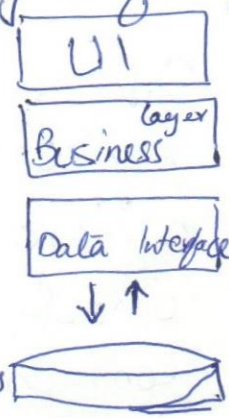
CARONA 1
COVID 19

15-MAY-2020
11:04 AM

CNC
CLOUD NATIVE COMPUTING

INTRODUCTION By Aamir Pinger

- MONOLITHIC ARCHITECTURE :-
  - A unified model for the design of software.
  - Component/layer of the program are interconnected and interdependent.
  - Tightly coupled architecture
  - Application is too large and complex to fully understand and made changes fast and correctly.
  - Redeploy on each update.
  - Size of the app can slow down the start up time.



- MICRO SERVICE :-
  - The micro service architecture is an approach to develop a single application as a suite of small service.
  - Better Organization.
  - Decoupled
  - Performance
- CLOUD
  - Private
  - Public
  - Hybrid

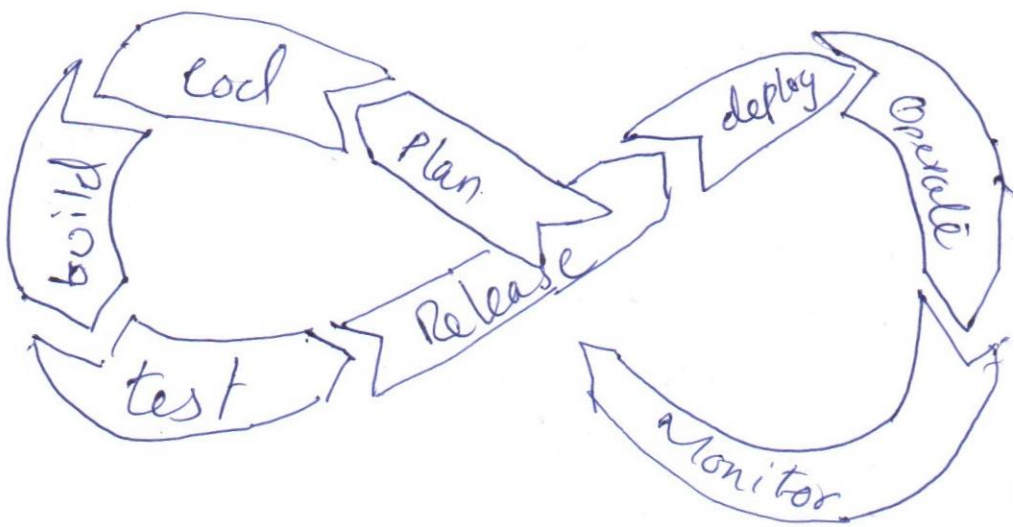
- CLOUD NATIVE COMPUTING FOUNDATION

An approach that build software application as microservices and runs them on a containerized and dynamically orchestrated platform to utilize the advantages of the cloud computing model.

- Devops.
- Agile methodology.
- Micro service.
- Cloud computing platforms.
- Containerize application
- Orchestration system.
- Continuous delivery. (CICD Continuous integration continuous development)

- DEVOPS :-

- Plan
- Code
- build
- test
- Release
- deploy
- Operate
- Monitor.



AGILE DEVELOPMENT

Iterative and incremental means we build then deploy and same repeat in incremental fashion.

we get feedback to improve the quality of product.

- Initiate project.
- review
- feedback

- Release to market (yes/No)

- Record & Incorporate changes
- Adjust & track.
- Next Iteration.

CONTAINER :-

- A runtime instance of an image.
- An Image is an executable package that include everything to run an application.

- the code
- a runtime
- libraries.
- environment variables
- and configuration file.

