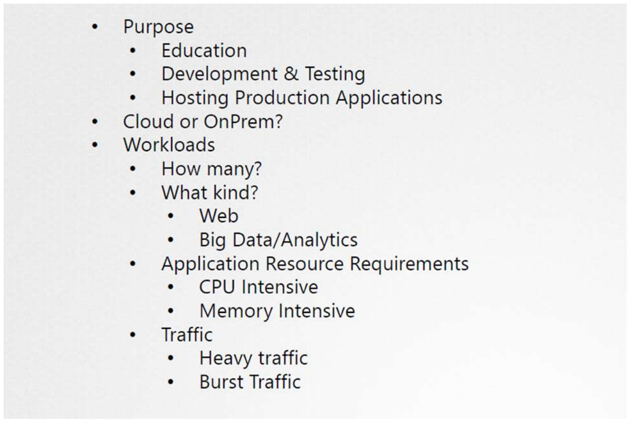
Design a Kubernetes Cluster

Hello and welcome to this lecture. In this lecture, we will discuss designing a Kubernetes cluster. Before you head into designing a cluster, you must ask the following questions:

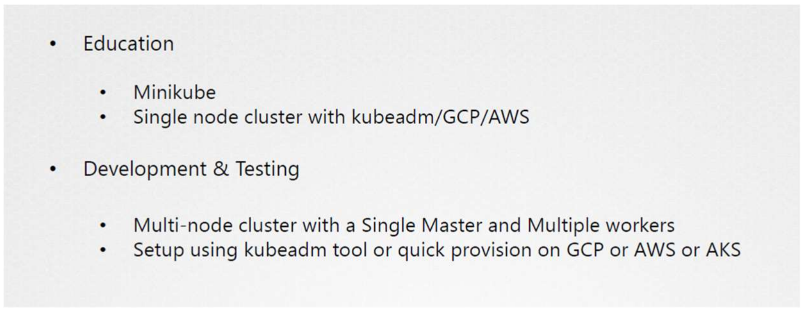
- What is the purpose of this cluster? Is it for learning, development, testing purposes, or for hosting production-grade applications?
- What is the cloud adoption at your organization? Do you prefer your platform to be managed by a cloud provider or self-hosted?
- What kind of workloads are you going to run on this cluster?
- How many applications are to be hosted on the cluster? Few or many?
- What kind of applications are going to be hosted on the cluster? Web applications, big data, or analytics? Depending on the kind of application, the resource requirements may vary.
- What type of network traffic are these applications expecting? Continuous heavy traffic or burst?



- Purpose
  - Education
  - Development & Testing
  - Hosting Production Applications
- Cloud or OnPrem?
- Workloads
  - How many?
  - What kind?
    - Web
    - Big Data/Analytics
  - Application Resource Requirements
    - CPU Intensive
    - Memory Intensive
  - Traffic
    - Heavy traffic
    - Burst Traffic

Well, let's try and break down some of these. If you want to deploy a cluster for learning purposes, then a solution based on Minikube or a single-node cluster deployed using kubeadm on local VMs or cloud providers like GCP or AWS should do. We have deployed such a cluster in the beginner's course.

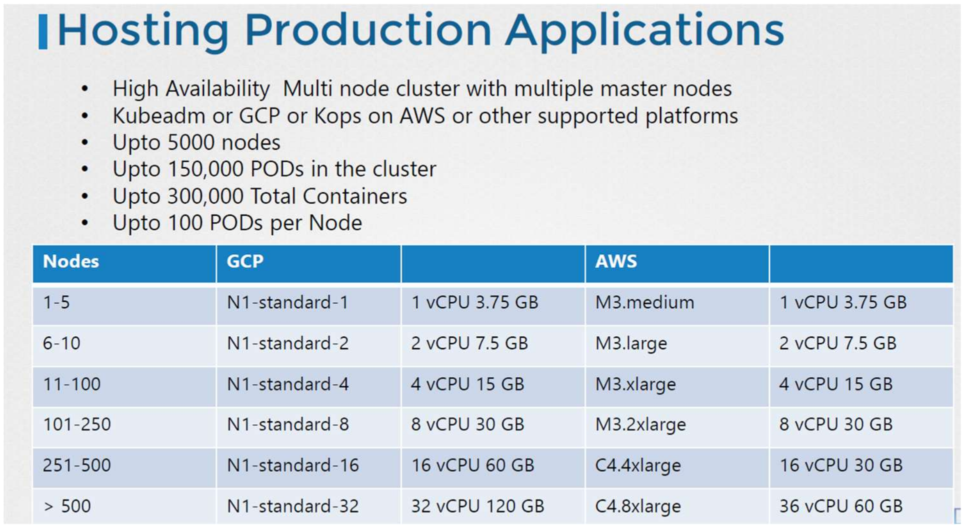
To deploy a cluster for development and testing purposes, a multi-node cluster with a single master and multiple worker nodes would help. Again, kubeadm is an appropriate tool, or if on managed cloud environments, then quickly provision a cluster using the GKE on GCP or EKS on AWS or AKS solution on Azure.



