Docker

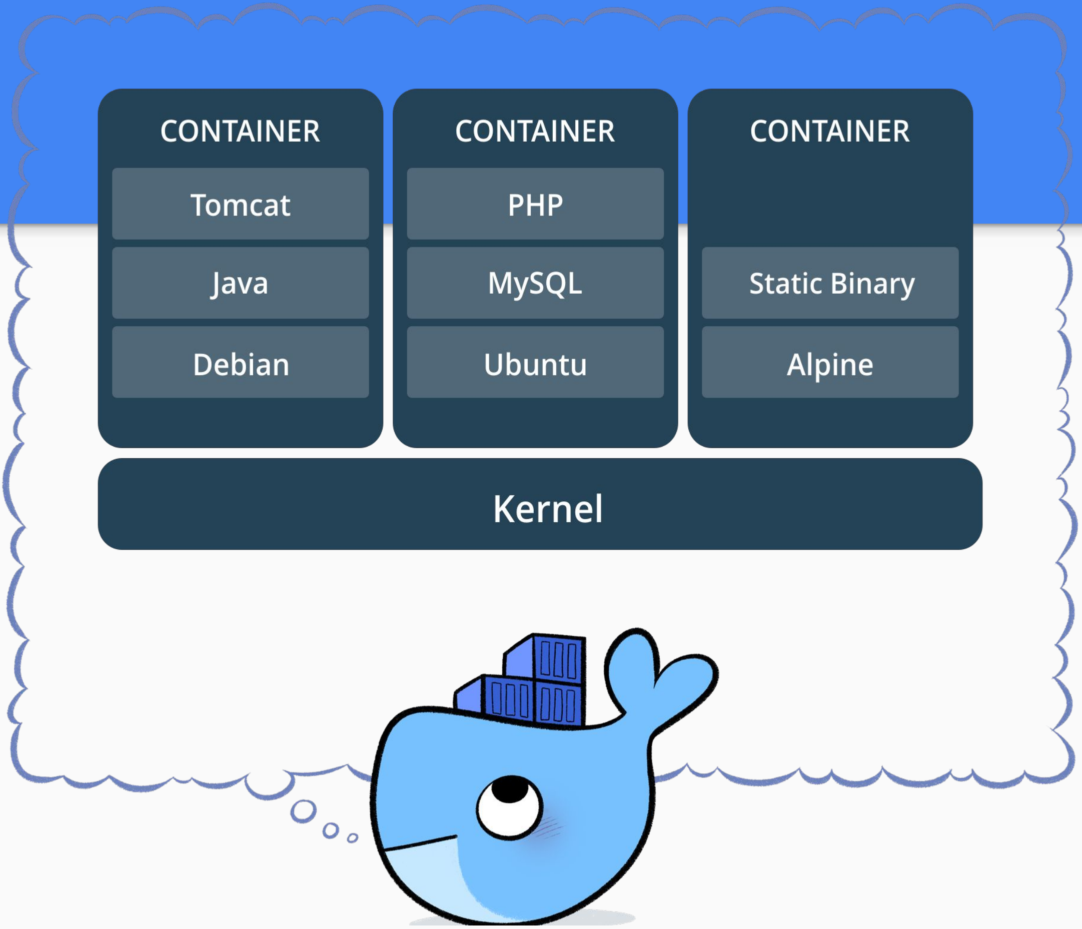
Presented by: Upasana Ghosh

What is Docker?

