What causes the error "CrashLoopBackOff" in a Kubernetes pod, and how do you troubleshoot it?

- A "**CrashLoopBackOff**" error indicates that a pod is starting, crashing, and then Kubernetes is restarting it repeatedly. To troubleshoot:
- **Check Logs:** Use **kubectl logs <pod-name>** to inspect the pod logs for errors.
- **Describe the Pod:** Use **kubectl describe pod <pod-name>** to see events and the state of containers.
- **Common Issues:** Look for issues like incorrect environment variables, failed dependencies, or misconfigurations in the container's command or arguments.

Pod is starting, crashing and Kubernetes is restarting it repeatedly

```
% kubectl logs pod1

% kubectl describe pod pod1
```

Incorrect environment variables

Failed dependencies

Misconfigurations in container's command or arguments

Why might a Kubernetes pod be stuck in the "Pending" state, and how do you resolve it?

- A pod in the "**Pending**" state means it's waiting for resources to be allocated. Possible causes and resolutions:
- **Insufficient Resources:** The cluster might lack enough CPU or memory. Check available resources with **kubectl describe nodes**.
- **Unschedulable Node:** The pod might have node affinity or taints that prevent scheduling. Check the node's taints and pod's affinity rules.
- **Missing Persistent Volume:** If the pod requires a persistent volume, ensure that the appropriate **PersistentVolume** exists and is bound.

Waiting for resources to be allocated

Lack enough CPU or memory

```
% kubectl describe nodes
```

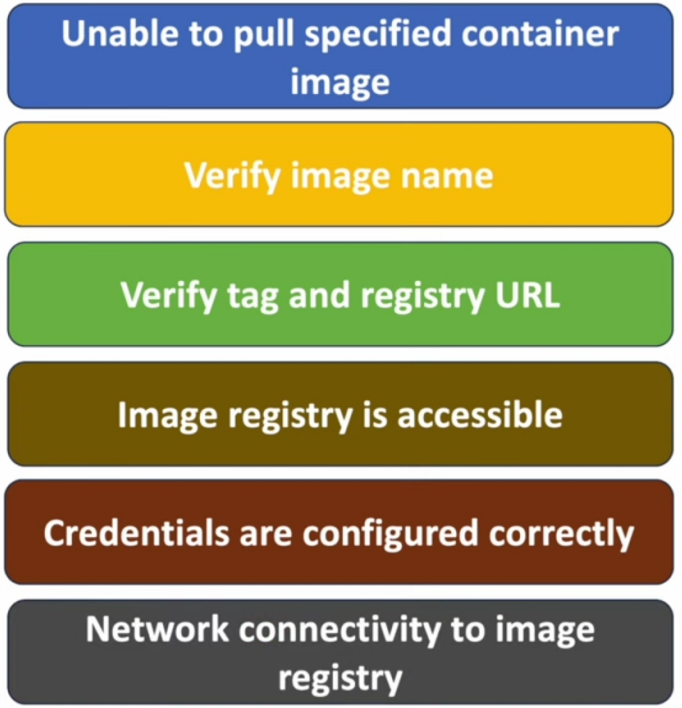
Node affinity or taints prevent scheduling

Check node's taints and pod's affinity rules

Appropriate PersistentVolume exists and is bound

What does the error "ImagePullBackOff" indicate, and how do you fix it?

- The "**ImagePullBackOff**" error indicates that Kubernetes is unable to pull the specified container image.
- **Check Image Name:** Verify the image name, tag, and registry URL in the pod's YAML.
- **Registry Access:** Ensure that the image registry is accessible and that the credentials (if needed) are correctly configured in a Secret.
- **Network Issues:** Check the network connectivity to the image registry.



How do you resolve the "ErrImagePull" error in Kubernetes?

