



Part 4 - 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.



The slide features a blue header bar with the text "KUBERNETES REAL-TIME TROUBLESHOOTING". To the right is a circular profile picture of a man and a "Follow" button. On the left is a large blue hexagonal icon containing a white steering wheel. Below it, the word "kubernetes" is written in a large, light gray sans-serif font. To the right of the icon is a bulleted list of troubleshooting topics:

- Service Discovery and Endpoint Resolution Issues
- Pod Resource Starvation
- Container Image Pull Failures
- Node Disk Pressure and Eviction
- Network Policy Enforcement Failures
- Custom Resource Definition (CRD) Version Compatibility

Scenario 16: Pod Resource Starvation

```
Status:      Running
IP:         192.168.12.38
IPs:
IP:         192.168.12.38
Controlled By: ReplicaSet/atomicred-5cbfbdd9df
Containers:
atomicred:
  Container ID:  containerd://035b87381e2a06ff03372ae44b7502fcda85cab37b91080612e7a009ffb2646f6
  Image:        issif/atomic-red:latest
  Image ID:    docker.io/issif/atomic-red@sha256:ad7a8166114f1c41dfd92f5b11fadba85605de02b86d73a5c9e3c13c5e2f1e493
  Port:        <none>
  Host Port:   <none>
  Command:
    sleep
    3560d
  State:       Running
  Started:    Fri, 19 Jan 2024 17:17:58 +0000
  Ready:      True
  Restart Count: 0
  Limits:
    cpu:    500m
    memory: 128Mi
  Requests:
    cpu:    250m
    memory: 64Mi
Environment: <none>
```

[FOLLOW – Prasad Suman Mohan \(for more updates\)](#)



Symptoms: Pods experience performance degradation or fail due to resource starvation, such as CPU or memory exhaustion.

Diagnosis: Monitor pod resource usage (`kubectl top pod`) and review resource requests and limits (`kubectl describe pod <pod_name>`) for any mismatches or overcommitments.

Solution:

1. Adjust resource requests and limits for pods based on observed usage patterns and performance requirements.
2. Implement Horizontal Pod Autoscaling (HPA) to dynamically adjust pod replicas based on resource utilization metrics.
3. Use Kubernetes resource quotas and limits to enforce resource allocation boundaries and prevent resource contention.
4. Optimize container resource usage by identifying and mitigating memory leaks, CPU-intensive processes, or inefficient application code.

Scenario 17: Container Image Pull Failures – n/w issues

NAME	READY	STATUS	RESTARTS	AGE
ambassador-7f7646f648-2q4vv	3/3	Running	0	1d
ambassador-7f7646f648-9g656	3/3	Running	0	1d
ambassador-7f7646f648-9j29n	3/3	Running	0	1d
argo-ui-74c857b6b5-5hf9	1/1	Running	0	1d
centraldashboard-fd4f5c56c-w6mhm	0/1	ImagePullBackOff	0	1d
modeldb-backend-64ff8bc7cd-449hr	0/1	ImagePullBackOff	0	1d
modeldb-db-6c9bc58dc-bxwzg	1/1	Running	0	1d
modeldb-frontend-cfd9d6df-6t2bl	0/1	ImagePullBackOff	0	1d
spartakus-volunteer-c56b8576c-9mw6l	0/1	ImagePullBackOff	0	1d
studyjob-controller-6f57f669f5-zd4k4	0/1	CrashLoopBackOff	471	1d
tf-hub-0	0/1	ImagePullBackOff	0	1d
tf-job-dashboard-7cc898fbcc-kck5z	0/1	ErrImagePull	0	1d
tf-job-operator-v1alpha2-69ccc996bc-646q4	0/1	ErrImagePull	0	1d
vizier-core-65574878f9-qwqfd	0/1	ImagePullBackOff	0	1d
vizier-db-5bbf77b76f-q6mxr	0/1	Pending	0	1d
vizier-suggestion-bayesianoptimization-5bcf5598bc-f8d84	0/1	ImagePullBackOff	0	1d
vizier-suggestion-grid-7f9df8d567-vjgvt	0/1	ImagePullBackOff	0	1d
vizier-suggestion-hyperband-769564d755-vw9gh	0/1	ImagePullBackOff	0	1d
vizier-suggestion-random-76fdfdb5b7-g6mnd	0/1	ImagePullBackOff	0	1d
workflow-controller-657666fd66-djhwg	1/1	Running	0	1d

Symptoms: Pods fail to start or remain in a pending state due to failures in pulling container images from the registry, resulting in image pull errors due to lack of network connectivity.

Diagnosis: Check pod events (`kubectl describe pod <pod_name>`) and container logs (`kubectl logs <pod_name> -c <container_name>`) for any image pull errors or connectivity issues.

Solution:

1. Verify network connectivity from the Kubernetes cluster to the container registry, ensuring firewall rules and network policies allow outbound traffic.
2. Check authentication credentials (e.g., Docker credentials, service account tokens) used to authenticate with the container registry and ensure they are valid and up-to-date.



