Name: Kazuki A. Ogata	**Date Performed:** November 21, 2023
Course/Section: CPE 232 - CPE31S5	**Date Submitted:** November 21, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st semester S.Y 2023-2024

Activity 11: Containerization

1. Objectives

Create a Dockerfile and form a workflow using Ansible as Infrastructure as Code (IaC) to enable Continuous Delivery process

2. Discussion

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Source: https://docs.docker.com/get-started/overview/

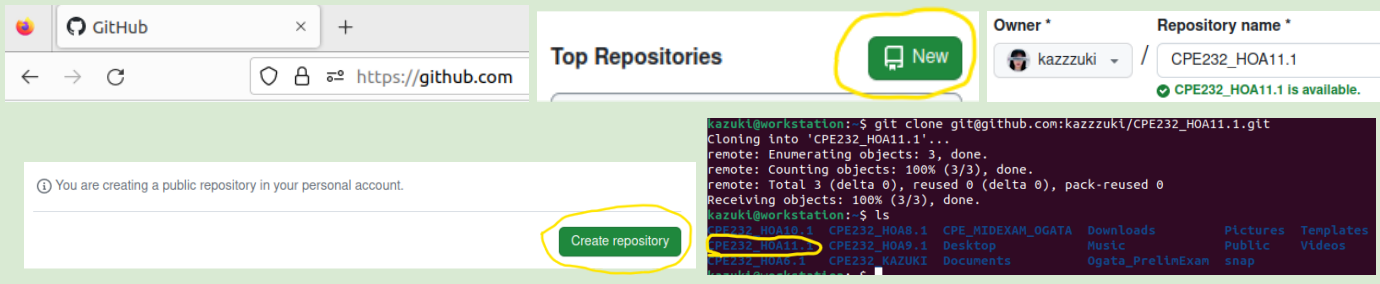
You may also check the difference between containers and virtual machines. Click the link given below.

Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

3. Tasks

1. Create a new repository for this activity.
2. Install Docker and enable the docker socket.
3. Add a Docker group to your current user.
4. Create a Dockerfile to install web and DB servers.
5. Install and build the Dockerfile using Ansible.
6. Add, commit and push it to your repository.

4. Output (screenshots and explanations)



I created a new repository for this activity and named it "CPE232_HOA11.1". Before cloning it to my workstation, although I already checked the installed git on my workstation from the past activities, I still check it today to check if it is still installed by using the command "which git". Then I proceed to clone my new repository to my workstation by using the command "git clone" to clone my new repository to my workstation. I used the command "ls" to show my cloned repository.

```
kazuki@workstation:~$ cd CPE232_HOA11.1
kazuki@workstation:~/CPE232_HOA11.1$ sudo nano inventory
kazuki@workstation:~/CPE232_HOA11.1$ cat inventory

[Hoa11_Ubuntu]
192.168.56.129  ansible_connection=ssh
```

```
kazuki@workstation:~/CPE232_HOA11.1$ sudo nano ansible.cfg
kazuki@workstation:~/CPE232_HOA11.1$ cat ansible.cfg

[defaults]

inventory = inventory
```

In this activity, I will only be using 1 server (Ubuntu). I created an inventory and ansble.cfg file. In the inventory file, I put the IP address of my Ubuntu server then added “ansible_connection=ssh” to specify the connection. In the ansible.cfg file, I put inventory as my default configuration.

```
kazuki@workstation:~/CPE232_HOA11.1$ ansible all --list-hosts
  hosts (1):
    192.168.56.129
kazuki@workstation:~/CPE232_HOA11.1$ ansible all -m ping
192.168.56.129 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

I used the command “ansible all -list-hosts” to make sure that ansible is allowed to my Ubuntu server. Then, I used the “ansible all -m ping” to show the connection. I did this to make sure that my target host is connected to my workstation so I can avoid errors.

```
kazuki@workstation:~/CPE232_HOA11.1$ mkdir -p roles/Hoa11_Ubuntu/tasks
kazuki@workstation:~/CPE232_HOA11.1$ ls
ansible.cfg  inventory  README.md  roles
kazuki@workstation:~/CPE232_HOA11.1$ cd roles
kazuki@workstation:~/CPE232_HOA11.1/roles$ ls
Hoa11_Ubuntu
kazuki@workstation:~/CPE232_HOA11.1/roles$ cd Hoa11_Ubuntu
kazuki@workstation:~/CPE232_HOA11.1/roles/Hoa11_Ubuntu$ ls
tasks
```

Applying the roles, I created directories using the “mkdir” command.

