で有効にする
#spring.profiles.active=aws
```

編集後(「~active=local」の行をコメント化し、「~active=aws」の行コメントを解除する。

```
#spring.profiles.active=local

#AWS へのデプロイ準備作業で有効にする
spring.profiles.active=aws
```

※開発環境用に戻す場合は、編集前の状態に戻します。

#### 手順5. 「application-aws.properties」を編集する

下記の内容に従い編集し、保存します。

##### 編集前

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xepdb1
spring.datasource.username=shared_shop_user
spring.datasource.password=systemsss
```

編集後(下線部は編集箇所、および追記箇所)

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:[RDSで設定したユーザ名]/[RDSで設定したパスワード]@[RDSから発行された
エンドポイント]:1521:[RDSで設定した最初のデータベース名]
#spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xepdb1
#spring.datasource.username=shared_shop_user
#spring.datasource.password=systemsss
```

※"[]"は不要です。

RDSのデータベースを単元「クラウド基礎」の「動的Webページデプロイ演習」と同手順で作成した場合、上記編集内容中の[]の箇所は下記内容に置き換えます。

- ・[RDSで設定したユーザ名]:oracleadmin
- ・[RDSで設定したパスワード]:MasterPass
- ・[RDSから発行されたエンドポイント]:RDSのダッシュボード画面で確認したURL
- ・[RDSで設定した最初のデータベース名]:oracledb

##### 記述例

```
spring.datasource.url=jdbc:oracle:thin:oracleadmin/MasterPass@xxx.cepr1tu9me6g.ap-northeast-1.rds.amazonaws.com:1521:oracledb
```

ローカル環境内のデータベースへの接続設定はデプロイ時には使用しないため先頭に#を付けて、コメントアウトします。

## 作業2.プロジェクトをビルドし JAR ファイルを生成する

### 【作業の目的】

JAR ファイルを使って実行できるようにプロジェクトをビルドし、JAR ファイルを生成します。

### 【作業概要】

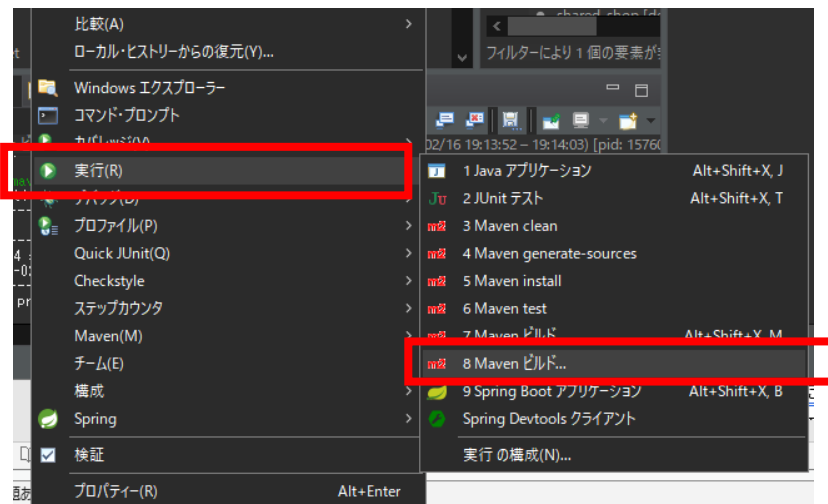
ここで行う作業の概要は下記の通りです。

- Maven のビルド構成を設定
- プロジェクトをビルドし、JAR ファイルを生成

### 【作業手順】

手順1.Maven のビルド構成設定メニューを選択する

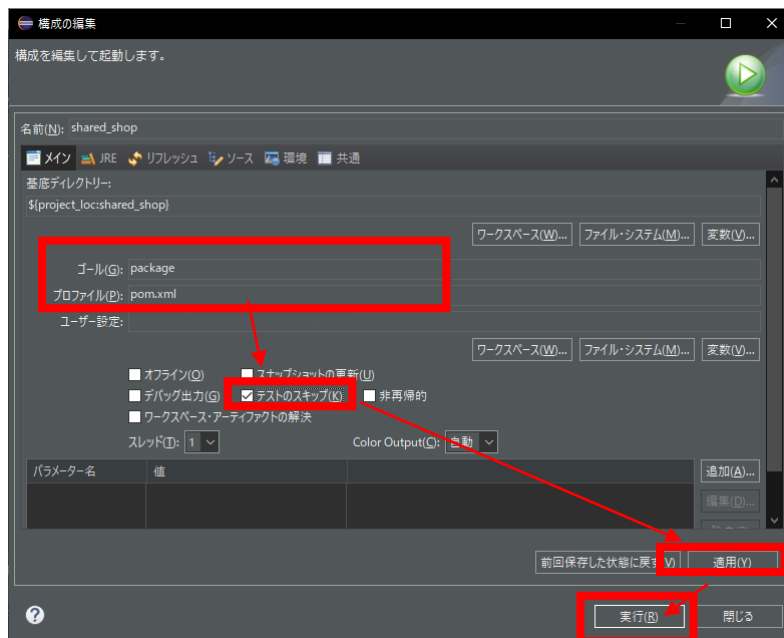
「プロジェクト・エクスプローラー」ビュー上でビルド対象のプロジェクトを選択して右クリックし、[実行] → [Maven ビルド...]を選択します。



## 手順2.ビルド構成の設定をする

[構成の編集]画面が表示されたら、ゴールとプロファイルの入力ボックスに下記内容を入力し、「テストのスキップ」にチェックを入れる。その後、[適用]ボタン→[実行]ボタンの順に押します。

ゴール: package  
プロファイル: pom.xml



## 手順3.ビルドの成功を確認する

Eclipse のコンソールに下記のような「BUILD SUCCESS」というメッセージが表示されることを確認します。

