Sure! As a DevOps engineer, you'll be working with Kubernetes often, and it's essential to know the key `kubectl` commands for managing clusters, pods, deployments, services, and more. Below are the most important **Kubernetes commands** you should know.

---

### 1. Basic Kubernetes Commands

These are foundational commands to interact with Kubernetes clusters.

- **`kubectl version`**
  Displays the Kubernetes client and server versions.

  ```
  kubectl version
  ```

- **`kubectl cluster-info`**
  Displays cluster information such as the URL of the Kubernetes control plane and services.

  ```
  kubectl cluster-info
  ```

- **`kubectl config view`**
  Displays the current Kubernetes configuration.

  ```
  kubectl config view
  ```

- **`kubectl get`**
  Lists resources (e.g., pods, services, deployments) in the cluster.

  ```
  kubectl get pods           # Lists all pods in the current namespace
  kubectl get services       # Lists all services
  kubectl get nodes          # Lists all nodes in the cluster
  kubectl get deployments    # Lists all deployments
  kubectl get namespaces     # Lists all namespaces
  kubectl get pods -n <namespace>  # Lists pods in a specific namespace
  ```

- **`kubectl describe`**
  Displays detailed information about a specific resource.

  ```
  kubectl describe pod <pod_name>         # Details about a specific pod
  kubectl describe service <service_name>  # Details about a specific service
  kubectl describe deployment <deployment_name>  # Details about a specific deployment
  ```

- **`kubectl apply`**
  Applies a configuration file (YAML or JSON) to create or update resources.

  ```
  kubectl apply -f <file_name>.yaml
  ```

- **`kubectl delete`**
  Deletes resources such as pods, services, or deployments.

```
kubectl delete pod <pod_name>           # Deletes a specific pod
kubectl delete deployment <deployment_name>  # Deletes a deployment
kubectl delete -f <file_name>.yaml      # Deletes resources defined in a file
```

---

### 2. Working with Pods

Pods are the smallest deployable units in Kubernetes.

- **kubectl get pods**
  Lists all pods in the current namespace.

  ```
  kubectl get pods
  ```

- **kubectl get pods -o wide**
  Shows more details about the pods (including node name, IP addresses).

  ```
  kubectl get pods -o wide
  ```

- **kubectl describe pod <pod_name>**
  Shows detailed information about a specific pod.

  ```
  kubectl describe pod <pod_name>
  ```

- **kubectl logs <pod_name>**
  Fetches logs from a container in a pod.

  ```
  kubectl logs <pod_name>
  kubectl logs <pod_name> -c <container_name>  # For multi-container pods
  ```

- **kubectl exec**
  Executes a command in a running pod's container.

  ```
  kubectl exec -it <pod_name> -- /bin/bash  # Open an interactive bash shell inside the c
  kubectl exec <pod_name> -- ls /app       # Execute a command inside the container
  ```

- **kubectl port-forward**
  Forwards a local port to a port on a pod (useful for debugging).

  ```
  kubectl port-forward <pod_name> <local_port>:<pod_port>
  ```

---

### 3. Working with Deployments

Deployments manage stateless applications and ensure they are running.

- **kubectl get deployments**
  Lists all deployments in the current namespace.

  ```
  kubectl get deployments
  ```

- **kubectl describe deployment <deployment_name>**
  Shows detailed information about a deployment.

```
kubectl describe deployment <deployment_name>
```

- **kubectl scale**
  Scales the number of replicas in a deployment.

  ```
  kubectl scale deployment <deployment_name> --replicas=<number_of_replicas>
  ```

- **kubectl rollout**
  Manages the rollout of a deployment (e.g., status, undo).

  ```
  kubectl rollout status deployment <deployment_name>   # Shows rollout status
  kubectl rollout undo deployment <deployment_name>     # Rolls back the deployment
  kubectl rollout history deployment <deployment_name>  # Shows rollout history
  ```

---

### 4. Working with Services

Services expose your application to network traffic.

- **kubectl get services**
  Lists all services in the current namespace.

  ```
  kubectl get services
  ```

- **kubectl describe service <service_name>**
  Displays detailed information about a specific service.

  ```
  kubectl describe service <service_name>
  ```

- **kubectl expose**
  Exposes a resource (e.g., a pod, deployment, or replicaset) as a new service.

  ```
  kubectl expose pod <pod_name> --port=<port> --target-port=<target_port>  # Exposes a po
  kubectl expose deployment <deployment_name> --port=<port> --target-port=<target_port>
  ```

---

### 5. Working with Namespaces

Namespaces allow you to organize Kubernetes resources into different groups.

