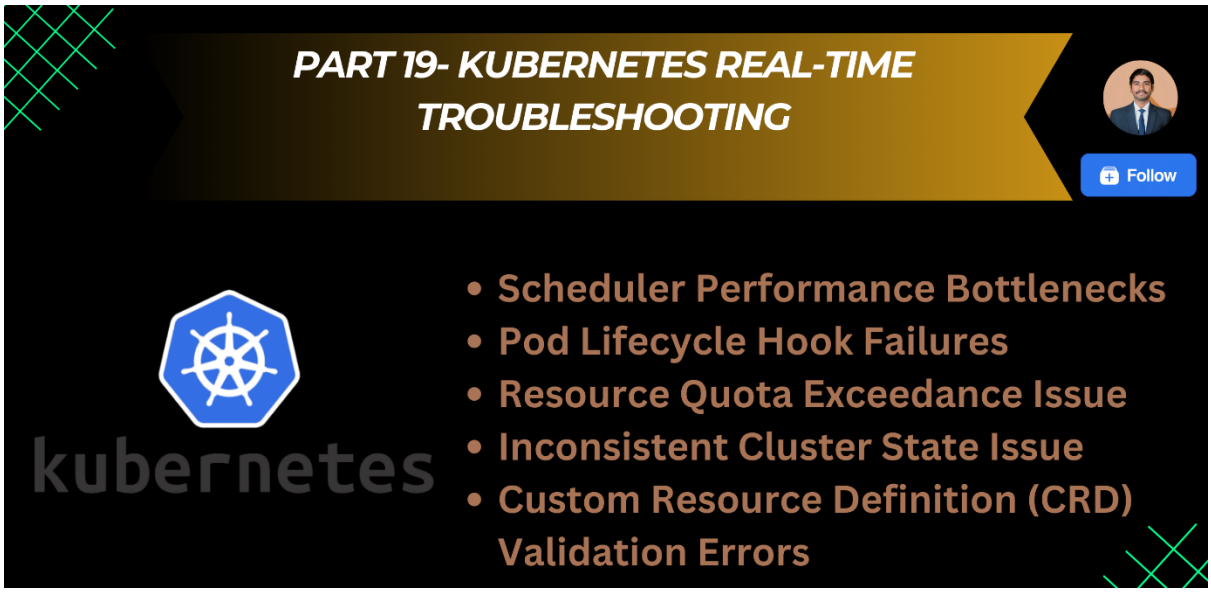
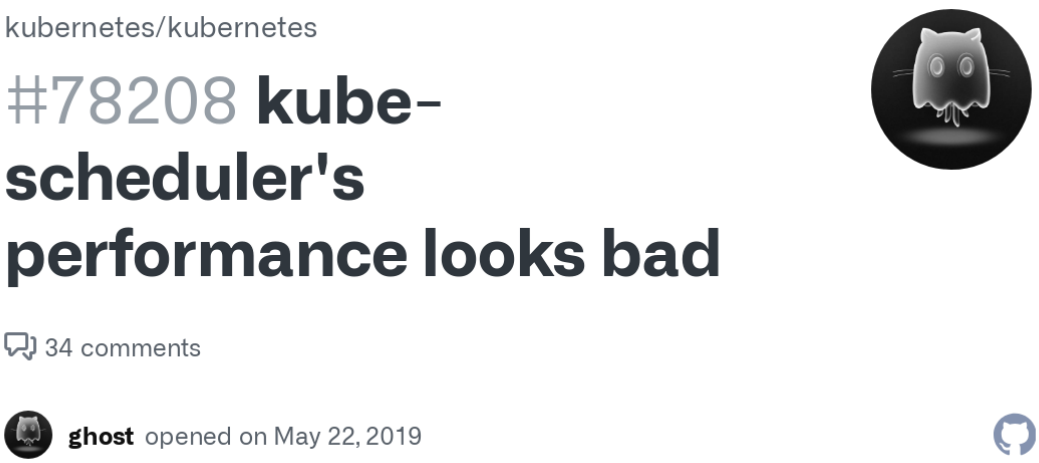
Part 19: Kubernetes Real-Time Troubleshooting

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 91: Scheduler Performance Bottlenecks Issue