```
[INFO] -----<jp.co.sss.shared_shop>-----
[INFO] Building shared_shop 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
<<<<<<< 省略 >>>>>>>
[INFO] ---maven-jar-plugin:3.2.2:jar (default-jar) @ shared_shop---
[INFO] Building jar: C:\¥pleiades-2022-09¥workspace¥shared_shop¥target¥shared_shop-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] ---spring-boot-maven-plugin:2.7.7:repackage (repackage) @ shared_shop---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.934 s
[INFO] Finished at: 2023-02-16T20:47:01+09:00
[INFO] -----
[WARNING] The requested profile "pom.xml" could not be activated because it does not exist.
```

※手順2で「テストのスキップ」にチェックを入れ忘れると、データベースの接続確認を行うため、ビルドに失敗してしまいます。失敗した場合は、再度設定を見直し実行してください。

手順4 .JAR ファイルが生成されていることを確認する

「プロジェクト・エクスプローラー」ビューで「target」フォルダを開き、JAR ファイルがあることを確認します。

JAR ファイルのファイル名 : [プロジェクト名]-0.0.1-SNAPSHOT.jar

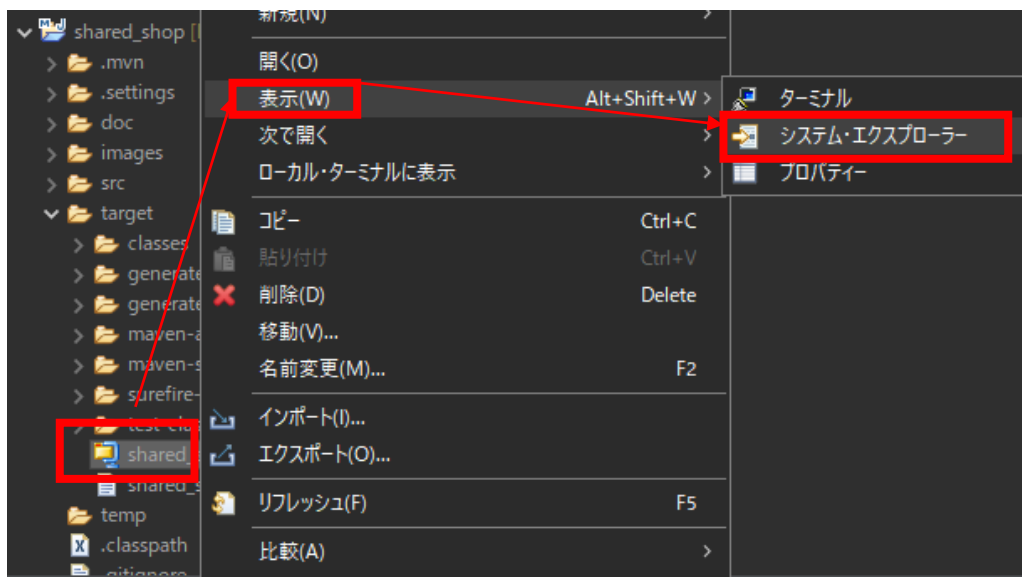
※ファイルが見つからない場合は、プロジェクトをリフレッシュ(F5 キーを押下)して再度確認してください。

手順5 .JAR ファイルをデスクトップ上にコピーする

※AWS 環境へアップロードする際に JAR ファイルの場所が把握しやすいように行う作業です。

下記の順に作業をしてください。

手順 5-1.JAR ファイルを選択して、右クリック→「表示」→「システム・エクスプローラー」をクリックする。



手順 5-2.手順 5-1 の操作で表示されたエクスプローラで、JAR ファイルの作成日時が作業を行った日時であることを確認する。

手順 5-3.JAR ファイルをデスクトップ上にコピーする。

ここまでで、プロジェクトをビルドして JAR ファイルを生成する作業は完了です。

