My Kubernetes Troubleshooting Checklist

When working with Kubernetes, especially in real-world deployments, you often run into issues where pods fail to start. These are usually labeled as:

- ImagePullBackOff
- ErrImagePull
- ImageInspectError
- ErrImageNeverPull

Each of these errors relates to startup or image issues, and understanding them helps you debug faster and deploy with confidence.

✓ Startup Errors

- Startup errors occur when a container fails during its initialization. These aren't limited to image problems — they can include:

  Application crashes (code bugs)

  Missing config/env variables

  Invalid mounts or volume issues

  Failing readiness/liveness probes

✓ ImagePullBackOff

When Kubernetes continually fails to get a container image from a registry, this error happens. When the initial attempt to get the image is unsuccessful, Kubernetes backs off and makes more attempts at progressively longer intervals.

In simple way :

Image Pull----------[Trouble in Pulling Image]----------Back Off

Note : For ImagePullBackOff error, the first error is "ErrorImagepull" then the error is changes to "ImagePullBackOff"

Possible ways to Causes:

- Incorrect image name or tag – The specified image does not exist.
- Authentication issues – The registry requires credentials that are missing or incorrect.
- Network problems – Connectivity issues prevent access to the registry.
- Registry quotas exceeded – The registry has rate limits that block further pulls

How to Debug:

```
kubectl describe pod <pod-name>
```

Look under Events, you'll see something like:

Expected Output (with error):

```
Failed to pull image "nginx:latestt": image not found Back-off pulling image "nginx:latestt"
```

Command Break Down

- *kubectl → The Kubernetes CLI tool.*
- *describe → Fetches detailed information about the specified resource.*
- *pod <pod-name> → Specifies the pod whose details you want to examine.*
- *When this command is used, detailed information on the pod is produced, including:*
  Namespace, Labels, and Pod Name: Namespace, annotations, and labels are examples of basic metadata.
- *Pod Status: Indicates if the pod has succeeded, failed, pending, or is running.*
  Displays current happenings that have an impact on the pod.
- *Node Assignment: Indicates which worker Kubernetes node the pod is operating on.*
  Details on the containers in the pod, including picture versions, are listed here.

Variables related to the container environment (env).

Requests for and limitations on CPU and memory resources.

- *Volume Mounts: Shows the pod's internal persistent storage mounts.*

Current Occurrences & Mistakes

Displays errors, warnings, and other pod-related system events.

Expected Output (with out error):

```
Name:           my-app-pod
Namespace:      default
Labels:         app=my-app
Status:         Running
Node:           worker-node-1
Containers:
  my-app:
    Image:        nginx:latest
    State:        Running
    Ready:        True
    Restart Count: 0
Events:
  Type    Reason     Age                From               Message
  ----    ------     ----               ----               -------
  Normal  Scheduled  3m                 default-scheduler  Successfully assigned default/my-app-pod to worker-node-1
  Normal  Started    2m                 kubelet            Started container my-app
```

This often gives more clues than logs: failed mounts, image pull issues, liveness probe failures, or event errors.

If your pod keeps crashing or restarting, follow these steps:

Check logs: kubectl logs <pod-name> → Identify specific error messages.
Inspect events: kubectl get events --sort-by=.metadata.creationTimestamp → Look for failed pod scheduling.

Check pod details: kubectl describe pod <pod-name> → Look for misconfigurations or failure reasons.

- ✓ ErrImagePull

This error indicates that the picture was not pulled immediately. ErrImagePull signals that Kubernetes is unable to obtain the image, as contrast to ImagePullBackOff, which attempts again.
Causes:

- Nonexistent image – The image name or tag is incorrect.
- Private registry access issues – Credentials are missing or invalid.
- Network restrictions – Firewalls or DNS issues prevent access

How to fix :

- Incorrect Image Name or Tag

  Problem: The specified image name is incorrect or the tag does not exist.

  Solution:

  Verify the image name and tag using:

  kubectl describe pod <pod-name> | grep Image

  Check if the image exists in your registry:

  docker pull <image-name>:<tag>

  If the image is incorrect, update the deployment:

  ```
  yaml
  containers:
    - name: my-container
      image: nginx:latest  # Ensure correct image name
  ```

- Image Does Not Exist in Registry

  Problem: The requested image is unavailable in the specified registry.

  Solution:

  For public images, check the registry:

  curl -s https://hub.docker.com/v2/repositories/library/nginx/tags/

  For private images, ensure the image exists in the repository.

- Authentication Issues (Private Registry)

Problem: Kubernetes cannot access the private registry due to missing credentials.

Solution:

Create a Kubernetes Secret for registry authentication:

```
kubectl create secret docker-registry myregistrykey \
  --docker-server=<registry-url> \
  --docker-username=<username> \
  --docker-password=<password>
```

Reference the secret in your Pod specification:

```
yaml
imagePullSecrets:
  - name: myregistrykey
```

- Network Issues Preventing Image Pull

  Problem: Kubernetes nodes cannot reach the image registry due to network failures.

