Google Kubernetes Overview and Architect

February 22, 2018 Posted by: Loges

Category: Blog, Cloud Computing, Containers

(https://www.as
kuberr
arcl

Google Kubernetes Overview and

Architecture

In this blog, we will share you about Google Kubernetes overview and Architecture for container environmen

OVERVIEW

- Kubernetes came out of **Google**.
- In 2014, It made open-source and handover to **Cloud Native Computing Foundation** (CNCF). It's a part of Linu
- It's one of the most important open-source projects and written in **GoLang**.
- Google uses the in-house projects like **Borg** and **Omega** for their search, Gmail and other technologies on conta
- The team who built borg and omega also build the **Kubernetes**.
- The word kubernetes came from the Greek word **helmsman**. It means the person who steers the ship.
- The shortened name of kubernetes is **K8S**.

WHAT IS KUBERNETES?

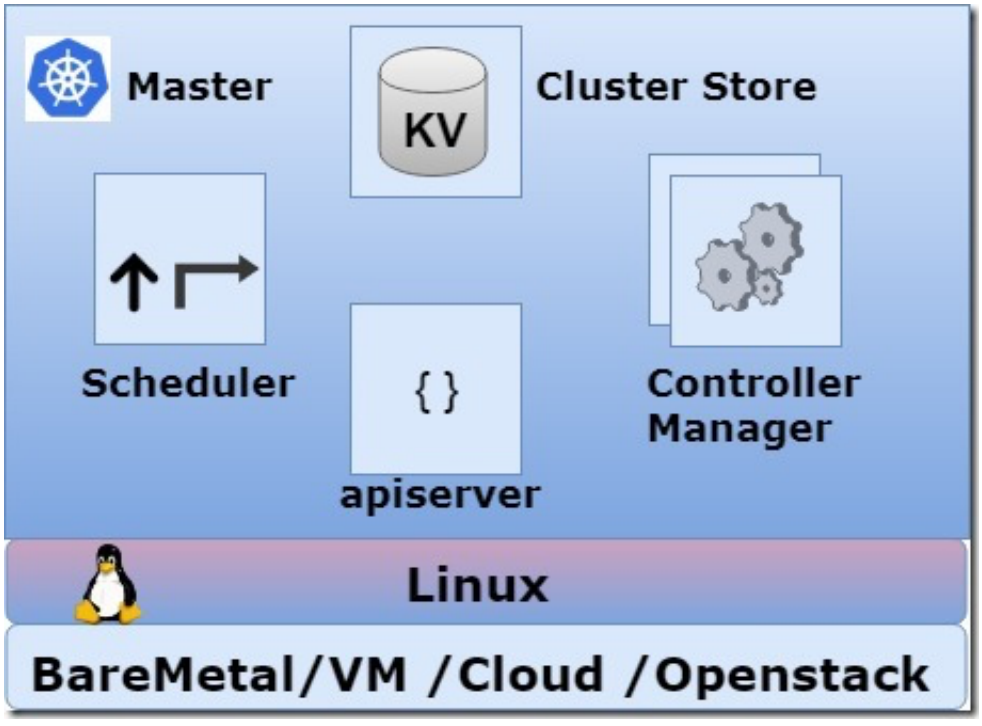
- Containers have the challenges in scalability and synchronization.
- Kubernetes is an **orchestrator** for microservice apps that run on containers.
- **Microservices** is a collection of small and independent services.
- Microservices Apps is made up of **multiple services**. Each packaged as POD and each one has a job to do.
- Kubernetes organize each service to make work **seamlessly**. This is called orchestration. It orchestrates all the s together.
- We create an application and packaged it and give to **kubernetes cluster**. The cluster will be one or more **mast**

of **Nodes**.

- **Masters** are in-charge and they make decisions in which nodes to work on.
- **Nodes** are actually to do the work and report back to the cluster.
- Kubernetes is all about running and orchestration the containerized apps.

