SCRATCH - ALPINE

Docker

Docker itself is not a VM, so there is no double layer of OS. Docker is a tool to run applications with settings that isolate them from other applications running on the same OS kernel.

Docker containers always run on a Linux kernel (except for the case of native Windows containers). On a non-Linux system (*e.g.*, a Mac) there is a Linux virtual machine, and containers always run on that VM. All containers share that "host" or VM kernel (and by default are prohibited from making changes that would affect the whole system).

"Operating system" can also imply a stack of libraries, utilities, and programs that run on top of the kernel. Docker lets you run an Ubuntu userspace, or an Alpine userspace, or some other userspace…or no userspace at all.

Docker does include a VM with Docker for Windows and Docker for Mac to run the Linux kernel so you can run Linux containers. There is an option to run native Windows containers with Server 2016, but if you are looking for minimal and efficiency, I would suggest looking elsewhere.

The closest things to what you are looking for are:

Unikernels: these are applications compiled into a kernel with everything else removed, designed to run inside of a VM for a very specialized task, often security related. These are still early in their development stage, but Docker does use some of their technology inside their project.

LinuxKit (part of the Moby Project): this is how Docker creates their VMs for Docker for Windows and Docker for Mac. It is a container based Linux operating system that you can custom compile with only the containers you want to run. Most of the focus of this is still designed for VMs, but bare metal is an option.

Scratch base image: if you statically compile your application to remove all of the library dependencies, you can have a container without any shell or other OS tools. This is often seen in Go binaries shipped as Docker containers to do a single task with a very small attack surface. As a Docker container, it still requires the underlying Linux OS to run the binary.

After you install and register GitLab runner,

```
$ sudo apt install docker.io
```

```
$ sudo nano /etc/sudoers
```

add the line below gitlab-runner ALL=(ALL) NOPASSWD: ALL

```
$ sudo usermod -aG docker gitlab-runner
```

```
$ sudo gitlab-runner restart
```

```
docker-compose up --build
```

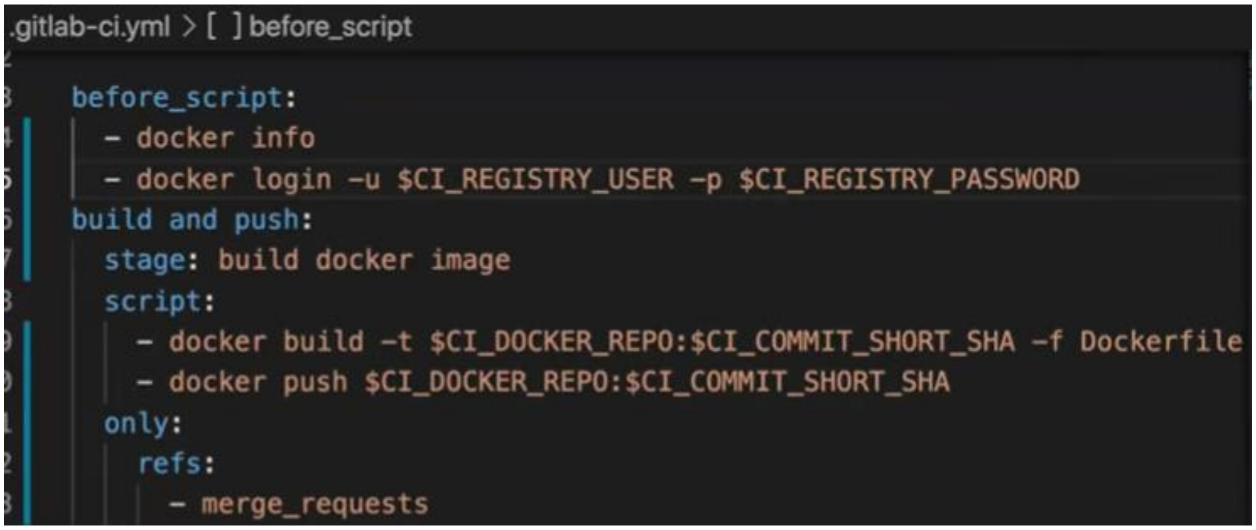
Docker

docker build –I <templateName>
docker image ls
docker image rm <imageName>
docker container run --name web –p 5000:80 alpine:3.9
docker ps –a <all contaners>

