100 KUBERNETES ERROS & SOLUTIONS

By

DEVOPS SHACK

[Click here for DevSecOps & Cloud DevOps Course]

DevOps Shack

100 Common Kubernetes Errors with Step-by-Step Resolutions

Table of Contents

19. Pod Status Unknown
20. DaemonSet Not Deploying Pods on All Nodes

21-30

21. The Connection to the Server Was Refused
22. Pods Stuck in ContainerCreating State
23. RBAC Permission Denied Error
24. Pod IP Not Reachable
25. Kubelet Service Not Running
26. Deployment Not Updating
27. FailedAttachVolume
28. Namespace Deletion Stuck
29. Pods Stuck in Init State
30. Ingress Shows 404

31-40

31. Invalid Memory/CPU Requests
32. Service NodePort Not Accessible
33. Pods Not Being Scheduled
34. Pod Cannot Connect to External Services
35. Deployment Not Scaling Properly
36. Cluster Autoscaler Not Adding Nodes
37. API Server Timeout
38. Pod Cannot Access ConfigMap
39. DaemonSet Pods Not Deploying
40. Pods Cannot Pull Secrets

41-50

41. Service External-IP Pending

42. Pod Terminated With Exit Code 137
43. Failed to Start Pod Sandbox
44. Helm Release Stuck in PENDING_INSTALL
45. Ingress Returning 502 Bad Gateway
46. Error: context deadline exceeded
47. Volume Mount Permissions Denied
48. Pods Exceeding Resource Quota
49. Cluster Certificate Expired
50. HPA Not Responding to Metrics

51-60

51. Pod IP Conflict
52. Service Account Not Found
53. Pod Fails Liveness Probe
54. Cannot Delete Namespace
55. Cannot Access API from External Client
56. Ingress Controller Pods CrashLooping
57. Kubelet Certificate Rotation Failing
58. Metrics Server Shows No Data
59. DNS Resolution Intermittent
60. Pod Cannot Access PersistentVolume

61-70

61. Error: Forbidden when Running kubectl Commands
62. Node Cannot Join Cluster
63. API Server High Latency
64. DaemonSet Pods Not Running on Specific Node
65. Service Not Forwarding Traffic

66. Error: Unauthorized when Accessing API Server
67. Pod Logs Not Available
68. Cannot Delete Stuck Pod
69. Pods Restarting Frequently
70. Cluster Nodes Unreachable

71-80

71. Pods Stuck in ImagePullBackOff
72. Node Disk Pressure Causes Pod Evictions
73. Pod Cannot Access Secret
74. PVC Pending Due to StorageClass Issues
75. Cannot Scale Deployment Beyond Node Capacity
76. CoreDNS Pods CrashLooping
77. HPA Not Responding to Custom Metrics
78. Pod Scheduling Ignored Node Selector
79. Ingress SSL/TLS Configuration Fails
80. Kube-proxy Failing

81-90

81. Cluster Autoscaler Scaling Too Slowly
82. Pod Logs Truncated
83. Pods Stuck in Evicted State
84. Pod Cannot Access Cluster Internal DNS
85. PersistentVolume Stuck in Released State
86. Job Failing to Complete
87. ConfigMap Too Large
88. Services Intermittently Unreachable
89. Error: Connection Refused When Accessing Service

90. Cluster High CPU Usage

91-100

91. Pod Stuck in Pending Due to Node Affinity
92. Control Plane Components Not Starting
93. Pods Stuck in Terminating State
94. Cluster Upgrade Fails
95. Network Policy Blocking Traffic
96. Pods Stuck in Unknown State
97. PersistentVolume Not Resizing
98. Ingress Redirect Loop
99. Pod Security Context Misconfiguration
100. Pods Overloaded Due to Missing HPA

Introduction

Kubernetes has become the de facto standard for container orchestration, enabling organizations to deploy, manage, and scale applications with ease. Despite its immense power and flexibility, managing Kubernetes clusters can be challenging, especially for newcomers and even experienced practitioners. Kubernetes environments often present complex scenarios, with errors stemming from configuration mistakes, resource limitations, or underlying infrastructure issues.

This guide is designed to serve as a comprehensive resource for identifying and resolving the most common Kubernetes errors. From beginner-level issues such as CrashLoopBackOff and ImagePullBackOff to more advanced challenges like Ingress Redirect Loops or PersistentVolume Resizing, this guide covers a wide spectrum of scenarios. Each problem is broken down into its cause, along with step-by-step solutions, ensuring clarity and actionable insights.

Whether you are troubleshooting problems in development or production, the solutions provided here aim to reduce downtime, improve system reliability, and enhance your overall Kubernetes expertise. By systematically addressing these errors, Kubernetes practitioners can build robust, scalable, and resilient infrastructure for their applications.

This guide is an essential companion for developers, DevOps engineers, and administrators working with Kubernetes, providing not just solutions but also best practices to prevent these issues from recurring. Let’s dive into the world of Kubernetes problem-solving and unlock the full potential of this powerful orchestration platform.

Before diving into specific Kubernetes errors, start by checking these essential aspects to quickly identify potential root causes:

10 Most Important Things to Check for Diagnosing Kubernetes Errors

Before diving into specific Kubernetes errors, start by checking these essential aspects to quickly identify potential root causes:

Category	What to Check	Command/Action
1. Pod Status	Verify if the pod is running, pending, or in a failed state.	kubectl get pods
2. Pod Events	Check events for detailed information about the error.	kubectl describe pod <pod-name>
3. Pod Logs	Review the application logs to identify runtime issues.	kubectl logs <pod-name>
4. Node Status	Verify the status of nodes and ensure they are healthy.	kubectl get nodes
5. Node Resources	Ensure nodes have sufficient CPU, memory, and disk resources for scheduling pods.	kubectl describe node <node-name>
6. Service Endpoints	Verify if endpoints for the service are available and correctly mapped.	kubectl get endpoints <service-name>
7. Image Availability	Ensure the container image exists in the registry and is accessible.	kubectl describe pod <pod-name> (Check for ImagePullBackOff)
8. Resource Quotas	Check if the namespace has sufficient quotas to allow the deployment of new pods.	kubectl describe namespace <namespace-name>

Category	What to Check	Command/Action
9. PersistentVolumes	Verify if PVCs are bound and available for the pods.	kubectl get pvc
10. Ingress Rules	Check ingress configuration and backend services to ensure proper routing.	kubectl describe ingress <ingress-name>

These are the first key areas to investigate when diagnosing Kubernetes errors to quickly identify and resolve issues.

