CLOUD ENGINEERING

Containers and Virtualization

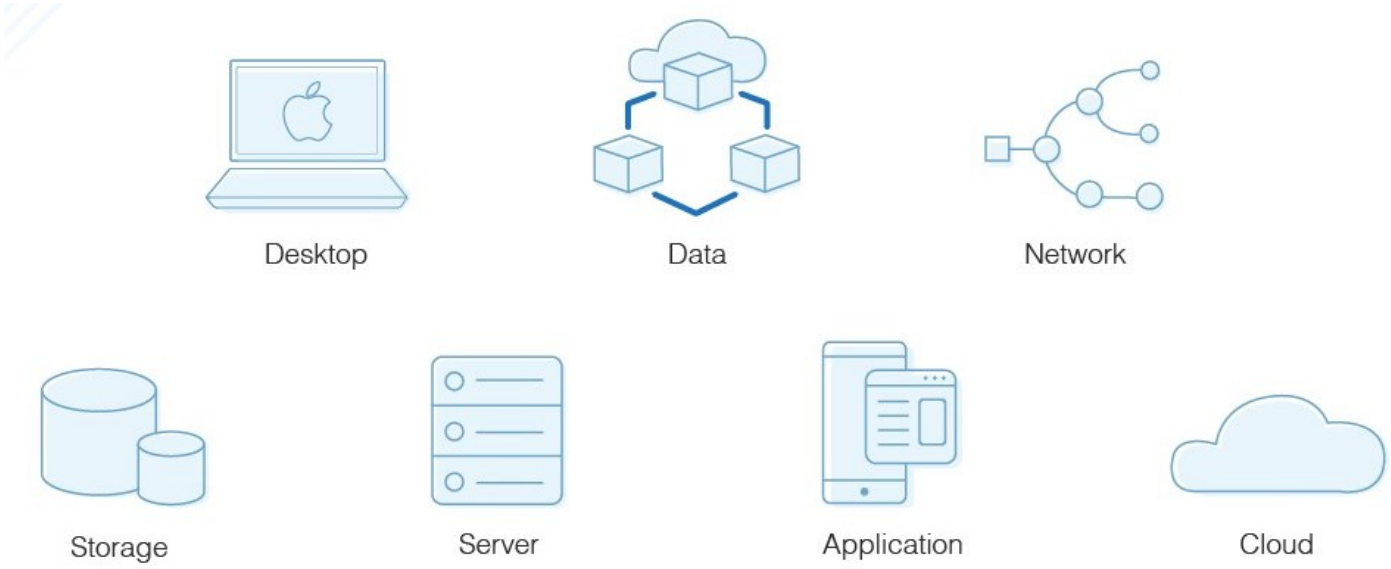
Ashish Pujari

Lecture Outline

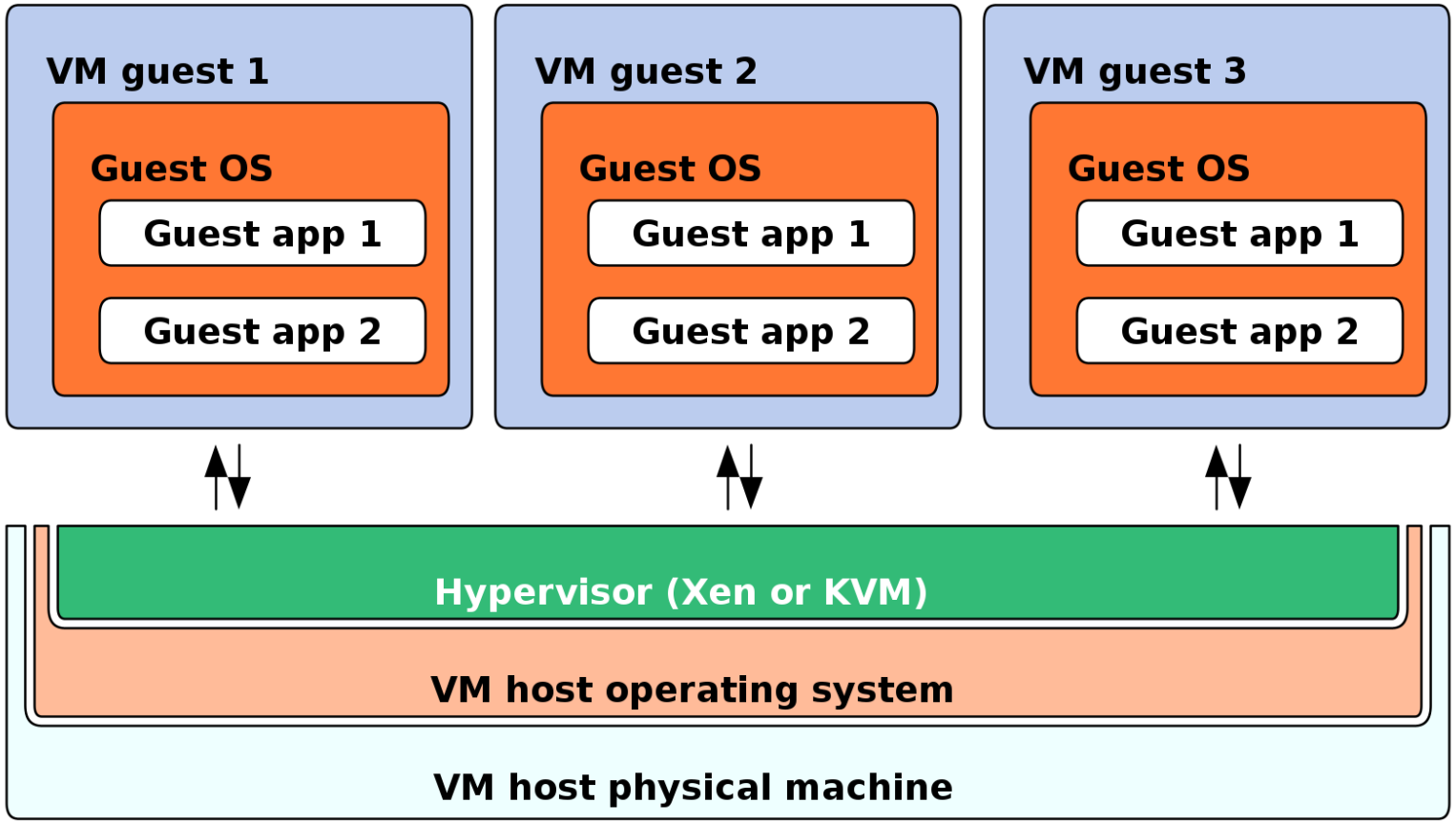
- Containers and Virtual Machines
- Docker Exercises

