**Introduction to K8S**

**(Kubernetes)**

* To meet enterprise customer requirements with enterprise capabilities, there are 2 major challenges have to be resolved, with regards to deploying and managing containers
  + Distributed Systems Management
    - Managing Nodes in the Cluster
    - Adding / Updating / Removing Nodes
    - Applying Configuration Changes
    - Upgrading System Software across nodes in the cluster
    - Monitoring / Observing Nodes
    - Identifying configuration changes and bring all nodes to the same configuration (DSC – Desired State Configuration)
  + Deployment Management
* What do we need?
  + Enterprise Container Management Platform
* K8S – Is an open source platform, built / managed by Google, that manages containers in the form of a cluster – to meet large / enterprise scale customer requirements.
* Along with the automated deployment and scaling of containers, it provides healing by automatically restarting failed containers and rescheduling them with their hosts die.
* This capability that is provided by K8S, enhances the applications’ HA / Resiliency / Robustness.

Benefits:

* Automated Deployment and Management
  + Declarative Specification of Deployment
* Scalability
  + Cluster Level
  + Pod / Container
  + Vertical Scaling
  + Horizontal Scaling
* HA
* Cost Effectiveness
* Improved Developer / Ops Productivity

Features and Characteristics of K8S:

* Master Node (Control Plane)
  + API Server
    - It’s the entry point for all REST commands issued by KUBECTL | any other UI tools to control the cluster. For example, if there’s any deployment to happen to the cluster, you use KUBECTL to talk to the Master Node API Server.
  + Scheduler
    - It’s a service in the master node, which is responsible for distributing the workload.
  + Controller Manager
    - AKA controller, it’s a daemon that runs in a non-terminating loop and is responsible for collecting and sending information to the API server to regulate K8S cluster by performing lifecycle functions such as Deployment, Monitoring, Observability, Collecting Logs, what changes are drifted and so on.
* Worker Node (Data Plane)
  + Kubelet
    - It’s an agent, which is installed in the worker node, helps to communicate with the master node and worker node to take instructions and apply changes. It gets the pod / deployment specific instructions from the Master Node and apply changes to the worker node.
  + Kube-Proxy
    - It’s the core networking component inside the cluster / worker node, responsible for maintaining the entire network configuration. It automatically knows how to handle Load Balancing across other Kube-Proxies with in the worker nodes of the cluster, provide HA, Fault Tolerance, Distributed / Load Balanced Requests and so on.
  + Pods
    - It’s a group of contains that are deployed together on the same host. It can be understood as a VM logically. With the help of Pods, we can deploy multiple dependent containers together so it acts as a wrapper around these containers so we can interact and manage these containers as if they’re running in a same machine.
    - Containers
      * Instance of a Container Image, which represents an application to be up and running.