- Education
  - Minikube
  - Single node cluster with kubeadm/GCP/AWS
- Development & Testing
  - Multi-node cluster with a Single Master and Multiple workers
  - Setup using kubeadm tool or quick provision on GCP or AWS or AKS

Let's talk about production-level clusters. For hosting production-grade applications, a highly available multi-node cluster with multiple master nodes is recommended. We look at more details about high availability setup with multiple master nodes later in this session. Again, this can be set up with kubeadm or GCP or using kOps on AWS or other supportive platforms.

You can have up to 5,000 nodes in the cluster, a total of 150,000 pods in the cluster, 300,000 containers in total, and up to 100 pods per node. Now, depending on the size of your cluster, the resource requirement of your node varies. Cloud service providers like GCP and AWS automatically select the right size nodes for you based on the number of nodes in the cluster.

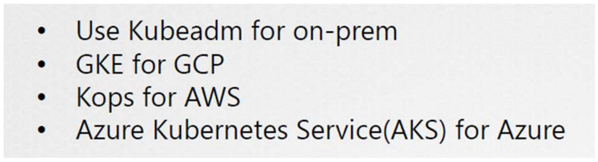


Hosting Production Applications

- High Availability Multi node cluster with multiple master nodes
- Kubeadm or GCP or Kops on AWS or other supported platforms
- Upto 5000 nodes
- Upto 150,000 PODs in the cluster
- Upto 300,000 Total Containers
- Upto 100 PODs per Node

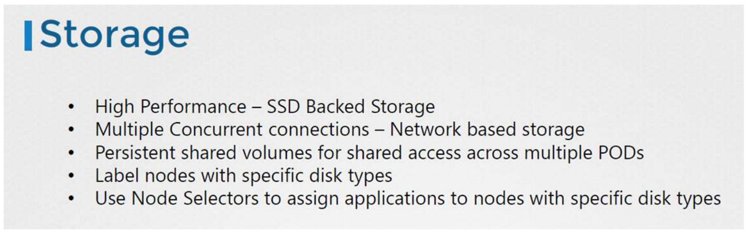
Nodes	GCP		AWS	
1-5	N1-standard-1	1 vCPU 3.75 GB	M3.medium	1 vCPU 3.75 GB
6-10	N1-standard-2	2 vCPU 7.5 GB	M3.large	2 vCPU 7.5 GB
11-100	N1-standard-4	4 vCPU 15 GB	M3.xlarge	4 vCPU 15 GB
101-250	N1-standard-8	8 vCPU 30 GB	M3.2xlarge	8 vCPU 30 GB
251-500	N1-standard-16	16 vCPU 60 GB	C4.4xlarge	16 vCPU 30 GB
> 500	N1-standard-32	32 vCPU 120 GB	C4.8xlarge	36 vCPU 60 GB

This table shows the size of the instances and their resource specifications for a specific number of nodes. If you're deploying on-prem nodes, then you could probably start with these numbers as a base. Cloud or on-prem, we've already discussed that all of these deployment options are available in any environment. For on-prem, kube admin is a very useful tool. Google Container Engine makes provisioning Kubernetes clusters on GCP very easy. It comes with a one-click cluster upgrade feature that makes it very easy to maintain the cluster. kOps is a nice tool to deploy Kubernetes clusters on AWS. And the Azure Kubernetes Service or AKS helps in managing the hosted Kubernetes environment on Azure.