- **kubectl get namespaces**
  Lists all namespaces in the cluster.

  ```
  kubectl get namespaces
  ```

- **kubectl create namespace <namespace_name>**
  Creates a new namespace.

  ```
  kubectl create namespace <namespace_name>
  ```

- **kubectl delete namespace <namespace_name>**
  Deletes a namespace and all its resources.

```
kubectl delete namespace <namespace_name>
```

- **kubectl get pods -n <namespace_name>**
  Lists pods in a specific namespace.

  ```
  kubectl get pods -n <namespace_name>
  ```

---

### 6. Working with ConfigMaps and Secrets

ConfigMaps and Secrets are used to store configuration data and sensitive information.

- **kubectl get configmaps**
  Lists all ConfigMaps in the current namespace.

  ```
  kubectl get configmaps
  ```

- **kubectl describe configmap <configmap_name>**
  Displays detailed information about a ConfigMap.

  ```
  kubectl describe configmap <configmap_name>
  ```

- **kubectl create configmap**
  Creates a new ConfigMap from a file or literal value.

  ```
  kubectl create configmap <configmap_name> --from-file=<file_name>  # From a file
  kubectl create configmap <configmap_name> --from-literal=<key>=<value>  # From literal
  ```

- **kubectl get secrets**
  Lists all Secrets in the current namespace.

  ```
  kubectl get secrets
  ```

- **kubectl describe secret <secret_name>**
  Displays detailed information about a Secret (note that secret values are encoded).

  ```
  kubectl describe secret <secret_name>
  ```

---

### 7. Working with Persistent Volumes and Claims

Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) manage storage in Kubernetes.

- **kubectl get pv**
  Lists all Persistent Volumes.

  ```
  kubectl get pv
  ```

- **kubectl get pvc**
  Lists all Persistent Volume Claims.

```
kubectl get pvc
```

- **kubectl describe pv <pv_name>**
  Displays detailed information about a Persistent Volume.

  ```
  kubectl describe pv <pv_name>
  ```

- **kubectl describe pvc <pvc_name>**
  Displays detailed information about a Persistent Volume Claim.

  ```
  kubectl describe pvc <pvc_name>
  ```

---

### 8. Working with Jobs and CronJobs

Jobs and CronJobs are used for running tasks in Kubernetes.

- **kubectl get jobs**
  Lists all jobs in the current namespace.

  ```
  kubectl get jobs
  ```

- **kubectl get cronjobs**
  Lists all CronJobs in the current namespace.

  ```
  kubectl get cronjobs
  ```

- **kubectl describe job <job_name>**
  Displays detailed information about a Job.

  ```
  kubectl describe job <job_name>
  ```

---

### 9. Monitoring and Troubleshooting

These commands are useful for troubleshooting and monitoring your cluster and resources.

- **kubectl top**
  Displays resource usage (CPU and memory) for nodes and pods.

  ```
  kubectl top nodes  # Displays node resource usage
  kubectl top pods   # Displays pod resource usage
  ```

- **kubectl logs**
  Fetches logs for a specific pod (useful for debugging).

  ```
  kubectl logs <pod_name>  # Get logs from the default container in a pod
  kubectl logs <pod_name> -c <container_name>  # Get logs from a specific container in a
  ```

- **kubectl describe**
  Shows detailed information about a resource, including events.

```
kubectl describe pod <pod_name>
kubectl describe deployment <deployment_name>
```

---

### 10. Advanced Commands

These are commands used for more advanced use cases.

- **`kubectl label`**
  Adds or modifies labels on resources.

  ```
  kubectl label pod <pod_name> <label_key>=<label_value>
  ```

- **`kubectl annotate`**
  Adds or modifies annotations on resources.

  ```
  kubectl annotate pod <pod_name> <annotation_key>=<annotation_value>
  ```

- **`kubectl cordon`**
  Marks a node as unschedulable, preventing new pods from being scheduled.

  ```
  kubectl cordon <node_name>
  ```

- **‘kubectl

drain‘**
Evicts all pods from a node (useful before maintenance). `bash kubectl drain <node_name> --ignore-daemonsets`

---

These commands are fundamental for managing Kubernetes resources effectively and troubleshooting issues in your clusters. As a DevOps engineer, being comfortable with these commands will help you automate and maintain deployments, scale applications, and ensure that everything is running smoothly. Let me know if you need more details on any specific command or concept!