Assalam-u-alaikum

BILAL KHAN

This is my 21st video of DevOps

Today we’re going to talk about Kubernetes.

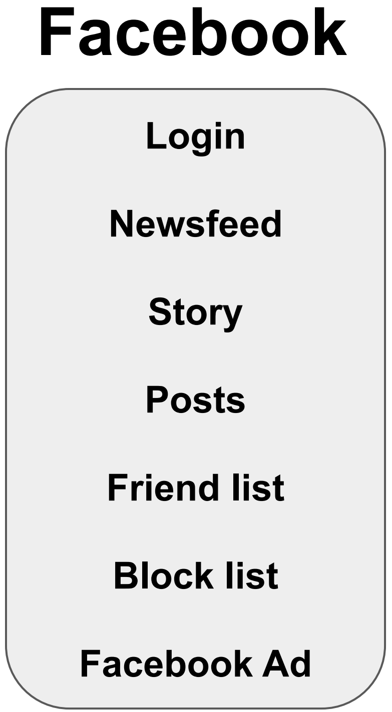
Previously the project structure was monolithic.

Monolithic architecture: It’s an architecture in which multiple parts of an applications are bundled together.

You can deploy it as a single application without dividing it in multiple parts.

Disadvantages:

- If you made changes in Login, it may affect other parts of an application.
- After making changes in one part of an application, you have to deploy the whole application again to the server.



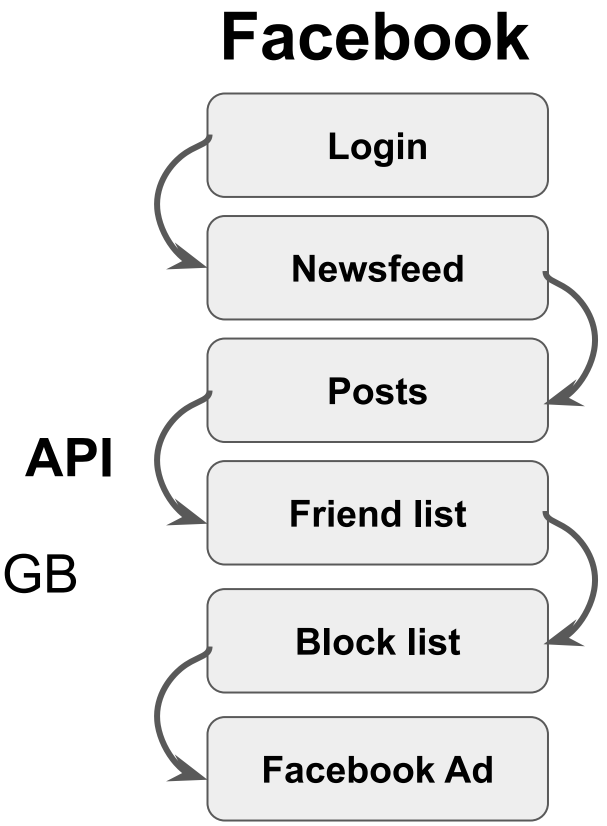
Microservices

Instead of deploying the whole application as a single component, it will divide an application into multiple components.

Each of the component will work independently and by making changes in one component, it will not affect another one.

APIs are used to communicate b/w each other. These components are loosely coupled. It means that they can work independently and they can communicate also.

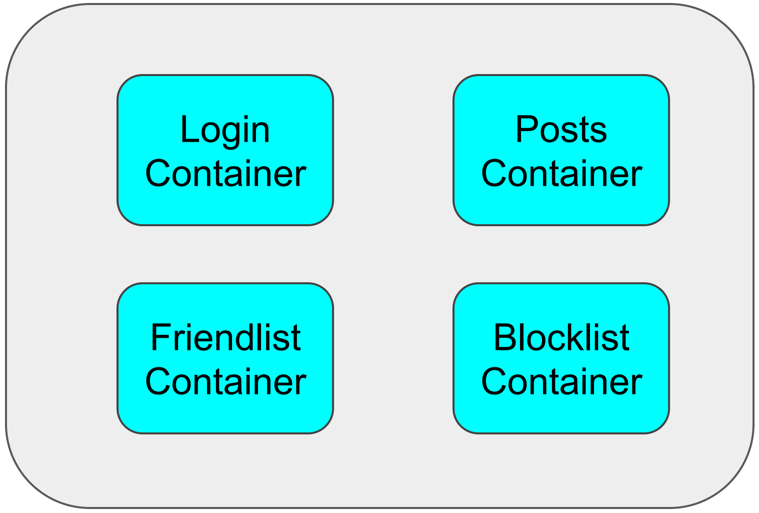
Example: Let's say if every component is given 64GB memory then it is not useful for each component. Some may require more than 64GB and some of them may require less than 64GB.



How to allocate the resources according to each component's need and not waste the resources?

Containers will solve this problem. It will only take the resources that are required for an application to run. In this way, the components(login, posts, newsfeed, etc) has no need to run on multiple servers. Instead there will be multiple containers inside one server that will contain multiple components.

Example: (1) Let's say that while running an application you want to create one more container. You will enter the command to create it but if many users came to an application and you require 1000 more containers, will you enter the command 1000 times to create the containers?



Server

(2) Let’s say that you want to create 500 containers in multiple images. Will you enter the command 500 times for each image?

(3) Let’s say that you created 500 containers but you only need 100 containers. Will you delete the 400 containers one by one?

