**🟢 Section 1: Deployments, Pods, and Services**

1. **Check Cluster Nodes**

bash

CopyEdit

kubectl get nodes

*Purpose:* Lists all nodes in your cluster.

*Expected Output:* Details of the node(s) with statuses like Ready.

1. **Create an NGINX Deployment**

bash

CopyEdit

kubectl create deployment nginx-deployment --image=nginx

*Purpose:* Deploys an NGINX application.[Kubernetes](https://kubernetes.io/docs/tutorials/hello-minikube/?utm_source=chatgpt.com)

*Expected Output:* Confirmation of deployment creation.[Kubernetes](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com" \t "_blank)

1. **List Deployments**

bash

CopyEdit

kubectl get deployments

*Purpose:* Displays all deployments in the current namespace.

*Expected Output:* Details of deployments including nginx-deployment.

1. **List Pods**

bash

CopyEdit

kubectl get pods

*Purpose:* Shows all pods running in the current namespace.[bluematador.com](https://www.bluematador.com/learn/kubernetes-glossary?utm_source=chatgpt.com)

*Expected Output:* Pod names with their statuses.[YouTube](https://m.youtube.com/watch?pp=ygUOI2tpcmliYWt1YmVkaXQ%3D&v=feLpGydQVio&utm_source=chatgpt.com" \t "_blank)

1. **Describe a Pod**

bash

CopyEdit

kubectl describe pod <pod-name>

*Purpose:* Provides detailed information about a specific pod.

*Expected Output:* Comprehensive details including events and configurations.

1. **Expose Deployment as a Service**

bash

CopyEdit

kubectl expose deployment nginx-deployment --port=80 --type=NodePort

*Purpose:* Creates a service to expose the NGINX deployment.

*Expected Output:* Confirmation of service creation.

1. **List Services**

bash

CopyEdit

kubectl get services

*Purpose:* Displays all services in the current namespace.

*Expected Output:* Details of services including nginx-deployment.

1. **Access Service via Minikube**

bash

CopyEdit

minikube service nginx-deployment

*Purpose:* Opens the NGINX service in your default web browser.

*Expected Output:* Browser opens displaying the NGINX welcome page.[Kubernetes+1bluematador.com+1](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com" \t "_blank)

1. **Scale Deployment**

bash

CopyEdit

kubectl scale deployment nginx-deployment --replicas=3

*Purpose:* Scales the deployment to 3 replicas.[CloudSigma+2bluematador.com+2Apptio+2](https://www.bluematador.com/learn/kubernetes-glossary?utm_source=chatgpt.com)

*Expected Output:* Confirmation of scaling action.[Apptio+1Kyverno+1](https://blog.kubecost.com/blog/kubernetes-taints/?utm_source=chatgpt.com" \t "_blank)

1. **Delete Deployment**

bash

CopyEdit

kubectl delete deployment nginx-deployment

*Purpose:* Removes the NGINX deployment.

*Expected Output:* Confirmation of deletion.

**🟡 Section 2: Service Discovery and Networking**

1. **Create a Service**

bash

CopyEdit

kubectl create service clusterip nginx-service --tcp=80:80

*Purpose:* Creates a ClusterIP service for NGINX.

*Expected Output:* Confirmation of service creation.

1. **Get Service Details**

bash

CopyEdit

kubectl get service nginx-service

*Purpose:* Displays details of the nginx-service.

*Expected Output:* Service details including cluster IP and ports.

1. **Describe Service**

bash

CopyEdit

kubectl describe service nginx-service

*Purpose:* Provides detailed information about the service.

*Expected Output:* Comprehensive service details including endpoints.

1. **List Endpoints**

bash

CopyEdit

kubectl get endpoints

*Purpose:* Shows the endpoints associated with services.

*Expected Output:* Endpoints linked to nginx-service.

1. **Port Forwarding**

bash

CopyEdit

kubectl port-forward service/nginx-service 8080:80

*Purpose:* Forwards port 80 of the service to localhost:8080.

*Expected Output:* Forwarding confirmation; access via http://localhost:8080.

**🔵 Section 3: Persistent Volumes and Storage**

1. **Create PersistentVolume**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

*Purpose:* Defines a persistent volume for MySQL data.

