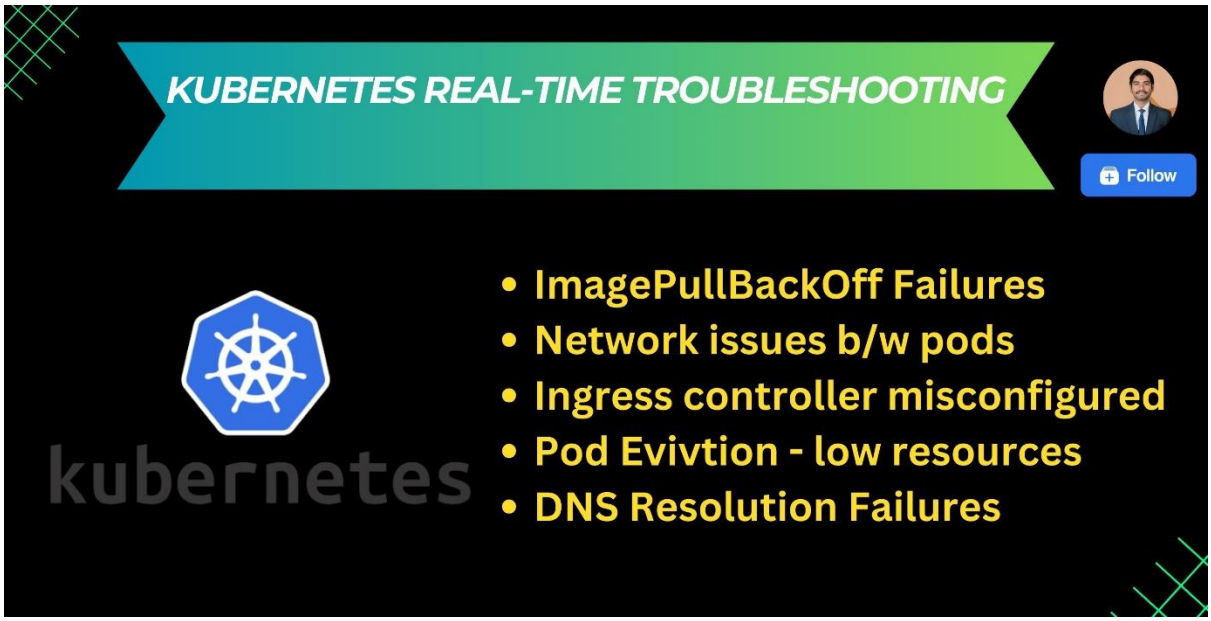
Part 2 - Kubernetes Real-Time Troubleshooting

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 6: ImagePullBackOff Failures

```
[root@chaos-testing-1 ec2-user]# vi imagepullbackoff.yaml
[root@chaos-testing-1 ec2-user]# kubectl apply -f imagepullbackoff.yaml
pod/imagepull-nginx created
[root@chaos-testing-1 ec2-user]# kubectl get pods
NAME                                READY   STATUS             RESTARTS       AGE
busybox-588798b547-cdrp7            0/1     CrashLoopBackOff   18 (5s ago)    10h
crashloop-example-7646cbb8cf-qqhhc  1/1     Running            208 (9h ago)   28h
crashloop-example-new-7646cbb8cf-ngrrm  1/1     Running        16 (9h ago)    10h
imagepull-nginx                     0/1     ImagePullBackOff   0              6s
nginx-deployment-7c79c4bf97-84c7k   1/1     Running            2 (69s ago)    28h
nginx-deployment-7c79c4bf97-dntdp   1/1     Running            2 (69s ago)    10h
nginx-deployment-7c79c4bf97-fd5vz   1/1     Running            2 (69s ago)    28h
nginx-deployment-7c79c4bf97-jz64b   1/1     Running            2 (69s ago)    10h
nginx-deployment-7c79c4bf97-p7tpg   1/1     Running            2 (69s ago)    28h
redis-7c888f4788-gggcp              1/1     Running            2 (69s ago)    9h
[root@chaos-testing-1 ec2-user]#
```

FOLLOW – Prasad Suman Mohan (for more updates)

Symptoms: Pods fail to start due to errors while pulling container images from the registry.

Diagnosis: Check the pod events (`kubectl describe pod <pod_name>`) for image pull errors. Verify network connectivity to the container registry and authentication credentials if applicable.

Solution:

1. Ensure the container image repository URL is correct and accessible from the Kubernetes cluster.
2. Check for any network proxy configurations that might be blocking outbound traffic to the registry.
3. Verify the image pull secret (if used) is correctly configured and has the necessary permissions to pull images.
4. Monitor registry logs for any issues on the repository side, such as rate limiting or downtime.