Virtualization

- Virtualization is the creation of a virtual (rather than actual) version of something, such as an operating system (OS), a server, a storage device or network resources.
- Virtualization uses software that simulates hardware functionality to create a virtual system



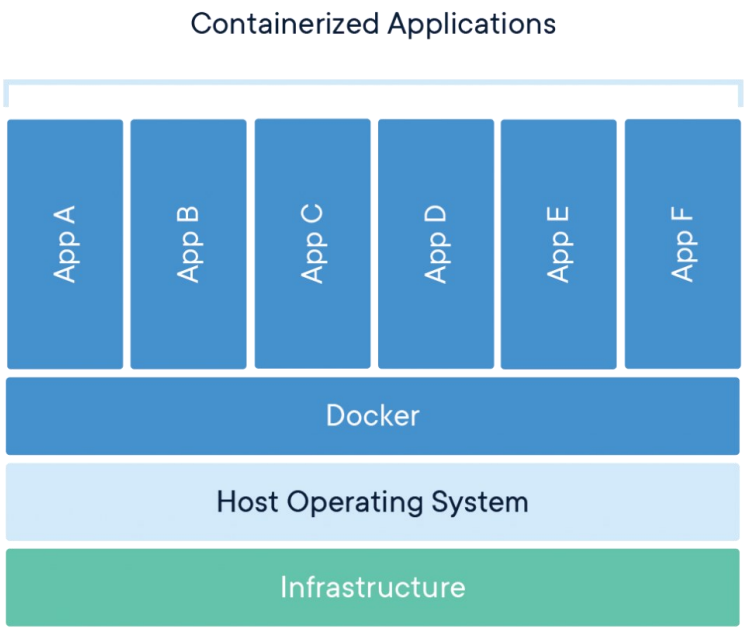
Virtual Machine (VM)



https://documentation.suse.com/smart/virtualization-cloud/html/concept-virtualization/index.html

Containers

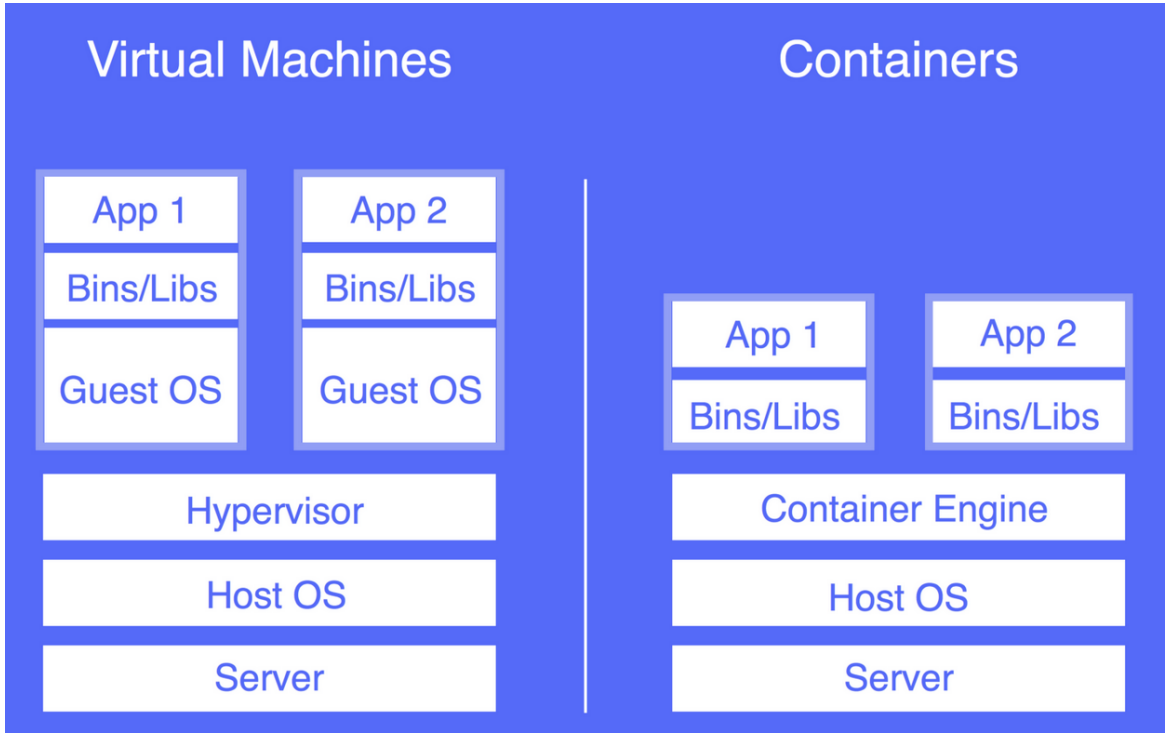
- Standard unit of software that packages up code and dependencies, so the applications run reliably from one computing environment to another
- E.g., Docker, LXD, OpenVZ, Rkt, Windows Server Containers, etc.



