**What is kubernetes:**

* Kubernetes is an opensource container cluster manager.
* Kubernetes primary goal is to provide a platform for automating deployment, scaling and operation of application containers across a cluster of hosts.

**Components of K8s:**

Building block of kubernetes

* Nodes
* Pods
* Labels
* Selectors
* Controllers
* Services
* Control Pane
* API

**What is Docker:**

* It’s a tool that packages up and application and all its dependencies in a “virtual container” so that it can be run on any linux system or distribution.

**When would I use docker:**

There are lot of reasons to use Docker. Although you will generally hear about Docker used in conjuction with development and deployment of application, there are aton of example for use.

1. Configuration Simplification
2. Enhance Developer Productivity
3. Server Consolidation and management
4. Application Isolation
5. Rapid Deployment
6. Build Management

**What is container:**

* It is an entirely isolated set of packages, libraries and/or applications that are completely independent from its surroundings.
* Container however, use shared operating systems and are more efficient in system resource terms.

**Main components of Docker,**

* Daemon
* Client
* Docker.io registry

Daemon & client communicate via RESTful API (Http connection with xml file)

**Docker Hub** – Public repository

**K8s Architecture:**

**Minions (Nodes):**

* You can think of these as “container clients”. These are the individual hosts (physical or virtual) that Docker is installed on and hosts the various containers within your managed cluster.
* Each minion will run ETCD(a key pair management and communication service, used by k8s for exchanging messages and reporting on cluster status) as well as the kubernetes proxy.

**Pods:**

* A pod consists of one or more containers. Those containers are guaranteed (by the cluster controller) to be located on the same host machine in order to facilitate sharing of resources.
* Pods are assigned unique IPs within each cluster. These allow an application to use ports without having to worry about conflicting port utilization.
* Pods can contain definitions of disk volumes or share, and then provide access from those to all the members (containers) within pod.
* Finally, pod management is done through the API or delegated to a controller.

**Labels:**

* Clients can attach “key-value pairs” to any object in the system (like pods or minions). These become the labels that identify them in the configuration and management of them.

**Selectors:**

* Label Selectors represent queries that are made against those labels. They resolve to the corresponding matching objects.
* These two items are primary way that grouping is done in kubernetes and determine which components that given operation applies to when indicated.

**Controllers:**

* These are used in the management of your cluster. Controllers are the mechanism by which your desired configuration state is enforced.
* Controllers manage a set of pods and, depending on the desired configuration state, may engage other controllers to handle replication and scaling (Replication Controller) of XX number of containers and pods across the cluster. It is also responsible for replacing any container in a pod that fails (based on the desired state of the cluster).
* Other controllers that can be engaged include a DaemonSet Controller (enforces a 1 to 1 ratio of pods to minions) and a Job Controller (that runs pods to “completion”, such as in batch jobs).
* Each set of pods any controller manages, is determined by the label selectors that are part of its definition.

**Services:**

* A pod consists of one or more containers. Those containers are guaranteed (by the cluster controller) to be located on the same host machine in order to facilitate sharing of resources.
* This is so that pods can ‘work together’, like in a multi-tiered application configuration. Each set of pods that define and implement a service (like MySQL or Apache) are defined by the label selector.
* Kubernetes can then provide service discovery and handle touting with the static IP for each pod as well as load balancing (round robin based) connections to that service among the pods that match the label selector indicated.
* By default, although a service is only exposed inside a cluster, it can also be exposed outside a cluster as needed.

**K8s installation:**

**Packages and dependencies:**

1. Install ntp package

# yum –y install ntp

# systemctl start ntp && systemctl enable ntp

# systemctl status ntp

1. Add the all node IPs to the /etc/hosts on all nodes. Run ping for verification
2. Disable firewalld & iptables

# systemctl stop firewalld

#systemctl stop iptables

1. Setup Repos

#vi /etc/yum.repos.d/virt7-docker-common-release

[virt7-docker-common-release]

name=virt7-docker-common-release

baseurl=http://cbs.centos.org/repos/virt7-docker-common-release/x86\_64/os/

gpgcheck=0

1. Install docker & kubernetes in Master node

# yum install –y –enablerepo=virt7-docker-common-release kubernetes docker