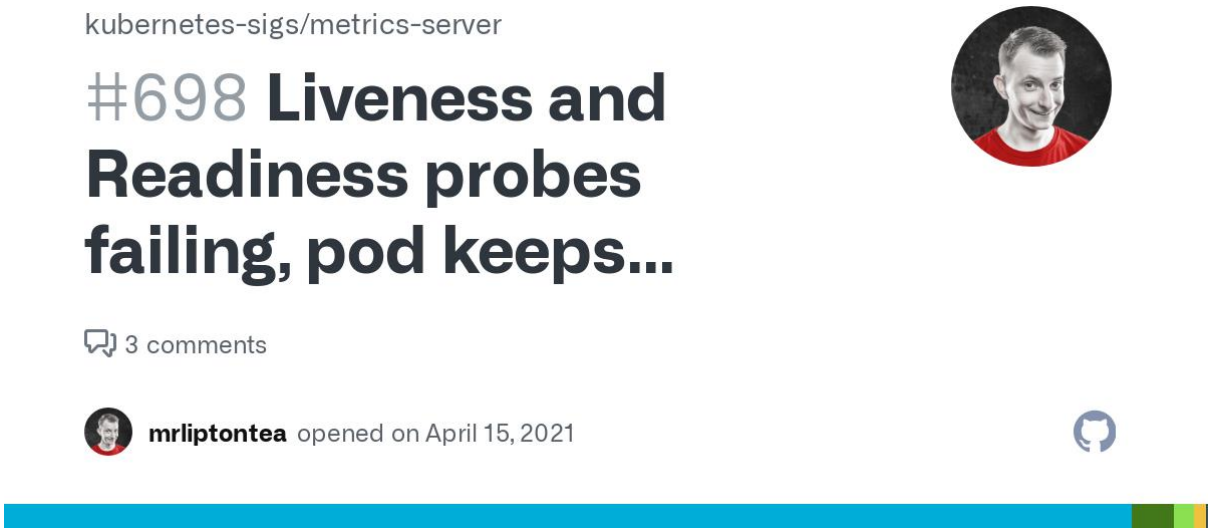
Part 16: Kubernetes Real-Time Troubleshooting

Introduction 🌐

Welcome to the world of Kubernetes troubleshooting, where every challenge is an opportunity to sharpen your skills and emerge victorious. Join us as we embark on a journey through common real-time scenarios, unraveling mysteries, and uncovering solutions along the way.

Scenario 76: Failed Start Readiness/Liveness Probes



Symptoms: Pods are frequently restarted or fail to become ready due to failed readiness or liveness probes.

Diagnosis: Describe the affected pods (`kubectl describe pod <pod_name>`) and inspect the events and probe configurations.

Solution:

1. Verify the probe configurations in the pod spec (e.g., paths, ports, initial delays).
2. Ensure the application endpoints are properly serving the probe requests.
3. Adjust probe thresholds and timings to accommodate the application's startup and runtime characteristics.

Scenario 77: Helm Release Fails to Upgrade

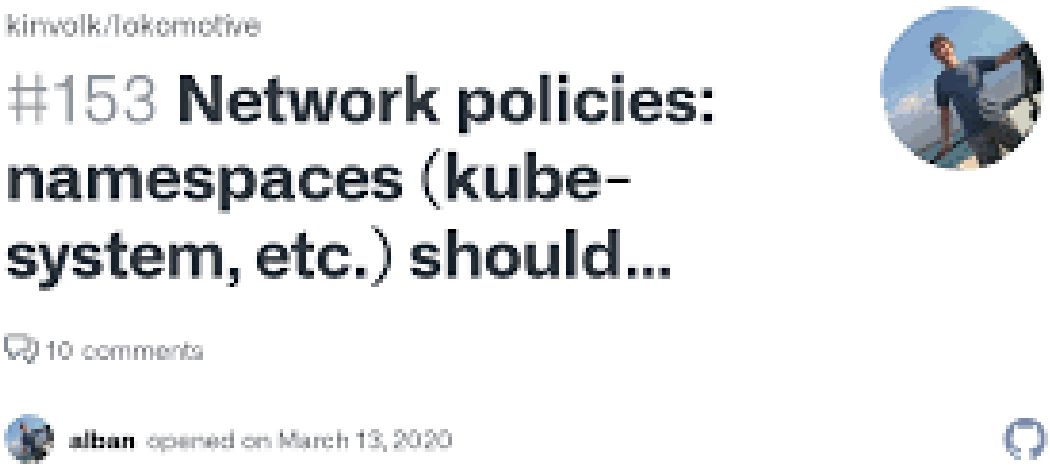
Symptoms: Helm release upgrade fails with errors and does not deploy new changes.

Diagnosis: Inspect the Helm release status (`helm status <release_name>`) and review the error logs.

Solution:

1. Identify and resolve the specific errors preventing the upgrade.
2. Roll back to the previous successful release (`helm rollback <release_name> <revision_number>`).
3. Ensure that Helm charts and values files are correctly configured and compatible.