```
kazuki@workstation:~/CPE232_HOA11.1$ sudo nano /home/kazuki/CPE232_HOA11.1/roles/Hoa11_Ubuntu/tasks/main.yml
kazuki@workstation:~/CPE232_HOA11.1$ cat /home/kazuki/CPE232_HOA11.1/roles/Hoa11_Ubuntu/tasks/main.yml

- name: Update Ubuntu Package Cache
  apt:
    update_cache: yes
    state: present

- name: Install Docker Prerequisites on Ubuntu
  apt:
    name: "{{ item }}"
    state: present
  loop:
    - apt-transport-https
    - ca-certificates
    - curl
    - software-properties-common
  async: 3600
  poll: 0

- name: Add Docker GPG key on Ubuntu
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present
  async: 3600
  poll: 0

- name: Add Docker Repository on Ubuntu
  apt_repository:
    repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
    state: present
    update_cache: yes
  async: 3600
  poll: 0

- name: Install Docker on Ubuntu
  apt:
    name: docker-ce
    state: present
  async: 3600
  poll: 0

- name: Enable Docker Service on Ubuntu
  systemd:
    name: docker
    enabled: yes
    state: started

- name: Add a Docker Group to my Current User on Ubuntu
  user:
    name: kazuki
    groups: docker
    append: yes
```

This is the playbook for installing docker and adding a group. In installing docker, I firstly installed all the prerequisites like ca-certificates and curl, then created a GPG key and Repository for it. Then proceed installing docker. Most of the time when docker was installed, the service is automatically started/enabled, but I still added a task that starts/enabled docker just in case that it is not enabled by default and it is a good practice to still add that task in the playbook.

```
kazuki@workstation:~/CPE232_HOA11.1$ sudo nano install_docker.yml
kazuki@workstation:~/CPE232_HOA11.1$ cat install_docker.yml
---
- hosts: Hoa11_Ubuntu
  become: true
  roles:
    - Hoa11_Ubuntu
```

This is my main playbook where I used the roles syntax.

```
kazuki@workstation:~/CPE232_HOA11.1$ ansible-playbook --ask-become-pass install_docker.yml
BECOME password:

PLAY [Hoa11_Ubuntu] ************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.56.129]

TASK [Hoa11_Ubuntu : Update Ubuntu Package Cache] ******************************************************************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Install Docker Prerequisites on Ubuntu] *******************************************************
changed: [192.168.56.129] => (item=apt-transport-https)
changed: [192.168.56.129] => (item=ca-certificates)
changed: [192.168.56.129] => (item=curl)
changed: [192.168.56.129] => (item=software-properties-common)

TASK [Hoa11_Ubuntu : Add Docker GPG key on Ubuntu] *****************************************************************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Add Docker Repository on Ubuntu] **************************************************************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Install Docker on Ubuntu] *********************************************************************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Add a Docker Group to my Current User on Ubuntu] **********************************************
changed: [192.168.56.129]

PLAY RECAP *********************************************************************************************************
192.168.56.129             : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

This is the output of the playbook. It all shows changed, meaning it successfully does the tasks in the playbook.

```
kazuki@server3:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-11-21 19:57:28 +08; 6min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 45749 (dockerd)
      Tasks: 8
     Memory: 59.9M
        CPU: 1.134s
     CGroup: /system.slice/docker.service
             └─45749 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 21 19:57:27 server3 dockerd[45749]: time="2023-11-21T19:57:27.553222442+08:00" level=info m>
Nov 21 19:57:27 server3 dockerd[45749]: time="2023-11-21T19:57:27.562615948+08:00" level=info m>
Nov 21 19:57:27 server3 dockerd[45749]: time="2023-11-21T19:57:27.718536093+08:00" level=info m>
Nov 21 19:57:27 server3 dockerd[45749]: time="2023-11-21T19:57:27.720981645+08:00" level=info m>
Nov 21 19:57:28 server3 dockerd[45749]: time="2023-11-21T19:57:28.184589976+08:00" level=info m>
Nov 21 19:57:28 server3 dockerd[45749]: time="2023-11-21T19:57:28.267603321+08:00" level=info m>
Nov 21 19:57:28 server3 dockerd[45749]: time="2023-11-21T19:57:28.369213060+08:00" level=info m>
Nov 21 19:57:28 server3 dockerd[45749]: time="2023-11-21T19:57:28.369528286+08:00" level=info m>
Nov 21 19:57:28 server3 dockerd[45749]: time="2023-11-21T19:57:28.437710808+08:00" level=info m>
Nov 21 19:57:28 server3 systemd[1]: Started Docker Application Container Engine.
lines 1-22/22 (END)
kazuki@server3:~$ id kazuki
uid=1000(kazuki) gid=998(docker) groups=998(docker),27(sudo)
```

I used the command “sudo systemctl status docker” to check the status of docker if it is installed. As we can see the status is currently active without using the starting commands in my playbook. And for the added group, I used the command “id kazuki”.

```
kazuki@workstation:~/CPE232_HOA11.1$ cat dockerfile

FROM ubuntu:20.04

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install -y apache2
RUN apt-get install -y apache2-utils
RUN apt-get install -y php
RUN apt-get install -y php-mysql
RUN apt-get clean
RUN rm -rf /var/lib/apt/lists/

EXPOSE 80
CMD service apache2 start && service myqsql start
```

