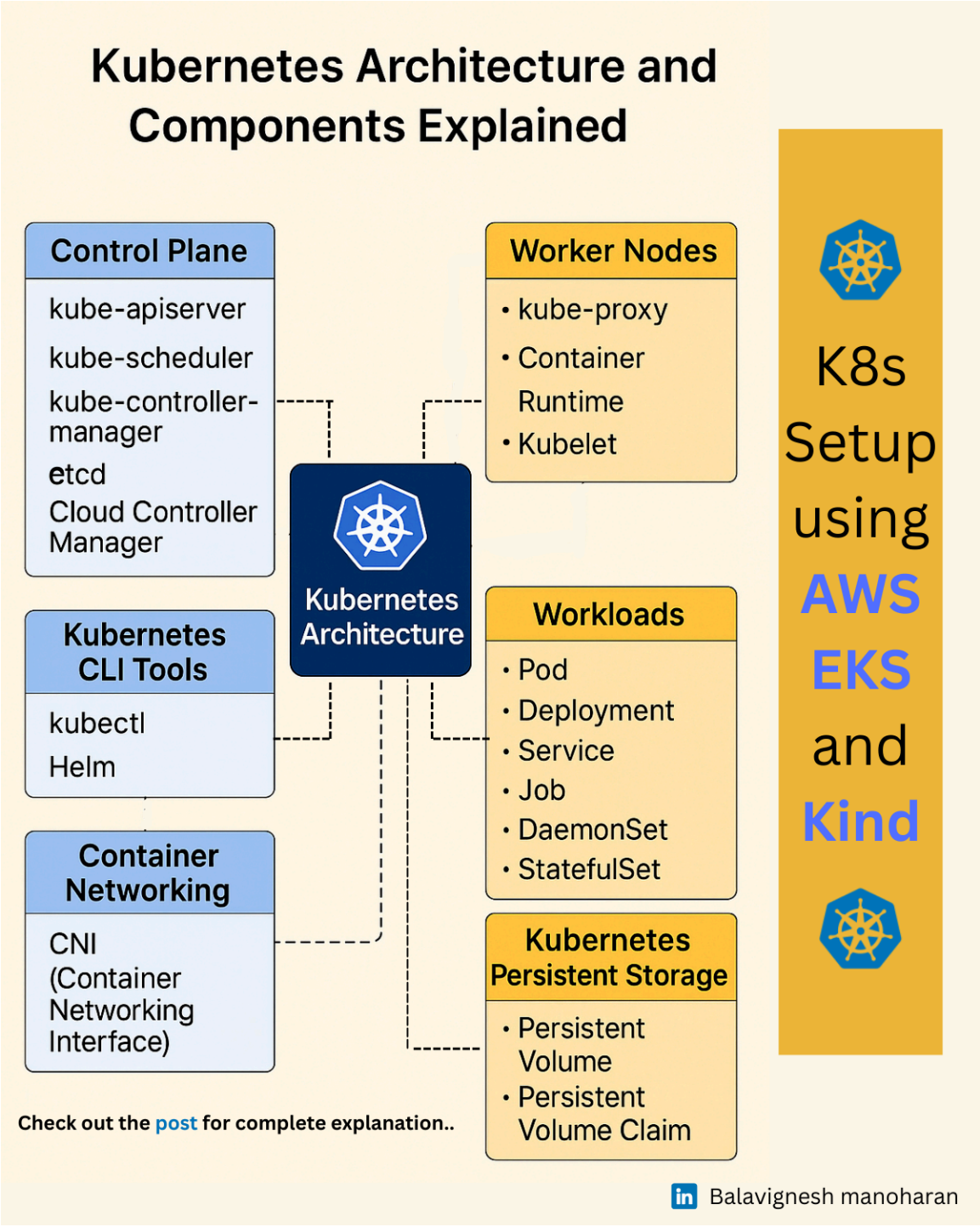
Kubernetes Architecture and Components



Components

1. Control Plane
2. Worker Nodes
3. Workloads
4. Kubernetes Persistent Storage
5. Container Networking
6. Kubernetes CLI Tools
7. Kubernetes Setup

- Kubernetes using AWS Management Console
- Kubernetes using Kind

Control Plane

The control plane consists of several components, we will look into each of them which is responsible for a specific task. They work together to maintain the desired state of the cluster.

