Metropolitan State University

ICS 432: Distributed and Cloud Computing

Containers on the Cloud

# Overview

The goal of this lab is to understand how to create Docker images and launch containers. you will learn how to host a basic website by using Dockers containers in AWS Cloud9.

These are the steps:

* Use AWS Cloud9 to create an Amazon Elastic Compute Cloud (Amazon EC2) development instance that has Docker installed.
* Use this instance to create a Docker image and then launch a container.
* Create an Amazon Elastic Container Registry (Amazon ECR) repository to send the Docker image you will build.

Diagram

Description automatically generated

After completing this lab, you will be able to:

* Create a Dockerfile.
* Create a Docker image by using a Dockerfile.
* Run a container from a Docker image.
* Interact with and administer your containers.
* Create an Amazon ECR repository.
* Authenticate the Docker client to Amazon ECR.
* Push a Docker image to Amazon ECR.

# Step 1: Create a Cloud9 instance with Docker

1. From AWS console 🡪 Services 🡪 Cloud9.
2. Click on Create Environment (Name: DockerLab).
3. Environment settings:
   1. Create a new EC2 instance for environment (direct access).
   2. Instance type: t2.micro
   3. Platform: Amazon Linux 2
   4. Cost-saving setting: After 30 minutes (default)
   5. Click on: Next Step
4. Click on Create environment.
5. Once the environment is open, close the Welcome screen.
6. In the command window, type:

$docker -v

You should see the current version of Docker.

Graphical user interface, text

Description automatically generated

# Step 2: Modify the security group

The AWS Cloud9 deployment created an EC2 instance in your account. You will need to modify the Security Group (firewall) for that EC2 instance to allow you to test the application that you will deploy.

1. Go to EC2 dashboard and take note of the Public IP address of that EC2 instance to do that test.

Graphical user interface, application

Description automatically generated

1. Click on the security tab and you should see the name of the security group that the instance belongs to.

Graphical user interface, text, application, email

Description automatically generated

1. Click on the security group name and then click on Edit Inbound rules.
2. Choose **Add Rule** and input the following:
   1. For **Type**, select HTTP.
   2. For **Source**, select Anywhere.
   3. The other settings will be automatically populated for you.
   4. Click **Save rules**.

# Step 3: Work with Docker from AWS Cloud9

In this task, you will create a Docker image, create a Dockerfile, build a Docker image, and run a Docker container from the AWS Cloud9 console.

## Create a Docker Image

1. To seed your AWS Cloud9 filesystem with resources that include the ICS432 website. From Cloud9 console, click on File🡪Upload Local Files, then choose ICS432Fall2021.zip. You should see that the file ICS432Fall2021.zip*.zip* file has been added in your AWS Cloud9 filesystem.
2. Run the following command to unzip the file:

unzip ICS432Fall2021.zip

## Create a Dockerfile

1. In the AWS Cloud9 menu bar, choose **File** and then choose **New File**.
2. Enter the following text into the new **Untitled1** tab that just opened

FROM ubuntu:16.04

# Install apache and remove the list of packages downloaded from apt-get update

RUN apt-get update -y && \

apt-get install -y apache2 && \

rm -r /var/lib/apt/lists/\*

# Copy the website into the apache web root directory

COPY ICS432Fall2021 /var/www/html

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]

1. This is an example of a Dockerfile that does the following:
   1. Downloads the image ubuntu from an image repository.
   2. Installs apache web server and removes the packages downloaded from apt-get update.
   3. Copies your web site into the image.
   4. Exposes port tcp/80 to allow HTTP connections inbound.
   5. Starts apache.
2. To save the file, choose **File** and then choose **Save as**.
3. For **Filename**, enter: dockerfile. Make sure there are no spaces in the file name.
4. Save the file in the root **Docker** folder.

## Build a Docker Image

* 1. In Cloud9’s terminal, enter the following command. **NOTE**: Make sure to copy the whole command including the dot at the end.

$docker build -t ics432website-image .

This command takes few moments to build an image from a Dockerfile located in the current directory (.).

* 1. You should now have a Docker image. You can verify your image by entering the following command:

$docker images

This command should return a list of the docker images on the EC2 instance that includes the ics432website-image.

A picture containing text

Description automatically generated

## Run a Docker container

In this section, you will launch a container from the Docker image that you built.

To launch a container, from the AWS Cloud9 terminal enter the following command:

$docker run --name ics432website -d -p 80:80 ics432website-image

This command requests Docker to run a container, with the name ics432website, in daemon mode (non-interactive) and map tcp/80 outside the container to tcp/80 on the inside of the container.

To see if you container is running, enter the following command:

docker ps -a

Next, you will test the website. To do this, use the **IPv4 Public IP** that you noted earlier and paste it into a web browser.