KUBERNETES ARCHITECTURE

MASTERS



(https://www.assistanz.com/wp-content/uploads/2018/02/K8S-Master.jpg)

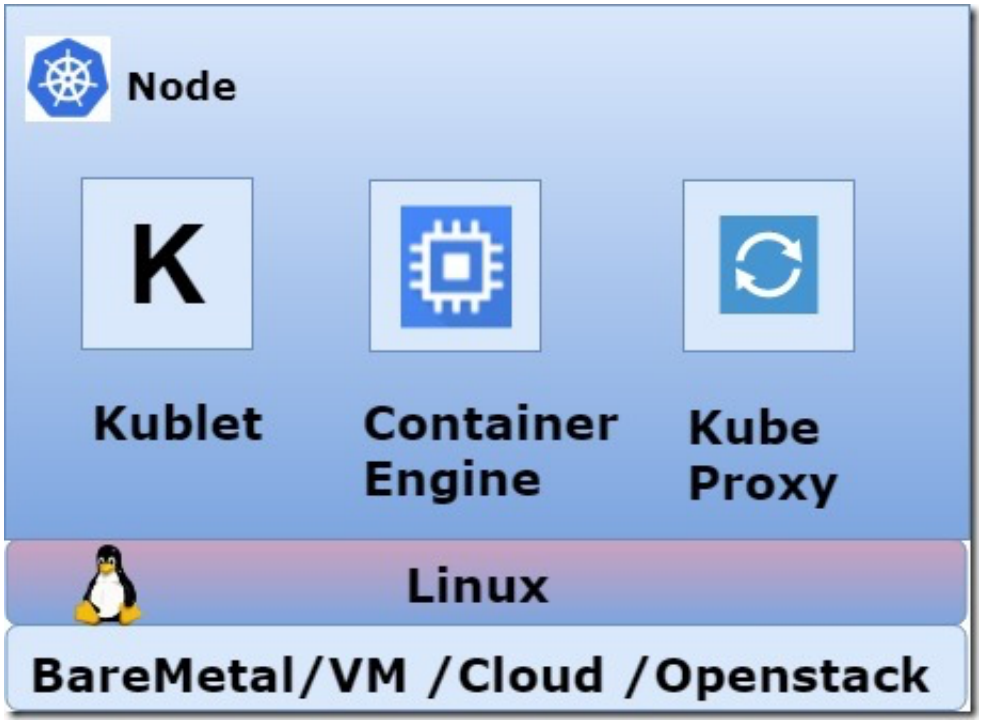
- Kubernetes is a bunch of Masters and nodes.
- It will run on Linux platform. Underneath it can either Baremetal, VM, Cloud instances, and OpenStack.

- We can have Multi-Master HA to share the workloads with multiple masters.
- **Kube-API server** is the front-end master control plane. It is the only master component with the proper externa and exposes through **REST** API that consumes **JSON**. It will expose to the **port 443**.
- **Cluster Store** is the memory of the cluster. It stores the **config** and **state of the cluster**. It used etcd. It's an ope distributed Key value (KV) store. It's developed by coreOS and uses **NoSQL** database.
- **Kube-Controller-Manager** is the controller of controllers. It has node controller, endpoint controller, namespa It has a controller for everything in K8S. It watches for changes and maintains the **Desired State**.
- **Kube-Scheduler** watches the API server for new pods and assigns them to nodes. It also manages affinity/anti-a resources management.

KUBERNETES WORKING OVERVIEW

- The commands and queries come through **kubectl** command.
- The command format will be in **JSON** format.
- It will reach the **API server**. This is where the authentication happens. Once it's completed, based on the comma other components like scheduler, controller, cluster store.
- Then the commands and action items will be hand-over to **nodes.**

NODES



(https://www.assistanz.com/wp-content/uploads/2018/02/Nodes.jpg)

- They are called kubernetes **workers**. It will take the instructions from the Master Server.
- It contains **Kublet**, **Container Runtime**, and **Kube Proxy**.
- **Kublet** is a main kubernetes agent of a node. It can installed it on a Linux host. It registers the node in the **kuber** watches the API SERVER for work assignments.
- Once it receives a task, it carries out the task and reports back to the master. It also reports the kublet failures ar to kubernetes master.
- Control Pane will decide what to do next. Kublet is not responsible for POD failure. It simply reports the status to
- Kublet exposes the endpoint in the localhost on **10255** port.
- The Spec endpoint provide information about the node. The healthz endpoint will perform health check operatic shows the current running PODS.