## 作業3.画像ファイルを tar ファイル形式でエクスポートする

### 【作業の目的】

画像ファイルを AWS 環境へアップロードするために、1つのファイルにまとめます。

### 【作業概要】

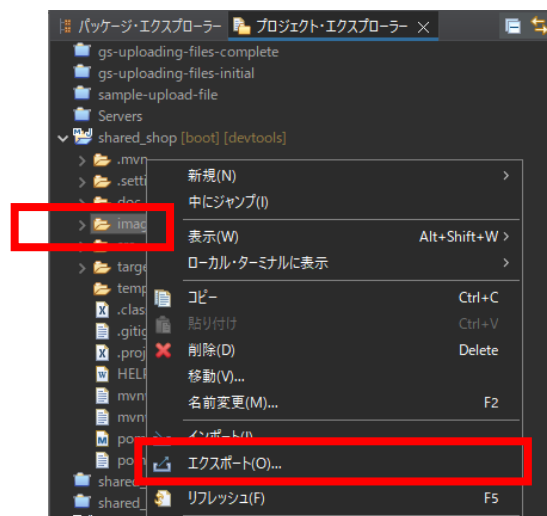
ここで行う作業の概要は下記の通りです。

- ・ 画像ファイルが格納されているフォルダの内容をアーカイブ・ファイルとして保存

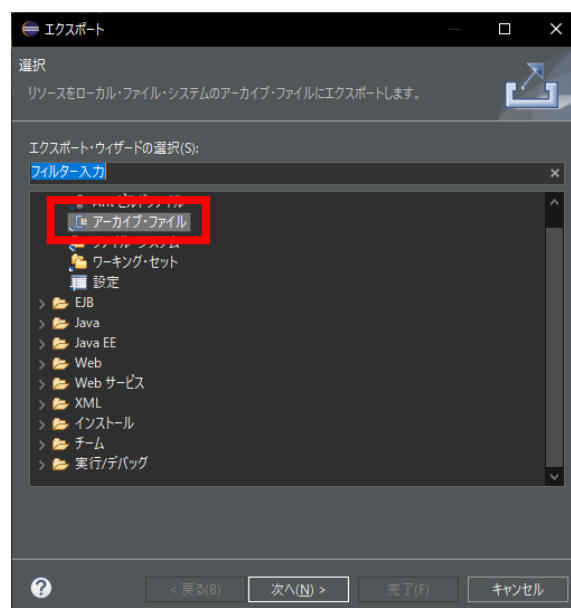
### 【作業手順】

手順1.Eclipse で images フォルダをエクスポートする

「プロジェクト・エクスプローラー」ビュー上で images フォルダを右クリックし、[エクスポート(O)...]を選択します。

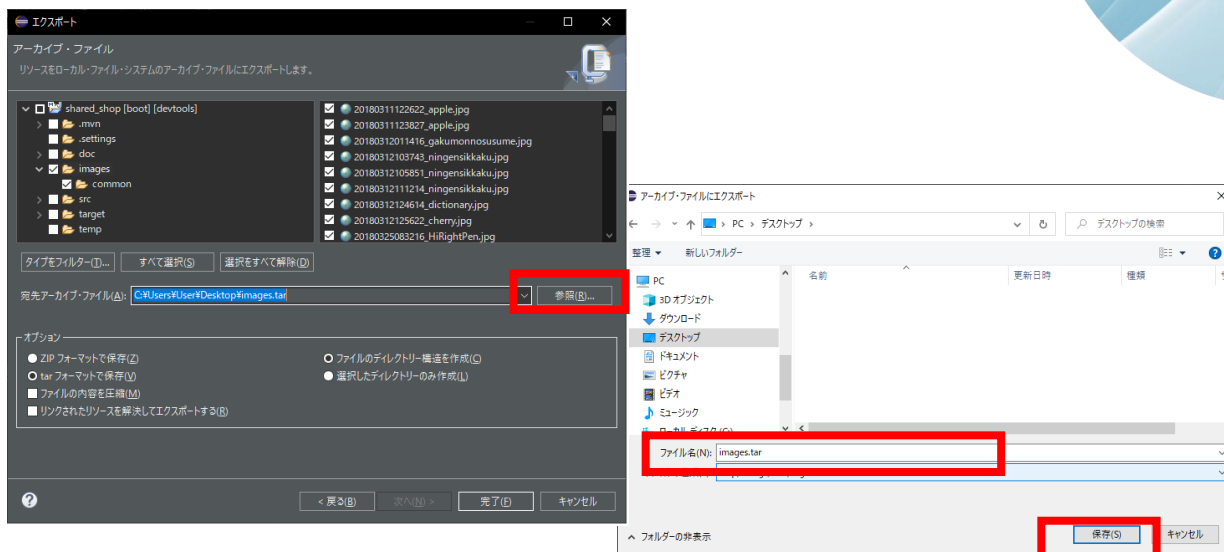


手順2.「エクスポート」画面で「一般」→「アーカイブ・ファイル」を選択し、「次へ」ボタンを押す



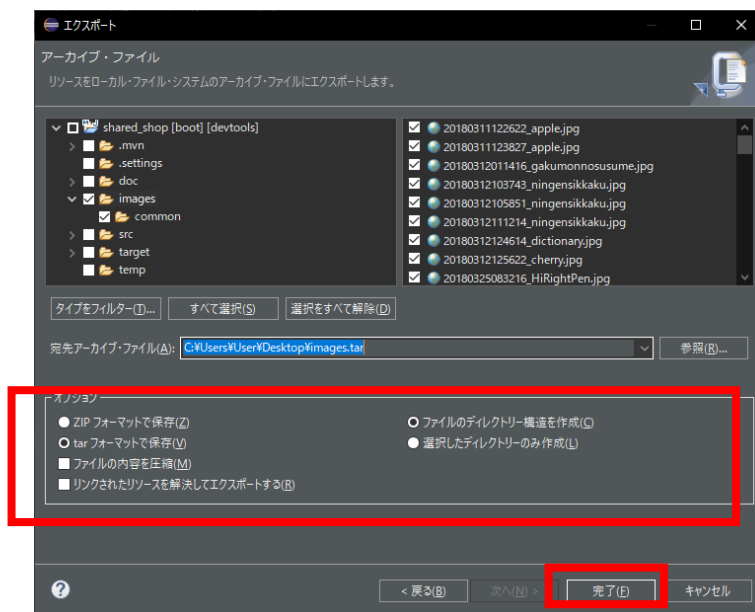
### 手順3.保存ファイル名と保存先を指定する

次の画面で「参照(R)...」ボタンを押し、デスクトップを選択後、「ファイル名」欄に「images.tar」と入力し、「保存(S)」ボタンを押します。



### 手順4.オプション設定を行い、エクスポート作業を完了する

「宛先アーカイブ・ファイル(A)」欄に保存先フォルダ名とファイル名が表示されていることを確認後、「オプション」欄で「tar フォーマットで保存」「ファイルのディレクトリー構造を作成」を選択し、「完了(F)」ボタンを押します。



### 手順5.デスクトップに images.tar ファイルが作成されていることを確認する

ここまでで、Web アプリケーションの準備作業は完了です。

作成した JAR ファイルと、tar ファイルを AWS 環境に転送してデプロイを行いましょう。

## 5 Web アプリケーションのデプロイ

### 5-1 Web アプリケーションのデプロイ作業の流れ

Web アプリケーションのデプロイ作業は下記の流れで行います。

1. Web アプリケーション用のデータベースを用意する
2. EC2 インスタンス(Web/AP)のセキュリティ設定を行う
3. Web アプリケーションをデプロイする
4. Web ブラウザから AWS 環境にアクセスし動作を確認する
5. 動作確認でバグを発見した場合、修正、再ビルド、デプロイを行う

### 5-2 Web アプリケーションのデプロイ作業の実施

作業1.Web アプリケーション用のデータベースを用意する

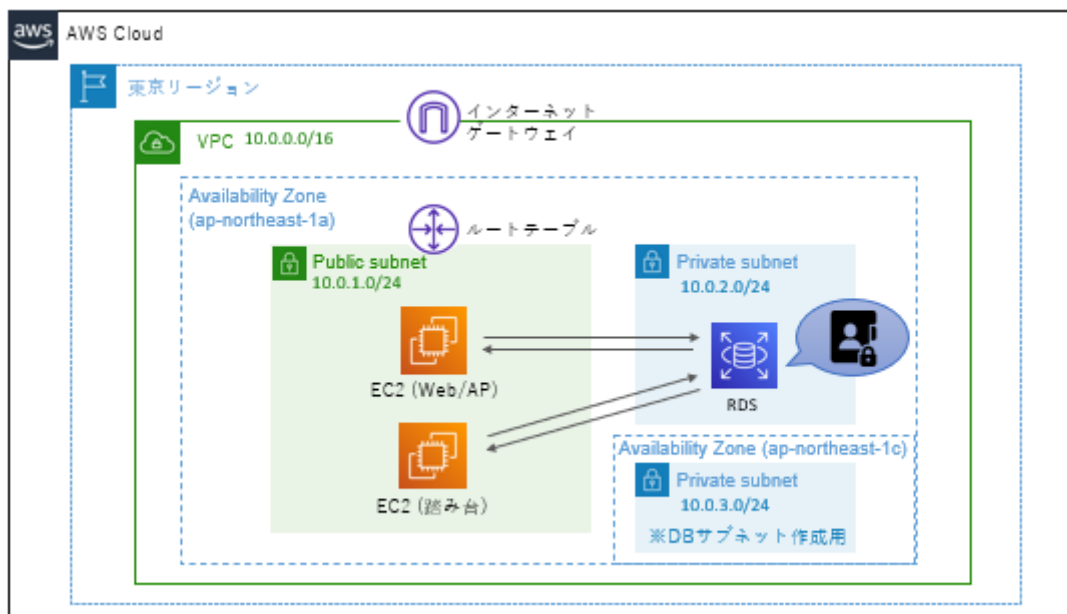