- Use Kubeadm for on-prem
- GKE for GCP
- Kops for AWS
- Azure Kubernetes Service(AKS) for Azure

Depending on the workloads configured, your node and disk configurations will differ. For high-performance workloads, rely on SSD-backed storage. For multiple concurrent access, consider network-based storage. For shared access to volumes across multiple pods, consider persistent storage volumes that we discussed in the storage section. Consider defining different classes of storage and allocating the right class to the right applications



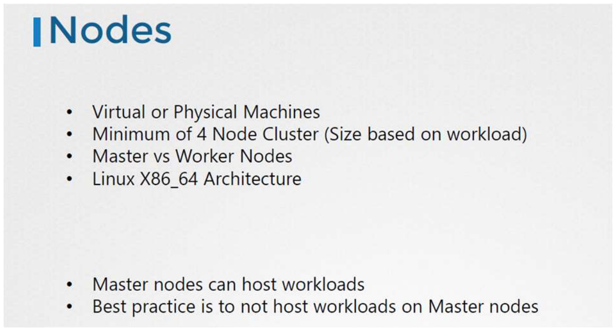
Storage

- High Performance – SSD Backed Storage
- Multiple Concurrent connections – Network based storage
- Persistent shared volumes for shared access across multiple PODs
- Label nodes with specific disk types
- Use Node Selectors to assign applications to nodes with specific disk types

The nodes forming a Kubernetes cluster can be physical or virtual. In our case, we will be deploying virtual machines on VirtualBox environments as nodes of our cluster. You may choose to deploy on physical machines or virtual machines or cloud environments like GCP, AWS, Azure, or any other platform of your choice.

We will be building a cluster with three nodes, one master and two worker nodes. Now, we know that master nodes are for hosting controlling components like API server and ETCD server, and others, while worker nodes are for hosting workloads. However, this is not a strict requirement; master nodes are also considered as nodes and can host workloads. As a best practice, it is recommended to dedicate master nodes for controlling components only, especially in a production environment.

Deployment tools like kubeadm prevent workloads from being hosted on master nodes by adding a taint to the master node. You must use a 64-bit Linux operating system for nodes.



Nodes

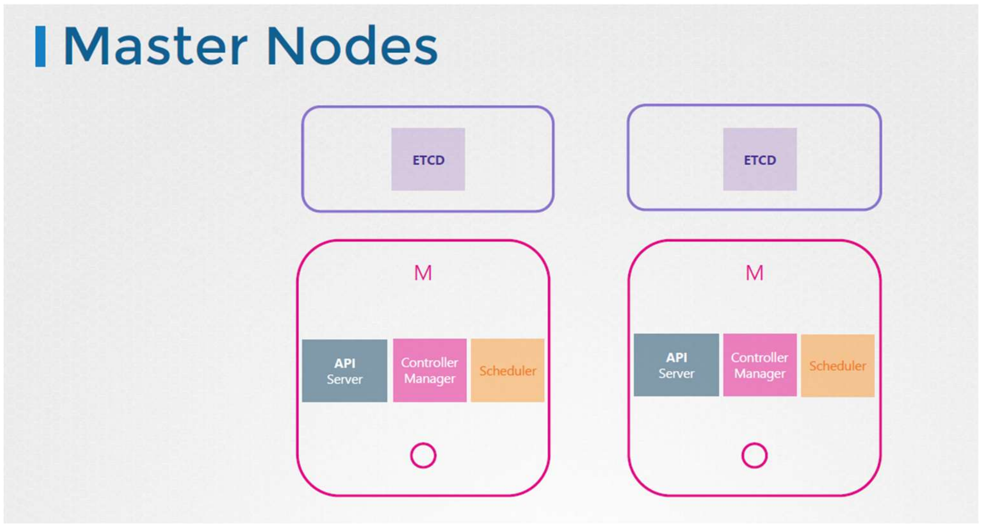
- Virtual or Physical Machines
- Minimum of 4 Node Cluster (Size based on workload)
- Master vs Worker Nodes
- Linux X86_64 Architecture

- Master nodes can host workloads
- Best practice is to not host workloads on Master nodes

Another thing to note is that typically you have all the control plane components on the master nodes. However, in large clusters, you may choose to separate the etcd clusters from the master node to its own cluster nodes. We will discuss more about the different approaches for that in the upcoming lecture when we talk about high-availability setup.

Well, those are some of the considerations for designing a Kubernetes cluster. Refer to the links in the references section for more details and some interesting reads.

Master Nodes


ETCD
ETCD
M
M
API Server
Controller Manager
Scheduler
API Server
Controller Manager
Scheduler