*Expected Output:* PersistentVolume mysql-pv created.

1. **Create PersistentVolumeClaim**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

*Purpose:* Requests storage from available PVs.

*Expected Output:* PersistentVolumeClaim mysql-pvc created and bound.

1. **Deploy MySQL with PVC**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: mysql

spec:

containers:

- name: mysql

image: mysql:5.6

env:

- name: MYSQL\_ROOT\_PASSWORD

value: password

ports:

- containerPort: 3306

volumeMounts:

- name: mysql-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: mysql-pvc

*Purpose:* Runs a MySQL pod using the defined PVC.

*Expected Output:* MySQL pod is running and using persistent storage.

1. **Check PVC Status**

bash

CopyEdit

kubectl get pvc

*Purpose:* Displays status of persistent volume claims.

*Expected Output:* mysql-pvc status as Bound.

1. **Delete PVC**

bash

CopyEdit

kubectl delete pvc mysql-pvc

*Purpose:* Removes the persistent volume claim.

*Expected Output:* Confirmation of deletion.

**🟣 Section 4: Namespaces and Resource Management**

1. **Create Namespace**

bash

CopyEdit

kubectl create namespace dev

*Purpose:* Creates a new namespace named dev.

*Expected Output:* Namespace dev created.

1. **List Namespaces**

bash

CopyEdit

kubectl get namespaces

*Purpose:* Displays all namespaces in the cluster.

*Expected Output:* List including dev namespace.

1. **Deploy in Specific Namespace**

bash

CopyEdit

kubectl create deployment nginx-dev --image=nginx --namespace=dev

*Purpose:* Deploys NGINX in the dev namespace.

*Expected Output:* Deployment nginx-dev created in dev.

1. **Set Default Namespace**

bash

CopyEdit

kubectl config set-context --current --namespace=dev

*Purpose:* Sets dev as the default namespace for kubectl commands.

*Expected Output:* Context updated to use dev namespace.

1. **Delete Namespace**

bash

CopyEdit

kubectl delete namespace dev

*Purpose:* Removes the dev namespace and its resources.

*Expected Output:* Namespace dev deleted.

**🟤 Section 5: Taints, Tolerations, and Node Labels**

1. **List Nodes**

bash

CopyEdit

kubectl get nodes

*Purpose:* Displays all nodes in the cluster.

*Expected Output:* List of nodes with their statuses.

1. **Add Taint to Node**

bash

CopyEdit

kubectl taint nodes <node-name> key=value:NoSchedule

*Purpose:* Prevents pods without matching tolerations from scheduling on the node.

*Expected Output:* Taint added to the specified node.

1. **Remove Taint from Node**

bash

CopyEdit

kubectl taint nodes <node-name> key=value:NoSchedule-

*Purpose:* Removes the specified taint from the node.

*Expected Output:* Taint removed from the node.

1. **Add Label to Node**

bash

CopyEdit

kubectl label nodes <node-name> disktype=ssd

*Purpose:* Assigns a label disktype=ssd to the specified node, which can be used for scheduling pods to specific nodes.[Kubernetes+1OpenShift Documentation+1](https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/?utm_source=chatgpt.com)

*Expected Output:* Confirmation that the label has been added to the node.

1. **View Node Labels**

bash

CopyEdit

kubectl get nodes --show-labels

*Purpose:* Displays all nodes along with their labels, helping to verify label assignments.

*Expected Output:* List of nodes with their respective labels.

1. **Schedule Pod to Specific Node Using Node Selector**

Create a pod definition file nginx-pod.yaml:

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx

nodeSelector:

disktype: ssd

Apply the configuration:

bash

CopyEdit

kubectl apply -f nginx-pod.yaml

*Purpose:* Schedules the pod to a node labeled with disktype=ssd.

*Expected Output:* Pod nginx-pod is created and scheduled on the appropriate node.

