LoadBalancerとAutoScaling

ロードバランサーは、サーバー負荷(ロード)低減のために、リクエストを複数のサーバーに振り分け（バランス）し、負荷分散を図る仕組みです（AWSに限らず広くネットワークで利用される）

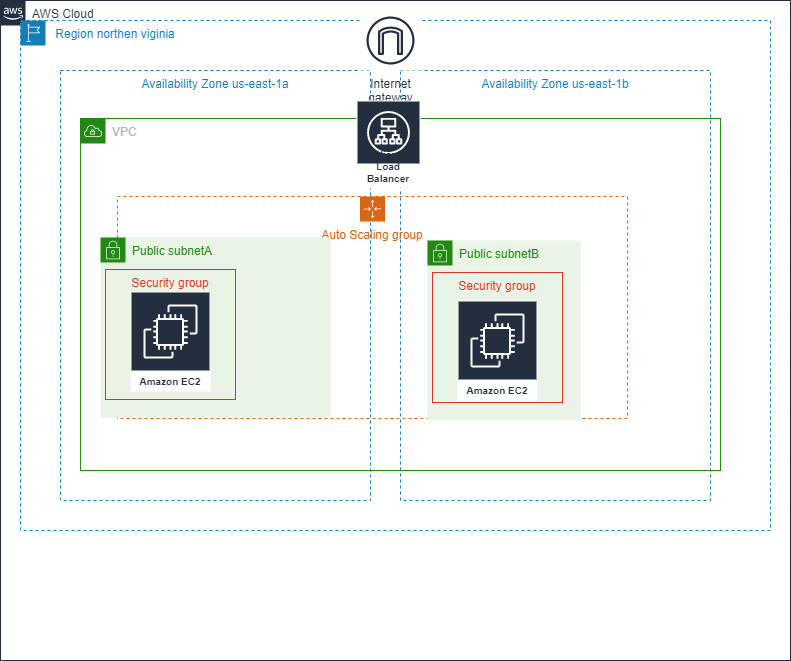
Autoscalingは、容量を自動で調整する仕組みです（AWSで構築したシステムの伸縮性を担う）

Learner Labの▶Start Labをクリックして、スタート（時間延長）

　ラボの制限時間4時間

シグナルでステータス確認できるラボ停止 ラボ起動途中 ラボ稼働中

Learner Labはサンドボックスラボと異なり、Resetしない限り作成したサービスは維持される（ラボを停止しても一定のクレジットが消費される）



VPC,EC2の準備と、準備の確認

お手軽設定で、VPCとパブリックサブネット２つ（インターネットゲートウェイ、ルートテーブルも）を作成  
EC2を１つ作成（名前：BastionSV /キーペアはSPIC0217.pem）、ユーザーデータに以下を設定  
  
  
  
  
  
  
  
  
  
ブラウザでパブリックIPv4アドレスにアクセスし、Apacheの起動を確認しておく  
起動に失敗している場合はSSHで直接接続して、設定を完了させる

#!/bin/bash

yum update -y

yum install -y httpd

systemctl enable httpd

amazon-linux-extras install -y php8.0

amazon-linux-extras install epel -y

yum install stress -y

systemctl start httpd

AS用に、EC2からAMIを作成する  
EC２を停止させ、アクション→イメージとテンプレート→イメージからAMIイメージを作成  
イメージ名：SPIC-AS-LB-AMI 説明もイメージ名と同じ

LB用セキュリティグループの作成（EC2と異なりSSH接続は要らないので専用ルールを作成する）

セキュリティグループ名：LB-sg VPCはデフォルトから変更し先ほど作ったものにする  
インバウンドルールを追加 タイプ：HTTP ソース：Anywhere-IPv4

LBの作成

1. ロードバランサー設定に必要なターゲットグループを作成する  
   Basic configuration：Instance  
   Target group name：SPIC-LB-trg  
   VPC： 準備で作成したVPC  
   Nextをクリックし、Register targetsの項目になるが現時点では設定できない  
   未設定のままCreate Target Groupをクリックし作成
2. ロードバランサー(Application Load Balancer)を作成(Create)する  
   Load balancer name：SPIC-LB  
   VPC： 準備で作成したVPC  
   Mappings：複数のAZを指定し、複数のサブネット(public)を選択  
   Security groups：LB-sg  
   Listeners and routing：SPIC-LB-trg(1で作成)  
   create load balancerをクリックし作成

