



Basic Linux and Docker



Introduction to Linux



Operating System (OS)



Running

Program running all the time



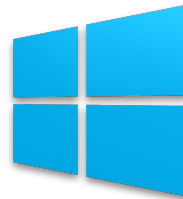
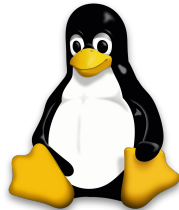
Interfaces

Interfaces between other programs and hardware

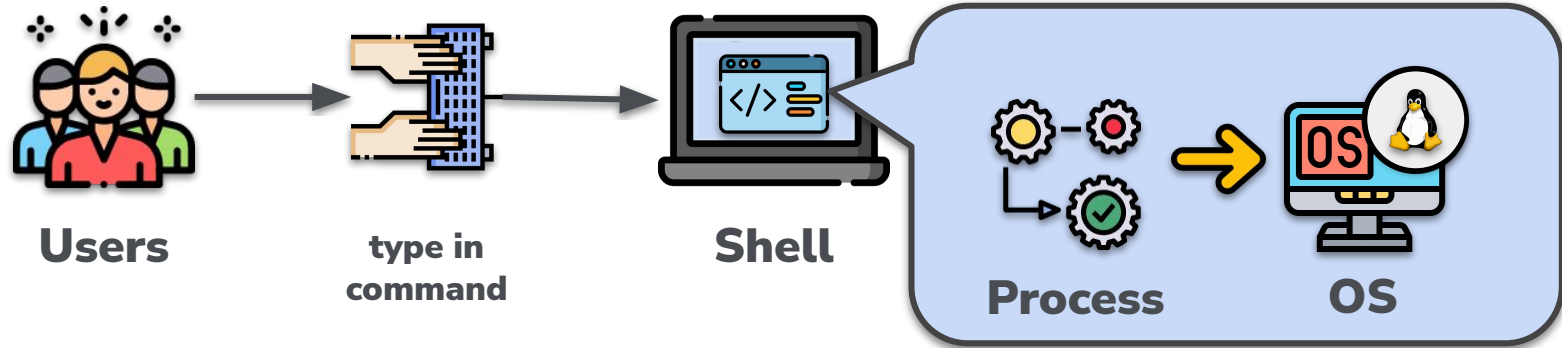


Abstractions

Provides abstractions (common interfaces, e.g. filesystems)



Kernels and Shells



- The kernel is the core of the operating system
- The shell is the interface between the user and the operating system



Basic Commands



Getting Help



```
$ man <command>
```

display user manual

```
$ man -k <command>
```

searches the given command as a
regular expression

- **\$ man** short for manual page
- **\$ man** used to display the user manual of any command that we can run on the terminal.

New Files and Directories



```
$ ls
```

Listing files and directories

```
$ mkdir <directory_name>
```

Making directories

```
$ touch <file_name>
```

Create a file without any content

File Manipulation



```
$ cp <file_name>
```

Copying Files

```
$ rm <file_name>
```

Removing Files and directories

```
$ mv <file_name> <destination>
```

Moving Files

cd

Examining File Contents



```
$ cat <file_name>
```

```
$ less <file_name>
```

Displaying the contents of a file on the screen

```
head
```

The first lines in a file

```
tail
```

The last lines in a file

Examining Files and Folders



```
$ ls <text>
```

display files and folders

```
$ ls -a <text>
```

display files and
folders with hidden
file

```
$ ls -l <text>
```

display a long listing format

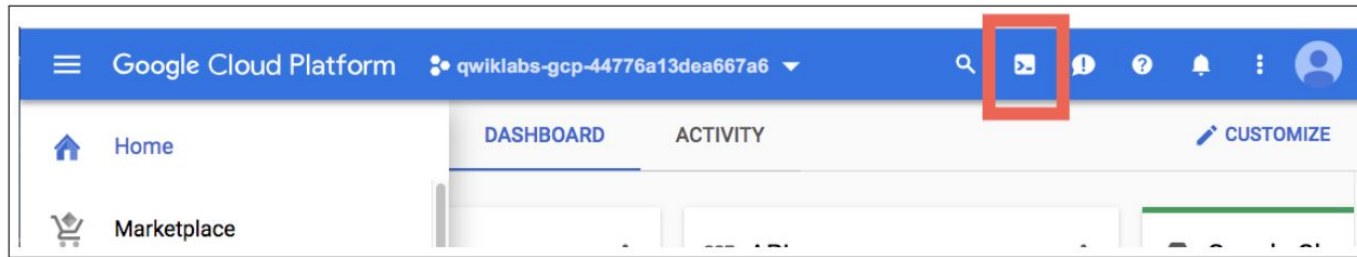
```
$ ls -s <text>
```

display file with size

```
$ ls -ltr <text>
```

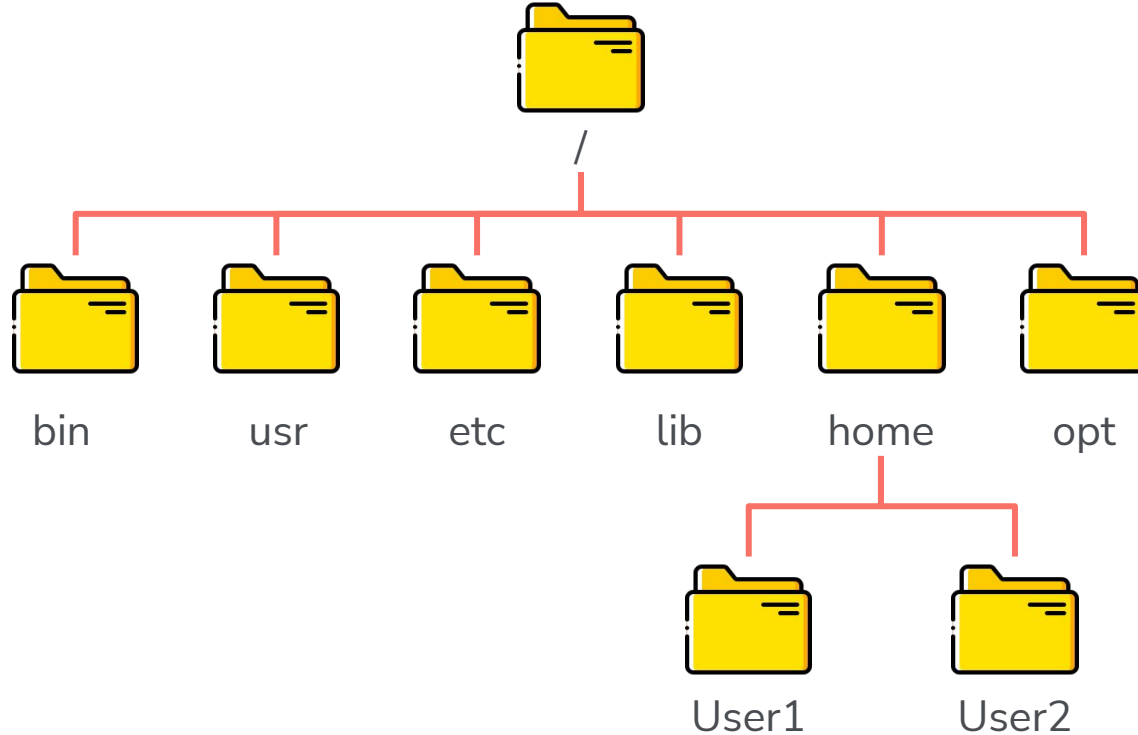
display all reverse order while
sorting and time