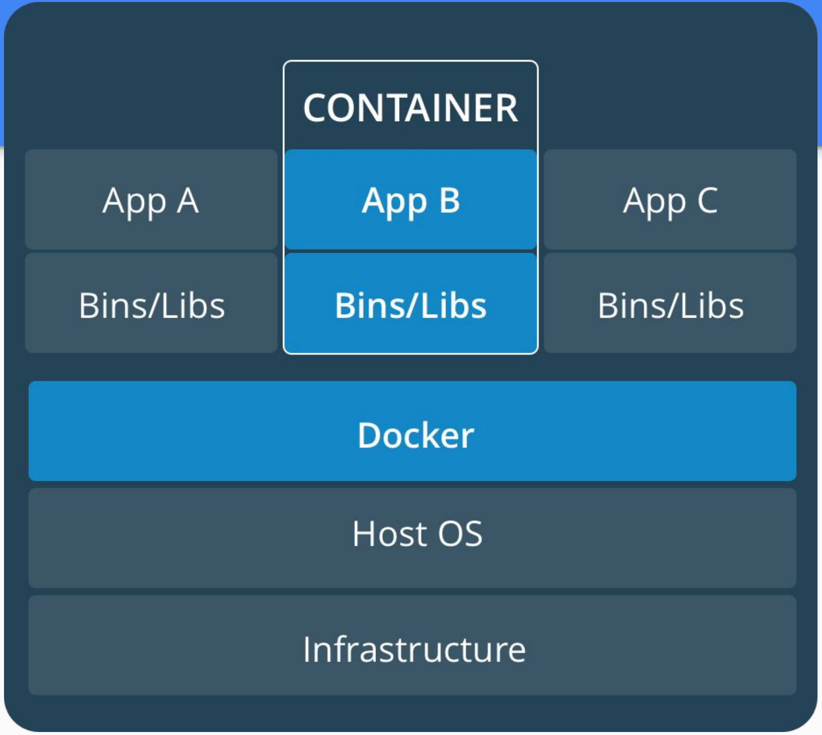
- Open source tool.
- Designed to makes it easier to create, deploy and run applications by using containers.
- Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package, called image.