3. Monitor container image pull latency and registry availability metrics to identify potential bottlenecks or performance issues.
4. Use local image caching or a private image registry within the Kubernetes cluster to reduce reliance on external registries and improve image pull performance and reliability.

Scenario 18: Node Disk Pressure and Eviction

Symptoms: Nodes experience disk pressure conditions, leading to pod evictions and disruptions in workload availability.

Diagnosis: Monitor node disk usage (`kubectl describe node <node_name>`) and inspect system logs (`journaldctl -u kubelet`) for disk-related errors or warnings.

Events:				
Type	Reason	Age	From	Message
Warning	FreeDiskSpaceFailed	33m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4337165926 bytes, but freed 0 bytes
Normal	NodeHasNoDiskPressure	31m (x4 over 4d)	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasNoDiskPressure
Warning	ImageGCFailed	28m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4340938342 bytes, but freed 0 bytes
Warning	FreeDiskSpaceFailed	28m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4340938342 bytes, but freed 0 bytes
Warning	ImageGCFailed	23m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4342310502 bytes, but freed 0 bytes
Warning	FreeDiskSpaceFailed	23m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4342310502 bytes, but freed 0 bytes
Warning	ImageGCFailed	18m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4351325798 bytes, but freed 0 bytes
Warning	FreeDiskSpaceFailed	18m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4351325798 bytes, but freed 0 bytes
Warning	ImageGCFailed	13m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4353996390 bytes, but freed 0 bytes
Warning	FreeDiskSpaceFailed	13m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4353996390 bytes, but freed 0 bytes
Warning	ImageGCFailed	8m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4356044390 bytes, but freed 0 bytes
Warning	ImageGCFailed	8m	kubelet, vggengapps01	Failed to garbage collect required amount of images. Wanted to free 4356044390 bytes, but freed 0 bytes
Normal	NodeHasDiskPressure	1m (x3 over 4d)	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasDiskPressure
Warning	CheckLimitsForResolvConf	1m (x159 over 4d)	kubelet, vggengapps01	Resolv.conf file '/etc/resolv.conf' contains search line consisting of more than 3 domains!
Warning	EvictionThresholdMet	1m (x12 over 4d)	kubelet, vggengapps01	Attempting to reclaim ephemeral-storage
Normal	NodeHasSufficientMemory	1m	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasSufficientMemory
Normal	NodeHasSufficientDisk	1m	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasSufficientDisk
Normal	Starting	1m	kubelet, vggengapps01	Starting kubelet.
Normal	NodeHasNoDiskPressure	1m (x2 over 1m)	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasNoDiskPressure
Normal	NodeHasSufficientID	1m	kubelet, vggengapps01	Node vggengapps01 status is now: NodeHasSufficientID
Normal	NodeNotReady	1m	kubelet, vggengapps01	Node vggengapps01 status is now: NodeNotReady
Normal	NodeAllocatableEnforced	56s	kubelet, vggengapps01	Updated Node Allocatable limit across pods
Warning	EvictionThresholdMet	56s	kubelet, vggengapps01	Attempting to reclaim ephemeral-storage
Warning	CheckLimitsForResolvConf	54s (x16 over 54s)	kubelet, vggengapps01	Resolv.conf file '/etc/resolv.conf' contains search line consisting of more than 3 domains!

Solution:

1. Identify and remove unnecessary files or logs consuming disk space on the affected nodes.
2. Configure log rotation and retention policies for container logs to prevent excessive disk usage and avoid node eviction.
3. Use node maintenance scripts or automation tools to periodically clean up temporary files and reclaim disk space.
4. Consider resizing persistent volumes or adding additional storage capacity to nodes experiencing persistent disk pressure issues.

Scenario 19: Network Policy Enforcement Failures

[root@kube kubectlTest]# kubectl get all -n ingress-nginx						
NAME	READY	STATUS	RESTARTS	AGE		
pod/ingress-nginx-admission-create-jr6q4	0/1	Completed	0	87s		
pod/ingress-nginx-admission-patch-wjnnj	0/1	Completed	1	87s		
pod/ingress-nginx-controller-8f7b9d799-w7dpt	1/1	Running	0	97s		

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/ingress-nginx-controller	LoadBalancer	10.98.18.122	<pending>	80:30529/TCP,443:31217/TCP	97s
service/ingress-nginx-controller-admission	ClusterIP	10.96.156.86	<none>	443/TCP	97s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/ingress-nginx-controller	1/1	1	1	97s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/ingress-nginx-controller-8f7b9d799	1	1	1	97s

NAME	COMPLETIONS	DURATION	AGE
job.batch/ingress-nginx-admission-create	1/1	2s	97s
job.batch/ingress-nginx-admission-patch	1/1	3s	97s

 [root@kube kubectlTest]# |

FOLLOW – Prasad Suman Mohan (for more updates)