Scenario 7: Network Connectivity Issues Between Pods

```
➜  ~ kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
apitier-555db4974d-sl7b5    1/1       Running   0          17m
webtier-6dbbc5fff6-k8748    1/1       Running   0          30m
➜  ~ kubectl port-forward webtier-6dbbc5fff6-k8748 8800 &
➜  ~ Forwarding from 127.0.0.1:8800 → 8800
Forwarding from [::1]:8800 → 8800
➜  ~ curl localhost:8800
Handling connection for 8800
Hello, from the API tier!
```

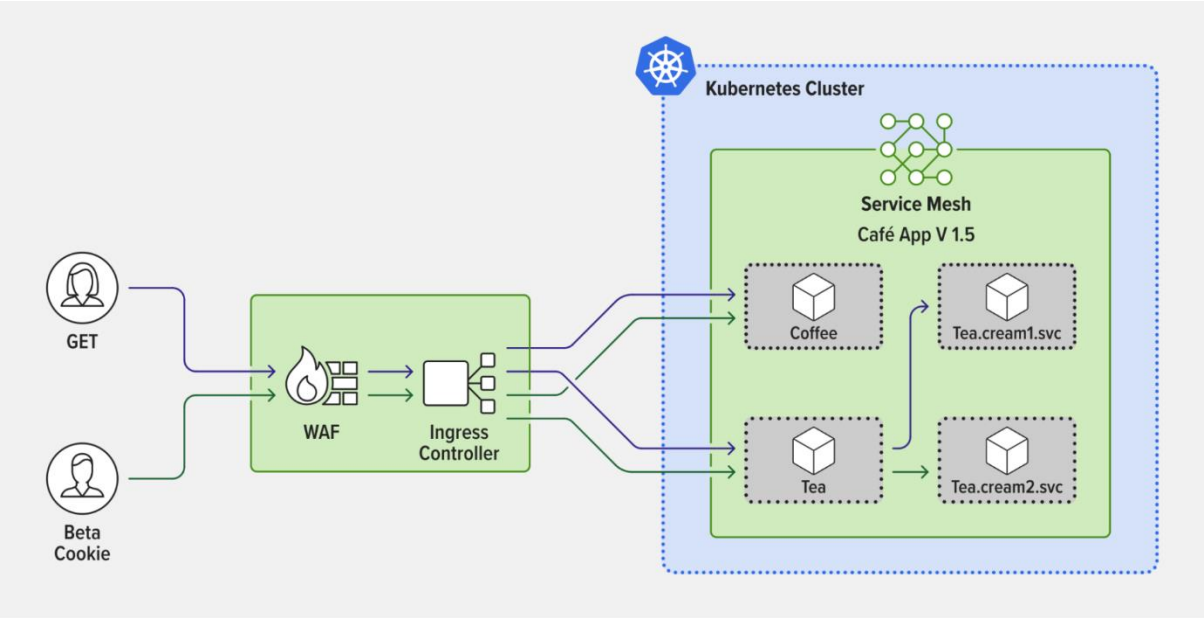
Symptoms: Pods are unable to communicate with each other over the network, leading to service disruptions or errors.

Diagnosis: Use `kubectl exec` to troubleshoot network connectivity from within the affected pods. Check for any network policies or firewall rules that might be blocking inter-pod communication.

Solution:

1. Review Kubernetes network policies to ensure they allow traffic between the affected pods.
2. Use tools like `traceroute` or `tcpdump` within the pods to diagnose network routing or connectivity issues.
3. Check for any misconfigured pod IP addresses or DNS resolution issues.
4. Verify that the underlying network infrastructure (e.g., overlay network, SDN) is functioning correctly and has adequate capacity.

Scenario 8: Ingress Controller Misconfiguration



Symptoms: External traffic fails to reach Kubernetes services configured with an ingress controller.

Diagnosis: Inspect the ingress controller logs (`kubectl logs <ingress_controller_pod>`) for any errors or warnings related to route configuration or request handling.

Solution:

1. Verify the ingress resource definition (`kubectl describe ingress <ingress_name>`) for correct backend service and routing rules.
2. Check for any misconfigured TLS certificates or termination settings if using HTTPS.
3. Ensure the ingress controller service is correctly exposed and accessible from external networks.
4. Monitor ingress controller metrics for any indications of high request latency or errors.

Scenario 9: Pod Eviction Due to Resource Pressure

Symptoms: Pods are being evicted from nodes due to resource pressure, leading to intermittent service disruptions.