Auto Scalingの設定

1. 起動テンプレートの作成  
   起動テンプレート名：SPIC-AStmp  
   Auto Scaling のガイダンス： チェックする→**□EC2 Auto Scaling で使用できるテンプレートを…**  
   起動テンプレートのコンテンツ：  
   自分のAMI：SPIC-AS-LB-AMI(準備で作成したもの)  
   インスタンスタイプ：t2.micro  
   キーペア：SPIC0217.pemを選択  
   ネットワーク設定  
    サブネット：起動テンプレートの設定に含めない  
    セキュリティグループ: 準備でEC2に適用しているもの(launch-wizard-1の可能性大)  
   起動テンプレートを作成をクリックし作成する
2. Auto Scaling グループの作成  
   Auto Scaling グループ名：SPIC-ASgrp  
   起動テンプレート：３で設定したSPIC-AStmpを選択  
   VPC： 準備で作成したVPC  
   アベイラビリティーゾーンとサブネット：Publicの複数AZ、複数サブネットを指定して 次へをクリック  
   ロードバランシング：既存のロードバランサーにアタッチ  
   既存のロードバランサーにアタッチ：SPIC-LB-trgを選択して 次へをクリック   
   グループサイズ：  
    希望する容量：3  
    最小容量：2  
    最大容量：6  
   スケーリングポリシー：  
    ターゲット追跡スケーリングポリシー  
    ターゲット値のみ30へ変更  
    次へをクリック  
    通知を追加する→次へをクリック  
    タグを追加する：タグを追加するをクリックし、  
    キー：Name 値：SPIC-ASedEC2 →次へをクリック  
   Auto Scaling グループを作成するをクリック

■確認

EC2インスタンスの一覧で複数のEC2が起動していることを確認

追加で起動したEC2(SPIC-ASedEC2)へアクセスし

Index.htmlを一部変更してみる→LBにより一定頻度で表示ページが変わる

stress -c 1コマンドを実行→5分後くらいにEC2が増える

追加の設定：

EFS：AWSのクラウド上にファイルシステムを用意し、EC2で立ち上げたシステムからアクセスできるようにするサービス EC2付属のストレージのEBSが1対１の接続に対し、EFSは複数のEC2からの接続に対応する(Webコンテンツの置き場所などとして利用)

準備：

1. EFS用セキュリティグループの作成  
   セキュリティグループ名：EFS-sg VPCは準備で作成したVPCに変更  
   インバウンドルールを追加 タイプ：NFS：Anywhere-IPv4
2. EC2へEFS用の設定追加（BastionSVにて実施）  
   sudo yum install -y amazon-efs-utils

ファイルシステム（EFS）の作成：

名前：SPIC-efs VPC: 準備で作成したVPC　可用性と耐久性：リージョン

＊**「カスタマイズ」**を選択して、詳細を設定

デフォルトセキュリティグループを削除して、EFS用に設定したセキュリティグループを適用

(複数のAZで設定、対象はpublicのサブネット)

EC2（BastionSV）とEFSを「アタッチ」する(作成したファイルシステム>アタッチ>DNS経由でマウント)。

「NFSクライアントの使用」をコピーし、EC２上でペースト．その後「efs」を「/var/www/html/」に変更（入力コマンドは以下の例のようになる）

例：sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport *fs-0f9c34d20db7552e1.efs.us-east-1.amazonaws.com:/* **/var/www/html/**

マウントが成功しているか、df -hコマンドで確認

fs-〜1.amazonaws.com:/ 8.0E 0 8.0E 0% **/var/www/html**

起動時に自動マウントするための設定を1行追加  
sudo vi /etc/fstab

*fs-0f9c34d20db7552e1.efs.us-east-1.amazonaws.com:/* /var/www/html/ efs defaults,\_netdev 0 0

viはiで編集モード、ESCでモード解除 :wqで保存して終了

ファイルのアップロードとEFSへのファイル保存：

S3-site-sample.zipをEC2へアップロードする（Windows：Teraterm、mac:scpコマンド）

zipファイルの展開  
unzip S3-site-sample.zip

ファイルのドキュメントルート（EFS）への移動

sudo mv S3-site-sample/\* /var/www/html/

起動テンプレート、AutoScalingの再設定

1. AMIイメージを再作成  
   イメージ名：SPIC-WEB-AMI2 説明もイメージ名と同じ
2. 起動テンプレートで、テンプレートを変更 (新しいバージョンを作成)をおこなう  
   AMIを①で作成したAMIに変更
3. Auto Scalingグループで起動テンプレートを編集し、Latestに変更
4. EC2のSPIC-ASedEC2のインスタンスを選択して、すべて停止

確認：

数分後Auto Scalingが動作し、SPIC-ASedEC2インスタンスの起動まで待つ  
ロードバランサーのDNS Nameにブラウザでアクセスする

サイトが表示されていれば、完了！