DevOps Shack

100 Kubernetes Commands With Examples

Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps

1. **kubectl create**: This command is used to create Kubernetes resources from files or stdin. For example:

   ```
   kubectl create -f pod.yaml
   ```

   This command creates a pod using the configuration specified in the `pod.yaml` file.

2. **kubectl get**: It is used to retrieve Kubernetes resources. For instance:

   ```
   kubectl get pods
   ```

   This command retrieves all pods in the current namespace.

3. **kubectl describe**: This command provides detailed information about a Kubernetes resource. For example:

   ```
   kubectl describe pod my-pod
   ```

   This command describes the pod named `my-pod`, displaying detailed information including its status, containers, and events.

4. **kubectl apply**: It applies changes to Kubernetes resources defined in YAML or JSON files. For example:

   ```
   kubectl apply -f deployment.yaml
   ```

This command applies the changes specified in the `deployment.yaml` file to the cluster.

5. **kubectl delete**: It is used to delete Kubernetes resources. For instance:

   ```
   kubectl delete pod my-pod
   ```

   This command deletes the pod named `my-pod`.

6. **kubectl exec**: This command executes commands inside a running container in a pod. For example:

   ```
   kubectl exec -it my-pod -- /bin/bash
   ```

   This command starts an interactive shell (`/bin/bash`) inside the pod named `my-pod`.

7. **kubectl logs**: It retrieves the logs of a pod. For instance:

   ```
   kubectl logs my-pod
   ```

   This command displays the logs of the pod named `my-pod`.

8. **kubectl port-forward**: This command forwards one or more local ports to a pod. For example:

   ```
   kubectl port-forward my-pod 8080:80
   ```

   This command forwards local port `8080` to port `80` on the pod named `my-pod`.

9. **kubectl scale**: It scales the number of replicas of a resource. For instance:

   ```
   kubectl scale --replicas=3 deployment/my-deployment
   ```

   This command scales the number of replicas of the deployment named `my-deployment` to 3.

10. **kubectl edit**: This command edits the resource definition in a text editor. For example:

    ```
    kubectl edit pod my-pod
    ```

    This command opens the resource definition of the pod named `my-pod` in a text editor, allowing you to make changes.

11. **kubectl rollout**: This command manages rollouts of updates to Kubernetes resources. For example:

    ```
    kubectl rollout status deployment/my-deployment
    ```

    This command checks the status of the rollout for the deployment named `my-deployment`.

12. **kubectl label**: It adds or updates labels on Kubernetes resources. For instance:

    ```
    kubectl label pod my-pod app=backend
    ```

    This command adds the label `app=backend` to the pod named `my-pod`.

13. **kubectl annotate**: This command adds or updates annotations on Kubernetes resources. For example:

    ```
    kubectl annotate pod my-pod description="This is my pod"
    ```

    This command adds the annotation `description="This is my pod"` to the pod named `my-pod`.

14. **kubectl cluster-info**: It displays cluster info such as server URL and Kubernetes version. For instance:

    ```
    kubectl cluster-info
    ```

    This command displays information about the Kubernetes cluster.

15. **kubectl apply -f -**: This command applies configuration from the standard input. For example:

    ```
    cat pod.yaml | kubectl apply -f -
    ```

    This command applies the configuration defined in `pod.yaml` piped from standard input.

16. **kubectl rollout history**: This command views rollout history of a deployment. For instance:

    ```
    kubectl rollout history deployment/my-deployment
    ```

    This command displays the rollout history of the deployment named `my-deployment`.

17. **kubectl rollout undo**: It rolls back a deployment to a previous revision. For example:

    ```
    kubectl rollout undo deployment/my-deployment
    ```

    This command rolls back the deployment named `my-deployment` to the previous revision.

18. **kubectl create namespace**: This command creates a new Kubernetes namespace. For instance:

    ```
    kubectl create namespace my-namespace
    ```

    This command creates a new namespace named `my-namespace`.

