### **What is Kubernetes?**

**Kubernetes** (often abbreviated as K8s) is an open-source platform designed for **automating the deployment, scaling, and management of containerized applications**. Originally developed by Google, Kubernetes has become one of the most popular and powerful tools for container orchestration, now maintained by the Cloud Native Computing Foundation (CNCF). It’s widely used to manage modern, cloud-native applications in microservices architecture.

### **Key Features of Kubernetes**

1. **Automated Container Orchestration**Kubernetes automates the process of deploying, managing, and scaling containers, allowing developers to focus more on development rather than infrastructure management.
2. **Scalability**Kubernetes can scale applications both up and down depending on demand. This feature ensures efficient use of resources by automatically adjusting the number of running containers.
3. **Self-Healing**Kubernetes automatically replaces and restarts containers that fail, ensuring high availability. If a container stops responding, Kubernetes will kill it and spin up a new instance to maintain stability.
4. **Service Discovery and Load Balancing**Kubernetes provides built-in service discovery and load balancing, allowing services within the cluster to communicate with each other reliably and efficiently.
5. **Automated Rollouts and Rollbacks**With Kubernetes, you can deploy updates to your application seamlessly. Kubernetes also provides easy rollbacks if something goes wrong with an update, allowing you to revert to a previous stable state.
6. **Configuration Management**Kubernetes separates configuration from the code using ConfigMaps and Secrets. This feature allows for easy changes in configuration without altering the codebase, which helps maintain secure handling of sensitive information.
7. **Persistent Storage Management**Kubernetes offers storage orchestration for containers, allowing you to use external storage solutions and persist data across container restarts and redeployments.
8. **Multi-Cloud and Hybrid Cloud Support**Kubernetes is platform-agnostic, enabling deployment across on-premises, private, or multiple public cloud environments, providing a consistent application environment.

### **Core Components of Kubernetes**

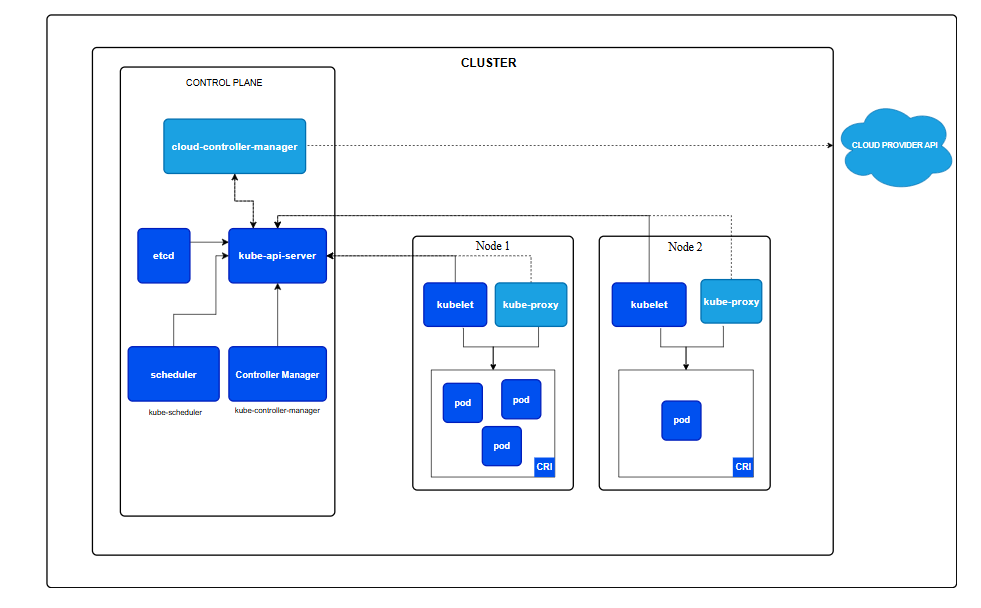
1. **Pods**
   * The smallest deployable unit in Kubernetes, a Pod represents one or more tightly coupled containers. A Pod generally includes one application container (but can have more) and provides a shared networking and storage context.
2. **Nodes**
   * Nodes are the worker machines in Kubernetes, either physical or virtual, where Pods are scheduled and run. Each node runs the necessary services to manage the networking and communication with the Kubernetes control plane.
3. **Control Plane**
   * The control plane is responsible for managing the Kubernetes cluster. Key components include:
     + **API Server:** Serves as the central management point for the entire cluster.
     + **etcd:** A consistent and highly available key-value store used for all cluster data.
     + **Scheduler:** Assigns Pods to nodes based on resource requirements and other constraints.
     + **Controller Manager:** Ensures the cluster’s desired state by managing various controllers for replication, endpoints, and more.
4. **Services**
   * A Service is an abstraction that defines a logical set of Pods and provides a stable endpoint (IP and DNS) for accessing these Pods, even as individual Pods come and go.
5. **Deployments**
   * A Deployment automates the creation and management of Pods. It ensures that a specified number of replicas of a container are running at any time.
6. **ConfigMaps and Secrets**
   * ConfigMaps allow you to decouple configuration details from your Pods, whereas Secrets manage sensitive information such as passwords and API keys.
7. **Ingress**
   * Ingress manages external access to services within the Kubernetes cluster, typically through HTTP/HTTPS. It allows load balancing, SSL termination, and name-based virtual hosting.
8. **Volumes**
   * Kubernetes volumes provide persistent storage for Pods. Unlike ephemeral container storage, volumes allow data to persist across container restarts.

