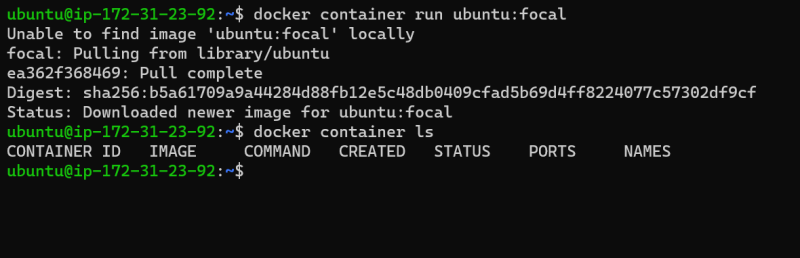
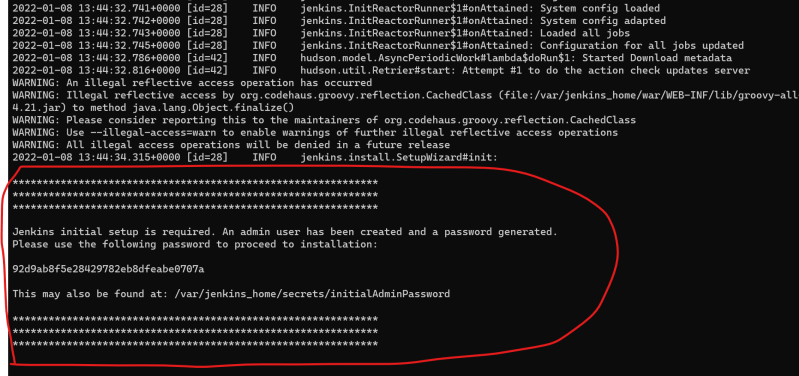
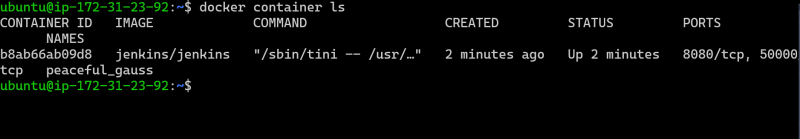
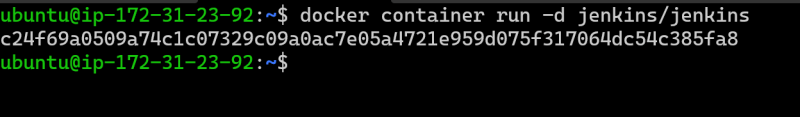
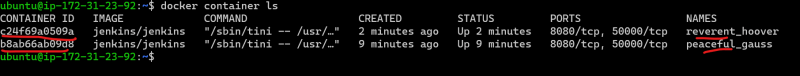
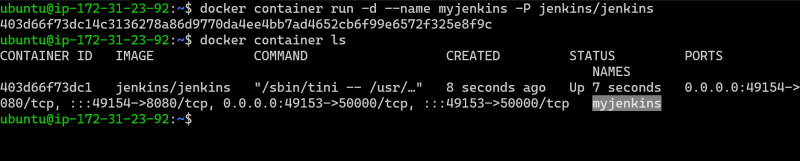
**Docker Container**

* To run a container, we need an image.
* Refer (<https://hub.docker.com/search?type=image> ) for the docker hub images
* Let’s try to create one container

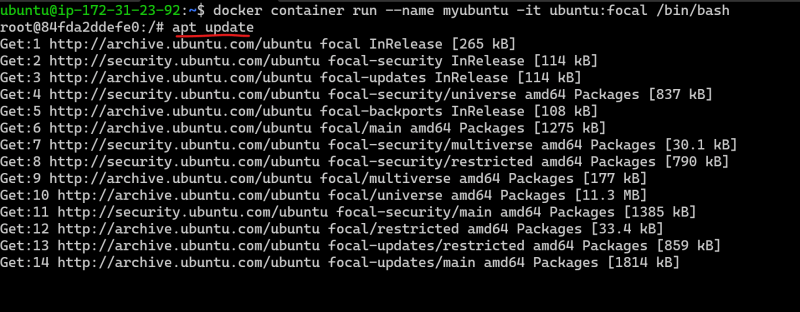
docker container run ubuntu:focal

  
\* We have observed no containers are running.  
\* When we try to look into all the containers we found our container is exited  
  