  Solution:

  Check if you can pull the image manually:

  ```
  docker pull nginx:latest
  ```

  Test internet connectivity from the node:

  ```
  curl -I https://hub.docker.com
  ```

  Ensure your firewall or security group allows traffic to the container registry.

- Misconfigured Node Permissions

  Problem: Kubernetes does not have permission to pull images. Solution:

  Check pod events:

  ```
  kubectl get events --sort-by='.metadata.creationTimestamp'
  ```

  Restart the node if necessary:

  ```
  sudo systemctl restart kubelet
  ```

- ✓ ImageInspectError

Kubernetes was able to pull the image, but could not inspect it. This means something is wrong with the image itself.

Possible Causes:

- Corrupted or malformed image

- Missing entrypoint or CMD
- Permissions issue
- Incompatible image format (like non-Linux or foreign architecture)

How to Fix:

Try docker pull <image> manually

Validate the image works locally

- Incorrect Image Name or Tag

Problem: The specified image name or tag is incorrect. Fix:

Verify the image name and tag using:

kubectl describe pod <pod-name> | grep Image

Pull the image manually to confirm availability:

docker pull <image-name>:<tag>

If the image is incorrect, update the Deployment YAML:

yaml

containers:

- name: my-container

image: nginx:latest # Ensure correct image name

To Apply the changes:

kubectl apply -f deployment.yaml

- Image Corruption or Registry Issues

Problem: The image might be corrupt or not properly stored in the registry.

Fix:

Remove the faulty image manually:

docker rmi <image-name>:<tag>

Pull the image again:

docker pull <image-name>:<tag>

Confirm the registry has the correct image by checking logs:

kubectl logs <pod-name> --previous

- Authentication Issues (Private Registry)

Problem: Kubernetes cannot access a private container registry due to missing credentials. Fix:

Check if your registry requires authentication.

Create a Kubernetes Secret for private registry authentication:

*kubectl create secret docker-registry myregistrykey *

*--docker-server=<registry-url> *

*--docker-username=<username> *

--docker-password=<password>

Reference the secret in your pod specification:

yaml

imagePullSecrets:

- name: myregistrykey

Restart the pod and check if the issue persists.

- Network Connectivity Issues

Problem: Kubernetes nodes cannot reach the registry due to firewall rules or DNS issues.

Fix:

Test connectivity by manually pulling the image:

docker pull <image-name>:<tag>

Check internet access using:

curl -I https://hub.docker.com

Restart the network service:

sudo systemctl restart network-manager

- Misconfigured Node Permissions

Problem: Kubernetes nodes lack permission to pull images due to incorrect role assignments.

Fix:

View pod events to check for permission errors:

kubectl get events --sort-by='.metadata.creationTimestamp'

Restart the Kubelet service:

sudo systemctl restart kubelet

Restart the entire node if necessary:

sudo reboot

Quick Fix: Force Kubernetes to Retry

After fixing the issue, delete and recreate the pod:

kubectl delete pod <pod-name>

kubectl apply -f <deployment.yaml>

- ✓ ErrImageNeverPull

This occurs when:

The pod has image pull policy Never and the image is not available locally on the node

yaml

imagePullPolicy: Never

This tells Kubernetes:

“Don't try to pull this image. Just use it if it's already on the node.”

If it’s not there — boom: ErrImageNeverPull.

<u>Tips to Avoid These Issues:</u>

- *Always use valid imagePullPolicy*

```
yaml
CopyEdit
imagePullPolicy: Always
        or
yaml
CopyEdit
imagePullPolicy: IfNotPresent
```

- *Use Kubernetes Secrets for Private Repos*

```
kubectl create secret docker-registry myregistrykey \
  --docker-server=<registry> \
  --docker-username=<user> \
  --docker-password=<pass>
```

Then reference the secret in your pod YAML.

- *Validate Image Locally Before Deploying*

```
docker pull <image>
docker inspect <image>
```

- *Use kubectl describe and kubectl logs*

They’re your first line of defense.

Tips to fix the issue

If your pod keeps crashing or restarting, follow these steps:

Check logs: kubectl logs <pod-name> → Identify specific error messages.
Inspect events: kubectl get events --sort-by=.metadata.creationTimestamp → Look for failed pod scheduling

Check pod details: kubectl describe pod <pod-name> → Look for misconfigurations or failure reasons.

Verify resource limitations: Adjust memory requests and limits (spec.containers[].resources in YAML) if the pod is OOMKilled.

Fix:

- *If the crash is due to missing dependencies, ensure the Docker image has the required libraries.*
- *If it's an OOMKilled issue, increase memory limits in deployment YAML.*
- *If it's an environment variable issue, verify config maps and secrets are correctly mounted.*