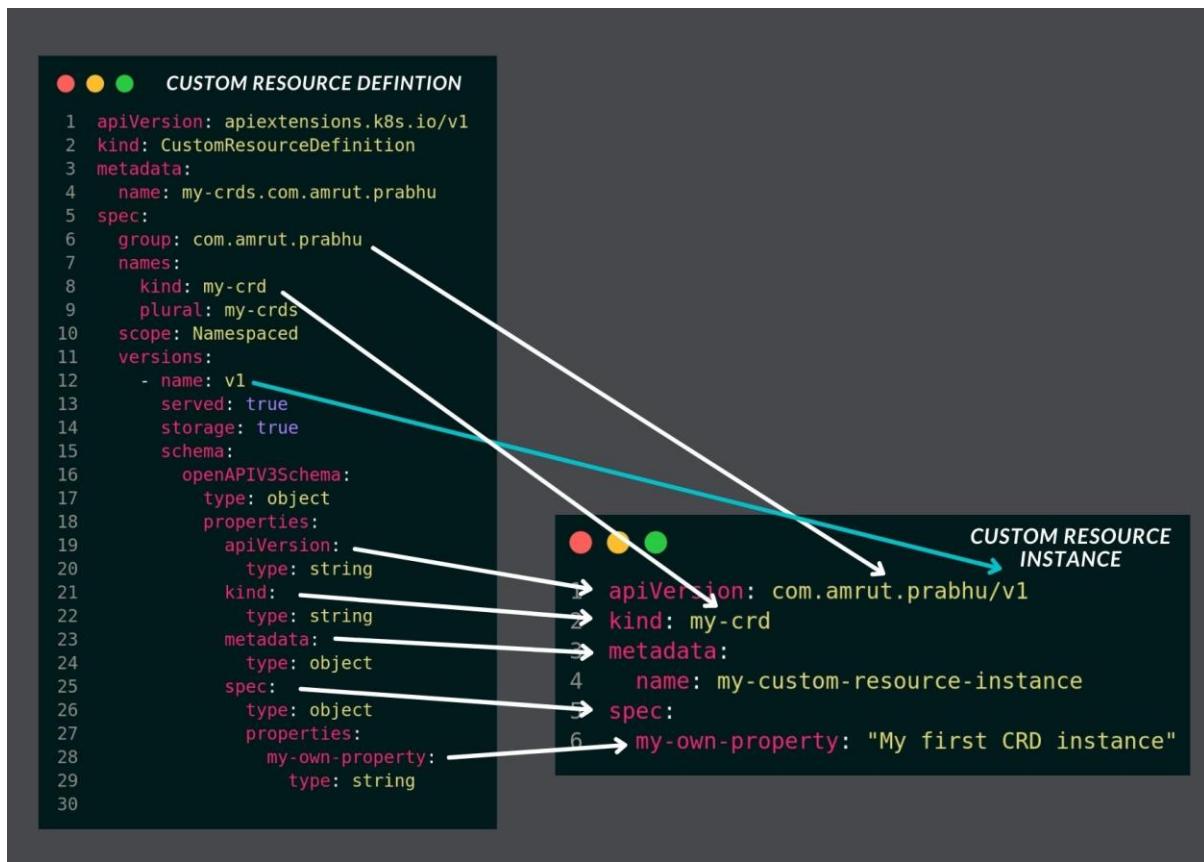
Symptoms: Network policies fail to enforce ingress or egress rules between pods, leading to unauthorized network access or security breaches.

Diagnosis: Review network policy configurations (`kubectl get networkpolicy`) and network plugin logs for any errors or policy enforcement failures.

Solution:

1. Verify that the network plugin (e.g., Calico, Flannel, Cilium) is properly configured and supports network policy enforcement in the Kubernetes cluster.
2. Use network policy visualization tools or network packet capture utilities to inspect network traffic and validate policy enforcement.
3. Implement network policy testing and validation procedures in staging or development environments to ensure policies behave as expected.
4. Update network policies to reflect changes in application requirements or security policies and apply them consistently across all namespaces and clusters.

Scenario 20: Custom Resource Definition (CRD) Version Compatibility



Symptoms: Operators or controllers fail to reconcile custom resources (CRs) due to version incompatibility or schema validation errors.

Diagnosis: Review operator logs (`kubectl logs <operator_pod>`) and CRD definitions (`kubectl get crd`) for any errors or mismatches between CR versions and API schemas.

FOLLOW – Prasad Suman Mohan (for more updates)



Solution:

1. Ensure that operators and controllers are compatible with the CRD versions used in the Kubernetes cluster, and update them as needed.
2. Validate CR specifications against the corresponding CRD schemas to ensure compliance with API requirements and avoid validation errors.
3. Implement CRD versioning and API evolution strategies to smoothly transition between different versions and avoid breaking changes in CR definitions.
4. Monitor operator health and reconciliation status to detect and address any issues with CR processing or lifecycle management.

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



VectorStock®

VectorStock.com/14301556

FOLLOW – Prasad Suman Mohan (for more updates)