1. Go to Cloud Console <https://console.cloud.google.com/>



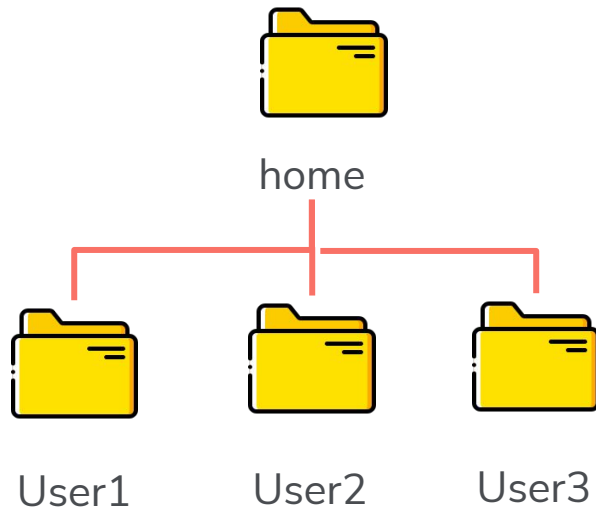
2. Check current directory
~\$ pwd

File system Hierarchy



~ mkdir

- **\$ mkdir** directory_name
Create a directory in the current location
- **\$ mkdir** {dir1,dir2,dir3,dir4}
Creates multiple directories in the current location. Do not use spaces inside {}



input:

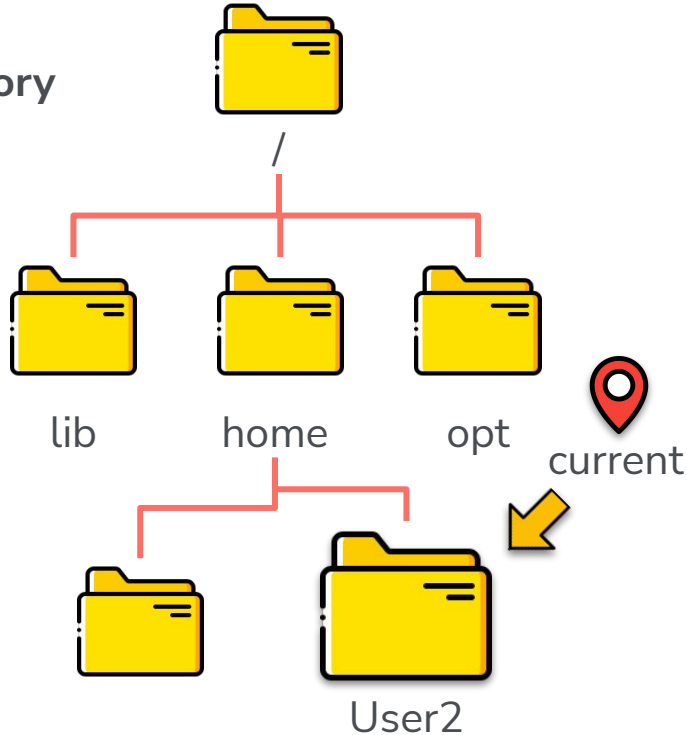
```
$ mkdir User1
```

input with {}:

```
$ mkdir {User2,User3}
```

cd command

- \$ **cd** stands for **change directory**
- \$ **cd** command: changing to a different directory



input:

```
$ cd User2
```

input:

```
$ pwd
```

output:

```
/home/User2
```

Directories Command



```
$ pwd
```

Finding where you are
with **\$ pwd** command

```
$ cd <directory_name>
```

Changing to a different
Directory with **\$ cd** command

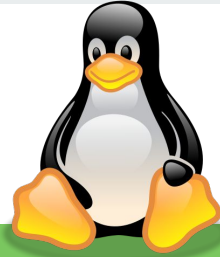
```
$ cd .
```

```
$ cd ..
```

```
$ cd ~
```

```
$ cd /
```

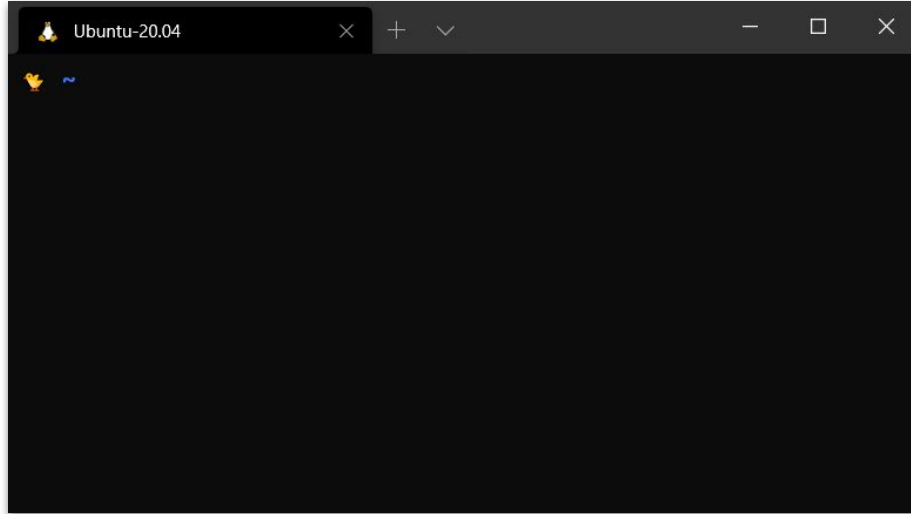
- `.` is current directory
- `..` is backward directory
- `~` is home directory
- `/` is back to root directory