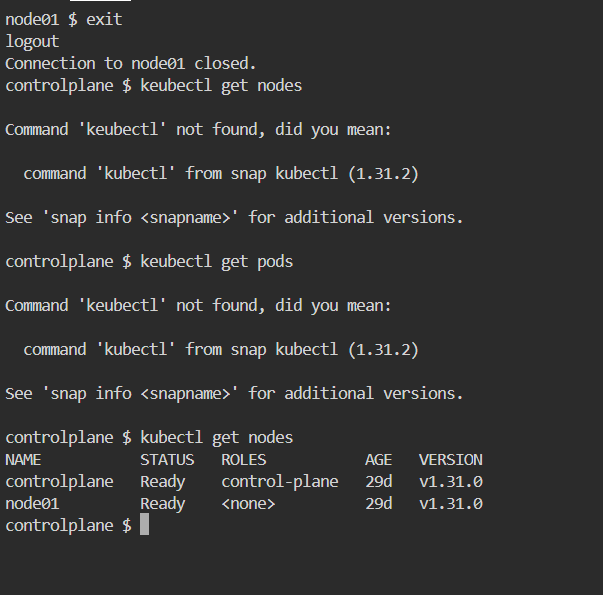
### **How Kubernetes Works**

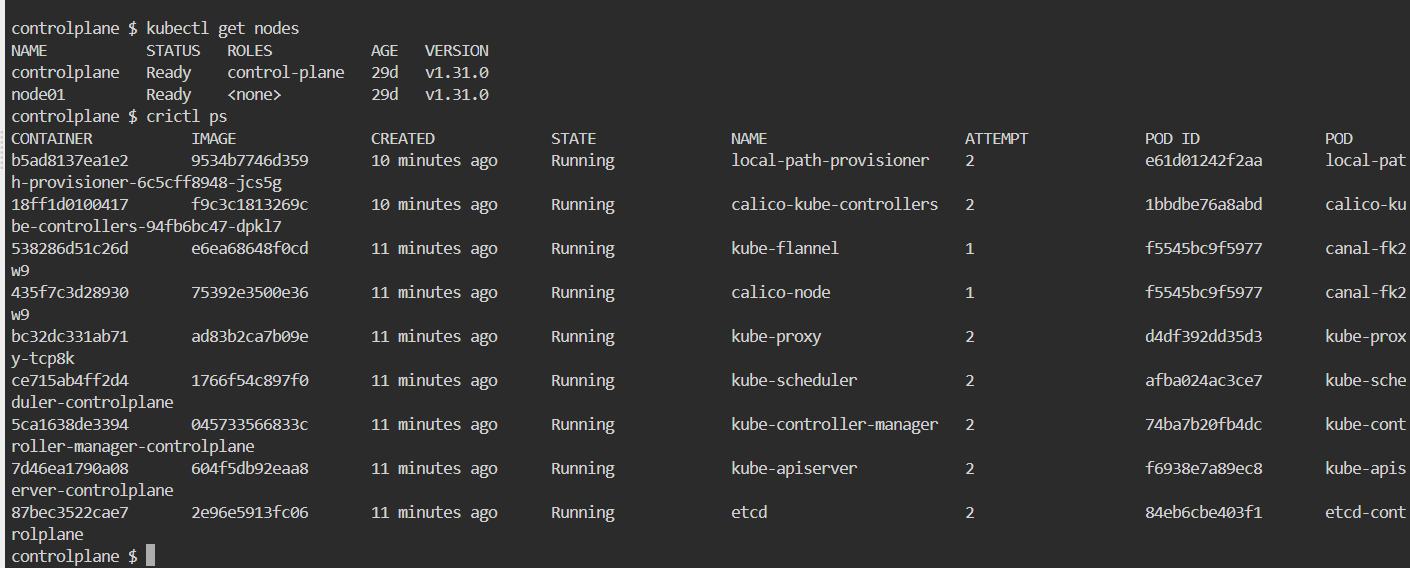
1. **Deployment and Management**Developers describe the desired state of their application using YAML or JSON configuration files, specifying details like the number of replicas, networking, storage, and configuration. Kubernetes then automatically deploys containers to match this configuration, adjusting as necessary.
2. **Networking**Kubernetes uses networking models to facilitate communication between Pods. Each Pod gets its own IP address, allowing direct communication within the cluster.
3. **Scaling and Load Balancing**Based on resource utilization and application load, Kubernetes can automatically scale the number of Pods up or down. It distributes traffic among instances using built-in load balancing, ensuring that application demand is met efficiently.
4. **Self-Healing**Kubernetes continuously monitors the health of Pods and replaces them if they fail, maintaining the desired state and preventing application downtime.