Error 1: CrashLoopBackOff Error

Cause:
The container is unable to start successfully and keeps restarting.

Solution:

1. Check the logs of the failing pod to identify the issue.
   Example:
   kubectl logs <pod-name>
2. Review the logs to understand the root cause of the issue. Common reasons include application misconfiguration, missing environment variables, or incorrect startup commands.
3. Verify that the container image is built and pushed correctly to the registry.
   Example:
   docker build -t <image-name> .
   docker push <image-name>
4. If environment variables are missing, update the deployment configuration.
   Example:
   Edit the deployment using:
   kubectl edit deployment <deployment-name>
   Add the missing environment variables under the env section.

Error 2: ImagePullBackOff

Cause:
Kubernetes is unable to pull the container image from the registry.

Solution:

1. Check the pod description to identify the error details.
   Example:
   kubectl describe pod <pod-name>
2. Verify the image name and tag in your deployment configuration. Ensure the image exists in the registry.

3. Authenticate with the container registry if required. For private registries, create a secret and link it to your deployment:
   Example:
   Create a secret:
   kubectl create secret docker-registry <secret-name> --docker-server=<registry-server> --docker-username=<username> --docker-password=<password>

Update the deployment to use the secret:
kubectl edit deployment <deployment-name>
Add the secret under the imagePullSecrets section.

Error 3: Pending Pods

Cause:
The pod remains in a Pending state because Kubernetes cannot schedule it.

Solution:

1. Check the pod's events for scheduling details.
   Example:
   kubectl describe pod <pod-name>
2. If insufficient resources are the issue, verify node capacity and usage.
   Example:
   kubectl get nodes
   kubectl describe node <node-name>
3. Adjust the resource requests and limits in your deployment.
   Example:
   Edit the deployment:
   kubectl edit deployment <deployment-name>
   Modify the resources section under containers.
4. If there are no matching nodes for affinity/anti-affinity rules, update or remove the rules.

Error 4: Node Not Ready

Cause:
A node is in a NotReady state due to issues like disk pressure, memory pressure, or network problems.

Solution:

1. Check the status of the nodes.
   Example:
   kubectl get nodes
2. Describe the problematic node to find the cause.
   Example:
   kubectl describe node <node-name>
3. Address the specific issue:
   - If disk pressure is mentioned, free up disk space.
   - If memory pressure is mentioned, reduce resource consumption or increase node capacity.
4. Restart the kubelet service on the affected node if necessary.

Error 5: Evicted Pods

Cause:
Pods are evicted due to insufficient resources on the node.

Solution:

1. Identify the evicted pod and reason.
   Example:
   kubectl get pods
   kubectl describe pod <pod-name>
2. Free up resources on the node by terminating unused pods or increasing node capacity.
3. Add more nodes to the cluster if resource requirements consistently exceed availability.
4. Adjust resource requests and limits in deployments.
   Example:
   Edit the deployment:

kubectl edit deployment <deployment-name>
Modify the resources section under containers.

Error 6: Pod Stuck in Terminating State

Cause:
The pod cannot terminate properly, often due to hanging processes or stuck volumes.

Solution:

1. Check the pod details and reason for termination delay.
   Example:
   kubectl get pods
2. If the pod is stuck, force-delete it.
   Example:
   kubectl delete pod <pod-name> --grace-period=0 --force
3. Investigate the application or volumes to identify the root cause.
4. Update the terminationGracePeriodSeconds in the deployment to allow graceful shutdown.
   Example:
   kubectl edit deployment <deployment-name>
   Modify the terminationGracePeriodSeconds value.

Error 7: Service Not Accessible

Cause:
The service is not exposing the application correctly.

Solution:

1. Check the service details.
   Example:
   kubectl get services
2. Verify the service configuration.
   Example:
   kubectl describe service <service-name>

3. Ensure the target port matches the container's exposed port.
4. If using a NodePort or LoadBalancer service, ensure that the firewall allows traffic on the specified port.
5. Test the service using a temporary pod.
   Example:
   kubectl run test-pod --image=busybox --rm -it -- /bin/sh
   Use curl to test the service from inside the cluster.

Error 8: Ingress Not Working

Cause:
Ingress is not routing traffic to the backend services.

Solution:

1. Verify the ingress configuration.
   Example:
   kubectl describe ingress <ingress-name>
2. Check the ingress controller logs for errors.
3. Ensure the DNS is pointing to the ingress IP.
4. Verify that the backend services are correctly configured and accessible.
5. Restart the ingress controller if necessary.

Error 9: ConfigMap Not Found

Cause:
The ConfigMap referenced in a pod or deployment does not exist.

Solution:

1. Verify the ConfigMap name in the deployment.
   Example:
   kubectl describe pod <pod-name>
2. Check the existing ConfigMaps.
   Example:
   kubectl get configmaps

3. Create the missing ConfigMap.
   Example:
   kubectl create configmap <configmap-name> --from-literal=<key>=<value>
4. Restart the affected deployment to apply the changes.
   Example:
   kubectl rollout restart deployment <deployment-name>

Error 10: Secrets Not Accessible

Cause:
The secret is not accessible to the pod.

Solution:

1. Verify the secret is created and accessible.
   Example:
   kubectl get secrets
2. Check if the secret is mounted in the pod.
   Example:
   kubectl describe pod <pod-name>
3. Update the deployment to mount the secret.
   Example:
   Edit the deployment:
   kubectl edit deployment <deployment-name>
   Add the secret under envFrom or volumeMounts.

Error 11: Pod Cannot Access PersistentVolume

Cause:
The PersistentVolume (PV) or PersistentVolumeClaim (PVC) is not properly bound.