\* Now let’s try to run one more container docker container run jenkins/jenkins  
  
\* In this case we see the jenkins application is starting but it uses our terminal as standard output, so we cannot use this session, either login into same node using different ssh terminal  
  
\* Docker by default runs in the attached mode where it uses your terminal for showing standard output/standard errors from container  
\* So we use the detached mode docker container run -d <image>  
  
\* Whenever a container is created each container will get a  
\* unique container id  
\* unique container name (if you don’t name the container)  
  
\* To all the interactions with container we can use container id or container name  
\* Lets create a container jenkins/jenkins with name myjenkins  
  
\* To attach to a container which is running in detached mode

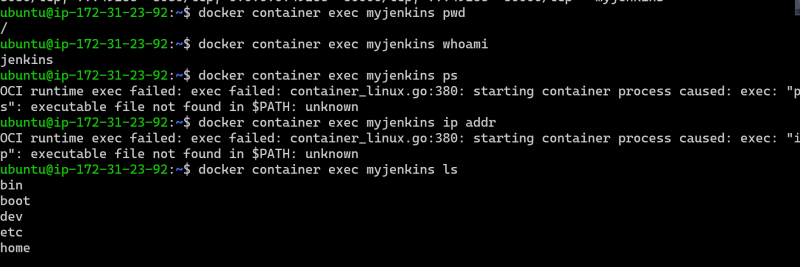
docker container attach <container name or id>

* Ensure you dont use Ctrl+C as it might stop the container
* When we start the ubuntu container it immedietly went into exited state, now lets start this container interactive using a terminal

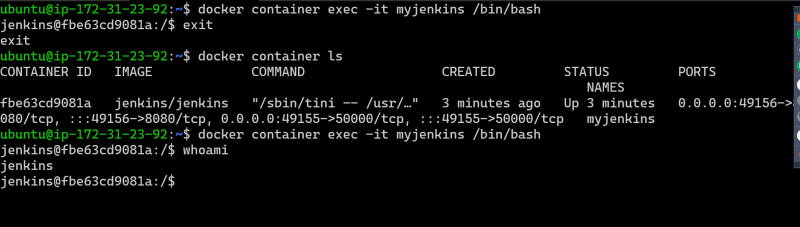
docker container run -it <image>

  
\* Now lets start the jenkins container, We can execute any comamnd inside the running contianer

