KODEKLOUD

MUMSHAD
MANNAMBETH

KODEKLOUD
www.kodekloud.com

KODEKLOUD

Objectives

- What are Containers?
- What is Docker?
- Why do you need it?
- What can it do?

- Run Docker Containers
- Create a Docker Image
- Networks in Docker
- Docker Compose

- Docker Concepts in Depth

- Docker for Windows/Mac

- Docker Swarm
- Docker vs Kubernetes

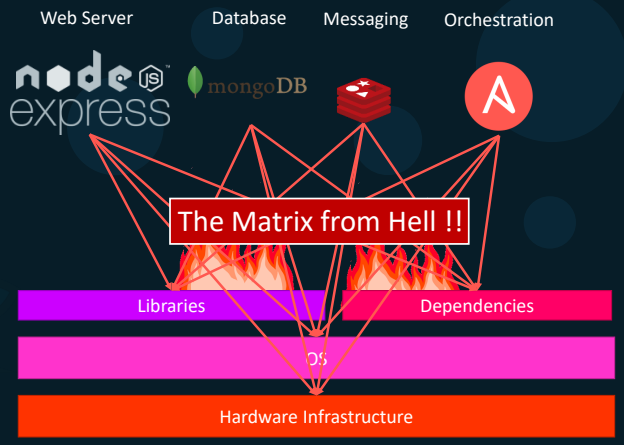
KODEKLOUD

KODEKLOUD

docker
overview

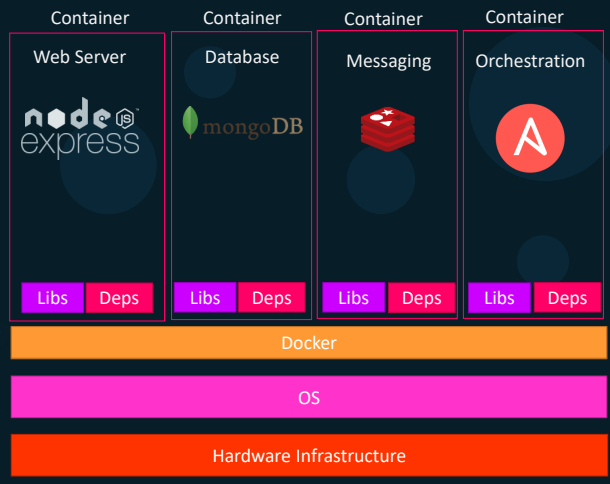
Why do you need docker?

- Compatibility/Dependency
- Long setup time
- Different Dev/Test/Prod environments

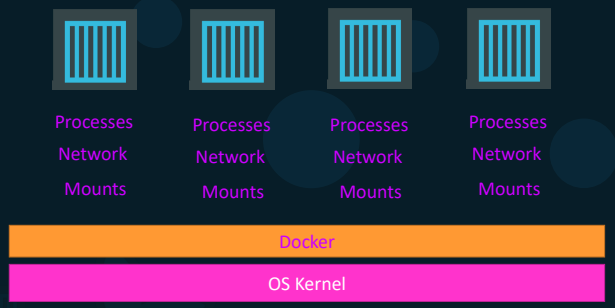


What can it do?

- Containerize Applications
- Run each service with its own dependencies in separate containers

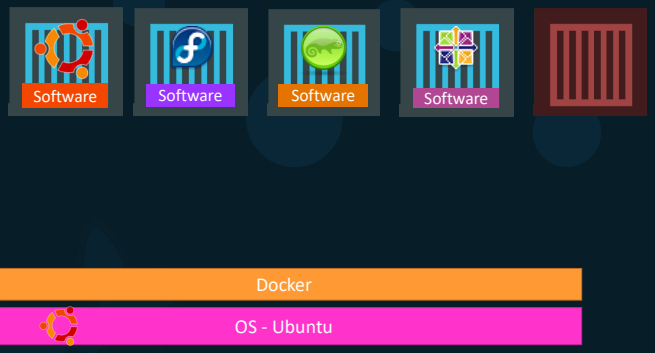


What are containers?

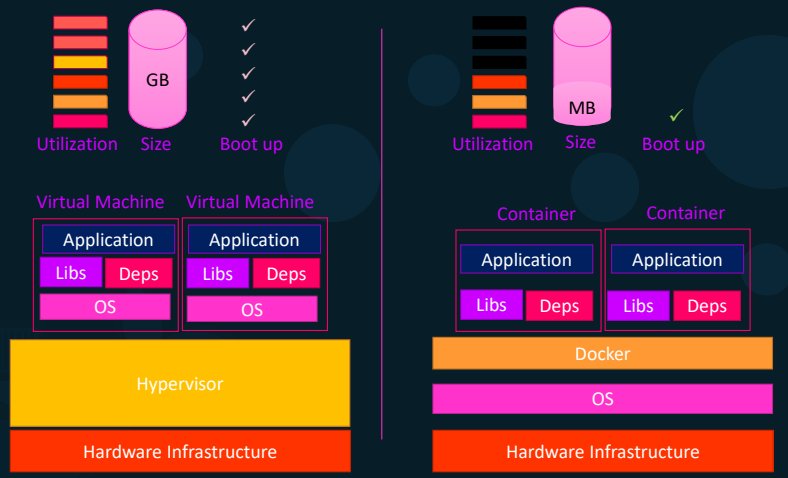


Operating System

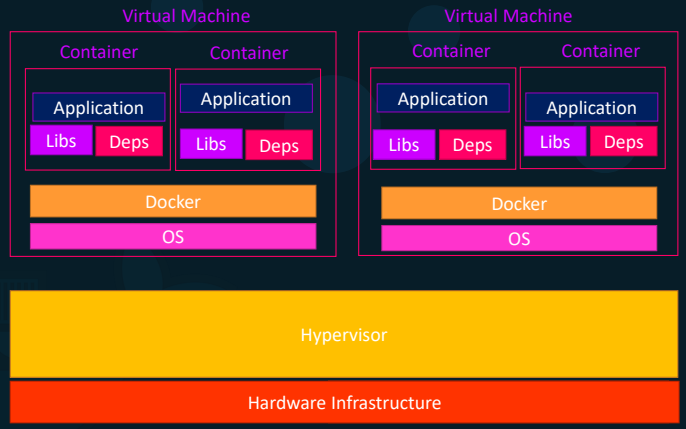
Sharing the kernel



Containers vs Virtual Machines



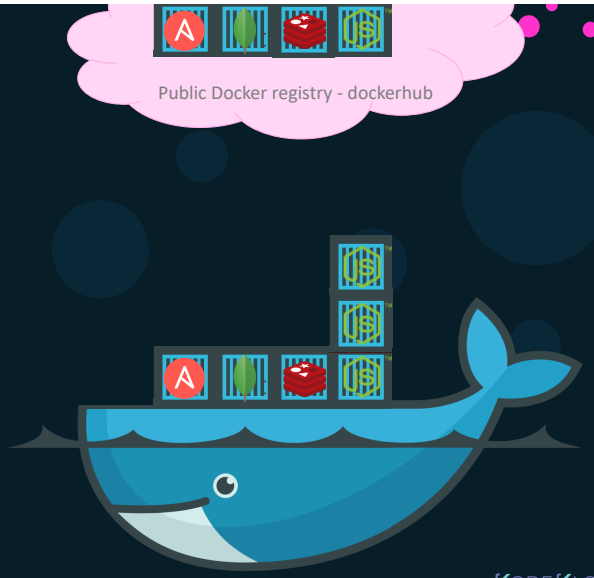
Containers & Virtual Machines



KODEKLOUD

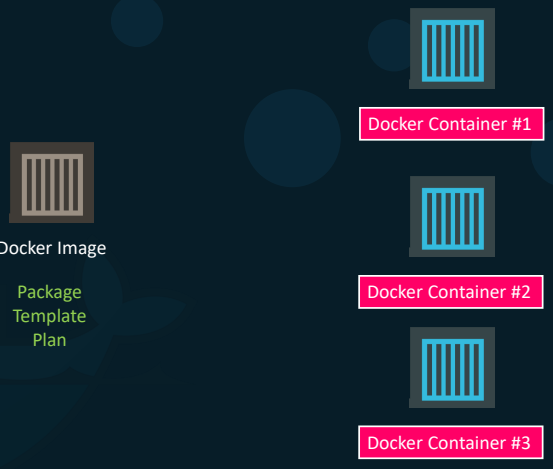
How is it done?

```
docker run ansible
docker run mongodb
docker run redis
docker run nodejs
docker run nodejs
docker run nodejs
```