docker

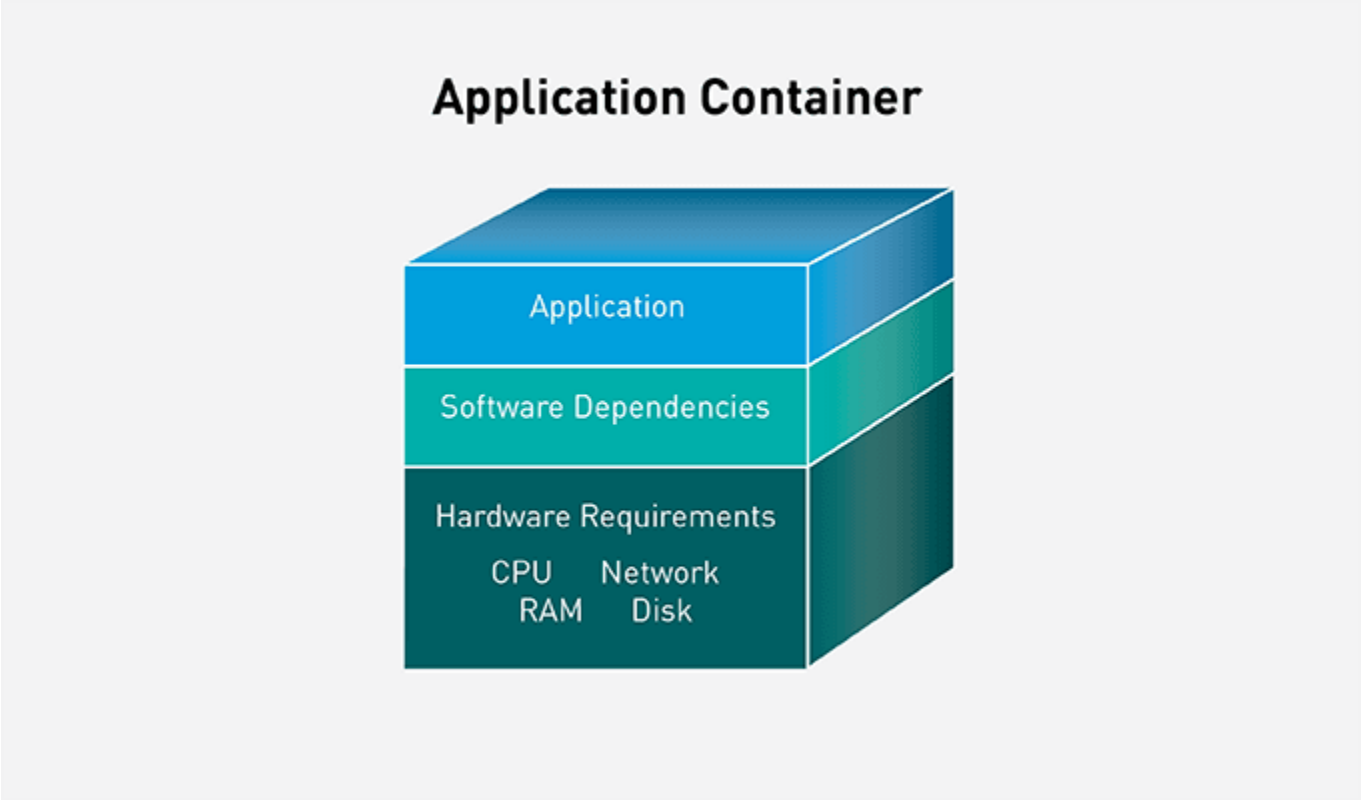
Virtual Machines Vs Containers

- Containers are a lightweight approach to virtualization that only provides the bare minimum that an application requires to run and function as intended



Container Contents

- Items usually bundled into a container include application, dependencies, libraries, binaries, configuration files

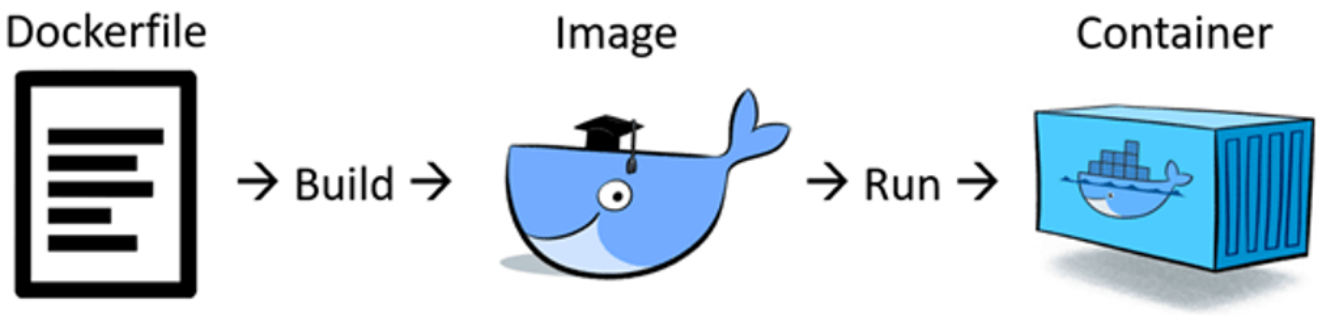


Advantages of using Containers

- Agility and Productivity
  - Decreases time needed for development, testing, and deployment of applications and services. E.g., microservices, functions, web applications, ML/AI pipelines.
- Rapid Scalability
  - Provisioning containers only take a few seconds or less, therefore, the data center can react quickly to a spike in user activity.
- Resource Utilization
  - A server can host significantly more containers than virtual machines.
  - Several operating system on one host, providing reduced hardware maintenance.
- Cost Effectiveness
  - Potentially help you to decrease operating cost and development cost (develop for one consistent runtime environment).

