DevOps with Docker & Kubernetes (Deployment)

Learning Objective

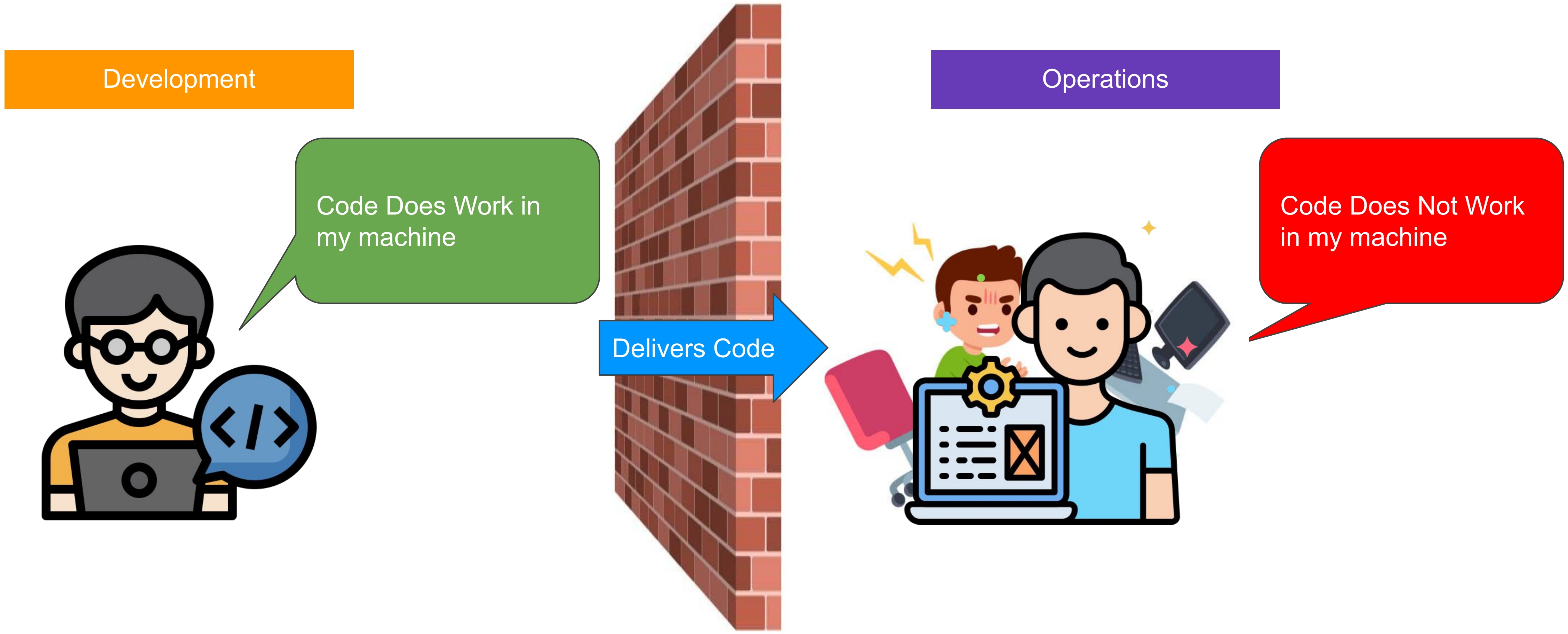
Upon completing this session, you will be able to:

- What is DevOps ?
- Why DevOps ?
- DevOps Principles
- CI / CD Pipelines
- Docker
- Kubernetes

The word “DevOps” is the blend of two terms “Development” and “Operations”.

What is DevOps?

