Kubernetes

Introduction to Kubernetes

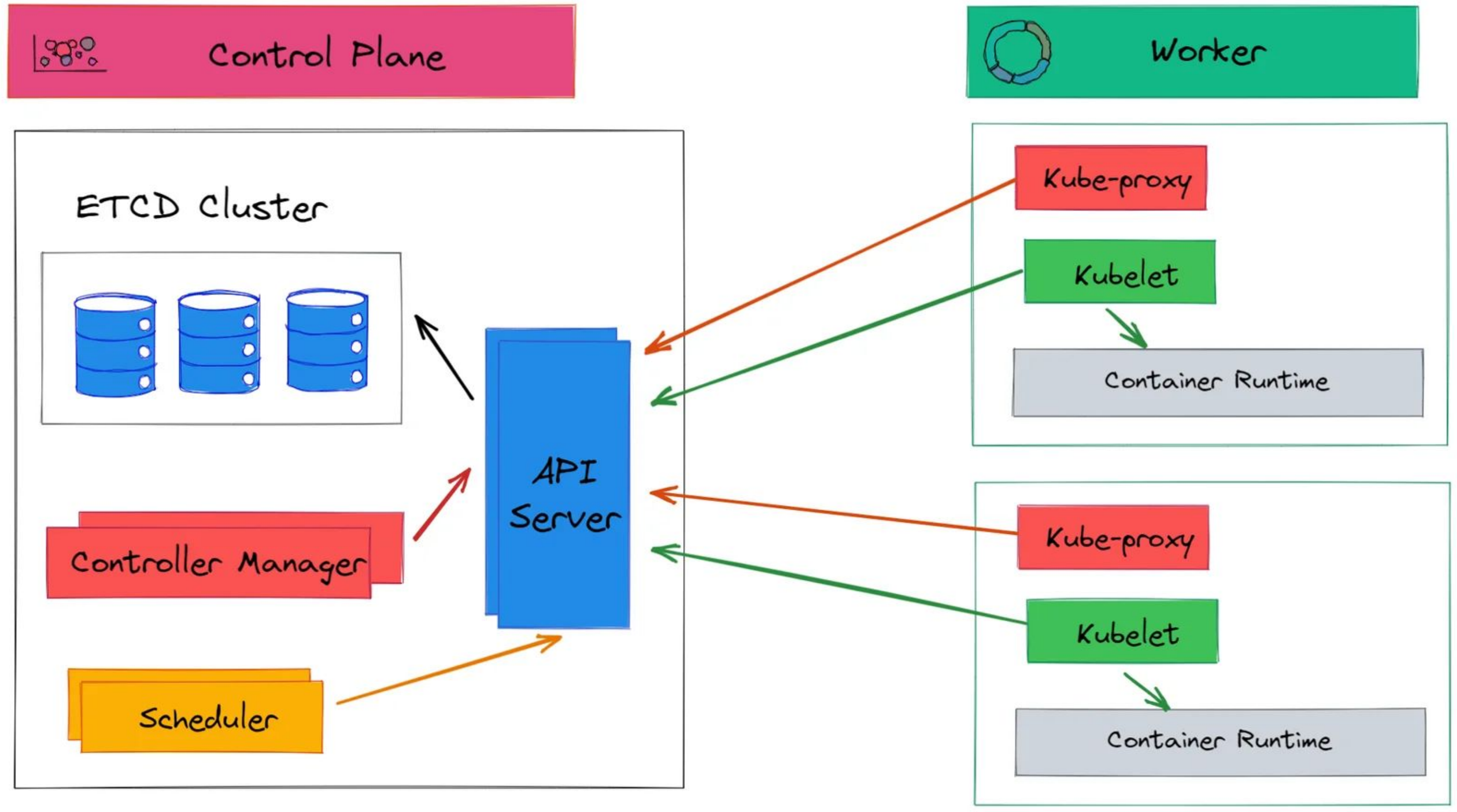
Kubernetes (K8s) is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications.

Why Kubernetes?

- Helps manage containerized applications across multiple hosts.
- Automates manual processes like deployment, scaling, and networking.
- Ensures high availability and load balancing.

Key Features:

- Auto-scaling and self-healing.
- Load balancing.
- Rollbacks and rollouts.


Control Plane
Worker
ETCD Cluster
API Server
Controller Manager
Scheduler
Kube-proxy
Kubelet
Container Runtime
Kube-proxy
Kubelet
Container Runtime

Basic Concepts & Main (K8s) Components

Cluster: A collection of machines (nodes) running Kubernetes.

