# Lab: DOCKER & python flask

## Learning Outcomes:

After completing this lab, you should be able:

1. To understand the challenges of deploying software

2. To understand docker framework.

3. To build simple docker image.

4. To run a custom docker container.

## Organisation

Please attempt this lab individually as you will need this to be completed in order to complete subsequent labs. Part 1 to Part 4 are compulsory. Part 5 is an optional challenging task.

# Part 1. Create DOCKER host

**to complete this part, you have two options:**

* **access amazon console management using your dit email and the password you created in aws educate.**

**in this part we will create a new virtual machine in aws.**

## Task 1: Launch Your Amazon EC2 Instance

* 1. In the **AWS Management Console** on the Services menu, click **EC2**.
  2. Click Launch Instance.
     1. Step 1: Choose an Amazon Machine Image (AMI)
* Select **Ubuntu Server 18.04 LTS (HVM)**
  + 1. Step 2: Choose an Instance Type:**t2.micro**
    2. Click Next: Configure Instance Details.
* For Network, select **Default VPC**
  + 1. Add Tag: key: Name, Value: Docker
* Configure Security Group, the final configuration of Security Group is shown in Figure
  + 1. **Enable SSH access at port 22, source: 0.0.0.0/0**
    2. **Enable HTTP access at port 80, source: 0.0.0.0/0**
    3. **Enable custom TCP Rule at port 5000, source: 0.0.0.0/0**

**A screenshot of a cell phone

Description automatically generated**

Figure SG Configuration

* 1. Review and lunch the instance and download the security key
     1. Select: Create a new key pair
     2. Key pair name: dockerkey
     3. Click on **Download Key Pair**

\*\* Save the key in secure place as you will need it to access the VM

* + 1. Lunch the instance
    2. Click **View** Instances.
    3. Click one the instance id to access EC2 dashboard

# Task 2: access Your Amazon EC2 Instance

To perform the next operations on your EC2 VM, you will login to the Amazon EC2 instance. Windows users should follow Task 2.1. Mac/Linux users should follow Task 2.2.

### Task 2.1: Windows SSH

This section is for **Windows users only**. If you are running Mac operating system or Linux, please click the "Mac/Linux Users" link above.

You will be using PuTTy to connect to the Amazon EC2 instance. If you do not have PuTTY installed on your computer, download it [here](https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe).

1. Launch **PuTTY** by running the putty.exe file you downloaded. The PuTTY Configuration dialog box that will be used for the next steps will appear.
2. For **Host Name**, enter the public IP address of your EC2 instance, which is shown to the bottom of the EC2 dashboard.
3. In the **Connection** list, expand **SSH**.
4. Click **Auth**.
5. In the **Private key file for authentication** box, browse to the .ppk file that you downloaded earlier, then click **Open**.
6. In the **PuTTY Security Alert** dialog box that opens, click **Yes** to add the key to PuTTY's cache.
7. For **login as:** type ubuntu and press **Enter**. You are now logged in to your **EC2 Server** instance.

### Task 2.1: MAC/Linux SSH

This section is for **Linux** and **Mac operating system** users only. If you are running **Windows**, click here to skip ahead to the next task.

1. Open the Terminal application on your computer.
2. To connect to your EC2 instance, run the following commands in Terminal (but substituting values as explained below):

**chmod 400 <path and name of pem file>**

**ssh -i <path and name of pem> ubuntu@<Public IP>**

* For **<path and name of pem file>**, substitute the path/filename to the .pem file you downloaded.
* For **<Public IP>**, enter the public IP address of your EC2 instance, which is shown to the left of the instructions you are currently reading.