Building Docker Images

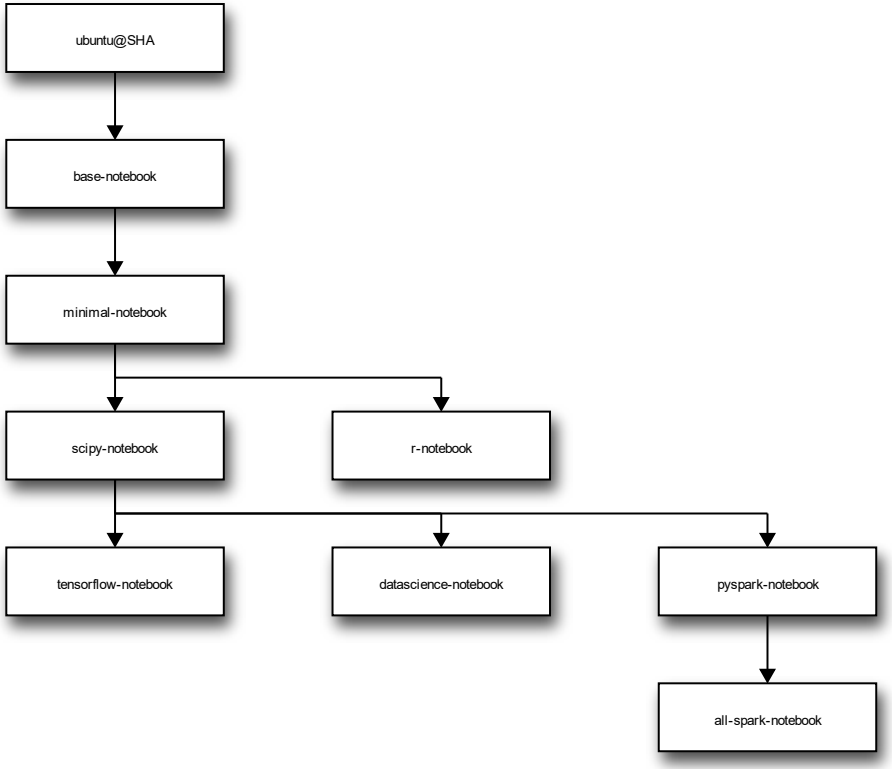
- Docker can build images automatically by reading the instructions from a Dockerfile - text document that contains all the commands a user could call on the command line to assemble an image



https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Docker: Composability

- Complex docker images are usually composed from preexisting base or parent images



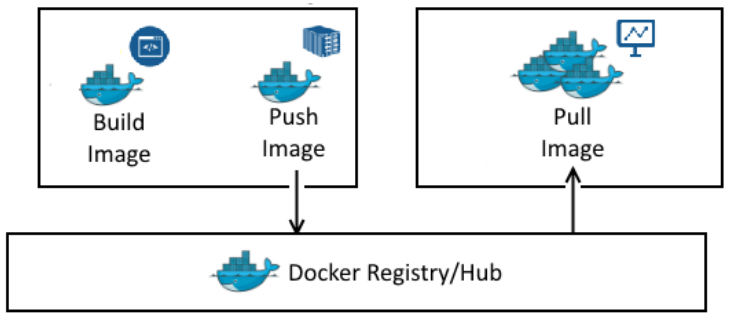
Depicts the build dependency tree of the core images

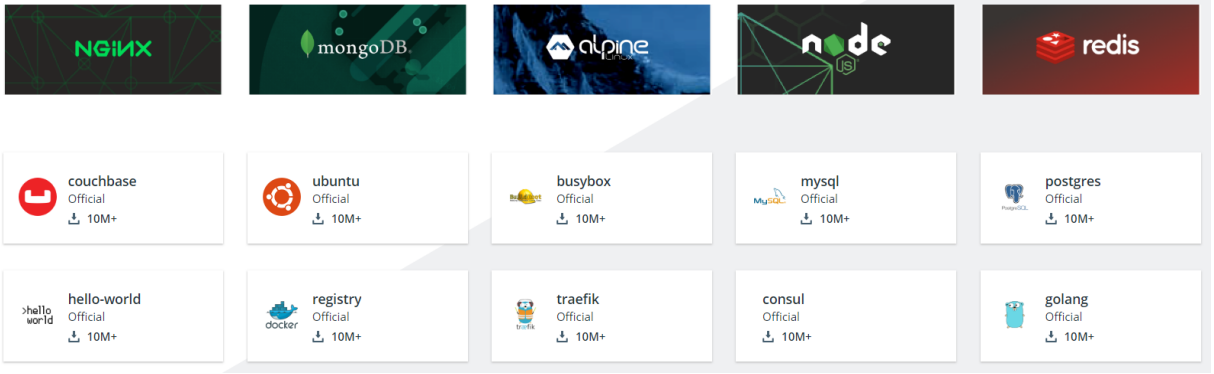
Container Registry

- Container Registry is a centralized storage for storage and management of container images.
- Examples:
  - Docker Hub
  - Amazon Elastic Container Registry (ECR)
  - Google Container Registry (GCR)