- Flexable
- lightweight
- Interchangable.
- Portable.R
- Scalablee
- Stakable:

ORCHESTRATION (KUBERNETES) :-

- Scaling app based on the current load of your system isn't that easy.
- Monitor your system.
- trigger the startup or shutdown of a container
- Make sure that all required configuration are in place.
- balance the load b/w the active application instances.
- share authentication secrets b/w your containers

CICD :- Continuous Integration & Continuous deployment / Continuous delivery.

- CI : is an automation process for developer success.

CI means new code changes to an app are regularly build, tested and merged to a shared repository.

- Continuous Delivery : changes to an app is automatically bug tested and uploaded to a repository (github or a container registry) where they can be deployed.

DOCKER

27-MAY-2020
12:05 PM

PREVIOUSLY

- One app for one server.
- Unable to judge resource requirement.
- Different architecture and dependencies like Linux & Window

DISADV.

1) Very costly.
2) Resource wastage.
3) Many servers to manage

VM WARE:

based on virtualization

ADV:

- multiple app on single server.
- Diff. OS and dependencies on same server using VM.
- Much better old one.
- Save lot of resources.

DISADV:

- OS consumes lots of resources
- licensing cost of every OS instance.

CONTAINERS

- To package an app so it can be with its dependencies, isolated from other processes.

- For major difference
  - Single OS
  - less hardware resource.
  - Reduced license fee.
  - Portable and fast.

TYPES OF CONTAINER

1) LINUX
2) WINDOWS.
   - These container designed to run on a window kernel will not run on linux host.
   - But linux containers can run windows machine

IS DOCKER A CONTAINER

NO

- Docker is a software that run on linux & window
  - Enterprise Edition
  - Community

CONTAINER ECOSYSTEM :-

Batteries are removable means you can swap out lot of the docker stuff and replace it with stuff from 3rd parties.

like networking stack

OCI (OPEN CONTAINER INITIATIVE)
for standardizing the most fundamental comp.

- The image -spec.
- The runtime spec.

DOCKER INSTALLATION :-

- Update the package index

  # sudo apt-get update

- Install latest version of Docker and containerd

  # sudo apt-get install docker-ce docker-ce-cli containerd.io.

- To check docker is installed

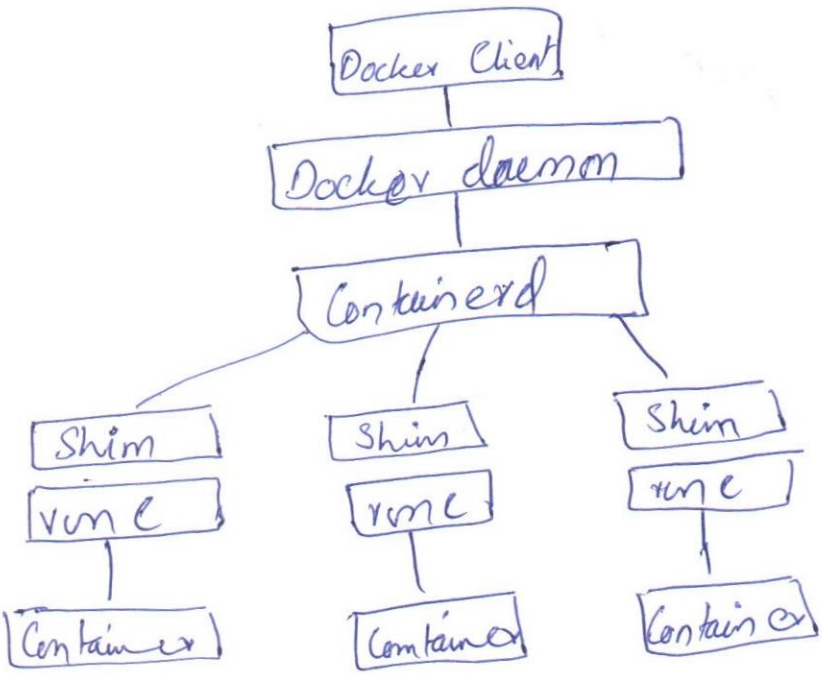
  # docker --version.