Scenario 78: Cross-Namespace Network Policy Issues



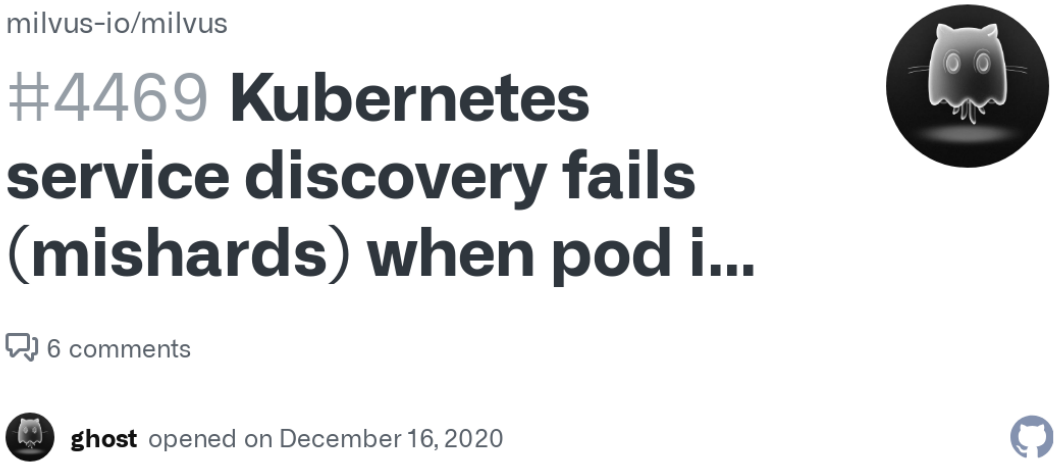
Symptoms: Pods in different namespaces cannot communicate due to network policies blocking traffic.

Diagnosis: Describe the network policies in both namespaces (`kubectl describe networkpolicy -n <namespace>`) and inspect pod labels and policy selectors.

Solution:

1. Review and adjust Network Policies to explicitly allow traffic between the namespaces.
2. Ensure that pod labels and selectors match the intended traffic rules across namespaces.
3. Test connectivity and refine policies as needed to achieve the desired traffic flow.

Scenario 79: Service Discovery Failure



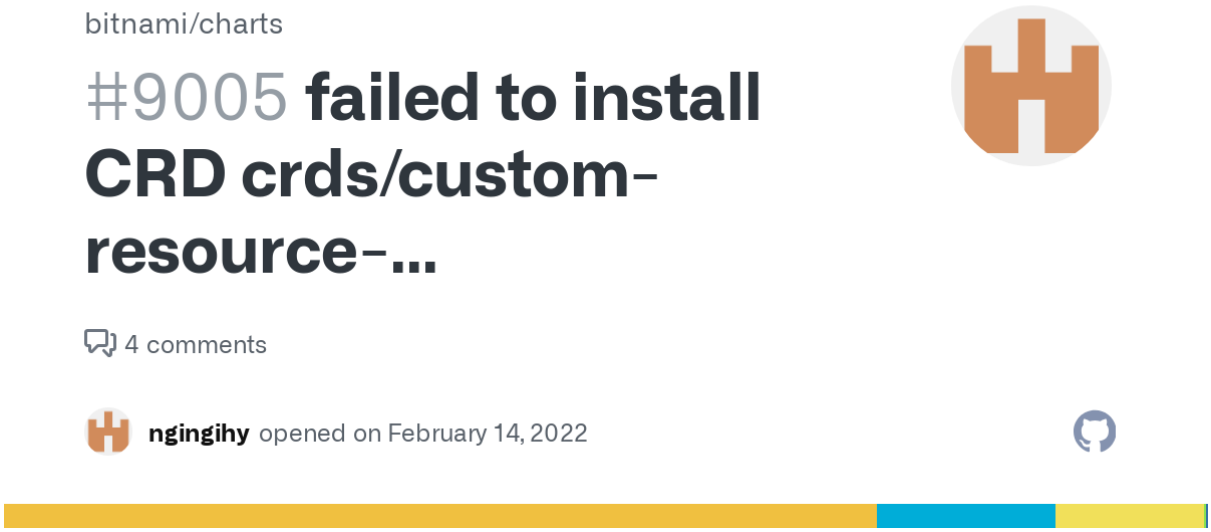
Symptoms: Pods are unable to resolve DNS names of other services within the cluster.

Diagnosis: Inspect DNS configuration (`kubectl get cm -n kube-system coredns`) and pod DNS settings (`kubectl exec -it <pod_name> -- cat /etc/resolv.conf`).

Solution:

1. Restart the CoreDNS pods (`kubectl delete pod -n kube-system -l k8s-app=kube-dns`).
2. Verify that DNS service IPs are reachable from pods (`kubectl exec -it <pod_name> -- nslookup <service_name>`).
3. Ensure that pod DNS settings are correctly configured to use the cluster DNS service.

Scenario 80: Custom Resource Definition (CRD) Installation Failure



Symptoms: Custom resources are not available in the cluster after installing CRDs.

Diagnosis: Check the CRD installation status (`kubectl get crd`) and review installation logs.

Solution:

1. Verify that CRD manifests are correctly formatted and compatible with the Kubernetes version.
2. Ensure RBAC permissions allow the creation of CRDs by the Kubernetes API server.
3. Check for conflicts or dependencies with other installed CRDs or resources.

In the up-coming parts, we will discussion on more troubleshooting steps for the different Kubernetes based scenarios. So, stay tuned for the and follow @Prasad Suman Mohan for more such posts.