- "**ErrImagePull**" is similar to "**ImagePullBackOff**" but is specific to an immediate failure in pulling the image. To resolve:
- **Correct Image Name:** Ensure the image name, tag, and registry URL are correct.
- **Check Secrets:** If pulling from a private registry, ensure the Secret with credentials is correctly referenced in the pod spec.
- **Registry Access:** Ensure that the image is available in the specified registry and that there are no access restrictions.

Similar to ImagePullBackOff

Specific to immediate failure in pulling image

Image name and tag are correct

Registry URL is correct

Credentials are correctly referenced

Image is available and no access restrictions

What might cause a pod to remain in the "Terminating" state indefinitely, and how do you resolve it?

- A pod stuck in the "**Terminating**" state usually occurs when Kubernetes can't gracefully shut down the containers. Possible causes and resolutions:
- **Grace Period:** Check if the pod has a long **terminationGracePeriodSeconds** set. You can force delete the pod with **kubectl delete pod <pod-name> --grace-period=0 --force**.
- **Finalizers:** The pod might have finalizers that are preventing deletion. You can remove them by editing the pod with **kubectl edit pod <pod-name>** and deleting the finalizers section.
- **Network Issues:** Ensure that the pod is accessible and that there are no network issues preventing it from terminating.

Cannot gracefully shut down the containers

Long terminationGracePeriodSeconds

```
% kubectl delete pod pod1 --grace-period=0 --force
```

Finalizers that are preventing deletion

```
% kubectl edit pod pod1
```

Pod is accessible and no network issues

What is a "NodeNotReady" error, and how do you troubleshoot it?

- "**NodeNotReady**" indicates that a node is not in a ready state to schedule pods. Troubleshooting steps:
- **Check Node Status:** Use **kubectl get nodes** to check the node's status.
- **Node Issues:** Log into the node and check for issues like high CPU/memory usage, disk pressure, or network problems.
- **Kubelet Logs:** Inspect the kubelet logs on the node (journalctl -u kubelet) to find errors related to node status.
- **Restart Services:** Restart kubelet or other relevant services on the node if necessary.

Node is not in ready state to schedule pods

```
% kubectl describe nodes
```

Check high CPU or memory usage

Disk pressure or network problems

Inspect kubelet logs on the node

Restart kubelet or other relevant services

How do you address a "PVC not bound" error in Kubernetes?

- A "**PVC not bound**" error occurs when a **PersistentVolumeClaim** (PVC) is not bound to a **PersistentVolume** (PV). Steps to resolve:
- **Check PVC and PV:** Use **kubectl describe pvc <pvc-name>** and **kubectl get pv** to see the status of the PVC and available PVs.
- **StorageClass Mismatch:** Ensure that the **StorageClass** of the PVC matches a PV or dynamically provisioned storage.
- **Volume Availability:** Verify that there's a matching PV with sufficient capacity and the correct access mode.

PVC is not bound to a PV

```
kubectl describe pvc <pvc-name>

kubectl get pv
```