**🟠 Section 6: Resource Quotas and Limits**

1. **Create Resource Quota**

Create a file quota.yaml:

yaml

CopyEdit

apiVersion: v1

kind: ResourceQuota

metadata:

name: mem-cpu-quota

spec:

hard:

pods: "10"

requests.cpu: "4"

requests.memory: 4Gi

limits.cpu: "10"

limits.memory: 10Gi

Apply the quota:

bash

CopyEdit

kubectl apply -f quota.yaml --namespace=dev

*Purpose:* Limits the number of pods and the total CPU and memory usage in the dev namespace.[Medium+1Kubernetes+1](https://medium.com/%40muppedaanvesh/a-hand-on-guide-to-kubernetes-resource-quotas-limit-ranges-%EF%B8%8F-8b9f8cc770c5?utm_source=chatgpt.com)

*Expected Output:* ResourceQuota mem-cpu-quota created in the dev namespace.

1. **View Resource Quotas**

bash

CopyEdit

kubectl get resourcequota --namespace=dev

*Purpose:* Displays the resource quotas set in the dev namespace.

*Expected Output:* List of resource quotas with their usage and limits.

1. **Create Limit Range**

Create a file limitrange.yaml:

yaml

CopyEdit

apiVersion: v1

kind: LimitRange

metadata:

name: cpu-mem-limit-range

spec:

limits:

- default:

cpu: 500m

memory: 512Mi

defaultRequest:

cpu: 200m

memory: 256Mi

type: Container

Apply the limit range:

bash

CopyEdit

kubectl apply -f limitrange.yaml --namespace=dev

*Purpose:* Sets default CPU and memory requests and limits for containers in the dev namespace.

*Expected Output:* LimitRange cpu-mem-limit-range created in the dev namespace.

1. **View Limit Ranges**

bash

CopyEdit

kubectl get limitrange --namespace=dev

*Purpose:* Displays the limit ranges set in the dev namespace.[OpenShift Documentation](https://docs.openshift.com/container-platform/3.11/dev_guide/compute_resources.html?utm_source=chatgpt.com)

*Expected Output:* List of limit ranges with their default requests and limits.

**🔴 Section 7: Helm - Kubernetes Package Manager**

1. **Install Helm**

Follow the official Helm installation guide for your operating system:

[Kubernetes](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com)

*Purpose:* Installs Helm, a package manager for Kubernetes applications.[Intellipaat+2OpenShift Documentation+2Medium+2](https://docs.openshift.com/container-platform/3.11/dev_guide/compute_resources.html?utm_source=chatgpt.com)

*Expected Output:* Helm is installed and ready to use.

1. **Add Helm Repository**

bash

CopyEdit

helm repo add bitnami https://charts.bitnami.com/bitnami

*Purpose:* Adds the Bitnami repository to Helm, providing access to a variety of charts.

*Expected Output:* Repository "bitnami" has been added to your repositories.

1. **Update Helm Repositories**

bash

CopyEdit

helm repo update

*Purpose:* Updates the local Helm chart repository cache.

*Expected Output:* Helm repositories are updated with the latest charts.

1. **Install NGINX Using Helm**

bash

CopyEdit

helm install my-nginx bitnami/nginx

*Purpose:* Deploys an NGINX application using the Bitnami Helm chart.

*Expected Output:* Release "my-nginx" has been deployed.[OpenShift Documentation+1Kubernetes+1](https://docs.openshift.com/container-platform/3.11/dev_guide/compute_resources.html?utm_source=chatgpt.com" \t "_blank)

1. **List Helm Releases**

bash

CopyEdit

helm list

*Purpose:* Lists all Helm releases in the current namespace.

*Expected Output:* Table of releases with their statuses.[OpenShift Documentation](https://docs.openshift.com/container-platform/4.8/nodes/scheduling/nodes-scheduler-taints-tolerations.html?utm_source=chatgpt.com" \t "_blank)

1. **Uninstall Helm Release**

bash

CopyEdit

helm uninstall my-nginx

*Purpose:* Removes the my-nginx release from the cluster.

*Expected Output:* Release "my-nginx" uninstalled.

**🟢 Section 8: Deploying WordPress and MySQL with Persistent Volumes**

1. **Create Namespace for WordPress**

bash

CopyEdit

kubectl create namespace wordpress

*Purpose:* Creates a separate namespace for the WordPress application.