Editor Commands



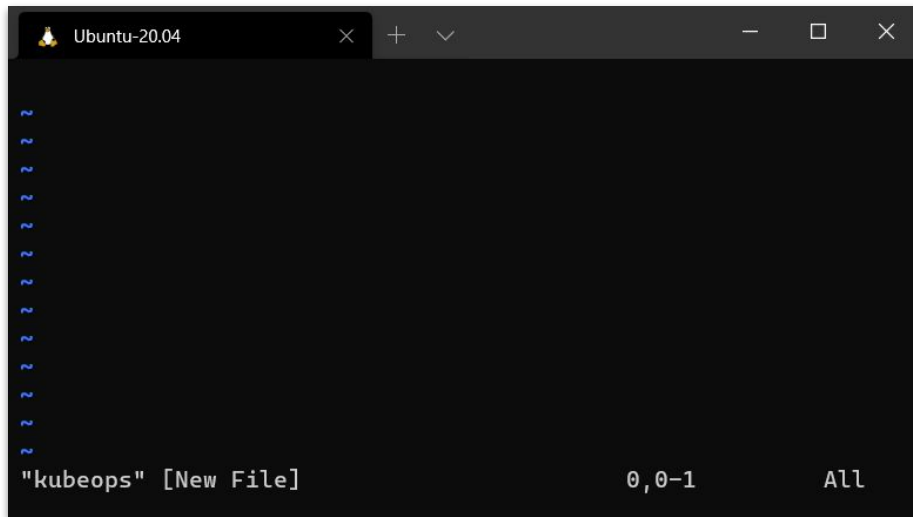
vi Editor



```
$ vi <file_name>
```

new/edit file
with vi editor

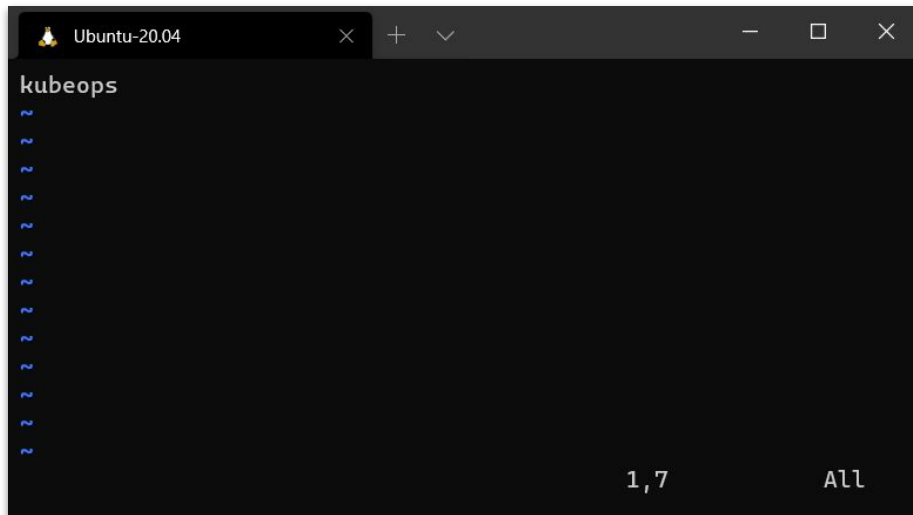
Insert Mode



press **i** to
insert mode



press **ESC** to
exit insert mode

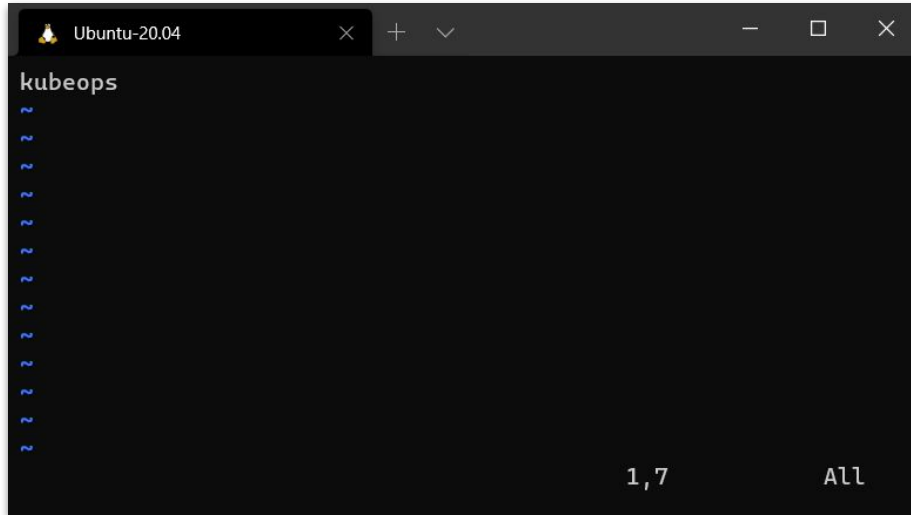


shift + c



**delete from current cursor to
the end of line, then enter to
insert mode**

Shortcut key vi Editor

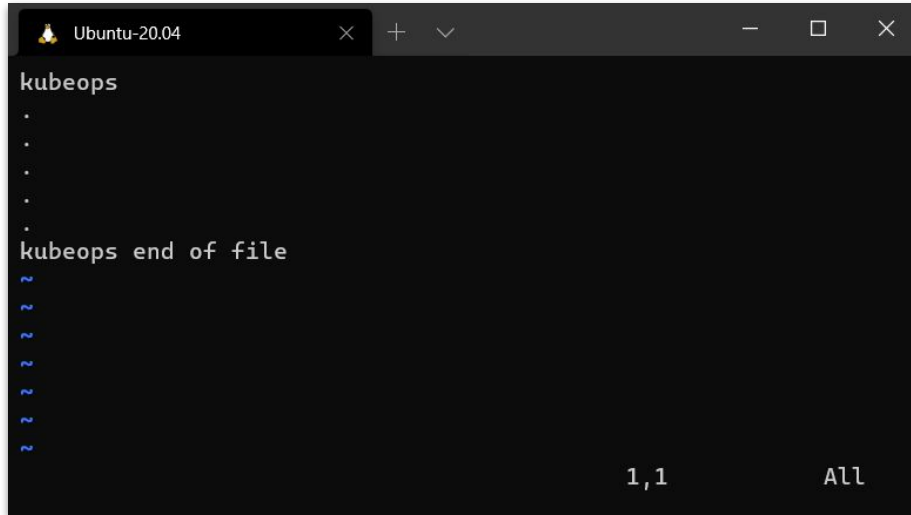


A screenshot of a terminal window titled 'Ubuntu-20.04'. The terminal shows the vi editor in insert mode, editing a file named 'kubeops'. The first line of the file contains the text 'kubeops'. The cursor is at the end of the first line, indicated by the status bar at the bottom right showing '1,7' and 'All'. The status bar also shows '1,7' and 'All'.