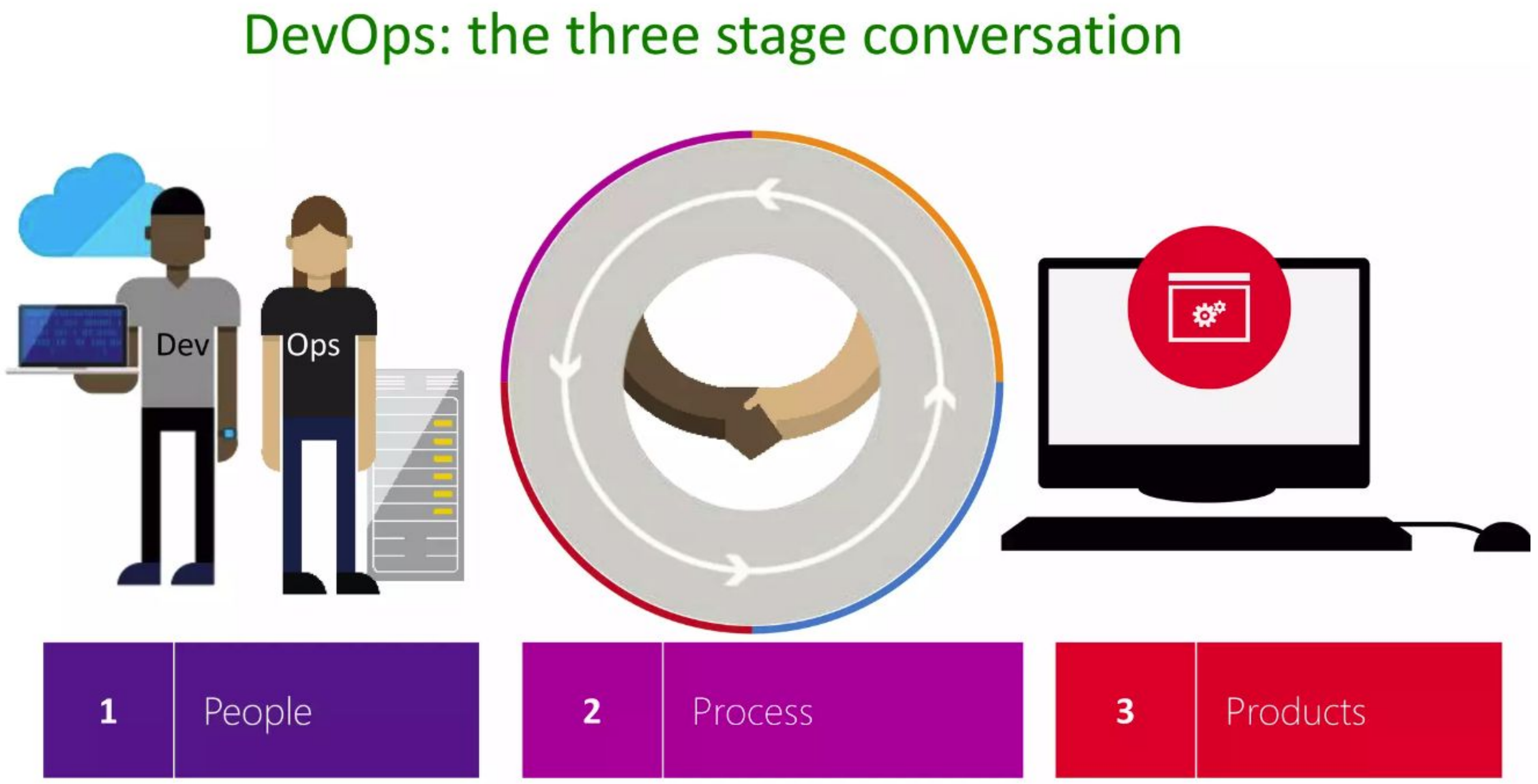
The Problem


Development
Code Does Work in my machine
Delivers Code
Operations
Code Does Not Work in my machine

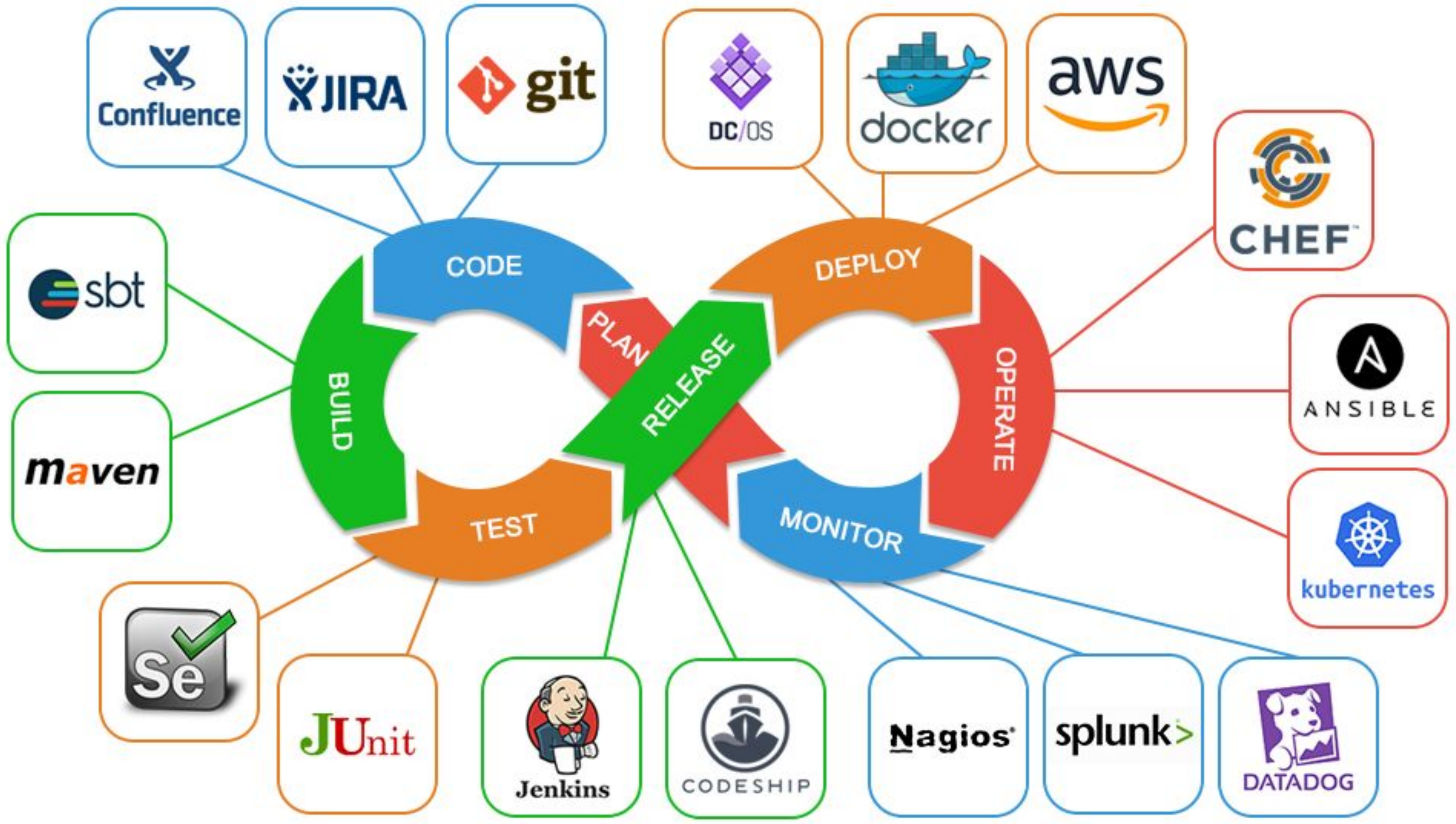
DevOps

- The word “DevOps” is the coalescence of two terms “Development” and “Operations”.
- It is a software development methodology that focuses on communication, collaboration & integration between developers teams and operations team (IT Team).
- It enables rapid evolution of products or services.
- It reduces risks, improve quality across portfolio and thus reduce cost.