*Expected Output:* Namespace wordpress created.[GopenSource+1Kubernetes+1](https://gopensource.com/deploy-wordpress-blog-with-mysql-on-a-bare-metal-kubernetes-cluster-8a9323c0f4c9?utm_source=chatgpt.com" \t "_blank)

1. **Deploy MySQL Using Helm**

bash

CopyEdit

helm install my-mysql bitnami/mysql --namespace wordpress

*Purpose:* Deploys a MySQL database using the Bitnami Helm chart in the wordpress namespace.

*Expected Output:* Release "my-mysql" has been deployed.

1. **Deploy WordPress Using Helm**

bash

CopyEdit

helm install my-wordpress bitnami/wordpress --namespace wordpress

*Purpose:* Deploys a WordPress application using the Bitnami Helm chart in the wordpress namespace.

*Expected Output:* Release "my-wordpress" has been deployed.[Kubernetes+1GopenSource+1](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com" \t "_blank)

1. **Retrieve WordPress URL**

bash

CopyEdit

kubectl get svc --namespace wordpress my-wordpress

*Purpose:* Retrieves the service details to access the WordPress application.

*Expected Output:* Service information including external IP or NodePort.

1. **Access WordPress Application**

Use the external IP or NodePort obtained in the previous step to access the WordPress application via a web browser.

*Purpose:* Verifies that the WordPress application is accessible.

*Expected Output:* WordPress setup page or homepage.

1. **Check Persistent Volume Claims**

bash

CopyEdit

kubectl get pvc --namespace wordpress

*Purpose:* Lists all PVCs in the wordpress namespace to verify storage provisioning.

*Expected Output:* List of PVCs with their statuses.

1. **Delete WordPress Release**

bash

CopyEdit

helm uninstall my-wordpress --namespace wordpress

*Purpose:* Removes the WordPress application from the cluster.

*Expected Output:* Release "my-wordpress" uninstalled.

1. **Delete MySQL Release**

bash

CopyEdit

helm uninstall my-mysql --namespace wordpress

*Purpose:* Removes the MySQL database from the cluster.

*Expected Output:* Release "my-mysql" uninstalled.

1. **Delete WordPress Namespace**

bash

CopyEdit

kubectl delete namespace wordpress

*Purpose:* Deletes the wordpress namespace and all associated resources.

*Expected Output:* Namespace wordpress deleted.

These commands provide a comprehensive walkthrough of Kubernetes operations, from basic deployments to managing complex applications with Helm. By practicing these commands, you'll gain a solid foundation in Kubernetes administration.

Kubernetes setup to deploy WordPress and MySQL with service discovery, utilizing ConfigMaps and Secrets for configuration management.

**🗂️ 1. Namespace**

Creating a dedicated namespace to isolate resources:[DevOpsCube](https://devopscube.com/deploy-wordpress-on-kubernetes/?utm_source=chatgpt.com" \t "_blank)

yaml

CopyEdit

apiVersion: v1

kind: Namespace

metadata:

name: wordpress

Apply with:

bash

CopyEdit

kubectl apply -f namespace.yaml

**🔐 2. Secrets**

Storing sensitive data like database credentials securely:

yaml

CopyEdit

apiVersion: v1

kind: Secret

metadata:

name: mysql-secret

namespace: wordpress

type: Opaque

data:

mysql-root-password: cGFzc3dvcmQ= # 'password' base64 encoded

mysql-user: dXNlcg== # 'user'

mysql-password: cGFzc3dvcmQ= # 'password'

mysql-database: d29yZHByZXNz # 'wordpress'

Apply with:

bash

CopyEdit

kubectl apply -f mysql-secret.yaml

**⚙️ 3. ConfigMaps**

Defining non-sensitive configuration data:[Clear Linux Project](https://www.clearlinux.org/node/31541.html?utm_source=chatgpt.com" \t "_blank)

yaml

CopyEdit

apiVersion: v1

kind: ConfigMap

metadata:

name: wordpress-config

namespace: wordpress

data:

WORDPRESS\_DB\_HOST: mysql

WORDPRESS\_DB\_NAME: wordpress

WORDPRESS\_DB\_USER: user

Apply with:

bash

CopyEdit

kubectl apply -f wordpress-config.yaml

**💾 4. Persistent Volume Claims**

Requesting persistent storage for MySQL and WordPress:

