Kubernetes Commands:

I. Basic commands

These are the fundamental commands used to start and instantiate resources.

- **Create**
    - **Operation:** creates a new resource explicitly. This is an "imperative" command (you tell kubernetes exactly *what* to do right now). It errors out if the resource already exists.
    - **Usage:** kubectl create -f pod.yaml or kubectl create deployment my-dep --image=nginx
- **Expose**
    - **Operation:** takes an existing controller (like a deployment or pod) and creates a **service** to forward traffic to it. It essentially assigns an ip address or dns name to access your application.
    - Usage: `kubectl expose deployment my-web --port=80 --type=nodeport`
- **Run**
    - **Operation:** the quickest way to start a single pod. It is similar to docker run.
    - **Usage:** kubectl run my-pod --image=nginx
- **Set**
    - **Operation:** updates specific fields on existing objects. Most commonly used to update the docker image version of a deployment without editing the yaml file.
    - **Usage:** kubectl set image deployment/my-dep nginx=nginx:1.19

II. Basic commands

Once resources are running, these commands help you interact with and modify them.

- **Explain**
    - **Operation:** acts as built-in documentation. It shows the structure, fields, and descriptions of kubernetes resources (the api schema).
    - **Usage:** kubectl explain pod.spec.containers
- **Get**
    - **Operation:** lists resources. This is likely the command you will use most. It retrieves the status of pods, services, deployments, etc.
    - **Usage:** kubectl get pods -o wide (shows extra info like ip/node) or kubectl get all
- **Edit**
    - **Operation:** opens the live configuration of a resource in your default text editor (like vim or nano). When you save and quit, kubernetes immediately attempts to apply the changes.
    - **Usage:** kubectl edit svc my-service
- **Delete**
    - **Operation:** removes resources from the cluster. You can delete by filename, resource name, or using label selectors.
    - **Usage:** kubectl delete pod my-pod or kubectl delete -f my-app.yaml

III. Deploy commands

These commands manage the lifecycle and scaling of your applications.

- **Rollout**
    - **Operation:** manages the rollout of resources like deployments. It allows you to check status, view revision history, and importantly, **undo** a deployment (rollback) if a new update fails.
    - **Usage:** kubectl rollout undo deployment/my-app
- **Scale**
    - **Operation:** manages the number of replicas (copies) of your application manually.
    - **Usage:** kubectl scale deployment/my-app --replicas=5
- **Autoscale**
    - **Operation:** creates a horizontal pod autoscaler (hpa). It configures kubernetes to automatically increase or decrease replicas based on cpu or memory usage.
    - **Usage:** kubectl autoscale deployment/my-app --min=2 --max=10 --cpu-percent=80

Iv. Cluster management commands

These are typically used by cluster administrators to manage the health and maintenance of the nodes (servers).

- **Certificate**
    - **Operation:** interacts with certificate signing requests (csrs). Used to approve or deny certificates for new users or nodes joining the cluster.
    - **Usage:** kubectl certificate approve <csr-name>
- **Cluster-info**
    - **Operation:** displays the addresses of the master node and key services (like coredns). Useful for verifying you are connected to the right cluster.
    - **Usage:** kubectl cluster-info
- **Top**
    - **Operation:** displays current resource utilization (cpu and memory) for nodes or pods. *Note: requires metrics server to be installed*.
    - **Usage:** kubectl top pods or kubectl top nodes
- **Cordon**
    - **Operation:** marks a node as "unschedulable." no new pods will be placed there, but existing pods keep running.
    - **Usage:** kubectl cordon node-1
- **Uncordon**
    - **Operation:** reverses cordon. The node becomes available to accept new pods again.
    - **Usage:** kubectl uncordon node-1
- **Drain**
    - **Operation:** prepares a node for maintenance (e.g., os upgrade). It cordons the node *and* evicts (deletes) all pods running on it so they can be rescheduled elsewhere.
    - **Usage:** kubectl drain node-1 --ignore-daemonsets

- **Taint**
    - **Operation:** apply "taints" to a node. Pods cannot be scheduled on a tainted node unless the pod has a matching "toleration."
    - **Usage:** kubectl taint nodes node-1 key=value:noschedule

V. Troubleshooting and debugging commands

Essential for figuring out why an application is failing.