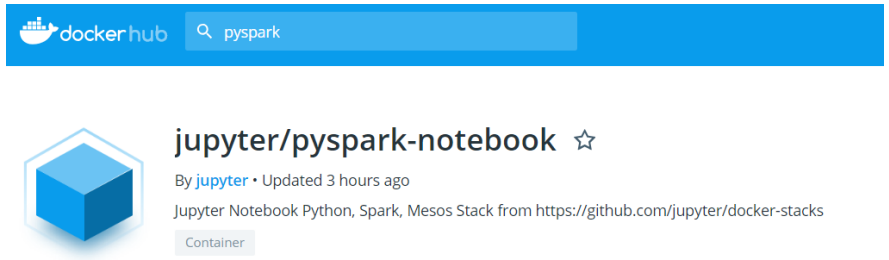
Docker Hub

- Browse through library of container images https://hub.docker.com/

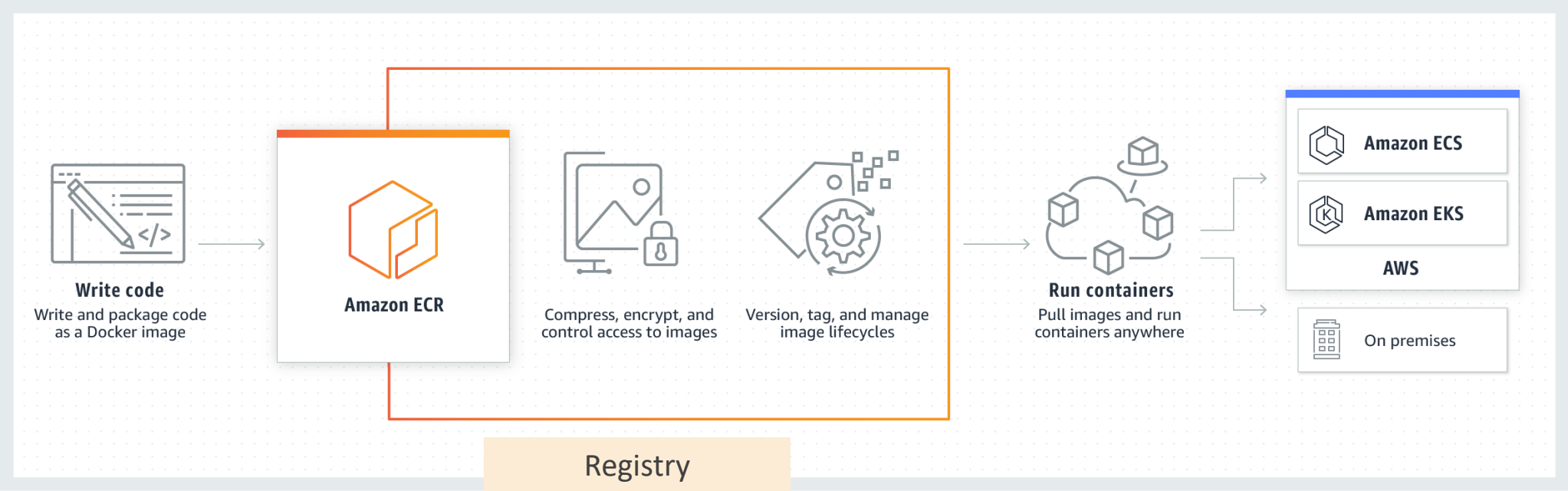




- Explore data science related images

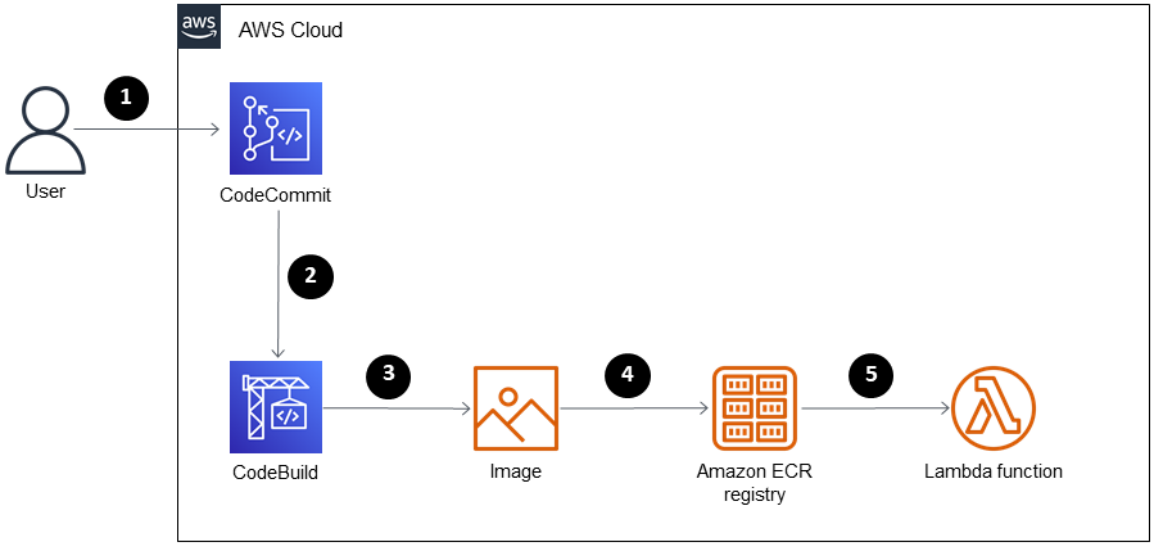


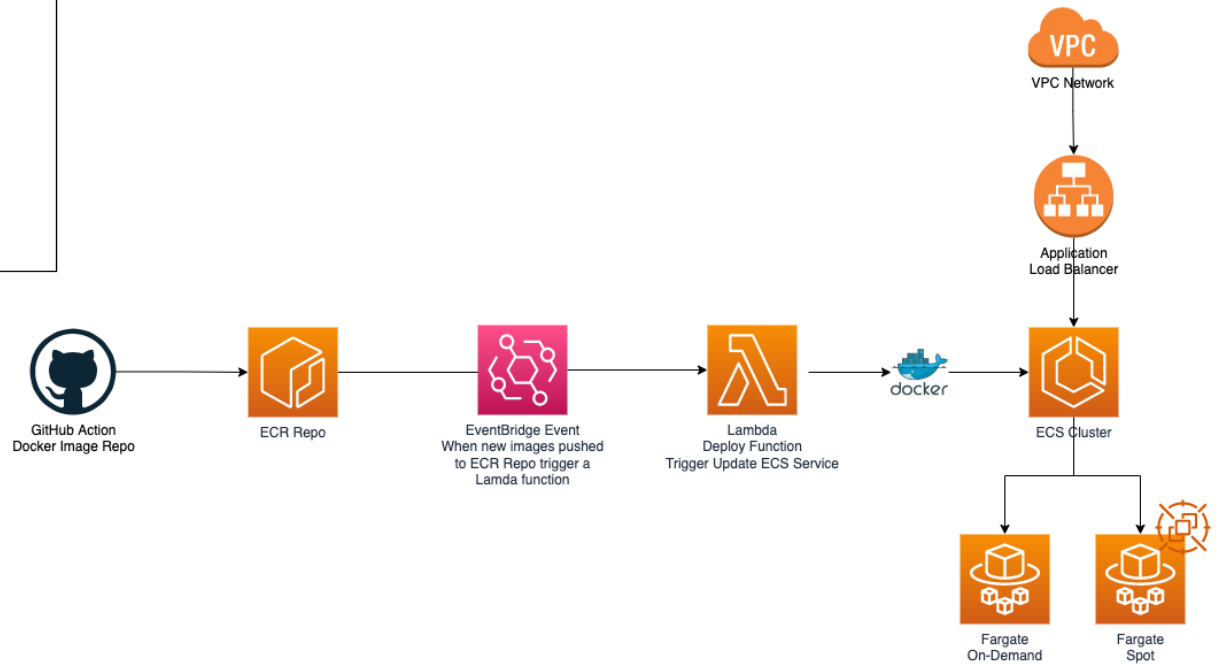
Amazon Elastic Container Registry (ECR)