19. **kubectl apply --dry-run**: It simulates the apply of configuration without actually executing it. For example:

    ```
    kubectl apply -f pod.yaml --dry-run=client
    ```

    This command checks if the configuration in `pod.yaml` can be applied without actually applying it.

20. **kubectl api-resources**: This command lists all available API resources. For instance:

    ```
    kubectl api-resources
    ```

    This command lists all the API resources supported by the Kubernetes API server.

21. **kubectl create -f** : This command creates resources defined in all .yaml files in a directory. For example:

    ```
    kubectl create -f ./my-resources/
    ```

    This command creates Kubernetes resources defined in all .yaml files located in the `my-resources` directory.

22. **kubectl get pods -o wide**: It retrieves pods with additional details including node name and IP address. For instance:

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

    This command displays pods along with additional details such as the node they are running on and their IP addresses.

23. **kubectl describe node**: This command provides detailed information about a Kubernetes node. For example:

    ```
    kubectl describe node my-node
    ```

    This command describes the node named `my-node`, displaying detailed information including its capacity, allocatable resources, and conditions.

24. **kubectl rollout pause**: It pauses a rollout of a deployment. For instance:

    ```
    kubectl rollout pause deployment/my-deployment
    ```

    This command pauses the rollout of the deployment named `my-deployment`.

25. **kubectl rollout resume**: This command resumes a paused rollout of a deployment. For example:

    ```
    kubectl rollout resume deployment/my-deployment
    ```

This command resumes the paused rollout of the deployment named `my-deployment`.

26. **kubectl delete namespace**: It deletes a Kubernetes namespace and all resources within it. For instance:

    ```
    kubectl delete namespace my-namespace
    ```

    This command deletes the namespace named `my-namespace` along with all resources within it.

27. **kubectl get events**: This command retrieves events from the cluster. For example:

    ```
    kubectl get events
    ```

    This command retrieves all events from the cluster, displaying information such as type, reason, and message.

28. **kubectl get pods --show-labels**: It displays additional labels associated with pods. For instance:

    ```
    kubectl get pods --show-labels
    ```

    This command displays pods along with all labels associated with them.

29. **kubectl exec -it my-pod -- ls /app**: This command executes a command (`ls /app`) inside a running container in a pod interactively. For example:

    ```
    kubectl exec -it my-pod -- ls /app
    ```

    This command lists the contents of the `/app` directory inside the pod named `my-pod`.

30. **kubectl create secret**: It creates a secret in the cluster. For instance:

    ```
    kubectl create secret generic my-secret --from-literal=username=admin
    --from-literal=password=passw0rd
    ```

    This command creates a secret named `my-secret` with two key-value pairs: `username=admin` and `password=passw0rd`.

31. **kubectl edit deployment**: This command opens the deployment configuration in a text editor, allowing you to make changes. For example:

    ```
    kubectl edit deployment/my-deployment
    ```

    This command opens the configuration of the deployment named `my-deployment` in a text editor.

32. **kubectl rollout restart**: It restarts a rollout of a deployment by reapplying the current configuration. For instance:

```
kubectl rollout restart deployment/my-deployment
```

This command restarts the rollout of the deployment named `my-deployment`.

33. **kubectl rollout status**: This command checks the status of a rollout for a deployment. For example:

    ```
    kubectl rollout status deployment/my-deployment
    ```

    This command checks the status of the rollout for the deployment named `my-deployment`.
34. **kubectl exec -it my-pod -- sh -c 'echo $ENV_VAR'**: This command executes a shell command (`echo $ENV_VAR`) inside a running container in a pod. For instance:

    ```
    kubectl exec -it my-pod -- sh -c 'echo $ENV_VAR'
    ```

    This command prints the value of the environment variable `ENV_VAR` inside the pod named `my-pod`.