- **Kube-apiserver**: This serves as the API server for the kubernetes control plane and a single entry point for all the communications with the cluster. It is responsible for handling external as well as internal requests.
- **etcd**: This is a key-value store which has data about the entire cluster state and its configuration. It is only accessible from the api-server to maintain security.
- **Kube-Controller-manager**: As the name suggests it manages all the controllers in the cluster. It starts their processes and ensures they are operational while the cluster is running.

  Types of controller:

  1. Deployment controller
  2. Replication controller
  3. StatefulSet controller

4. DaemonSet controller

- **Kube-scheduler**: The kube-scheduler is the component which identifies the best worker node that satisfies the pod requirement such as CPU, memory, affinity etc. Once it identifies, then it schedules the pod on the right node.
- **Cloud Controller Manager**: When deploying kubernetes to the cloud environment, it is essential to bridge cloud platform APIs and the kubernetes cluster. This is done using the cloud controller manager. This component is involved whenever you need to provision a load balancer, adding a block storage volume, or creating a virtual machine.

Worker Nodes

Nodes are the physical or virtual machines that host the pods in your cluster. Each worker node can host multiple pods, each containing one or more containers running inside them.

- **Kubelet**: Kubelet is a component which runs on every node in the kubernetes cluster. It creates, modifies and deletes containers for the pod. It executes the actions which are commanded by the control plane. Once the containers are up, kubelet monitors them to ensure they remain healthy.
- **Kube-proxy**: It is a component which facilitates network communication between the nodes in the cluster. The kube proxy helps in maintaining a routing table that maps service IP addresses to the IP addresses on the pods.
- **Container Runtime**: Just like java runtime(JRE) is required to run java programs, container runtime is required to run containers. It manages the entire lifecycle of a container on a host. kubernetes supports multiple container runtime complaint with CRI (Container runtime interface) such as CRI-O, docker engine, containerd.

Workloads

Workloads in kubernetes are resources used to run and manage applications. They handle the complexities of scaling, updating and maintaining the applications lifecycle.

- **Pod**: A pod serves as a single application instance and is considered as the smallest unit in kubernetes. A pod is a group of one or more containers that share the same application.
- **Deployment**: A deployment is used to tell kubernetes how to create, modify instances of the pods that hold a containerized application. It guarantees a certain number of replicas of a pod will be running in the cluster.
- **Service**: Service exposes the Pod to the network. It is an API object which creates a network endpoint for different pods which is then used to communicate to a group of pods.
- **Job**: Kubernetes jobs are a resource within kubernetes that helps schedule and automatically carry out batch tasks.
- **DaemonSet**: A DaemonSet in Kubernetes is a specific kind of workload controller that ensures a copy of a pod runs on either all or some specified nodes
- **StatefulSet**: StatefulSets are Kubernetes objects for running stateful applications in your cluster. A StatefulSet is a set of pods with a unique, persistent hostname and ID. They are essential for managing stateful application that require stable identities and persistent storage.

Kubernetes Persistent Storage

- **Persistent Volume (PV)**: Persistent Volume is a storage in the kubernetes cluster. PV resources belong to the cluster and exist independently of pods. Any disks or data represented by a PV continues existing even as the cluster changes and pods are deleted and recreated.
- **Persistent Volume Claim (PVC)**: A PersistentVolumeClaim (PVC) is a request for storage by a user. The claim can include specific storage parameters required by the application. For example, an amount of storage, or a specific type of access, storageClassName etc.

