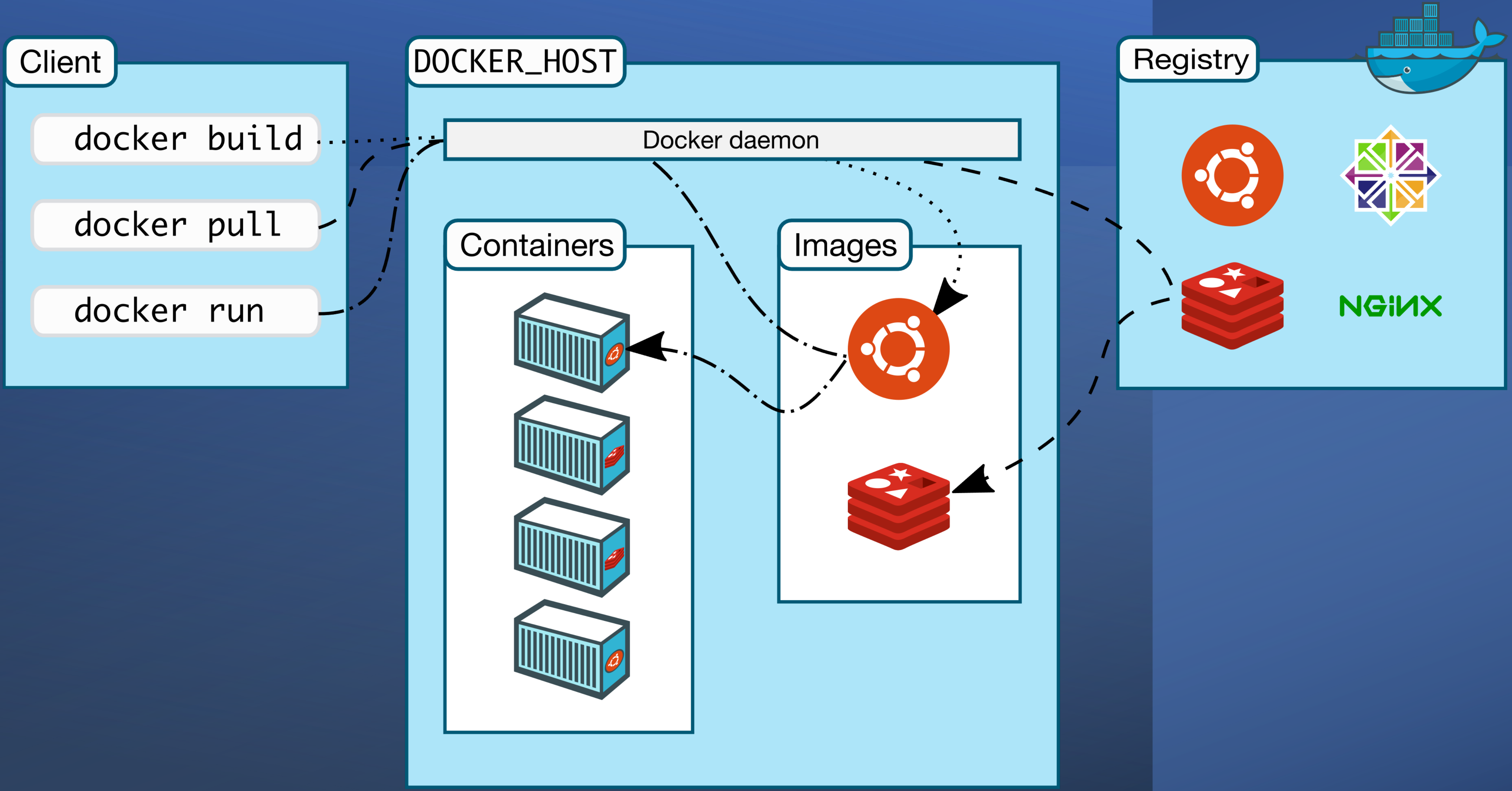
Present By Amit Ganvir

docker

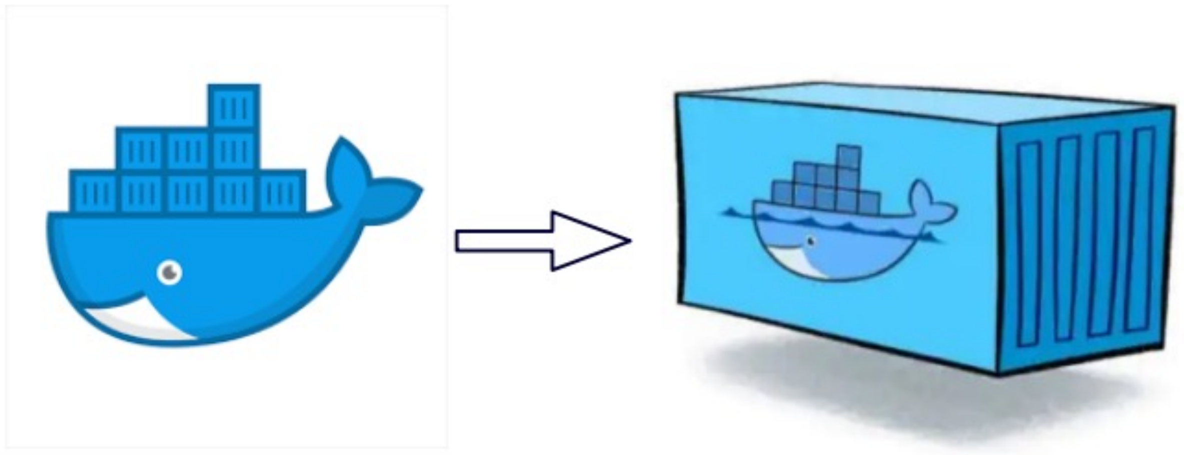
What is Docker

- 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
- Docker manage your applications
- Reduce the delay between writing code and running it in production

Docker Architecture



How Docker Container run



Install Docker

Step1: Install packages RHEL/Fedora/Centos

```
# yum install docker -y
```

OR

Step1: Install packages on Ubuntu/Debain

```
# apt-get install docker.io -y
```

Step2: To Start your docker service

```
# systemctl start docker
```

Step3: To Enable your docker service

```
# systemctl enable docker
```

Step4: To Check service status

```
# systemctl status docker
```

How to Run Docker Container

Step1: PULL Docker Image from Docker HUB

```
# docker pull busybox
```

Step2: Check your Docker Image

```
# docker images
REPOSITORY   TAG      IMAGE             ID      CREATED           SIZE
busybox      latest   c51f86c28340      4       weeks ago         1.109 MB
```

Step3: Execute your CMD/Task/Code/App/Process with DockerImage

```
# docker run busybox  echo "Hello World"
```

Step2: Check Running Docker Containers (–a for all)

```
# docker ps
```

How to Jump in a Docker Container

Step1: PULL Docker Image from Docker HUB

```
# docker run --name lc1 -itd  busybox  sleep 1d
```

Step2: Check Running Docker Containers (-a for all)

```
# docker ps
CONTAINERID   IMAGE      COMMAND              CREATED          STATUS         PORTS    NAMES
558d15ef2761  busybox    "sleep 1d"           3 seconds ago    Up 2 seconds             lc1
```

Step3: Use exec with containerID or ContainerName including supported Shell

```
docker exec -it lc1 sh
```

How to remove Docker Container

Step1: Check Running Containers (-a for all)

```
# docker ps
CONTAINERID   IMAGE      COMMAND         CREATED          STATUS        PORTS   NAMES
558d15ef2761  busybox    "sleep 1d"      3 seconds ago    Up 2 seconds          lc1
```

REMOVE Container with containerID or ContainerName (-f for forcefully)

```
# docker rm lc1
```

STOP Container - Use rm with containerID or ContainerName

```
# docker stop lc1
```

START Container - Use rm with containerID or ContainerName

```
# docker start lc1
```

RESTART Container - Use rm with containerID or ContainerName

```
# docker restart lc1
```

