Kubernetes Cheat Sheet

A handy reference for essential Kubernetes commands, concepts, and configurations, designed to aid developers and operators in managing containerized applications.

Core Concepts

Pods

Definition: The smallest deployable unit in Kubernetes, representing a single instance of a running process.

- A Pod encapsulates one or more containers, storage resources, a unique network IP, and options that govern how the container(s) should run.
- Pods are ephemeral; they are not designed to be persistent.

Creating a Pod:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx:latest
```

Apply with: `kubectl apply -f pod.yaml`

Common Commands:

- `kubectl get pods` : List all pods.
- `kubectl describe pod <pod-name>` : Get detailed information about a specific pod.
- `kubectl delete pod <pod-name>` : Delete a pod.

Deployments

Definition: A Deployment provides declarative updates for Pods and ReplicaSets.

- It ensures a specified number of pod replicas are running at any given time.
- Deployments support rolling updates and rollbacks.

Creating a Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: httpd:latest
```

Apply with: `kubectl apply -f deployment.yaml`

Common Commands:

- `kubectl get deployments` : List all deployments.
- `kubectl describe deployment <deployment-name>` : Get details about a specific deployment.
- `kubectl scale deployment <deployment-name> --replicas=<number>` : Scale a deployment.
- `kubectl rollout status deployment <deployment-name>` : Check the rollout status.
- `kubectl rollout undo deployment <deployment-name>` : Rollback to the previous version.

Services

Definition: An abstraction which defines a logical set of Pods and a policy by which to access them.

- Services enable loose coupling between dependent Pods.
- Types include ClusterIP, NodePort, LoadBalancer, and ExternalName.

Creating a Service:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: ClusterIP
```

Apply with: `kubectl apply -f service.yaml`

Common Commands:

- `kubectl get services` : List all services.
- `kubectl describe service <service-name>` : Get details about a specific service.
- `kubectl expose deployment <deployment-name> --port=<port> --target-port=<target-port>` : Expose a deployment as a new service.

Namespaces

Definition: Provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces.

- Namespaces allow you to divide cluster resources between multiple users or teams.

Creating a Namespace:

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```

Apply with: `kubectl apply -f namespace.yaml`

Common Commands:

- `kubectl get namespaces` : List all namespaces.
- `kubectl create namespace <namespace-name>` : Create a new namespace.
- `kubectl config set-context --current --namespace=<namespace-name>` : Set the current namespace for kubectl.

Configuration and Storage

ConfigMaps

Definition: A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.

- ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable.

Creating a ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  key1: value1
  key2: value2
```

Apply with: `kubectl apply -f configmap.yaml`

Common Commands:

- `kubectl get configmaps`: List all configmaps.
- `kubectl describe configmap <configmap-name>`: Get details about a specific configmap.
- `kubectl create configmap <configmap-name> --from-literal=key1=value1 --from-literal=key2=value2`: Create a configmap from literals.

Secrets

Definition: A Secret is an API object used to store sensitive information, such as passwords, OAuth tokens, and SSH keys.

- Storing sensitive information in a Secret is safer and more flexible than putting it verbatim in a Pod definition or in a container image.

Creating a Secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  username: $(echo -n 'myuser' | base64)
  password: $(echo -n 'mypassword' | base64)
```

Apply with: `kubectl apply -f secret.yaml`

Note: Data must be base64 encoded.

Common Commands:

- `kubectl get secrets`: List all secrets.
- `kubectl describe secret <secret-name>`: Get details about a specific secret.
- `kubectl create secret generic <secret-name> --from-literal=username=myuser --from-literal=password=mypassword`: Create a generic secret.

Volumes

Definition: A Volume is a directory, possibly with some data in it, which is accessible to the containers in a pod.

- Volumes have a lifetime that is tied to the pod, but can persist data through container restarts.

Volume Types:

- `emptyDir`: A temporary directory that lasts as long as the Pod is running.
- `hostPath`: Mounts a file or directory from the host node's filesystem into your Pod.
- `persistentVolumeClaim`: Used to request storage from a PersistentVolume.

Using a Volume:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx:latest
    volumeMounts:
    - mountPath: /data
      name: my-volume
  volumes:
  - name: my-volume
    emptyDir: {}
```

PersistentVolumes and PersistentVolumeClaims

PersistentVolume (PV): A piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.

PersistentVolumeClaim (PVC): A request for storage by a user. It is a claim on a PV.

Creating a PersistentVolume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/pv
```

Creating a PersistentVolumeClaim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Usage: The PVC is then mounted as a volume in a pod.

Networking

Ingress

Definition: An API object that manages external access to the services in a cluster, typically HTTP.

- Ingress may provide load balancing, SSL termination and name-based virtual hosting.

Creating an Ingress:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-service
            port:
              number: 80
```

Note: Requires an Ingress Controller to be running in the cluster.

Common Commands:

- `kubectl get ingress` : List all ingresses.
- `kubectl describe ingress <ingress-name>` : Get details about a specific ingress.

Network Policies

Definition: An application-centric view of which connections are allowed. They specify how pods are allowed to communicate with each other and other network endpoints.

- Network Policies use labels to select pods and define rules which specify what traffic is allowed to and from the selected pods.

Creating a Network Policy:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-network-policy
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
```

Note: Requires a Network Policy Controller to be running in the cluster.

Common Commands:

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

DNS

Service Discovery: Kubernetes provides internal DNS resolution so pods can discover services by their DNS name.

- Pods can reach services using `<service-name>.<namespace>.svc.cluster.local`.

Example: A service named `my-service` in the `default` namespace can be accessed from within the cluster at `my-service.default.svc.cluster.local`.

Advanced Topics

Helm

Definition: A package manager for Kubernetes, allowing you to define, install, and upgrade even the most complex Kubernetes application.

- Helm uses charts, which are packages of pre-configured Kubernetes resources.

Common Commands:

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

Operators

Definition: Operators are software extensions to Kubernetes that manage applications and their components.

- Operators automate tasks such as deployment, scaling, backups, and upgrades.

Key Concepts: Operators leverage Kubernetes' extensibility to define custom resources and controllers that implement application-specific logic.

Troubleshooting

Common Issues and Commands:

- **Pod Failing to Start:**
  - `kubectl describe pod <pod-name>` : Check events for errors.
  - `kubectl logs <pod-name> -c <container-name>` : View container logs.
- **Service Not Accessible:**
  - `kubectl get endpoints <service-name>` : Verify endpoints are configured correctly.
  - `kubectl describe service <service-name>` : Check service configuration.
- **Node Issues:**
  - `kubectl get nodes` : Check node status.
  - `kubectl describe node <node-name>` : Get node details.