**MySQL PVC:**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

namespace: wordpress

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 20Gi

**WordPress PVC:**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: wordpress-pvc

namespace: wordpress

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 20Gi

Apply with:[Medium+5Kubernetes+5Clear Linux Project+5](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com" \t "_blank)

bash

CopyEdit

kubectl apply -f mysql-pvc.yaml

kubectl apply -f wordpress-pvc.yaml

**🐬 5. MySQL Deployment and Service**

**Deployment:**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

namespace: wordpress

spec:

selector:

matchLabels:

app: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: mysql

spec:

containers:

- image: mysql:5.6

name: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secret

key: mysql-root-password

- name: MYSQL\_DATABASE

valueFrom:

secretKeyRef:

name: mysql-secret

key: mysql-database

- name: MYSQL\_USER

valueFrom:

secretKeyRef:

name: mysql-secret

key: mysql-user

- name: MYSQL\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secret

key: mysql-password

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- name: mysql-persistent-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pvc

**Service:**

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: mysql

namespace: wordpress

spec:

ports:

- port: 3306

selector:

app: mysql

Apply with:[kubernetes-by-gaurav.hashnode.dev+13Kubernetes+13Discuss Kubernetes+13](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com" \t "_blank)

bash

CopyEdit

kubectl apply -f mysql-deployment.yaml

kubectl apply -f mysql-service.yaml

**📝 6. WordPress Deployment and Service**

**Deployment:**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress

namespace: wordpress

spec:

selector:

matchLabels:

app: wordpress

strategy:

type: Recreate

template:

metadata:

labels:

app: wordpress

spec:

containers:

- image: wordpress:4.8-apache

name: wordpress

env:

- name: WORDPRESS\_DB\_HOST

valueFrom:

configMapKeyRef:

name: wordpress-config

key: WORDPRESS\_DB\_HOST

- name: WORDPRESS\_DB\_NAME

valueFrom:

configMapKeyRef:

name: wordpress-config

key: WORDPRESS\_DB\_NAME

- name: WORDPRESS\_DB\_USER

valueFrom:

configMapKeyRef:

name: wordpress-config

key: WORDPRESS\_DB\_USER

- name: WORDPRESS\_DB\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secret

key: mysql-password

ports:

- containerPort: 80

name: wordpress

volumeMounts:

- name: wordpress-persistent-storage

mountPath: /var/www/html

volumes:

- name: wordpress-persistent-storage

persistentVolumeClaim:

claimName: wordpress-pvc

**Service:**

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: wordpress

namespace: wordpress

spec:

type: NodePort

ports:

- port: 80

targetPort: 80

nodePort: 30080

selector:

app: wordpress

Apply with:

bash

CopyEdit

kubectl apply -f wordpress-deployment.yaml

kubectl apply -f wordpress-service.yaml

**🔍 7. Service Discovery**

In Kubernetes, services are discoverable within the cluster by their DNS names. The WordPress pod uses the WORDPRESS\_DB\_HOST environment variable, set to mysql, to connect to the MySQL service. Kubernetes' internal DNS resolves this to the appropriate service IP, facilitating seamless communication between WordPress and MySQL.

**🚀 8. Accessing WordPress**

To access the WordPress site from your browser:

1. Retrieve the Node IP:

bash

CopyEdit

minikube ip

1. Access WordPress:[Clear Linux Project+8Kubernetes+8Medium+8](https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/?utm_source=chatgpt.com)

Open your browser and navigate to http://<NodeIP>:30080.

This setup ensures a secure and organized deployment of WordPress and MySQL on Kubernetes, leveraging best practices for configuration management and service discovery.