Registry
Authorization
Repository
Repository policy
Image

ECR: Deploying Lambda Functions





Docker: Commands

```
#List all images
docker images

#List all containers and their status
docker ps -a

#Enter the container
docker exec -i -t <container name> bash

#Start a stopped container
docker start <container name>

#Stop a running container
docker stop <container name>
```

```
#Save container state
#you are encouraged to commit frequently
docker commit <container name> <image name>

#Copy file from local machine into
container root directory
docker cp <file name> <container name>:/
<file name>

#Delete all stopped containers
docker rm $(docker ps -q -f status=exited)

#Remove a docker image
docker rmi <image name>
```

https://docs.docker.com/engine/reference/commandline/docker/

Docker: Hello World

- Run simple Docker commands

```
#Download "Hello World" Docker image
docker pull hello-world

#Run the image
docker run hello-world
```

```
Bhaarat@DESKTOP-RV4I2EU MINGW64 /c/Program Files/Docker Toolbox
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest

Bhaarat@DESKTOP-RV4I2EU MINGW64 /c/Program Files/Docker Toolbox
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

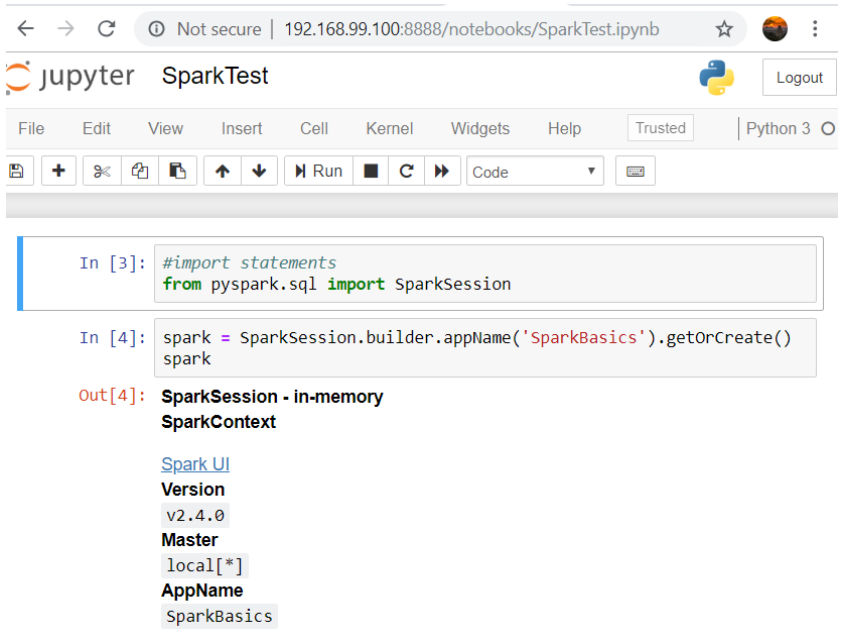
Docker: Spark Container

```
docker pull jupyter/pyspark-notebook
docker images
```

```
Bhaarat@DESKTOP-RV4I2EU MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
jupyter/pyspark-notebook  latest              419a7429a36e        3 hours ago         5.03GB
hello-world               latest              fce289e99eb9        2 months ago        1.84kB
hello-world               <none>              4ab4c602aa5e        6 months ago        1.84kB
```