For example: **http://<PublicIpAddress>**

You should see a web page similar to the following:

## Interact with your Docker container

In this section, you will use various commands to interact with your Docker container.

1. Go back to your AWS Cloud9 environment.
2. To open a bash prompt on your running container, run the following command:

$docker exec -i -t ics432website /bin/bash

The docker exec command runs a new command in a running container. The flag *i* makes it interactive by redirecting the STDIN.

You should now be logged into your container.

The prompt will look like the following:

**root@container-id:/#**



In the AWS Cloud9 terminal, enter the following command:

ls /var/www/html

Look at the filesystem. Confirm that the files in your resources folder are now in the */var/www/html* folder.

1. To exit from your container, run the following command:

exit

To stop the container, take a note of the container id by running

docker ps -a



ake note of your *CONTAINER ID*. You can use the CONTAINER ID to start and stop your containers. For example, if your container-id is *4b6c759654cd* and the name is *webapp*, you could stop your container using any of the following commands:

docker stop webapp

or

docker stop 4b6c759654cd

or

docker stop 4b

**Note** The last command works because Docker can see that there is only one container running that has a container id starting with '4b'. This can be useful and faster than using the entire id, particularly for testing.

1. To stop your container, enter the following command:

docker stop dac116ab04ae or docker stop ics432website

If you try to access the web site now from the browser then you will get an error ‘This site can’t be reached’ because the web server is stopped.

You can start the container again using:

docker start ics432website

## Task 5: Create an ECR Repository and Push Your Image

Now that you have an image, the next step a developer would normally do is to send the image to a repository.

In this section, you will be using Amazon ECR as your private repository. You will:

* Create a repository.
* Tag your image with that new repository.
* Authenticate the docker client to your ECR repository.
* Push your image in that repository.

The first step is to create the ECR repository called ics432website. To create the repository, enter the following command:

The return value should be similar to the following:

Text

Description automatically generated

Note the **repositoryUri** without the double-quotes. You will use it in the following steps. In this example, the repositoryUri is: *436412075180.dkr.ecr.us-east-1.amazonaws.com/webapp*

1. Next, tag the ics432website-image ECR repository that you have just created. To tag the image, in the following usign the **repositoryUri** that you noted in the previous step and run the command:

docker tag ics432website-image <repositoryUri>

For example

docker tag ics432website-image 436412075180.dkr.ecr.us-east-1.amazonaws.com/ics432website

Text

Description automatically generated

To be able to do a push into the ECR repository, the first step is to authenticate the Docker client. AWS created an AWS CLI command, *aws ecr get-login*, to simplify this process. The command returns the *docker login -u ...* command. Instead of having you copy the return statement and paste it back, you will run the command by using *eval*, which will run the command returned by *aws ecr get-login*.

1. To authenticate the *docker* client, enter the following command:

eval $(aws ecr get-login --no-include-email)

The output should indicate *Login Succeeded*. You can ignore the Warning because using the *eval* command allowed you to not paste the password on standard input and to be visible in the history.

1. Now that you are logged in, you can push the image into the ECR repository. To push the image, in the following command, replace the <FMI> with the **repositoryUri** and run the command:

docker push <FMI>

Verify that the image was pushed to the repository:

aws ecr list-images --repository-name ics432website

The output of this command should look similar to the following:

Text

Description automatically generated

Go to Services🡪 ECS 🡪 Repositories

You should see the ics432website image listed in your repo.

## Running the container in GCP

From Cloud9 command line, run the following command to get the login credentials for your repository:

aws ecr get-login --no-include-email

You will get output similar to the following

A screenshot of a computer

Description automatically generated with medium confidence

Copy the login command and you are going to use it to log in from GCP command window.

Login to GCP console and make sure you are using your course account.

Open a command window and run docker version to make sure Docker is installed.

Text

Description automatically generated

Paste the login command that you copied from Cloud9. You should get a message Login Succeeded as follows.

Text

Description automatically generated

docker pull 436412075180.dkr.ecr.us-east-1.amazonaws.com/ics432website:latest

Note that the repository name is <your-aws-account-id>.dkr.ecr.us-east-1.amazonaws.com

run docker images to make sure the image is pulled

Graphical user interface, text

Description automatically generated

Run a container out of this image:

docker run --name ics432website -p 5000:80 <Image ID>

Click on Web preview

Graphical user interface, text, application, chat or text message

Description automatically generated

Change port, enter 5000, and click Change and Preview

Graphical user interface, text, application, chat or text message

Description automatically generated

You should then see the ICS 432 web site.

# Clean up

In AWS: delete the Cloud9 Environment.

delete the container from GCP by opening a new command window (from the + sign at the top of the command window) and run the following commands in the new command window.

docker ps 🡪 to get container ID

docker stop <container ID>

docker rm <container ID>