#### 【作業の目的】

RDS の Oracle データベースを操作し、WEB アプリケーションの実行に必要なデータの登録を行います。  
SQL Developer を使って、EC2 インスタンス(踏み台)を経由し、RDS の Oracle データベースを操作します。

#### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ SQL Developer を使って、RDS に接続
- ・ SQL 文を実行し、テーブル、シーケンスの作成とデータの登録を実施





## 【作業手順】

手順1.SQL Developer を開く

手順2.SQLDeveloper の SSH ホストで接続状態を確認する  
接続できていない場合、[RDS への接続確立作業](#)を行ってください。

手順3.SQL Developer の画面左上の接続ウィンドウにある[rds\_connection] をダブルクリックする  
パスワードの入力が要求された場合は、データベースのマスターパスワードを入力して [OK]ボタンをクリックし、ワークシート画面が表示されることを確認します。

手順4.事前に準備した SQL 文「AWS 環境用 SQL 文」を実行し、コミットする。

ここまでで、Web アプリケーション用データベースの用意が完了しました。

## 作業2.EC2 インスタンス(Web/AP)のセキュリティ設定を行う

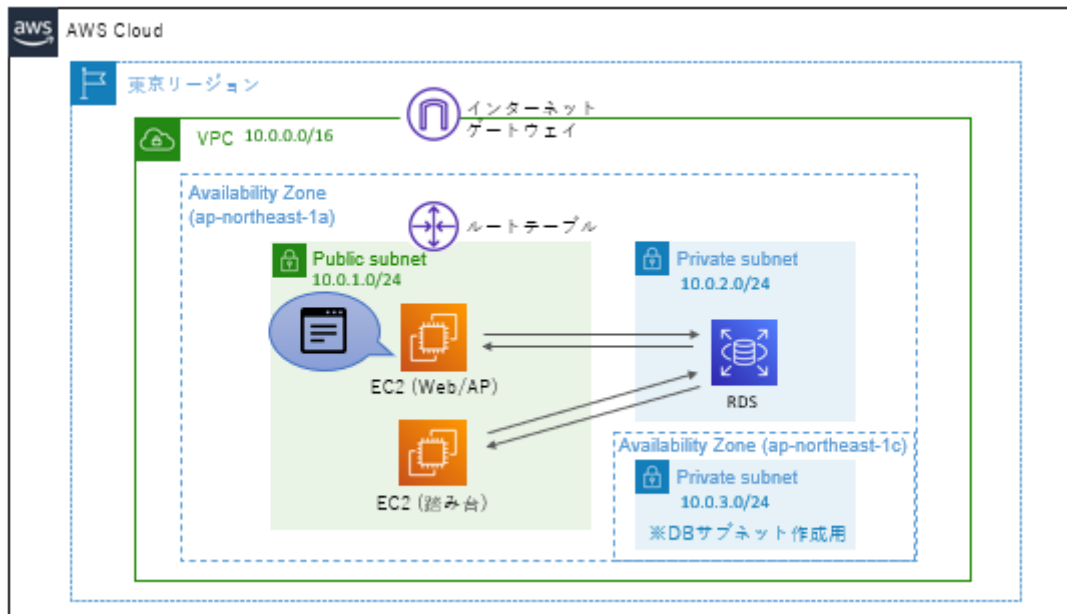
### 【作業の目的】

Web ブラウザからインターネット経由で、EC2 インスタンス上の WEB アプリケーションを利用できるように接続設定を行います。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- ・ EC2 インスタンス(WEB/AP)のセキュリティ設定を追加

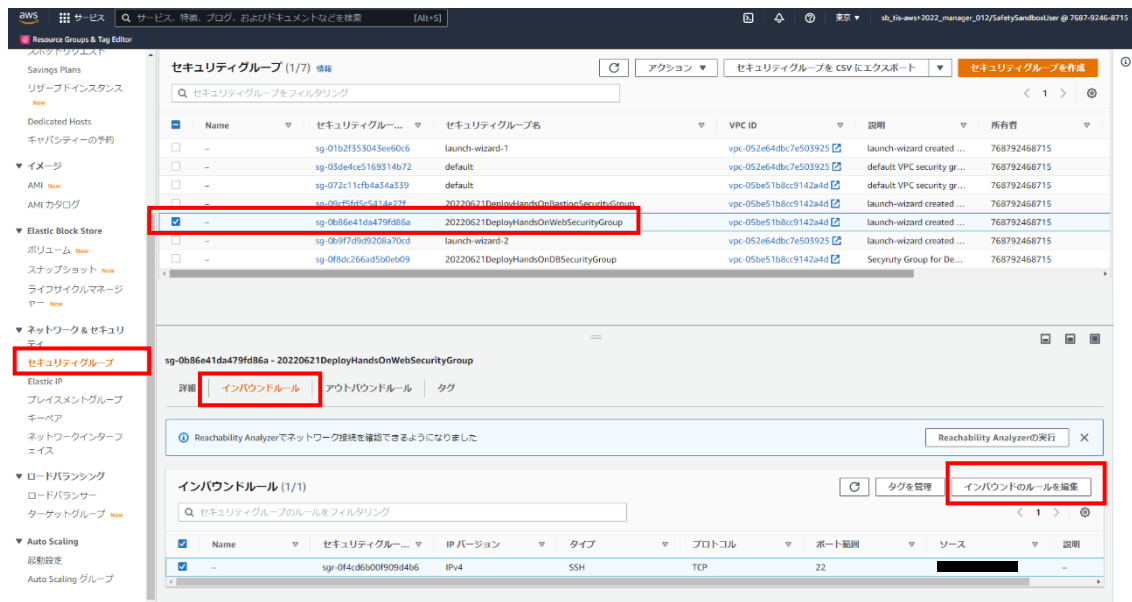


## 【作業手順】

手順1.EC2 インスタンス(WEB/AP)のセキュリティグループを選択する

手順2.インバウンドルールを編集する

EC2 ダッシュボードでセキュリティグループを開き、EC2 インスタンス(WEB/AP)のセキュリティグループの[インバウンドルール]タブで[インバウンドのルールを編集]をクリックします。



手順3.インバウンドルールを設定し、ルールを追加する

インバウンドルールの設定画面で下記の設定を行います。

- セキュリティグループ名:[YYYYMMDDDeployHandsOnWebSecurityGroup]
- タイプ:[カスタム TCP]
- ポート範囲:[55000]
- ソース:[AnywhereIPv4] [0.0.0.0/0]