Nodes:

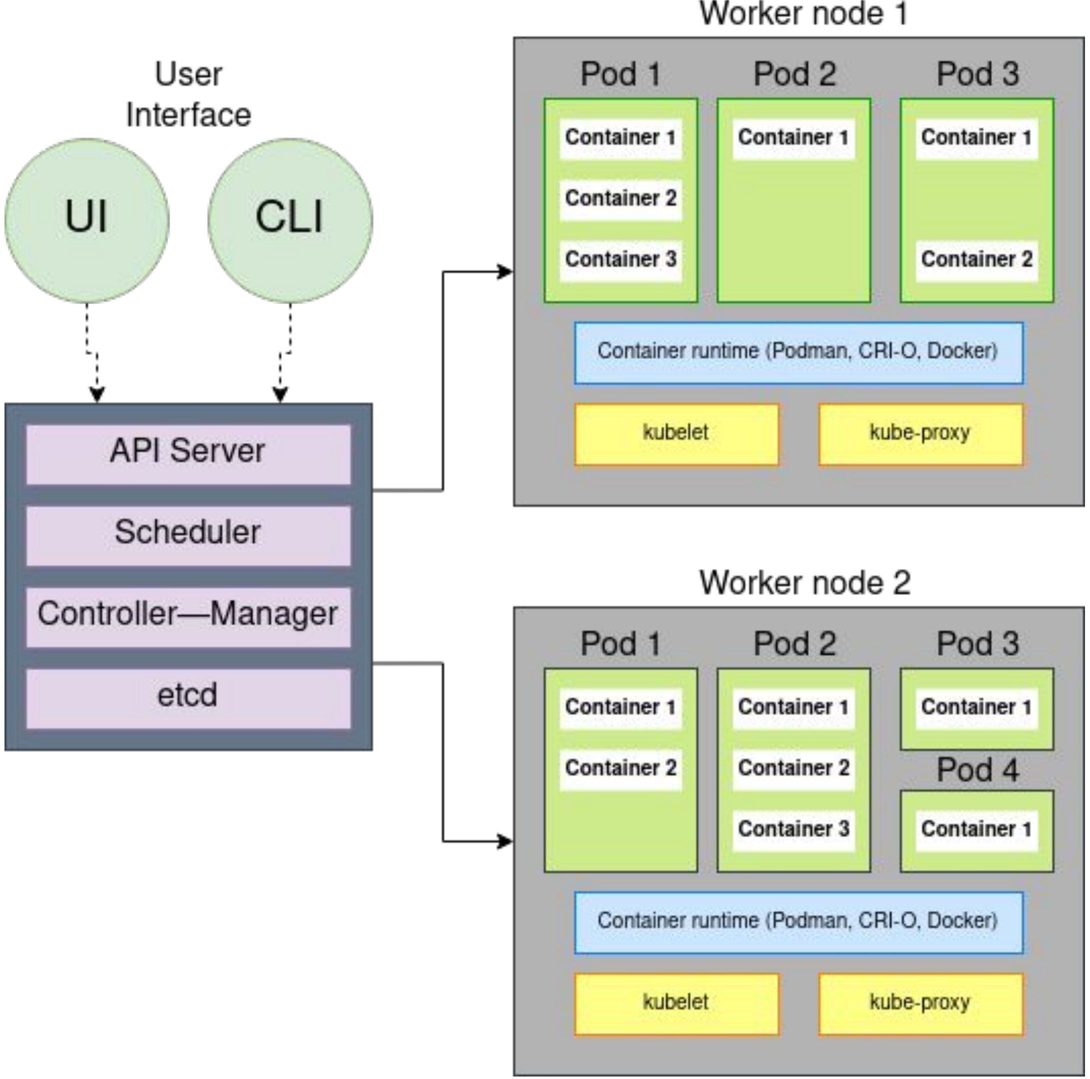
- **Master Node / Control Plane**: Manages the Kubernetes cluster (scheduler, controller, API server).
- **Worker Node**: Runs applications (Pods).

Pods: The smallest deployable unit in Kubernetes, which can contain one or more containers.

Controllers:

- **ReplicationController**: Ensures a specified number of Pods are running.
- **Deployment**: Manages updates and rollbacks for Pods.

Services: Expose a set of Pods as a network service.


User Interface
UI
CLI
API Server
Scheduler
Controller—Manager
etcd
Worker node 1
Pod 1
Pod 2
Pod 3
Container 1
Container 2
Container 3
Container 1
Container 1
Container 2
Container runtime (Podman, CRI-O, Docker)
kubelet
kube-proxy
Worker node 2
Pod 1
Pod 2
Pod 3
Pod 4
Container 1
Container 2
Container 1
Container 2
Container 3
Container 1
Container 1
Container runtime (Podman, CRI-O, Docker)
kubelet
kube-proxy

Kubernetes CLI - Main kubectl Commands

Basic kubectl commands:

- kubectl get nodes: List all nodes in the cluster.
- kubectl get pods: List all running Pods.
- kubectl apply -f <file.yaml>: Apply a configuration file.
- kubectl delete pod <pod-name>: Delete a specific Pod.
- kubectl logs <pod-name>: View logs of a Pod.
- kubectl exec -it <pod-name> -- /bin/sh: Access a shell inside a Pod.

