Kubernetes debug techniques

Vang / 2025

whoami

Viorel Anghel

Head of Cloud and Infrastructure

@ eSolutions.ro

whatis

- Kubernetes Debug Techniques
- a beginner to intermediate level workshop

Kubernetes debug techniques - TOC

1. Basic kubectl get / describe commands
2. Kubectl logs and exec
3. Distroless containers and kubectl debug
4. Helm tips
5. Common errors
6. All-in with *cert-manager*

How To workshop

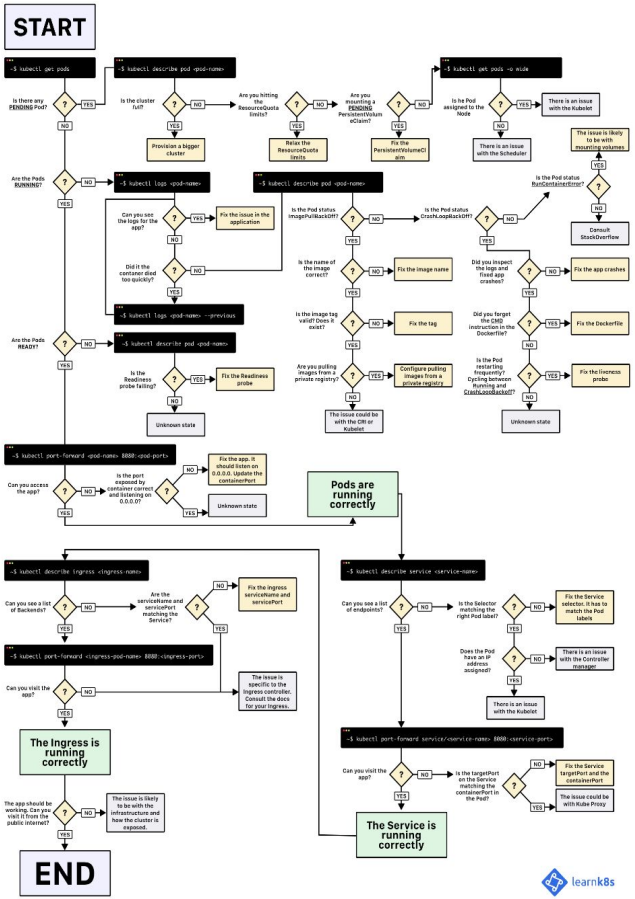
- Kubectl
- Rancher Desktop
- Feel free to interrupt me at any time

Github repo for this workshop

https://github.com/viorel-anghel/kubernetes-debug-techniques-2025

Kubernetes debug diagram

- From Learnk8s.io
- To debug a web application
- Logical step by step diagram



Only a few commands

```
kubectl [-n namespace] get <OBJECT-TYPE> [-o wide]
kubectl [-n namespace] describe <OBJECT-TYPE> <OBJECT_NAME>


kubectl -n namespace] logs [-f] <POD-NAME>
```

Details

```
$ kubectl run nginx --image=nginx
$ kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          2s
```

Kubectl describe vs get

```
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  8m19s  default-scheduler  Successfully assigned
default/nginx to lima-rancher-desktop
  Normal  Pulling    8m18s  kubelet            Pulling image "nginx"
  Normal  Pulled     7m51s  kubelet            Successfully pulled image
"nginx" in 27.635464013s
  Normal  Created    7m51s  kubelet            Created container nginx
  Normal  Started    7m50s  kubelet            Started container nginx
```

“Controlled By”

```
  $ kubectl describe pod nginx | grep Controlled
   $ # nada

   $ kubectl create deploy nginx --image=nginx
   $ kubectl describe pod nginx-8f458dc5b-gbzkc
[...]
Controlled By:  ReplicaSet/nginx-8f458dc5b
   $ kubectl describe replicaset nginx-8f458dc5b
[...]
Controlled By:  Deployment/nginx
```

Controlled By … CNPG

install CNPG and create a simple PostgreSQL cluster. then:

```
$ kubectl -n app1 describe pod simple-pg-1 | grep Controlled
Controlled By:  Cluster/app1-pg

$ kubectl -n app1 get cluster
[...]

$ kubectl -n app1 describe cluster
```

Kubectl get all

```
$ kubectl -n <NS> get all
```

```
$ kubectl api-resources
NAME                SHORTNAMES   APIVERSION    NAMESPACED   KIND
bindings                         v1            true         Binding
componentstatuses   cs           v1            false        ComponentStatus
configmaps          cm           v1            true         ConfigMap
endpoints           ep           v1            true         Endpoints
[... output truncated . . .]
```

Kubectl get all (2)

```
$ kubectl get crds
```

```
$ kubectl api-resources --namespaced --verbs=list  -o name
```

```
$ kubectl api-resources --namespaced --verbs=list  -o name | \
  xargs -n 1 kubectl get --show-kind --ignore-not-found -n<ns>
```