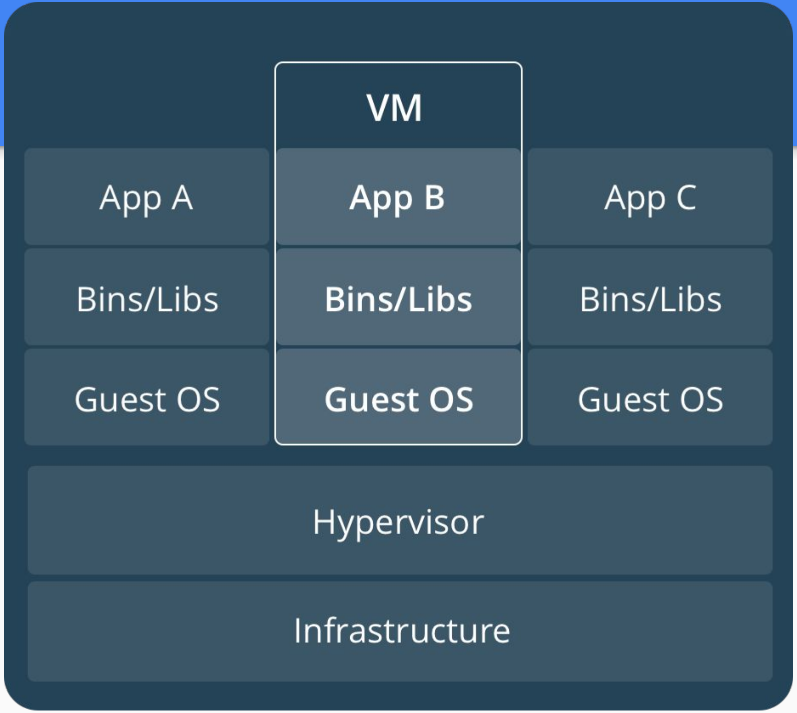
Container Image

- Lightweight
- Stand-alone
- Executable package
- Portable



Comparing Containers and Virtual Machines





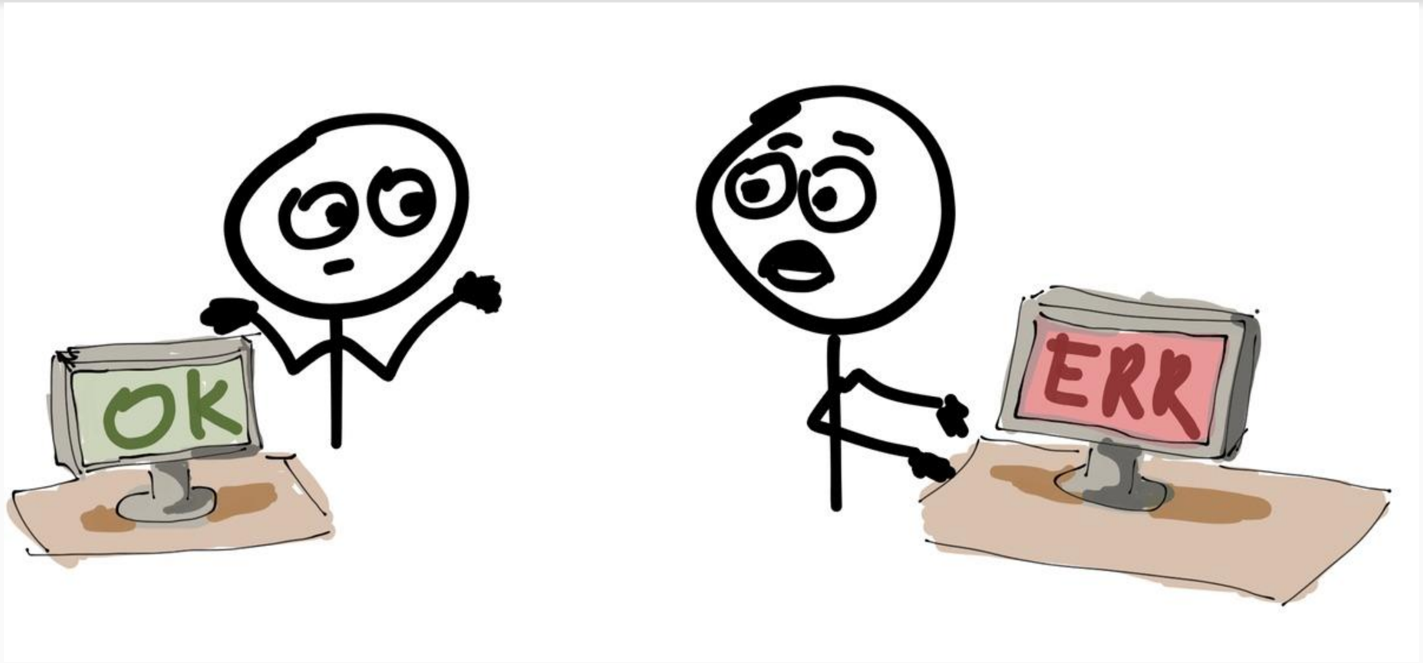
Containers

Virtual Machines

Comparing Containers and Virtual Machines

- Both provide resource isolation and allocation benefits but are functionally differently.
- Containers virtualize the operating system instead of hardware.
- Containers are more portable and efficient.
- Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.
- Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