YAML Configuration File in Kubernetes

It is a human-readable format to define resources in Kubernetes. Used for Pods, Services, Deployments, and other objects.

- **Key fields**:
  - **apiVersion**: Defines the API version.
  - **kind**: Type of object (e.g., Pod, Service).
  - **metadata**: Object’s metadata (name, labels).
  - **spec**: Specifies the configuration.

Organizing Components with Namespaces

Namespaces are virtual clusters within a Kubernetes cluster, used to organize and isolate resources.

Default namespaces: default, kube-system, kube-public.

- **Usage**:
  - Useful in multi-tenant environments.
  - Group related resources.
- **Commands**:
  - kubectl create namespace <namespace-name>.
  - kubectl get namespaces.

Kubernetes Service

A Kubernetes Service is an abstraction that defines a logical set of Pods and a policy to access them.

Types of Services:

- **ClusterIP**: Exposes the service within the cluster.
- **NodePort**: Exposes the service on each Node's IP and a static port.
- **LoadBalancer**: Provisions a load balancer for external access.

Usage:

- Connects applications to each other and external clients.
- Provides load balancing across Pods.

Kubernetes Ingress

Ingress:

- Manages external access to services in a cluster, typically HTTP.
- Routes traffic based on URL or hostname.

Kubernetes Benefit

- **Portability**: Runs on any cloud provider or on-premise, supporting hybrid cloud environments.
- **Scalability**: Automatically scales applications up or down based on demand.
- **Self-healing**: Automatically replaces failed containers, restarts them, or rolls back updates.
- **Declarative Management**: Describes the desired state in a YAML file, and Kubernetes will manage and ensure that state.
- **Community Support**: Kubernetes has a large open-source community, providing extensive support and a wide array of plugins and integrations

Kubernetes Use Cases

- Microservices-based applications.
- Continuous integration and continuous deployment (CI/CD).
- Hybrid and multi-cloud deployments.
- High-traffic web applications.

Labs

- Install Minikube: https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download
- Lab Guide: https://docs.google.com/document/d/1ILxiV7GE0Dcc3W1IpN3VkGGUd6EoaNJwIlt3HnXDw8s/edit?usp=sharing
  - Create an Nginx Deployment
  - Create a service to expose the Nginx deployment and make it accessible via a browser.
  - Deploy Nginx using the custom configuration and persistent storage.

Azure Kubernetes Service

Azure Kubernetes Service

Azure Kubernetes Service (AKS) is a managed Kubernetes service that you can use to deploy and manage containerized applications.

You need minimal container orchestration expertise to use AKS. AKS reduces the complexity and operational overhead of managing Kubernetes by offloading much of that responsibility to Azure.

AKS is an ideal platform for deploying and managing containerized applications that require high availability, scalability, and portability, and for deploying applications to multiple regions, using open-source tools, and integrating with existing DevOps tools.