Kubectl logs

```
 kubectl -n <NAMESPACE> logs [-f] <POD-NAME>

# if the  pod has multiple containers to select a particular one:
  kubectl -n <NAMESPACE> logs [-f] <POD-NAME> -c <CONTAINER-NAME>

# logs on a deployment
  kubectl -n <NAMESPACE> logs [-f] deploy/<DEPLOYMENT-NAME>
```

Kubectl exec

```
$ kubectl exec -ti nginx -- bash # on my laptop
    root@nginx:/# pwd            # inside the container
    /
    [... more commands inside the container …]
    root@nginx:/# exit
$                                # back to my laptop

$ kubectl exec nginx -- hostname # run command inside container
nginx                            # command result
$                                # back to my laptop

$ kubectl cp <POD>:<FILE> <LOCAL-FILE> # SRC DST
```

PAUSE

Distroless containers

```
$ cd distroless
$ docker buildx build --platform linux/amd64 -t go-http-server:0.1 .


$ kubectl create deploy go-http-server --image=go-http-server:0.1

$ kubectl exec -ti go-http-server-7bb984744d-d7dhx  -- /bin/sh

OCI runtime exec failed: exec failed: unable to start container process:
exec: "/bin/sh": stat /bin/sh: no such file or directory: unknown
command terminated with exit code 126
```

Kubectl debug - ephemeral containers

```
$ kubectl debug -it -c debug --image=busybox go-http-server-588774479b-dvkft
```

Shared PID namespace

1. Kubectl edit deployment at the level `spec.template.spec` to add **`shareProcessNamespace: true`**

```
$ kubectl patch deploy go-http-server -p '{"spec": {"template":
{"spec": {"shareProcessNamespace": true }}}}'
```

2. Run the debug command

```
kubectl debug -it -c debug --image=busybox
go-http-server-657964647c-lfxxc
```

3. `ps -ef ; ls  /proc/<PID>/root`

Kubectl debug … copy-to

```
$ kubectl debug -it -c debug --image=busybox
--share-processes --copy-to  debug-pod
go-http-server-7bb984744d-d7dhx
```

Kubectl debug node

```
$   kubectl debug node/<NODE-NAME>  -it --image=ubuntu
```

- Node filesystem is mounted at **/host**
- Shared PID namespace (ps -ef show node processes)

Dummy - my debugging image

```
$ kubectl run dummy --image vvang/dummy:amd64
```

https://github.com/viorel-anghel/dummy-pod

PAUSE

Helm tips

- helm show values - before install

```
$ helm upgrade --install --create-namespace -n helm-pg -f values.yaml
mypg oci://registry-1.docker.io/bitnamicharts/postgresql
```

- helm get values - after install

Helm with ArgoCD

- ArgoCD is using *helm template* instead of *helm install*

PAUSE

Common errors

```
kubectl -n <NAMESPACE> get events --sort-by='.lastTimestamp'

kubectl -n <NAMESPACE> events

# -A / –all-namespaces
```

CrashLoopBackoff

```
kubectl -n <namespace> logs <podname> --previous
```

ImagePullBackoff

- The specified image cannot be downloaded
- Private image repositories - `imagePullSecrets`

PVC pending / not bound

- Kubectl get / describe both PVC and PV
- Kubectl get / describe *storageclass*
- Disk space?
- `kubectl -n longhorn-system get pods`

Cannot Attach Volume

```
FailedAttachVolume: AttachVolume.Attach failed for volume
"my-volume" : volume is already exclusively attached to
another node
```

AccessMode RWO

Pod stuck in waiting or pending

- Pending - the pod has not yet been assigned to a node
- Waiting - the pod has been assigned to a node but the container has not yet started

```
0/3 nodes are available: 3 Insufficient memory.
```

Lack of resources, taints etc.

Container OOMKilled

- Check the memory requests/limits in the container definition (kubectl describe)
- Use ‘`kubectl top nodes`’ ‘`kubectl top -n <namespace>`’

Cluster resources

- `kube-capacity --util` – requests, limits, used
- `kubectl describe node` - check the sections named **`Capacity`** and **`Allocatable`**

PAUSE

All-in with cert-manager

cert-manager creates TLS certificates for workloads in a Kubernetes cluster and renews the certificates before they expire.

Excellent debugging documentation:

https://cert-manager.io/docs/troubleshooting/

The short guide for debugging cert-manager

- `kubectl get / describe` each of them in turn to hunt for errors:

[Ingress, Issuer] → Certificate → certificateRequest → Order → Challenge

- `kubectl get crds | grep cert-manager`

- **solvers** are defined in Issuer: http01 or dns01

Cert-manager - when nothing works

- Read the documentation again
- Get some sleep
- Assign the task to another colleague

Thank you

- https://github.com/viorel-anghel/kubernetes-debug-techniques-2025
- Quick recap
    - Kubectl get / describe for everything
    - Kubectl logs / exec / debug
- Questions? Ask the experts!

EOF