List of possible errors of the Kubernetes ecosystem

Resolving issues in Kubernetes can be systematic and efficient if you follow a structured approach. So, in this article, we will try to discuss different objects and associated possible errors and we will also develop a progressive and right directional approach to figure out core reason of causes.

1. **Pod and Container Issues**

- ***CrashLoopBackOff***: Pod is crashing repeatedly due to incorrect configurations or application failures.
- **ImagePullBackOff**: Kubernetes cannot pull the specified image from the container registry.
- **ErrImagePull**: Error pulling the image (due to incorrect image name, tag, or authentication issues).
- ***OOMKilled***: Pod is terminated because it exceeds its memory limits.
- **ContainerCreating**: Pod is stuck in ContainerCreating state due to volume, network, or image pull issues.
- **Completed Pods**: Pods are stuck in Completed state due to job completion but not being cleaned up.
- **Terminating Pods**: Pods are stuck in Terminating state, often caused by finalizer misconfigurations or unresponsive cleanup.
- **PodEviction**: Pods are evicted from nodes due to resource pressure or taints and tolerations mismatch.

2. **Deployment and ReplicaSet Issues**

- **ReplicaSet Misalignment**: Desired replicas don't match actual replicas due to scheduling failures or insufficient resources.
- **FailedScheduling**: Pods fail to schedule due to resource constraints, taints, or node affinity/anti-affinity rules.
- **Rolling Update Issues**: New deployments cause downtime due to misconfigured readiness/liveness probes or missing maxUnavailable/maxSurge settings.
- **HPA (Horizontal Pod Autoscaler) Issues**:
  - HPA not scaling due to missing metrics (e.g., CPU/Memory not exposed).
  - Incorrect resource requests/limits affecting scaling decisions.

3. **Networking Issues**

- **DNS Resolution Failures**: Pods cannot resolve service names due to CoreDNS misconfigurations or failures.
- ***Connection Refused***: Services are not accessible due to misconfigured or absent network policies.
- **Service Not Reachable**: Issues with ClusterIP, NodePort, or LoadBalancer services caused by configuration errors or cloud provider limitations.
- **NetworkPolicy Blocking Traffic**: Improperly defined NetworkPolicies block desired ingress/egress traffic.

- **Kube-proxy Issues**: Problems with service networking due to kube-proxy misconfigurations.

4. **Storage Issues**

- **VolumeMount Failure**: Persistent Volumes (PVs) fail to attach or mount to Pods.
- **PVC Pending**: Persistent Volume Claims (PVCs) remain in a Pending state due to insufficient or unavailable storage.
- **Data Loss**: Misconfigured StatefulSets or volume binding lead to data loss when Pods are rescheduled.

5. **Control Plane Issues**

- **API Server Unreachable**: Issues due to high load, misconfiguration, or certificate problems.
- **etcd Performance Degradation**: etcd cluster has high latency or data corruption, affecting the overall cluster performance.
- **Controller Manager Issues**: Failures in applying desired state due to misconfigurations or bugs.
- **Scheduler Failures**: Scheduler fails to allocate Pods to nodes due to configuration or resource exhaustion.

6. **Authentication and Authorization Issues**

- **RBAC Denied Errors**: Users or Pods cannot perform actions due to missing or misconfigured Role-Based Access Control (RBAC) permissions.
- **Service Account Issues**: Missing or misconfigured Service Accounts cause authentication failures for workloads.
- **Webhook Authentication Errors**: External authenticators fail due to connectivity or misconfigurations.

7. **Node Issues**

- ***Node Not Ready***: Nodes are marked NotReady due to kubelet, disk pressure, or network issues.
- **DiskPressure or MemoryPressure**: Nodes face resource exhaustion, evicting Pods or causing degraded performance.
- **Uncordoned Nodes**: Nodes remain cordoned after maintenance or upgrades, causing scheduling failures.
- **Kubelet Failures**: Kubelet crashes or becomes unresponsive, affecting Pods on the node.

8. **Ingress and Load Balancer Issues**

- **Ingress Not Routing**: Misconfigured Ingress resources or absent Ingress controllers.
- **SSL/TLS Errors**: Missing or improperly configured certificates for HTTPS traffic.
- **LoadBalancer Not Created**: Cloud Load Balancer fails to provision due to provider-specific limitations or misconfigurations.
- **Backend Service Unhealthy**: Services backing the Ingress are not passing health checks.

9. **Configuration Issues**

- **Environment Variables Missing**: Pods crash due to missing or incorrectly set environment variables.
- **Secret and ConfigMap Issues**: Secrets or ConfigMaps are not mounted correctly, causing application errors.
- **Improper Resource Requests and Limits**: Over or under-provisioned resource requests causing scheduling or runtime failures.

10. **Logging and Monitoring Issues**

- **Log Aggregation Issues**: Centralized logging tools (e.g., Fluentd, Logstash) fail to collect or forward logs.
- **Metrics Missing**: Metrics server or Prometheus is misconfigured, leading to missing metrics for scaling or monitoring.
- **Application Debugging Issues**: Lack of application-specific logs due to inadequate logging configurations.

11. **Cluster Upgrades and Maintenance Issues**

- **Version Skew**: Components (e.g., kube-apiserver, kubelet) running incompatible versions after upgrades.
- **Downtime During Upgrade**: Misconfigured workloads cause downtime during control plane or node upgrades.
- **Custom CRD Breakages**: Breaking changes in custom resource definitions (CRDs) due to API deprecations.

12. **Scaling and Performance Issues**

- **Node Resource Exhaustion**: Nodes run out of CPU, memory, or disk, causing degraded performance or Pod evictions.
- **Cluster Autoscaler Failures**: Cluster Autoscaler fails to provision nodes due to cloud provider limitations or incorrect configurations.
- **Throttling in API Server**: High API request volume causes API server throttling.

13. **Debugging and Observability Challenges**

- **Lack of Metrics or Logs**: Missing metrics/logs make debugging difficult.
- **Event Overwrites**: Kubernetes events have a short retention window, leading to lost debugging context.
- **Misinterpreting Liveness/Readiness Probes**: Incorrectly configured probes cause unnecessary Pod restarts.