KODEKLOUD

Container vs image



KODEKLOUD

KODEKLOUD

d o c k e r

Getting Started

KODEKLOUD

Docker Editions

Community Edition

Enterprise Edition

KODEKLOUD

Community Edition

Linux

MAC

Windows

Cloud

KODEKLOUD

KODEKLOUD

docker
commands

KODEKLOUD

Run – start a container

```
docker run nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```

KODEKLOUD

ps – list containers

```
docker ps
CONTAINER ID  IMAGE   COMMAND                  CREATED         STATUS          PORTS      NAMES
796856ac413d  nginx   "nginx -g 'daemon of…"   7 seconds ago   Up 6 seconds    80/tcp     silly_sammet
```

```
docker ps -a
CONTAINER ID  IMAGE   COMMAND                  CREATED         STATUS                     NAMES
796856ac413d  nginx   "nginx -g 'daemon of…"   7 seconds ago   Up 6 seconds               silly_sammet
cff8ac918a2f  redis   "docker-entrypoint.s…"   6 seconds ago   Exited (0) 3 seconds ago   relaxed_aryabhata
```

KODEKLOUD

STOP – stop a container

```
docker ps
CONTAINER ID  IMAGE   COMMAND                  CREATED         STATUS          PORTS      NAMES
796856ac413d  nginx   "nginx -g 'daemon of…"   7 seconds ago   Up 6 seconds    80/tcp     silly_sammet
```

```
docker stop silly_sammet
silly_sammet
```

```
docker ps -a
CONTAINER ID  IMAGE   COMMAND                  CREATED         STATUS                     NAMES
796856ac413d  nginx   "nginx -g 'daemon of…"   7 seconds ago   Exited (0) 3 seconds ago   silly_sammet
cff8ac918a2f  redis   "docker-entrypoint.s…"   6 seconds ago   Exited (0) 3 seconds ago   relaxed_aryabhata
```

KODEKLOUD

Rm – Remove a container

```
docker rm silly_sammet
```

```
silly_sammet
```

```
docker ps -a
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      NAMES
cff8ac918a2f   redis     "docker-entrypoint.s…"   6 seconds ago    Exited (0) 3 seconds ago    relaxed_aryabhata
```


images – List images

```
docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f68d6e55e065        4 days ago          109MB
redis               latest              4760dc956b2d        15 months ago       107MB
ubuntu              latest              f975c5035748        16 months ago       112MB
alpine              latest              3fd9065eaf02        18 months ago       4.14MB
```


rmi – Remove images

```
docker rmi nginx
```

```
Untagged: nginx:latest
Untagged: nginx@sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Deleted: sha256:f68d6e55e06520f152403e6d96d0de5c9790a89b4cfc99f4626f68146fa1dbdc
Deleted: sha256:1b0c768769e2bb66e74a205317ba531473781a78b77feef8ea6fd7be7f4044e1
Deleted: sha256:34138fb60020a180e512485fb96fd42e286fb0d86cf1fa2506b11ff6b945b03f
Deleted: sha256:cf5b3c6798f77b1f78bf4e297b27cfa5b6caa982f04caeb5de7d13c255fd7a1e
```

! Delete all dependent containers to remove image

Pull – download an image

```
docker run nginx
```

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```

```
docker pull nginx
```

```
Using default tag: latest
latest: Pulling from library/nginx
fc7181108d40: Pull complete
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```

```
docker run ubuntu
```

```
docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
```

```
docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
45aacca36850        ubuntu              "/bin/bash"         43 seconds ago      Exited (0) 41 seconds ago
```



```
docker run ubuntu
```

```
docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
```

```
docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
45aacca36850        ubuntu              "/bin/bash"         43 seconds ago      Exited (0) 41 seconds ago
```


Append a command

```
docker run ubuntu
```

```
docker run ubuntu sleep 5
```

Exec – execute a command

```
docker ps -a
```

```
CONTAINER ID        IMAGE     COMMAND       CREATED             STATUS              NAMES
538d037f94a7        ubuntu    "sleep 100“   6 seconds ago       Up 4 seconds        distracted_mcclintock
```

```
docker exec distracted_mcclintock cat /etc/hosts
```

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.18.0.2      538d037f94a7
```

Run – attach and detach

```
docker run kodekloud/simple-webapp
```

```
This is a sample web application that displays a colored background.
 * Serving Flask app "app" (lazy loading)
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

```
docker run -d kodekloud/simple-webapp
```

```
a043d40f85fefa414254e4775f9336ea59e19e5cf597af5c554e0a35a1631118
```

```
docker attach a043d
```

docker

run

Run – tag

```
docker run redis
```

```
Using default tag: latest
latest: Pulling from library/redis
f5d23c7fed46: Pull complete
Status: Downloaded newer image for redis:latest

1:C 31 Jul 2019 09:02:32.624 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 31 Jul 2019 09:02:32.624 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 2019 09:02:32.626 # Server initialized
```

```
docker run redis:4.0
```

TAG

```
Unable to find image 'redis:4.0' locally
4.0: Pulling from library/redis
e44f086c03a2: Pull complete
Status: Downloaded newer image for redis:4.0

1:C 31 Jul 09:02:56.527 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 31 Jul 09:02:56.527 # Redis version=4.0.14, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 09:02:56.530 # Server initialized
```

RUN - STDIN

```
~/prompt-application$ ./app.sh
Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

```
docker run kodekloud/simple-prompt-docker

Hello and Welcome !
```

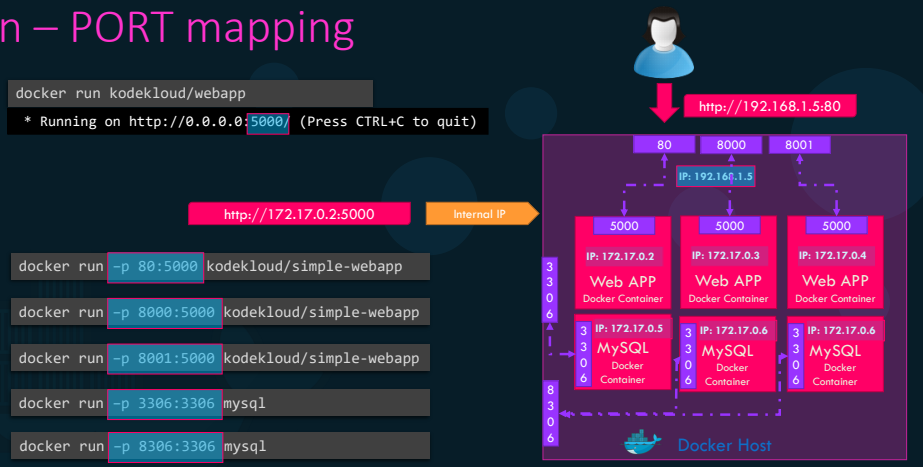
```
docker run -i kodekloud/simple-prompt-docker
Mumshad

