AIOPS Assignment 5

Submitted by **Diana Varghese**

(dianavarghese100@gmail.com)

1. What exactly is Kubernetes?

   Kubernetes is a container orchestration platform for scheduling and automating the deployment, management, and scaling of containerized applications.

   Kubernetes was first developed by engineers at Google before being open sourced in 2014. It is a descendant of Borg, a container orchestration platform used internally at Google. Kubernetes is Greek for helmsman or pilot, hence the helm in the Kubernetes logo.

   Today, Kubernetes and the broader container ecosystem are maturing into a general-purpose computing platform and ecosystem that rivals virtual machines as the basic building blocks of modern cloud infrastructure and applications. This ecosystem enables organizations to deliver a high-productivity Platform-as-a-Service (PaaS) that addresses multiple infrastructure-related and operations-related tasks and issues surrounding cloud-native development so that development teams can focus solely on coding and innovation.

2. What do you mean by Kubernetes load balancing?

   Load Balancing is the method by which we can distribute network traffic or client's request to multiple servers.
   Internal Load Balancing: In Kubernetes, the most basic Load Balancing is for load distribution which can be done at the dispatch level. This can be done by kube-proxy, which manages the virtual IPs assigned to services. Its default mode is iptables which works on rule-based random selection. But that is not really a Load Balancer like Kubernetes Ingress, which works internally with a controller in a customized Kubernetes pod.
   As **Ingress** is Internal to Kubernetes, it has access to Kubernetes functionality. Considering this, the configurable rules defined in an Ingress resource allow details and granularity very much. These can be modified as per the requirements of an application and its pre-requisites.
   External Load Balancing: This distributes the external traffic towards a service among available pods as external Load Balancer can't have direct to pods/containers. An External Load balancer is possible either in the cloud if you have your environment in the cloud or in such an environment which supports an external load balancer. Clouds like AWS, Azure, GCP provides external Load.

3. What exactly do you mean by "operator," and why do we need them?

   An operator is a custom Kubernetes controller that uses custom resources (CR) to manage applications and their components. High-level configuration and settings are provided by the user within a CR. The Kubernetes operator translates the high-level directives into the low level actions, based on best practices embedded within the operator's logic.

It builds upon the basic Kubernetes resource and controller concepts, but includes domain or application-specific knowledge to automate the entire life cycle of the software it manages.

4. What is the best way to operate Kubernetes locally?
   Running microservices in Kubernetes usually requires a cluster running in the cloud or on-premise. During the development or when debugging, developers often need to run their application quickly in Kubernetes. Spinning up a new cluster or configuring the deployment to an existing one might take longer than the time they actually need the cluster.

   The solution to this problem is to run Kubernetes locally on your development machine using Docker Desktop.

5. How can the Kubernetes Cluster be monitored?

   There are several cluster orchestration tools, but Kubernetes (K8S) is becoming increasingly popular when compared to its competitors. A container orchestration tool such as Kubernetes handles containers in several computers, and removes the complexity of handling distributed processing.

   There are several Kubernetes metrics to monitor. These can be separated into two main components: (1) monitoring the cluster itself, and (2) monitoring pods.

   Kubernetes Cluster Monitoring
   For cluster monitoring, the objective is to monitor the health of the entire Kubernetes cluster. As an administrator, we are interested in discovering if all the nodes in the cluster are working properly and at what capacity, how many applications are running on each node, and the resource utilization of the entire cluster. Measurable metrics:
   - Node resource utilization – Resource utilization metrics like network bandwidth, disk utilization, CPU and memory utilization.
   - The number of nodes – the number of nodes available
   - Running pods – the number of pods running will show you if the number of nodes available is sufficient for the entire workload in case a node fails.

   Kubernetes Pod Monitoring
   The act of monitoring a pod can be separated into three categories:
   (1) Kubernetes metrics - we can monitor how a specific pod and its deployment are being handled by the orchestrator (number of instances a pod has at the moment and how many were expected, how the on-progress deployment is going, health checks and network data)
   (2) Pod container metrics - available mostly through cAdvisor and exposed by Heapster, which queries every node about the running containers (metrics like CPU, network, and memory usage compared with the maximum allowed)
   (3) Application metrics - developed by the application itself and are related to the business rules it addresses

6. What exactly do you mean when you say GKE?
   Google Kubernetes Engine (GKE) is a platform for running Kubernetes that is created by engineering contributors to K8s. Using GKE, you can begin with single-click clusters and have an option for scaling up to 15000 nodes. Moreover, you will get support for a high-availability control plane with multi-zonal and regional clusters. GKE can remove the operational overhead with industry-first four-way auto-scaling and performs a secure scanning of container images and data encryption.
   Using GKE, you get options for:
   - Firstly, developing a large variety of apps with support for stateful, serverless, and application accelerators.
   - Secondly, using Kubernetes-native CI/CD tooling for securing and speeding every stage of the build-and-deploy life cycle.
   - Thirdly, select the suitable channel as per the business needs.
   - Lastly, getting back time for focusing on your applications using Google Site Reliability Engineers (SREs).
7. What is the difference between Kubernetes and Docker Swarm?

Aspect	Kubernetes	Docker Swarm
Installation and setup	Manual installation can differ for each operating system; No installation is required for managed offerings from cloud providers.	There is simple installation with Docker, and instances are typically consistent across operating systems; Swarm is easier to install and configure.
Scalability	Horizontal autoscaling is built in; offers all-in-one scaling based on traffic	Offers autoscaling of groups on demand; emphasizes scaling quickly.
Load balancing	No auto balancing but an external load balancer can easily be integrated via third-party tools; has access to container applications through an IP address or HTTP route.	Has automatic load balancing with internal balancers
High availability	By diverting traffic away from unhealthy pods, Kubernetes is self-healing.	Swarm Managers offer availability controls, and microservices can be easily duplicated.