The screenshot shows the 'Edit inbound rule' form in the AWS Management Console. The form has tabs for 'Type', 'Protocol', 'Port range', and 'Source'. The 'Type' tab is active, showing the rule ID 'sgr-0f4cd6b00f909d4b6'. The 'Type' is set to 'SSH', 'Protocol' to 'TCP', 'Port range' to '22', and 'Source' to 'Custom'. A new rule is being added with 'Type' set to 'Custom TCP', 'Protocol' to 'TCP', 'Port range' to '55000', and 'Source' to 'Anywhere...'. The 'Add rule' button is highlighted with a red box.

ここまでで、EC2 インスタンス(WEB/AP)のセキュリティ設定は完了です。

## 作業3.Web アプリケーションをデプロイする

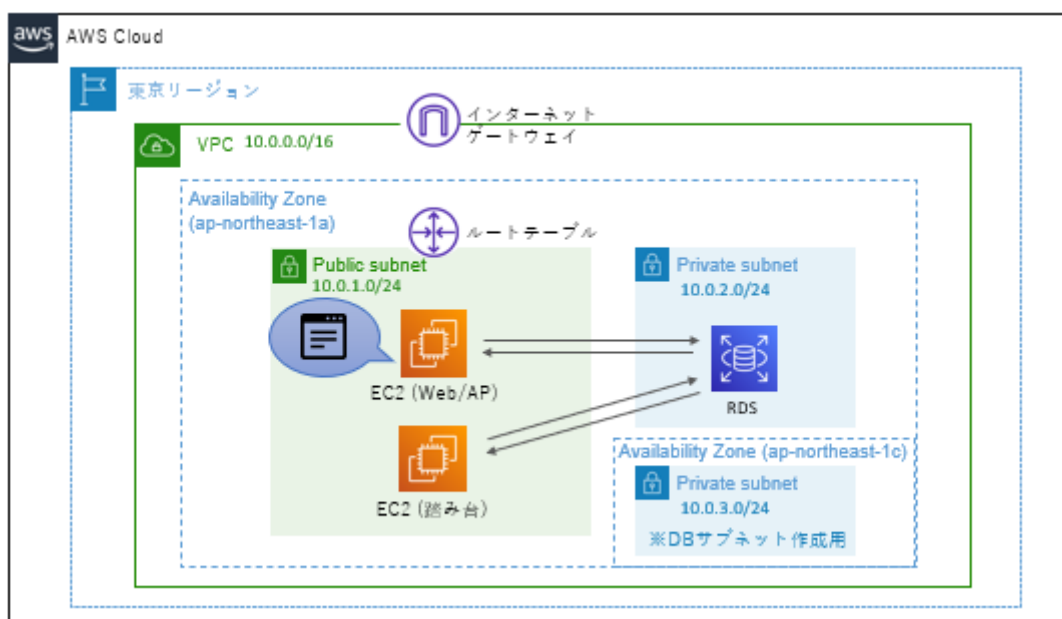
### 【作業の目的】

WEB アプリケーションを EC2 インスタンスにデプロイ(配置)し、起動します。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- WEB アプリケーションのファイルを、EC2 インスタンス(WEB/AP)に転送
- 画像ファイルを EC2 インスタンス(WEB/AP)に転送
- EC2 インスタンス(WEB/AP)上で WEB アプリケーションを起動



### 【作業手順】

手順1.TeraTerm で EC インスタンス(WEB/AP)に接続する

手順2.JAR ファイルを EC2 インスタンス(WEB/AP)に転送する

TeraTerm で下記の操作を行い、Web アプリケーションの準備作業で作成した JAR ファイルを EC2 インスタンス(WEB/AP)上に転送します。

手順2-1.TeraTerm で[ファイル] > [SSH SCP]をクリック

手順2-2.Secure File Copy のウィンドウで、[from] の右側の [...] をクリック

手順2-3.デスクトップにコピーした JAR ファイルを選択して [Send] をクリック

手順3.JAR ファイルが転送できたことを確認する

以下のコマンドを実行して、jar が正しく転送されたことを確認します。

