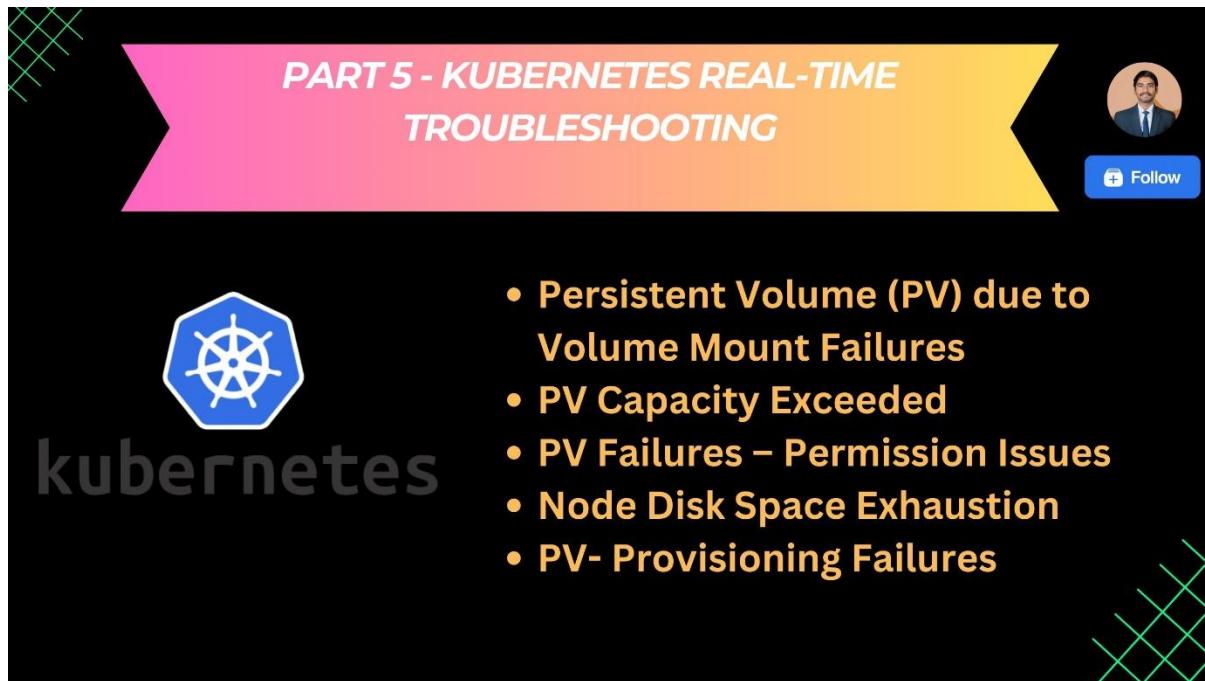




Part 5 - Kubernetes Real-Time Troubleshooting

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. Below, we address cases related to Kubernetes storage failures:



The banner features a pink-to-yellow gradient background with the text "PART 5 - KUBERNETES REAL-TIME TROUBLESHOOTING". To the right is a circular profile picture of a man and a "Follow" button with a GitHub icon.

kubernetes 

- Persistent Volume (PV) due to Volume Mount Failures
- PV Capacity Exceeded
- PV Failures – Permission Issues
- Node Disk Space Exhaustion
- PV- Provisioning Failures

Scenario 21: Persistent Volume (PV) due to Volume Mount Failures

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc ← Name of PVC
spec:
  storageClassName: nfs ← Name of Storage Class Name , use
                        same class name defined in pv
  accessModes:
    - ReadWriteMany ← Specify the same access mode
                      which is defined in pv
  resources:
    requests:
      storage: 10Gi ← Amount of Storage request
```

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



Symptoms: Pods fail to start or restart due to persistent volume mount failures, resulting in volume-related errors.

Diagnosis: Check pod events (`kubectl describe pod <pod_name>`) and PV status (`kubectl get pv`) for any mount errors or volume provisioning issues.

Solution:

1. Verify that the PV is bound to the correct PersistentVolumeClaim (PVC) and that the PVC is in the Bound state.
2. Check storage provider logs or cloud storage services for any errors or connectivity issues impacting PV provisioning.
3. Ensure that the correct storage class is specified in the PVC and that it matches the storage backend configuration.
4. Troubleshoot node connectivity issues or network policies that may prevent pods from accessing PVs.

Scenario 22: Persistent Volume (PV) Capacity Exceeded