Hello and Welcome Mumshad!
```

```
docker run -it kodekloud/simple-prompt-docker
Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

KODEKLOUD

Run – PORT mapping

```
docker run kodekloud/webapp
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```




```
docker run -p 80:5000 kodekloud/simple-webapp

docker run -p 8000:5000 kodekloud/simple-webapp

docker run -p 8001:5000 kodekloud/simple-webapp

docker run -p 3306:3306 mysql

docker run -p 8306:3306 mysql

docker run -p 8306:3306 mysql
```

```
root@osboxes:/root # docker run -p 8306:3306 -e MYSQL_ROOT_PASSWORD=pass mysql
docker: Error response from daemon: driver failed programming external connectivity on endpoint boring_bhabha (
5079d342b7e8ee11c71d46): Bind for 0.0.0.0:8306 failed: port is already allocated.
```

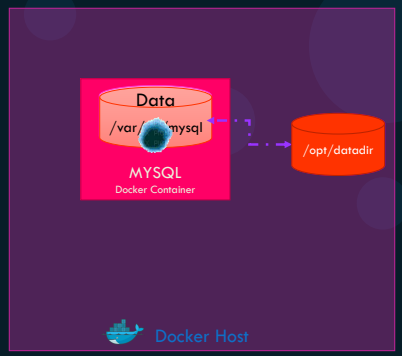
KODEKLOUD

RUN – Volume mapping

```
docker run mysql
```

```
docker stop mysql
docker rm mysql
```

```
docker run -v /opt/datadir:/var/lib/mysql mysql
```



KODEKLOUD

Inspect Container

```
docker inspect blissful_hopper
```

```
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "Name": "/blissful_hopper",
        "Path": "python",
        "Args": [
            "app.py"
        ],
        "State": {
            "Status": "running",
            "Running": true,
        },

        "Mounts": [],
        "Config": {
           "Entrypoint": [
                "python",
                "app.py"
            ],
        },
        "NetworkSettings": {..}
    }
]
```

KODEKLOUD

Container Logs

```
docker logs blissful_hopper
```

```
 This is a sample web application that displays a colored background.
 A color can be specified in two ways.

 1. As a command line argument with --color as the argument. Accepts one of
red,green,blue,blue2,pink,darkblue
 2. As an Environment variable APP_COLOR. Accepts one of
red,green,blue,blue2,pink,darkblue
 3. If none of the above then a random color is picked from the above list.
 Note: Command line argument precedes over environment variable.


No command line argument or environment variable. Picking a Random Color =blue
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

KODEKLOUD

KODEKLOUD

docker

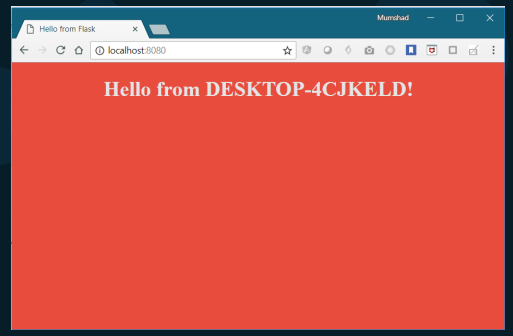
environment variables

KODEKLOUD

Environment Variables

app.py

```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```




```
python app.py
```

KODEKLOUD

Environment Variables

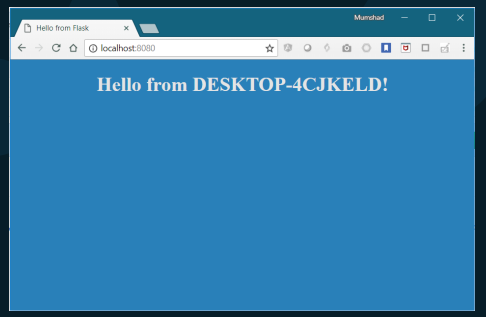
app.py

```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = "red"

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```


Environment Variables

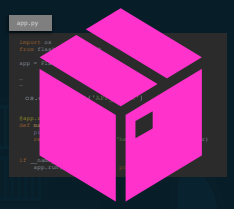
app.py

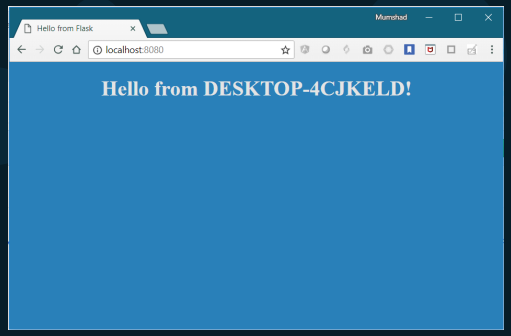
```
import os
from flask import Flask

app = Flask(__name__)

…
…

color = os.environ.get('APP_COLOR')

@app.route("/")
def main():
    print(color)
    return render_template('hello.html', color=color)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```




```
export APP_COLOR=blue; python app.py
```


ENV Variables in Docker





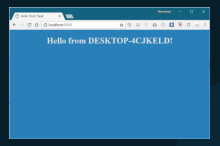

```
docker run -e APP_COLOR=blue
```

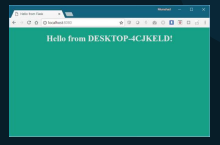

ENV Variables in Docker

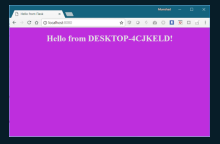
```
docker run -e APP_COLOR=blue    simple-webapp-color
```

```
docker run  -e APP_COLOR=green simple-webapp-color
```

```
docker run  -e APP_COLOR=pink simple-webapp-color
```







Inspect Environment Variable

```
docker inspect blissful_hopper
```

```
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "State": {
            "Status": "running",
            "Running": true,
        },

        "Mounts": [],
        "Config": {
            "Env": [
                "APP_COLOR=blue",
                "LANG=C.UTF-8",
                "GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D",
                "PYTHON_VERSION=3.6.6",
                "PYTHON_PIP_VERSION=18.1"
            ],
            "Entrypoint": [
                "python",
                "app.py"
            ],
        }
]
```

KODEKLOUD

KODEKLOUD

docker images

KODEKLOUD

What am I containerizing?

Flask
web development,
one drop at a time

KODEKLOUD

How to create my own image?

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

1. OS - Ubuntu
2. Update apt repo
3. Install dependencies using apt
4. Install Python dependencies using pip
5. Copy source code to /opt folder
6. Run the web server using "flask" command

```
docker build Dockerfile -t mmumshad/my-custom-app
```

```
docker push mmumshad/my-custom-app
```

KODEKLOUD

Dockerfile

Dockerfile

```
INSTRUCTION  ARGUMENT
```

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Start from a base OS or another image

Install all dependencies

Copy source code

Specify Entrypoint

KODEKLOUD

Layered architecture

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
docker build Dockerfile -t mmumshad/my-custom-app
```

Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint with "flask" command	0 B

```
root@osboxes:/root/simple-webapp-docker # docker history mmumshad/simple-webapp
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
1a45ba829f10        About an hour ago   /bin/sh -c #(nop)  ENTRYPOINT ["/bin/sh" "...   0B
37d37ed8fe99        About an hour ago   /bin/sh -c #(nop) COPY file:29b92853d73898...   229B
d6aaebf8ded0        About an hour ago   /bin/sh -c pip install flask flask-mysql        6.39MB
e4c055538e60        About an hour ago   /bin/sh -c apt-get update && apt-get insta...   306MB
ccc7a11d65b1        2 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           2 weeks ago         /bin/sh -c mkdir -p /run/systemd && echo '...   7B
<missing>           2 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\...   2.76kB
<missing>           2 weeks ago         /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>           2 weeks ago         /bin/sh -c set -xe   && echo '#!/bin/sh' >...   745B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:39d3593ea220e68...   120MB
```

