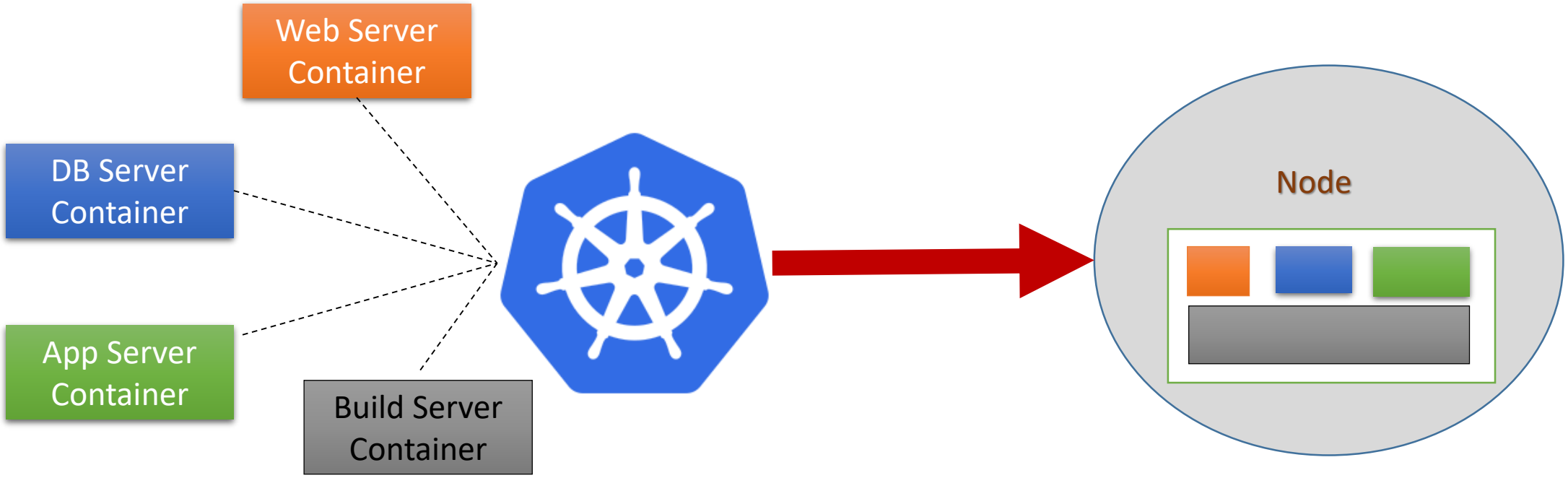
Presented By Raman

Kubernetes

Production-Grade Container Orchestration

Introduction

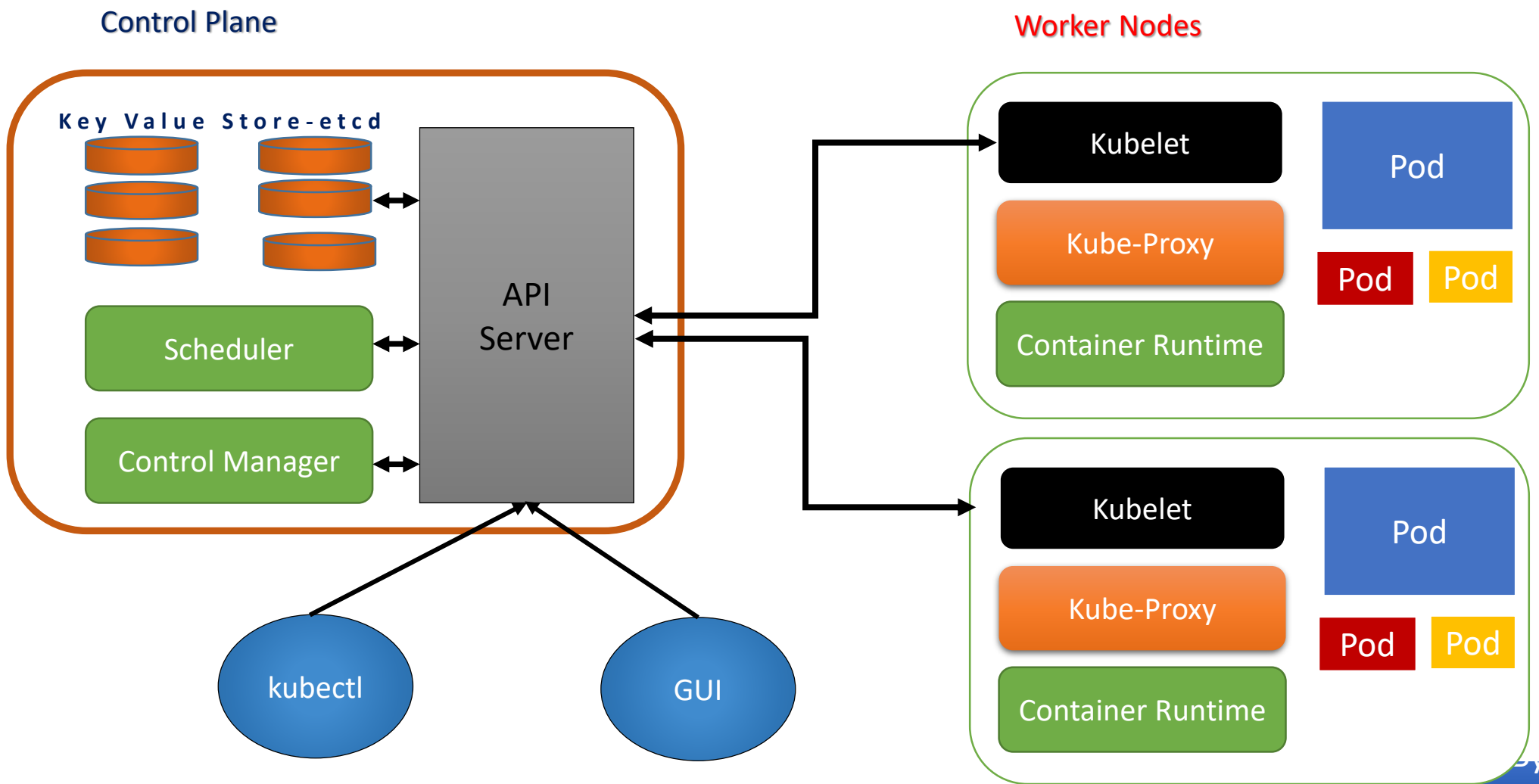
- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications



Kubernetes-Advantages

- Kubernetes can scale without increasing your ops team
- Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.
- Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

Kubernetes Architecture



Control Plane

- **Kube-API Server**

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.
The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

Control Plane

etcd	Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data. If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.
kube-scheduler	Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on. Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

Control Plane

kube-controller-manager	Control Plane component that runs controller processes. Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process. 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

Control Plane

Cloud-Controller Manager	A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that just interact with your cluster. The following controllers can have cloud provider dependencies: Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding Route controller: For setting up routes in the underlying cloud infrastructure Service controller: For creating, updating and deleting cloud provider load balancers

Node Components

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 is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept. kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.
Container RunTime	The container runtime is the software that is responsible for running containers.

Addons

- Addons use Kubernetes resources (DaemonSet, Deployment, etc) to implement cluster features. Because these are providing cluster-level features, namespaced resources for addons belong within the kube-system namespace

Addons	
DNS	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.
WebUI	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 records generic time-series metrics about containers in a central database, and provides a UI for browsing that da