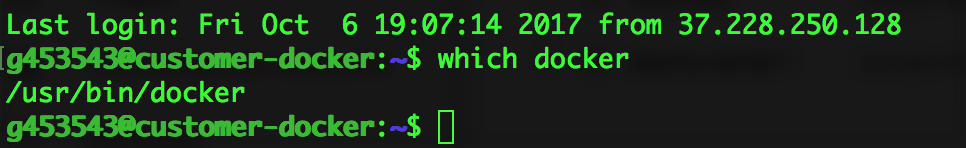
This FOLLOWING IS A SAMPLE OF THE FULL COMMAND:

ssh -i "dockerkey.pem" ubuntu@ec2-52-51-216-104.eu-west-1.compute.amazonaws.com

# Task 3: install docker in the vm

* 1. Verify that Docker is installed in the VM:

which docker



* if docker is not installed, install it using the following command:

curl -fsSL get.docker.com|sh

* Verity that Docker is running with full privilege without sudo:

sudo docker --version

* If you do not need sudo to run the docker commands, use the following command to run docker commands without sudo:

sudo usermod -aG docker <username>

* For example, the username in the figure above is: ubuntu, the command will be sudo usermod -aG docker ubuntu. In your case it should be your student number.
* **Make sure to restart the VM from compute engine after executing the command above. Using sudo reboot or from AWS console.**
  1. **Connect to the VM and check you can run docker commands without sudo**

# part 2: Learn Docker Containers

Docker is a software container platform. Developers use Docker to ship the software quickly and to eliminate interoperability problem of software dependencies. A container is used to package software in a format that can run isolated on a shared operating system.

* 1. **Visit the docker guide** [**https:/ /docs.docker.com/get-started**](https://docs.docker.com/get-started) **and complete the Tutorials from 1 to 2. (No need to do docker setup as we just did it in Part 1 of this lab sheet).**

**(Part 3 to 6 of docker tutorials are advanced tasks require you to install docker in your personal computer)**

# PART 3: build a docker container

* 1. Create a GitHub repository and call lab4-image
  2. Clone my repo from <https://github.com/baselm/lab4-base-image.git>

git clone <https://github.com/baselm/lab4-base-image.git>

* 1. Go inside the folder lab4-base-image

cd lab4-base-image

* 1. build a docker image and call it *mywebapp. Be careful about the dot in the command bellow. It tells docker to read the docker file form the current directory.*

docker build -t mywebapp .

* 1. check the image is added using *docker images* command.

docker images

* 1. Run the mywebapp image as a container

docker run -p 80:80 -d mywebapp

* 1. Check that the container is running:

docker ps

* 1. The container should be running at port 5000, but we specify to map the traffic from the container port 5000 into the instance port 80.
  2. Visit the instance public dns http://*public-dns.* you should be able to see the content of index.html
  3. Stop the running container using the container-id:
     1. To get the container id do:

docker ps

docker stop <container-id>

# Part 4: pull and run ready made docker image

* 1. Create a new Virtual machine from AWS EC2 and call it Dev-machine. Access the new created VM using ssh.
  2. Install Docker as described in Task 3.
  3. In this step you will use a pre-built docker image to run the application “mywebapp”. Inside the dev-machine VM ssh session pull mywebapp image using the following command:

docker pull baselm/mywebapp

* 1. Now run the image using

docker run -p 80:5000 -d baselm/mywebapp

* 1. Verify you can access the container from the web browser using the VM public dns/public IP
  2. Observe how it is easy to deploy a python application using docker containers.

# Part 5: Challenging Tasks (Optional)

* 1. Create a GitHub repo and call it lab4-repo.
  2. Pull the repo into the VM dev-machine.
  3. You need to use Flask framework, you can get more information about flask in this [link](http://flask.pocoo.org/docs/1.0/tutorial/factory/).
  4. Create a flask application with two routes:
  5. Main app.route(“/”) should return index.html page displaying a short BIO about yourself. You can include images …etc.
  6. App.route(“/contact-me”) should enable the user to send you an email. HTML forms or emailto tag can be used or any other method.

<a href="mailto:name@email.com">Link text</a>

* 1. Create a docker file for your application
  2. Create requirments.txt file
  3. Build the docker image
  4. Run it so you can share your BIO at port 80 of the VM.