KODEKLOUD

Docker build output

```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM ubuntu
 ---> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-setuptools python-de
 ---> Running in a7840dbfad17
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [46.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [440 kB]
Step 3/5 : RUN pip install flask flask-mysql
 ---> Running in a4a6c9190ba3
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting flask-mysql
  Downloading Flask_MySQL-1.4.0-py2.py3-none-any.whl
Removing intermediate container a4a6c9190ba3
Step 4/5 : COPY app.py /opt/
 ---> e7cdab17e782
Removing intermediate container faaaaf63c512
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
 ---> Running in d452c574a8bb
 ---> 9f27c36920bc
Removing intermediate container d452c574a8bb
Successfully built 9f27c36920bc
```

KODEKLOUD

failure

- Layer 1. Base Ubuntu Layer
- Layer 2. Changes in apt packages
- Layer 3. Changes in pip packages
- Layer 4. Source code
- Layer 5. Update Entrypoint with “flask” command

```
docker build Dockerfile -t mmumshad/my-custom-app
```

```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon   5.12kB
Step 1/5 : FROM ubuntu
 ---> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-pip
 ---> Using cache
 ---> e4c055538e60
Step 3/5 : RUN pip install flask
 ---> Running in aacdaccd7403
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Removing intermediate container aacdaccd7403
Step 4/5 : COPY app.py /opt/
 ---> af41ef57f6f3
Removing intermediate container a49cc8befc8f
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
 ---> Running in 3d745ff07d5a
 ---> 910416d360b6
Removing intermediate container 3d745ff07d5a
Successfully built 910416d360b6
```

KODEKLOUD

What can you containerize?



Containerize Everything!!!

KODEKLOUD

KODEKLOUD

docker

CMD

vs

ENTRYPOINT

KODEKLOUD

```
docker run ubuntu
```

```
docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
```

```
docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
45aacca36850        ubuntu              "/bin/bash"         43 seconds ago      Exited (0) 41 seconds ago
```



```
# Install Nginx.
RUN \
  add-apt-repository -y ppa:nginx/stable && \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/cor

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]
```

```
ARG MYSQL_SERVER_PACKAGE_URL=https://repo.mysql.com/yum/mysql-8.0-community/docker/x86_
ARG MYSQL_SHELL_PACKAGE_URL=https://repo.mysql.com/yum/mysql-tools-community/el/7/x86_(

# Install server
RUN rpmkeys --import https://repo.mysql.com/RPM-GPG-KEY-mysql \
  && yum install -y $MYSQL_SERVER_PACKAGE_URL $MYSQL_SHELL_PACKAGE_URL libpwquality \
  && yum clean all \
  && mkdir /docker-entrypoint-initdb.d

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /entrypoint.sh
COPY healthcheck.sh /healthcheck.sh
ENTRYPOINT ["/entrypoint.sh"]
HEALTHCHECK CMD /healthcheck.sh
EXPOSE 3306 33060
CMD ["mysqld"]
```



```
# Pull base image.
FROM ubuntu:14.04

# Install.
RUN \
  sed -i 's/# \(.*multiverse$\)/\1/g' /etc/apt/sources.list && \
  apt-get update && \
  apt-get -y upgrade && \
  apt-get install -y build-essential && \
  apt-get install -y software-properties-common && \
  apt-get install -y byobu curl git htop man unzip vim wget && \
  rm -rf /var/lib/apt/lists/*

# Add files.
ADD root/.bashrc /root/.bashrc
ADD root/.gitconfig /root/.gitconfig
ADD root/.scripts /root/.scripts

# Set environment variables.
ENV HOME /root

# Define working directory.
WORKDIR /root

# Define default command.
CMD ["bash"]
```

```
docker run ubuntu [COMMAND]
```

```
docker run ubuntu sleep 5
```

KODEKLOUD

```
FROM Ubuntu

CMD sleep 5
```

CMD command param1

CMD ["command", "param1"]

CMD sleep 5

CMD ["sleep", "5"] ✓

CMD ["sleep 5"] ✗

```
docker build -t ubuntu-sleeper .
```

```
docker run ubuntu-sleeper
```

KODEKLOUD

```
FROM Ubuntu

CMD sleep 5
```

```
docker run ubuntu-sleeper sleep 10
```

Command at Startup: sleep 10

```
FROM Ubuntu

ENTRYPOINT ["sleep"]
```

```
docker run ubuntu-sleeper 10
```

Command at Startup:

```
docker run ubuntu-sleeper
sleep: missing operand
Try 'sleep --help' for more information.
```

Command at Startup:

KODEKLOUD

```
FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]
```

```
docker run ubuntu-sleeper
sleep: missing operand
Try 'sleep --help' for more information.
```

Command at Startup:

```
docker run ubuntu-sleeper 10.0
```

Command at Startup:

```
docker run --entrypoint sleep2.0 ubuntu-sleeper 10
```

Command at Startup:

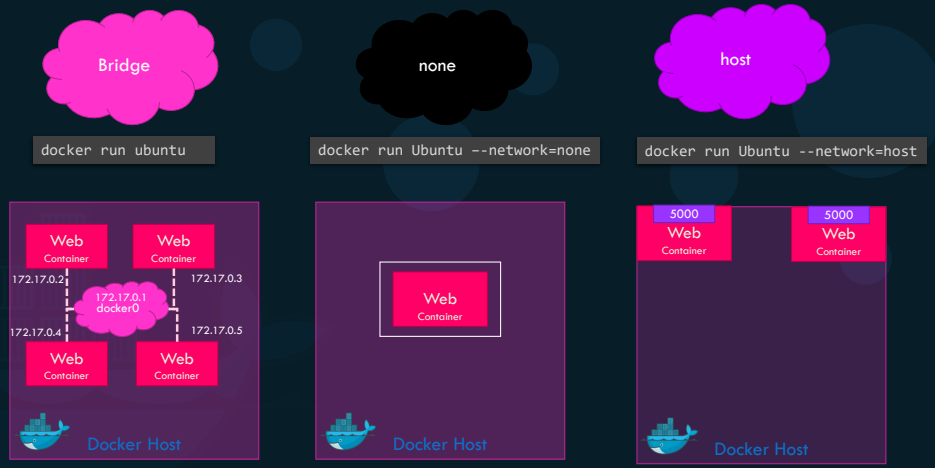
KODEKLOUD

KODEKLOUD

docker networking

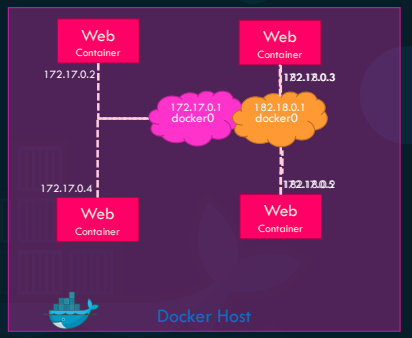
KODEKLOUD

Default networks



KODEKLOUD

User-defined networks




```
docker network create \
    --driver bridge \
    --subnet 182.18.0.0/16
    custom-isolated-network
```

```
docker network ls
```

```
root@osboxes:/root # docker network ls
NETWORK ID          NAME                          DRIVER              SCOPE
dba0fb9370fe        bridge                        bridge              local
46d476b87cd9        customer-isolated-network     bridge              local
6de685cec1ce        docker_gwbridge               bridge              local
e29d188b4e47        host                          host                local
mmrho7vsb9zm        ingress                       overlay             swarm
d9f11695f0d6        none                          null                local
d371b4009142        simplewebappdocker_default    bridge              local
```