35. **kubectl apply -f deployment.yaml --record**: It applies changes to a deployment and records the changes in the revision history. For example:

    ```
    kubectl apply -f deployment.yaml --record
    ```

    This command applies the changes specified in the `deployment.yaml` file to the deployment and records the changes in the revision history.
36. **kubectl get pods --field-selector=status.phase=Running**: This command retrieves pods with a specific status phase, such as `Running`. For instance:

    ```
    kubectl get pods --field-selector=status.phase=Running
    ```

    This command retrieves all pods in the current namespace that are in the `Running` phase.

37. **kubectl delete pod --grace-period=0 --force my-pod**: It forcefully deletes a pod without waiting for the grace period. For example:

    ```
    kubectl delete pod --grace-period=0 --force my-pod
    ```

    This command forcefully deletes the pod named `my-pod` without waiting for the grace period to elapse.

38. **kubectl describe service**: This command provides detailed information about a Kubernetes service. For instance:

    ```
    kubectl describe service my-service
    ```

    This command describes the service named `my-service`, displaying detailed information including its endpoints and selectors.

39. **kubectl create deployment**: It creates a deployment using the specified image. For example:

```
kubectl create deployment my-deployment --image=my-image:tag
```

This command creates a deployment named `my-deployment` using the image `my-image:tag`.

40. **kubectl get deployment -o yaml**: This command retrieves deployments and outputs the result in YAML format. For instance:

    ```
    kubectl get deployment -o yaml
    ```

    This command retrieves all deployments in the current namespace and outputs the result in YAML format.

41. **kubectl scale deployment**: This command scales the number of replicas of a deployment. For example:

```
kubectl scale deployment/my-deployment --replicas=3
```

This command scales the deployment named `my-deployment` to have 3 replicas.

42. **kubectl rollout history deployment**: It displays the revision history of a deployment. For instance:

```
kubectl rollout history deployment/my-deployment
```

This command shows the revision history of the deployment named `my-deployment`.

43. **kubectl rollout undo deployment --to-revision=**: This command rolls back a deployment to a specific revision. For example:

```
kubectl rollout undo deployment/my-deployment --to-revision=3
```

This command rolls back the deployment named `my-deployment` to the third revision.

44. **kubectl apply -f pod.yaml --namespace=**: It applies a YAML file to a specific namespace. For example:

```
kubectl apply -f pod.yaml --namespace=my-namespace
```

This command applies the configuration specified in `pod.yaml` to the namespace `my-namespace`.

45. **kubectl logs -f my-pod**: This command streams the logs of a pod continuously. For instance:

```
kubectl logs -f my-pod
```

This command continuously streams the logs of the pod named `my-pod` to the terminal.

46. **kubectl get svc**: It retrieves information about services in the cluster. For example:

```
kubectl get svc
```

This command retrieves information about all services in the current namespace.

47. **kubectl get pods -n** : This command retrieves pods from a specific namespace. For instance:

```
kubectl get pods -n my-namespace
```

This command retrieves all pods from the namespace `my-namespace`.

48. **kubectl delete -f pod.yaml**: It deletes resources specified in a YAML file. For example:

```
kubectl delete -f pod.yaml
```

This command deletes the resources specified in the `pod.yaml` file.

49. **kubectl rollout status deployment/my-deployment**: This command checks the status of a deployment rollout. For instance:

```
kubectl rollout status deployment/my-deployment
```

This command checks the status of the rollout for the deployment named `my-deployment`.

50. **kubectl exec -it my-pod -- /bin/bash**: This command starts an interactive shell inside a pod. For example:

```
kubectl exec -it my-pod -- /bin/bash
```

This command opens an interactive shell (`/bin/bash`) inside the pod named `my-pod`, allowing you to execute commands within it.

51. **kubectl apply -f --recursive**: This command applies all YAML files in a directory and its subdirectories. For example:

```
kubectl apply -f ./my-resources/ --recursive
```

This command applies all YAML files located in the `my-resources` directory and its subdirectories.

52. **kubectl rollout history deployment/my-deployment --revision=3**: It displays details of a specific revision in the rollout history of a deployment. For instance:

```
kubectl rollout history deployment/my-deployment --revision=3
```

This command shows details of the third revision in the rollout history of the deployment named `my-deployment`.