Solution:

1. Verify the PVC status.
   Example:
   kubectl get pvc

2. If the PVC is in a Pending state, describe it to see the reason.
   Example:
   kubectl describe pvc <pvc-name>
3. Ensure the storage class and volume configuration match the PVC request.
   Example:
   Update the storage class or create a matching PV using:
   kubectl apply -f <pv-definition.yaml>
4. Check the pod's volume configuration and ensure the PVC is referenced correctly.

Error 12: Pod Fails Readiness Probe

Cause:
The application is not passing the readiness probe checks.

Solution:

1. Check the pod events for readiness probe failures.
   Example:
   kubectl describe pod <pod-name>
2. Review the readiness probe configuration in the deployment.
   Example:
   Edit the deployment:
   kubectl edit deployment <deployment-name>
   Verify the readinessProbe section for correct settings.
3. Test the readiness endpoint manually to confirm it responds as expected.
4. Adjust the probe parameters, such as initialDelaySeconds and periodSeconds, to match the application's startup time.

Error 13: Node Disk Pressure

Cause:
The node's disk usage exceeds the defined threshold, causing pod evictions.

Solution:

1. Check the node status.
   Example:
   kubectl get nodes
   kubectl describe node <node-name>
2. Identify and clean up unused Docker images and containers on the node.
   Example:
   docker system prune -f
3. Increase disk space or attach additional storage to the node.
4. If using a cloud provider, scale the cluster to add more nodes.

Error 14: Unauthorized Error When Accessing Kubernetes API

Cause:
The client does not have the required permissions to access the Kubernetes API.

Solution:

1. Verify the current user's permissions.
   Example:
   kubectl auth can-i <action> <resource>
2. Update the role or role binding associated with the user or service account.
   Example:
   Create or edit a role binding:
   kubectl create rolebinding <binding-name> --clusterrole=<role> --user=<user-name>
3. Ensure the correct kubeconfig file is being used.

Error 15: HPA (Horizontal Pod Autoscaler) Not Scaling

Cause:
The HPA is not scaling pods as expected.

Solution:

1. Check the HPA details.
   Example:
   kubectl describe hpa <hpa-name>
2. Verify the CPU or memory metrics are available.
   Example:
   kubectl top pod
3. Ensure that resource requests and limits are set in the deployment.
   Example:
   Edit the deployment:
   kubectl edit deployment <deployment-name>
   Add resource requests and limits under resources.
4. If metrics are missing, verify that the metrics server is running.
   Example:
   kubectl get pods -n kube-system | grep metrics-server

Error 16: Service Not Resolving DNS

Cause:
The DNS resolution within the cluster is not working.

Solution:

1. Check the CoreDNS pod status.
   Example:
   kubectl get pods -n kube-system | grep coredns
2. If CoreDNS pods are not running, describe the pods to identify the issue.
   Example:
   kubectl describe pod <coredns-pod> -n kube-system
3. Verify the kube-dns service is running.
   Example:
   kubectl get svc -n kube-system
4. Restart the CoreDNS pods if necessary.
   Example:
   kubectl rollout restart deployment coredns -n kube-system

Error 17: PVC in Lost State

Cause:
The PersistentVolumeClaim (PVC) is in a Lost state because the underlying storage is unavailable.

Solution:

1. Check the PV and PVC status.
   Example:
   kubectl get pv
   kubectl get pvc
2. Describe the PV to find the reason for the Lost state.
   Example:
   kubectl describe pv <pv-name>
3. Verify that the storage backend (e.g., NFS, EBS, etc.) is accessible and functioning.
4. If the storage backend is no longer available, recreate the PV and PVC with a new backend.

Error 18: ClusterRoleBinding Missing Permissions

Cause:
The ClusterRoleBinding does not have the necessary permissions for the intended action.

Solution:

1. Describe the ClusterRoleBinding to review its configuration.
   Example:
   kubectl describe clusterrolebinding <binding-name>
2. Edit the ClusterRoleBinding to add the required permissions.
   Example:
   Edit the binding:
   kubectl edit clusterrolebinding <binding-name>
   Update the rules section.

3. Test the permissions using:
   Example:
   kubectl auth can-i <action> <resource>

Error 19: Pod Status Unknown

Cause:
The pod status is Unknown due to node communication issues.

Solution:

1. Check the node status.
   Example:
   kubectl get nodes
2. Verify the pod's node allocation.
   Example:
   kubectl describe pod <pod-name>
3. Restart the kubelet service on the problematic node.
4. If the node is unreachable, remove it from the cluster.
   Example:
   kubectl delete node <node-name>

Error 20: DaemonSet Not Deploying Pods on All Nodes

Cause:
The DaemonSet is not deploying pods on all nodes due to affinity rules or insufficient resources.

Solution:

1. Describe the DaemonSet to review its configuration.
   Example:
   kubectl describe daemonset <daemonset-name>
2. Verify the node selector or affinity rules.
3. Check the node capacity and ensure there are sufficient resources for the DaemonSet pods.

4. Restart the DaemonSet to reapply its configuration.
   Example:
   kubectl rollout restart daemonset <daemonset-name>

Error 21: Error: The connection to the server was refused

Cause:
This occurs when the Kubernetes API server is not running or the kubeconfig is misconfigured.

Solution:

1. Verify that the API server is running on the master node.
   Example:
   systemctl status kube-apiserver
2. Check if the kubeconfig file is correctly set up.
   Example:
   Ensure KUBECONFIG points to the correct file:
   export KUBECONFIG=/path/to/config
3. Restart the API server if it's not running.
   Example:
   systemctl restart kube-apiserver
4. Test the connection to the cluster using:
   kubectl cluster-info

Error 22: Pods Stuck in ContainerCreating State

Cause:
This usually happens due to missing container images, insufficient resources, or issues with volume mounting.

Solution:

1. Describe the pod to get more details about the issue.
   Example:
   kubectl describe pod <pod-name>
2. Verify that the required container images are available in the registry.

3. If a volume mount is causing the problem, ensure that the referenced PersistentVolume is bound correctly.
4. If resource constraints are an issue, free up resources or scale the cluster.

