Kubernetes Pod and Service Troubleshooting Runbook

This runbook will help you troubleshoot common issues with Kubernetes pods and services using kubectl commands. We'll cover steps to identify, diagnose, and resolve problems related to pods and services in your Kubernetes cluster.

Step 1: Verify Kubernetes Cluster and kubectl Setup Ensure that you have access to the Kubernetes cluster and that kubectl is properly configured to interact with the cluster.

bash

Copy code

```
kubectl version
kubectl get nodes
kubectl get pods --all-namespaces
```

Step 2: Check Pod Status List all pods in the namespace to check their status and identify any pods with issues.

bash

Copy code

```
kubectl get pods -n YOUR_NAMESPACE
```

Step 3: Describe Pod Details Describe the problematic pod to get more details about its status and events.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE
```

Step 4: Check Pod Logs Inspect the logs of the pod to identify any errors or issues.

bash

Copy code

```
kubectl logs YOUR_POD_NAME -n YOUR_NAMESPACE
```

Step 5: Check Resource Requests and Limits Ensure that the pod's resource requests and limits are appropriately configured.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i "resources\|limits"
```

Step 6: Verify Node Affinity and Taints/Tolerations Check if the pod has specific node affinity requirements or if there are any node taints causing scheduling issues.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i
"affinity\|toleration\|taint"
```

Step 7: Check Service Status List all services in the namespace to check their status and identify any services with issues.

bash

Copy code

```
kubectl get services -n YOUR_NAMESPACE
```

Step 8: Describe Service Details Describe the problematic service to get more details about its status and configuration.

bash

Copy code

```
kubectl describe service YOUR_SERVICE_NAME -n YOUR_NAMESPACE
```

Step 9: Verify Service Endpoints Ensure that the service has valid endpoints to reach the corresponding pods.

bash

Copy code

```
kubectl describe endpoints YOUR_SERVICE_NAME -n YOUR_NAMESPACE
```

Step 10: Test Service Connectivity Check if you can connect to the service from within the cluster using a temporary pod.

bash

Copy code

```
kubectl run test-pod -n YOUR_NAMESPACE --image=busybox --restart=Never -- /bin/sh
-c "wget -qO- YOUR_SERVICE_NAME.YOUR_NAMESPACE"
```

Step 11: Check Network Policies (If Applicable) If Network Policies are applied in the cluster, verify that they are not blocking pod-to-pod communication.

bash

Copy code

```
kubectl describe networkpolicies -n YOUR_NAMESPACE
```

Step 12: Check for Deployment/Rollout Issues If the pod is managed by a Deployment, review the rollout status and history for any issues.

bash

Copy code

```
kubectl rollout status deployment YOUR_DEPLOYMENT_NAME -n YOUR_NAMESPACE
kubectl rollout history deployment YOUR_DEPLOYMENT_NAME -n YOUR_NAMESPACE
```

Step 13: Verify DNS Resolution Check if DNS resolution is working correctly within the cluster.

bash

Copy code

```
kubectl run test-pod -n YOUR_NAMESPACE --image=busybox --restart=Never -- nslookup
YOUR_SERVICE_NAME.YOUR_NAMESPACE
```

Step 14: Check Persistent Volume Claims (If Applicable) If the pod uses Persistent Volume Claims (PVCs), verify the status and availability of the associated volumes.

bash

Copy code

```
kubectl get pvc -n YOUR_NAMESPACE
kubectl describe pv -n YOUR_NAMESPACE
```

Step 15: Check for Resource Quotas (If Applicable) Verify that the pod's resource requirements comply with any defined resource quotas in the namespace.

bash

Copy code

```
kubectl describe resourcequotas -n YOUR_NAMESPACE
```

Step 16: Review Service Accounts and RBAC Ensure that the pod has the necessary service account and RBAC permissions to function correctly.

bash

Copy code

```
kubectl describe serviceaccount YOUR_SERVICE_ACCOUNT -n YOUR_NAMESPACE
kubectl get role,rolebinding,clusterrole,clusterrolebinding -n YOUR_NAMESPACE
```

Step 17: Check for ConfigMap and Secrets (If Applicable) Verify that the pod can access the required ConfigMaps and Secrets.

bash

Copy code

```
kubectl describe configmap YOUR_CONFIG_MAP_NAME -n YOUR_NAMESPACE
kubectl describe secret YOUR_SECRET_NAME -n YOUR_NAMESPACE
```

Step 18: Review Pod Annotations Inspect any annotations applied to the pod that might affect its behavior.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i "annotations"
```

Step 19: Check for Image Pull Issues If the pod is unable to pull the container image, inspect the image pull status and authentication settings.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i
"image\|pull\|registry"
```

Step 20: Review Pod Security Context (If Applicable) If the pod has a security context specified, verify its configuration.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i "security\|seccomp"
```

Step 21: Review Pod Network Settings (If Applicable) If the pod requires specific network settings, check its network configuration.

bash

Copy code

```
kubectl describe pod YOUR_POD_NAME -n YOUR_NAMESPACE | grep -i "network\|host"
```

Step 22: Check for AWS Support (If Required) If the issue persists or is beyond your expertise, consider reaching out to AWS Support for further assistance.