53. **kubectl rollout undo deployment/my-deployment --to-revision=2**: This command rolls back a deployment to a specific revision. For example:

```
kubectl rollout undo deployment/my-deployment --to-revision=2
```

This command rolls back the deployment named `my-deployment` to the second revision.

54. **kubectl apply -f pod.yaml --validate**: It validates the configuration file before applying changes. For instance:

```
kubectl apply -f pod.yaml --validate=true
```

This command validates the `pod.yaml` file before applying changes to the cluster.

55. **kubectl logs my-pod --tail=100**: This command retrieves the last 100 lines of logs from a pod. For example:

```
kubectl logs my-pod --tail=100
```

This command retrieves the last 100 lines of logs from the pod named `my-pod`.

56. **kubectl get services -o wide**: It retrieves services with additional details including node port and cluster IP. For instance:

```
kubectl get services -o wide
```

This command retrieves services along with additional details such as node port and cluster IP.

57. **kubectl get pods --field-selector=status.phase!=Running**: This command retrieves pods with a status phase other than `Running`. For example:

```
kubectl get pods --field-selector=status.phase!=Running
```

This command retrieves all pods in the current namespace that are not in the `Running` phase.

58. **kubectl delete pod my-pod --force --grace-period=0**: It forcefully deletes a pod without waiting for the grace period. For example:

```
kubectl delete pod my-pod --force --grace-period=0
```

This command forcefully deletes the pod named `my-pod` without waiting for the grace period to elapse.

59. **kubectl describe service my-service**: This command provides detailed information about a Kubernetes service. For instance:

```
kubectl describe service my-service
```

This command describes the service named `my-service`, displaying detailed information including its endpoints and selectors.

60. **kubectl expose deployment my-deployment --type=LoadBalancer --port=80 --target-port=8080**: It exposes a deployment as a service with a specified type, port, and target port. For example:

```
kubectl expose deployment my-deployment --type=LoadBalancer --port=80 --
target-port=8080
```

This command exposes the deployment named `my-deployment` as a LoadBalancer service on port 80, targeting port 8080 on the pods.

61. **kubectl get deployments -l app=my-app**: This command retrieves deployments labeled with `app=my-app`. For example:

```
kubectl get deployments -l app=my-app
```

This command retrieves all deployments labeled with `app=my-app`.

62. **kubectl rollout pause deployment/my-deployment**: It pauses the rollout of a deployment. For instance:

```
kubectl rollout pause deployment/my-deployment
```

This command pauses the rollout of the deployment named `my-deployment`.

63. **kubectl rollout resume deployment/my-deployment**: This command resumes the rollout of a deployment. For example:

```
kubectl rollout resume deployment/my-deployment
```

This command resumes the paused rollout of the deployment named `my-deployment`.

64. **kubectl logs my-pod --container=nginx**: It retrieves logs from a specific container within a pod. For instance:

```
kubectl logs my-pod --container=nginx
```

This command retrieves logs from the container named `nginx` within the pod `my-pod`.

65. **kubectl apply -f pod.yaml --dry-run=client**: This command validates the configuration file without actually applying changes. For example:

```
kubectl apply -f pod.yaml --dry-run=client
```

This command checks if the configuration in `pod.yaml` can be applied without actually applying it.

66. **kubectl get pods --sort-by=.metadata.creationTimestamp**: It retrieves pods sorted by creation timestamp. For instance:

```
kubectl get pods --sort-by=.metadata.creationTimestamp
```

This command retrieves all pods in the current namespace sorted by their creation timestamp in ascending order.

67. **kubectl describe persistentvolumeclaim my-pvc**: This command provides detailed information about a persistent volume claim. For example:

```
kubectl describe persistentvolumeclaim my-pvc
```

This command describes the persistent volume claim named `my-pvc`, displaying detailed information including its status and storage class.

