KUBERNETES

certification

wo certifications provided by CNCF(**Cloud Native Computing Foundation**) for Kubernetes

1. Kubernetes Administrator <https://www.cncf.io/certification/cka/>
   * <https://github.com/cncf/curriculum/blob/master/certified_kubernetes_administrator_exam_v1.9.0.pdf>
2. Kubernetes Application Developer <https://www.cncf.io/certification/ckad/>
   * <https://github.com/cncf/curriculum/blob/master/certified_kubernetes_application_developer_exam_v1.0.pdf>

Play Ground with NO CLUSTER SETUP : <https://www.katacoda.com/courses/kubernetes>

Certificates : <https://www.youtube.com/watch?v=gXz4cq3PKdg>

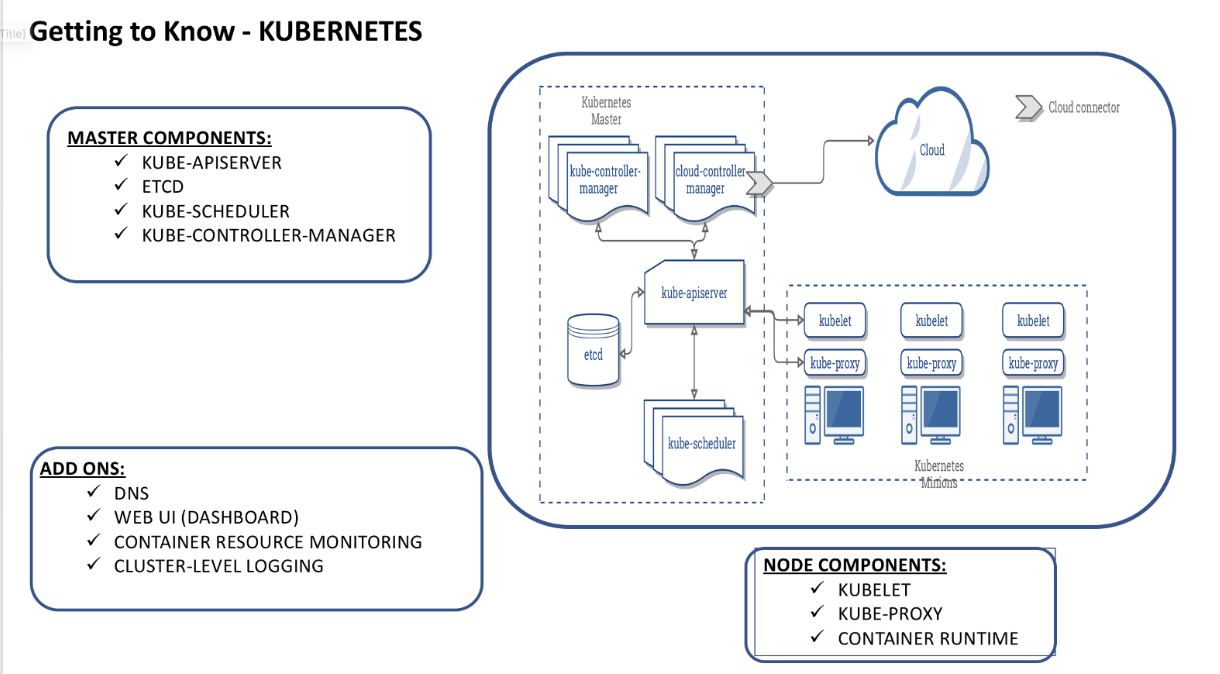
RBAC : <https://www.youtube.com/watch?v=Nw1ymxcLIDI>

Auth & Access Control : <https://www.youtube.com/watch?v=WvnXemaYQ50&t=39s>

* Single node cluster setup(Minikube)  : <https://kubernetes.io/docs/setup/minikube/>

* Online interactive tutorials : <https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/>

* Linux Academy sessions with free cloud servers setup : <https://linuxacademy.com/cp/modules/view/id/155> (**Certified Kubernetes Administrator (CKA))**



This platform is based out of typical master-slave architecture.There are 4 major components in Kubernetes architecture(For ease understanding this section description has been referred from Kubernetes docs).

* **Master Components**

* + **Kube API Server:** The API Server services REST operations and provides the frontend to the cluster’s shared state for the Kubernetes pods, services, replication controllers and others.

* + **Kube Scheduler :**Collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference and deadlines.

* + **Kube Controller Manager :**These controllers include
    - Node Controller: Responsible for noticing and responding when nodes go down.
    - Replication Controller: Responsible for maintaining the correct number of pods for every replication controller object in the system.
    - Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods).
    - Service Account & Token Controllers: Create default accounts and API access tokens for new namespaces.

* **ETCD Cluster**

* + Consistent and highly-available key value store used as Kubernetes’ backing store for all cluster data.
  + Always have a backup plan for etcd’s data for your Kubernetes cluster. For in-depth information on etcd, see [etcd documentation.](https://github.com/coreos/etcd/blob/master/Documentation/docs.md)

* **Node/Slave Components :**Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

* + **kubelet:**An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those  PodSpecs are running and healthy. The kubelet doesn’t manage containers which were not created by Kubernetes.

* + **kube-proxy:**[kube-proxy](https://kubernetes.io/docs/admin/kube-proxy/) enables the Kubernetes service abstraction by maintaining network rules on the host and performing connection forwarding.

* + **Container Runtime:**The container runtime is the software that is responsible for running containers. Kubernetes supports several runtimes: [Docker](http://www.docker.com/), [rkt](https://coreos.com/rkt/), [runc](https://github.com/opencontainers/runc) and any OCI [runtime-spec](https://github.com/opencontainers/runtime-spec) implementation.

* **ADD ONS :**Addons are pods and services that implement cluster features. The pods may be managed by Deployments, ReplicationControllers, and so on. Namespaced addon objects are created in the **kube-system** namespace.

* + **DNS:**While the other add-ons are not strictly required, all Kubernetes clusters should have [cluster DNS](https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/), as many examples rely on it. Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services.Containers started by Kubernetes automatically include this DNS server in their DNS searches.

* + **Web UI (Dashboard):**[Dashboard](https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/) is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.

* + **Container Resource Monitoring:**[Container Resource Monitoring](https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/) records generic time-series metrics about containers in a central database, and provides a UI for browsing that data.

* + **Cluster-level Logging:**A [Cluster-level logging](https://kubernetes.io/docs/concepts/cluster-administration/logging/) mechanism is responsible for saving container logs to a central log store with search/browsing interface.

which platform to choose?

* The platform should be **shared large cluster platform(one platform being shared by all LOB teams)**with effective container management, container rolling updates, resource management, pluggable networking frameworks and cloud agnostic  features - **Kubernetes.**

* The platform should be used for small scale cluster deployments (one cluster per team with small size container deployments), less overhead with platform maintenance and no cloud agnostic requirements - **AWS Managed ECS.**

* The platform should be **shared hybrid larger cluster platform**(not only container  management support , other distributed services management like Spark,Cassandra, Kafka, Redis) with effective clusters management, resource management  - MESOS