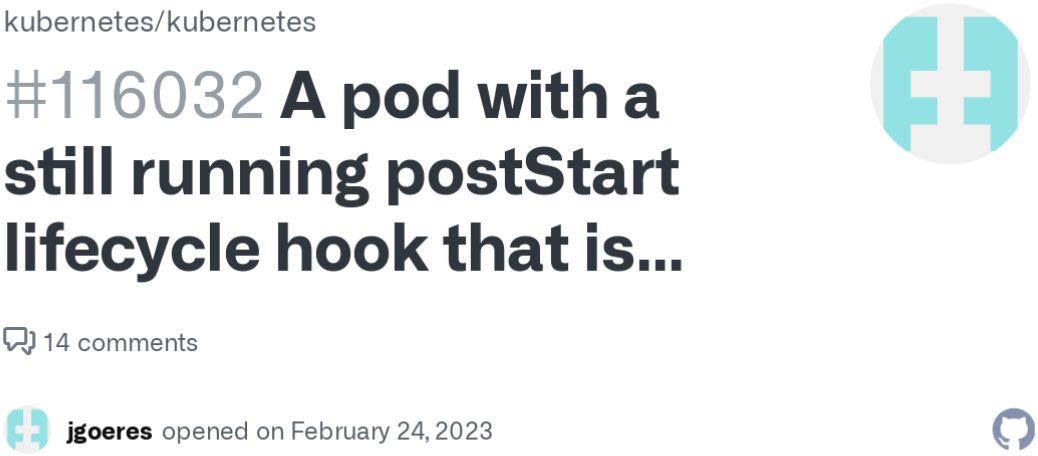
Symptoms: Pod scheduling is slow or pods remain in a pending state for extended periods.

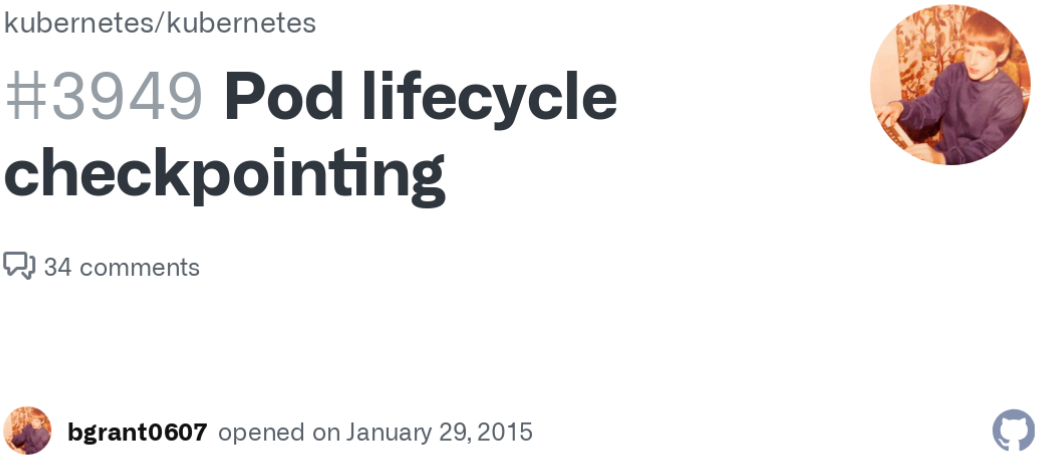
Diagnosis: Monitor scheduler metrics (`kubectl top pod -n kube-system`) and review scheduler logs.

Solution:

1. Scale up the scheduler deployment to handle increased scheduling workload.
2. Adjust scheduler configuration parameters such as parallelism and scheduling algorithm.
3. Consider using advanced scheduling techniques like NodeAffinity or NodeSelector to optimize pod placement.

