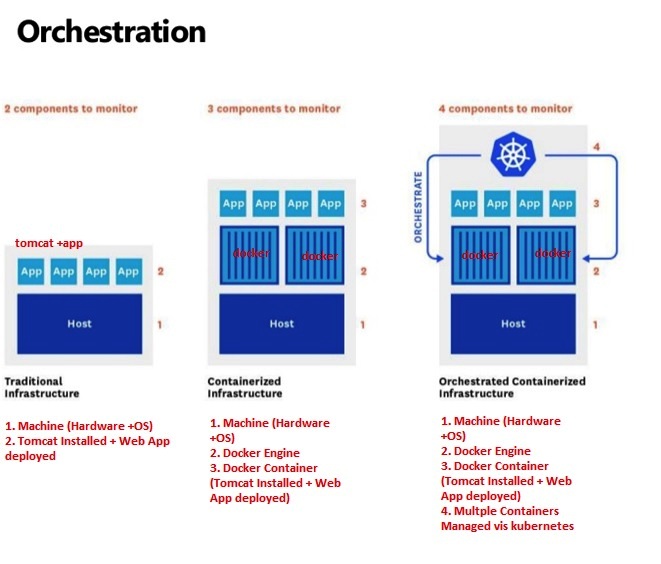
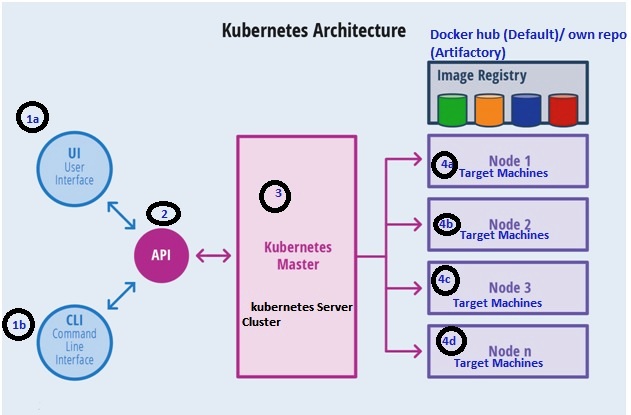
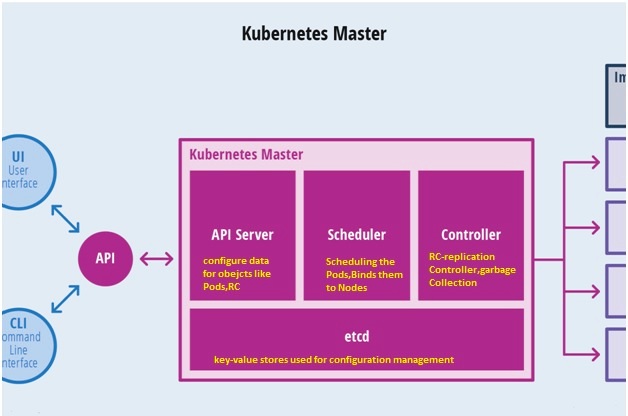
**Containers Introduction**

Architecture of Kubernetes provides a flexible, loosely-coupled mechanism for service discovery. Like most distributed computing platforms, a Kubernetes cluster consists of at least one master and multiple compute nodes.



The [master](http://kubernetes.io/docs/admin/high-availability/) is responsible for exposing the application program interface (API), scheduling the deployments and managing the overall cluster. Each [node](http://kubernetes.io/docs/admin/node/) runs a container runtime, such as [Docker](https://github.com/docker/docker) or [rkt](https://github.com/coreos/rkt), along with an agent that communicates with the master. The node also runs additional components for logging, monitoring, service discovery and optional add-ons. Nodes are the workhorses of a Kubernetes cluster. They expose compute, networking and storage resources to applications. Nodes can be virtual machines (VMs) running in a cloud or bare metal servers running within the data center.





1.API Server Main Management Component of Entire Cluster

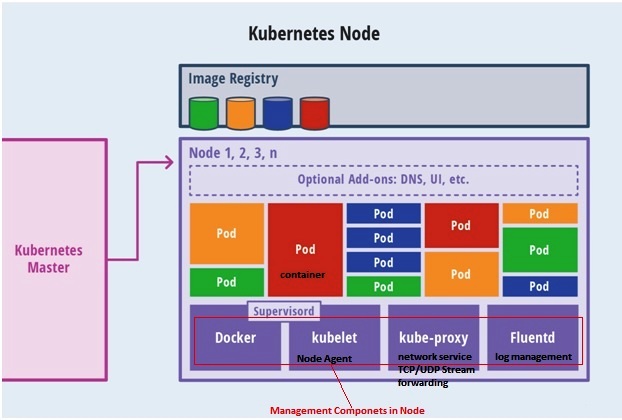
2.API Server is the Only Components connects to etcd, all the other components must go through the API Server to with Cluster State

3. API Server responsible for Authentication/Authorization mechanism

The definition of Kubernetes objects, such as pods, replica sets and services, are submitted to the master. Based on the defined requirements and availability of resources, the master schedules the pod on a specific node. The node pulls the images from the container image registry and coordinates with the local container runtime to launch the container.

[etcd](https://github.com/coreos/etcd) is an open source, distributed key-value database from CoreOS, which acts as the single source of truth ([SSOT](https://en.wikipedia.org/wiki/Single_source_of_truth)) for all components of the Kubernetes cluster. The master queries etcd to retrieve various parameters of the state of the nodes, pods and containers.

This architecture of Kubernetes makes it modular and scalable by creating an abstraction between the applications and the underlying infrastructure.



A pod is a collection of one or more containers.. The grouping mechanism of pods make up for the differences between containerization and virtualization by making it possible to run multiple dependent processes together.

Replica sets deliver the required scale and availability by maintaining a pre-defined set of pods at all times. A single pod or a replica set can be exposed to the internal or external consumers via services. Services enable the discovery of pods by associating a set of pods to a specific criterion. Pods are associated to services through key-value pairs called labels and selectors. Any new pod with labels that match the selector will automatically be discovered by the service. This architecture provides a flexible, loosely-coupled mechanism for service discovery.

Pod

pod is a collection of one or more containers (Such as Docker Containers), Application Container

Pod represents a running process on your cluster

The pod serves as Kubernetes’ core unit of management. Pods act as the logical boundary for containers sharing the same context and resources

At runtime, pods can be scaled by creating replica sets, which ensure that the deployment always runs the desired number of pods.