shift + a



move cursor to the end
of line and enter to
insert mode

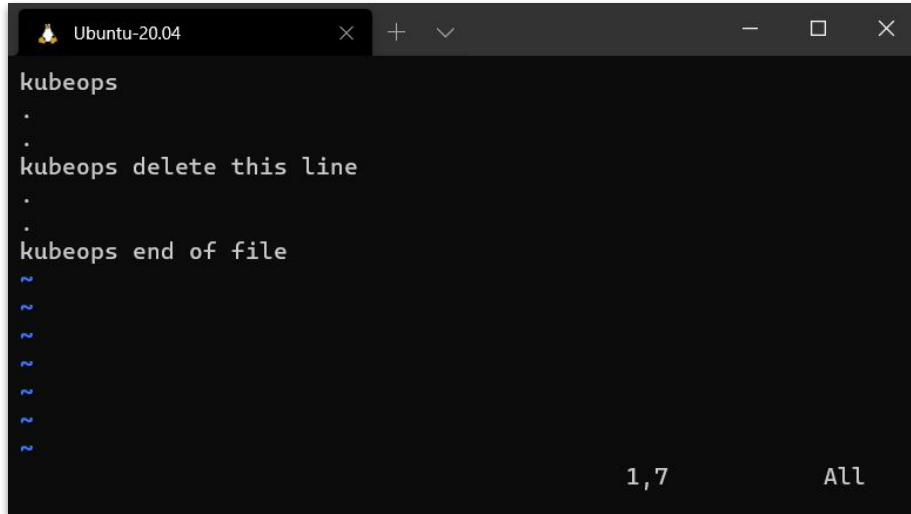


shift + g



move cursor to
the end of file

Shortcut key vi Editor

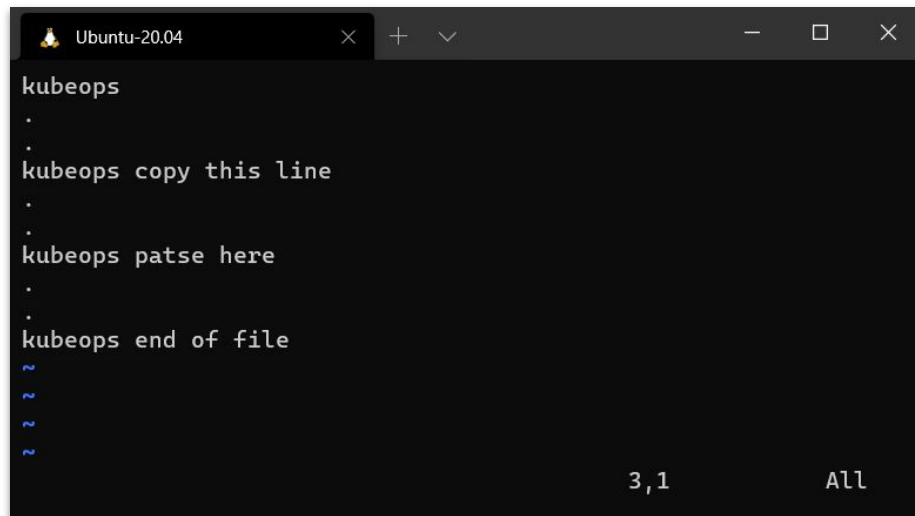


```
Ubuntu-20.04
kubeops
.
.
kubeops delete this line
.
.
kubeops end of file
~
~
~
~
~
~
~
~
~
~
1,7 All
```

dd
delete line



Shortcut key vi Editor



```
Ubuntu-20.04
kubeops
.
.
kubeops copy this line
.
.
kubeops patse here
.
.
kubeops end of file
~
~
~
~
~
~
~
~
3,1  ALL
```

yy

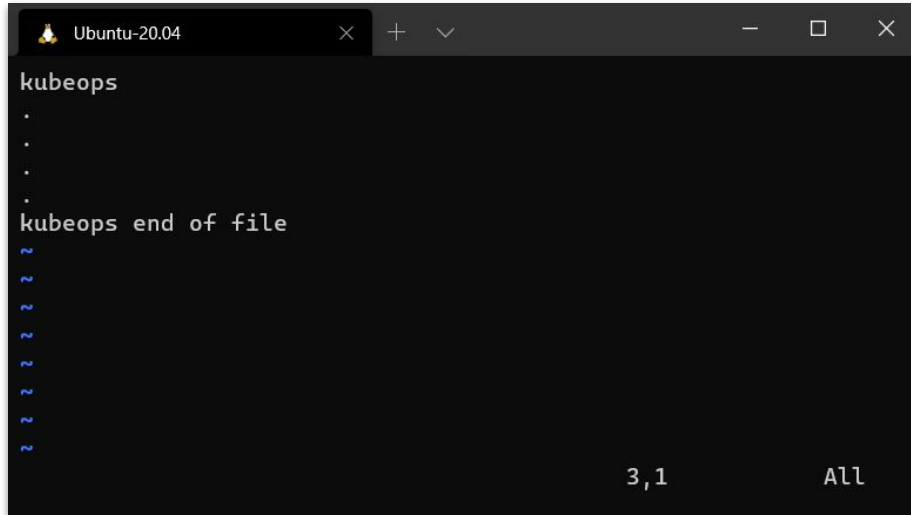
copy line



p

paste from
clipboard



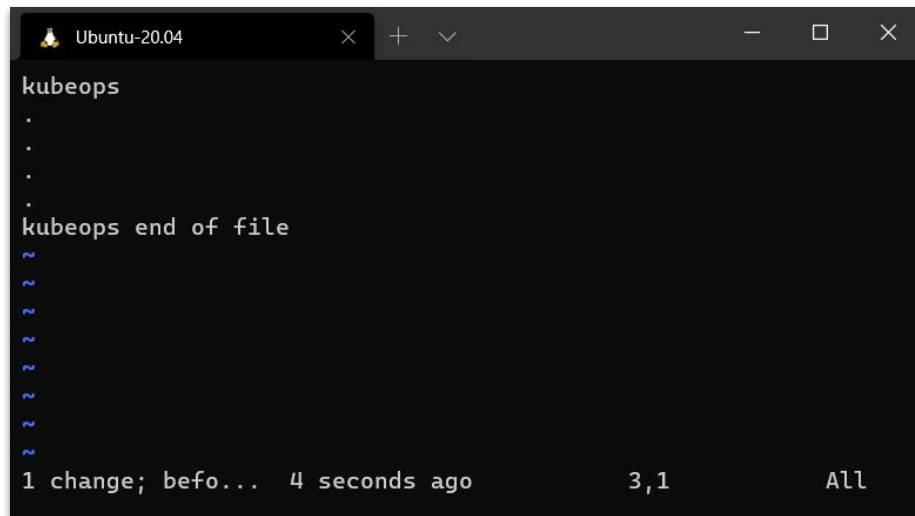


u

undo



Save vi Editor

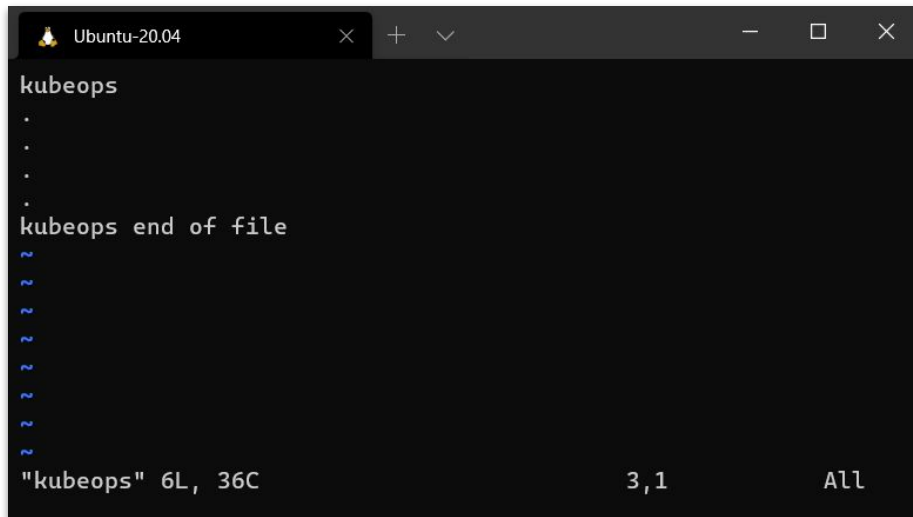


```
Ubuntu-20.04
kubeops
.
.
.
.
.
kubeops end of file
~
~
~
~
~
~
~
~
~
~
1 change; befo... 4 seconds ago    3,1    All
```

`:wq`



save and quit



:q



quit without save

:q!



