**Kubernetes Full Roadmap Notes**

**Module 1: Introduction to Kubernetes**

**What is Kubernetes?**

* Kubernetes (K8s) is an open-source container orchestration platform.
* Manages containerized workloads and services.
* Supports automated deployment, scaling, and operations.

**Why use Kubernetes?**

* Handles scaling, deployment, and operations.
* Provides service discovery, load balancing, storage orchestration.
* Automates rollouts, rollbacks, and self-healing.

**Traditional vs Containerized Deployments**

* Traditional: Apps run on VMs, tightly coupled to OS.
* Containerized: Apps run in containers, isolated and portable.

**Kubernetes vs Docker Swarm**

* Kubernetes: Complex but highly scalable, rich ecosystem.
* Docker Swarm: Simpler setup, less feature-rich.

**Kubernetes Architecture**

* Master Node: Controls and manages the cluster.
* Worker Nodes: Run application workloads.

**Key Components**

* API Server: Exposes Kubernetes API.
* etcd: Stores cluster state.
* Scheduler: Assigns workloads to nodes.
* Controller Manager: Maintains cluster state.
* Kubelet: Node agent.
* Kube-proxy: Handles networking.

**Installing Kubernetes**

* **Minikube**: Local single-node K8s cluster.
* **Kind**: K8s in Docker, useful for CI.
* **K3s**: Lightweight K8s for IoT/Edge.
* **Kubeadm**: Production-grade, manual setup.

**First Kubernetes Deployment**

* Create a Pod or Deployment using kubectl.

**Module 2: Kubernetes Core Components**

**Understanding Nodes, Pods, and Clusters**

* Cluster: Group of nodes (master + workers).
* Node: Single machine in K8s.
* Pod: Smallest deployable unit with one/more containers.

**What is a Pod?**

* A wrapper over containers.
* Shares IP, storage, namespace.

**Creating and Managing Pods**

kubectl run nginx --image=nginx

kubectl get pods

**ReplicaSets and Deployments**

* ReplicaSet ensures specified pod count is maintained.
* Deployment provides declarative updates for Pods and ReplicaSets.

**Labels, Selectors, Annotations**

* Labels: Key-value pairs to organize/select resources.
* Selectors: Filter resources by labels.
* Annotations: Attach metadata to resources.

**Taints & Tolerations, Affinity & Anti-Affinity**

* Taints: Prevent pods from scheduling on certain nodes.
* Tolerations: Allow exceptions to taints.
* Affinity: Control where pods are scheduled.

**Module 3: Configuration & Management**

**Working with YAML Files**

* Define K8s resources in YAML (or JSON).

**Creating Deployments**

kubectl apply -f deployment.yaml

**ConfigMaps & Secrets**

* ConfigMap: Store non-confidential data.
* Secret: Store sensitive data (base64 encoded).

**Environment Variables**

* Set environment variables via ConfigMaps or inline in Pod specs.

**Init Containers & Sidecar Containers**

* Init: Runs before app container.
* Sidecar: Runs alongside main container (e.g., log shipper).

**Module 4: Networking**

**Kubernetes Networking Model**

* Each Pod gets its own IP.
* Flat network, all pods can communicate without NAT.

**Services: ClusterIP, NodePort, LoadBalancer, Ingress**

* ClusterIP: Internal service.
* NodePort: Exposes service on node's IP and static port.
* LoadBalancer: Exposes service via cloud provider's LB.
* Ingress: HTTP/HTTPS routing.

**Ingress Controllers**

* NGINX Ingress is most common.

**Network Policies**

* Define how groups of pods communicate.

**DNS & CoreDNS**

* Internal DNS for service discovery.

**Module 5: Storage & Data Management**

**Persistent Storage**

* Data that persists beyond pod lifecycle.

**PVs & PVCs**

* PV: Provisioned storage.
* PVC: Request for storage by a user.

**Storage Classes & Dynamic Provisioning**

* Automates volume provisioning.

**Stateful Applications**

* Use StatefulSets for identity and storage.

**Storage Types**

* Local, NFS, AWS EBS, GCE PDs, Azure Disks.

**Module 6: Workload Management**

**StatefulSets vs Deployments**

* StatefulSet: Maintains pod identity.
* Deployment: Stateless apps.

**DaemonSets**

* Run a copy of pod on every node.

**Jobs & CronJobs**

* Job: Run-to-completion task.
* CronJob: Scheduled job.

**HPA & VPA**

* HPA: Auto-scale pods based on metrics.
* VPA: Adjust resource requests/limits.

**Resource Requests & Limits**

* Set CPU/memory boundaries per pod.

**Module 7: Security & RBAC**

**RBAC**

* Define access permissions via roles.

**Roles and RoleBindings**

* Role: Permissions within namespace.
* ClusterRole: Cluster-wide.

**Service Accounts**

* Assign identities to pods.

**Securing Secrets**

* Use encryption, restrict access.

**Network & Pod Security Policies**

* Control pod communications and permissions.

**Security Contexts**

* Define privilege, user, etc., per container.

**Image Scanning & Best Practices**

* Use trusted registries, scan images.

**Module 8: Logging & Monitoring**

**Monitoring Clusters**

* Use Prometheus + Grafana.

**kubectl logs & describe**

* Logs: View container logs.
* Describe: View pod configuration/events.

**ELK Stack**

* Elasticsearch, Logstash, Kibana for logging.

**Liveness & Readiness Probes**

* Liveness: Restart unhealthy pods.
* Readiness: Delay traffic until pod is ready.

**Module 9: Service Mesh**

**What is a Service Mesh?**

* Infrastructure layer for service-to-service communication.

**Istio & Linkerd**

* Istio: Feature-rich, complex.
* Linkerd: Lightweight, simple.

**Setting up Istio**

* Install CRDs, deploy control plane.

**Traffic Management**

* Routing, retries, timeouts, etc.

**Security & Observability**

* mTLS, tracing, metrics.

**Module 10: CI/CD & GitOps**

**CI/CD Pipelines**

* Jenkins, GitHub Actions, GitLab CI.

**GitOps**

* Use Git for deployment state.
* Tools: ArgoCD, FluxCD.

**Automating Deployments**

* Use Helm for packaging.

**Helm Charts**

* Pre-configured templates for apps.

**Module 11: Kubernetes in Cloud**

**GKE (Google Kubernetes Engine)**

* Managed K8s on GCP.

**Cloud-Native Apps**

* Design for scalability, resilience.

**Scaling & Load Balancing**

* Use cloud provider services.

**Module 12: Troubleshooting & Debugging**

**Common Errors**

* CrashLoopBackOff, ImagePullBackOff.

**Debugging Tools**

kubectl logs pod-name

kubectl describe pod pod-name

kubectl exec -it pod-name -- bash

**Diagnosing Node/Network Issues**

* Check node status, pod events.

**Best Practices**

* Use readiness/liveness probes.
* Log everything.
* Limit permissions with RBAC.