1. How do you monitor the Kubernetes cluster?

The most straightforward solution 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**.

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

This architecture depends upon the application and many other factors. Following are the common logging patterns

* Node level logging agent.
* Streaming sidecar container.
* Sidecar container with the logging agent.
* Export logs directly from the application.

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?

The important component of node status are:

* Condition
* Capacity
* Info
* Address

1. How does Kubernetes make containerized deployment more manageable?

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

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

|  |  |  |
| --- | --- | --- |
| **Features** | **Kubernetes** | **Docker Swarm** |
| **Installation & Cluster Config** | Setup is very complicated, but once installed cluster is robust. | Installation is very simple, but the cluster is not robust. |
| **GUI** | GUI is the [Kubernetes Dashboard](https://www.edureka.co/blog/kubernetes-dashboard/" \t "_blank). | There is no GUI. |
| **Scalability** | Highly scalable and scales fast. | Highly scalable and scales 5x faster than Kubernetes. |
| **Auto-scaling** | Kubernetes can do auto-scaling. | Docker swarm cannot do auto-scaling. |
| **Load Balancing** | Manual intervention needed for load balancing traffic between different containers and pods. | Docker swarm does auto load balancing of traffic between containers in the cluster. |
| **Rolling Updates & Rollbacks** | Can deploy rolling updates and does automatic rollbacks. | Can deploy rolling updates, but not automatic rollback. |
| **DATA Volumes** | Can share storage volumes only with the other containers in the same pod. | Can share storage volumes with any other container. |
| **Logging & Monitoring** | In-built tools for logging and monitoring. | 3rd party tools like ELK stack should be used for logging and monitoring. |

1. What exactly does Kubernetes controller manager imply?

Multiple controller processes run on the master node but are compiled together to run as a single process which is the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.

So, the different types of controller manager running on the master node are :  