DevOps Process



Techs used during DevOps



Principles of DevOps

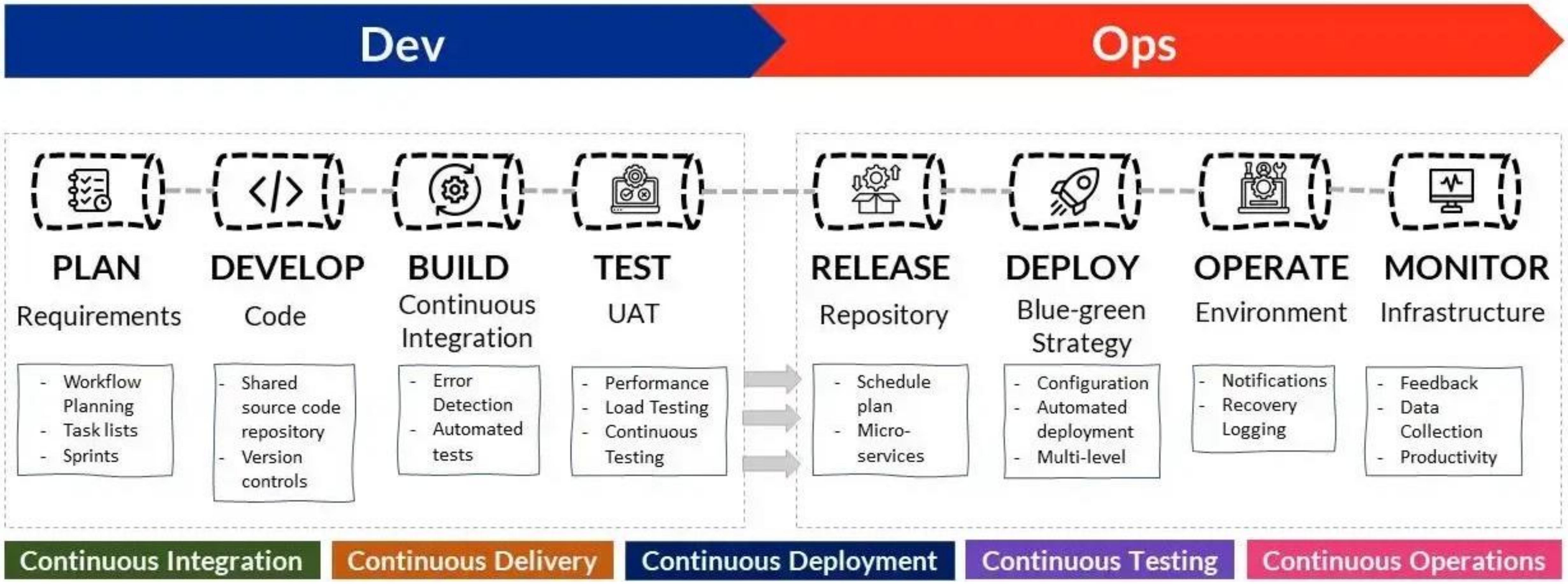
- Develop & test in an environment similar to production.
- Deploy build frequently.
- Validate Operation quality continuously.

What is CI/CD Pipeline?

CI / CD Pipeline

- A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software.
- They are a practice focused on improving software delivery throughout the software development life cycle via automation.
- We can develop higher quality code faster by automating CI/CD throughout development, testing, production, and monitoring phases of the software development lifecycle

CI / CD Process

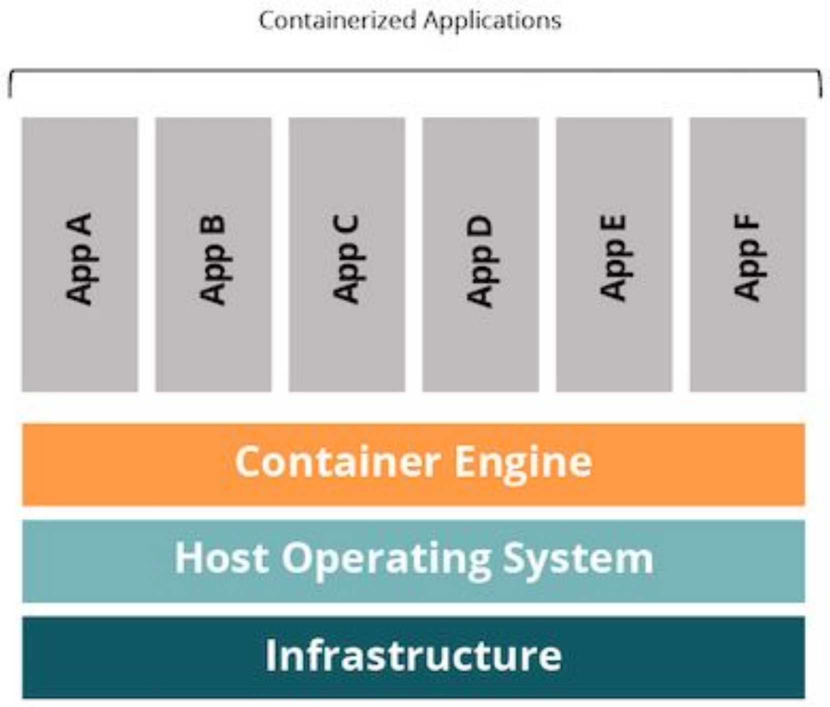


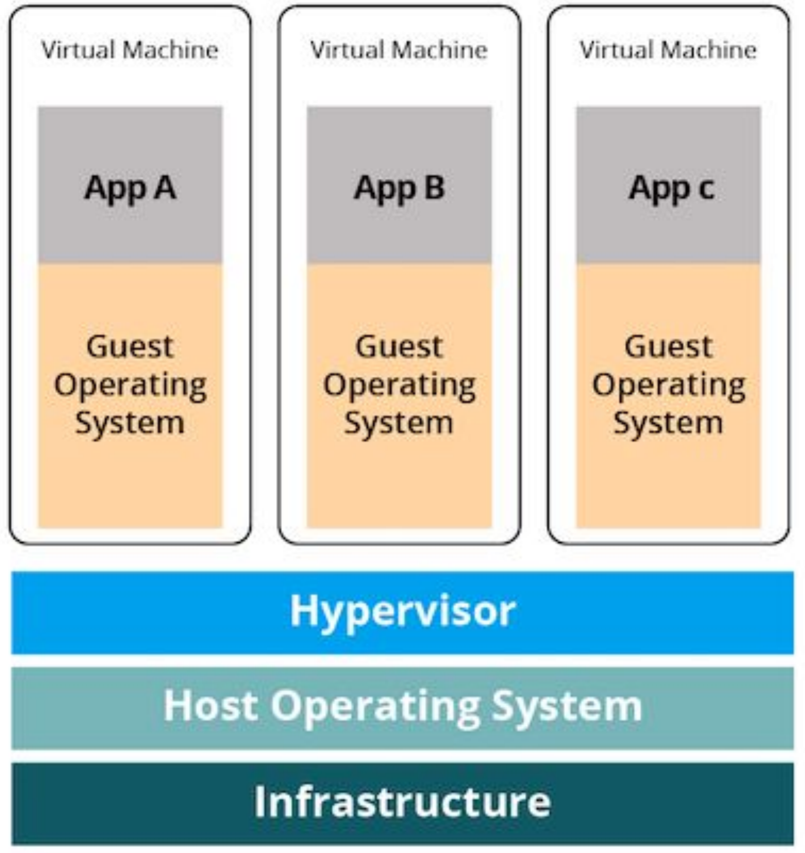
What is Docker ?