Diagnosis: Review node metrics (`kubectl top nodes`) to identify nodes experiencing high resource utilization. Check pod eviction events (`kubectl get events --field-selector involvedObject.kind=Pod --sort-by=.metadata.creationTimestamp`) for reasons related to resource constraints.

```
dbafromthecold@kubernetes-m1:~$ kubectl get nodes --watch
NAME            STATUS    ROLES    AGE   VERSION
kubernetes-m1   Ready     master   9h    v1.18.0
kubernetes-w1   Ready     <none>   9h    v1.18.0
kubernetes-w2   Ready     <none>   9h    v1.18.0
kubernetes-w1   NotReady   <none>   9h    v1.18.0
kubernetes-w1   NotReady   <none>   9h    v1.18.0
kubernetes-w1   NotReady   <none>   9h    v1.18.0

dbafromthecold@kubernetes-m1:~$ kubectl get pods -o=custom-columns=NAME:metadata.name,STATUS:
.status.phase,NODE:spec.nodeName --watch
NAME                                          STATUS    NODE
sqlserver2019-deployment-7999749697-zj9zl   Running   kubernetes-w1
sqlserver2019-deployment-7999749697-zj9zl   Running   kubernetes-w1
```

Node reported as down

Solution:

1. Scale up the affected nodes or add more nodes to the cluster to distribute workload.
2. Optimize resource requests and limits for pods to prevent excessive resource consumption.
3. Implement resource quotas to prevent individual pods or namespaces from monopolizing cluster resources.
4. Monitor and tune the Kubernetes scheduler to prioritize critical workloads and optimize resource allocation.

Scenario 10: DNS Resolution Failures

```
I0929 13:46:52.825288       5 server.go:113] FLAG: --federations=""
I0929 13:46:52.825294       5 server.go:113] FLAG: --healthz-port="8081"
I0929 13:46:52.825299       5 server.go:113] FLAG: --initial-sync-timeout="1m0s"
I0929 13:46:52.825305       5 server.go:113] FLAG: --kube-master-url=""
I0929 13:46:52.825311       5 server.go:113] FLAG: --kubecfg-file=""
I0929 13:46:52.825315       5 server.go:113] FLAG: --log-backtrace-at=":0"
I0929 13:46:52.825322       5 server.go:113] FLAG: --log-dir=""
I0929 13:46:52.825327       5 server.go:113] FLAG: --log-flush-frequency="5s"
I0929 13:46:52.825332       5 server.go:113] FLAG: --logtostderr="true"
I0929 13:46:52.825337       5 server.go:113] FLAG: --nameservers=""
I0929 13:46:52.825342       5 server.go:113] FLAG: --stderrthreshold="2"
I0929 13:46:52.825347       5 server.go:113] FLAG: --v="2"
I0929 13:46:52.825352       5 server.go:113] FLAG: --version="false"
I0929 13:46:52.825359       5 server.go:113] FLAG: --vmodule=""
I0929 13:46:52.825516       5 server.go:176] Starting SkyDNS server (0.0.0.0:10053)
I0929 13:46:52.825731       5 server.go:198] Skydns metrics enabled (/metrics:10055)
I0929 13:46:52.825741       5 dns.go:147] Starting endpointsController
I0929 13:46:52.825747       5 dns.go:150] Starting serviceController
I0929 13:46:52.825858       5 logs.go:41] skydns: ready for queries on cluster.local. for tcp://0.0
.0.0:10053 [rcache 0]
I0929 13:46:52.825867       5 logs.go:41] skydns: ready for queries on cluster.local. for udp://0.0
.0.0:10053 [rcache 0]
I0929 13:46:53.326078       5 dns.go:171] Initialized services and endpoints from apiserver
I0929 13:46:53.326106       5 server.go:129] Setting up Healthz Handler (/readiness)
I0929 13:46:53.326116       5 server.go:134] Setting up cache handler (/cache)
I0929 13:46:53.326123       5 server.go:120] Status HTTP port 8081
I0929 19:52:39.119744       5 logs.go:41] skydns: error from backend: Invalid IP Address mypod
[me@kubernetes k8s]$
```

Symptoms: Pods are unable to resolve DNS names or domain names, leading to service discovery issues.

Diagnosis: Use `kubectl exec` to troubleshoot DNS resolution from within the affected pods. Check for any misconfigured DNS settings or DNS service failures.

Solution:

1. Verify that the CoreDNS or kube-dns service is running correctly and has access to DNS servers.
2. Check pod DNS configuration (`/etc/resolv.conf` or `/etc/hosts`) for any misconfigurations or overrides.
3. Ensure that DNS service discovery is enabled and functioning correctly for all pods and services.
4. Monitor DNS-related logs and metrics for any indications of errors or failures.

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.