#docker info

Docker info is to show no. of containers, running, paused, stopped, images, version etc.

DOCKER ENGINE :-

is the core software that run and manage containers.



Docker Daemon

it listen for Docker API requests and manages Docker objects such as images, containers

CONTAINER D

act as a bridge b/w daemon and runc

- Starting and stopping container
- Pausing and unpausing.
- Destroying the container

RUNC

- runc often refer as a container runtime
- It has a single purpose in life is to create container

SHIM

- The shim is integral to the implementation of daemon containers.
- Maintainance and upgrades on the docker without impacting running container.

* IMAGE

- It is a lightweight standalone, executable package of software that includes everything needed to run an application.
  - Code
  - Runtime
  - System tools
  - System libraries
  - Settings
- Image become containers when they run on Docker engine.

⊙ Images are made up of multiple layers that get stacked on top of each other and represented as a single object.

⊙ Inside of the image a cut down (OS) and all the files and dependencies required to run an application.

⊙ Common layers among different images are downloaded only once and are stored only once and get re-use in all images.

⊙ Once a container is up and running made from an image, they are dependent on each other. and you cannot delete the image until the last container using it has been stopped and destroyed.

- fast & lightweight.

IMAGE REGISTRIES

- Docker images are stored in Image registry
- Docker Hub. (Room as Registry)
  - Official
  - Unofficial

* IMAGE NAMING & TAGGING

docker image pull nginx:latest

From unofficial repository is almost

docker image pull aamirpinger/helloworld :latest.

Note:
latest tag does not mean that the image is latest but with the tag latest.

docker pull aamirpinger/helloworld

* IMAGE LISTING :-

docker image ls

or

docker images

* IMAGE REMOVING :-

docker image rm alpine:latest

* RUNNING A CONTAINER :-

docker run -it aamirpinger/helloworld sh

- -it for interactive mode.
- sh/bash for shell.

docker container ls

* To list all the containers

docker container ls -a

Ctrl + P + Q to quit from container without stopping it.

* To list all running container

docker container ps.

* To list all container (state)

docker ps -a

* To come into interactive mode of any running container.

docker exec -it container_id or name

* STOPPING A CONTAINER :-

docker container stop container_id or name

* STARTING A CONTAINER :-

docker container start container_id or name

* REMOVING A CONTAINER :-

docker container rm container id or name

* CONTAINER IN DETACHED MODE

docker container run -d aamirpinger/flag

* PUBLISHING PORT :-

docker container run -d -p 5010:80 aamirpinger/helloworld:latest

* CUSTOM NAME :-

docker container run -d --name docker-app -p 5020:80 aamirpinger/flag

* CONTAINERIZING AN APP :-

To make a container from a image which we called Containerization or Dockerizing

- Start with your application code.
- Create a dockerfile that describe your app, its dependencies and how to run it.
- feed this dockerfile with docker build command to create an image.

DOCKER FILE :-

- Example 1 :-

```
FROM nginx
COPY . /usr/share/nginx/html
```

- Example 2 :-

```
FROM ALPINE
LABEL maintainer= "email address"
RUN apk add --update nodejs nodejs-npm

COPY . /src

WORKDIR /src
RUN npm install
ENV CREATED BY = "Aamir Pinger"
EXPOSE 8080
ENTRYPOINT ["node", "./app.js"]
```

BUILDING AN CONTAINER IMAGE

docker build -t node-app-image .

docker container run --name=first-node-cont
-d -p 8350:8080 node-app-image

* PUSHING IMAGES :-

docker push ~~node~~ - aamirpinger/node-app-image

* TAG

docker tag node-app-image aamirpinger/node-app-image

* DOCKER HISTORY & INSPECT :-

docker history container name (node-app-js)

docker inspect node-app-js

* BIND MOUNT :-

docker container run -it --name=test-app
-v source_location:destination image_name sh