What is Docker ?

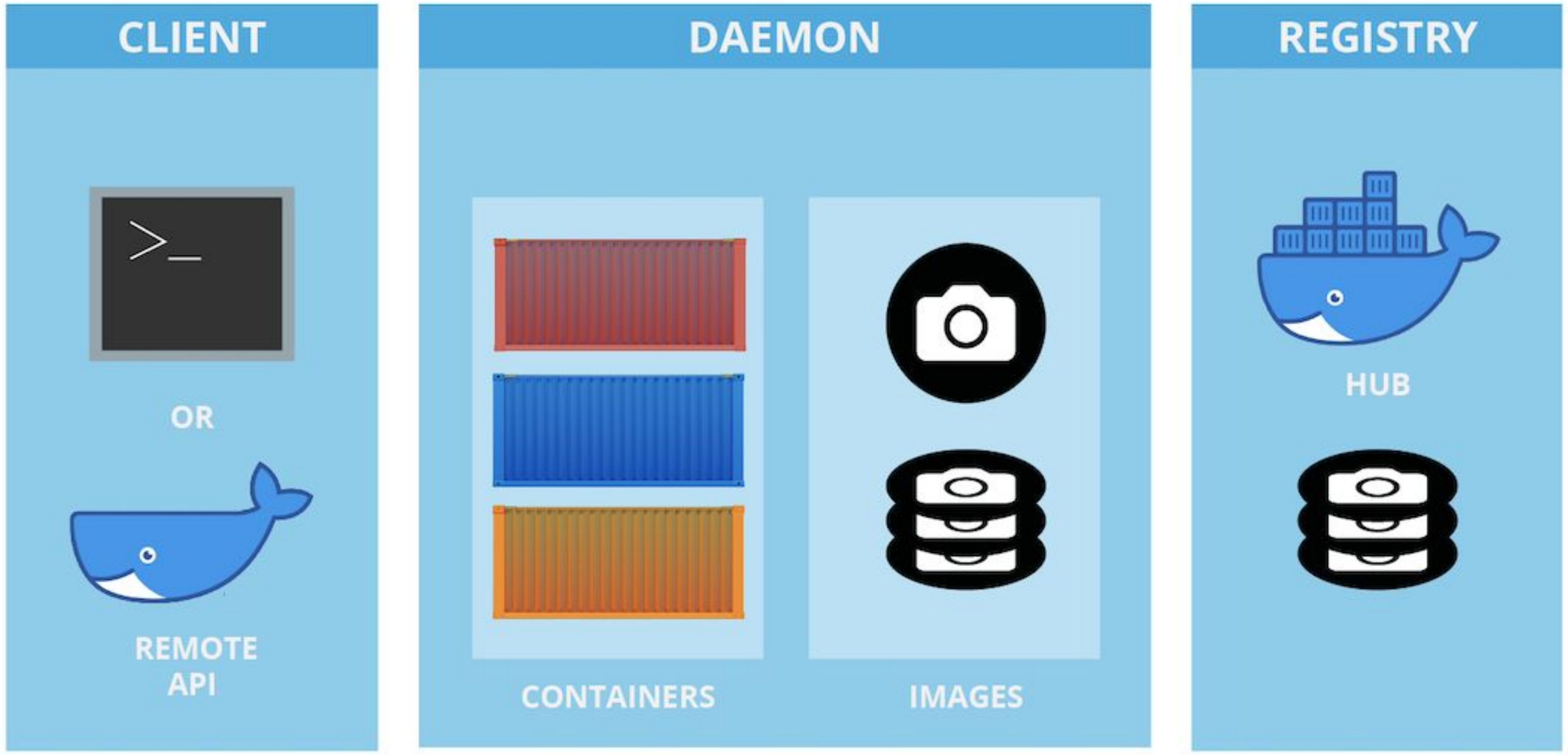
- Docker is a containerization platform.
- It is free and open source.
- Is a software platform that allows you to build, test, and deploy applications quickly, packaging software into standardized units called containers.
- Containerization enables "write once, run anywhere" programs.
- Containers are standardized executable components that combine application source code with the OS libraries and dependencies needed to run that code in any environment.
- Containers are "lightweight"ed, implying that they share the machine's OS kernel and eliminating the overhead of associating an OS with each application.