```
radamuz@radamuz:~$ kubectl describe pvc pvcsc-vsanc
Name:          pvcsc-vsanc
Namespace:    default
StorageClass: fast
Status:       Pending
Volume:
Labels:      <none>
Annotations: volume.beta.kubernetes.io/storage-class: fast
              volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/vsphere-volume
Finalizers:   [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:  Filesystem
Used By:    <none>
Events:
  Type     Reason            Age           From                  Message
  ----     ----            ----         ----                  -----
  Warning  ProvisioningFailed 3s (x6 over 74s)  persistentvolume-controller  Failed to provision
           volume with StorageClass "fast": cloud provider not initialized properly
```

Symptoms: Persistent volumes reach full capacity, causing pod failures or data loss due to inability to write data to disk.

Diagnosis: Check PV and PVC status (`kubectl get pv, pvc`) and examine storage provider metrics for disk usage and capacity utilization.

Solution:

1. Increase the size of existing PVs or provision additional PVs with larger storage capacity to accommodate growing data volumes.
2. Implement data retention policies and automatic data pruning mechanisms to remove obsolete or unused data and free up storage space.
3. Monitor disk usage trends and forecast storage requirements to proactively scale storage resources and prevent capacity bottlenecks.
4. Configure alerts and notifications to notify administrators when PV capacity thresholds are nearing or exceeded to take timely action.

FOLLOW – Prasad Suman Mohan (for more updates)



Scenario 23: Persistent Volume Mount Failures – Permission Issues

```
invisible-squirrel-mariadb      Bound   local-pv-87bc97ab  9647604Ki  RWO   1
al-storage 17h
nonplussed-swan-mariadb       Bound   local-pv-6923a82c  9647604Ki  RWO   1
al-storage 17h
ornery-lemur-joomla-apache    Bound   local-pv-9bfd6008  9647604Ki  RWO   1
al-storage 17h
ornery-lemur-joomla-joomla   Bound   local-pv-125a2537  9647604Ki  RWO   1
al-storage 17h
ornery-lemur-mariadb         Bound   local-pv-20cd15aa  9647604Ki  RWO   1
al-storage 17h
bitnami@bitnami-kubernetes-sandbox-dm-35c0:~$ kubectl describe pvc incindiciary-opossum-wordpress
Name:           incindiciary-opossum-wordpress
Namespace:      default
StorageClass:   local-storage
Status:         Pending
Volume:
Labels:         app=incindiciary-opossum-wordpress
                chart=wordpress-0.7.7
                heritage=Tiller
                release=incindiciary-opossum
Annotations:    volume.beta.kubernetes.io/storage-provisioner=local-storage
Capacity:
Access Modes:
Events:
  Type     Reason          Age           From            Message
  ----     ----          ----          ----           -----
  Normal   ExternalProvisioning 16h (x322 over 17h) persistentvolume-controller waiting for a volume to be created, either by external provisioner "local-storage" or manually created by system administrator
  Normal   ExternalProvisioning  3m (x162 over 43m)  persistentvolume-controller waiting for a volume to be created, either by external provisioner "local-storage" or manually created by system administrator
```

Symptoms: Pods fail to mount persistent volumes (PVs) due to storage provisioning errors, permissions issues, or volume attachment failures.

Diagnosis: Check pod events (`kubectl describe pod <pod_name>`) and PV status (`kubectl get pv`) for any errors or warnings related to volume provisioning, attachment, or access.

Solution:

1. Ensure that PVs are correctly provisioned and dynamically provisioned storage classes are available and accessible to the Kubernetes cluster.
2. Verify storage provider credentials and permissions to ensure that pods have sufficient privileges to mount and access persistent volumes.
3. Check node disk and filesystem integrity to identify any disk failures, corruption, or capacity constraints that may affect volume attachment and mounting operations.
4. Implement volume health checks and monitoring to detect and remediate storage issues proactively, such as disk failures, I/O errors, or storage latency spikes.

FOLLOW – Prasad Suman Mohan (for more updates)



Scenario 24: Node Disk Space Exhaustion

```
INFO:root:RawImageIterAsync: loading image list...
Traceback (most recent call last):
  File "test.py", line 142, in <module>
    val_path, args.batch_size)
  File "test.py", line 59, in get_data
    val_img_list=val_list)
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/data_factory.py", line 134, in get_data_iter
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 405, in get_data_iter
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 184, in __init__
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 184, in <listcomp>
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/context.py", line 129, in RawArray
    return RawArray(typecode_or_type, size_or_initializer)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/sharedctypes.py", line 60, in RawArray
    obj._new_(value_type)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/sharedctypes.py", line 40, in _new_value
    wrapper = heap.BufferWrapper(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heaps.py", line 248, in __init__
    block = BufferWrapper._heap_malloc(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heaps.py", line 230, in malloc
    (arena, start, stop) = self._malloc(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heaps.py", line 128, in _malloc
    arena = Arena(length)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heaps.py", line 77, in __init__
    f.writezeros()
OSError: [Errno 28] No space left on device
Exception ignored in: <bound method RawImageIterAsync.__del__ of <moxing.mxnet.data.imageraw_dataset_async.RawImageIterAsync object at 0x7fa18588f9b0>>
Traceback (most recent call last):
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 222, in __del__
```

Symptoms: Nodes experience disk space exhaustion, causing critical system components (e.g., kubelet, container runtime) to fail and impacting pod scheduling and execution.

Diagnosis: Monitor node disk usage (`df -h`) and inspect system logs (`/var/log/messages`, `journalctl`) for disk-related errors or warnings indicating low disk space conditions.

Solution:

1. Identify and remove unnecessary files or temporary data consuming disk space, such as old log files, unused containers/images, or orphaned volumes.
 2. Implement disk space monitoring and alerting to detect and mitigate disk space issues proactively before they lead to service disruptions.
 3. Scale out storage capacity by attaching additional disks or expanding existing volumes to accommodate growing storage requirements and prevent future disk space shortages.
 4. Configure log rotation and retention policies to limit the size of log files and prevent them from consuming excessive disk space over time.

Scenario 25: Persistent Volume (PV) Provisioning Failures

