Register No: 20L31A05J4 Experiment No: 05 Date:

S. No	Component	Max. Marks	Marks Secured
1	Preparedness	2	
2	Viva-Voce	2	
3	Experiment	3	
4	Analysis & Record	3	
Total		10	
Date		Signature of the Lab teacher	

AIM: To Build, Deploy and Manage web or java application on Docker.

Theory:

Docker Build:

This command is used to build the Dockerfile to create images of application that can be shipped and run anywhere.

1. Docker Build with URL:

$docker build github.com/creack/docker-firepox

This will clone the git repository and use the cloned repository as context. The Dockerfile at the root will be used as "Dockerfile" to build the image. You can specify git:// or git@ scheme too.

Register No : | Experiment No : | Date:

$docker build -f ctx/Dockerfile http://server/ctx.tar.gz

This sends the URL http://server/ctx.tar.gz to the Docker daemon, which downloads and extracts the referenced tarball. The -f ctx/Dockerfile parameter specifies a path inside ctx.tar.gz to the Dockerfile that is used to build the image.

2. Docker Build with "-":

docker build - < Dockerfile

This will read a Dockerfile from STDIN without context.

$docker build - < context.tar.gz

This will build an image for a compressed context read from STDIN. Supported formats are: bzip2, gzip and xz.

3. Docker Build with .dockerignorefile:

$docker build .

docker build searches for a .dockerignore file relative to the Dockerfile name.

Using a Dockerfile based .dockerignore is useful if a project contains multiple Dockerfiles that expect to ignore different sets of files.

Register No : | Experiment No : | Date:

4. Docker Build with tag "-t":

$docker build -t vieux/apache:2.0.

5. Docker Build with "-f":

$docker build -f Dockerfile.colo.

This will use Dockerfile.colo to build the image instead of Dockerfile.

6. Docker build with add-host:

$docker build --add-host=docker:10.180.0.1.

7. Docker build with --target:

$docker build -t mybuildimage --target build-env

8. Docker build with --output:

docker build --output type=tar, dest=out.tar

Deploy:

Specifies how the container needs to be deployed.

- Endpoint_mode: can be either VIP or dns-rr (DNS-round robin)
- Labels: have labels specified for services.
- Mode: this can be either global / replicated.
- Placement: provides preferences, constraints and max number of replicas per container that it can spin up to.

Register No : Experiment No : Date:

- Constraints : such as

1) The role of the users using or creating the container.

2) The type of the os the container should run on.

- preferences :

spread : how the replicas should be spread across (zones)

- max number of replicas.
- replicas : replicas that can be running at any given time.
- Resources : provides the system limits that a container can have.

1) CPUS
2) Memory

- restart policy :

1) Condition : always/on-failure/none

2) Delay : how long to wait before start

3) max_attempts : how many times should the docker host try to bring up the container in case of failures.

4) window : how long to wait before deciding if restart is succeeded for not.

Register No : Experiment No : Date:

- Update-config :

1) Parallelism : how many containers can be updated.

2) Delay : time between an update b/w a group of containers.

3) Failure_action : if we should continue, rollback or pause.

4) Monitor : how long should we monitor before the next task is rolled out.

5) Max-failure-rate : failures to tolerate during an update.

6) Order : how to carry the update

i) stop-first : default ; old task is stopped first before starting a new one.

ii) start-first : start new task and then stop the old one.