How to control the creation, deletion of containers and automate the process?

Kubernetes will automate the process of creating, and deleting the containers. If many users logged in at the same time for a match then kubernetes will monitor them and automatically create the containers but if the match is over and users are logged out then kubernetes will automatically delete the containers.

Kubernetes is not specific to be used only for the Docker container. It is used for all the containers like Containerd, rocket etc. It supports all the containers.

Definition

- **Kubernetes** is an open source container management tool which automates the container deployment, container scaling, & load balancing.
- It schedules, runs and manages the isolated containers that are running on virtual/physical/cloud machines.
- All top cloud providers support kubernetes.

History

- Google developed an internal system called borg(later named as omega) to deploy and manage thousands of google applications and servers on their clusters.
- In 2014, google introduced kubernetes an open source platform written in Golang, and later donated to CNCF(Cloud native computing foundation).

Cloud native will take good features and qualities from the cloud & develop something from it.

Kubernetes is also called K8s.

$$\text{K}\underbrace{\text{ubernete}}_{8}\text{s}$$

Online platforms for K8s

- Kubernetes playground
- Play with K8s
- Play with kubernetes classroom

Cloud based K8s services

- **GKE** →Google Kubernetes services
- **AKS** → Azure Kubernetes services
- **Amazon EKS** → Amazon elastic kubernetes services

Kubernetes installation tools

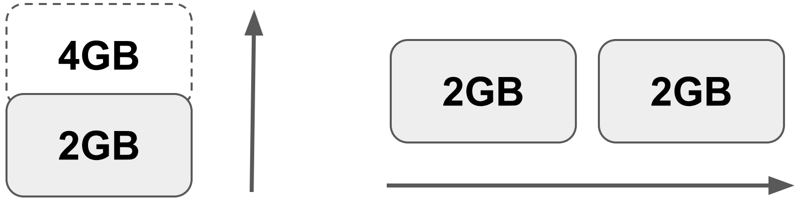
- Minikube
- Kubeadm

Problems with scaling up the containers

- Containers cannot communicate with each other.
- Autoscaling and load balancing was not possible.
- Containers had to be managed carefully.

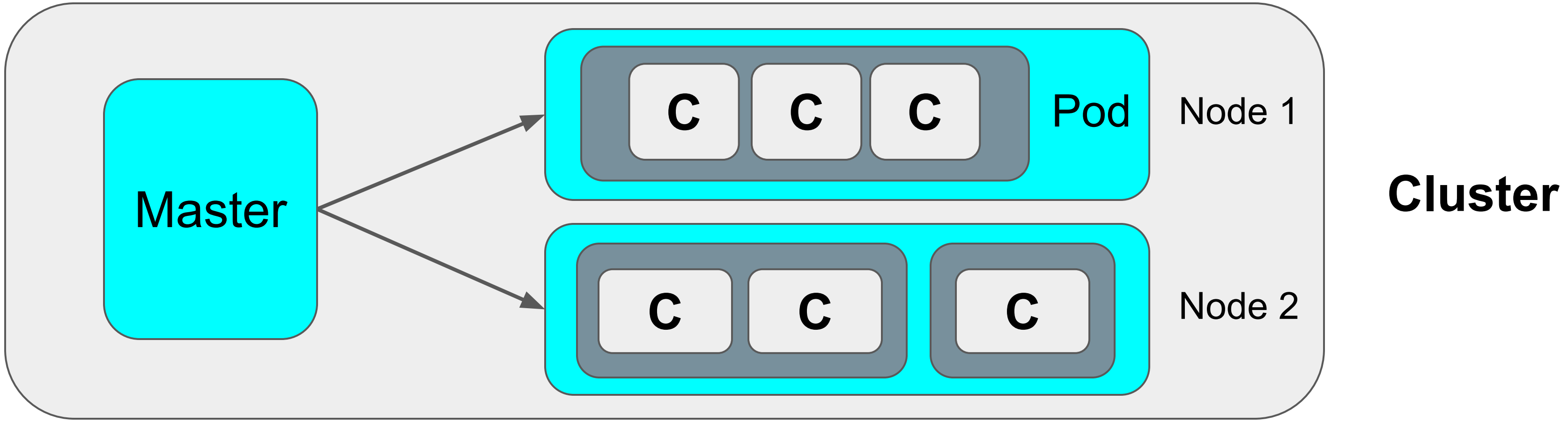
Features of Kubernetes

- **Orchestration**(clustering of any no. of containers running on different networks.)
  - If one container is running physically, one is running virtually and one is running on cloud then these three can make a cluster and it means that it supports **hybrid cloud**.
- **Autoscaling:** It supports vertical and horizontal scaling.



- **Load balancing:** The requests or any kind of load should be equally distributed among containers.
- **Platform independent:** It supports cloud, virtual and physical environments.

- **Fault Tolerance:** It will monitor the pod and if it is failed then it will automatically create new one.
- **Rollback:** It will help you go to the next and previous version of an application.
- **Health monitoring of containers:** It will continuously monitor the health of containers. If one container is failed then it will create new one.
- **Batch execution:** You will give commands in YAML and it will execute them. The execution can be done all at once or sequential(first do this then do that) or parallel(run multiple executions at the same time.)



What we have learned?

Monolithic and microservices

Containers and their downsides

Kubernetes and its history

Kubernetes features and its small architecture

That’s It

I hope you will like this video.

Make sure to subscribe to my channel.

Ask questions in the comment section.