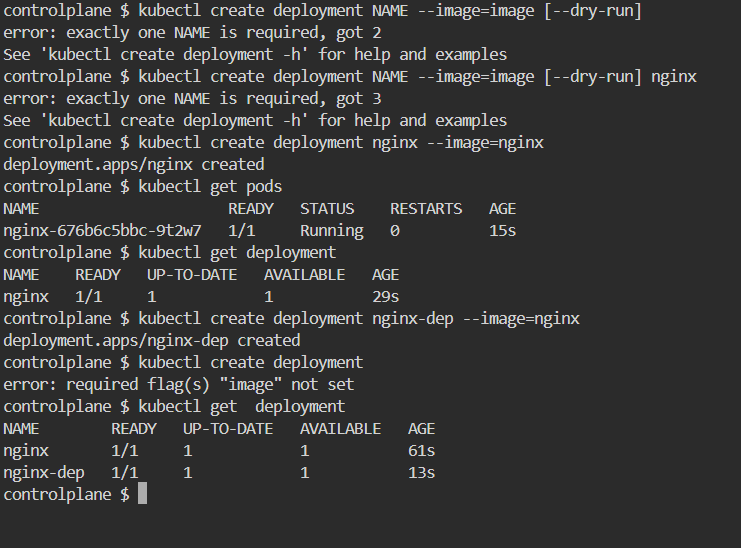
### **Kubernetes Use Cases**

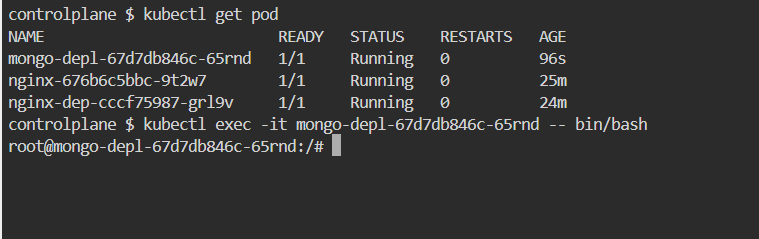
1. **Microservices Architecture**Kubernetes is ideal for managing microservices, as it simplifies the deployment and scaling of independent services within a distributed application.
2. **CI/CD Automation**Kubernetes integrates seamlessly with CI/CD pipelines, enabling automated deployments and updates while providing zero-downtime deployments with rolling updates.
3. **Cloud-Native Applications**Kubernetes enables cloud-native development, allowing applications to run reliably in hybrid and multi-cloud environments.
4. **Big Data and Machine Learning Workloads**Kubernetes supports distributed processing frameworks, making it a fit for big data and ML workloads that require dynamic resource allocation.