Before making a task in the playbook, I created a separate file for dockerfile. I included the version of my Ubuntu, working directory, update, install web, install db servers, and then start them. These are my resources for this:

http://surl.li/nkcli | http://surl.li/nkcls

```
- name: Copy dockerfile on Ubuntu
  copy:
    src: dockerfile
    dest: /var/app/dockerfile

- name: Install and Build Dockerfile on Ubuntu
  command: "docker build -t dockerimage:tag /var/app"

- name: Create + run web server
  docker_container:
    name: apache2-container
    image: httpd:2.4
    ports:
      - "80:80"
    state: started

- name: Create + run db server container
  docker_container:
    name: mysql-container
    image: mysql:latest
    env:
      MYSQL_ROOT_PASSWORD: mysecretpassword
    ports:
      - "3306:3306"
    state: started
```

This is the tasks for installing and building a dockerfile. I first created a task that copied the file that I created earlier to my target host. The proceed installing and building dockerfile using the command “docker build”. I added a task in creating and running a container for web and db servers and added images according to their specifications.

```
TASK [Hoa11_Ubuntu : Copy dockerfile on Ubuntu] ***************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Install and Build Dockerfile on Ubuntu] **
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Create + run web server] ******************
changed: [192.168.56.129]

TASK [Hoa11_Ubuntu : Create + run db server container] ********
changed: [192.168.56.129]
```

This is their output, the copy dockerfile was successful.

```
kazuki@server3:~$ cat /var/app/dockerfile

FROM ubuntu:20.04

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install -y apache2
RUN apt-get install -y apache2-utils
RUN apt-get install -y php
RUN apt-get install -y php-mysql
RUN apt-get clean
RUN rm -rf /var/lib/apt/lists/

EXPOSE 80
CMD service apache2 start && service myqsql start
```

This is the proof that the task “copy dockerfile” was successfully executed. I used the command “cat” to show the contents of the destination path I put in my playbook.

```
kazuki@server3:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
dockerimage   tag       52f7c549fa9d   27 minutes ago   265MB
```

In building the docker image, I used the command “docker images” to show the image build when the playbook was executed. It shows the “dockerimage” meaning the execution was successful.

```
kazuki@server3:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                 NAMES
34d80e270e02   mysql:latest   "docker-entrypoint.s…"   37 minutes ago   Up 37 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp     mysql-c
ontainer
508b82115bfa   httpd:2.4      "httpd-foreground"       37 minutes ago   Up 37 minutes   0.0.0.0:80->80/tcp                    apache2
-container
```

This is the proof of the tasks that create containers for web and db servers. I used the command “docker ps” and it shows the container ID, image, created (time), status, ports, and names.

```
kazuki@server3:~$ docker exec -it apache2-container /bin/bash
root@508b82115bfa:/usr/local/apache2# apache2 --version
bash: apache2: command not found
root@508b82115bfa:/usr/local/apache2#
exit
kazuki@server3:~$ docker exec -it mysql-container /bin/bash
bash-4.4# mysql --version
mysql  Ver 8.2.0 for Linux on x86_64 (MySQL Community Server - GPL)
bash-4.4# exit
```

The dockerfile that I created successfully installed the db server. I used the command “docker exec” to go inside the container and checked the installed db server, as we can see, it successfully shows the version of the db server. In my web server, I tried every solution I found on the Internet why it is not installed but none works.

```
kazuki@workstation:~/CPE232_HOA11.1$ git add .
kazuki@workstation:~/CPE232_HOA11.1$ git commit -m "hoa11"
[main 70ae528] hoa11
 2 files changed, 10 insertions(+), 10 deletions(-)
kazuki@workstation:~/CPE232_HOA11.1$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 669 bytes | 223.00 KiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:kazzzuki/CPE232_HOA11.1.git
   11f607f..70ae528  main -> main
```

I used the command “git add .” to add all the files in my current directory to my GitHub repository. I used the command “git commit -m “hoa10” command to commit all changes and added a message. Lastly, I push it.

GitHub Link: https://github.com/kazzzuki/CPE232_HOA11.1

Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?
   - The benefits of implementing containerization are consistency in different environments, enhanced application isolations, efficient use of resources, and cost-effectiveness. Let me give a scenario, for example there is a librarian organizing books. Without “containerizations” each time this librarian moves the book collection, they have to rearrange the shelves again and again. While with “containerizations” it is like having a special box that contains all in the same category books and it is neatly organized, making it easier to move all of them together without worrying about rearranging each shelf. Basically, containerizations promotes consistency and efficiency in management.

Conclusions:

- In conclusion, this activity successfully demonstrated the implementation of containerization using Docker and Ansible for IaC. By creating a Dockerfile using Ansible. The containerization is supposed to work quickly but in this activity, I experience that the building of the dockerfile takes too long… Overall, this activity taught me the importance of using tools like Docker for efficient development and deployment of web and db servers.
- Additional (my solution to my problem). The problem that I encountered is that the building of docker images takes too long to execute/run. So, I researched and found the command “ENV DEBIAN_FRONTEND noninteractive” . This command sets the Debian package manager to run in non-interactive mode meaning it will not entertain unnecessary packages during the image building. They also said that this command is helpful for Docker building image processes to be more automated and efficient and especially reduced the execution time.