Error 23: RBAC Permission Denied Error

Cause:
The user or service account does not have the required RBAC permissions.

Solution:

1. Check the current user's permissions.
   Example:
   kubectl auth can-i <action> <resource>
2. Add or update the RoleBinding or ClusterRoleBinding to grant the required permissions.
   Example:
   Create a binding:
   kubectl create rolebinding <name> --role=<role-name> --user=<user-name>
3. Verify the updated permissions.

Error 24: Pod IP Not Reachable

Cause:
Network issues in the Kubernetes cluster are preventing pod communication.

Solution:

1. Verify the pod's IP address using:
   Example:
   kubectl get pods -o wide
2. Check the network plugin (e.g., Calico, Flannel) to ensure it is running.
   Example:
   kubectl get pods -n kube-system | grep <network-plugin>
3. Restart the network plugin pods if necessary.

4. Ensure that the nodes can communicate with each other on the required ports.

Error 25: Kubelet Service Not Running

Cause:
The kubelet service is not running on a node.

Solution:

1. Check the status of the kubelet service.
   Example:
   systemctl status kubelet
2. If the service is stopped, start it:
   Example:
   systemctl start kubelet
3. Review the kubelet logs for errors.
   Example:
   journalctl -u kubelet
4. Fix any issues mentioned in the logs and restart the service.

Error 26: Deployment Not Updating

Cause:
Changes to the deployment are not being applied.

Solution:

1. Verify the deployment status.
   Example:
   kubectl get deployment <deployment-name>
2. Check if there is an update strategy specified in the deployment configuration.
3. Force a rollout restart to apply the changes.
   Example:
   kubectl rollout restart deployment <deployment-name>

4. Monitor the rollout progress.
   Example:
   kubectl rollout status deployment <deployment-name>

Error 27: FailedAttachVolume

Cause:
The volume cannot be attached to the pod.

Solution:

1. Describe the pod to find details about the volume attachment error.
   Example:
   kubectl describe pod <pod-name>
2. Verify that the PersistentVolumeClaim is bound and available.
   Example:
   kubectl get pvc
3. Ensure that the node where the pod is scheduled has access to the storage backend.
4. If the issue persists, delete and recreate the pod.

Error 28: Namespace Deletion Stuck

Cause:
The namespace is stuck in a terminating state due to resources that are not deleted.

Solution:

1. Check the resources remaining in the namespace.
   Example:
   kubectl get all -n <namespace-name>
2. Force delete the remaining resources.
   Example:
   kubectl delete <resource-type> <resource-name> -n <namespace-name> --grace-period=0 --force

3. If the namespace still doesn't delete, edit the namespace and remove the finalizers.
   Example:
   kubectl edit namespace <namespace-name>

Error 29: Pods Stuck in Init State

Cause:
The init container in the pod is unable to complete.

Solution:

1. Check the status of the init container.
   Example:
   kubectl describe pod <pod-name>
2. Verify the init container's command and ensure it is completing successfully.
3. Test the init container command manually.
4. Adjust the init container's configuration if necessary and redeploy the pod.

Error 30: Ingress Shows 404

Cause:
The ingress is not properly routing traffic to the backend service.

Solution:

1. Check the ingress configuration for routing rules.
   Example:
   kubectl describe ingress <ingress-name>
2. Verify that the backend service and pods are running and accessible.
3. Ensure the DNS is correctly pointing to the ingress IP.
4. Restart the ingress controller if required.
   Example:
   kubectl rollout restart deployment <ingress-controller-name>

Error 31: Invalid Memory/CPU Requests

Cause:
The resource requests or limits specified in the deployment are invalid or exceed node capacity.

Solution:

1. Check the pod description to find the invalid resource specification.
   Example:
   kubectl describe pod <pod-name>
2. Verify the current node capacity.
   Example:
   kubectl describe node <node-name>
3. Update the deployment to set valid resource requests and limits.
   Example:
   Edit the deployment:
   kubectl edit deployment <deployment-name>
   Ensure resources.requests and resources.limits are set appropriately.

Error 32: Service NodePort Not Accessible

Cause:
A NodePort service is not reachable from outside the cluster.

Solution:

1. Verify the service configuration.
   Example:
   kubectl get svc <service-name>
2. Check the firewall rules on the nodes to ensure the NodePort is open.
3. If the service is exposed through a specific interface, ensure the external IP is accessible.
4. Test connectivity to the NodePort using:
   Example:
   curl <node-ip>:<node-port>

Error 33: Pods Not Being Scheduled

Cause:
The scheduler cannot find a suitable node for the pod due to resource constraints or taints.

Solution:

1. Describe the pod to see why it is not being scheduled.
   Example:
   kubectl describe pod <pod-name>
2. Check for node taints that might prevent scheduling.
   Example:
   kubectl describe node <node-name>
3. Update the pod's tolerations or affinity rules if necessary.
4. Ensure sufficient resources are available on the nodes.

Error 34: Pod Cannot Connect to External Services

Cause:
The pod is unable to reach external services due to network configuration issues.

Solution:

1. Check the pod's network configuration.
   Example:
   kubectl exec <pod-name> -- ifconfig
2. Verify the cluster's DNS configuration.
   Example:
   kubectl get svc -n kube-system | grep coredns
3. Test external connectivity from within the pod.
   Example:
   kubectl exec <pod-name> -- curl <external-service-url>
4. Update the network policies to allow egress traffic.

Error 35: Deployment Not Scaling Properly

Cause:
The deployment is not scaling the number of replicas as expected.

Solution:

1. Verify the current number of replicas.
   Example:
   kubectl get deployment <deployment-name>
2. Check for resource constraints on the nodes.
3. Scale the deployment manually to test scaling functionality.
   Example:
   kubectl scale deployment <deployment-name> --replicas=<number>
4. If using HPA, ensure the metrics server is functioning correctly.

Error 36: Cluster Autoscaler Not Adding Nodes

Cause:
The cluster autoscaler is not adding nodes despite high resource demand.