```
[root@ip-10-40-78-3 ~]# kubectl describe pvc archive-yt-test1-0
Name:           archive-yt-test1-0
Namespace:      default
StorageClass:   csi-qcf
Status:         Bound
Volume:         pvc-5bc115039edfile8
Labels:         AppName=yt-test
                CreatedByUser=true
                DnsName=yt-test
                DBType=mysql
                Name=yt-test
Annotations:    pv.kubernetes.io/bind-completed=yes
                pv.kubernetes.io/bound-by-controller=yes
                volume.beta.kubernetes.io/storage-class=external
                volume.beta.kubernetes.io/storage-class=csi-qcf
                volume.beta.kubernetes.io/storage-provisioner=csi-qcfplugin
[Finalizers]:   [kubernetes.io/pvc-protection]
Capacity:       -
Access Modes:   RWO
Events:         
```

Type	Reason	Age	From	Message
Normal	ExternalProvisioning	1m (x 2 over 1m)	persistentvolume-controller	waiting for a volume to be created, either by external provisioner "csi-qcfplugin" or manually created by system administrator
Normal	ProvisioningSucceeded	1m (x 2 over 1m)	csi-qcfplugin external-provisioner-# 4#005701-9e05-11eb-aec2-ba509fe400c	External provisioner is provisioning volume for claim "default/archive-yt-test1-0"
Warning	ProvisioningFailed	1m	csi-qcfplugin external-provisioner-# 4#005701-9e05-11eb-aec2-ba509fe400c	Successfully provisioned volume pvc-5bc115039edfile8
				Error creating provisioned pv object for claim default/archive-yt-test1-0: persistentvolumes "pvc-5bc115039edfile8" already exists. Deleting

Symptoms: Pods fail to mount persistent volumes (PVs) due to provisioning failures or storage backend issues, resulting in data loss or application downtime.

Diagnosis: Check PV and PersistentVolumeClaim (PVC) statuses (`kubectl get pv,pvc`) and inspect storage provider logs (e.g., CSI driver logs, storage backend logs) for any provisioning errors or resource constraints.

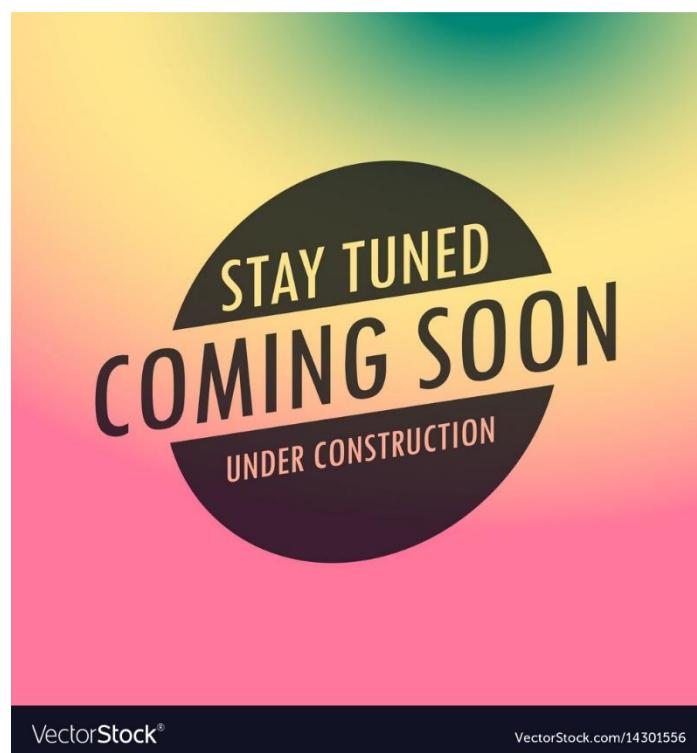
FOLLOW – Prasad Suman Mohan (for more updates)



Solution:

1. Troubleshoot storage backend connectivity issues, such as network connectivity, authentication, or authorization problems between Kubernetes cluster nodes and storage systems.
2. Verify storage class configurations (`kubectl get storageclass`) and ensure that the appropriate storage classes are available and accessible for dynamic provisioning of PVs based on PVC requests.
3. Monitor storage resource utilization and capacity metrics to identify any resource shortages or bottlenecks that may be impacting PV provisioning and allocation.
4. Implement storage resiliency features, such as data replication, redundancy, or snapshotting, to protect against data loss and ensure high availability of persistent volumes in case of storage failures.

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.



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