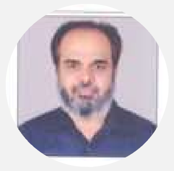
Kubernetes- Container Orchestration



Prakash Ramamurthy
prakash.ramamurthy@wipro.com

Kubernetes Concepts

- Kubernetes is a portable, open-source, expandable platform for managing containerized applications, and manage them through automation
- Kubernetes provides environment to manage containerized services.
- Orchestrates containerized services on large server cluster and manage the network and storage between them.
- Kubernetes serve as a platform for containerized microservices, and allow portability across cloud platforms.

Kubernetes Platform

- Kubernetes serve as a platform to ease deploy, scale and manage the applications, and helps build the ecosystem with required components and tools
- Application development workflow can be accelerated and streamlined.
- Provide robust environment to for applications and help them scale for production
- Labels in the form of key-value pairs are attached to objects.
- These labels are used to relate and organize the resources

Kubernetes Platform

- Kubernetes works at application level with containers above the operating system.
- Kubernetes provide features like deployment, load balancing, scaling, logging and monitoring similar to PaaS offerings.
- Kubernetes is pluggable, extensible with many features being optionable.
- Any application that can be containerized can be managed with Kubernetes
- Kubernetes supports diverse variety of Applications. These may be stateful, stateless or data-processing apps.

Kubernetes Platform

- Kubernetes is generally used as deployment tool for testing or production activity.
- Kubernetes is normally not used to build applications. However this depends on CI/CD workflow used by organization as well technical requirement.
- Kubernetes do not provide any middleware like message buses, databases or data-processing. These may run on Kubernetes or may be accessed by applications through portable mechanisms.
- Kubernetes do not have any comprehensive configuration, management or maintenance mechanisms, built into it.

Kubernetes

Application Deployment: Old Way

- Applications bundled using package manager are used to deploy them on specific operating system
- This require proper configuration of libraries and other dependencies required by application, during deployment
- This requires virtual-machine images that cannot be altered to get predictable rollouts and rollbacks.
- However machine image including operating system are heavy and non-portable.

Kubernetes

Application Deployment – New Way

- Containers which are virtualized at Operating System are level are used to deploy application, rather than VMs which are virtualized at hardware level
- Containers are isolated from the host system environment as well from each other.
- Containers have their own filesystems, which must be matching with the file system on the host system.
- Containers images can be built and launched easily in the operating system environment.
- Since containers have no dependency on underlying infrastructure, they can be easily ported across different Operating System distributions and also across cloud platforms

Kubernetes-Containers

- Container image is smaller and containers can be started fast. Applications can also be distributed across multiple containers
- Each part of application can be separately containerized and hence do not need tight integration with rest of the application.
- Once the application is containerized it has no dependency on specific production infrastructure environment.
- Container images can be created at build/release time, which can be used for production.
- Helps use consistent environment through the application development life cycle.
- As multiple parts of application are containerized it would be easy to deploy them.

Kubernetes cluster - Master

- Kuberenetes Master/Manager manages the entire cluster of servers using following processes.
- Kubernetes API server configures and validates the data for each API object like Pods, Controllers and Services.
- Kubernetes controllers manage and regulate the state of system. Replication controller, namespace controller, endpoints controllers are some of them.
- Kubernetes Scheduler manages the Pods on cluster servers also called worker nodes. Also takes care of node affinity or anti-affinity

Kubernetes cluster – Worker Nodes

- Kubernetes worker nodes run the pods having containerized applications, scheduled by Master Node. This uses following processes.

- Kubelet which is an agent on each node and communicate with the Master node. Responsible for running containers in the pod. This is done according to PodSpecs.

- Kube-proxy is responsible for Kubernetes networking services on each worker node.

- Container Runtime is responsible for creating and running containers. Kubernetes supports Docker, cri-o, containerd, rktlet as CRI (Container Runtime Interface)

Kubernetes cluster – Addons

- Cluster features are implanted using Addons in the form of pods and services. These are managed by various controllers. Often they are Namespaced into kube-system namespace
- DNS: This records for Kubernetes services. Kubernetes containers inside the Pod automatically include DNS in their name searches.
- Web UI (Dashboard): This is browser-based User Interface. This allows users to manage the applications running the cluster.
- Container Resource Monitoring: This records time-series metrics in a central database about the containers. Also provides an User Interface for browsing the data collected.
- Cluster-level Logging: This mechanism is responsible for collecting containers logs to a central log store. This also as a search and browsing interface.

Kubernetes Objects

- Kubernetes Pod: This is the basic build block that encapsulate application containers ad storage resouces. Pods get a unique IP in the cluster. Represent a unit of deployment. Pod may have single or multiple containers which are tightly coupled and share resources.

- Kubernetes Service is an abstraction to access application on a logical set of pods, which are usually multiple instances of particular application deployment.

- Kubernetes volume are storage resources. Containers store data in these volumes. This data is preserved across container restarts.