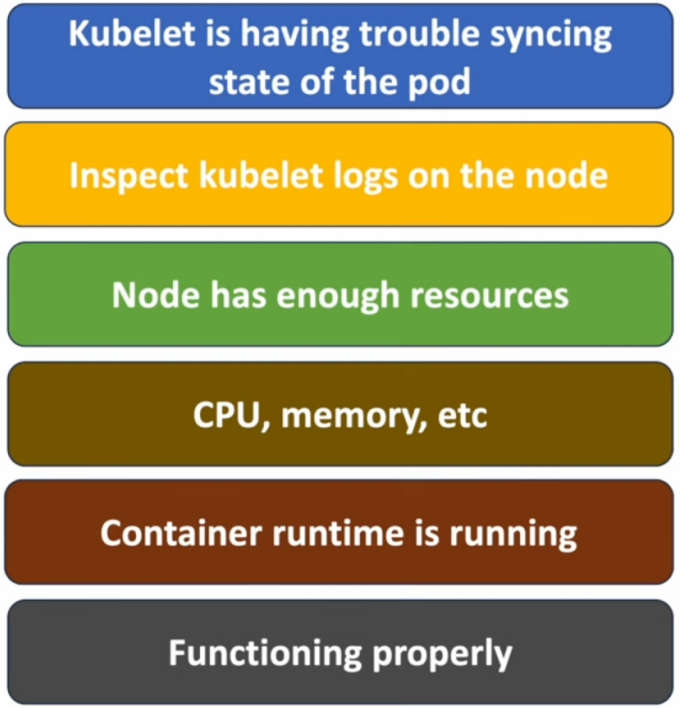
StorageClass of PVC matches a PV

Dynamically provisioned storage

Matching PV with sufficient capacity and correct access mode

What does the "Error syncing pod" message mean, and how do you troubleshoot it?

- The "**Error syncing pod**" message indicates that the kubelet is having trouble syncing the state of the pod. Troubleshooting steps:
- **Check Kubelet Logs:** Inspect the kubelet logs on the node where the pod is scheduled (journalctl -u kubelet).
- **Resource Constraints:** Verify that the node has enough resources (CPU, memory) to handle the pod.
- **Container Runtime:** Check if the container runtime (e.g., Docker, containerd) is running and functioning properly.



Why might a Kubernetes service not be accessible from within the cluster, and how do you resolve it?

- If a service isn't accessible from within the cluster, potential issues include:
- **Pod Selector Mismatch:** Ensure that the service's selector matches the labels of the target pods.
- **Network Policies:** Check for network policies that might be blocking traffic to or from the pods.
- **IPTables Issues:** On the nodes, verify that iptables rules are correctly configured to allow traffic.
- **DNS Resolution:** Ensure that the DNS in the cluster is correctly resolving service names.

Service's selector matched labels of target pods

Check network policies

Blocking network traffic to or from pods

Iptables rules are correctly configured

Allow traffic

DNS in the cluster is correctly resolving service names

How do you fix the "Failed to create pod sandbox" error?

- "**Failed to create pod sandbox**" is related to issues in setting up the container runtime environment. Troubleshooting steps:
- **Check Runtime Logs:** Inspect the logs for the container runtime (e.g., Docker, containerd) on the node.
- **Node Resources:** Verify that the node has enough CPU, memory, and disk space.
- **Network Plugins:** Ensure that the CNI (Container Network Interface) plugins are correctly installed and functioning.
- **Restart Node Services:** Restarting kubelet and container runtime services might resolve transient issues.

Issues in setting up container runtime environment

Inspect logs from container runtime

Node has enough CPU, memory and disk space

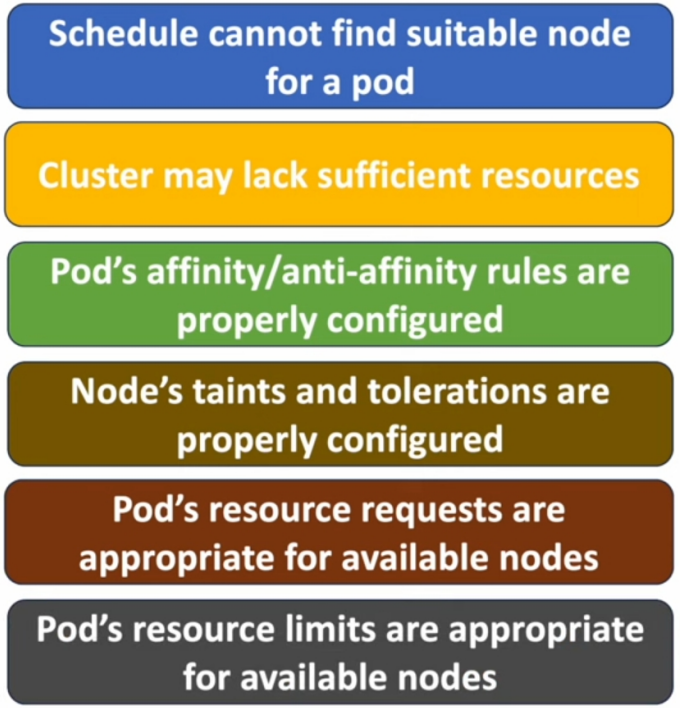
CNI plugins are correctly installed and functioning

