CPD Lab 3 - Virtualisation and Containerisation

Introduction

This lab covers the basic concepts of virtualisation and containerisation in AWS. You have already used EC2 instances (virtual machines or servers) and will cover more features of Amazon Machine Images (AMI).
You will use the docker commands in order to create and understand the container application architecture. For clusters you will interact with AWS Fargate which is a serverless compute engine for containers that works with Amazon Elastic Container Service (ECS) making it easier to focus on building applications.

Tasks:

- Task 1: Configuring and connecting to EC2 instance
- Task 2: Confirming configuration settings from Task 1
- Task 3: Creating an AMI/template
- Task 4: Using Docker to run a containerised application
- Task 5: Running a cluster with ECS
- Task 6: Releasing the allocated resources

Task 1: Configuring and connecting to EC2 instance

In this task you will use an Amazon Linux AMI to create an instance. You will configure EC2 such that a script runs at initial boot to perform update actions and some other configurations related to security.
For further details refer to ‘Link 1’ and ‘Link 2’ under ‘Links’ Section at the end of the document.

Hands-on

- Select EC2 from services in AWS Management Console
- Click ‘Launch Instance’
- Step 1: ‘Choose an Instance Type’
  - Click select button for following Amazon Machine Images (AMI) which would be at the top of the list shown

    Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0389b2a3c4948b1a0

- Step 2: Choose an Instance Type
  - Select t2.micro ‘Free tier eligible’, click Next
- Step 3: Configure Instance Details
  - For ‘Enable termination protection’ check ‘Protect against accidental termination’. Hover mouse over the ⓘ for information related to this.

- For 'Monitoring' check 'Enable CloudWatch detailed monitoring'. Hover mouse over the ⓘ for information related to this.
- Expand the 'Advanced Details' and for 'User data' enter following 'As text'

```
#!/bin/bash
sudo yum update –y
```

- Click Next

- Step 4: Add Storage – click Next
- Step 5: Add Tag – For 'Key' enter Name and for Value enter 'MyDockerEC2' and click Next
- Step 6: Configure Security Group
  Security group settings are important as it acts as a firewall. By default all inbound traffic is disallowed except for port 22 (SSH)
  - Select 'Create a new security group', and enter MySG for 'Security group name' and any appropriate description of your choice under 'Description'
  - For 'Type' select 'All ICMP-IPv4', which autofills 'Protocol' as 'All', for 'Source' select 'My IP' which displays your IP address. Also add HTTP with port 80 and for source select 'My IP'
  - Click 'Review and Launch'
- Step 7: Review Instance Launch – review and launch the instance by clicking 'Launch'
- In the dialog box that pops up: Select an existing key pair or create a new key pair
  - Choose 'Create a new key pair'
  - Key pair name: MyKeyPair
  - Click 'Download Key pair' which downloads the MyKeyPair.pem file.
  - Click 'Launch Instances'
- Check the newly created instance under EC2 running instances

Task 2: Confirming configuration settings from Task 1

Step 1: Termination protection

You had enabled 'Termination Protection' in Task 1, Step 3. This prevents you from accidentally terminating the instance. You would now verify that this is the case

- Select MyDockerEC2 from the running instances
- Click 'Actions->Instance State->Terminate'
- In the resulting dialog box, you would not be able to terminate the instance

Step 2: Monitoring using CloudWatch

- Open the monitoring tab for MyDockerEC2 and you will see CloudWatch metrics
- Navigate to 'CPU Utilisation', 'Network In' and 'Network Out'.

Step 3: ICMP traffic: Checking tracert (Windows) / traceroute (Linux)

You had allowed inbound ICMP traffic in Task 1, Step 6 to your device's IP address. ICMP traffic is used by tracert to determine the path to the provided IP address https://support.microsoft.com/en-gb/help/314868/how-to-use-tracert-to-troubleshoot-tcp-ip-problems-in-windows

- Open a command window and enter > tracert "Public-IP-address-of-your-EC2-instance (MyDockerEC2)"
- This should trace complete path to your EC2 instance
- Note that according to the settings in Task 1, Step6, ICMP traffic will only be allowed from the IP address of your machine and will be rejected from all other IP addresses

Task 3: Creating an AMI/template

In task 1, you have created and configured an EC2 instance. Assume that it has lot of configuration settings peculiar to your needs that may have to be used again. In such a case, it is useful to make use of AMI/template feature to help quickly create such EC2 instances.

For further details refer to 'Link 3' and 'Link 4' under 'Links' Section at the end of the document.