Scenario 92: Pod Lifecycle Hook Failures





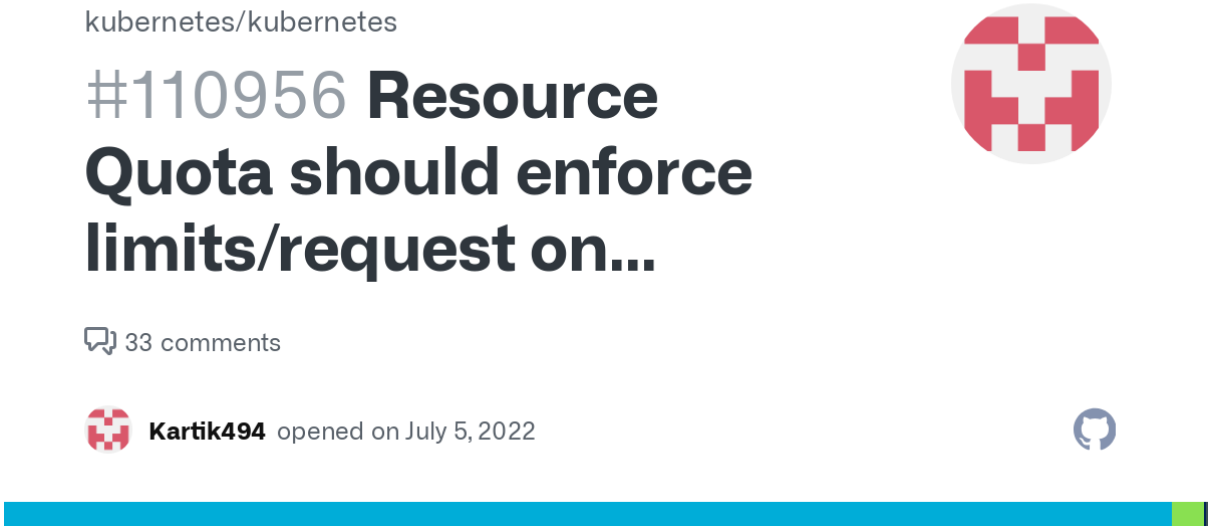
Symptoms: Pods fail to start or terminate due to errors in lifecycle hook execution.

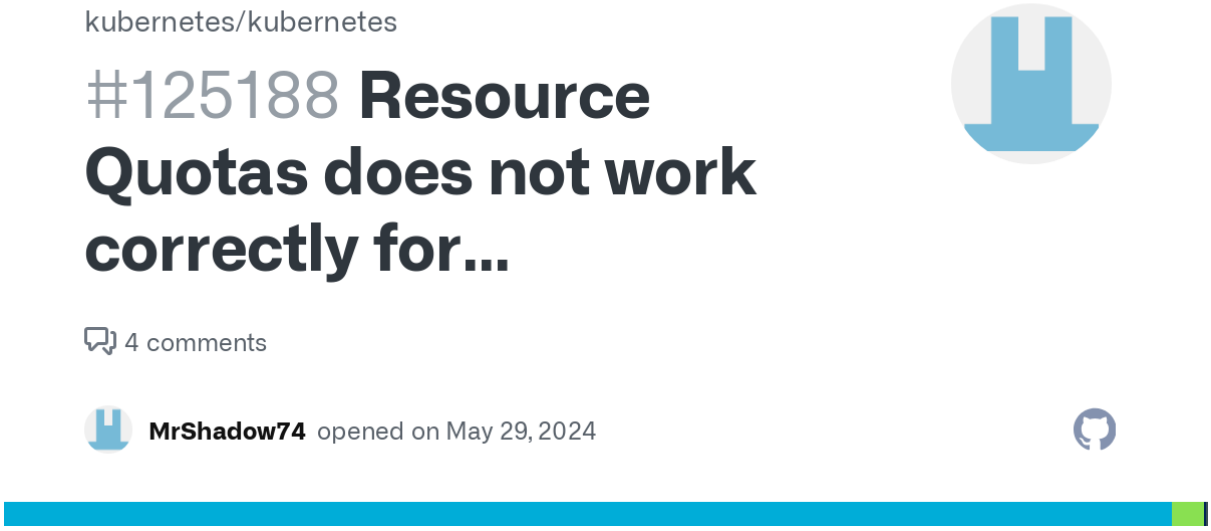
Diagnosis: Describe pod (`kubectl describe pod <pod_name>`) and review pod lifecycle hook configurations.

Solution:

1. Validate lifecycle hook scripts for syntax errors or misconfigurations.
2. Check for dependencies or external services required by lifecycle hooks that may not be accessible from the pod.
3. Use pod readiness probes to delay pod readiness until lifecycle hooks complete successfully.