How to Run Nginx Container

Step1: PULL Docker Image from Docker HUB

```
# docker pull nginx:latest
```

Step2: Check your Docker Image

```
# docker images
REPOSITORY   TAG      IMAGE           ID      CREATED             SIZE
nginx        latest   ad80nkldf0      4       weeks ago          142 MB
```

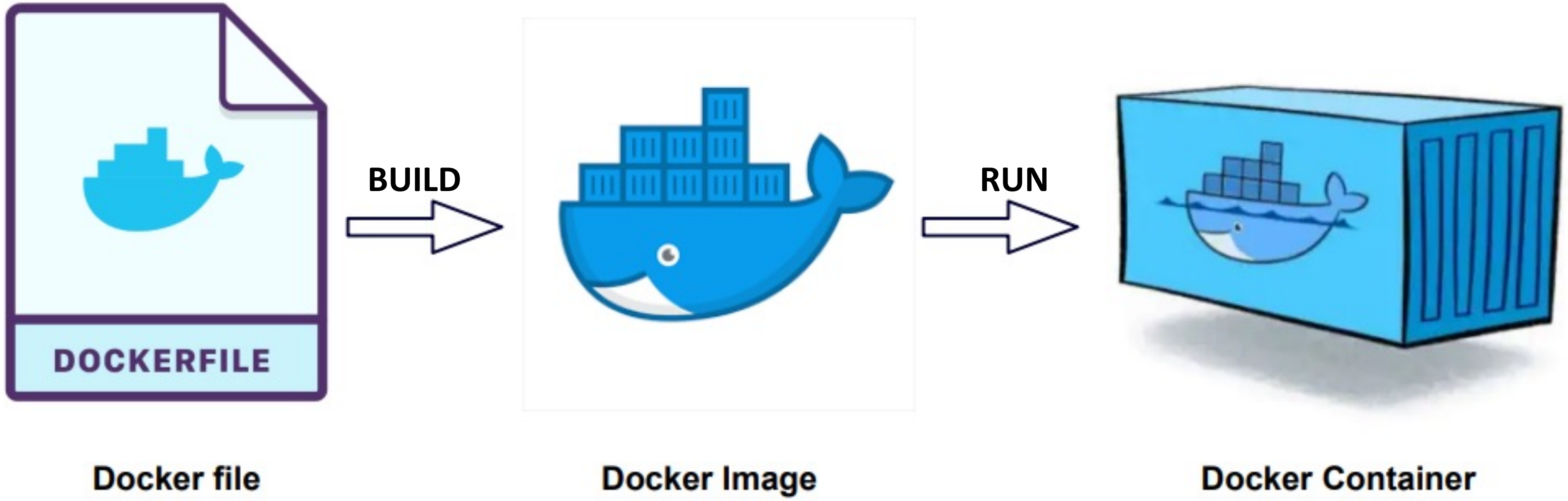
Step3: Use –p to port–forward trafick from your local machine to Container

```
# docker run -itd --name c1 -p 8080:80 nginx
```

Step4: Check WebPage on browser

```
http://localhost:8080
```

How to Create Docker Image



Dockerfile

- **FROM** – Select the base image to build the new image.
- **RUN** - Triggers command while we build the docker image.
- **CMD** - Triggers command while we launch the created docker image.
- **ENTRYPOINT** - Is also closely related to CMD and can modify the way a container starts an image.
- COPY - only supports the basic copying of local files into the container,
- ADD - It has some features like auto tar extraction and remote URL support
- EXPOSE – Define which Container ports to expose
- USER – Define the default User all commands will be run
- WORKDIR – Define the default working directory
- ENV – Set/modify the environment variables
- VOLUME – Creates a mount point within the Container linking it back to file systems accessible by the Docker Host

Lets Create Docker Image

Step1: Create your JAVA application docker Image with entrypoint using Dockerfile