```
ls -l
```

```

-rw-r--r-- 1 ec2-user ec2-user 160437916 Feb 25 13:48 jdk-16.0.2_linux-x64_bin.rpm
-rw-r--r-- 1 ec2-user ec2-user 53758240 Jun 24 2022 oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 702200 Jun 24 2022 oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 48849376 Feb 25 13:55 shared_shop-0.0.1-SNAPSHOT.jar

```

手順4.tar ファイル「images.tar」を EC2 インスタンスに転送する

手順2と同様に TeraTerm で[SSH SCP]を利用して EC2 インスタンスへ転送します。

手順5.tar ファイル「images.tar」が転送できたことを確認する

以下のコマンドを実行して、tar が正しく転送されたことを確認します。

```
ls -l
```

```
-rw-r--r-- 1 ec2-user ec2-user 2426880 Feb 25 13:58 images.tar
-rw-r--r-- 1 ec2-user ec2-user 160437916 Feb 25 13:48 jdk-16.0.2_linux-x64_bin.rpm
-rw-r--r-- 1 ec2-user ec2-user 53758240 Jun 24 2022 oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 702200 Jun 24 2022 oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 48849376 Feb 25 13:56 shared_shop-0.0.1-SNAPSHOT.jar
```

手順6.「images.tar」ファイルを展開する

以下のコマンドを実行して、「images.tar」ファイルを展開します。

```
tar -xvf images.tar
```

手順7.tar ファイルの展開されているかを確認する

以下のコマンドを実行して、「images.tar」ファイルが展開され、「shared\_shop」ディレクトリが作成されていることを確認します。

```
ls -l
```

```
-rw-r--r-- 1 ec2-user ec2-user 2426880 Feb 25 13:58 images.tar
-rw-r--r-- 1 ec2-user ec2-user 160437916 Feb 25 13:48 jdk-16.0.2_linux-x64_bin.rpm
-rw-r--r-- 1 ec2-user ec2-user 53758240 Jun 24 2022 oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 702200 Jun 24 2022 oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm
drwxrwxr-x 3 ec2-user ec2-user 20 Feb 25 14:35 shared_shop
-rw-r--r-- 1 ec2-user ec2-user 48849376 Feb 25 13:56 shared_shop-0.0.1-SNAPSHOT.jar
drwxrwxr-x 2 ec2-user ec2-user 6 Feb 25 14:18 temp
```

手順8.images ディレクトリを移動する

以下のコマンドを実行して、「shared\_shop/images」ディレクトリを jar ファイルと同じ階層に移動します。

```
mv shared_shop/images/ ~
```

手順9.images ディレクトリが jar ファイルと同じ階層にあることを確認する

以下のコマンドを実行して、jar ファイルと images ディレクトリが同じ階層にあることを確認します。

```
ls -l
```

```
drwxrwxr-x 3 ec2-user ec2-user 4096 Feb 25 14:35 images
-rw-r--r-- 1 ec2-user ec2-user 2426880 Feb 25 13:58 images.tar
-rw-r--r-- 1 ec2-user ec2-user 160437916 Feb 25 13:48 jdk-16.0.2_linux-x64_bin.rpm
-rw-r--r-- 1 ec2-user ec2-user 53758240 Jun 24 2022 oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 702200 Jun 24 2022 oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm
drwxrwxr-x 2 ec2-user ec2-user 6 Feb 25 14:41 shared_shop
-rw-r--r-- 1 ec2-user ec2-user 48849376 Feb 25 13:56 shared_shop-0.0.1-SNAPSHOT.jar
drwxrwxr-x 2 ec2-user ec2-user 6 Feb 25 14:18 temp
```

#### 手順1 0.Web アプリケーションを起動する

以下のコマンドを実行し、アプリケーションを起動します。

```
java -jar shared_shop-0.0.1-SNAPSHOT.jar
```

#### 手順1 1.Web アプリケーションの起動状態を確認する

コマンド実行後は、様々な起動時の情報が表示され、最後の方に「Started SharedShopApplication」と確認できれば正常に起動できています。