Restart kubelet and container runtime services

Might resolve transient issues

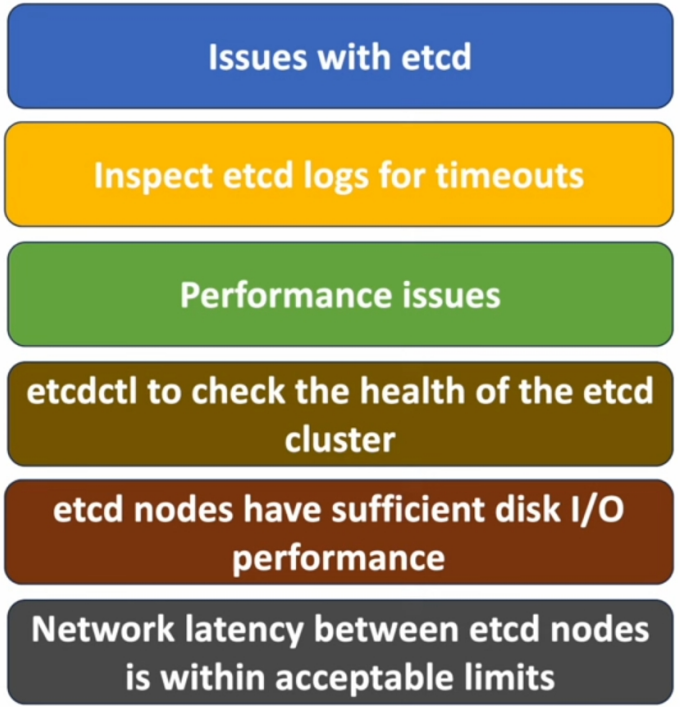
What causes "FailedScheduling" errors in Kubernetes, and how do you resolve them?

- "**FailedScheduling**" occurs when the scheduler can't find a suitable node for a pod. Possible causes and solutions:
- **Insufficient Resources:** The cluster may lack sufficient resources. Check node availability and resource usage.
- **Node Affinity/Taints:** Verify that the pod's affinity/anti-affinity rules and the node's taints and tolerations are properly configured.
- **Pod Resource Requests:** Ensure that the pod's resource requests and limits are appropriate for the available nodes.



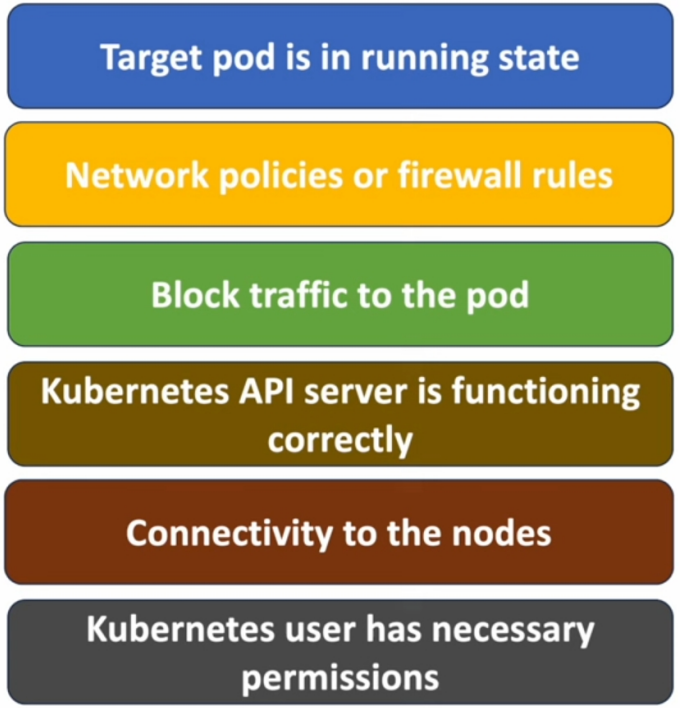
How do you resolve "etcdserver: request timed out" errors in Kubernetes?

