

K8s Troubleshoot #01: Intro to Dodging Production Hell

Check GitHub for helpful DevOps tools:

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



Ask Personal AI Document assistant to learn interactively (FASTER)!

1


Download PDF

1

<https://github.com/MichaelRobotics/DevOpsTools/blob/main/K8sTrouble01.pdf>

2

Go to website

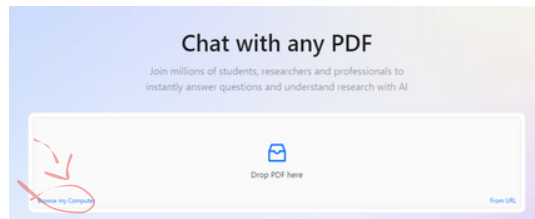
 | Click there to go to ChatPdf website

2

3

Browse file

3



4

Chat with Document

Ask questions about document!


4

Completely new to Kubernetes?

If you are completely new to this topic, using a document assistant to understand the many definitions can be helpful. However, the best way to start is by watching this video, which I believe provides the best explanation for beginners starting their journey with Kubernetes

Kubernetes Crash Course for Absolute Beginners

Hands-On Kubernetes Tutorial | Learn Kubernetes in 1 Hour - Kubernetes Course for Beginners

 https://www.youtube.com/watch?v=s_o8dwzRlu4&ab_channel=TechWorldwithNana



What is Kubernetes Troubleshooting

Kubernetes troubleshooting involves diagnosing and resolving issues within a Kubernetes cluster, such as deployment failures, pod crashes, or network problems. It requires examining logs, monitoring system metrics, and analyzing cluster configurations to identify and fix the root causes of problems.



kubernetes

How Kubernetes troubleshooting is done?

Kubernetes troubleshooting involves using tools like **kubectl** to inspect pod logs, events, and resource statuses to diagnose issues. Additionally, analyzing cluster metrics and leveraging Kubernetes' built-in debugging features, such as **kubectl exec** to access pod shells, can help identify and resolve problems.

When Kubernetes troubleshooting is done?

Kubernetes troubleshooting is performed when issues arise, such as

- ImagePullBackOff,
- Loopbackoff errors,
- unschedulable pods,
- resource sharing conflicts,
- breaches of resource quotas or limits,
- problems with StatefulSets and Persistent Volumes
- security breaches due to faulty network policies.

This PDF will focus on first from the list: **ImagePullBackOff**

System Requirements

- 2 CPUs or more
- 2GB of free memory
- 20GB of free disk space
- Internet connection
- ubuntu 22.04

If you want to install it on a different cloud provider, ask in the comments and I'll provide a solution for you!

Kubernetes: Main components & packages

install minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

start cluster

```
minikube start
```

let minikube download proper version of kubectl

```
minikube kubectl -- get po -A
```

K8s Troubleshoot #01: ImagePullBackOff

1) Understand popular case study

A company faced frequent Kubernetes ImagePullBackOff errors while deploying their application in a production environment, leading to failed container startups and deployment delays.

2) Error explanation

ImagePull indicates that Kubernetes kubelet cannot pull image from ECR or Docker registry

The BackOff part indicates that Kubernetes will keep trying to pull the image, with an increasing back-off delay.

Kubernetes raises the delay between each attempt until it reaches a compiled-in limit, which is 300 seconds (5 minutes).

3) Problem causes

After investigating, the team discovered

- incorrect container image tags
- issues with ECR / DOCKER registry authentication.
- Network issues

By correcting the image tag, configuring imagePullSecrets and investigating network related logs, they successfully resolved the error and ensured smooth deployments.

3) Solution #1 - Correct image tag

Check this deployment file, I intentionally modified image:nginx:1.14.2 to nginy:1.14.2

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginy:1.14.2
          ports:
            - containerPort: 80
```

After deployment, check Pods

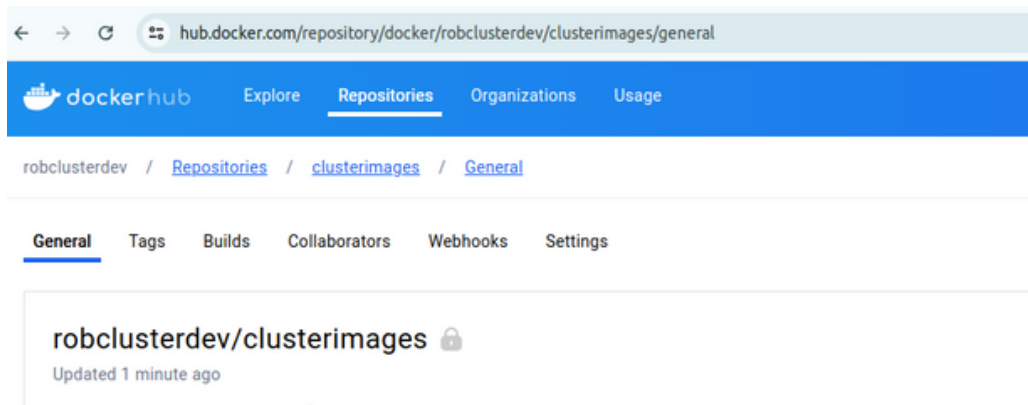
```
laptopdev@laptopdev2:~$ kubectl get pods -w
NAME                                READY   STATUS                    RESTARTS   AGE
nginx-deployment-5758546f8f-g747z  0/1     ContainerCreating        0          3s
nginx-deployment-5758546f8f-p47zv  0/1     ContainerCreating        0          3s
nginx-deployment-5758546f8f-p47zv  0/1     ErrImagePull             0          5s
nginx-deployment-5758546f8f-g747z  0/1     ErrImagePull             0          7s
nginx-deployment-5758546f8f-p47zv  0/1     ImagePullBackOff         0          19s
nginx-deployment-5758546f8f-g747z  0/1     ImagePullBackOff         0          23s
```

As expected, the first thrown error is ErrImagePull, which indicates that the kubelet cannot find this image in the public registry. Secondly, ImagePullBackOff means that it cannot pull the image after repeated attempts, with increasing time intervals.

Now, fix the typo (change nginy:1.14.2 to nginx:1.14.2) and redeploy the pods. Everything should work fine this time.

4) Solution #2 - Create secret to authenticate

Create docker account and your own repo. Make it private



Then Login into docker hub through CLI