- Kubernetes Namespaces allow creation of virtual clusters on the existing physical luster. This allow same names to be used by resources across namespaces. Namespaces must be unique within a given namespace.

Kubernetes Controllers

- ReplicaSets are used to create replicated pods of given containerized applications.
- Deployment controller manage application deployment using ReplicaSets and Pods. Responsible for maintaining desired state of application in the deployment
- StatefulSet manages the stateful applications and create replicated pods. StatefulSets maintain a sticky identity which is maintained across multiple rescheduling.
- DaemonSet runs one copy of Pod on each node. As new nodes are added to cluster, DaemonSet Pod is automatically added to them. These Pods are garbage collected when the nodes are removed.

Kubernetes Pods

- Pod is the basic entity to be scheduled on nodes.
- Pod encapsulates one or more containers and volumes used as storage resources.
- Applications in these containers must be tightly coupled and share data over volumes.
- All the containers within the Pod are automatically scheduled together within the Pod on the same servers in the cluster
- Each Pod maintains a single instance of applications.
- Pods may be scaled horizontally by creating multiple pods representing multiple instances of the same application.
- Such replicated Pods are usually created and managed by a controller

Kubernetes Pods

- Each Pod gets a unique IP address in the cluster. Services hosted in the containers may expose on differnet ports on the Pod.
- Each container within the Pod shares the Network connectivity which include the IP address and the Ports
- While containers within the Pod share IP address and Ports, they can find each other using localhost.
- They may also communicate using standard inter-process communication mechanisms like SystemV semaphores or Posix shared memory.
- When the services within containers of same Pod required to communicate outside the Pod, they use the Pod IP address and Ports.
- Containers within the Pod share the data over volumes.
- Volumes persist the data even when the corresponding container is restarted.

Kubernetes – Replication Controller

- ReplicationController can replicate multiple Pods of identical types and manage them.
- ReplicationController will automatically replaces the Pods, whenever they fail, or get deleted or terminated.
- ReplicationController is often represented as “rc” or “rcs” in the kubectl commands.
- Pod template in a ReplicationController specify appropriate labels which are used by both the controller and the Pods to define the relationship.
- While deleting ReplicationController, first it would be scaled to zero to ensure termination of all the Pods, before deleting the controller itself.
- ReplicationController would disown the Pods, once the Pod labels are changed.
- ReplicationController then may be deleted leaving the Pods intact.
- When a new ReplicationController is created with the same label, that can adopt the old pods.

Kubernetes - Replicasets

- Repicaset is used to create replicated Pods of identicle types and manage them.
- Selector label define the relationship between the Replicaset and the Pods.
- With the change of labels Replicaset can disown the pods or acquire the old pods.
- Depending the need of scaling Replicaset can create or delete pods

- Deployment Controller uses ReplicaSets and manages multiple Pods in the application deployment. Here ReplicaSets are controlled by Deployment controller.
- Generally it is recommended to use Deployment Controller rather than ReplicaSets.

Kubernetes – Deployments

- Deployment controller manages the application deployment in Pods through ReplicaSets.
- Desired state for the application deployment involving scaling and image are managed through Deployment controller. Deployment Controller can scale the Pods as well update the images of named containers within the Pod.
- Deployment can disown a ReplicaSet and create a new one with associated Pods.
- Existing Deployment may also disown all its resources which may be acquired by a new Deployment.

Kubernetes – StatefulSets

- StatefulSet is used to manage stateful applications.
- Create the Pods as part of Deployment and also scale them as per requirement.
- Maintains a sticky identity with each Pod, which is persisted when any Pod is deleted and recreated.
- Also orders the Pods when they are scaled.
- StatefulSet also updates the named containers within the Pod.

Kubernetes – DaemonSet

- DaemoSet creates one Pod for each Node.
- New Pods are created whenever a new Node is added to cluster. When the Nodes are removed these Pods are garbage collected.
- When a DaemonSet is deleted all the Pods are deleted.
- Deleting a DaemonSet will clean up the Pods it created
- Normally these are used to collected information from each node in the cluster.

- Typical use cases:
  - Running a cluster storage daemon, (Ex: glusterd, ceph)
  - Runninng a log collection daemon, (Ex: fluentd or logstash)
  - Running a node monitoring daemon (Ex: Prometheus Node Explorer, Dynatrace OneAgent etc.

Kubernetes Services

- Kubernetes Service is the way to access the service which is replicated into multiple Pods.
- ClusterIP associated with the Service is used to access the service encapsulated within the Pod. As Pod IPs change, service IP provide a stable access point through DNS.
- Services are basically REST object which can be accessed through http url.
- Set of Pods controlled by a controller is usually determined by a selector label
- Clients accessing the service may access any of the replicated Pods at the backend. Thus services abstract the applications instantiated in the backend.
- Kubernetes offers a virtual-IP based bridge to Services which redirects to the backend Pods.

Thank You

Public