- This error indicates issues with etcd, the key-value store used by Kubernetes. Steps to resolve:
- **Check etcd Logs:** Inspect etcd logs for timeouts or performance issues.
- **Cluster Health:** Use **etcdctl** to check the health of the etcd cluster.
- **Disk I/O:** Ensure that the etcd nodes have sufficient disk I/O performance.
- **Network Latency:** Verify that network latency between etcd nodes is within acceptable limits.

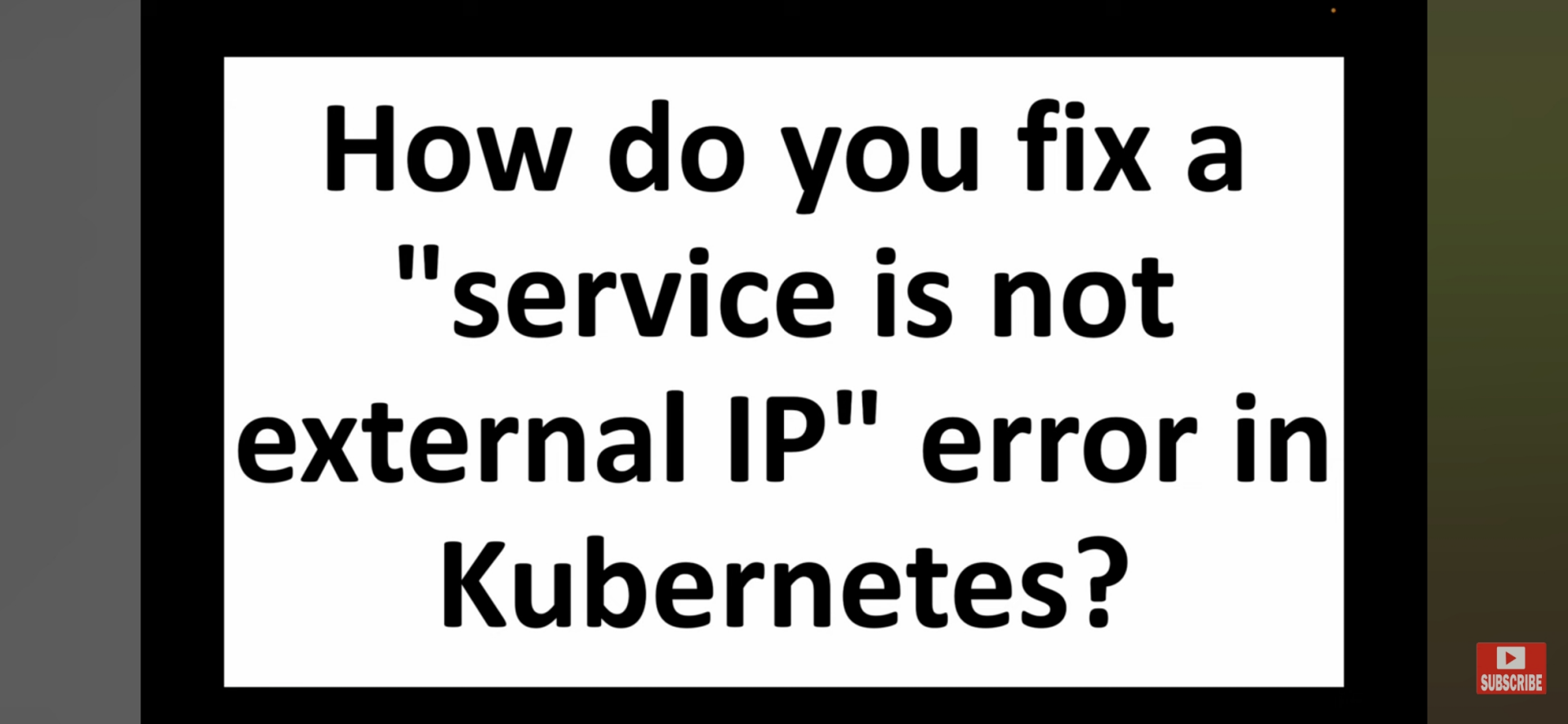


Why might a "kubectl exec" command fail, and how do you troubleshoot it?

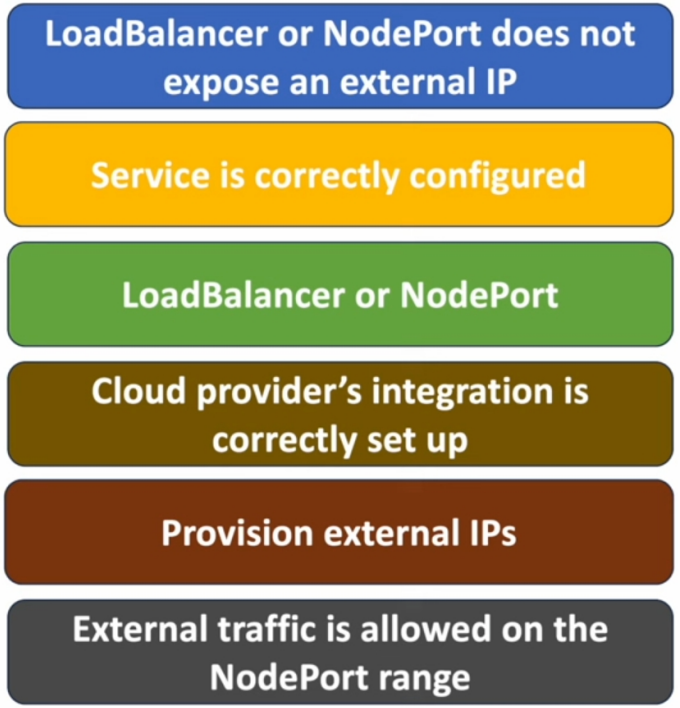
- A "**kubectl exec**" command can fail due to several reasons:
- **Pod Not Running:** Ensure the target pod is in a running state.
- **Network Connectivity:** Check network policies or firewall rules that might block traffic to the pod.
- **APIServer Issues:** Verify that the Kubernetes API server is functioning correctly and has connectivity to the nodes.
- **Permission Issues:** Ensure that the Kubernetes user has the necessary permissions to execute commands in the pod.



How do you fix a "service is not external IP" error in Kubernetes?



- This error occurs when a service of type LoadBalancer or NodePort doesn't expose an external IP. Steps to resolve:
- **Check Service Type:** Ensure the service is correctly configured as LoadBalancer or NodePort.
- **Cloud Provider Integration:** If using a LoadBalancer, ensure that the cloud provider's integration is correctly set up and that it can provision external IPs.
- **NodePort Range:** Verify that the external traffic is allowed on the NodePort range in the cluster's firewall settings.



How do you resolve "certificate signed by unknown authority" errors in Kubernetes?

- This error typically occurs when there's an issue with the TLS certificates used in the cluster. Troubleshooting steps:
- **Check Certificate Validity:** Verify that the certificates are valid and haven't expired.
- **CA Bundle Configuration:** Ensure that the CA bundle is correctly configured and includes the necessary root and intermediate certificates.
- **Kubelet Configuration:** Ensure that the kubelet is configured to use the correct CA certificate for verifying the API server's identity.