force quit

Lab 1

1. Run command `history 200 > linux_[student_id].txt`
2. Download file to local and submit

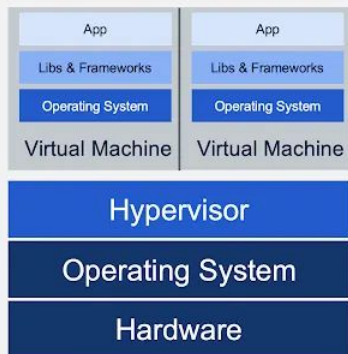


Basic Docker

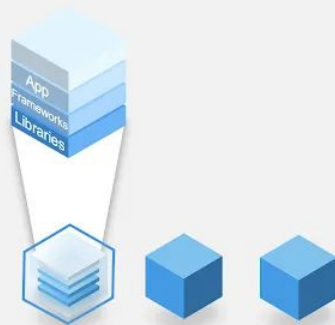




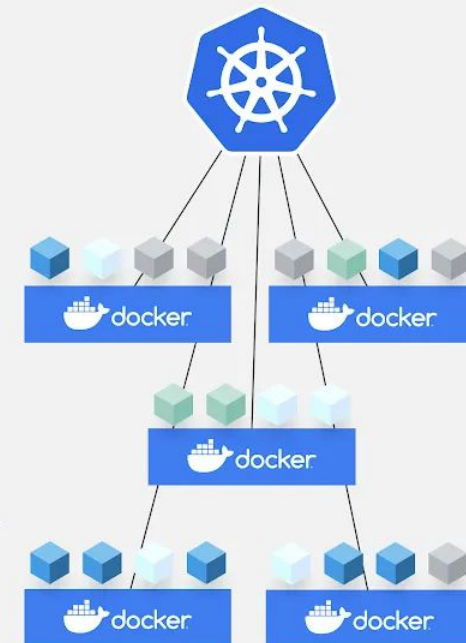
**Traditional
Deployment**



**Virtualized
Deployment**



**Container
Deployment**



**Kubernetes
Deployment**

**Kubernetes & Docker work
together to build & run
containerized applications**

Where should I run my stuff?



#GCPSketchnotes

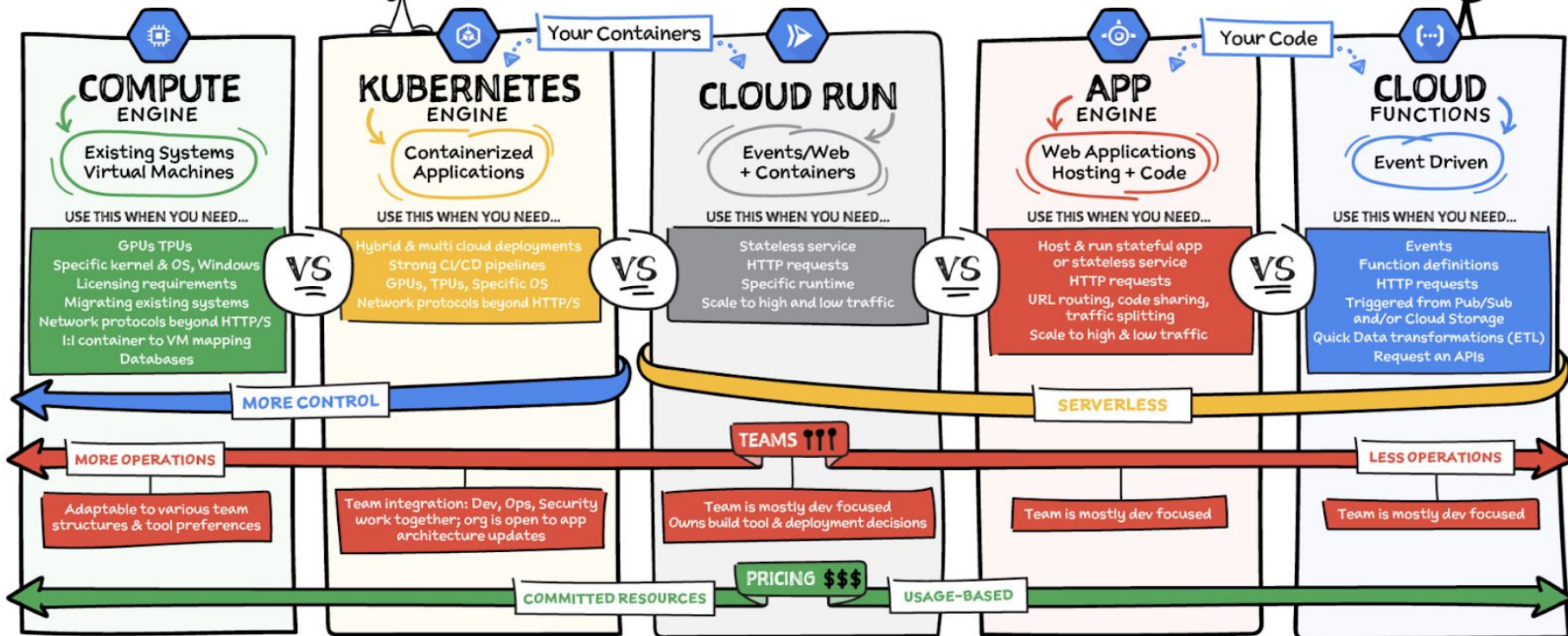
@PVERGADIA THECLOUDGIRL.DEV

Where should I run my stuff?

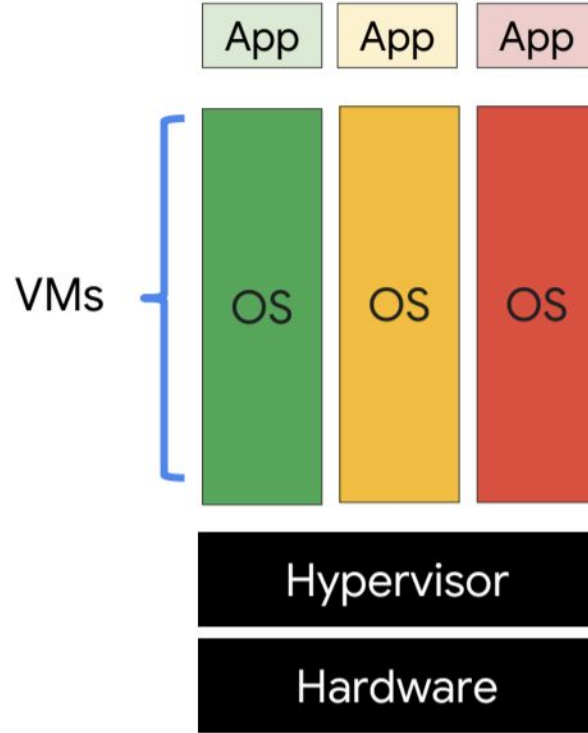
IT DEPENDS...