Docker vs Virtual machine

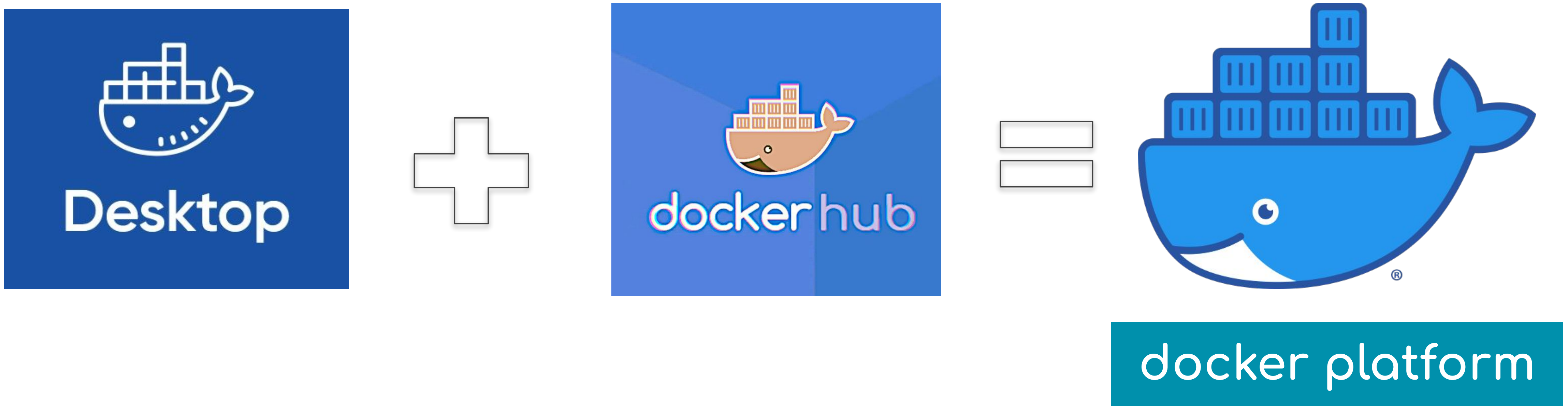




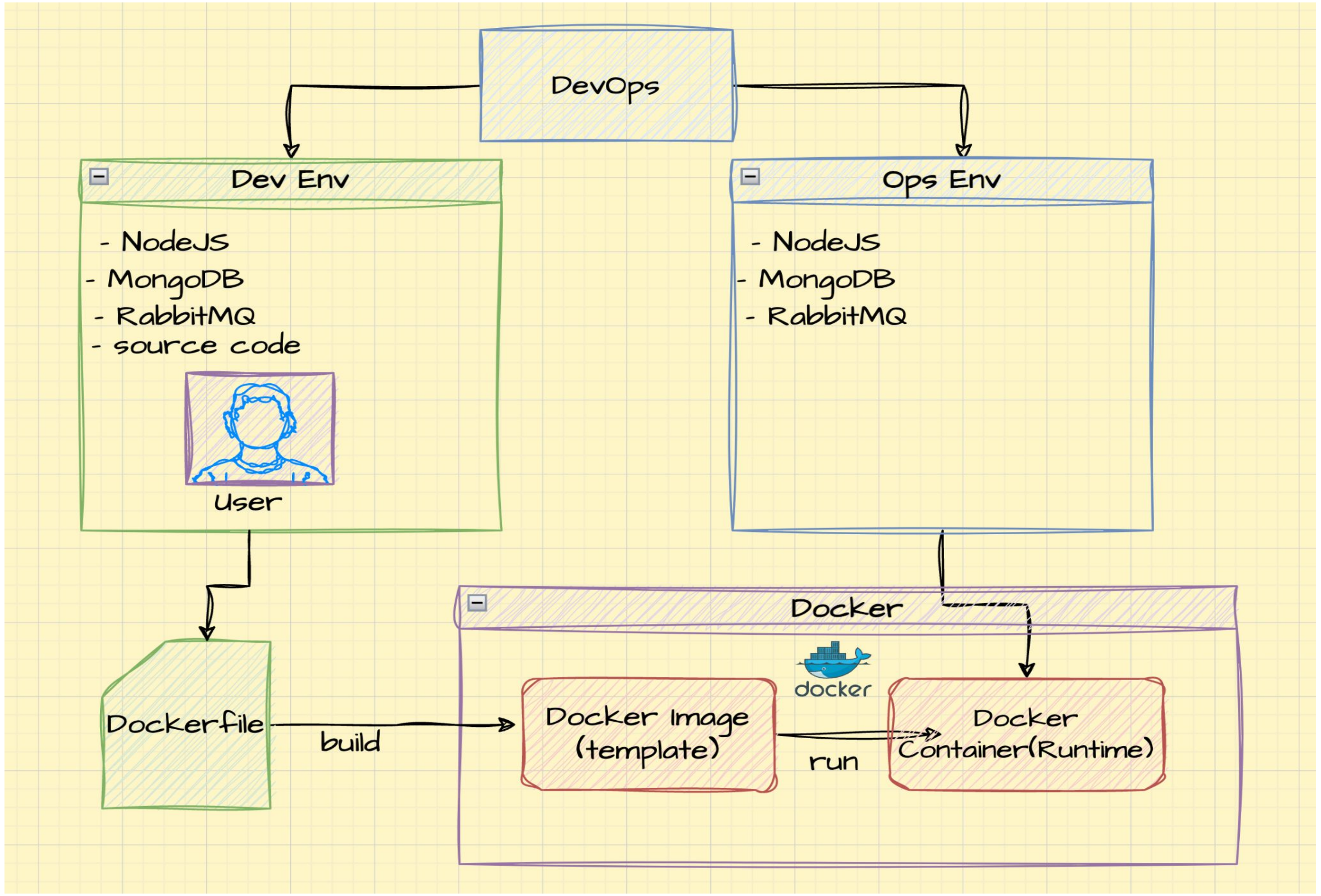
Docker Architecture



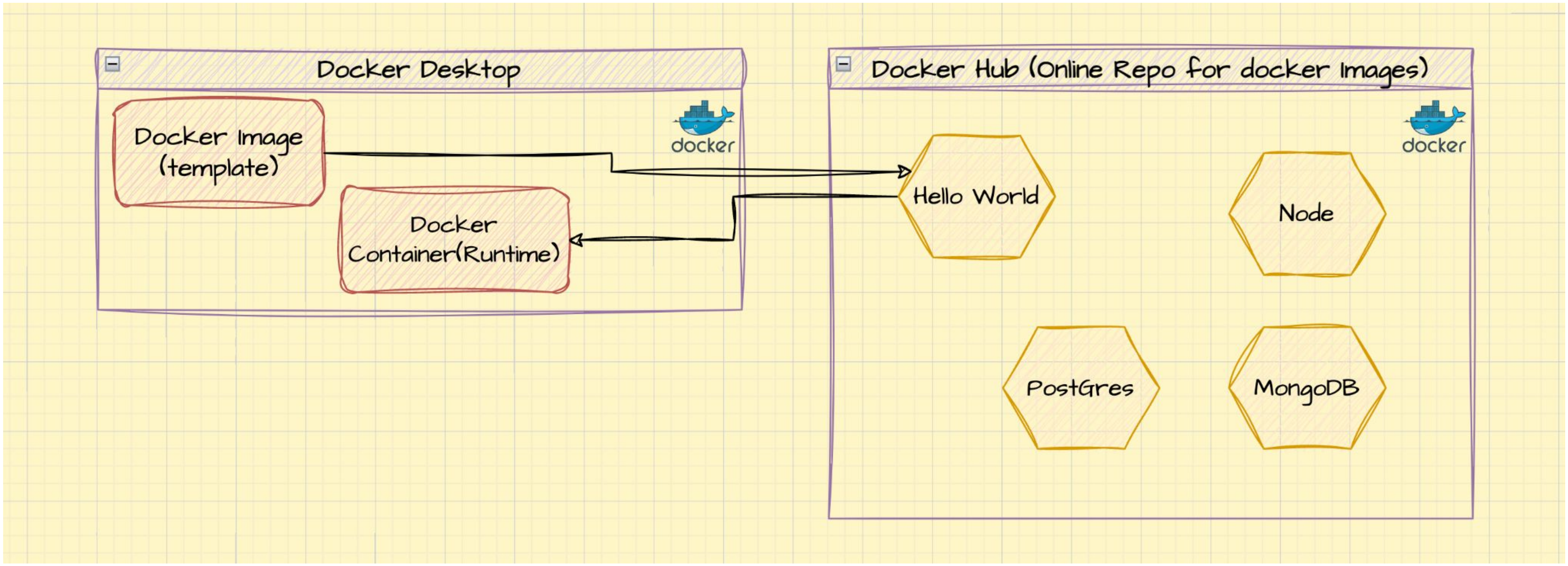
Docker Platform



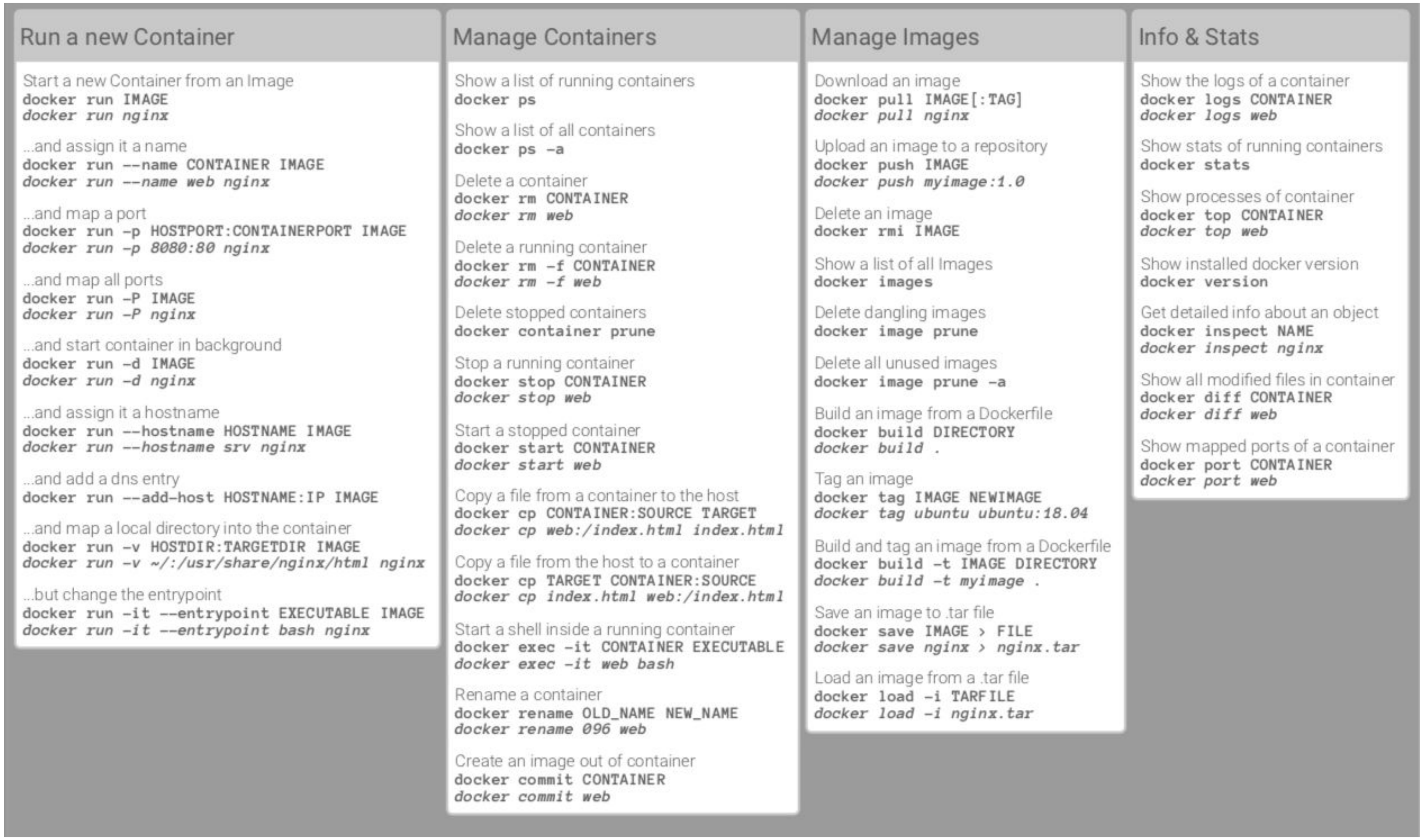
Docker Workflow



Pulling & Running Docker Image



Docker Commands - Cheat Sheet for Docker CLI



Run a new Container

Start a new Container from an Image
```
docker run IMAGE
docker run nginx
```
...and assign it a name
```
docker run --name CONTAINER IMAGE
docker run --name web nginx
```
...and map a port
```
docker run -p HOSTPORT:CONTAINERPORT IMAGE
docker run -p 8080:80 nginx
```
...and map all ports
```
docker run -P IMAGE
docker run -P nginx
```
...and start container in background
```
docker run -d IMAGE
docker run -d nginx
```
...and assign it a hostname
```
docker run --hostname HOSTNAME IMAGE
docker run --hostname srv nginx
```
...and add a dns entry
```
docker run --add-host HOSTNAME:IP IMAGE
```
...and map a local directory into the container
```
docker run -v HOSTDIR:TARGETDIR IMAGE
docker run -v ~/:/usr/share/nginx/html nginx
```
...but change the entrypoint
```
docker run -it --entrypoint EXECUTABLE IMAGE
docker run -it --entrypoint bash nginx
```