KODEKLOUD

Inspect Network

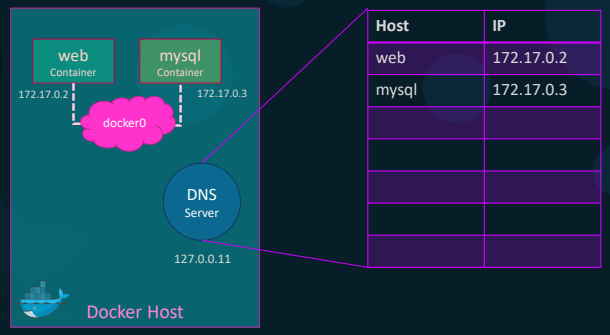
```
docker inspect blissful_hopper
```


```
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "Name": "/blissful_hopper",
        "NetworkSettings": {
            "Bridge": "",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.6",
            "MacAddress": "02:42:ac:11:00:06",
            "Networks": {
                "bridge": {
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.6",
                    "MacAddress": "02:42:ac:11:00:06",
                }
            }
        }
    }
]
```


KODEKLOUD

Embedded DNS

```
mysql.connect(  mysql      )
```



KODEKLOUD

KODEKLOUD

docker storage

KODEKLOUD

File system

- /var/lib/docker
  - aufs
  - containers
  - image
  - volumes

KODEKLOUD

Layered architecture

Dockerfile

```
FROM Ubuntu
RUN apt-get update && apt-get -y install python
RUN pip install flask flask-mysql
COPY . /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
docker build Dockerfile -t mmumshad/my-custom-app
```

Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

Dockerfile2

```
FROM Ubuntu
RUN apt-get update && apt-get -y install python
RUN pip install flask flask-mysql
COPY app2.py /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app2.py flask run
```

```
docker build Dockerfile2 -t mmumshad/my-custom-app-2
```

Layer 1. Base Ubuntu Layer	0 MB
Layer 2. Changes in apt packages	0 MB
Layer 3. Changes in pip packages	0 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

KODEKLOUD

Layered architecture

Container Layer

Read Write

Layer 6. Container Layer

```
docker run mmumshad/my-custom-app
```

Image Layers

Read Only

- Layer 5. Update Entrypoint with “flask” command
- Layer 4. Source code
- Layer 3. Changes in pip packages
- Layer 2. Changes in apt packages
- Layer 1. Base Ubuntu Layer

```
docker build Dockerfile -t mmumshad/my-custom-app
```

KODEKLOUD

COPY-ON-WRITE

Container Layer

Read Write

temp.txt

Image Layers

Read Only

app.py

KODEKLOUD

volumes

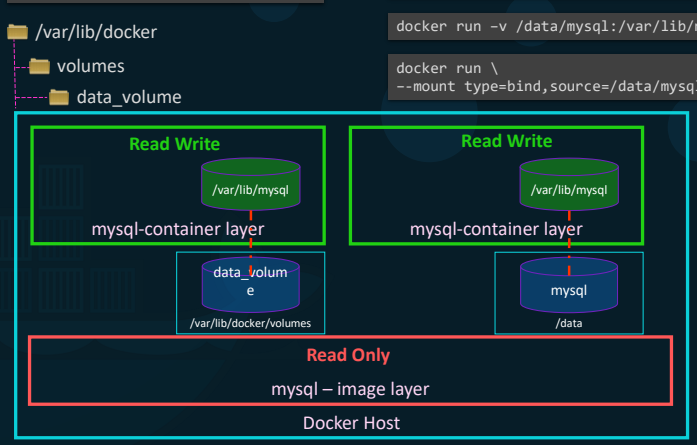
```
docker volume create data_volume
```

```
docker run -v data_volume:/var/lib/mysql mysql
```

```
docker run -v data_volume2:/var/lib/mysql mysql
```

```
docker run -v /data/mysql:/var/lib/mysql mysql
```

```
docker run \
--mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql
```



KODEKLOUD

Storage drivers

- AUFS
- ZFS
- BTRFS
- Device Mapper
- Overlay
- Overlay2

KODEKLOUD

KODEKLOUD

docker compose

KODEKLOUD

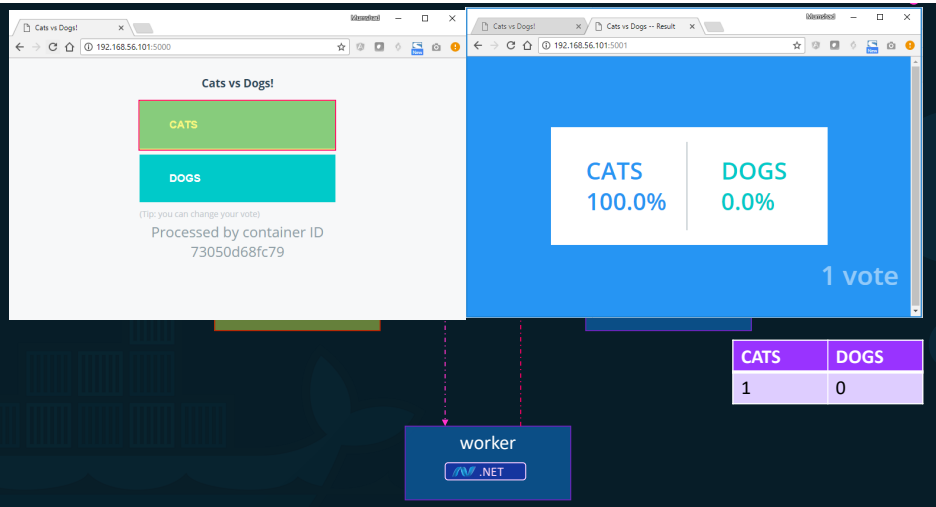
Docker compose

```
docker run mmumshad/simple-webapp
docker run mongodb
docker run redis:alpine
docker run ansible
```

docker-compose.yml

```
services:
     web:
        image: “mmumshad/simple-webapp"
     database:
        image: “mongodb“
     messaging:
        image: "redis:alpine“
     orchestration:
        image:  “ansible“
```

```
docker-compose up
```

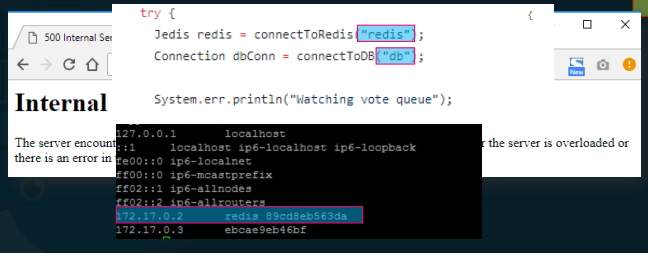


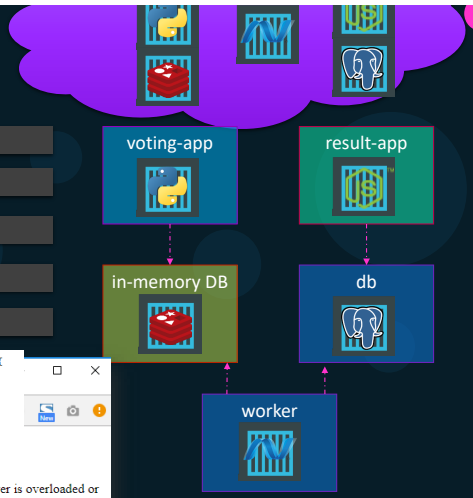
KODEKLOUD

KODEKLOUD

docker run --links

```
docker run -d --name=redis redis
docker run -d --name=db
docker run -d --name=vote -p 5000:80 --link redis:redis
docker run -d --name=result -p 5001:80 --link db:db
docker run -d --name=worker --link db:db --link redis:redis
```