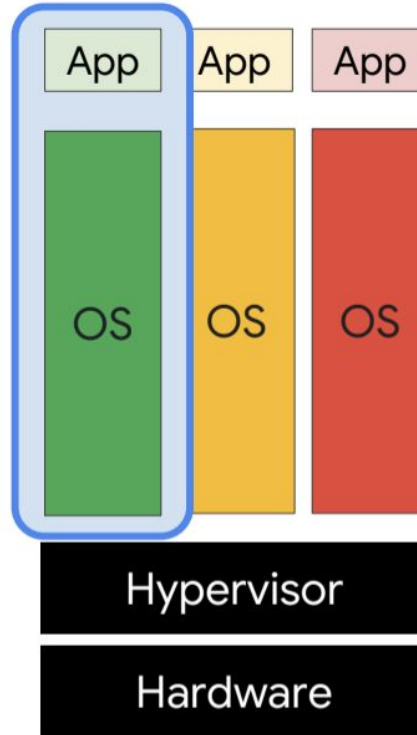
PRO TIP: YOU CAN USE THEM TOGETHER



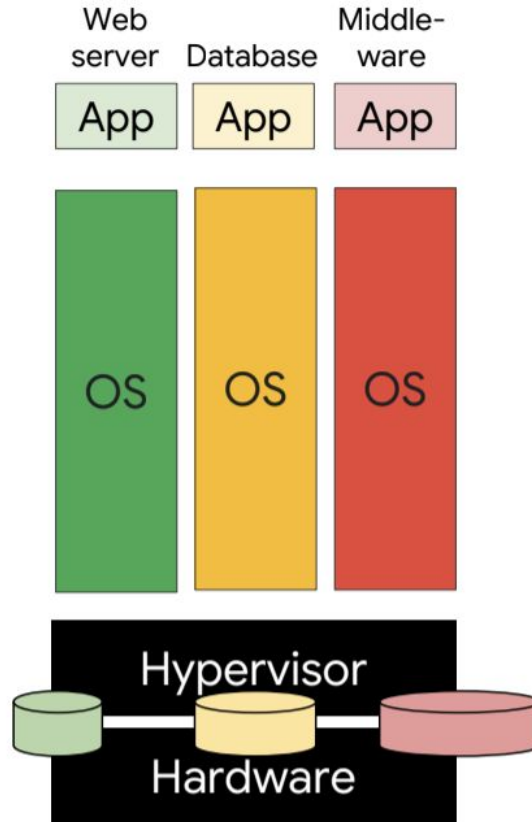
IaaS



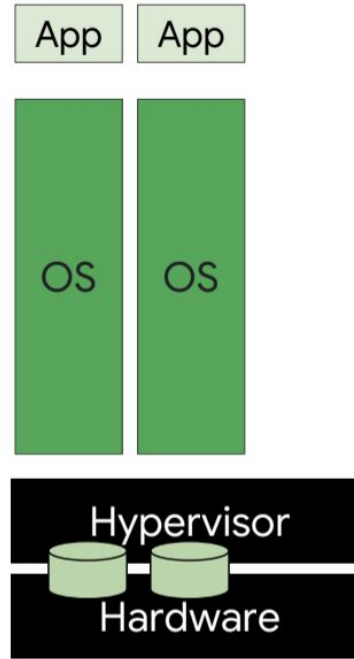
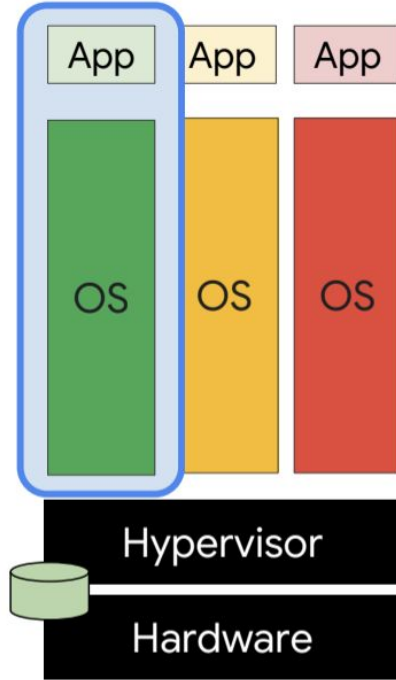
IaaS



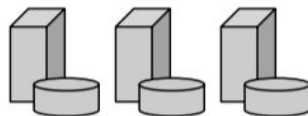
IaaS



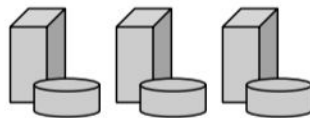
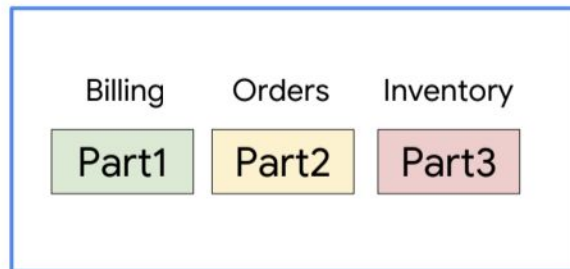
IaaS



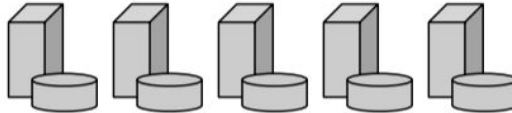
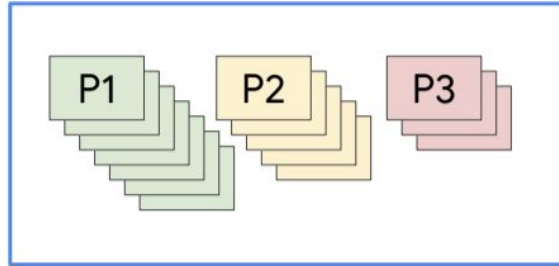
App Engine