```
docker run --name "sparknb" -it -p 8888:8888 jupyter/pyspark-notebook
```

To view the notebook running inside the container open a new browser and navigate to:
http://192.168.99.100:8888/tree?
http://127.0.0.1:8888/tree?



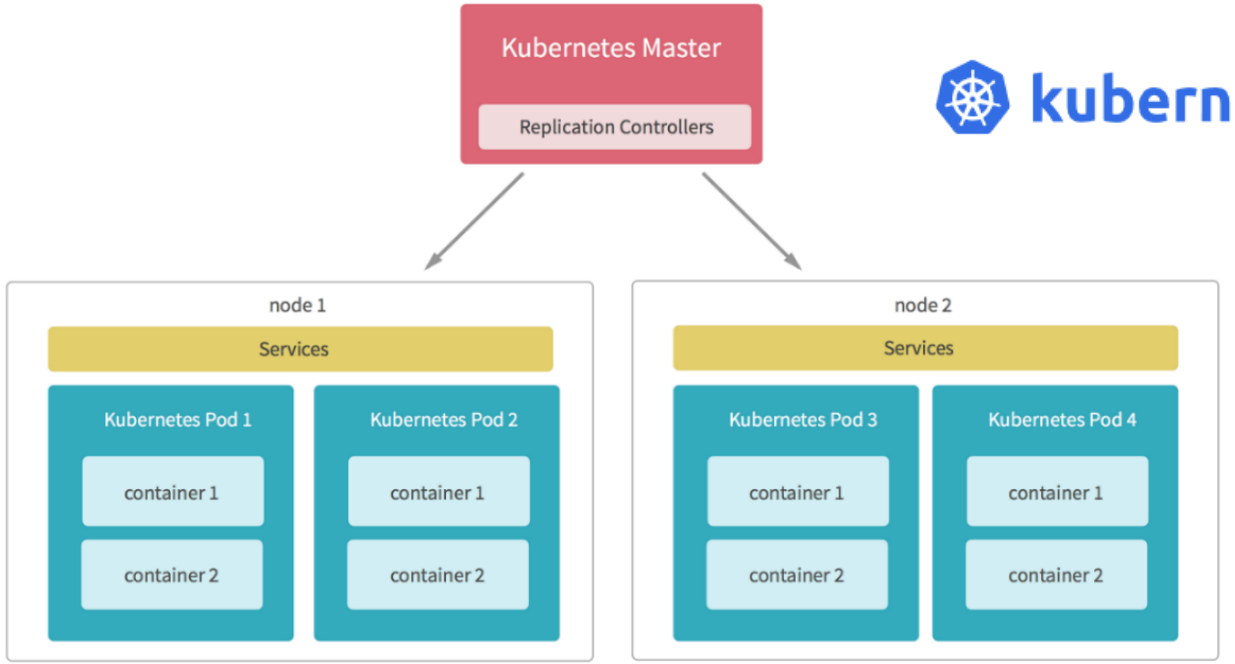
Container Orchestration

- Tools used for automating deployment, scaling, and management of containerized applications E.g., Kubernetes, Docker Swarm
- Features
  - Service discovery, Load balancing
  - Automated rollouts and rollbacks
  - Horizontal Scaling
  - Batch execution
  - Configuration Management

https://kubernetes.io/

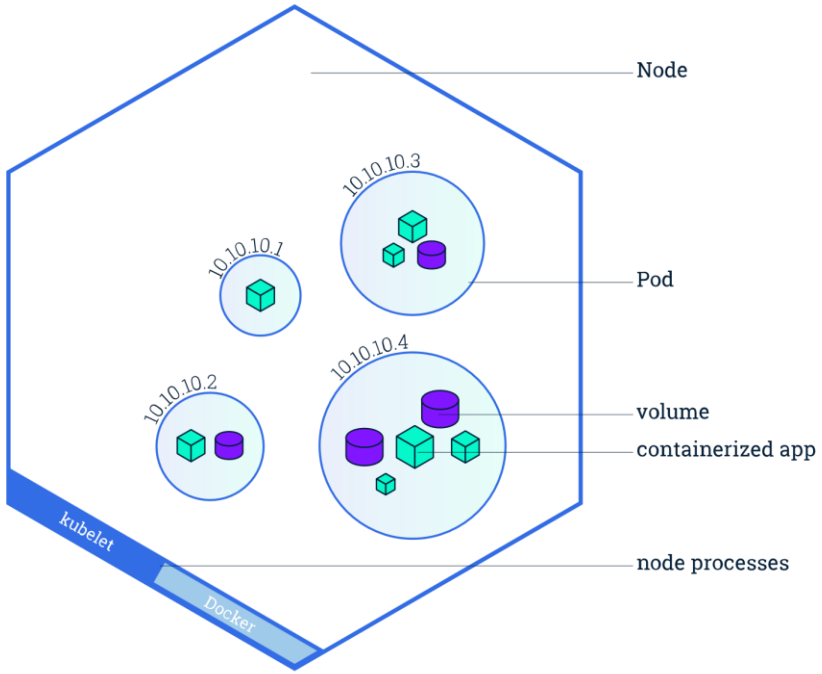
Kubernetes (K8s)

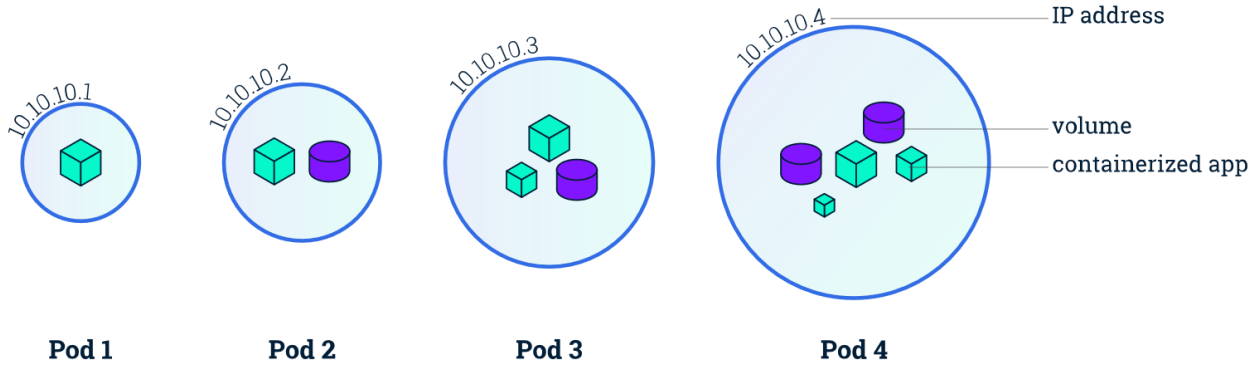
- Open-source container orchestration platform that provides capabilities for managing and automating the deployment, scaling, and operation of containers across clusters of hosts.