Manage Containers

Show a list of running containers
```
docker ps
```
Show a list of all containers
```
docker ps -a
```
Delete a container
```
docker rm CONTAINER
docker rm web
```
Delete a running container
```
docker rm -f CONTAINER
docker rm -f web
```
Delete stopped containers
```
docker container prune
```
Stop a running container
```
docker stop CONTAINER
docker stop web
```
Start a stopped container
```
docker start CONTAINER
docker start web
```
Copy a file from a container to the host
```
docker cp CONTAINER:SOURCE TARGET
docker cp web:/index.html index.html
```
Copy a file from the host to a container
```
docker cp TARGET CONTAINER:SOURCE
docker cp index.html web:/index.html
```
Start a shell inside a running container
```
docker exec -it CONTAINER EXECUTABLE
docker exec -it web bash
```
Rename a container
```
docker rename OLD_NAME NEW_NAME
docker rename 096 web
```
Create an image out of container
```
docker commit CONTAINER
docker commit web
```

Manage Images

Download an image
```
docker pull IMAGE[:TAG]
docker pull nginx
```
Upload an image to a repository
```
docker push IMAGE
docker push myimage:1.0
```
Delete an image
```
docker rmi IMAGE
```
Show a list of all Images
```
docker images
```
Delete dangling images
```
docker image prune
```
Delete all unused images
```
docker image prune -a
```
Build an image from a Dockerfile
```
docker build DIRECTORY
docker build .
```
Tag an image
```
docker tag IMAGE NEWIMAGE
docker tag ubuntu ubuntu:18.04
```
Build and tag an image from a Dockerfile
```
docker build -t IMAGE DIRECTORY
docker build -t myimage .
```
Save an image to .tar file
```
docker save IMAGE > FILE
docker save nginx > nginx.tar
```
Load an image from a .tar file
```
docker load -i TARFILE
docker load -i nginx.tar
```

Info & Stats

Show the logs of a container
```
docker logs CONTAINER
docker logs web
```
Show stats of running containers
```
docker stats
```
Show processes of container
```
docker top CONTAINER
docker top web
```
Show installed docker version
```
docker version
```
Get detailed info about an object
```
docker inspect NAME
docker inspect nginx
```
Show all modified files in container
```
docker diff CONTAINER
docker diff web
```
Show mapped ports of a container
```
docker port CONTAINER
docker port web
```

Activity:

What is Kubernetes ?

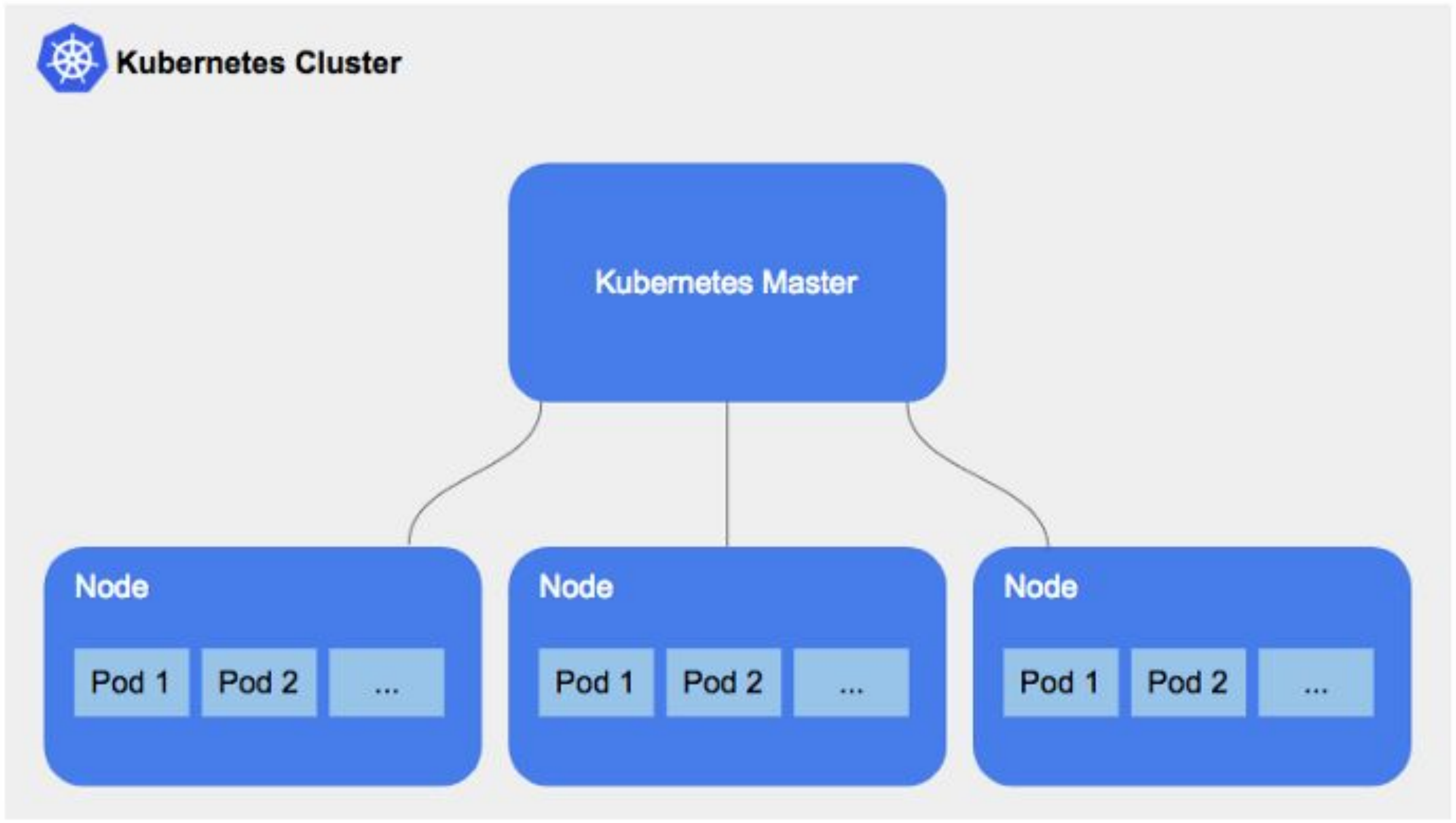
Kubernetes - K8s

- Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.
- It is a container orchestration system.
- Originally designed by Google, it is now maintained by the Cloud Native Computing Foundation.
- K8s, counting the eight letters between the "K" and the "s"