**1. Node Affinity**

**Objective:** Schedule a pod on nodes with specific labels using Node Affinity.[devtron.ai+6StackState+6Komodor+6](https://www.stackstate.com/blog/mastering-node-affinity-in-kubernetes/?utm_source=chatgpt.com)

**Step 1:** Label a node.[Medium+17Stack Overflow+17Learnk8s+17](https://stackoverflow.com/questions/70415874/mistakes-in-my-ingress-yaml-file-or-process?utm_source=chatgpt.com)

bash

CopyEdit

kubectl label nodes <node-name> disktype=ssd

**Step 2:** Create a deployment with Node Affinity.

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-affinity

spec:

replicas: 1

selector:

matchLabels:

app: nginx-affinity

template:

metadata:

labels:

app: nginx-affinity

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: disktype

operator: In

values:

- ssd

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**Explanation:** This deployment ensures that the pod is scheduled only on nodes labeled with disktype=ssd.[Spot.io+3Densify+3StackState+3](https://www.densify.com/kubernetes-autoscaling/kubernetes-affinity/?utm_source=chatgpt.com)

**2. Pod Affinity**

**Objective:** Schedule a pod on the same node as another pod with specific labels using Pod Affinity.[Kubernetes+5Medium+5Densify+5](https://medium.com/%40prasad.midde3/understanding-node-affinity-pod-affinity-node-selector-and-pod-anti-affinity-in-kubernetes-7899e218ac6d?utm_source=chatgpt.com)

**Step 1:** Deploy a pod with a specific label.

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: existing-pod

labels:

app: frontend

spec:

containers:

- name: nginx

image: nginx

**Step 2:** Create a deployment with Pod Affinity.

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-pod-affinity

spec:

replicas: 1

selector:

matchLabels:

app: nginx-pod-affinity

template:

metadata:

labels:

app: nginx-pod-affinity

spec:

affinity:

podAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- frontend

topologyKey: "kubernetes.io/hostname"

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**Explanation:** This deployment ensures that the pod is scheduled on the same node as the existing pod labeled app=frontend.

**3. Namespaces and Resource Quotas**

**Objective:** Demonstrate how resource quotas can restrict resource usage within a namespace.[Matthew Palmer+12Medium+12Kubernetes+12](https://medium.com/%40muppedaanvesh/a-hand-on-guide-to-kubernetes-resource-quotas-limit-ranges-%EF%B8%8F-8b9f8cc770c5?utm_source=chatgpt.com)

**Step 1:** Create a namespace.

yaml

CopyEdit

apiVersion: v1

kind: Namespace

metadata:

name: dev

Apply the namespace:

bash

CopyEdit

kubectl apply -f namespace.yaml

**Step 2:** Apply a resource quota to the namespace.[Kubernetes+1Kubernetes+1](https://kubernetes.io/docs/concepts/policy/resource-quotas/?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: v1

kind: ResourceQuota

metadata:

name: dev-quota

namespace: dev

spec:

hard:

pods: "2"

requests.cpu: "1"

requests.memory: 1Gi

limits.cpu: "2"

limits.memory: 2Gi

Apply the resource quota:[Groundcover+9Kubernetes+9Kubernetes+9](https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/quota-pod-namespace/?utm_source=chatgpt.com" \t "_blank)

bash

CopyEdit

kubectl apply -f resource-quota.yaml

**Step 3:** Attempt to deploy more pods than allowed.

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-overquota

namespace: dev

spec:

replicas: 3

selector:

matchLabels:

app: nginx-overquota

template:

metadata:

labels:

app: nginx-overquota

spec:

containers:

- name: nginx

image: nginx

resources:

requests:

cpu: "500m"

memory: "512Mi"

limits:

cpu: "1"

memory: "1Gi"

Apply the deployment:[Spacelift+9StackState+9Medium+9](https://www.stackstate.com/blog/mastering-node-affinity-in-kubernetes/?utm_source=chatgpt.com" \t "_blank)

bash

CopyEdit

kubectl apply -f deployment.yaml

**Expected Outcome:** The deployment will fail to create all pods due to the resource quota limits.[Kubernetes](https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/quota-pod-namespace/?utm_source=chatgpt.com" \t "_blank)

**4. Blue-Green Deployment**

**Objective:** Implement a blue-green deployment strategy to minimize downtime during updates.[Medium+1Spacelift+1](https://medium.com/cloud-native-daily/blue-green-deployments-with-kubernetes-a-comprehensive-guide-5d196dad1976?utm_source=chatgpt.com)