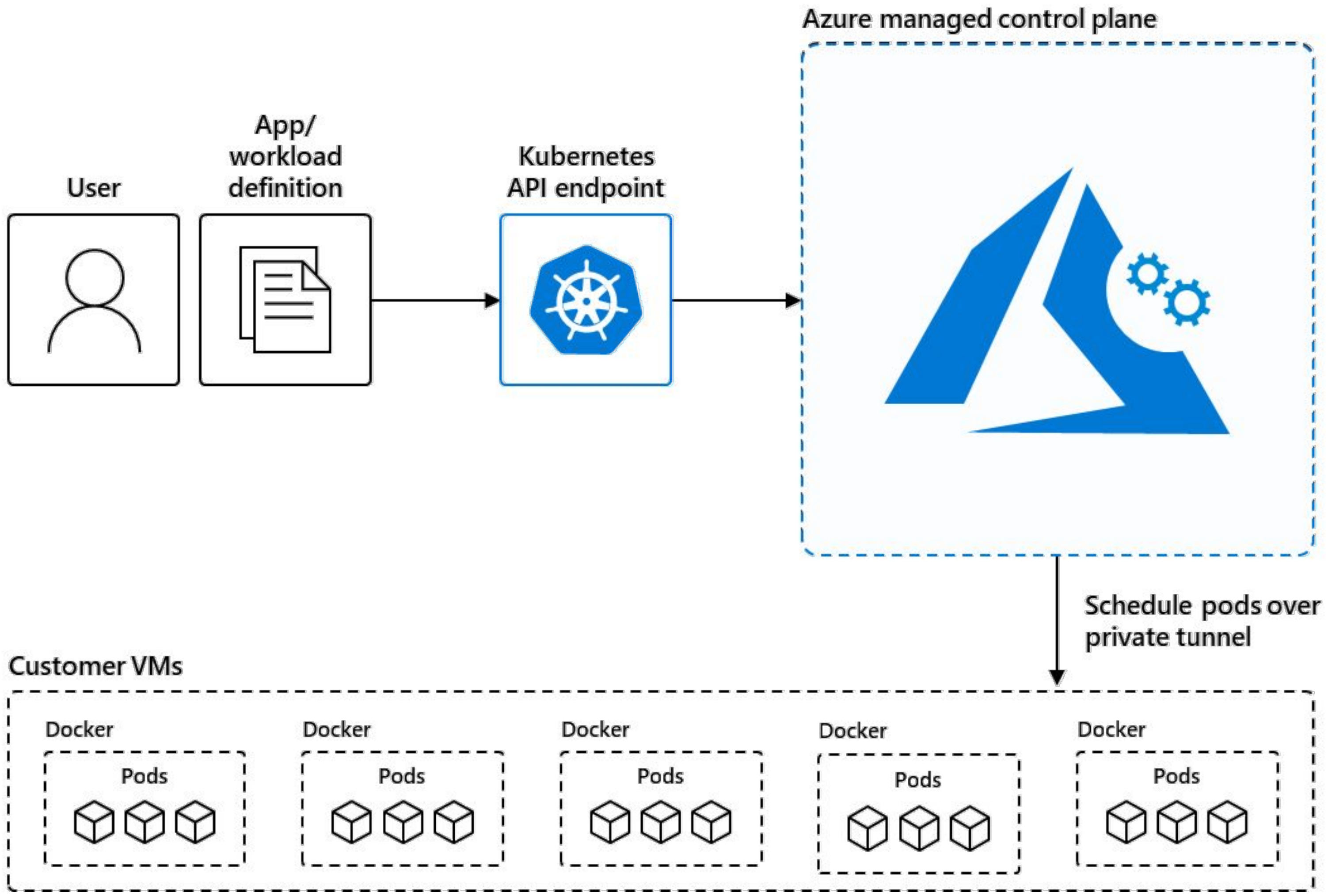
Overview of AKS

AKS reduces the complexity and operational overhead of managing Kubernetes by shifting that responsibility to Azure.

When you create an AKS cluster, Azure automatically creates and configures a control plane for you at no cost.

The Azure platform manages the AKS control plane, which is responsible for the Kubernetes objects and worker nodes that you deploy to run your applications.

Azure takes care of critical operations like health monitoring and maintenance, and you only pay for the AKS nodes that run your applications.


Azure managed control plane
User
App/ workload definition
Kubernetes API endpoint
Schedule pods over private tunnel
Customer VMs
Docker
Pods
Docker
Pods
Docker
Pods
Docker
Pods
Docker
Pods

Summary

AKS simplifies container orchestration in the cloud, allowing for flexible scaling, enterprise-grade security, and full integration with Azure services.

By leveraging AKS, organizations can streamline their development and operations, ensuring that their applications remain highly available, scalable, and secure.

Labs

Lab 1 - Creating and Scaling Workloads on Azure AKS with Azure CLI

https://yourtechie.hashnode.dev/effortless-kubernetes-management-creating-and-scaling-workloads-on-azure-aks-with-azure-cli

Lab 2 - Deploy an Application on AKS Cluster Using a Kubernetes Manifest File

https://docs.google.com/document/d/1KEhSAMunYzsDS_hssvi1T2joolTfnxhrJZtKAas_qx8/edit?usp=sharing

Group Projects

Cloud Infrastructure Setup and Management (Group 1, 3 and 5)

- **Task:** Design and deploy a cloud infrastructure. You can use Infrastructure as Code (IaC) tools like Terraform or ARM templates. Develop a cost management strategy for the cloud environment using tools like Azure Cost Management, Implement budget alerts, resource tagging, and cost optimization techniques.
- **Details:** Include virtual networks, subnets, virtual machines, storage accounts, and load balancers. Implement best practices for security, scalability, and availability

Network Design and Configuration (Group 2 and 4)

- **Task:** Design and configure a secure and scalable cloud network.
- **Details:** Set up virtual networks, subnets, and peering. Implement VPN gateways or ExpressRoute for secure connectivity. Configure network security groups and application gateways. Implement best practices for security, scalability, and availability