Solution:

1. Verify the cluster autoscaler configuration.
   Example:
   kubectl get configmap -n kube-system cluster-autoscaler-config
2. Ensure the autoscaler has the required permissions to scale the cluster.
3. Check the instance group or node pool limits and increase them if necessary.
4. Review the autoscaler logs for errors.

Error 37: API Server Timeout

Cause:
The API server is unable to respond within the expected time, often due to high load or networking issues.

Solution:

1. Check the API server's logs for details.
   Example:
   kubectl logs <apiserver-pod-name> -n kube-system
2. Verify the API server's resource usage.
3. Scale the master nodes or increase API server resource limits if the load is high.
4. Optimize API calls to reduce unnecessary load.

Error 38: Pod Cannot Access ConfigMap

Cause:
The ConfigMap referenced in the pod is missing or incorrectly configured.

Solution:

1. Verify the ConfigMap exists.
   Example:
   kubectl get configmap
2. Check the pod description to confirm the ConfigMap is correctly referenced.
   Example:
   kubectl describe pod <pod-name>
3. If the ConfigMap is missing, recreate it.
   Example:
   kubectl create configmap <configmap-name> --from-literal=<key>=<value>
4. Restart the affected pods to load the updated ConfigMap.

Error 39: DaemonSet Pods Not Deploying

Cause:
DaemonSet pods are not being deployed to all nodes.

Solution:

1. Describe the DaemonSet to identify the issue.
   Example:
   kubectl describe daemonset <daemonset-name>
2. Check the node selector or tolerations in the DaemonSet configuration.
3. Ensure that the nodes have sufficient resources to schedule the pods.
4. Restart the DaemonSet.
   Example:
   kubectl rollout restart daemonset <daemonset-name>

Error 40: Pods Cannot Pull Secrets

Cause:
The pod is unable to access the secret due to incorrect permissions or configuration.

Solution:

1. Verify the secret exists.
   Example:
   kubectl get secret
2. Check the pod description to ensure the secret is referenced correctly.
   Example:
   kubectl describe pod <pod-name>
3. Recreate the secret if it's missing.
   Example:
   kubectl create secret generic <secret-name> --from-literal=<key>=<value>
4. Restart the pods to apply the updated secret.

Error 41: Service External-IP Pending

Cause:
A LoadBalancer service is stuck in the Pending state because the cloud provider's load balancer is not being provisioned.

Solution:

1. Verify the cloud provider integration with the cluster.
   Example:
   kubectl get nodes -o wide (Check if the nodes have the correct cloud provider labels)
2. Check the service description for details.
   Example:
   kubectl describe svc <service-name>
3. Ensure that the cloud provider account has sufficient permissions to create load balancers.
4. If using a local cluster, use a NodePort service instead of a LoadBalancer.

Error 42: Pod Terminated With Exit Code 137

Cause:
The pod was terminated due to an out-of-memory (OOM) error.

Solution:

1. Check the pod logs to confirm the OOM error.
   Example:
   kubectl logs <pod-name>
2. Update the deployment to increase the memory requests and limits.
   Example:
   kubectl edit deployment <deployment-name>
   Add or update resources.limits.memory and resources.requests.memory.
3. Monitor resource usage using:
   Example:
   kubectl top pod

Error 43: Failed to Start Pod Sandbox

Cause:
The container runtime is unable to start the pod sandbox.

Solution:

1. Check the kubelet logs for sandbox errors.
   Example:
   journalctl -u kubelet
2. Restart the container runtime on the node.
   Example:
   systemctl restart docker (or containerd depending on the runtime)
3. If the issue persists, check network configurations like CNI plugins.
4. Remove any stale sandboxes.
   Example:
   docker ps -a | grep <sandbox-id> and remove it.

Error 44: Helm Release Stuck in PENDING_INSTALL

Cause:
The Helm release is stuck due to errors during the installation process.

Solution:

1. Check the Helm release status.
   Example:
   helm status <release-name>
2. Check the logs of the failing pods.
3. Roll back the release.
   Example:
   helm rollback <release-name>
4. Fix the issue in the chart values or templates and try reinstalling.
   Example:
   helm upgrade --install <release-name> <chart-name>

Error 45: Ingress Returning 502 Bad Gateway

Cause:
The ingress is unable to route traffic to the backend service.

Solution:

1. Check the ingress logs for errors.
   Example:
   kubectl logs <ingress-controller-pod> -n kube-system
2. Verify the backend service and pod are running and accessible.
   Example:
   kubectl get svc and kubectl get pods
3. Ensure the backend service's target port matches the pod's exposed port.
4. Test the service manually to ensure it responds.

Error 46: Error: context deadline exceeded

Cause:
This error occurs when a Kubernetes API request times out.

Solution:

1. Check the API server logs for timeouts.
2. Increase the timeout for the kubectl command.
   Example:
   kubectl get pods --timeout=60s
3. Reduce the load on the API server by optimizing requests or scaling the master nodes.
4. Check network connectivity to the API server.

Error 47: Volume Mount Permissions Denied

Cause:
The container lacks permission to access the mounted volume.

Solution:

1. Verify the volume's permissions on the node.
2. Update the deployment to specify a security context.
   Example:

kubectl edit deployment <deployment-name>
Add:

```
securityContext:
  runAsUser: <uid>
  fsGroup: <gid>
```

3. Ensure the volume has the correct ownership and permissions.

Error 48: Pods Exceeding Resource Quota

Cause:
The namespace has a resource quota, and the requested resources exceed it.

Solution:

1. Check the resource quota for the namespace.
   Example:
   kubectl get resourcequota -n <namespace>
2. Adjust the resource requests and limits in the pod's configuration.
3. Increase the resource quota if necessary.
   Example:
   kubectl edit resourcequota <quota-name> -n <namespace>

Error 49: Cluster Certificate Expired

Cause:
The cluster certificates have expired, causing authentication failures.

Solution:

1. Verify the expiration date of the certificates.
   Example:
   kubeadm certs check-expiration
2. Renew the certificates.
   Example:
   kubeadm certs renew all
3. Restart the control plane components to apply the updated certificates.
4. Update the kubeconfig file if necessary.