- **Describe**
    - **Operation:** shows detailed configuration and status of a resource. Crucially, it lists **events** at the bottom (e.g., "imagepullbackoff", "failedscheduling"), which is usually where errors are found.
    - **Usage:** kubectl describe pod my-pod-123
- **Logs**
    - **Operation:** prints the standard output (stdout/stderr) of a container. Equivalent to docker logs.
    - **Usage:** kubectl logs my-pod -c my-container --follow
- **Attach**
    - **Operation:** attaches your current terminal shell to the running process inside the container. This is rarely used compared to exec.
    - **Usage:** kubectl attach my-pod -i
- **Exec**
    - **Operation:** runs a command *inside* a container. Most often used to open a shell inside a running pod to look at files or test connectivity.
    - **Usage:** kubectl exec -it my-pod -- /bin/bash
- **Port-forward**
    - **Operation:** tunnels traffic from your local machine (localhost) to a pod inside the cluster. Great for testing internal databases or dashboards without exposing them publicly.
    - **Usage:** kubectl port-forward svc/my-db 5432:5432
- **Proxy**
    - **Operation:** creates a proxy server between your machine and the kubernetes api server. It handles authentication so you can make raw http requests to the api.
    - **Usage:** kubectl proxy
- **Cp**
    - **Operation:** copies files/directories between your local machine and a container in a pod.
    - **Usage:** kubectl cp local-file.txt my-pod:/tmp/remote-file.txt
- **Auth**
    - **Operation:** checks permissions. The can-i subcommand is very useful to verify if a user or service account has rights to perform an action.
    - **Usage:** kubectl auth can-i create deployments

- **Debug**
    - **Operation:** a newer command that creates an "ephemeral container" specifically for debugging. This allows you to inspect a crashed pod (distroless image) by attaching a toolbox container to it.
    - **Usage:** kubectl debug -it my-pod --image=busybox
- **Events**
    - **Operation:** lists recent events in the cluster. Similar to get events but provides a dedicated stream.
    - **Usage:** kubectl events

Vi. Advanced commands

Commands for declarative management and complex workflows.

- **Diff**
    - **Operation:** shows the difference between the configuration you want to apply (in your file) and what is currently running in the cluster.
    - **Usage:** kubectl diff -f deployment.yaml
- **Apply**
    - **Operation:** the standard "declarative" command. It manages applications through files defining the desired state. It creates resources if they don't exist, and updates them if they do.
    - **Usage:** kubectl apply -f deployment.yaml
- **Patch**
    - **Operation:** updates field(s) of a resource using a json merge patch or strategic merge patch. It is good for scripting quick changes.
    - **Usage:** kubectl patch svc my-svc -p '{"spec": {"type": "loadbalancer"}}'
- **Replace**
    - **Operation:** deletes the old resource and creates a new one based on the file provided. It is more destructive than apply.
    - **Usage:** kubectl replace -f pod.yaml --force
- **Wait**
    - **Operation:** pauses the execution of a script until a specific condition is met (e.g., until a pod is "ready"). Essential for ci/cd pipelines.
    - **Usage:** kubectl wait --for=condition=ready pod/my-pod --timeout=60s
- **Kustomize**
    - **Operation:** renders manifest files using kustomize (a configuration management tool built into kubectl). It lets you manage overlays (dev/staging/prod) without templating engines.
    - **Usage:** kubectl kustomize ./overlays/production

Vii. Settings and other commands

- **Label / annotate**: specific commands to add metadata to resources (e.g., kubectl label pods my-pod env=prod).
- **Completion**: generates a shell script (bash/zsh) to enable tab-completion for kubectl commands. (this is highly recommended for productivity).
- **Api-resources**: lists all the resource types (pods, services, cronjobs) available on the server and their short names.
- **Api-versions**: lists which api versions (e.g., v1, apps/v1, batch/v1) are enabled.
- **Config**: modify your local ~/.kube/config file. Used to switch between different clusters (kubectl config use-context my-cluster).