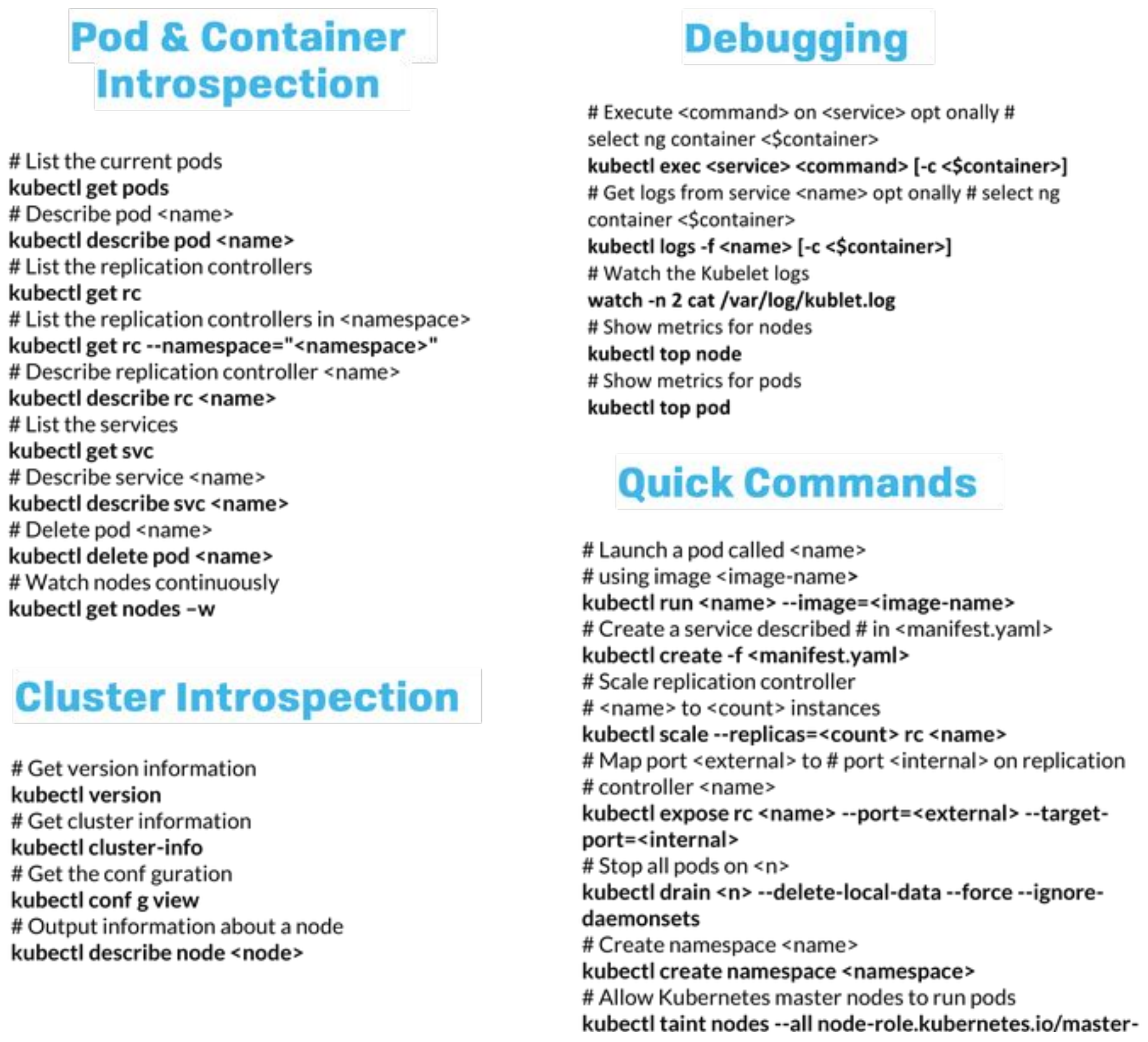
Kubernetes - K8s - Terminologies

- **Cluster** - A set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.
- **Container** - A lightweight and portable executable image that contains software and all of its dependencies.
- **Kubectl** - Also known as:kubectl - Command line tool for communicating with a Kubernetes cluster control plane, using the Kubernetes API.
- **Node** - A node is a worker machine in Kubernetes.
- **Pod** - The smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster.

Kubernetes - K8s



Kubernetes - K8s - Cheat Sheet Commands for CLI



Pod & Container Introspection

```
# List the current pods
kubectl get pods
# Describe pod <name>
kubectl describe pod <name>
# List the replication controllers
kubectl get rc
# List the replication controllers in <namespace>
kubectl get rc --namespace="<namespace>"
# Describe replication controller <name>
kubectl describe rc <name>
# List the services
kubectl get svc
# Describe service <name>
kubectl describe svc <name>
# Delete pod <name>
kubectl delete pod <name>
# Watch nodes continuously
kubectl get nodes –w
```

Cluster Introspection

```
# Get version information
kubectl version
# Get cluster information
kubectl cluster-info
# Get the conf guration
kubectl conf g view
# Output information about a node
kubectl describe node <node>
```

Debugging

```
# Execute <command> on <service> opt onally #
select ng container <$container>
kubectl exec <service> <command> [-c <$container>]
# Get logs from service <name> opt onally # select ng
container <$container>
kubectl logs -f <name> [-c <$container>]
# Watch the Kubelet logs
watch -n 2 cat /var/log/kublet.log
# Show metrics for nodes
kubectl top node
# Show metrics for pods
kubectl top pod
```

Quick Commands

```
# Launch a pod called <name>
# using image <image-name>
kubectl run <name> --image=<image-name>
# Create a service described # in <manifest.yaml>
kubectl create -f <manifest.yaml>
# Scale replication controller
# <name> to <count> instances
kubectl scale --replicas=<count> rc <name>
# Map port <external> to # port <internal> on replication
# controller <name>
kubectl expose rc <name> --port=<external> --target-
port=<internal>
# Stop all pods on <n>
kubectl drain <n> --delete-local-data --force --ignore-
daemonsets
# Create namespace <name>
kubectl create namespace <namespace>
# Allow Kubernetes master nodes to run pods
kubectl taint nodes --all node-role.kubernetes.io/master-
```

Activity:

Thank you