- docker run ... creates and runs a container
- docker stop ... stops a running container
- docker start ... starts a stopped container
- docker rm ... removes a stopped container

Docker Compose

- docker-compose up creates and runs a **collection** of containers
- docker-compose stop stops the containers
- docker-compose start starts the containers
- docker-compose down stops and removes the containers



.gitlab-ci.yml > [ ] before_script

```
before_script:
  - docker info
  - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
build and push:
  stage: build docker image
  script:
    - docker build -t $CI_DOCKER_REPO:$CI_COMMIT_SHORT_SHA -f Dockerfile
    - docker push $CI_DOCKER_REPO:$CI_COMMIT_SHORT_SHA
  only:
    refs:
      - merge_requests
```

1. The main aim of docker containers is to get rid of the infrastructure dependency while deploying and running applications. This means that any containerized application can run on any platform irrespective of the infrastructure being used beneath.

2. Docker images are executable packages(bundled with application code & dependencies, software packages, etc.) for the purpose of creating containers.

3. Docker images are created using Dockerfile

4. Docker Compose is a YAML file consisting of all the details regarding various services, networks, and volumes that are needed for setting up the Docker-based application.

Dockerfile:

```
        FROM ngnix
        ADD “index.html” “/usr/share/ngnix/html”
```

```
docker build –t <tagName>  <locationOfDockerFile>
```

To run:

```
docker run –p 80:80 <tagName>
```

docker-compose-build.yml

```
version: “3.8”
services:
        web:
                build: ngnix    - build anything under ngnix directory – has Dockerfile
```

```
docker-compose –file docker-compose-build.yml up --detach
```

Build image
Build – ngnix

5. Docker namespace is basically a Linux feature that ensures OS resources partition in a mutually exclusive manner. This forms the core concept behind containerization as namespaces introduce a layer of isolation amongst the containers. In docker, the namespaces ensure that the containers are portable and they don't affect the underlying host.

6.List status of all docker containers

```
docker ps -a
```

7. Docker image registry, in simple terms, is an area where the docker images are stored. Instead of converting the applications to containers each and every time, a developer can directly use the images stored in the registry.

8. Export a docker image as an archive

```
docker save -o <exported_name>.tar <container-name>
```

9. Import a pre-exported Docker image into another Docker host

```
docker load -i <export_image_name>.tar
```

10. Where are docker volumes stored in docker?

Volumes are created and managed by Docker and cannot be accessed by non-docker entities. They are stored in Docker host filesystem at `/var/lib/docker/volumes/`

11. What is the best way of deleting a container?

- docker stop <container_id>
- docker rm <container_id>

12. How will you ensure that a container 1 runs before container 2 while using docker compose?

Use depends_on

```
services:
 backend:
   build: .
   depends_on:
     - db
```

`docker-compose up -` command starts and runs the services in the dependency order specified.

`docker-compose up backend -` creates and starts DB (dependency of backend).

Docker supports more resource allocation options:

- CPU shares, via -c flag
- Memory limit, via -m flag
- Specific CPU cores, via --cpuset flag

If you want to change the filesystem size for Docker containers using the Device Mapper storage driver, you should use the `--storage-opt` flag of the Docker Engine.

Kubernetes:

PV vs PVC – A persistent volume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamic provisioned using Storage Classes. PVs are volume plugins like Volumes, but have a life cycle independent of any individual PODs that uses the PV.This API object captures the details of the storage, be that NFS, iSCSI or a cloud provider-specific storage system.

A PVC (Persistent Volume Claim) is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific level of resources (CPU and Memory). Claims can request specific size and access mode.

Tomcat

```
FROM centos
MAINTAINER aksarav@middlewareinventory.com
RUN mkdir /opt/tomcat/
WORKDIR /opt/tomcat
RUN curl -O https://www-eu.apache.org/dist/tomcat/tomcat-
8/v8.5.40/bin/apache-tomcat-8.5.40.tar.gz
RUN tar xvfz apache*.tar.gz
RUN mv apache-tomcat-8.5.40/* /opt/tomcat/.
RUN yum -y install java
RUN java -version
WORKDIR /opt/tomcat/webapps
RUN curl -O -L
https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
```