Scenario 93: Resource Quota Exceedance





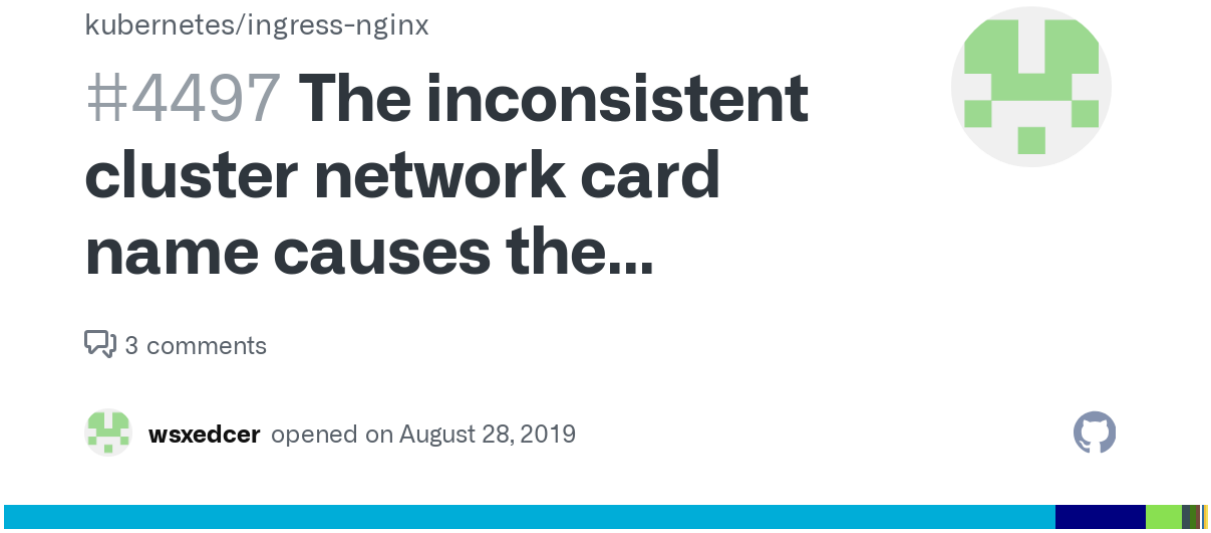
Symptoms: Pods fail to start due to resource quota limits being exceeded in namespaces.

Diagnosis: Describe resource quota (`kubectl describe quota -n <namespace>`) and review resource requests/limits in pod specifications.

Solution:

1. Increase resource quota limits for affected namespaces to allow for additional resource allocation.
2. Optimize resource requests and limits for pods to better utilize available cluster resources.
3. Implement pod priority and preemption to prioritize critical workloads during resource contention.

Scenario 94: Inconsistent Cluster State



kubernetes/kops

#8435 cluster unstable after a major version upgrade

5 comments

 **cnw004** opened on January 29, 2020

Symptoms: Cluster resources are in an inconsistent or degraded state, impacting application reliability.

Diagnosis: Review Kubernetes API server logs (`kubectl logs -n kube-system <api_server_pod_name>`) and inspect cluster state.

Solution:

1. Perform a cluster-wide audit of resources and reconcile any inconsistencies using tools like `kubectl` or Kubernetes API clients.
2. Implement automated validation checks and reconciliation workflows to maintain consistent cluster state.
3. Backup critical cluster resources and etcd data regularly to restore the cluster to a stable state in case of failures.

Scenario 95: Custom Resource Definition (CRD) Validation Errors

Symptoms: CRD resources fail to validate or are rejected by the Kubernetes API server.

Diagnosis: Describe CRD (`kubectl describe crd <crd_name>`) and review custom resource specifications.

Solution:

1. Validate CRD resource schemas against the Kubernetes API validation schema to ensure compatibility.
2. Use admission controllers or custom webhooks to enforce additional validation rules for CRD resources.
3. Monitor CRD usage and versioning to detect and address compatibility issues with existing resources.

Issues available on Github for Custom Resource Definition (CRD) Validation Errors:

kubernetes/kubernetes

#88252 CRD validation error for x-kubernetes-preserve-unknown-...

24 comments

tamalsaha opened on February 17, 2020

kubernetes/kubernetes

#71138 Changing CRD validation causes existing WATCH...

8 comments

drigz opened on November 16, 2018

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.