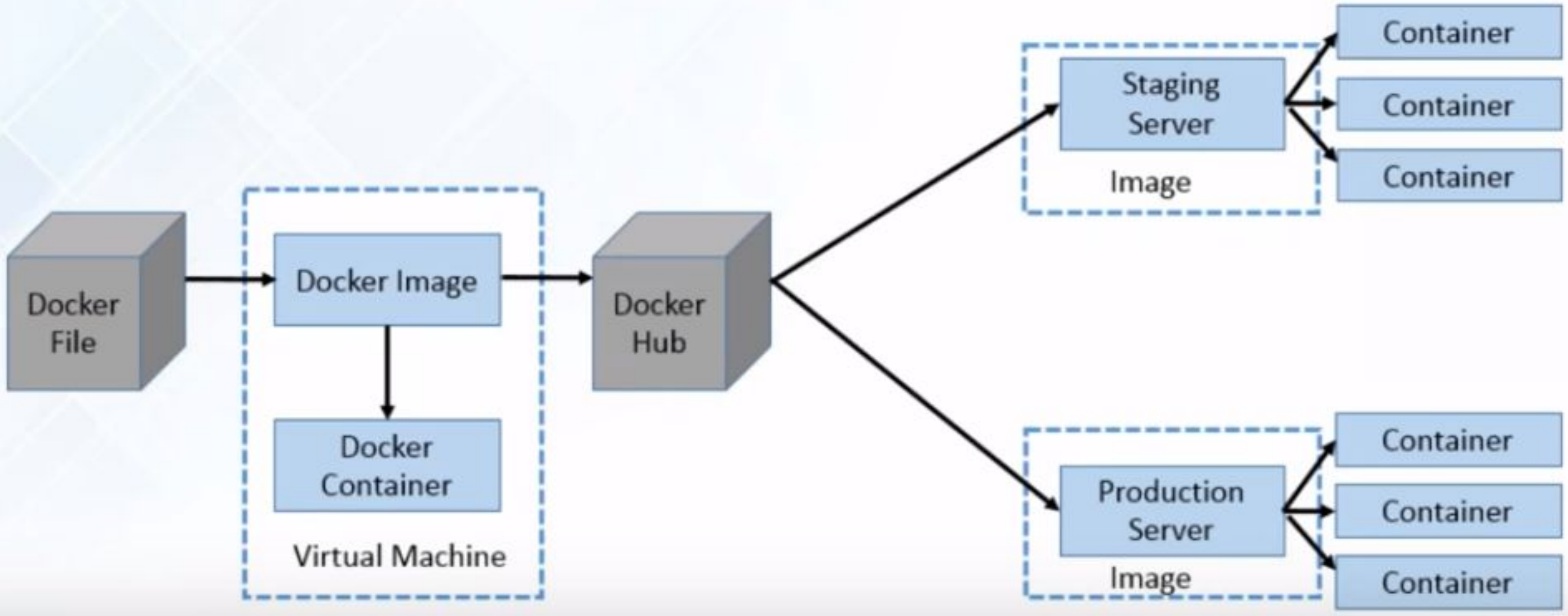
Why docker



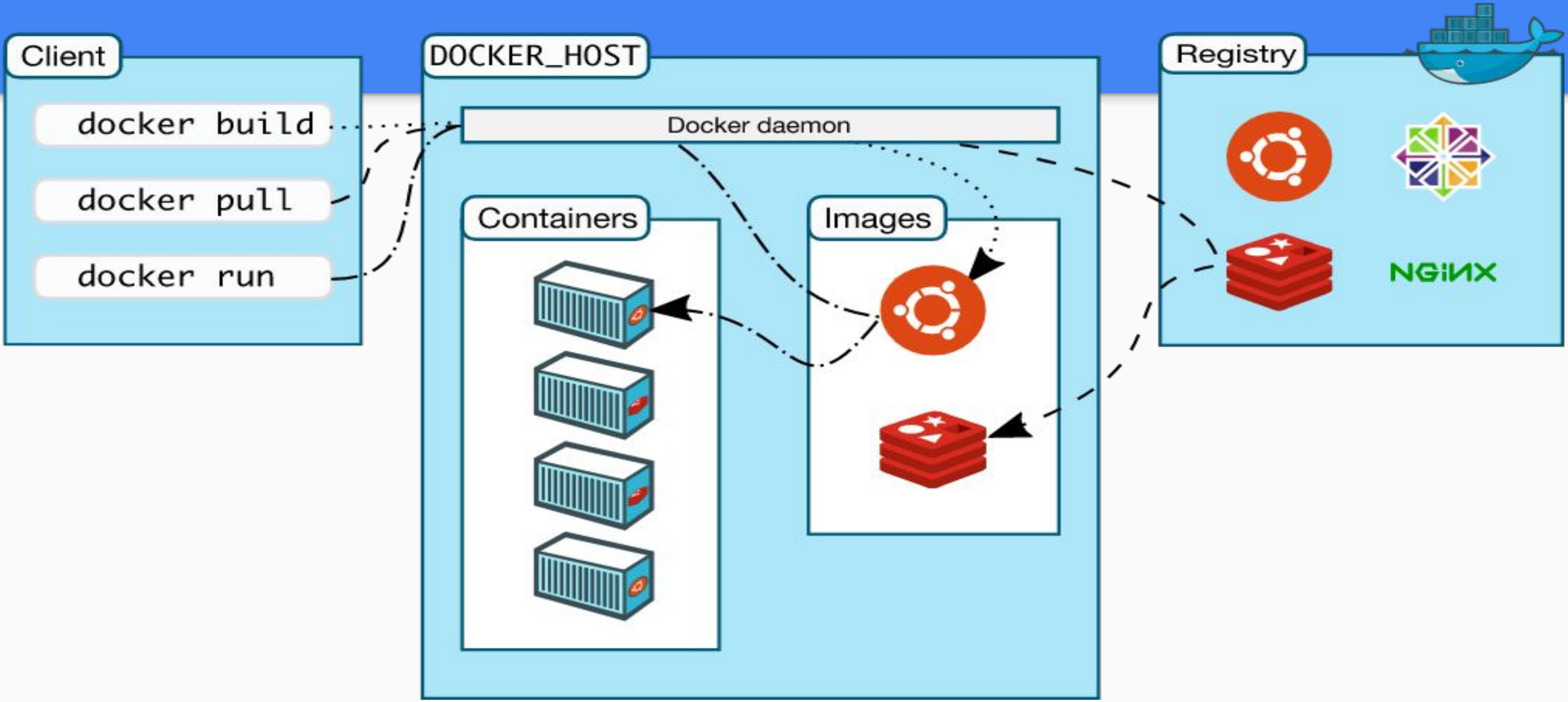
Why docker

- Ensures that the **application will run the same** no matter which server or laptop its running on.
- This way, it **eliminates the “it works on my machine”** problem.
- Developers will not spend time in setting up environments or debugging environment-specific issues.
- Ensures consistent environments from development to production.