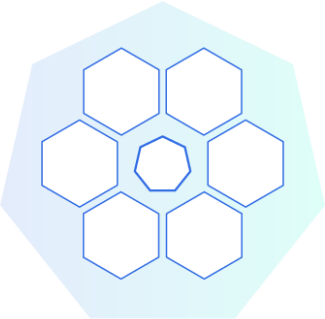
Kubernetes: Pods

- Pods are logical units which can run one or more containers together. Pods provide a way to group related containers and share resources, such as networks and storage.

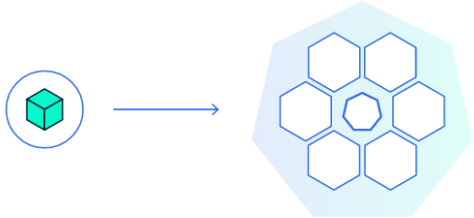




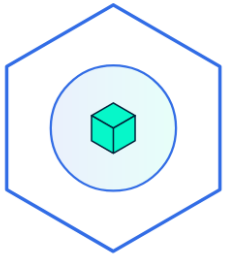
Kubernetes: Modules



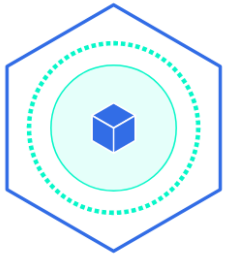
1. Create a Kubernetes cluster



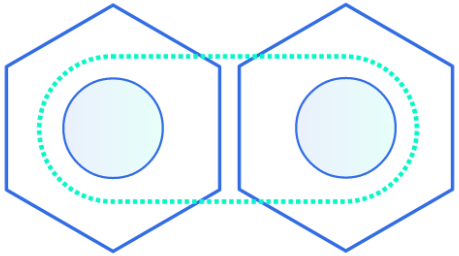
2. Deploy an app



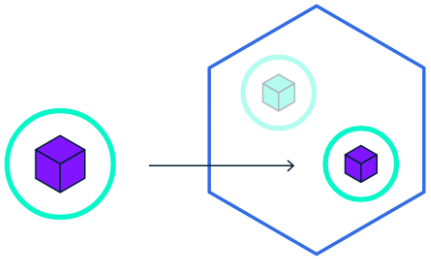
3. Explore your app



4. Expose your app publicly



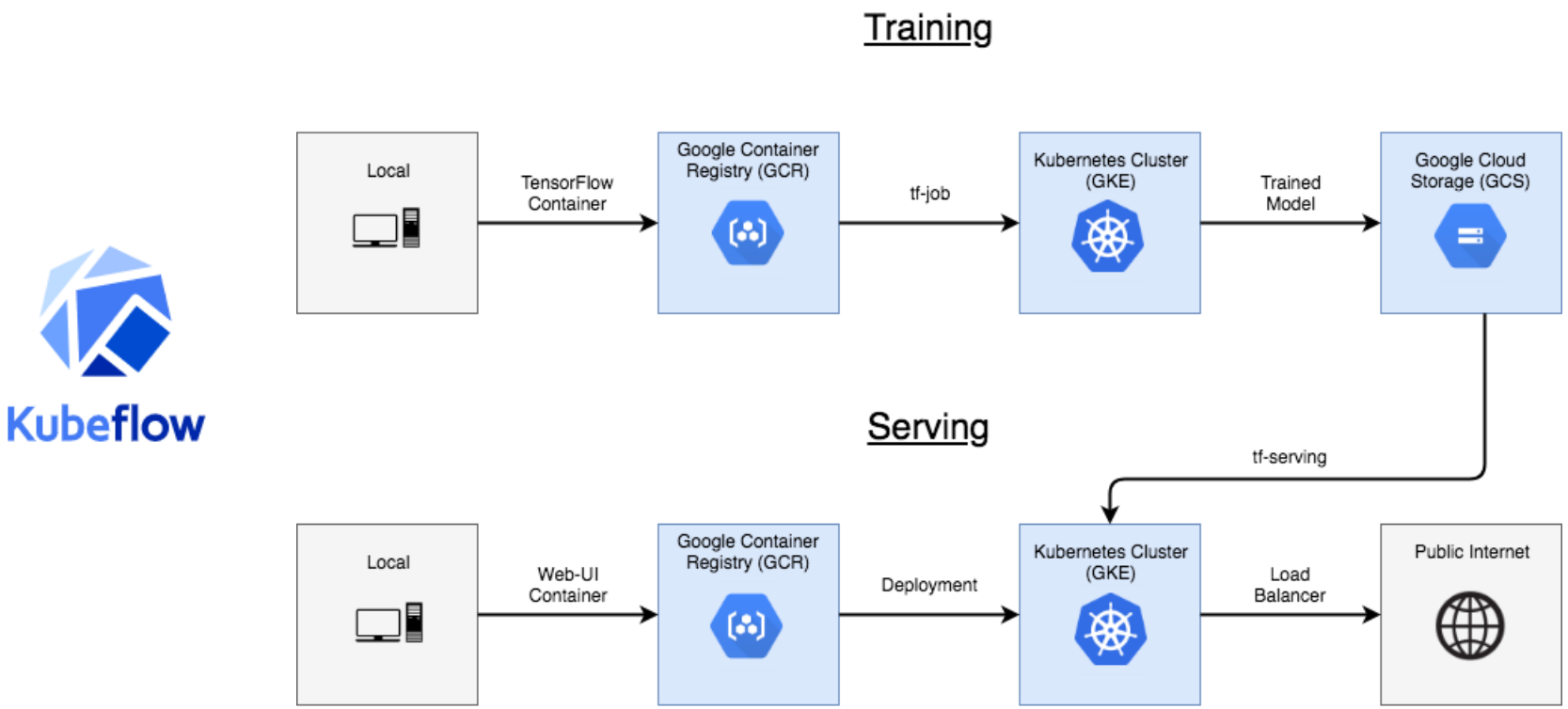
5. Scale up your app



6. Update your app

Kubernetes: Kubeflow

- Open-source project which aims to make running ML workloads on Kubernetes simple, portable and scalable



https://codelabs.developers.google.com/codelabs/kubeflow-introduction/index.html#0

Kubernetes: Kubeflow