Deprecation Warning

KODEKLOUD

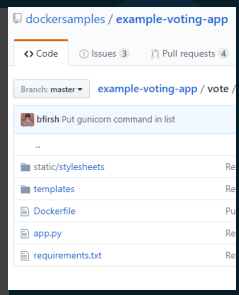
Docker compose - build

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
result:
    image: result
    ports:
        - 5001:80
    links:
        - db
worker:
    image: worker
    links:
        - db
        - redis
```

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    build: ./vote
    ports:
        - 5000:80
    links:
        - redis
result:
    build: ./result
    ports:
        - 5001:80
    links:
        - db
worker:
    build: ./worker
    links:
        - db
        - redis
```



KODEKLOUD

Docker compose - versions

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
```

version: 1

docker-compose.yml

```
version: 2
services:
    redis:
        image: redis
    db:
        image: postgres:9.4
    vote:
        image: voting-app
        ports:
            - 5000:80
        depends_on:
            - redis
```

version: 2

docker-compose.yml

```
version: 3
services:
```

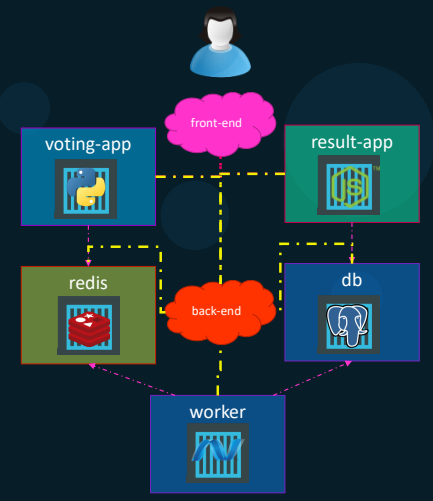
version: 3

KODEKLOUD

Docker compose

docker-compose.yml

```
version: 2
services:
    redis:
        image: redis

        networks:
            - back-end
    db:
        image: postgres:9.4
        networks:
            - back-end
    vote:
        image: voting-app
        networks:
            - front-end
            - back-end
    result:
        image: result
        networks:
            - front-end
            - back-end

networks:
    front-end:
    back-end:
```



KODEKLOUD

KODEKLOUD

docker
registry

KODEKLOUD

Image

```
docker run nginx
```

KODEKLOUD

Image

docker.io
Docker Hub

image: `docker.io/nginx/nginx`

Registry — User/ Account — Image/ Repository

`gcr.io/ kubernetes-e2e-test-images/dnsutils`

KODEKLOUD

Private Registry

```
docker login private-registry.io
```

```
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: registry-user
Password:
WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded
```

```
docker run private-registry.io/apps/internal-app
```

KODEKLOUD

Deploy Private Registry

```
docker run -d -p 5000:5000 --name registry registry:2
```

```
docker image tag my-image localhost:5000/my-image
```

```
docker push localhost:5000/my-image
```

```
docker pull localhost:5000/my-image
```

```
docker pull 192.168.56.100:5000/my-image
```

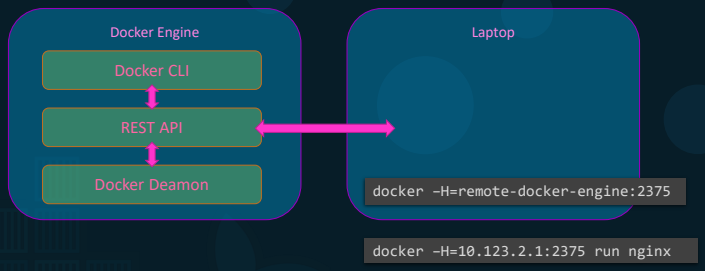
KODEKLOUD

KODEKLOUD

docker
engine

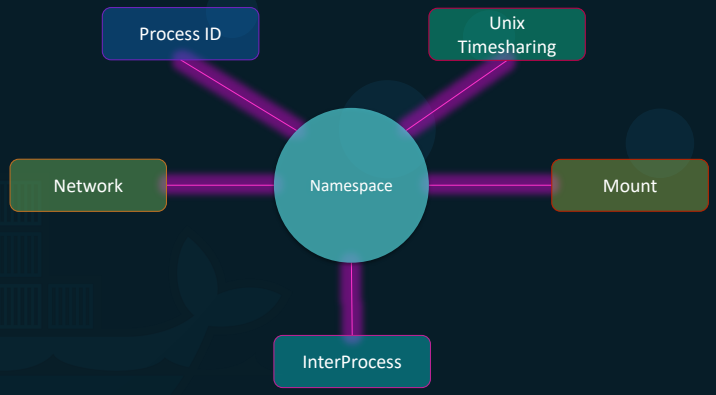
KODEKLOUD

Docker Engine



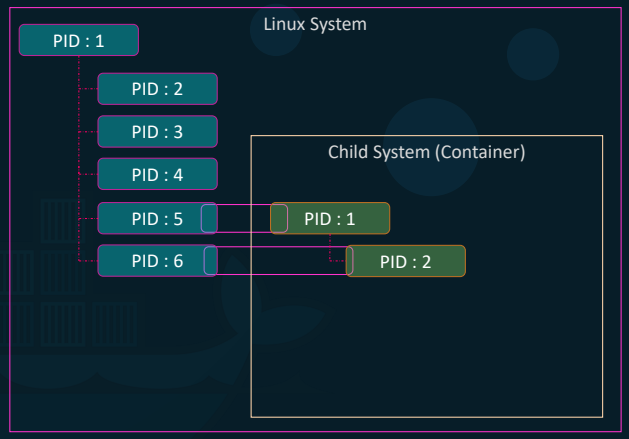
KODEKLOUD

containerization



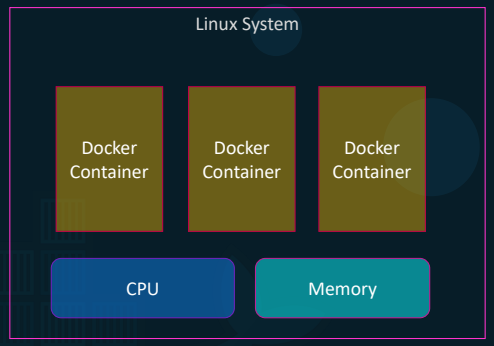
KODEKLOUD

Namespace - PID



KODEKLOUD

cgroups