**Step 1:** Deploy the "blue" version.[Medium+33Reddit+33Semaphore+33](https://www.reddit.com/r/kubernetes/comments/177laoe/blue_green_deployment/?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-blue

labels:

app: nginx

version: blue

spec:

replicas: 2

selector:

matchLabels:

app: nginx

version: blue

template:

metadata:

labels:

app: nginx

version: blue

spec:

containers:

- name: nginx

image: nginx:1.14

ports:

- containerPort: 80

**Step 2:** Create a service pointing to the "blue" deployment.[StackState](https://www.stackstate.com/blog/mastering-node-affinity-in-kubernetes/?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

version: blue

ports:

- protocol: TCP

port: 80

targetPort: 80

**Step 3:** Deploy the "green" version.[Medium+1Semaphore+1](https://medium.com/cloud-native-daily/blue-green-deployments-with-kubernetes-a-comprehensive-guide-5d196dad1976?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-green

labels:

app: nginx

version: green

spec:

replicas: 2

selector:

matchLabels:

app: nginx

version: green

template:

metadata:

labels:

app: nginx

version: green

spec:

containers:

- name: nginx

image: nginx:1.16

ports:

- containerPort: 80

**Step 4:** Switch the service to point to the "green" deployment.

bash

CopyEdit

kubectl patch service nginx-service -p '{"spec":{"selector":{"app":"nginx","version":"green"}}}'

**Explanation:** This approach allows you to switch traffic between versions with minimal downtime.[Medium](https://medium.com/cloud-native-daily/blue-green-deployments-with-kubernetes-a-comprehensive-guide-5d196dad1976?utm_source=chatgpt.com)

**5. Ingress Controller**

**Objective:** Expose services externally using an Ingress Controller.[Kubernetes+3Matthew Palmer+3Gist+3](https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-ingress-guide-nginx-example.html?utm_source=chatgpt.com)

**Step 1:** Install NGINX Ingress Controller (for Minikube).[Spacelift](https://spacelift.io/blog/kubernetes-ingress?utm_source=chatgpt.com" \t "_blank)

bash

CopyEdit

minikube addons enable ingress

**Step 2:** Create a deployment and service.[StackState+5Kubernetes+5Semaphore+5](https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-app

spec:

replicas: 2

selector:

matchLabels:

app: nginx-app

template:

metadata:

labels:

app: nginx-app

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx-app

ports:

- protocol: TCP

port: 80

targetPort: 80

**Step 3:** Create an Ingress resource.[Kubernetes+3Medium+3kubernetes.github.io+3](https://nidhiashtikar.medium.com/kubernetes-ingress-d71127920357?utm_source=chatgpt.com)

yaml

CopyEdit

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: nginx.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

**Step 4:** Update your /etc/hosts file (for local testing).

bash

CopyEdit

echo "$(minikube ip) nginx.example.com" | sudo tee -a /etc/hosts

**Explanation:** This setup routes external traffic to your service using the NGINX Ingress Controller.

**6. Role-Based Access Control (RBAC)**

**Objective:** Restrict user access to specific resources using RBAC.[Spacelift+1Kubernetes+1](https://spacelift.io/blog/kubernetes-rbac?utm_source=chatgpt.com)

**Step 1:** Create a namespace.

yaml

CopyEdit

apiVersion: v1

kind: Namespace

metadata:

name: dev

**Step 2:** Create a service account.

yaml

CopyEdit

apiVersion: v1

kind: ServiceAccount

metadata:

name: dev-user

namespace: dev

**Step 3:** Create a role with specific permissions.

A Role defines a set of permissions within a specific namespace. In this example, we'll create a Role that allows read-only access to Pods in the dev namespace.[Loft+10DEV Community+10Apptio+10](https://dev.to/haamid/kubernetes-service-account-and-rbac-tutorial-2a6d?utm_source=chatgpt.com)

**role.yaml**

yaml

CopyEdit

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: dev

name: pod-reader

rules:

- apiGroups: [""]

resources: ["pods"]

verbs: ["get", "list", "watch"]