- **Container Engine** manages the container management like pulling images, starting and stopping the containers
- The container runtime will be **Docker** Mostly. We can also use Core OS **rkt (Rocket)** on kubernetes. These are lik

- **Kube Proxy** is network brains of a node. It will take care of getting a unique IP address for each POD.
- It like one IP Per POD.
- All the containers inside a POD share a single IP.
- It also took care of Load Balancing across all PODS in a **service.**

DECLARATIVE MODEL AND DESIRED STATE

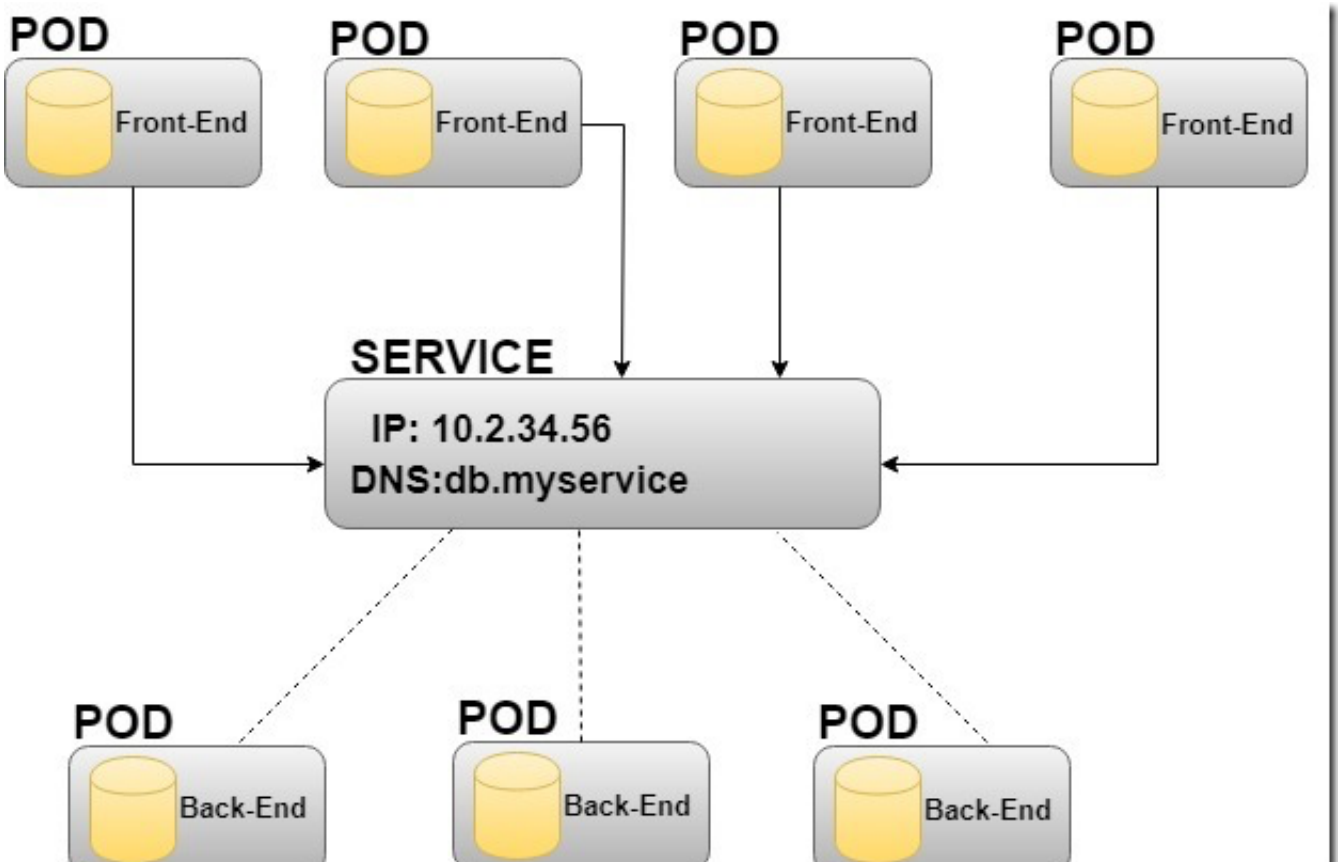
- Kubernates operates on a Declarative Model. It means, we provide the manifest files to API server and describe h cluster to look and feel.
- For example, We tell K8s that use this image and always make 5 copies of that. K8s take care of all the work such creating containers, assigning network, etc.., To do this, we issue to K8s with the manifest file with what we want like.
- Once the K8s build the environment as per the manifest file is called **Desired State** of the cluster.
- Sometimes, when the **Actual State** of the cluster drifts from the **Desired State**, kubernetes takes own action to
- Kubernetes will constantly be checking the actual state of the cluster that matches with the Desired State. If thes it will work continuously till it matches.