docker container exec <container name/id> <command>

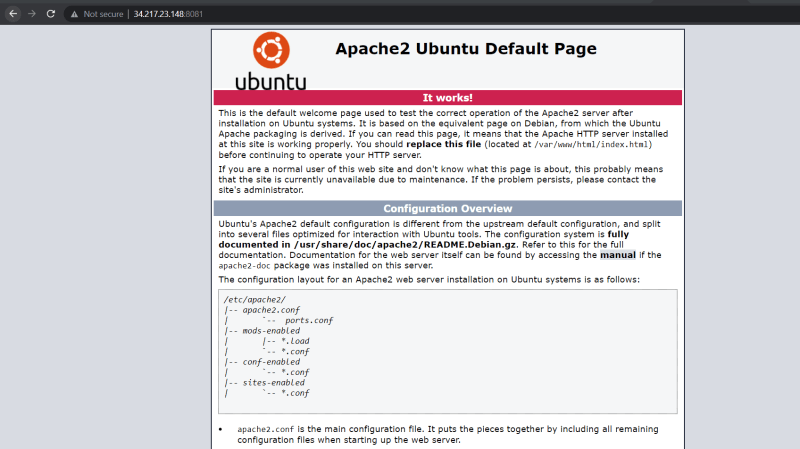
  
\* Using exec, we can login into the machine using terminal that is available

docker container exec -it <container name/id> <path-to-terminal>

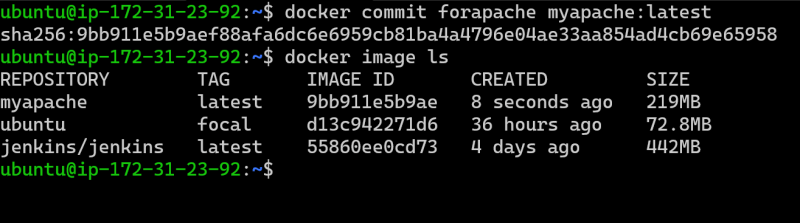
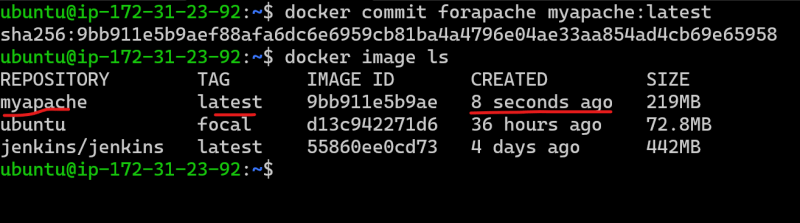
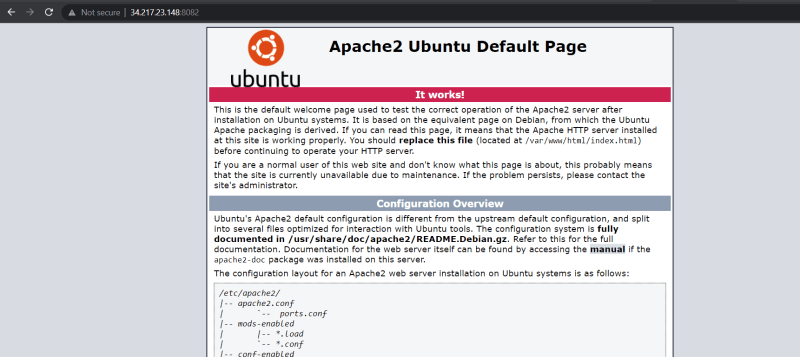


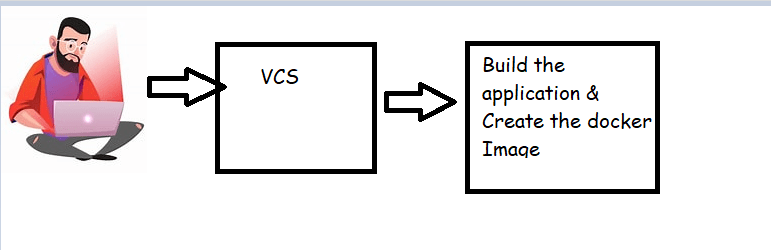
**How to create Docker Image?**

* We create Docker Image where our application will be configured.
* First Step: To create a Docker Image is to know how to configure/install/deploy your application manually.
* To create docker images we have two approaches
* Stupid Approach: Creating a Docker image from a running container
  + Lets try to create a docker image which will have apache server
  + The manual steps (on ubuntu flavor) are apt update && apt install apache2 -y && service apache2 start
  + Since the basic assumption is we need an ubuntu flavor, let create a container with ubuntu:focal image docker container run -it --name forapache -p 8081:80 ubuntu:focal /bin/bash

Now we can see the application is accessible over 8081 port of the docker host  


Now lets create an image from the container docker commit <container-name> <image>:<tag>

Now lets try to create a new container with our image  
  
  


Let’s consider first two steps of DevOps pipeline  


Whenever developer commits the code you need to create a docker image with a new tag.

* + Problem1: But in this approach we have to manually create a container, run the application installation/deployment steps and the create the docker image
  + Problem2: When we make changes in the docker image i.e. create a new tag it is very difficult to have history of changes made.

Pragmatic Approach: Creating a Docker image from Dockerfile Instructions

Let me create a sample Dockerfile

FROM ubuntu:xenial

RUN apt update

RUN apt install apache2 net-tools -y

EXPOSE 80

CMD ["service", "apache2", "start"]

\* Save this as `Dockerfile`

\* Whenever Developer submits code i will call `docker image build -t <image>:<tag> .`

\* Dockerfile is a text file which can be stored in Git (VCS) and we can record all the changes or track the history of changes.

**Dockerfile**

* Ensure you have all the steps for manually deploying your application.
* To create a Docker image i need to choose any existing image which is referred as base image.
* The docker image should be as lightweight as possible and should just enough tools to run our application