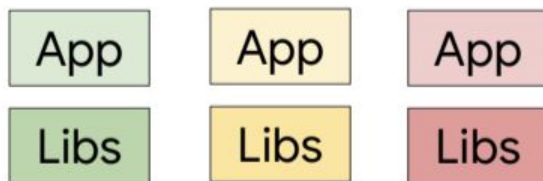
App Engine



App Engine

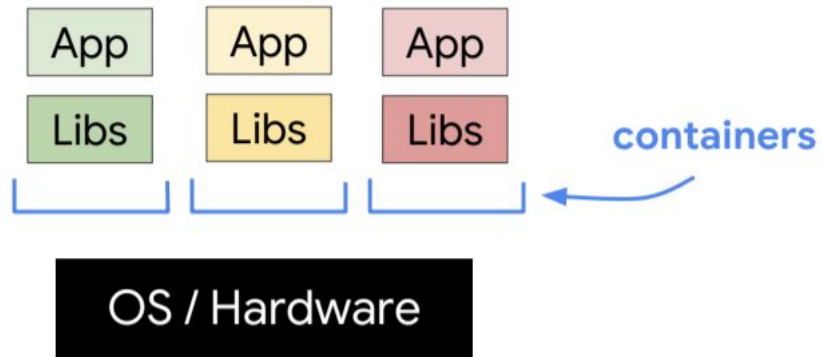


Containers

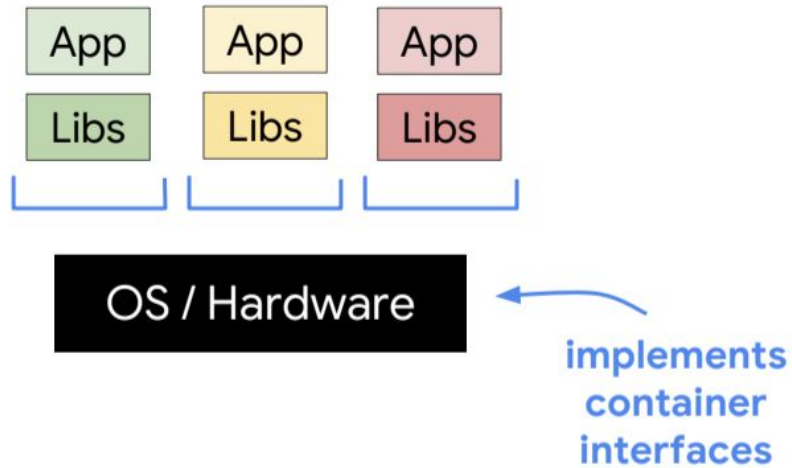


OS / Hardware

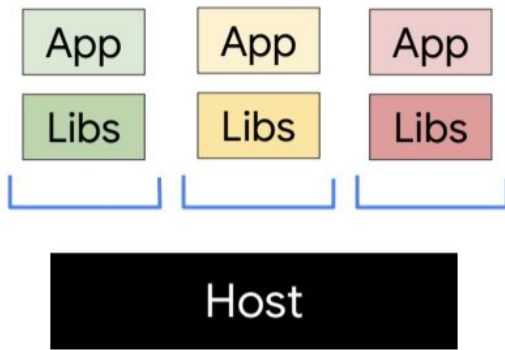
Containers



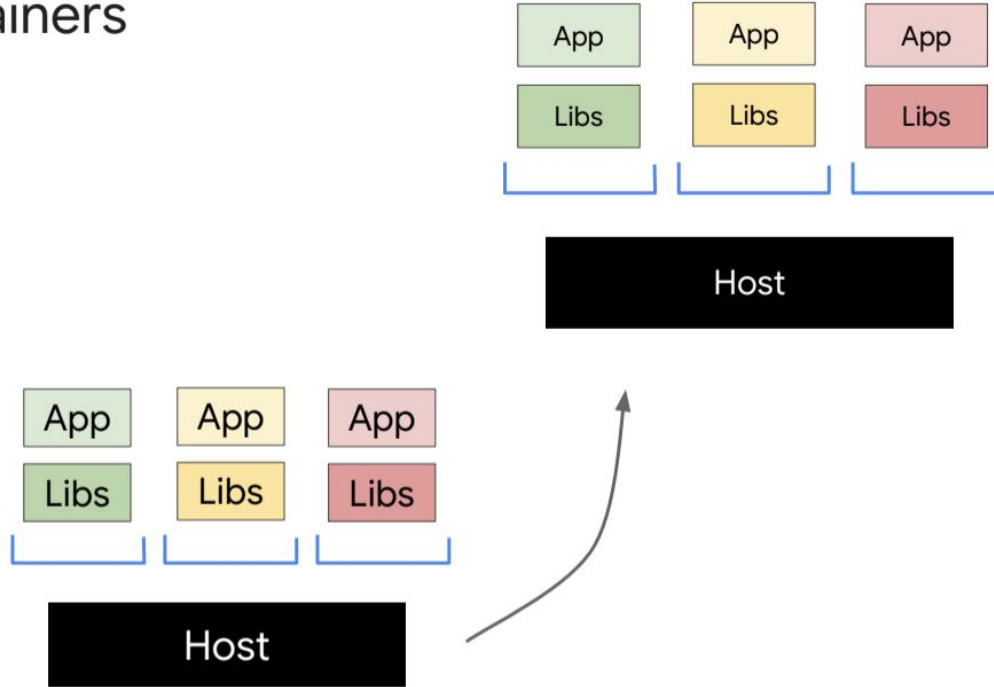
Containers



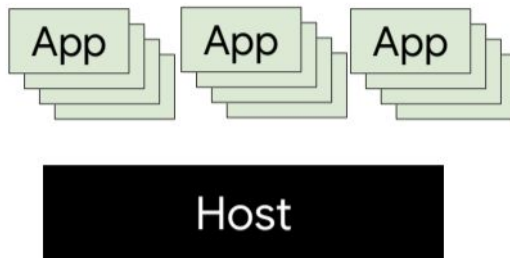
Containers



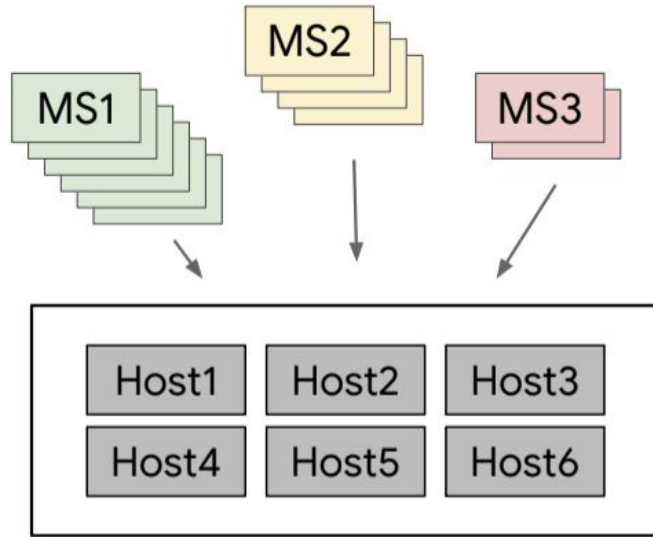
Containers



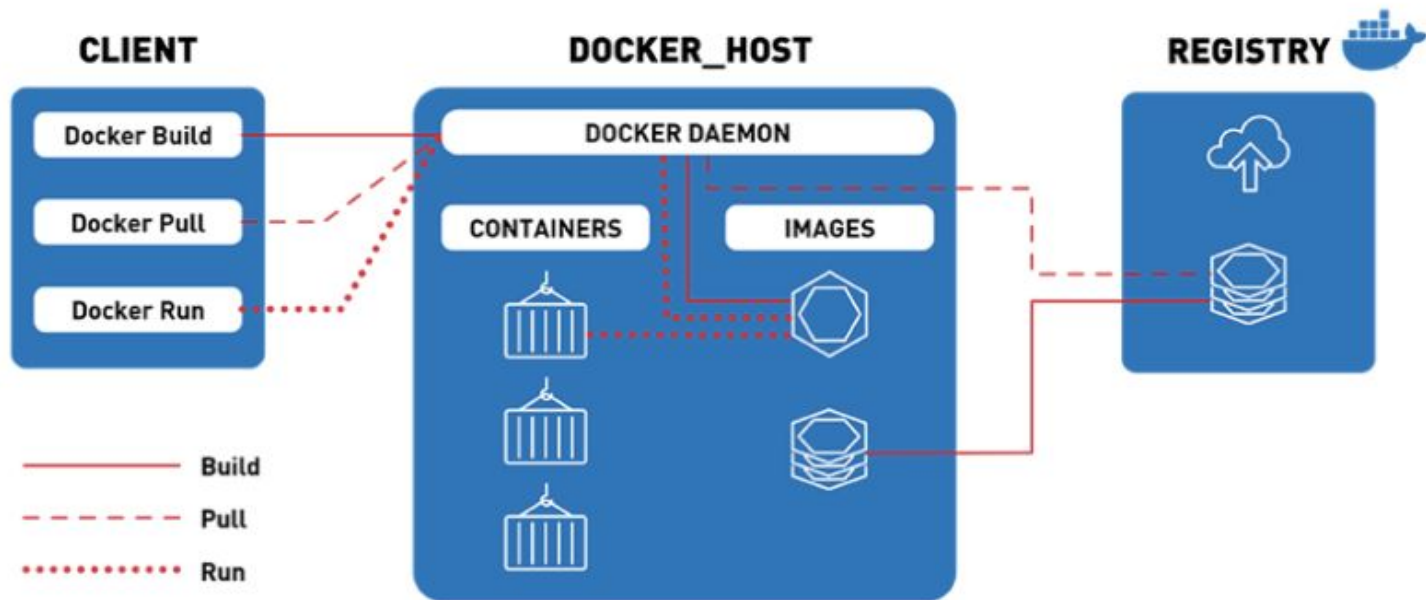
Containers



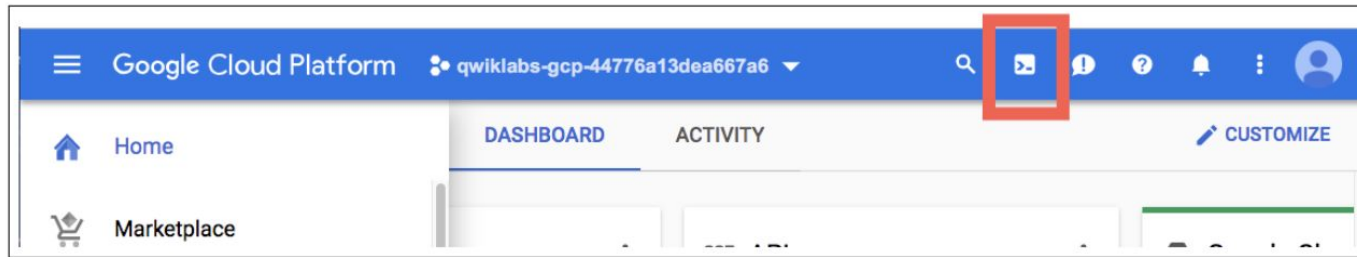
Containers



Docker Architecture and Process



1. Go to Cloud Console <https://console.cloud.google.com/>



2. Check current directory
~\$ pwd

Create file app.py

app.py

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!\n"

@app.route("/version")
def version():
    return "Helloworld 1.0\n"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```



App.py

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!\n"

@app.route("/version")
def version():
    return "Helloworld 1.0\n"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!\n"

@app.route("/version")
def version():
    return "Helloworld 1.0\n"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```



App.py

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!\n"

@app.route("/version")
def version():
    return "Helloworld 1.0\n"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!\n"

@app.route("/version")
def version():
    return "Helloworld 1.0\n"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```



Create file requirements.txt

requirements.txt

```
Flask==0.12  
uwsgi==2.0.15
```



apt-get command

apt-get is a command-line tool which helps in handling packages in Linux. Its main task is to retrieve the information and packages from the authenticated sources for installation, upgrade and removal of packages along with their dependencies. Here APT stands for the *Advanced Packaging Tool*.

