**Kubernetes Assignment 3**

1. How do you monitor the Kubernetes cluster?

* there are various tools and methods for monitoring. Let me list few of them
* Heapster, cAdvisor, Prometheus, InfluxDB, Grafana
* let see few of them
* Prometheus is an application that is used for monitoring and alerting. It can be called out to your systems, grab real-time metrics, compress it, and stores properly in a database.
* to monitor your Kubernetes cluster is by using a combination of Heapster to collect metrics, InfluxDB to store it in a time series database, and Grafana to present and aggregate the collected information.
* The Prometheus ecosystem consists of multiple components.
* Mainly Prometheus server which scrapes and stores time-series data.
* Special-purpose exporters for services like StatsD, HAProxy, Graphite, etc.
* An alert manager to handle alerts on various support tools.
* Client libraries for instrumenting application code.
* Push gateway for supporting short-lived jobs.

1. How do we get POD's central logs?

* Pod is smallest unit inside worker node insidee which containers are running. it will be encapsulated inside the pod. you can n no. of pods. under which your applications runs
* A pod is the smallest deployable unit that can be managed by Kubernetes.
* A pod is a logical group of one or more containers that share the same IP address and port space.
* The main purpose of a pod is to support co-located processes, such as an application server and its local cache
* This architecture depends upon the application and many other factors.
* let us see few of the common logging patterns
* Streaming sidecar container.
* Node level logging agent.
* Export logs directly from the application.
* Sidecar container with the logging agent.
* In the setup, journalbeat and filebeat are running as daemonset. Logs collected by these are dumped to the kafka topic which is eventually dumped to the ELK stack.
* The same can be achieved using EFK stack and fluentd-bit.

1. What are the components of a Kubernetes node?

* A node is a machine either physical or virtual machine on which kubernetes is installed
* A node is a worker machine and this is where containers inside the pods will be launched by kubernetes
* Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment
* you can have N no. of nodes, depending on how much big your app is.
* Inside worker node we below:

1. **pod**

* It is smallest unit inside worker node inside which containers are running. it will be encapsulated inside the pod. you can n no. of pods. under which you applications runs

1. **Docker**

* it provides containerization solution

1. **Kubelet**

* It is a node agent which tracks/stores, create, update,delete contrainer. Component is being managed by kubelet.
* Ensures that containers are running in a Pod by interacting with the Docker engine , the default program for creating and managing containers. Takes a set of provided PodSpecs and ensures that their corresponding containers are fully operational.

1. **Kube proxy**

* it is used for network proxy
* **Manages network connectivity and maintains network rules across nodes. Implements the Kubernetes Service concept across every node in a given cluster.**

1. How does Kubernetes make containerized deployment more manageable?

Below are the few points or highlights which makes kubernetes more manageable. Let us go in little depth

* Kubernetes is also known as K8s is an open-source system for automating deployment, scaling, and management of containerized applications.
* it manages containers
* It groups containers that make up an application into logical units for easy management and discovery.
* Kubernetes is used to manage your containers
* It is an open-source container Orchestration Framework.
* With Containers we can bundle out application and run it.
* In realtime while in a production environment you need to manage the containers that run the applications and ensure that there is no downtime. let say due to some reason if production container goes down, another container needs to start. It would be more easy if system can take care of that task.
* So that's the place where kubernetes comes into picture
* Kubernetes provides you with a framework to run distributed systems resiliently.
* It takes care of scaling and failover for your application, provides deployment patterns, and more. For example, Kubernetes can easily manage a canary deployment for your system.
* Below are some of the essential Kubernetes features:
* Automated Scheduling/Automates various manual processes - for instance, Kubernetes will control for you which server will host the container, how it will be launched etc.
* Interacts with several groups of containers - Kubernetes is able to manage more cluster at the same time
* Provides additional services - as well as the management of containers, Kubernetes offers security, networking and storage services
* Self-monitoring - Kubernetes checks constantly the health of nodes and containers
* Horizontal scaling - Kubernetes allows you scaling resources not only vertically but also horizontally, easily and quickly
* Storage orchestration: Kubernetes mounts and add storage system of your choice to run apps
* Automates rollouts and rollbacks: if after a change to your application something goes wrong, Kubernetes will rollback for you sometimes also known for Self-Healing Capabilities
* Container balancing: Kubernetes always knows where to place containers, by calculating the “best location” for them
* Run everywhere: Kubernetes is an open source tool and gives you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you move workloads to anywhere you want.
* Because of all above features it makes kubernetes to manage deployement container easy

1. What is the difference between Kubernetes and Docker Swarm?

Let's see the major difference.

|  |  |
| --- | --- |
| Kubernetes | Docker swarm |
| Installation is complicated | Installation is easy |
| Once installation is done cluster is robust | Once installation is done cluster is not robust |
| It can process auto-scaling | It cannot process auto scaling |
| Manually configuration is required for load balancing | It does auto load balancing |
| Kubernetes provides such integrated tools for logging and monitoring purposes. | Docker requires third-party tools such as the ELK stack for logging and monitoring. |
| In Kubernetes, we can only share storage volumes with containers in the same pod. | In Docker Swarm, we can easily share storage volumes with any container. |
| we can deploy rolling updates as well as automatic rollbacks. | We can deploy rolling updates in Docker Swarm but can't deploy automatic rollbacks. |
| Kubernetes provides a GUI in the form of a dashboard. | Docker Swarm doesn't provide a GUI. |

1. What exactly does Kubernetes controller manager imply?

* It is a part of Master Node
* task of Control Manager is to create note, delete note, update.
* it keeps a track of each resources, eg../how many/ what no. of nodes are running. etc.
* Runs controller processes and reconciles the cluster’s actual state with its desired specifications. Manages controllers such as node controllers, endpoints controllers and replication controllers.