Error 50: HPA Not Responding to Metrics

Cause:
The Horizontal Pod Autoscaler (HPA) is not scaling due to missing or invalid metrics.

Solution:

1. Check the HPA description for details.
   Example:
   kubectl describe hpa <hpa-name>
2. Verify that the metrics server is running and accessible.
   Example:
   kubectl get pods -n kube-system | grep metrics-server
3. Ensure that the deployment has resource requests defined for CPU and memory.
4. Restart the metrics server if it is not working properly.
   Example:
   kubectl rollout restart deployment metrics-server -n kube-system

Error 51: Pod IP Conflict

Cause:
Two pods are assigned the same IP due to CNI plugin misconfiguration.

Solution:

1. Check the CNI plugin logs.
   Example:
   kubectl logs <cni-plugin-pod-name> -n kube-system
2. Restart the CNI plugin pods to resolve transient issues.
   Example:
   kubectl rollout restart daemonset <cni-plugin-name> -n kube-system
3. Verify the pod CIDR configuration and ensure it doesn't overlap.

4. If necessary, reconfigure the CNI plugin with a new CIDR range.

Error 52: Service Account Not Found

Cause:
The service account referenced by a pod does not exist.

Solution:

1. Verify the service account exists.
   Example:
   kubectl get serviceaccount
2. If missing, create the service account.
   Example:
   kubectl create serviceaccount <account-name>
3. Ensure the pod references the correct service account in its configuration.

Error 53: Pod Fails Liveness Probe

Cause:
The liveness probe is failing, causing the pod to restart.

Solution:

1. Check the pod events for liveness probe failures.
   Example:
   kubectl describe pod <pod-name>
2. Test the liveness probe endpoint manually to confirm its response.
3. Update the liveness probe configuration to match the correct path or timeout.
   Example:
   kubectl edit deployment <deployment-name>
   Update the livenessProbe section.

Error 54: Cannot Delete Namespace

Cause:
The namespace is stuck due to finalizers on resources.

Solution:

1. Describe the namespace to identify the finalizers.
   Example:
   kubectl describe namespace <namespace-name>
2. Remove the finalizers manually.
   Example:
   kubectl edit namespace <namespace-name>
3. Delete the namespace again.
   Example:
   kubectl delete namespace <namespace-name>

Error 55: Cannot Access API from External Client

Cause:
The external client is unable to connect to the Kubernetes API.

Solution:

1. Check the API server's endpoint and ensure it’s accessible.
   Example:
   kubectl cluster-info
2. Verify the external client is using the correct kubeconfig.
3. Open the API server port in the firewall if necessary.
4. Test the connection using:
   Example:
   curl -k https://<api-server-ip>:<port>

Error 56: Ingress Controller Pods CrashLooping

Cause:
The ingress controller is unable to start due to configuration errors.

Solution:

1. Check the ingress controller pod logs.
   Example:
   kubectl logs <ingress-pod-name>
2. Verify the ingress controller configuration.
3. Restart the ingress controller pods.
   Example:
   kubectl rollout restart deployment <ingress-controller-name>
4. Test ingress functionality with a basic configuration.

Error 57: Kubelet Certificate Rotation Failing

Cause:
The kubelet cannot rotate its certificate due to permissions or expiration.

Solution:

1. Check the kubelet logs for rotation errors.
   Example:
   journalctl -u kubelet
2. Manually renew the kubelet certificate.
   Example:
   kubeadm certs renew kubelet
3. Restart the kubelet service.
   Example:
   systemctl restart kubelet

Error 58: Metrics Server Shows No Data

Cause:
The metrics server is not collecting or reporting data.

Solution:

1. Check the metrics server pod logs.
   Example:
   kubectl logs <metrics-server-pod> -n kube-system

2. Verify that the metrics server is deployed with valid configurations.
3. Restart the metrics server deployment.
   Example:
   kubectl rollout restart deployment metrics-server -n kube-system
4. Test metrics collection using:
   Example:
   kubectl top pod

Error 59: DNS Resolution Intermittent

Cause:
CoreDNS is experiencing intermittent issues due to high load or configuration errors.

Solution:

1. Check the CoreDNS pod logs.
   Example:
   kubectl logs <coredns-pod-name> -n kube-system
2. Scale up the CoreDNS deployment if the cluster has grown.
   Example:
   kubectl scale deployment coredns -n kube-system --replicas=<number>
3. Update the CoreDNS ConfigMap with optimized configurations.

Error 60: Pod Cannot Access PersistentVolume

Cause:
The volume is not attached to the pod due to misconfiguration.

Solution:

1. Check the PVC and PV status.
   Example:
   kubectl get pvc and kubectl get pv
2. Ensure the PV and PVC are bound correctly.

3. If the storage backend is unavailable, resolve the issue or recreate the volume.

Error 61: Error: Forbidden when Running kubectl Commands

Cause:
The user does not have sufficient permissions for the action.

Solution:

1. Check the current user's permissions.
   Example:
   kubectl auth can-i <action> <resource>
2. Update the Role or ClusterRole to grant the required permissions.
   Example:
   kubectl create clusterrolebinding <binding-name> --clusterrole=<role> --user=<user>

Error 62: Node Cannot Join Cluster

Cause:
The node is unable to join the cluster due to token expiration or network issues.

Solution:

1. Generate a new join token.
   Example:
   kubeadm token create --print-join-command
2. Ensure the node can reach the control plane node over the required ports.

Error 63: API Server High Latency

Cause:
The API server is under heavy load or resource constraints.

Solution:

1. Check API server logs for high latency events.
2. Increase the resources allocated to the API server.
3. Reduce the number of requests to the API server by optimizing scripts or automation.

Error 64: DaemonSet Pods Not Running on Specific Node

Cause:
The node has taints or insufficient resources.

Solution:

1. Check the node description for taints.
   Example:
   kubectl describe node <node-name>
2. Update the DaemonSet tolerations to match the node's taints.

Error 65: Service Not Forwarding Traffic

Cause:
The service configuration does not correctly route traffic to pods.