Apply the Role:

bash

CopyEdit

kubectl apply -f role.yaml

**Step 4: Create a RoleBinding**

A RoleBinding grants the permissions defined in a Role to a user or ServiceAccount within a namespace. We'll bind the pod-reader Role to the dev-user ServiceAccount.[Loft+12Apptio+12Spacelift+12](https://www.kubecost.com/kubernetes-best-practices/kubernetes-rbac-best-practices/?utm_source=chatgpt.com)

**rolebinding.yaml**

yaml

CopyEdit

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: read-pods

namespace: dev

subjects:

- kind: ServiceAccount

name: dev-user

namespace: dev

roleRef:

kind: Role

name: pod-reader

apiGroup: rbac.authorization.k8s.io

Apply the RoleBinding:

bash

CopyEdit

kubectl apply -f rolebinding.yaml

**Step 5: Test Permissions**

To verify that the dev-user ServiceAccount has the intended permissions, we'll create a Pod that uses this ServiceAccount and attempts to list Pods in the dev namespace.

**test-pod.yaml**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: rbac-test

namespace: dev

spec:

serviceAccountName: dev-user

containers:

- name: kubectl

image: bitnami/kubectl:latest

command: ["sh", "-c", "kubectl get pods"]

Apply the test Pod:

bash

CopyEdit

kubectl apply -f test-pod.yaml

Check the logs to see the output of the kubectl get pods command:

bash

CopyEdit

kubectl logs rbac-test -n dev

**Expected Outcome:**

The output should list the Pods in the dev namespace, confirming that the dev-user ServiceAccount has read access to Pods.

**Note:** If you attempt to perform an action not permitted by the Role (e.g., deleting a Pod), Kubernetes will return a "Forbidden" error, indicating that the ServiceAccount lacks the necessary permissions.

This exercise demonstrates how to use RBAC in Kubernetes to grant specific permissions to a ServiceAccount within a namespace, enhancing the security and access control of your cluster.

**7. Pod Anti-Affinity**

**Objective:** Ensure that pods of the same application are not scheduled on the same node to improve fault tolerance.[eksworkshop.com+2Apptio+2Medium+2](https://blog.kubecost.com/blog/kubernetes-node-affinity/?utm_source=chatgpt.com)

**Scenario:** You have a deployment of NGINX pods, and you want to prevent multiple pods from running on the same node.

**Step 1:** Apply the following deployment configuration:

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-anti-affinity

spec:

replicas: 3

selector:

matchLabels:

app: nginx-anti-affinity

template:

metadata:

labels:

app: nginx-anti-affinity

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchLabels:

app: nginx-anti-affinity

topologyKey: "kubernetes.io/hostname"

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**Explanation:** The podAntiAffinity rule ensures that no two pods with the label app: nginx-anti-affinity are scheduled on the same node. The topologyKey set to kubernetes.io/hostname indicates that the rule applies at the node level.

**Expected Outcome:** Each NGINX pod will be scheduled on a different node, provided there are enough nodes available.

**8. Node Anti-Affinity**

**Objective:** Prevent pods from being scheduled on nodes with specific labels, which might be reserved for other workloads.

**Scenario:** You have nodes labeled with disktype=spinning, and you want to avoid scheduling certain pods on these nodes.

**Step 1:** Label the nodes you want to avoid:

bash

CopyEdit

kubectl label nodes <node-name> disktype=spinning

**Step 2:** Apply the following deployment configuration:

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-node-anti-affinity

spec:

replicas: 3

selector:

matchLabels:

app: nginx-node-anti-affinity

template:

metadata:

labels:

app: nginx-node-anti-affinity

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: disktype

operator: NotIn

values:

- spinning

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**Explanation:** The nodeAffinity rule with operator: NotIn ensures that pods are not scheduled on nodes labeled with disktype=spinning.[Medium](https://medium.com/%40prasad.midde3/understanding-node-affinity-pod-affinity-node-selector-and-pod-anti-affinity-in-kubernetes-7899e218ac6d?utm_source=chatgpt.com)

**Expected Outcome:** Pods will be scheduled only on nodes that do not have the label disktype=spinning.

By implementing these affinity and anti-affinity rules, you can fine-tune pod scheduling to meet your application's performance and availability requirements.