68. **kubectl rollout status deployment/my-deployment --watch**: It continuously monitors the status of a deployment rollout. For instance:

```
kubectl rollout status deployment/my-deployment --watch
```

This command continuously monitors the status of the rollout for the deployment named `my-deployment`.

69. **kubectl get pods --field-selector=status.phase=Pending**: This command retrieves pods with a status phase of `Pending`. For example:

```
kubectl get pods --field-selector=status.phase=Pending
```

This command retrieves all pods in the current namespace that are in the `Pending` phase.

70. **kubectl create secret generic my-secret --from-file=./my-secret-file**: It creates a generic secret from a file. For instance:

```
kubectl create secret generic my-secret --from-file=./my-secret-file
```

This command creates a generic secret named `my-secret` from the contents of the file `my-secret-file`.

71. **kubectl rollout restart deployment/my-deployment**: This command restarts a rollout of a deployment by reapplying the current configuration. For example:

```
kubectl rollout restart deployment/my-deployment
```

This command restarts the rollout of the deployment named `my-deployment`.

72. **kubectl label namespace my-namespace env=dev**: It adds a label to a namespace. For instance:

```
kubectl label namespace my-namespace env=dev
```

This command adds the label `env=dev` to the namespace named `my-namespace`.

73. **kubectl delete deployment my-deployment**: This command deletes a deployment. For example:

```
kubectl delete deployment my-deployment
```

This command deletes the deployment named `my-deployment`.

74. **kubectl get pods --namespace=my-namespace**: It retrieves pods from a specific namespace. For instance:

```
kubectl get pods --namespace=my-namespace
```

This command retrieves all pods from the namespace `my-namespace`.

75. **kubectl describe secret my-secret**: This command provides detailed information about a secret. For example:

```
kubectl describe secret my-secret
```

This command describes the secret named `my-secret`, displaying detailed information including its type and data.

76. **kubectl delete service my-service**: It deletes a service. For instance:

```
kubectl delete service my-service
```

This command deletes the service named `my-service`.

77. **kubectl get nodes**: This command retrieves information about nodes in the cluster. For example:

```
kubectl get nodes
```

This command retrieves information about all nodes in the cluster.

78. **kubectl create configmap my-config --from-literal=key1=value1 --from-literal=key2=value2**: It creates a config map from literal values. For instance:

```
kubectl create configmap my-config --from-literal=key1=value1 --from-
literal=key2=value2
```

This command creates a config map named `my-config` with two key-value pairs: `key1=value1` and `key2=value2`.

79. **kubectl rollout history deployment/my-deployment --revision=3**: This command displays details of a specific revision in the rollout history of a deployment. For example:

```
kubectl rollout history deployment/my-deployment --revision=3
```

This command shows details of the third revision in the rollout history of the deployment named `my-deployment`.

80. **kubectl top pods**: It displays resource usage (CPU and memory) of pods in the cluster. For instance:

```
kubectl top pods
```

This command displays resource usage of all pods in the cluster.

81. **kubectl explain pod**: This command provides documentation about the Pod resource, including all its fields and their descriptions. For example:

```
kubectl explain pod
```

This command displays detailed documentation about the Pod resource.

82. **kubectl delete namespace my-namespace**: It deletes a namespace and all resources within it. For instance:

```
kubectl delete namespace my-namespace
```

This command deletes the namespace named `my-namespace` along with all resources within it.

83. **kubectl get pv**: This command retrieves information about persistent volumes in the cluster. For example:

```
kubectl get pv
```

This command retrieves information about all persistent volumes in the cluster.

84. **kubectl rollout status deployment/my-deployment --timeout=2m**: It checks the status of a rollout and waits for a specific timeout before exiting. For instance:

```
kubectl rollout status deployment/my-deployment --timeout=2m
```

This command checks the status of the rollout for the deployment named `my-deployment` and waits for a maximum of 2 minutes.

85. **kubectl apply -f pod.yaml --namespace=my-namespace**: This command applies a configuration file to a specific namespace. For example:

```
kubectl apply -f pod.yaml --namespace=my-namespace
```

This command applies the configuration specified in `pod.yaml` to the namespace `my-namespace`.