Solution:

1. Verify the service configuration.
   Example:
   kubectl describe svc <service-name>
2. Check the endpoints to ensure pods are registered.
   Example:
   kubectl get endpoints <service-name>

Error 66: Error: Unauthorized when Accessing API Server

Cause:
The token or credentials used are invalid or expired.

Solution:

1. Verify the credentials in the kubeconfig file.
2. Generate a new token if needed and update the configuration.

Error 67: Pod Logs Not Available

Cause:
The pod logs are not accessible due to a runtime or logging issue.

Solution:

1. Check the runtime logs (e.g., Docker or containerd).
2. Ensure logging is enabled and the log files are not deleted.

Error 68: Cannot Delete Stuck Pod

Cause:
The pod is stuck in a terminating state due to dependencies.

Solution:

1. Force delete the pod.
   Example:
   kubectl delete pod <pod-name> --grace-period=0 --force
2. Check the dependent resources and clean them up if necessary.

Error 69: Pods Restarting Frequently

Cause:
The application or environment is causing frequent pod restarts.

Solution:

1. Check the pod logs to identify the issue.
2. Verify resource requests and limits to ensure there is no OOM kill.

Error 70: Cluster Nodes Unreachable

Cause:
Nodes cannot communicate due to network issues.

Solution:

1. Check node status.
   Example:
   kubectl get nodes
2. Verify the network configuration and resolve connectivity issues.

Error 71: Pods Stuck in ImagePullBackOff

Cause:
The image pull fails due to incorrect credentials, a missing image, or connectivity issues.

Solution:

1. Check the pod's events for details.
   Example:
   kubectl describe pod <pod-name>
2. Verify the image exists in the container registry.
3. If it's a private registry, ensure the secret for authentication is configured and linked to the pod.
   Example:
   kubectl create secret docker-registry <secret-name> --docker-username=<username> --docker-password=<password>

Error 72: Node Disk Pressure Causes Pod Evictions

Cause:
Disk usage exceeds the threshold on the node, triggering pod evictions.

Solution:

1. Check the node status and conditions.
   Example:
   kubectl describe node <node-name>

2. Free up disk space by removing unused images and containers.
   Example:
   docker system prune -f
3. Add additional storage to the node if necessary.

Error 73: Pod Cannot Access Secret

Cause:
The secret referenced in the pod does not exist or is misconfigured.

Solution:

1. Verify the secret exists.
   Example:
   kubectl get secrets
2. Check the pod's description for the referenced secret.
   Example:
   kubectl describe pod <pod-name>
3. Recreate the secret if it's missing.
   Example:
   kubectl create secret generic <secret-name> --from-literal=<key>=<value>

Error 74: PVC Pending Due to StorageClass Issues

Cause:
The requested storage class is unavailable or misconfigured.

Solution:

1. Verify the PVC and storage class.
   Example:
   kubectl describe pvc <pvc-name>
2. Check the available storage classes.
   Example:
   kubectl get storageclass
3. Update the PVC to use a valid storage class.

Error 75: Cannot Scale Deployment Beyond Node Capacity

Cause:
The nodes lack sufficient resources to schedule additional pods.

Solution:

1. Check the deployment's resource requests and limits.
2. Verify node capacity.
   Example:
   kubectl describe nodes
3. Scale the cluster or add new nodes.

Error 76: CoreDNS Pods CrashLooping

Cause:
CoreDNS configuration errors or insufficient resources.

Solution:

1. Check the CoreDNS pod logs.
   Example:
   kubectl logs <coredns-pod-name> -n kube-system
2. Update the CoreDNS ConfigMap to fix any configuration issues.
3. Scale CoreDNS if needed.
   Example:
   kubectl scale deployment coredns -n kube-system --replicas=<number>

Error 77: HPA Not Responding to Custom Metrics

Cause:
The custom metrics are not being provided or collected.

Solution:

1. Verify the metrics server is functioning correctly.
2. Check if the custom metrics API is properly configured.

3. Update the HPA to reference the correct custom metrics.

Error 78: Pod Scheduling Ignored Node Selector

Cause:
The node selector specified in the pod is incorrect or mismatched.

Solution:

1. Check the pod's node selector configuration.
   Example:
   kubectl describe pod <pod-name>
2. Ensure the nodes have the required labels.
   Example:
   kubectl get nodes --show-labels
3. Update the pod's node selector to match the node labels.

Error 79: Ingress SSL/TLS Configuration Fails

Cause:
The ingress is misconfigured for SSL/TLS.

Solution:

1. Verify the TLS secret exists.
   Example:
   kubectl get secret <tls-secret-name>
2. Check the ingress rules for the tls section.
3. Restart the ingress controller if necessary.

Error 80: Kube-proxy Failing

Cause:
Kube-proxy is not functioning, affecting service routing.

Solution:

1. Check the kube-proxy pod logs.
   Example:
   kubectl logs <kube-proxy-pod-name> -n kube-system
2. Restart the kube-proxy daemonset.
   Example:
   kubectl rollout restart daemonset kube-proxy -n kube-system

Error 81: Cluster Autoscaler Scaling Too Slowly

Cause:
The cluster autoscaler is configured with a low scaling rate.

Solution:

1. Check the cluster autoscaler configuration.
2. Adjust the scaling parameters to increase speed.
3. Ensure the node pool has sufficient capacity to scale.

Error 82: Pod Logs Truncated

Cause:
Log rotation or limited buffer size in the container runtime.

Solution:

1. Check the container runtime log settings.
2. Update the runtime configuration to increase the log size.
3. Use centralized logging solutions like Fluentd or ELK for log aggregation.

Error 83: Pods Stuck in Evicted State

Cause:
The pods were evicted but not cleaned up.

Solution:

1. Delete the evicted pods.
   Example:
   kubectl delete pod <pod-name>
2. Address the resource issue causing the evictions.

Error 84: Pod Cannot Access Cluster Internal DNS

Cause:
CoreDNS or kube-dns is misconfigured or unavailable.