AMI

- Select the Instance MyDockerEC2
- Select 'Actions->Create Image'
- In the dialog box, enter Image name as MyDockerAMI
- Click 'Create Image'
- You can see the AMI being created under EC2 Dashboard, under Images->AMIs
- Note that the visibility by default is 'private'
- Note the virtualization type and refer to https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/virtualization_types.html for further details
- Click 'Launch instance' and select 'My AMIs' where your own MyDockerAMI should be visible, click 'Select' in front of your MyDockerAMI
- This lets you configure the EC2 instance from step 1 through 7.

Template

You can also make use of the template feature to create EC2 instances

- In EC2 Dashboard, under Instances, select 'Launch Templates'
- Click 'Create launch template'
- Enter 'MyDockerTemplate' as the name

- Under 'Launch template contents', in the drop-down for AMI, select MyAMI created in Task 3 above
- You can select a lot of other configuration settings to customise the template but for this task, click 'Create launch template'
- Selecting 'Launch Instance' will now have an option to create EC2 instance using this template saving effort and time

Task 4: Using Docker to run a containerised application

The package manager **yum** will be used to install docker software on MyDockerEC2 (Amazon Linux machines)
Docker is open source software for developing, building, testing and deploying your applications inside containers. It allows packaging of an application with all dependencies.
For further details refer to 'Link 5' under 'Links' Section at the end of the document.

Step 1: Connect to MyDockerEC2 using browser-based SSH connection (or PuTTY)

- Select 'Actions->Connect' and select 'EC2 Instance Connect from under 'connection method'
- The username will be pre-selected as ec2-user
- Click 'Connect' to open an SSH window
- If this does not work then you could open a PuTTY connection as covered in Lab 2

Step 2: Installing docker

- In the SSH connection at the command prompt enter the following command to install docker:

```
sudo yum install –y docker
```

- Start the docker service as:

```
sudo service docker start
```

- To check if everything went well

```
sudo docker info
```

  This should print out useful information and you should observe entries for the 'Containers' and 'Images' being zero at this time

- Add the user ec2-user (logged in user) to docker group

```
sudo usermod –a –G docker ec2-user
```

Step 3: Running an AWS sample application in a container

- In order to download the sample php application you can install git though the following command in the SSH session:

  ```
  sudo yum install –y git

  git clone https://github.com/awslabs/ecs-demo-php-
  simple-app
  ```

- The directory structure can be explored using ls after changing to the directory

  ```
  cd ecs-demo-phpsimple-app

  ls
  ```

- To open the Dockerfile, you could use nano text editor as follows

  ```
  nano Dockerfile
  ```

- Note that the Dockerfile describes the dependencies, installation and configuration commands and that the application will run on port 80

- To close the nano editor press ctrl-x

- To create an image from the Dockerfile (note the . at the end),

  ```
  docker build –t mydockerimg .
  ```

- To check that the image has been created

  ```
  docker images
  ```

- To run the image **mydockerimg** in a container on MyDockerEC2

  ```
  docker run –p 80:80 mydockerimg
  ```

  this binds port 80 of the container to port 80 of the host machine

- You should now be able to connect to the running php application (running on the container) by connecting to the public IP of MyDockerEC2 instance
- Close the SSH connection

Task 5: Running a cluster with ECS using Fargate

In this task we will use Fargate to run an application in cluster.
For further details refer to ‘Link 6’ under ‘Links’ Section at the end of the document.

- Select ‘ECS’ from services

- On the 'Get Started' page there is a video available which you should view
- Click 'Get Started'
- From 'Container definition' select nginx and click Next
- Select 'Application Load Balancer'and click Next
- Specify 'Cluster name' as MyCluster and click Next
- Click Create
- The 'Launch Status' will be shown and will take few minutes to complete
- In order to connect to the service:
  - Select Tasks tab and click Task name
  - Click on 'ENI Id' to select the public IP
  - Open the public IP in a browser to connect to nginx
- Deleting the resources
  - Select Clusters->MyCluster
  - Select Tasks tab
  - Check the entry for the running task and click Stop
  - Click 'Delete Cluster'

Task 6: Releasing the allocated resources

Make sure that you have released all the resources created in Tasks above:

- Deleted EC2 instance, and KeyPair from Task 1
- Deleted template and AMI from Task 3
- Deleted the cluster from Task 5

Links

The links below are for your reference only in case further information is required to help complete the 'Tasks' above:

1. Amazon Machine Images https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

2. Running Commands on Your Linux Instance at Launch https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

3. Creating an AMI from an EC2 instance https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/tkv-create-ami-from-instance.html

4. Launch an instance from launch template https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html

5. Docker basics for ECS

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html

6. Deploy Docker Containers on Cluster (AWS ECS with Fargate)
   https://aws.amazon.com/fargate/
   https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_clusters.html