```
2022-06-21 05:46:01.864 INFO 2689 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding
welcome page template: index
2022-06-21 05:46:02.299 INFO 2689 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
started on port(s): 55000 (http) with context path '/shared_shop_answer'
2022-06-21 05:46:02.323 INFO 2689 --- [main] j.c.s.shop.SharedShopAnswerApplication : Started
SharedShopAnswerApplication in 12.022 seconds (JVM running for 13.163)
```

以上で Web アプリケーションのデプロイは終了です。

なお、正常に起動できなかった場合、エラーメッセージが表示されます。エラーメッセージの内容から判断して、application.properties やソースコードの修正を行い再ビルドし、デプロイする必要があります。

## 作業4.Web ブラウザからアプリケーションにアクセスする

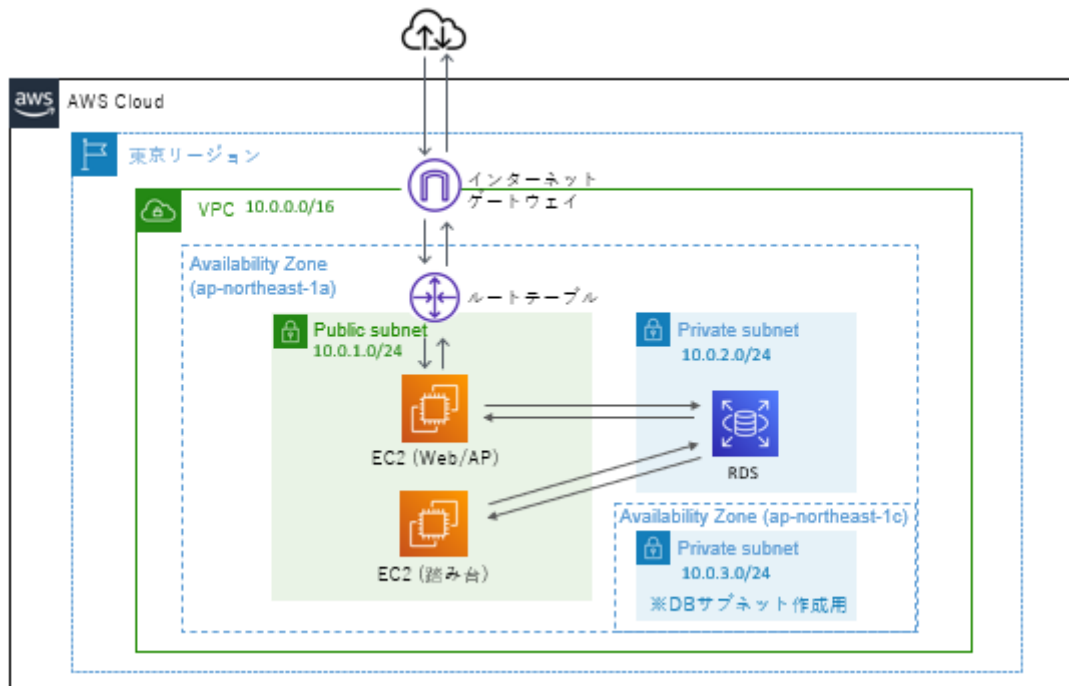
### 【作業の目的】

EC2 インスタンス上で起動している WEB アプリケーションを、作業用 PC の Web ブラウザからアクセスし利用します。

### 【作業概要】

ここで行う作業の概要は下記の通りです。

- 作業用 PC 上の Web ブラウザから EC2 インスタンス上の WEB アプリケーションが利用できることを確認



【作業手順】

手順1.Web ブラウザを起動する

手順2.Web ブラウザの URL 欄に EC2 インスタンス(WEB/AP)のパブリック IP アドレスを含む URL を指定する  
Web ブラウザのアドレスバーに以下の URL を貼り付けてアプリケーションにアクセスします。

http://[EC2 インスタンス (WEB/AP) のパブリック IP アドレス]:55000/shared\_shop/

手順3.Web アプリケーションの動作を確認する

以下のようにアプリケーションが表示されれば、ログインや画面遷移ができることを確認しましょう。特にデモンストレーションを行う機能については、しっかりと動作確認をしておきましょう。





## 作業5.【デプロイ後修正した場合】、再ビルドしデプロイする

AWS 環境へデプロイした後、下記のような状況が発生した場合、再度 JAR ファイルをビルドする必要があります。

- ・ 状況 1: Web アプリケーション起動時に DB 関連のエラーが発生して起動できない
  - 考えられる原因: application.properties の設定が間違っている
- ・ 状況 2: Web アプリケーションは正常に起動できたが、特定の画面に遷移するとシステムエラーが発生し、TeraTerm の画面に Thymeleaf 関連のエラーが表示されている
  - 考えられる原因: コントローラクラスのメソッドで、return に指定している文字列の先頭に / を付けている等  
→ [2-2 デプロイ用のソースコード、画像ファイルなどの準備](#)を確認の上修正してください。

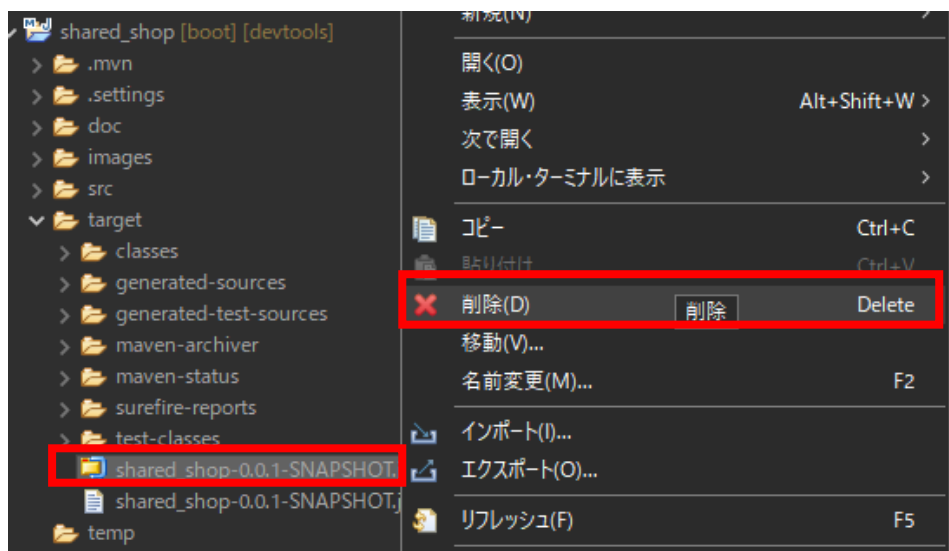
上記の状況以外でも、デプロイ後バグを発見した場合は、ソースコードを編集し、再度 JAR ファイルをビルドする必要があります。

その場合、下記の手順に従ってください。

### 【作業手順】

#### 手順1.JAR ファイルを削除する

「プロジェクト・エクスプローラー」ビュー上で、JAR ファイルを選択して右クリック→「削除」をクリックします。



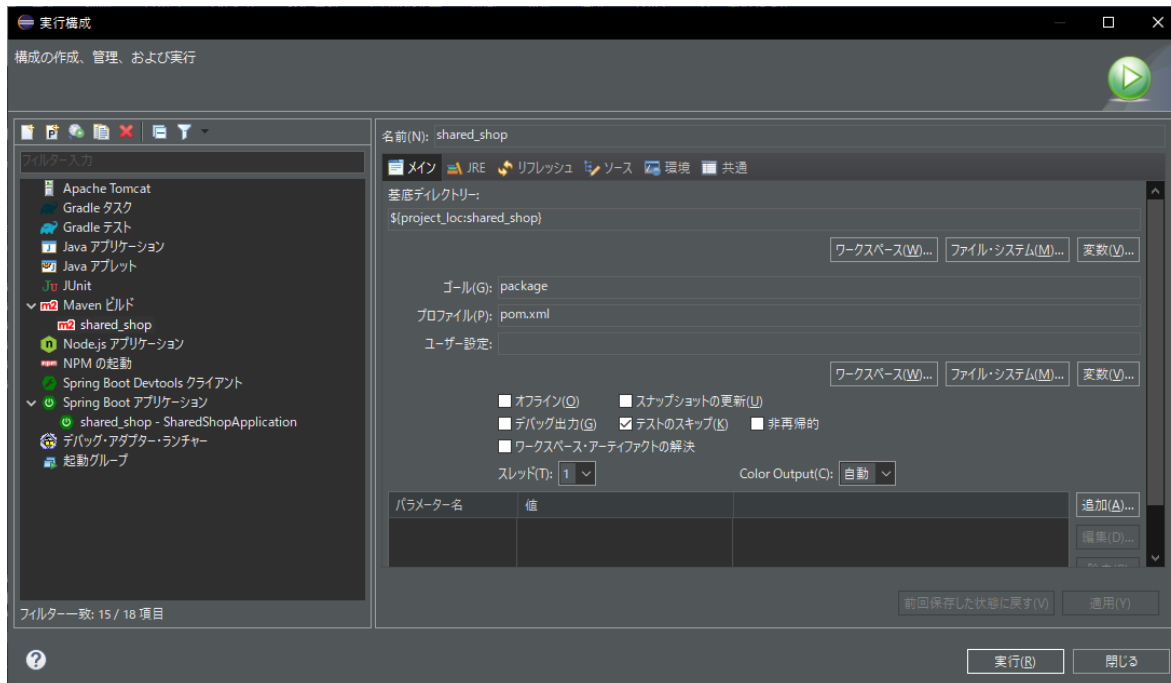
#### 手順2.実行の構成を開く

「プロジェクト・エクスプローラー」ビュー上でビルド対象のプロジェクトを選択して右クリックし、[実行] → [実行の構成(N)…]を選択します。

### 手順3.プロジェクトをビルドする

[実行の構成画面]で、表示された内容が下記になっていることを確認し「実行」をクリックします。

- ゴール: package
- プロファイル: pom.xml
- テストのスキップ: チェックあり



### 手順4.ビルド結果を確認する

Eclipse のコンソールに下記のような「BUILD SUCCESS」というメッセージが表示されることを確認します。