Solution:

1. Check the CoreDNS pod status.
2. Restart the CoreDNS pods.
   Example:
   kubectl rollout restart deployment coredns -n kube-system

Error 85: PersistentVolume Stuck in Released State

Cause:
The PV is bound to a PVC that no longer exists.

Solution:

1. Check the PV status.
   Example:
   kubectl describe pv <pv-name>
2. Delete and recreate the PV if necessary.

Error 86: Job Failing to Complete

Cause:
The job's pod fails repeatedly or completes with errors.

Solution:

1. Check the pod logs for errors.
2. Update the job's retry policy if necessary.

Error 87: ConfigMap Too Large

Cause:
ConfigMaps have a size limit of 1MB.

Solution:

1. Split the ConfigMap into smaller ConfigMaps.
2. Use external storage solutions for larger configurations.

Error 88: Services Intermittently Unreachable

Cause:
Networking or service configuration issues.

Solution:

1. Check the service endpoints.
   Example:
   kubectl get endpoints <service-name>
2. Restart the service if needed.

Error 89: Error: Connection Refused When Accessing Service

Cause:
The service is not routing traffic correctly.

Solution:

1. Verify the service and pod configurations.
2. Test connectivity using a temporary pod.

Error 90: Cluster High CPU Usage

Cause:
Control plane or node components are overutilized.

Solution:

1. Check resource usage.
   Example:
   kubectl top nodes
2. Scale up the cluster or optimize workloads.

Error 91: Pod Stuck in Pending Due to Node Affinity

Cause:
The pod cannot find a node that matches the affinity or anti-affinity rules.

Solution:

1. Check the pod's affinity rules.
   Example:
   kubectl describe pod <pod-name>
2. Verify the labels on the nodes.
   Example:
   kubectl get nodes --show-labels
3. Update the pod’s affinity configuration to match available nodes.

Error 92: Control Plane Components Not Starting

Cause:
Critical components like kube-apiserver or etcd are not starting due to configuration issues or resource constraints.

Solution:

1. Check the logs of the failing component.
   Example:
   journalctl -u kube-apiserver
2. Verify the configuration files, such as kube-apiserver.yaml or etcd.yaml.
3. Ensure sufficient resources (CPU, memory) are allocated to the control plane nodes.

Error 93: Pods Stuck in Terminating State

Cause:
Pods are stuck because Kubernetes cannot clean up resources or communicate with the node.

Solution:

1. Force delete the pods.
   Example:
   kubectl delete pod <pod-name> --grace-period=0 --force
2. Check the resources (e.g., volumes, secrets) that the pod was using.

Error 94: Cluster Upgrade Fails

Cause:
A cluster upgrade fails due to component version mismatches or configuration issues.

Solution:

1. Check the upgrade logs for errors.
2. Ensure all cluster components are compatible with the target Kubernetes version.
3. Retry the upgrade using the correct kubeadm commands.
   Example:
   kubeadm upgrade apply <version>

Error 95: Network Policy Blocking Traffic

Cause:
A network policy is unintentionally blocking traffic.

Solution:

1. Verify the applied network policies.
   Example:
   kubectl get networkpolicy -n <namespace>

2. Describe the policies to check the rules.
   Example:
   kubectl describe networkpolicy <policy-name> -n <namespace>
3. Update the policy to allow the required traffic.

Error 96: Pods Stuck in Unknown State

Cause:
Node communication with the API server is lost.

Solution:

1. Verify the node status.
   Example:
   kubectl get nodes
2. Restart the kubelet service on the affected node.
   Example:
   systemctl restart kubelet

Error 97: PersistentVolume Not Resizing

Cause:
The requested PVC resize is not being applied to the PV.

Solution:

1. Verify the PVC's status.
   Example:
   kubectl describe pvc <pvc-name>
2. Ensure the storage class allows volume expansion.
3. Resize the PV manually if supported by the backend.

Error 98: Ingress Redirect Loop

Cause:
Misconfigured ingress rules cause a loop in traffic redirection.

Solution:

1. Check the ingress rules for misconfigurations.
   Example:
   kubectl describe ingress <ingress-name>
2. Ensure that backend services are correctly configured and do not redirect traffic unnecessarily.

Error 99: Pod Security Context Misconfiguration

Cause:
The pod fails to start due to invalid security context settings.

Solution:

1. Check the security context configuration in the pod spec.
   Example:
   kubectl describe pod <pod-name>
2. Update the security context to valid values.
   Example:

```
securityContext:
  runAsUser: 1000
  runAsGroup: 1000
  fsGroup: 2000
```

Error 100: Pods Overloaded Due to Missing HPA

Cause:
Pods experience high load because no Horizontal Pod Autoscaler (HPA) is configured.

Solution:

1. Configure an HPA for the deployment.
   Example:
   kubectl autoscale deployment <deployment-name> --cpu-percent=80 --min=1 --max=10

2. Ensure the metrics server is running and resource requests/limits are defined for the pods.

Conclusion

Kubernetes is a powerful but intricate system, and its flexibility comes with its own set of challenges. This guide has aimed to equip you with the knowledge and tools necessary to address 100 of the most common Kubernetes errors. Each error has been dissected to uncover its root cause and followed by practical, step-by-step solutions to resolve the problem efficiently.

By understanding these common pitfalls and learning how to fix them, you can significantly enhance your troubleshooting skills and confidence in managing Kubernetes environments. Beyond resolving immediate issues, the insights provided in this guide encourage the adoption of best practices, such as monitoring resource usage, automating repetitive tasks, and implementing proper security measures, which are crucial for long-term success in Kubernetes operations.

As Kubernetes continues to evolve, so too will the challenges and opportunities it presents. Staying informed, leveraging community support, and continuously updating your knowledge base are key to staying ahead in this dynamic ecosystem. This guide is a stepping stone, but it’s your curiosity, experimentation, and willingness to learn that will make you a Kubernetes expert.

Thank you for taking this journey into Kubernetes problem-solving. May this guide empower you to build, manage, and scale your applications with greater confidence and reliability. Remember, every challenge is an opportunity to grow, and every error resolved brings you one step closer to mastering Kubernetes.