Container Networking Interface (CNI)

A CNI or Container Networking Interface is a networking plugin for kubernetes. It adds onto the clusters networking capabilities allowing features such as network

policies, load balancing and more. CNI policy was introduced to provide a standard way for creating and managing the network in kubernetes.

Some of the popular CNI

- Calico
- Flannel
- Weave Net

Kubernetes CLI Tools

- **Kubectl**: The primary CLI tool for kubernetes. It allows managing applications and clusters. With the help of kubectl you can inspect cluster resources, create, update and delete components.
- **Helm**: A kubernetes package manager for application installing and managing. It manages the application through Helm charts which define, install and upgrade complex kubernetes application.

Let us dive into Kubernetes Setup

Kubernetes using AWS EKS

Steps:

There are multiple steps involved in the creation of EKS cluster.

- Create an IAM role, and give it necessary permission to manage EKS resources on your behalf.
- Install eksctl, kubectl, AWSCLI to manage the cluster.
- Updating the context to point to the EKS cluster.
- Launching EC2 worker nodes.

Let us go through each step by step.

Pre-requisites:

- AWS Account
- AWS CLI installed

- kubectl installed
- eksctl installed

Create a new EKS cluster using AWS Management Console

1. **Create Control Plane**

- Go to AWS Console and click on the search bar and type EKS and go to the EKS dashboard.
- Click on the Add Cluster and click on create.
- Set name to “my-cluster”
- Select Kubernetes version (use latest stable)
- Select cluster service role and VPC settings
- Click “Create” to launch your EKS cluster
- This will take 10-15 mins to provision the kubernetes control plane.

2. **Adding the worker nodes**

- In your EKS cluster, switch to compute tab. Scroll down and click on “Add Node group” button to create the worker nodes.
- Give it a name, check the node IAM role which you created earlier is selected in the dropdown.
- On the next page, select the EC2 instance configuration. You can keep the defaults as it is.
- Check the remaining pages and create your node group. This may take a while and in some time you can see the new EC2 instances are created.

3. Once worker nodes are created, connect to the cluster using kubectl. Go to your local terminal and run the following command.

```
aws eks update-kubeconfig --name <cluster-name>
```

4. Check the running nodes using the below command.

```
kubectl get nodes
```

5. To create EKS cluster using command line, you need to run the following command.

```
eksctl create cluster \
  --name my-cluster \
  --region us-west-2 \
  --nodegroup-name linux-nodes \
  --node-type t3.medium \
  --nodes 2 \
  --nodes-min 1 \
  --nodes-max 2 \
```

6. To delete the cluster using management console, first you need to go to the compute tab, delete the worker nodes.
7. Then select the cluster and delete the cluster.
8. To delete the EKS cluster using command line.

```
eksctl delete cluster --name my-cluster --region us-west-2
```

This will delete the entire cluster **along with the managed node group** and associated AWS resources created by `eksctl`.

9. And that's it! We walked through creating an EKS cluster using AWS Management Console.

Kubernetes using Kind (Kubernetes in Docker)

Kind is a tool which is designed to run kubernetes clusters in docker containers. It is particularly useful for local development and testing.

Steps:

1. Go to the official documentation here, and run the quick start steps as per your OS to download kind.

2. Once installed, we will create the cluster.
3. You can create kind cluster using the below command.

```
kind create cluster --name k8s
```

4. To check the number of nodes running, run the below command

```
kubectl get nodes
```

5. The advantage of using Kind cluster other than minikube is, using kind you can create multiple clusters. Using the below command you can create another cluster.

```
kind create cluster --name k8s-2
```

6. Now since you have your clusters up and running, you can create a deployment and service.
7. Deleting the kind cluster

You just have to simply enter the following command.

```
kind delete cluster --name [name of the cluster]
```

Finally we saw both creation of kubernetes cluster using AWS EKS as well as Kind cluster.

Hope you got to learn something!

Happy learning

.

.

Give it a like👍

Repost♻️

Follow @Bala Vignesh for more such posts 💯🚀