```
laptopdev@laptopdev2:~$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a
docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants bet
nizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: robclusterdev
Password:
WARNING! Your password will be stored unencrypted in /home/laptopdev/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Pull nginx image, tag it accordingly to your repo name (Free account can only handle 1 repository) and push this image to your repo

```
laptopdev@laptopdev2:~$ docker pull nginx:1.14.2
1.14.2: Pulling from library/nginx
27833a3ba0a5: Pull complete
0f23e58bd0b7: Pull complete
8ca774778e85: Pull complete
Digest: sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160306b8d
Status: Downloaded newer image for nginx:1.14.2
docker.io/library/nginx:1.14.2
laptopdev@laptopdev2:~$ docker tag nginx:1.14.2 robclusterdev/clusterimages:nginx
laptopdev@laptopdev2:~$ docker push robclusterdev/clusterimages:nginx
The push refers to repository [docker.io/robclusterdev/clusterimages]
82ae01d5004e: Preparing
b8f18c3b860b: Preparing
5dacd731af1b: Preparing
```

Modfiy manifest, set image as you named you nginx container, mine is:

robclusterdev/clusterimages:nginx

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: robclusterdev/clusterimages:nginx
        ports:
        - containerPort: 80
```

Same error occurs, as expected.

```
laptopdev@laptopdev2:~$ nano nginx-deploy.yaml
laptopdev@laptopdev2:~$ kubectl get pods
No resources found in default namespace.
laptopdev@laptopdev2:~$ kubectl apply -f nginx-deploy.yaml
deployment.apps/nginx-deployment created
laptopdev@laptopdev2:~$ kubectl get pods -w
NAME                                READY   STATUS             RESTARTS   AGE
nginx-deployment-65fc8f6cc-mnx98    0/1     ErrImagePull       0          8s
nginx-deployment-65fc8f6cc-xpk6p    0/1     ErrImagePull       0          8s
nginx-deployment-65fc8f6cc-xpk6p    0/1     ImagePullBackOff   0          19s
nginx-deployment-65fc8f6cc-mnx98    0/1     ImagePullBackOff   0          22s
```


To resolve issue, create an kubernetes secret that stores your login credentials

```
kubectl create secret docker-registry secret-tiger-docker \
  --docker-email=tiger@acme.example \
  --docker-username=<your_username>
  --docker-password=<your_pass>
  --docker-server=https://index.docker.io/v1/
```

Modify manifest file adding a secret **secret-tiger-docker**:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: robclusterdev/clusterimages:nginx
          ports:
            - containerPort: 80
          imagePullSecrets:
            - name: secret-tiger-docker
```

Everything should work fine. If error persists, check if you put right parameters into secrets and manifest file.

4) Solution #3 - Look for network related issues

There is a lot to check, so I've prepared a comprehensive list of checks that you can use when diagnosing this kind of issue

Network Policies in Kubernetes define how pods communicate with each other and with external services. Misconfigured network policies can block legitimate traffic or allow unintended access.

DNS Resolution is critical for service discovery and communication between services. DNS issues can cause failures in service-to-service communication or external requests.

Registry Availability Issues with pulling images from a Docker registry can stem from connectivity problems or registry downtime.

Ingress Controllers manage external access to services within the cluster. Misconfigurations can block external traffic or expose services unintentionally.

Kubernetes Services expose pods and manage load balancing. Issues with services can lead to connectivity problems.

Linux Firewalls on nodes can block traffic to or from Kubernetes components.

Cloud Provider Firewalls control access to your nodes and services in cloud environments.

Proxy Configurations are necessary if your Kubernetes environment or services require a proxy to connect to external resources.


6) Prevention and Problem Investigation

Follow tags naming conventions

Check this great article

Recommended Labels

A common set of labels allows tools to work interoperably, describing objects in a common manner that all tools can understand.

 https://www.youtube.com/watch?v=s_o8dwzRlu4&ab_channel=TechWorldwithNana



Pre-pull images

Create a system of local-images storage, so when cluster cannot connect to registry it will pull images from local environment

Use observability solutions like Datadog for alerts

Create log analysis system to automate problem investigation process

Kubernetes

Run the Datadog Agent in your Kubernetes cluster to start collecting your cluster and applications metrics, traces, and logs.

 <https://docs.datadoghq.com/containers/kubernetes/#overview>



Validate Kubernetes manifest

Check: Kubeval, kube-score, helm lint, kubeuadit

Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.



<https://github.com/MichaelRobotics>



PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!