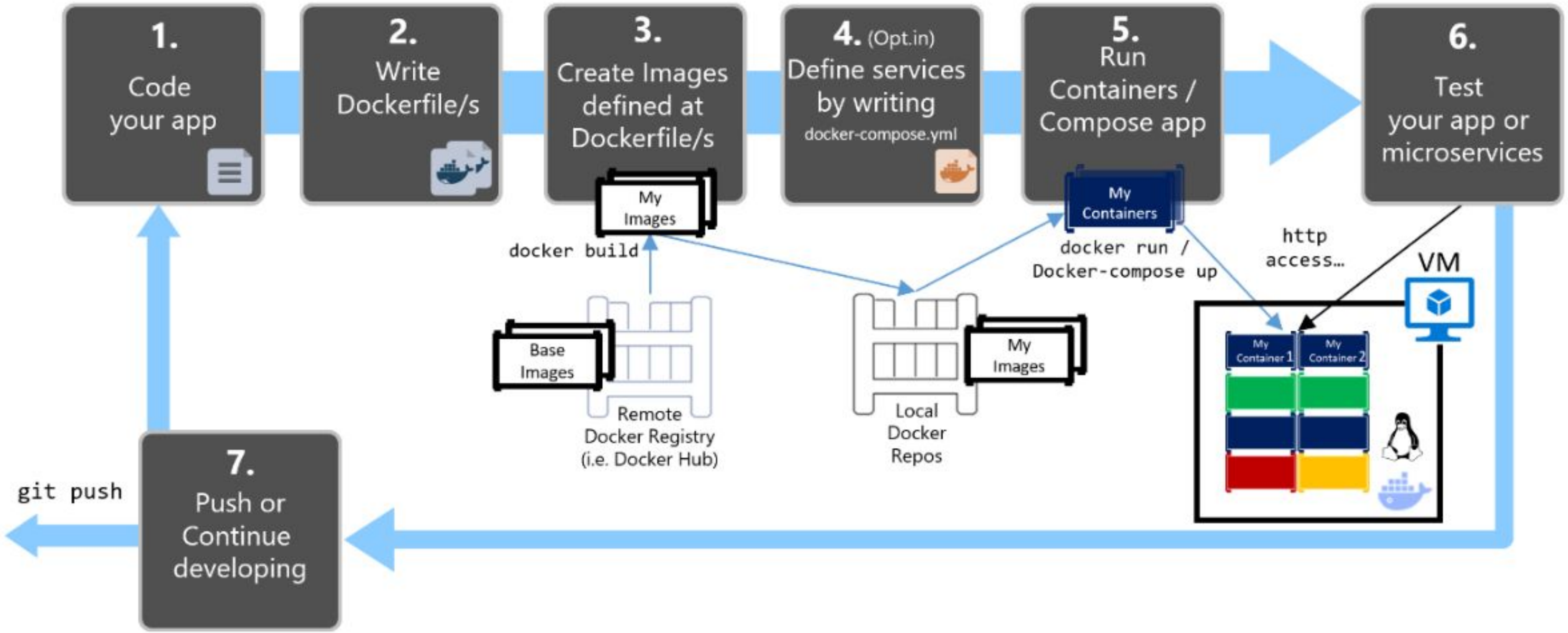
Docker in a nutshell



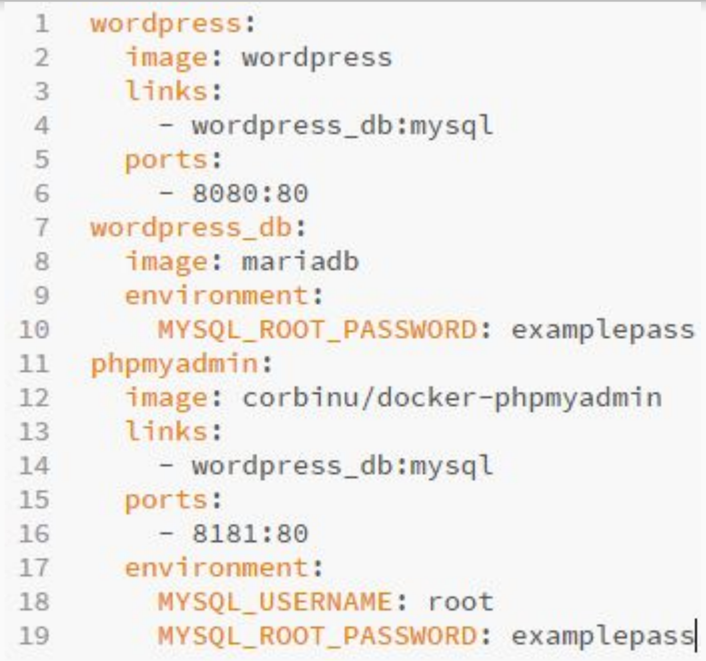
Docker architecture



Inner-Loop development workflow for Docker apps

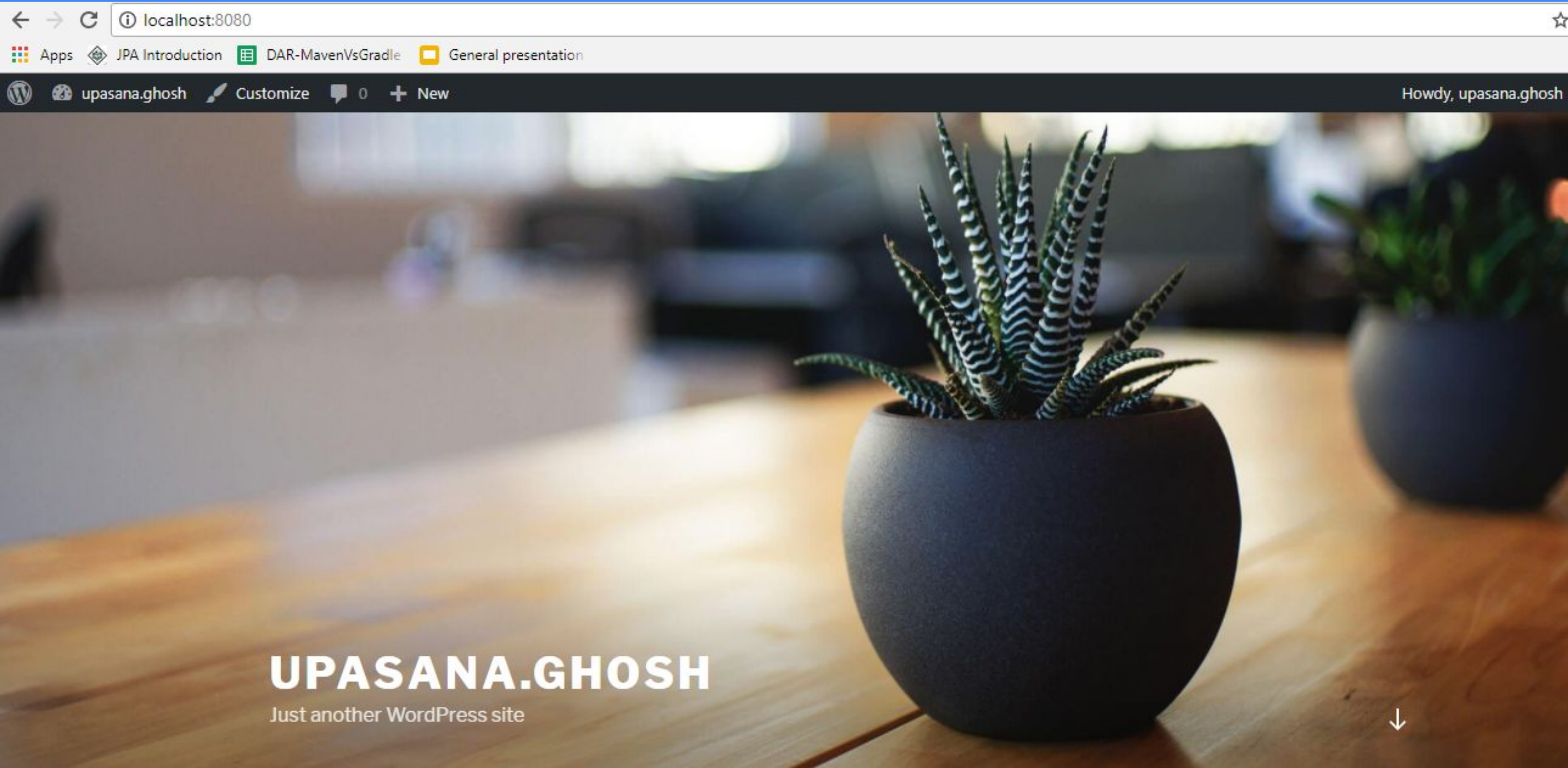

1. Code your app
2. Write Dockerfile/s
3. Create Images defined at Dockerfile/s
4. (Opt.in) Define services by writing docker-compose.yml
5. Run Containers / Compose app
6. Test your app or microservices
7. Push or Continue developing
My Images
docker build
Base Images
Remote Docker Registry (i.e. Docker Hub)
Local Docker Repos
My Images
My Containers
docker run / Docker-compose up
http access…
VM
My Container 1
My Container 2
git push

