Kubernetes troubleshooting can be challenging due to its distributed and dynamic nature. Here are some common issues encountered in Kubernetes environments and potential resolutions:

**1. Pods Stuck in Pending State**

**Issue**: Pods remain in a "Pending" state indefinitely.

* **Causes**:
  + Insufficient resources (CPU/memory) on nodes.
  + NodeSelector, Tolerations, or Affinity rules are too restrictive.
  + Issues with persistent volumes (PV) or persistent volume claims (PVC).

**Resolution**:

* Check available resources on the nodes: kubectl describe nodes.
* Review scheduling constraints: kubectl describe pod <pod-name>.
* Investigate storage issues if using PV/PVC.

**2. CrashLoopBackOff**

**Issue**: Pods crash and restart continuously.

* **Causes**:
  + Application misconfiguration or bug.
  + Readiness/Liveness probes misconfigured.
  + Incorrect environment variables or secrets.

**Resolution**:

* Review logs for the crashing container: kubectl logs <pod-name> -c <container-name>.
* Inspect pod events: kubectl describe pod <pod-name>.
* Check probes configuration in your deployment.
* Verify environment variables, secrets, and config maps.

**3. ImagePullBackOff**

**Issue**: Kubernetes can't pull container images.

* **Causes**:
  + Incorrect image name or tag.
  + Lack of access to private container registries.
  + Network issues preventing pulling from registry.

**Resolution**:

* Check image names and tags in your deployment: kubectl describe pod <pod-name>.
* Ensure Kubernetes has the correct imagePullSecrets for private registries.
* Investigate DNS or network connectivity to the container registry.

**4. FailedScheduling**

**Issue**: Pods cannot be scheduled on any node.

* **Causes**:
  + Lack of resources (CPU, memory) on the available nodes.
  + NodeSelector, taints, or tolerations preventing scheduling.
  + Pod anti-affinity or resource limits are too restrictive.

**Resolution**:

* Check node capacity and pod resource requests: kubectl describe nodes.
* Examine scheduling constraints in the pod: kubectl describe pod <pod-name>.
* Adjust resource requests/limits or scaling of nodes in the cluster.

**5. Node Not Ready**

**Issue**: Nodes are in a "NotReady" state.

* **Causes**:
  + Network issues or node connectivity problems.
  + Issues with kubelet or Docker runtime.
  + Disk pressure, memory issues, or other resource exhaustion.

**Resolution**:

* Check the node's status and events: kubectl describe node <node-name>.
* Investigate kubelet and container runtime logs on the node.
* Ensure there are no resource pressures on the node (e.g., disk or memory).

**6. DNS Resolution Failures**

**Issue**: Pods cannot resolve internal or external DNS names.

* **Causes**:
  + Misconfigured or non-functional CoreDNS service.
  + Network policies blocking DNS traffic.
  + Firewall rules or outbound connectivity issues.

**Resolution**:

* Check the CoreDNS pods: kubectl get pods -n kube-system | grep coredns.
* Investigate DNS logs: kubectl logs <coredns-pod-name> -n kube-system.
* Ensure there are no network policies blocking DNS traffic in the cluster.

**7. High CPU/Memory Usage**

**Issue**: Pods or nodes are experiencing resource exhaustion.

* **Causes**:
  + Misconfigured resource requests/limits.
  + Inefficient application code causing resource spikes.
  + Pod memory leaks.

**Resolution**:

* Monitor pod and node resources: kubectl top pod, kubectl top node.
* Set proper resource requests and limits in your pod manifests.
* Investigate resource consumption patterns with monitoring tools like Prometheus or Datadog.

**8. Service Not Reachable**

**Issue**: A service is not accessible or responds with timeouts.

* **Causes**:
  + Misconfigured service or network policies.
  + Problems with kube-proxy or iptables rules.
  + Ingress misconfigurations or issues with load balancers.

**Resolution**:

* Check service details: kubectl describe service <service-name>.
* Inspect endpoints for the service: kubectl get endpoints <service-name>.
* Validate network policies or ingress configuration.
* Review logs of ingress controllers or load balancers.

**9. PersistentVolume Not Bound**

**Issue**: PersistentVolumeClaims (PVCs) are stuck in a "Pending" state.

* **Causes**:
  + No matching PersistentVolume (PV) available.
  + StorageClass misconfiguration.
  + PV access modes not matching PVC requirements.

**Resolution**:

* Verify available PVs: kubectl get pv.
* Check PVC and PV compatibility (storage class, access modes): kubectl describe pvc <pvc-name>.
* Ensure that the storage backend is functioning properly.

**10. ETCD Issues**

**Issue**: ETCD database becomes unhealthy, affecting the control plane.

* **Causes**:
  + ETCD cluster nodes are down or out of sync.
  + Storage or disk space issues on ETCD nodes.
  + High network latency or connectivity issues.

**Resolution**:

* Check ETCD member health: etcdctl endpoint health.
* Ensure ETCD has sufficient disk space and I/O performance.
* Monitor ETCD metrics for latency and performance bottlenecks.

**11. RBAC Issues**

**Issue**: Unauthorized errors due to misconfigured Role-Based Access Control (RBAC).

* **Causes**:
  + Incorrect roles or role bindings for users or service accounts.
  + Missing RBAC permissions for actions on certain resources.

**Resolution**:

* Check RBAC role bindings: kubectl get rolebinding <name>.
* Review cluster roles and permissions: kubectl describe clusterrole <role-name>.
* Adjust the necessary roles and bindings for proper access control.

If you encounter other specific issues in Kubernetes, let me know for more detailed assistance!