86. **kubectl get secrets**: It retrieves information about secrets in the cluster. For instance:

```
kubectl get secrets
```

This command retrieves information about all secrets in the current namespace.

87. **kubectl create service nodeport my-service --tcp=80:8080**: This command creates a NodePort service to expose a deployment on a specific port. For example:

```
kubectl create service nodeport my-service --tcp=80:8080
```

This command creates a NodePort service named `my-service` to expose a deployment on port 8080.

88. **kubectl rollout undo deployment/my-deployment --to-revision=2 --dry-run**: It simulates rolling back a deployment to a specific revision without actually performing the rollback. For example:

```
kubectl rollout undo deployment/my-deployment --to-revision=2 --dry-run
```

This command simulates rolling back the deployment named `my-deployment` to the second revision without actually performing the rollback.

89. **kubectl create -f pod.yaml --dry-run=client**: This command validates a configuration file without actually creating the resource. For example:

```
kubectl create -f pod.yaml --dry-run=client
```

This command validates the configuration in `pod.yaml` without actually creating the pod.

90. **kubectl exec -it my-pod --container=my-container -- /bin/bash**: This command starts an interactive shell inside a specific container within a pod. For example:

```
kubectl exec -it my-pod --container=my-container -- /bin/bash
```

This command opens an interactive shell (`/bin/bash`) inside the container named `my-container` within the pod named `my-pod`.

91. **kubectl create role**: This command creates a role within a namespace. For example:

```
kubectl create role my-role --verb=get --resource=pods
```

This command creates a role named `my-role` with permissions to get pods within the namespace.

92. **kubectl apply -f deployment.yaml --namespace=my-namespace --record**: It applies changes to a deployment within a specific namespace and records the changes. For example:

```
kubectl apply -f deployment.yaml --namespace=my-namespace --record
```

This command applies the changes specified in `deployment.yaml` to the deployment in the namespace `my-namespace` and records the changes.

93. **kubectl describe persistentvolume my-pv**: This command provides detailed information about a persistent volume. For example:

```
kubectl describe persistentvolume my-pv
```

This command describes the persistent volume named `my-pv`, displaying detailed information including its capacity and access modes.

94. **kubectl create serviceaccount my-service-account**: It creates a service account within a namespace. For instance:

```
kubectl create serviceaccount my-service-account
```

This command creates a service account named `my-service-account` within the current namespace.

95. **kubectl get events --sort-by=.metadata.creationTimestamp**: This command retrieves events sorted by creation timestamp. For example:

```
kubectl get events --sort-by=.metadata.creationTimestamp
```

This command retrieves all events in the current namespace sorted by their creation timestamp in ascending order.

96. **kubectl describe ingresses.extensions**: It provides detailed information about an Ingress resource. For instance:

```
kubectl describe ingresses.extensions
```

This command describes all Ingress resources in the current namespace, displaying detailed information about each Ingress.

97. **kubectl rollout undo deployment/my-deployment --dry-run=client**: It simulates rolling back a deployment to the previous revision without actually performing the rollback. For example:

```
kubectl rollout undo deployment/my-deployment --dry-run=client
```

This command simulates rolling back the deployment named `my-deployment` to the previous revision without actually performing the rollback.

98. **kubectl scale deployment/my-deployment --replicas=5 --record**: This command scales the number of replicas of a deployment to 5 and records the change. For example:

```
kubectl scale deployment/my-deployment --replicas=5 --record
```

This command scales the deployment named `my-deployment` to have 5 replicas and records the change.

99. **kubectl delete secret my-secret**: It deletes a secret. For instance:

```
kubectl delete secret my-secret
```

This command deletes the secret named `my-secret`.

100. **kubectl get ingress**: This command retrieves information about Ingress resources in the cluster. For example:

```
kubectl get ingress
```

This command retrieves information about all Ingress resources in the current namespace.