Example: docker-compose.yml

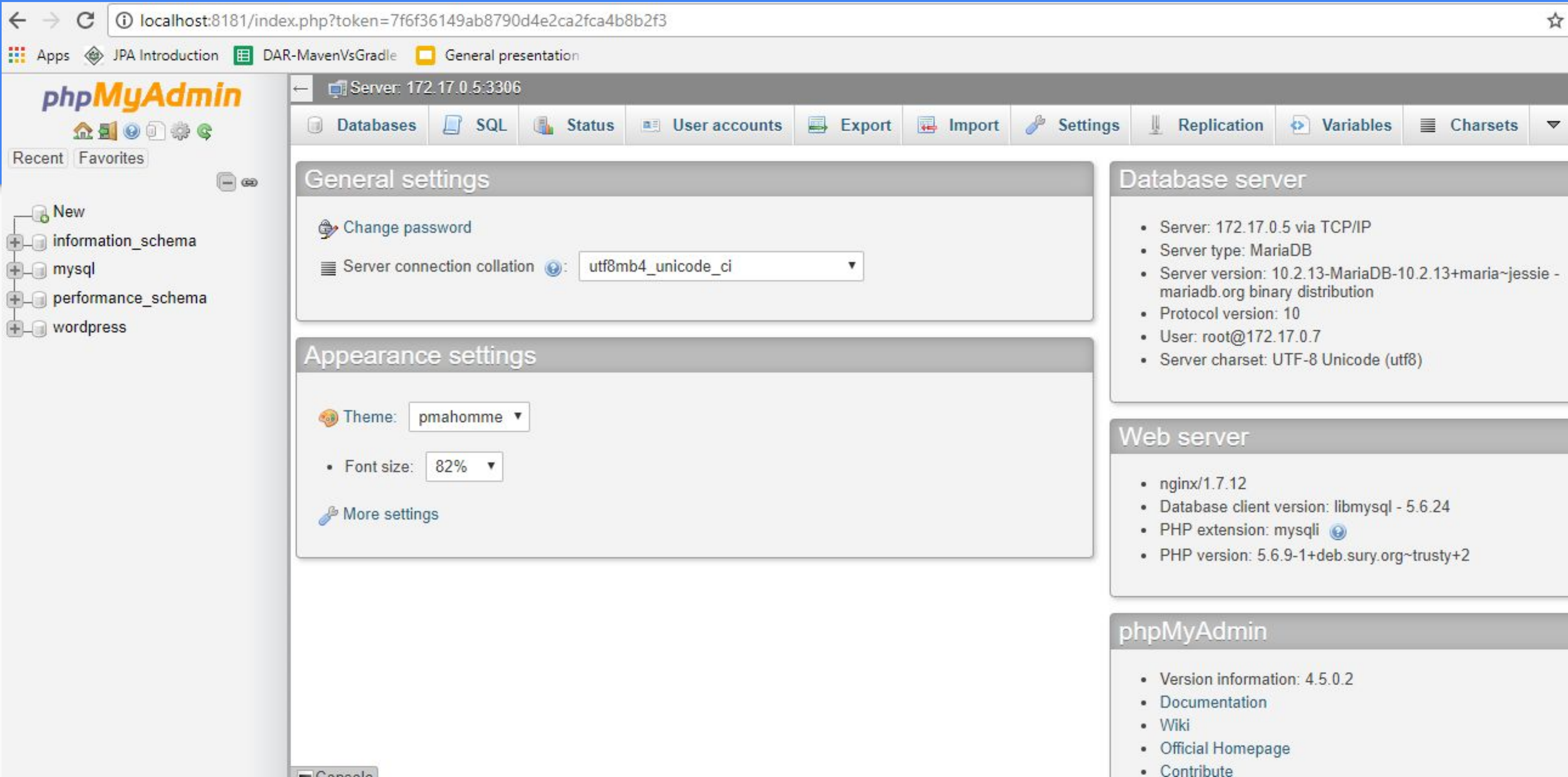


```yaml
wordpress:
  image: wordpress
  links:
    - wordpress_db:mysql
  ports:
    - 8080:80
wordpress_db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: examplepass
phpmyadmin:
  image: corbinu/docker-phpmyadmin
  links:
    - wordpress_db:mysql
  ports:
    - 8181:80
  environment:
    MYSQL_USERNAME: root
    MYSQL_ROOT_PASSWORD: examplepass
```

Localhost:8080 (wordpress)



Localhost:8181 (phpMyAdmin)



General settings

Change password

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Theme: pmahomme

- Font size: 82%

More settings

Database server

- Server: 172.17.0.5 via TCP/IP
- Server type: MariaDB
- Server version: 10.2.13-MariaDB-10.2.13+maria~jessie - mariadb.org binary distribution
- Protocol version: 10
- User: root@172.17.0.7
- Server charset: UTF-8 Unicode (utf8)

Web server

- nginx/1.7.12
- Database client version: libmysql - 5.6.24
- PHP extension: mysqli
- PHP version: 5.6.9-1+deb.sury.org~trusty+2

phpMyAdmin

- Version information: 4.5.0.2
- Documentation
- Wiki
- Official Homepage
- Contribute

Container Orchestration

- Refers to the automated arrangement, coordination, and management of software containers.
- It provides:

- Load Balancing
- Storage management
- Health checks
- Auto-[scaling/restart/healing] of containers and nodes
- Zero-downtime deploys

Kubernetes

- Open source container orchestration tool.
- Used to automate deployments, scaling, and operations of application containers across clusters of hosts
- Capable of doing auto-placement, auto-restart, auto-replication and auto-healing of containers extremely well.

Common terms associated with Kubernetes

- Kubernetes deploys and schedules containers in groups called **pods**.
- **API Server**: This component is the management hub for the Kubernetes master node. It facilitates communication between the various components, thereby maintaining cluster health.
- **Controller Manager**: This component ensures that the cluster’s desired state matches the current state by scaling workloads up and down.
- **Scheduler**: This component places the workload on the appropriate node – in this case all workloads will be placed locally on your host.
- **Kubelet**: This component receives pod specifications from the API Server and manages pods running in the host.

References:

- https://www.docker.com/
- https://docs.docker.com/
- https://docs.docker.com/docker-for-windows/install/
- https://www.docker.com/what-container
- https://kubernetes.io/
- https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

Thanks!