```
docker run --cpus=.5 ubuntu
```

```
docker run --memory=100m ubuntu
```

KODEKLOUD

KODEKLOUD

docker On Windows

KODEKLOUD

Docker on windows

1. Docker on Windows using Docker Toolbox
2. Docker Desktop for Windows

KODEKLOUD

1. Docker toolbox

- 64-bit operating
- Windows 7 or higher.
- Virtualization is enabled

- Oracle Virtualbox
- Docker Engine
- Docker Machine
- Docker Compose
- Kitematic GUI

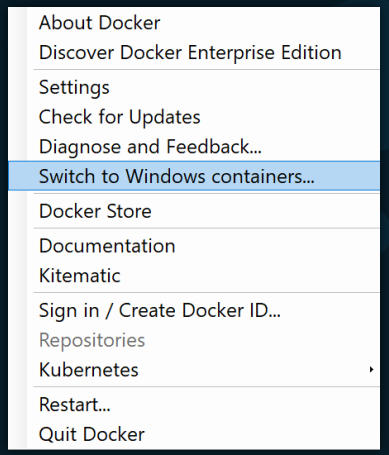
KODEKLOUD

2. Docker Desktop for Windows

Microsoft Hyper-V

Support: Windows 10 Enterprise/Professional Edition
Windows Server 2016

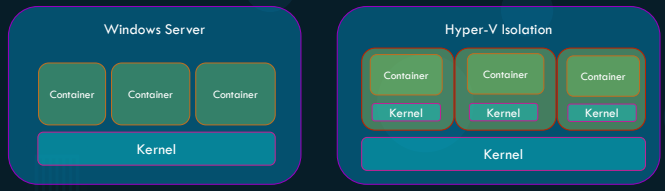
Linux Containers (Default)
Or
Windows Containers



KODEKLOUD

Windows containers

Container Types:



Base Images:

- Windows Server Core
- Nano Server

Support

- Windows Server 2016
- Nano Server
- Windows 10 Professional and Enterprise (Hyper-V Isolated Containers)

KODEKLOUD

VirtualBox Or Hyper-V

KODEKLOUD

KODEKLOUD

docker

On Mac

KODEKLOUD

Docker on Mac

1. Docker on Mac using Docker Toolbox
2. Docker Desktop for Mac

KODEKLOUD

1. Docker toolbox

- macOS 10.8 “Mountain Lion” or newer

- Oracle Virtualbox
- Docker Engine
- Docker Machine
- Docker Compose
- Kitematic GUI

KODEKLOUD

2. Docker Desktop for Mac

HyperKit

Support: macOS Sierra 10.12 or newer
Mac Hardware - 2010 model or newer

Linux Containers

KODEKLOUD

KODEKLOUD

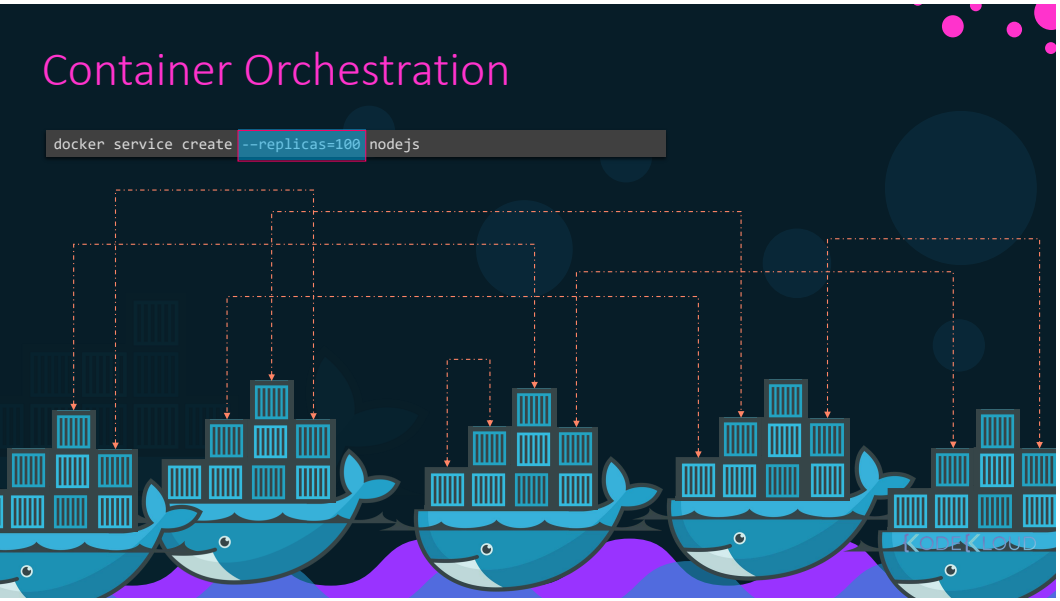
container
orchestration

KODEKLOUD

Why Orchestrate?

Container Orchestration

Container Orchestration

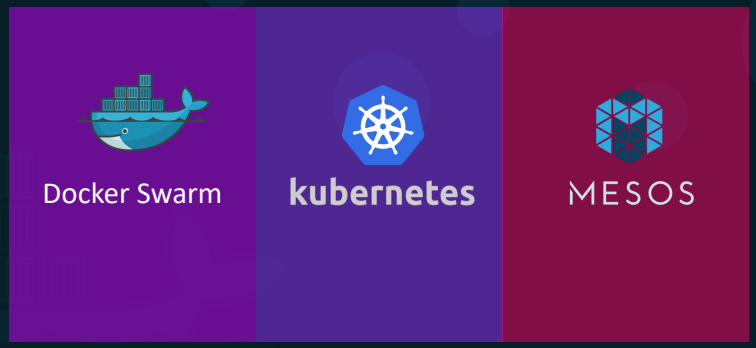


KODEKLOUD

docker swarm

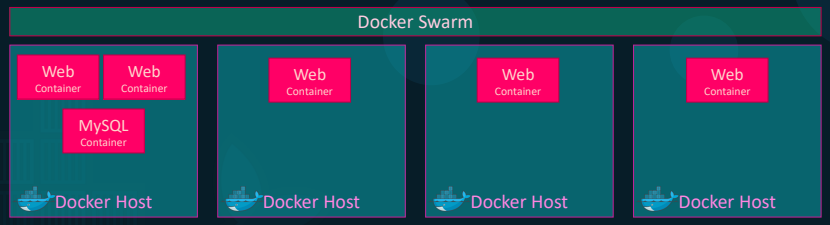
KODEKLOUD

Solutions



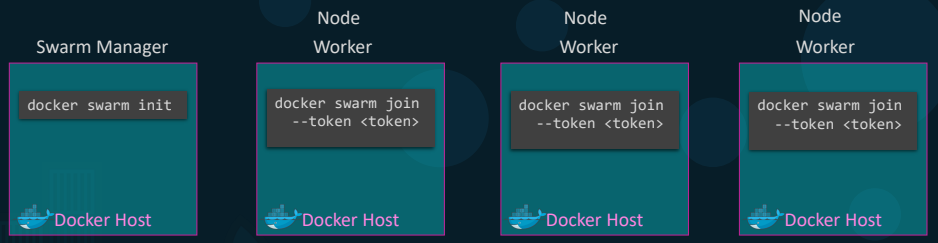
KODEKLOUD

Docker swarm



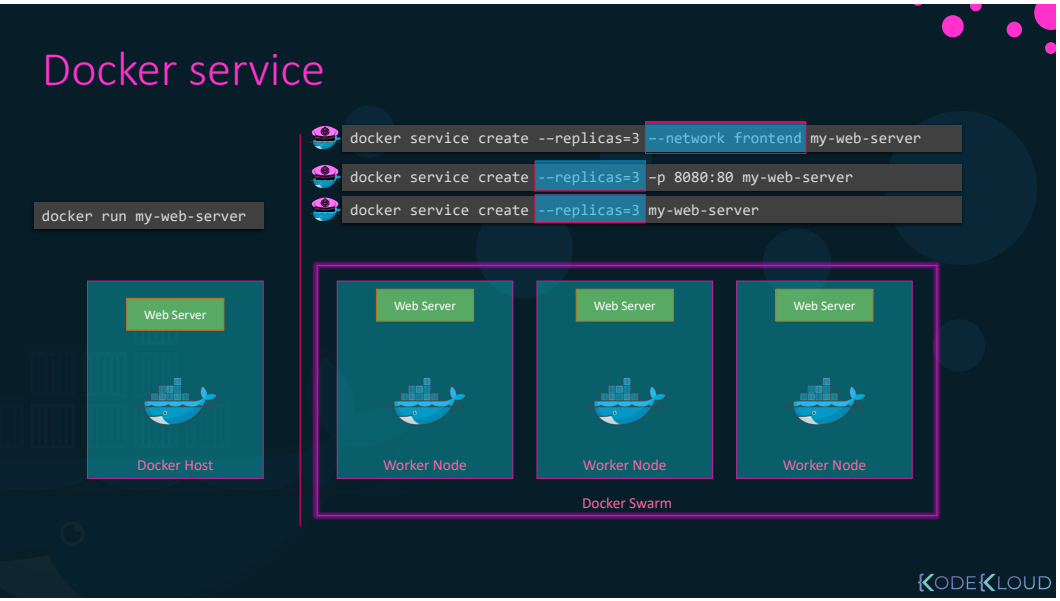
KODEKLOUD

Setup swarm



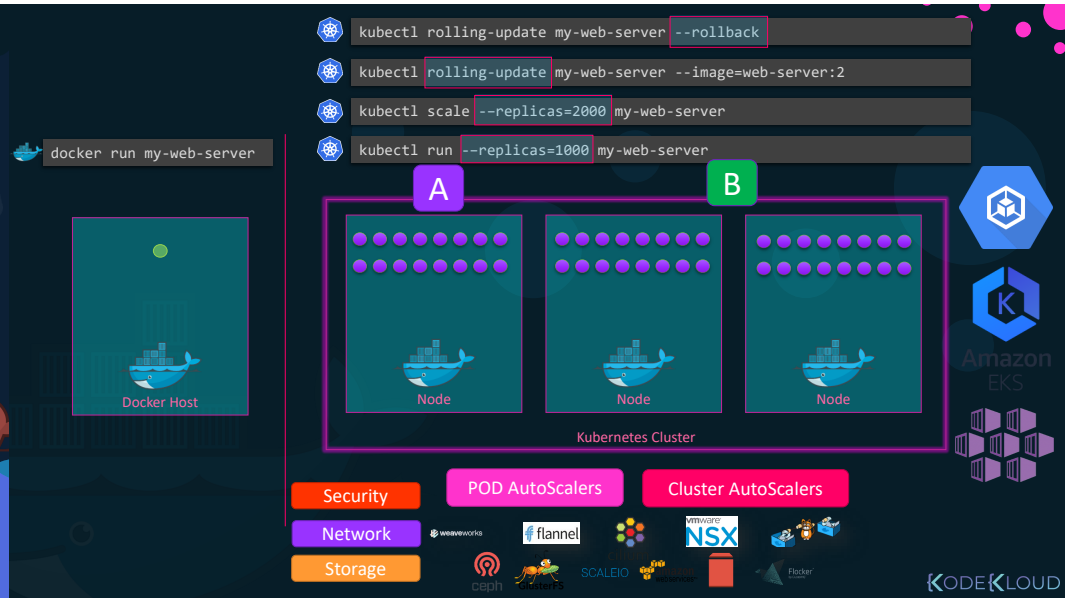