Ref link https://github.com/amitganvir23/kubernetes-minishift-openshift/tree/master/kubernetes/hello-java-app-run/helo-app-docker-image

```
# cat Dockerfile
FROM anapsix/alpine-java:8_jdk_nashorn
MAINTAINER AMIT GANVIR
RUN mkdir /myapp
COPY entrypoint.sh /
COPY hello.jar /myapp/app.jar
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

Step2: Check content of entryopoint.sh

```
#!/bin/sh
set -x
java -jar /myapp/app.jar
```

OR

Create your JAVA application docker Image with CMD using Dockerfile

```
# cat Dockerfile
FROM anapsix/alpine-java:8_jdk_nashorn
MAINTAINER AMIT GANVIR
RUN mkdir /myapp
COPY hello.jar /myapp/app.jar
CMD ["/java”,  “-jar”,  “/myapp/app.jar"]
```

Ref link https://github.com/amitganvir23/kubernetes-minishift-openshift/tree/master/kubernetes/hello-java-app-run/helo-app-docker-image

Step3: Build Docker image

```
# docker build . –t “amitganvir6/java-app1:v1”
```

Step4: Test your application by running container

```
# docker run -itd --name c3 -p 8081:8080 amitganvir6/java-app1:v1
```

Check the pod

```
# docker ps
```

Check the pod logs

```
# docker logs -f c3
```

Check Your App WebPage on browser

```
http://localhost:8081
```

Lets make Docker Image tag

```
Step1: docker tag <Image Name/ID> <New IAMGE TAG repo/app:tag>
```

```
# docker tag amitganvir6/java-app1:v1 amitganvir6/java-app1:v2
```

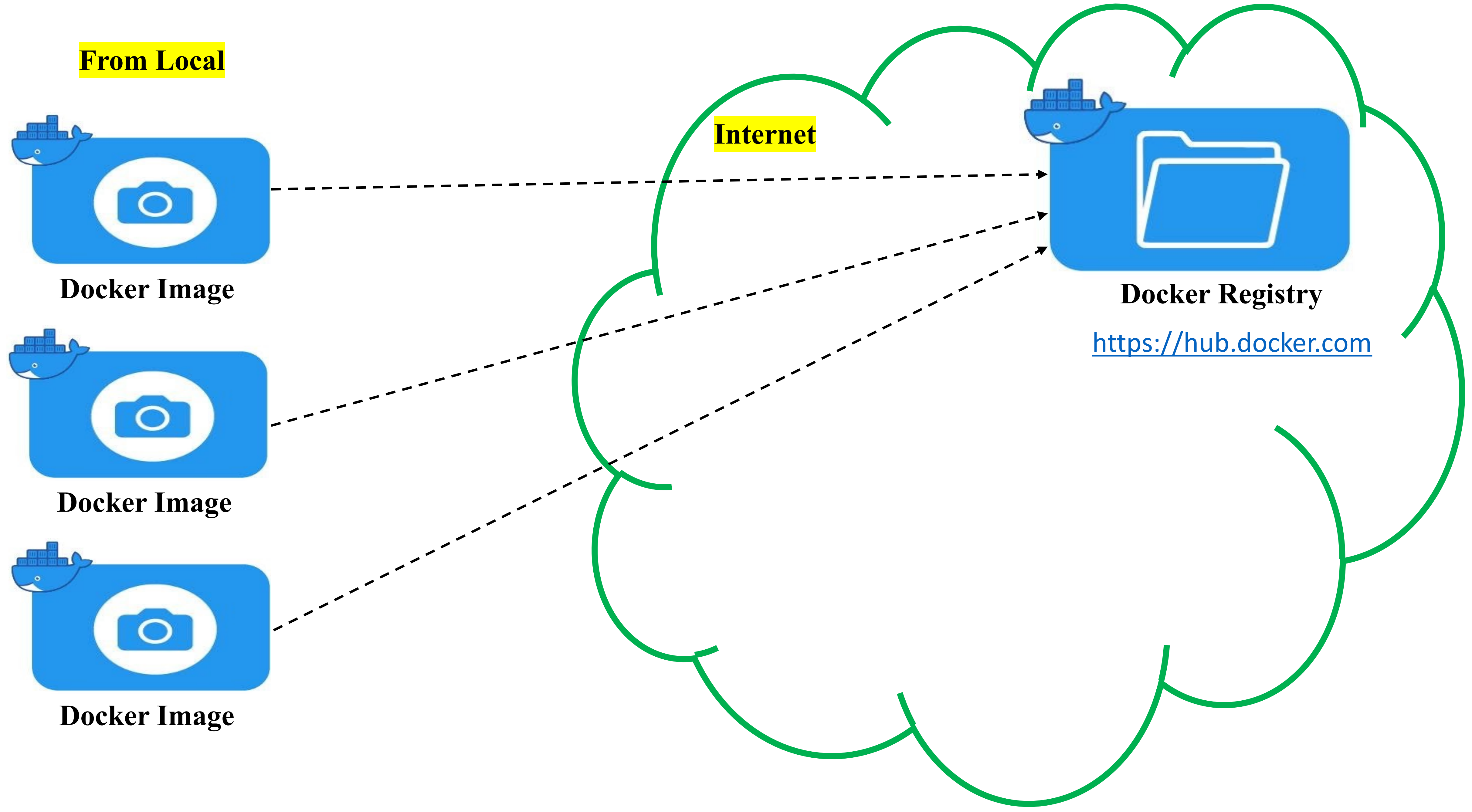
Docker Registries

- A Docker *registry* stores Docker images

- Docker Hub is a public registry that anyone can use (https://hub.docker.com/)

- You can even run your own private registry

- When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry


From Local
Docker Image
Docker Image
Docker Image
Internet
Docker Registry
https://hub.docker.com

How to push Docker Image in Docker Registry

Step1: Create an account on Dockerhub registry – https://hub.docker.com

Step2: Create your access token – https://hub.docker.com

Step3: Login with credentials on your local machine. Don’t use your login password instead use token