PODS

- Containers always run inside a PODS.
- We cannot deploy a container directly in kubernetes. Containers without a POD in kubernetes is called **Naked Co**
- PODS can have one or multiple containers.
- A POD is just a ring-fenced environment to run containers. It's just like a SAN box with the containers in it.
- Inside the POD, we create network stack, kernel namespaces and n number of containers.
- Multiple containers will share **single POD** environment. The two containers inside POD will speak each other thr interface.

- If the couple of containers that shares same memory and volume then we can place them in a single POD. This is **Coupling**.
- If the containers are not required to be a couple, place them in a separate POD and coupled them through a netv **Loose Coupling**.
- Scaling in the kubernetes can be done by adding and removing the PODS. But don't scale more containers to an
- For **Multi-container PODs**, we can have two or more complimentary containers which are related to each other primary container will be the **main container** and secondary container will call as **sidecar container**.
- PODs can be scaled within few seconds. A POD will never declare as available until the whole POD is up and runr
- We cannot have single POD spread over multiple nodes.
- The three phases of **POD life cycle is** Pending, Running & succeeded (or) Failed.
- When a POD dies, the replication control starts another one in its place.
- We can deploy PODS directly in kubernetes through API server using manifests file.

SERVICE



- In the above example, we have a bunch of web server as **front-end** which is taking to **back-end** DB server.
- We place a service object in between these two ends. Service is a kubernetes object like POD (or) Deployments w the manifest file. Provide the manifest file to the API server and create the service object.
- It provides a stable IP and DNS name for the DB pods.
- Using single IP in the service it load-balance the request across the DB PODs.
- If any POD fails and created a new one, Service will detect and update its records automatically.
- A POD belongs to a service via **Labels**. Labels are the simplest and most powerful thing in kubernetes.
- Labels are used to identify the particular PODS for the load balancing.
- Services will send traffic only to healthy PODs. If the PODs fails in the health check, service will not send traffic to
- Services perform load balancing on the random basis. Round Robin is supporting and it can be turned on.

- By default, it uses TCP but it supports UDP also.

DEPLOYMENTS

- Deployments are all about declarative.
- In deployment model, replication controller is replaced with **replica sets**.
- Replica sets are the next generation replication controller.
- It has a powerful **update model** and **simple rollbacks.**
- **Rolling updates** are the core feature of deployments.
- We can multiple concurrent deployment versions.
- Kubernates will detect and stop rollouts if they are not working.
- It's a new model in the kubernetes world.

Thanks for reading this blog. We hope it was useful for you to know about kubernetes and it's archite