```
root@osboxes:/root/simple-webapp-docker # docker swarm init --advertise-addr 192.168.1.12
Swarm initialized: current node (0j76dum2r56p1xfne4ub1ps2c) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-35va8b3fi5krpdskefqqxgttmulw3z828daucri7y526ne0sgu-2eek9qm33d41xzoq6we9i8izp 192.168.1.12:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

KODEKLOUD

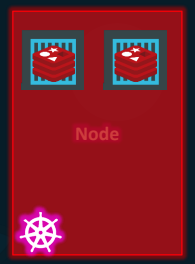

Docker service
docker service create --replicas=3 --network frontend my-web-server
docker service create --replicas=3 -p 8080:80 my-web-server
docker service create --replicas=3 my-web-server
docker run my-web-server
Web Server
Docker Host
Web Server
Worker Node
Web Server
Worker Node
Web Server
Worker Node
Docker Swarm
KODEKLOUD

KODEKLOUD


kubernetes
KODEKLOUD

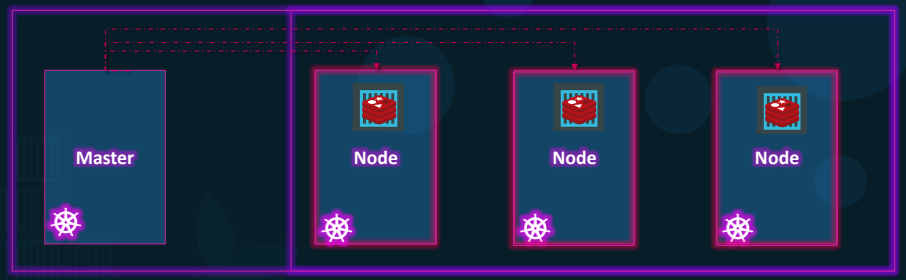

kubectl rolling-update my-web-server --rollback
kubectl rolling-update my-web-server --image=web-server:2
kubectl scale --replicas=2000 my-web-server
kubectl run --replicas=1000 my-web-server
docker run my-web-server
Docker Host
A
B
Node
Node
Node
Kubernetes Cluster
Amazon EKS
Security
Network
Storage
POD AutoScalers
Cluster AutoScalers
KODEKLOUD

Nodes (Minions)

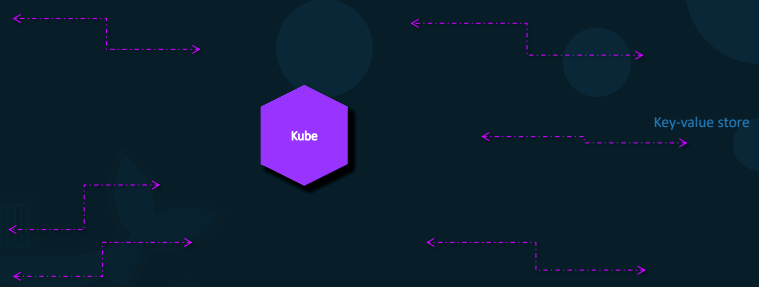


Cluster

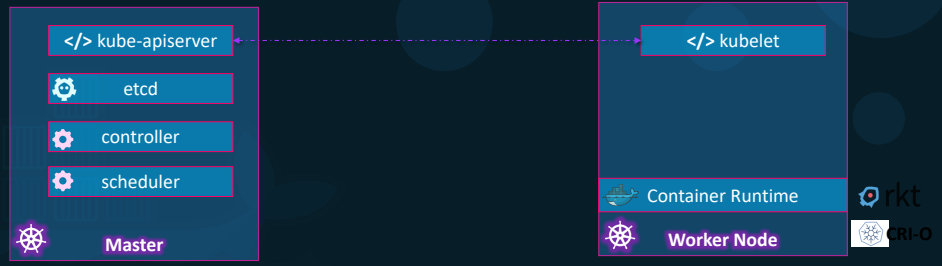
Master



Components



Master vs Worker Nodes



KODEKLOUD

kubectl

```
kubectl run hello-minikube
```

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl run my-web-app --image=my-web-app --replicas=100
```

KODEKLOUD

KODEKLOUD

CONCLUSION

KODEKLOUD

KODEKLOUD

Kubernetes for the Absolute Beginners - Hands-on

Certified Kubernetes Administrator with Practice Tests

OpenShift for the Absolute Beginners

Ansible for the Absolute Beginners

Chef for the Absolute Beginners

Puppet for the Absolute Beginners

THANK YOU

www.kodekloud.com