```
[INFO]
[INFO] -----<jp.co.sss.shared_shop>-----
[INFO] Building shared_shop 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] <<<<<<< 省略 >>>>>>>
[INFO] ---maven-jar-plugin:3.2.2:jar (default-jar) @ shared_shop---
[INFO] Building jar: C:\¥pleiades-2022-09¥workspace¥shared_shop¥target¥shared_shop-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] ---spring-boot-maven-plugin:2.7.7:repackage (repackage) @ shared_shop---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 7.934 s
[INFO] Finished at: 2023-02-16T20:47:01+09:00
[INFO]
[WARNING] The requested profile "pom.xml" could not be activated because it does not exist.
```

手順5.JAR ファイルが生成されていることを確認する

「プロジェクト・エクスプローラー」ビューで「target」フォルダを開き、JAR ファイルがあることを確認します。

JAR ファイルのファイル名 : [プロジェクト名]-0.0.1-SNAPSHOT.jar

※ファイルが見つからない場合は、プロジェクトをリフレッシュ(F5 キー)を押して再度確認してください。

手順6.JAR ファイルをデスクトップ上にコピーする

※AWS 環境へアップロードする際に JAR ファイルの場所が把握しやすいように行う作業です。

手順7.TeraTerm で EC インスタンス(WEB/AP)に接続する

手順8.JAR ファイルを EC2 インスタンス(WEB/AP)に転送する

TeraTerm で[ファイル] > [SSH SCP]をクリックし、再作成した JAR ファイルを EC2 インスタンス(WEB/AP)上に転送します。

手順9.JAR ファイルが転送できたことを確認する

以下のコマンドを実行して、JAR ファイルの日付が最新になっていることを確認します。

```
ls -l
```

```
-rw-r--r-- 1 ec2-user ec2-user 160437916 Feb 25 13:48 jdk-16.0.2_linux-x64_bin.rpm
-rw-r--r-- 1 ec2-user ec2-user 53758240 Jun 24 2022 oracle-instantclient19.3-basic-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 702200 Jun 24 2022 oracle-instantclient19.3-sqlplus-19.3.0.0.0-1.x86_64.rpm
-rw-r--r-- 1 ec2-user ec2-user 48849376 Feb 25 13:56 shared_shop-0.0.1-SNAPSHOT.jar
```

手順10.Web アプリケーションを起動する

以下のコマンドを実行し、アプリケーションを起動します。

```
java -jar shared_shop-0.0.1-SNAPSHOT.jar
```

手順11.Web アプリケーションの起動状態を確認する

コマンド実行後は、様々な起動時の情報が表示され、最後の方に「Started SharedShopApplication」と確認できれば正常に起動できています。

## 6 成果報告会当日の作業

本研修で利用している AWS 環境は、夜間に RDS や EC2 のインスタンスを停止しているため、デプロイ作業を初めて実施したときと成果報告会当日では IP アドレスが異なっています。また、作業用 PC も IP アドレスが変更されている可能性があります。成果報告会当日に必ず下記の作業を行い、動作確認しておきましょう。

### 【作業手順】

手順1.作業用 PC の IP アドレスを EC2 インスタンス(WEB/AP)のセキュリティ設定に反映する  
EC2 インスタンス(WEB/AP)のセキュリティグループで、インバウンドルールを編集します。

- ・ 編集箇所:タイプ SSH のソースタイプ

ソースタイプを一度削除し、再度[自分の IP]を選択すると、新たな IP アドレスが設定されます。

手順2.RDS を起動する

RDS のインスタンスを選択し、アクション→インスタンスの開始をクリックすると、インスタンスを開始できます。

RDS インスタンスの一覧画面を何度かリロードして、DB インスタンスのステータスが [利用可能] になっていることを確認後、次の手順に進みます。

手順3.EC2 インスタンス(WEB/AP)を開始する

EC2 インスタンス(WEB/AP)を選択し、アクション→インスタンスの開始をクリックすると、インスタンスを開始できます。

EC2 インスタンス(WEB/AP)のステータスが[実行中]になっていることを確認後、次の手順に進みます。

手順4.起動した EC2 インスタンスのパブリック IP アドレスを確認する

インスタンスの一覧から起動した EC2 インスタンスをチェックすると、画面下部にインスタンスの詳細情報が表示されます。その中に[パブリック IPv4 アドレス]が表示されています。

EC2 インスタンス(WEB/AP)のパブリック IP アドレスをコピーしておきます。

手順5.EC2 インスタンス(WEB/AP)に TeraTerm で接続する

手順4 でコピーした IP アドレスに SSH で接続します。キーペアファイルは準備作業で作成したものを利用します。

手順6.Web アプリケーションを起動する

TeraTerm で以下のコマンドを実行し、アプリケーションを起動します。

```
java -jar shared_shop-0.0.1-SNAPSHOT.jar
```

手順7.Web アプリケーションの起動状態を確認する

コマンド実行後は、様々な起動時の情報が表示され、最後の方に「Started SharedShopApplication」と確認できれば正常に起動できています。

手順8.Web ブラウザからアクセスし動作を確認する

手順4 でコピーした IP アドレスを含めた下記の URL にアクセスし動作を確認します。

```
http://[EC2 インスタンス(WEB/AP)のパブリック IP アドレス]:55000/shared_shop/
```