Kubernetes

Easy Understanding Concepts of Kubernetes:

Notes:

1. **Kubernetes Basics:**
   - **Pods**: Smallest deployable units in Kubernetes that can hold one or multiple containers.
   - **Nodes**: Worker machines in Kubernetes.
   - **Clusters**: A set of worker machines, called nodes, that run containerized applications.
   - **Kubectl**: Command-line tool for interacting with a Kubernetes cluster.
   - **Kubelet**: An agent running on each node in the cluster.
   - **API Server**: Front-end to the cluster's control plane.
   - **Etcd**: Consistent and highly-available key-value store used as Kubernetes' backing store for all cluster data.
   - **Control Plane**: The collection of processes that control Kubernetes nodes.
   - **Namespace**: Virtual clusters backed by the same physical cluster.

2. **Workloads & Controllers:**

   - **Deployments**: Manages a replicated application.
   - **ReplicaSets**: Ensures that a specified number of replicas of a Pod are running at all times.
   - **StatefulSets**: Manages the deployment and scaling of a set of Pods, providing guarantees about ordering and uniqueness.
   - **DaemonSets**: Ensures that all (or some) nodes run a copy of a Pod.
   - **Jobs**: Creates one or more Pods and ensures that a specified number of them successfully terminate.
   - **CronJobs**: Manages time-based Jobs, such as running a Job at a specific time or periodic intervals.
   - **Horizontal Pod Autoscaler**: Automatically scales the number of Pods in a deployment, replica set, or replication controller based on observed CPU or memory usage.

3. **Services & Networking:**

- **Services**: A way to expose an application running in Pods as a network service.
- **Ingress**: Manages external access to services within a cluster.
- **Network Policies**: Define how Pods communicate with each other.
- **Service Discovery**: Mechanism to connect to services dynamically based on a logical name.
- **Load Balancer**: A service that distributes network traffic across multiple Pods.

4. **Configuration & Secrets:**

- **ConfigMaps**: Manage configuration data separately from container images.
- **Secrets**: Manages sensitive information, such as passwords, OAuth tokens, and ssh keys.
- **Environment Variables**: Used within Kubernetes for service discovery.
- **Volumes**: Persistent storage in Kubernetes.
- **Persistent Volume Claims (PVCs)**: Request for storage by a user.

5. **Monitoring, Logging, and Debugging:**

- **Kube-state-metrics**: A simple service that listens to the Kubernetes API server and generates metrics about the state of the objects.
- **Prometheus**: An open-source system monitoring and alerting toolkit.
- **Grafana**: Open platform for beautiful analytics and monitoring (often used with Prometheus).
- **ELK Stack**: Elasticsearch, Logstash, and Kibana used for logging in Kubernetes.
- **Kubernetes Dashboard**: General-purpose web UI for Kubernetes clusters.
- **Kubectl Debug**: Tool for debugging pods.
- **Heapster**: Collects monitoring and performance metrics.

6. **Security & Authorization:**

- **Role-Based Access Control (RBAC)**: Access control system.
- **Security Policies**: Security constraints applied to a Pod.
- **Service Account**: Identity attached to Pods to interact with the Kubernetes API Server.
- **Transport Layer Security (TLS)**: Protocol for privacy and data integrity.

7. **Kubernetes Storage:**

- **Persistent Volumes (PVs)**: Offers storage to the cluster that is independent of Pod life cycles.
- **Storage Classes**: Allow administrators to describe the "classes" of storage offered.

8. **Kubernetes Cluster Maintenance:**

- **Node Maintenance**: Taking a node down for service or replacing a failing node.
- **Cluster Upgrades**: Upgrading the cluster to a newer version.
- **Backup and Disaster Recovery**: Ensuring data continuity and integrity.

9. **Extending Kubernetes:**

- **Custom Resources**: Extension of the Kubernetes API.
- **API Server Extension**: Custom endpoints in the Kubernetes API.
- **Custom Controllers**: Automate handling of Custom Resources.
- **Operators**: Application-specific controllers to create, configure, and manage instances of complex stateful applications.

10. **Advanced Features:**

- **Service Mesh (e.g., Istio)**: Manage microservices in a transparent way.
- **Pod Priority & Preemption**: Specifies priorities for Pods and allows preemption of lower-priority Pods.
- **Taints and Tolerations**: Allows a node to repel a set of Pods.