```
# docker login -u amitganvir6
```

OR

```
# docker login -u amitganvir6 –h https://hub.docker.com/repositories/amitganvir6
```

Step4: Now push your image to Dockerhub Registry

```
# docker push amitganvir6/java-app1:v1
```

Step5: Now check your Image in dockerhub registry– https://hub.docker.com

Lets Run a Mlflow Container and integrated with DB

```
docker pull ghcr.io/mlflow/mlflow:v2.1.1
docker pull bitnami/mariadb:latest

export DB_PASSWORD=1234
docker run -p 3306:3306 --name mariadb -e MARIADB_ROOT_PASSWORD=$DB_PASSWORD -itd
bitnami/mariadb:latest



docker run -p 5100:5100 -v /var/mm/:/var/mm --name mlflow -itd ghcr.io/mlflow/mlflow:v2.1.1

DB_IP=$(docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mariadb)

docker run -p 5100:5100 -v /var/mm/:/var/mm --name mlflow -itd ghcr.io/mlflow/mlflow:v2.1.1 sh -c
"pip install pymysql;mlflow server --host 0.0.0.0 -p 5100 --backend-store-uri
mysql+pymysql://root:${DB_PASSWORD}@${DB_IP}:3306/test --default-artifact-root
file:///var/mm/mlflow/artifacts"
```

http://localhost:5100

Docker command list

docker run – Runs a command in a new container
docker start – Starts one or more stopped containers
docker stop – Stops one or more running containers
docker build – Builds an image form a Docker file
docker login - To login on Container registry
docker pull – Pulls an image or a repository from a registry
docker push – Pushes an image or a repository to a registry
docker exec – Runs a command in a run-time container (-it)
docker ps - To list containers (-a)
docker images - To list all the local images
docker rm - to remove the container
docker rmi - To remove images
docker search - To search images on the registry
docker kill – To kill pods
docker export – Exports a container’s filesystem as a tar archive
docker search – Searches the Docker Hub for images
docker attach – Attaches to a running container
docker commit – Creates a new image from a container’s changes

Data Availabe in Github Repository

https://github.com/amitganvir23/devops-session

Thank You!

Present By **Amit Ganvir**
amitganvir6@gmail.com