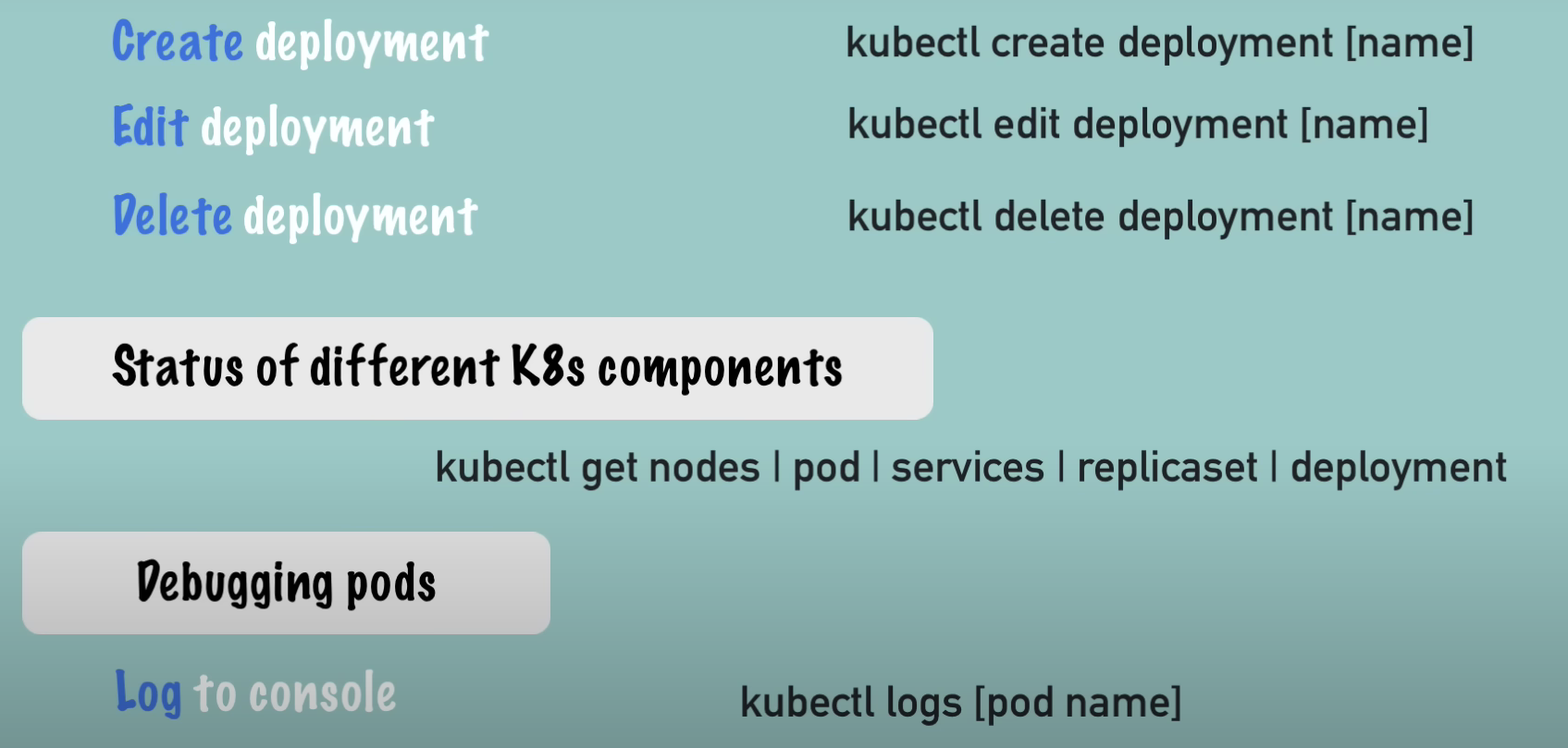


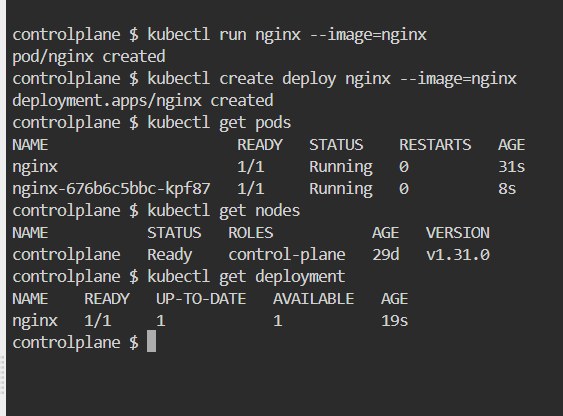


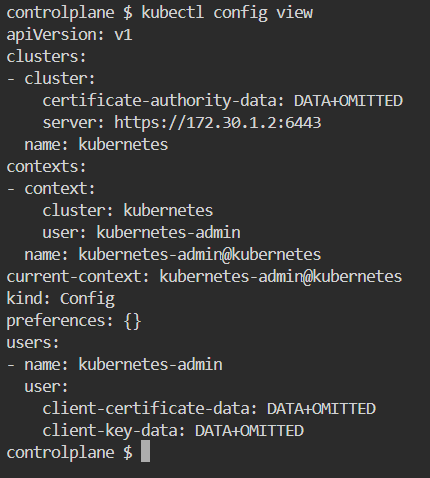


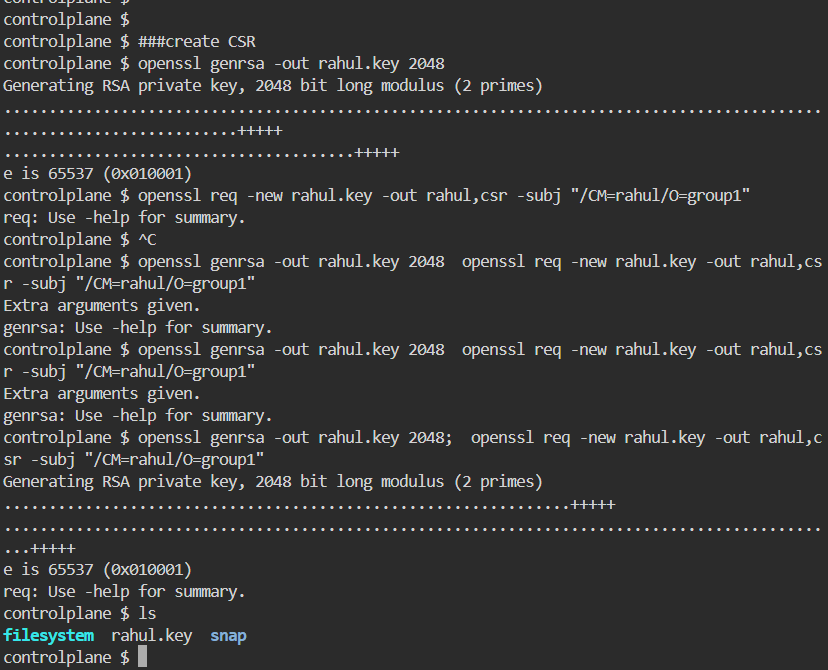


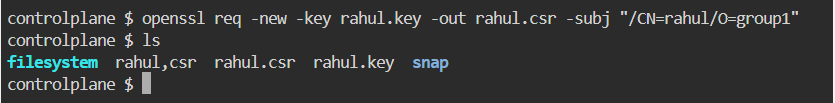


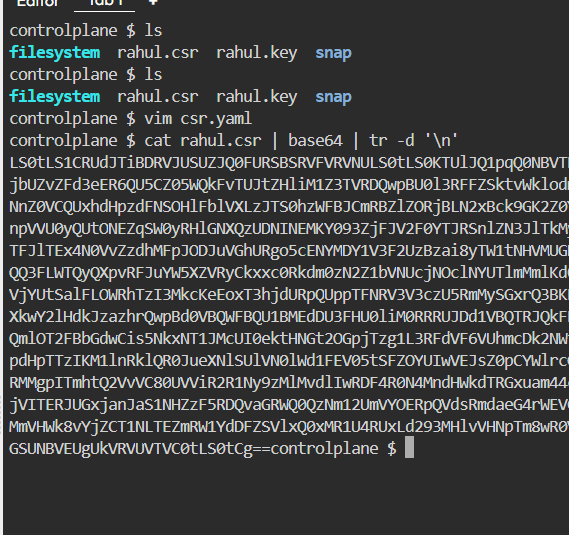


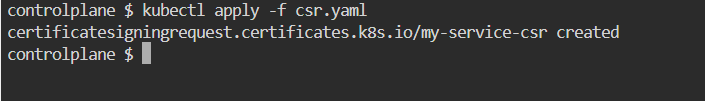












Touch csr.yaml

apiVersion: certificates.k8s.io/v1

kind: CertificateSigningRequest

metadata:

name: rahul

spec:

request: 

signerName: kubernetes.io/kube-apiserver-client

usages:

- digital signature

- key encipherment

- client auth

**Re-apply the CSR**If the CSR is missing, apply the csr.yaml file again to create it:  
  
kubectl apply -f csr.yaml

**Check CSR Creation**Confirm the CSR was created by listing it again:  
  
kubectl get certificatesigningrequests

**Approve the CSR (if needed)**Once the CSR is successfully created, it may need approval:  
bash  
Copy code  
kubectl certificate approve rahul(metadata -name : \*)

Replace "rahul" with the correct CSR name if it differs.

controlplane $

kubectl get csr my-services-csr -o jsonpath='{.status.certificate}' | base64 --decode > rahul.crt

The command you’re running is intended to extract the signed certificate from the CertificateSigningRequest (CSR) once it has been approved and the certificate is issued by Kubernetes.