- **Node Affinity**: Controls where a Pod runs based on labels on nodes and conditions called node affinity rules.
- **Pod Presets**: Injects information like secrets, volume mounts, and environment variables into pods at creation time.

11. **Kubernetes Cloud Integration:**

- **Amazon EKS**: Managed Kubernetes service on AWS.
- **Google Kubernetes Engine (GKE)**: Managed Kubernetes service on Google Cloud.
- **Azure AKS**: Managed Kubernetes service on Azure.

12. **Continuous Deployment/Integration in Kubernetes:**

- **Jenkins**: Popular open-source tool to perform continuous integration and build automation.
- **GitLab CI**: Continuous integration service included with GitLab that builds and tests the software whenever a developer pushes code to the application.

13. **Helm: Kubernetes Package Manager:**

- **Helm Charts**: Packages of pre-configured Kubernetes resources.
- **Helm Repository**: A collection of charts for Kubernetes packages.

14. **Kubernetes Custom Scheduling:**

- **Custom Scheduler**: You can implement a custom scheduler to have more control over the scheduling of your Pods.

15. **Kubernetes Federation:**

- **Cluster Federation**: Allows the synchronization of resources across multiple clusters.

16. **Windows in Kubernetes:**

- **Windows Nodes**: Support for Windows worker nodes and Windows containers in Kubernetes.

17. **Kubernetes Testing:**

- **Kubetest**: Kubernetes end-to-end testing.

18. **Kubernetes Add-ons:**

- **DNS**: DNS server for service name resolution.
- **Web UI (Dashboard)**: Web-based Kubernetes user interface.
- **Container Resource Monitoring**: Time-series monitoring of resource utilization.
- **Cluster-level Logging**: Save container logs to a logging backend.

19. **Other Essential Concepts:**

- **Quotas**: Constraints applied to resources like Pods, Persistent Volumes.
- **Annotations**: Attach arbitrary non-identifying metadata to objects.
- **Labels and Selectors**: Labels are key/value pairs attached to objects and selectors are used to select objects based on labels.
- **Liveness and Readiness Probes**: Health checks for running applications.

20. **Kubernetes Failure Handling:**

- **Pod Disruption Budgets (PDBs)**: Provides constraints that limit voluntary disruptions for Pods.

21. **Kubernetes Architecture Concepts:**

- **Cloud Controller Manager**: Embeds cloud-specific control logic for the cloud provider.
- **Kube-Proxy**: Maintains network rules and enables connection forwarding.
- **Container Runtime**: The software used to run containers, e.g., Docker, containers.

22. **Kubernetes Community and Development:**

- **Special Interest Groups (SIGs)**: The Kubernetes project is divided into several Special Interest Groups or SIGs.
- **Contributing to Kubernetes**: Guide to becoming an active contributor to the Kubernetes project.

23. **Kubernetes Best Practices:**

- **Logging Best Practices**: Guidelines for logging in Kubernetes.
- **Monitoring Best Practices**: Guidelines for monitoring in Kubernetes.
- **Security Best Practices**: Guidelines for securing your applications in Kubernetes.
- **Performance Best Practices**: Guidelines for ensuring optimal performance in Kubernetes.

24. **Kubernetes Automation and Autoscaling:**

- **Cluster Autoscaler**: Automatically adjusts the size of the cluster, scaling it up or down as necessary.
- **Vertical Pod Autoscaler**: Automatically adjusts the amount of CPU and memory requested by containers in a Pod.
- **Horizontal Pod Autoscaler (HPA)**: Automatically scales the number of Pods in a deployment or replica set based on observed CPU or memory utilization.

25. **Kubernetes Development Tools:**

- **Minikube**: Runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.
- **Skaffold**: Command line tool that facilitates continuous development for Kubernetes applications.
- **Kompose**: Conversion tool for all Docker Compose users to help them move to Kubernetes.
- **Kubeadm**: Tool for bootstrapping a best-practice Kubernetes cluster.

26. Advanced Kubernetes Networking:

- **Network Plugins**: Extend Kubernetes networking.
- **CNI (Container Network Interface)**: Standard for writing plugins to configure network interfaces in Linux containers.
- **Flannel**: Overlay network provider.
- **Calico**: Provides secure network connectivity.