```
apt-get [options] command
```

or

```
apt-get [options] install|remove pkg1 [pkg2 ...]
```

or

```
apt-get [options] source pkg1 [pkg2 ...]
```

Exercise:

1. RUN :

```
apt-get update -y &&\n    apt-get install -y python3-pip python3-dev
```

2. RUN:

```
pip3 install -r requirements.txt
```

Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENDPOINT ["python3", "app.py"]
```



Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENDPOINT ["python3", "app.py"]
```

Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENDPOINT ["python3", "app.py"]
```

Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENTRYPOINT ["python3", "app.py"]
```

Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENTRYPOINT ["python3", "app.py"]
```


Dockerfile

```
FROM ubuntu:18.10
RUN apt-get update -y && \
    apt-get install -y python3-pip python3-dev
COPY requirements.txt /app/requirements.txt
WORKDIR /app
RUN pip3 install -r requirements.txt
COPY . /app
ENDPOINT ["python3", "app.py"]
```

Build and run

```
$> docker build -t py-server .  
$> docker run -d py-server
```

Docker play with Docker

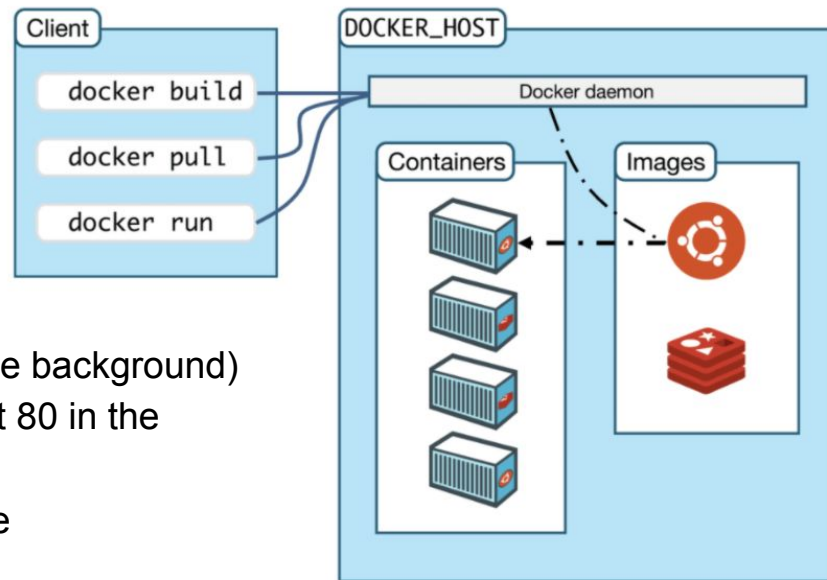
3. Check Docker version with this command :

```
~$ docker --version
```

4. Run Docker container with this command :

```
~$ docker run -dp 8080:80 docker/getting-started
```

- -d - run the container in detached mode (in the background)
- -p 8080:80 - map port 8080 of the host to port 80 in the container
- dockersamples/101-tutorial - the image to use



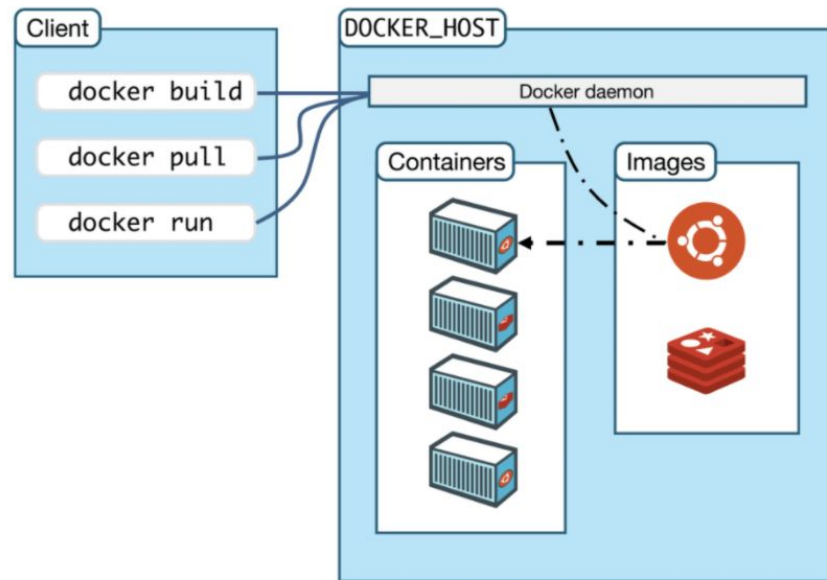
Docker play with Docker

5. Check Docker container with this command :

```
~$ docker ps
```

6. Check Docker images with this command :

```
~$ docker images
```



Lab 2

1. Run command `history 100 > docker_[student_id].txt`
2. Download file to local and submit