27. Kubernetes Performance Tuning:

- **Kubelet Garbage Collection**: Cleanup of unused images or containers.
- **Kernel Tuning**: Adjusting Linux kernel parameters to optimize performance.

28. Kubernetes Service Mesh:

- **Istio**: Open platform to connect, manage, and secure microservices.
- **Linkerd**: Another popular service mesh for Kubernetes.

Kubernetes Commands & Usages:

1- **Basic Cluster Information:**

- `kubectl cluster-info`: Display information about the cluster.
- `kubectl version`: Display version info.

2- **Working with Nodes and Cluster:**

- `kubectl get nodes`: List nodes in a cluster.
- `kubectl describe node <node-name>`: Show details of a specific node.

3- **Working with Pods:**

- `kubectl get pods`: List all pods in all namespaces.
- `kubectl run <name> --image=<image>`: Deploy a new Pod with a given image.
- `kubectl describe pod <pod-name>`: Describe a specific pod.
- `kubectl logs <pod-name>`: Fetch the logs from a pod.
- `kubectl delete pod <pod-name>`: Delete a specific pod.

4- **Working with Deployments:**

- `kubectl create deployment <name> --image=<image>`: Create a new deployment.
- `kubectl get deployments`: List all deployments.
- `kubectl describe deployment <deployment-name>`: Describe a specific deployment.
- `kubectl scale deployment <deployment-name> --replicas=<num>`: Scale up/down a deployment.

5- **Working with Services:**

- `kubectl expose deployment <name> --type=LoadBalancer --port=8080`: Expose a deployment as a service.
- `kubectl get services`: List all services.
- `kubectl describe service <service-name>`: Describe a specific service.

6- **Config and Storage:**

- `kubectl get configmaps`: List all config maps.
- `kubectl create configmap <name> --from-file=<path>`: Create a config map from a file.
- `kubectl get secrets`: List all secrets.
- `kubectl create secret`: Create a secret.
- `kubectl get pv`: List all persistent volumes.
- `kubectl get pvc`: List all persistent volume claims.

7- **Namespaces and Context:**

- `kubectl get namespaces`: List all namespaces.
- `kubectl config get-contexts`: Show all contexts.
- `kubectl config use-context <context-name>`: Switch to a different context.

8- **Others:**

- `kubectl apply -f <filename>`: Apply a configuration from a file.
- `kubectl delete -f <filename>`: Delete resources defined in a file.
- `kubectl exec -it <pod-name> -- /bin/sh`: Execute a command inside a running pod.
- `kubectl port-forward <pod-name> <local-port>:<pod-port>`: Forward a port from a running pod to a local port.

9- **Advanced:**

- `kubectl get all`: List all resources.
- `kubectl rollout status deployment/<deployment-name>`: View the rollout status of a deployment.
- `kubectl rollout history deployment/<deployment-name>`: View the history of a deployment.
- `kubectl rollout undo deployment/<deployment-name>`: Rollback to a previous version of a deployment.

10- Monitoring & Logging:

- `kubectl top nodes`: Display resource (CPU/Memory/Storage) usage of nodes.
- `kubectl top pods`: Display resource (CPU/Memory/Storage) usage of pods.

11- Autoscaling:

- `kubectl autoscale deployment <deployment-name> --min=<min-pods> --max=<max-pods> --cpu-percent=<cpu-util-percentage>`: Auto scale a deployment based on CPU utilization.

12- Working with Helm:

- `helm list`: List releases.
- `helm install <chart>`: Install a helm chart.
- `helm uninstall <release-name>`: Uninstall a helm release.

13- Using Network Policies:

- `kubectl get networkpolicies`: List all network policies.
- `kubectl describe networkpolicy <policy-name>`: Describe a specific network policy.

14- Using CronJobs and Jobs:

- `kubectl get cronjobs`: List all cronjobs.
- `kubectl get jobs`: List all jobs.
- `kubectl logs job/<job-name>`: Fetch logs from a job.

15- Working with RBAC:

- `kubectl get roles`: List all roles in the current namespace.
- `kubectl get clusterroles`: List all cluster roles.

16- Using kubectl Plugins:

- `kubectl krew search`: Search plugins available for kubectl.
- `kubectl krew install <plugin-name>`: Install a kubectl plugin.