

33Kubernetes 系列（三十一）POD 排障，解决你 99% 异常问题

Kubernetes 系列（三十一）POD 排障，解决你 99% 异常问题

您应该从底层开始对部署进行故障排除。

首先，检查 POD 是否处于运行状态。

如果“Pods”已经准备好，您应该调查服务是否可以将流量分配到“Pods”。

最后，您应该检查 Ingress 和服务之间的连接。

大多数情况下，问题出现在Pods本身。你应该确保 Pods 处于运行和就绪状态。

你可以查看Pods的状态如下所示，

```
controlplane ~ → kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
nginx                1/1     Running   0           7m41s
newpods-jcbdf        1/1     Running   0           7m14s
newpods-xshmj        1/1     Running   0           7m14s
newpods-n5lx7        1/1     Running   0           7m14s
busybox-pod          0/1     ErrImagePull 0           云原生拓展
```

在输出中，最后一个Pod既不是Running也不是Ready。

您可以使用以下四个命令来调查出错的地方。

1. **kubectl logs <pod name>** :用于查看pod容器的日志


```
controlplane ~ → kubectl logs busybox-pod
Error from server (BadRequest): container "busybox-pod" in pod "busybox-pod" is waiting to start
: trying and failing to pull image 云原生拓展
```

2. **kubectl describe pod <pod name>** :用于检索与pod相关的事件列表

```
controlplane ~ → kubectl describe pod busybox-pod | grep "Events" -A10
Events:
  Type       Reason      Age                From              Message
  ----       -
  Normal     Scheduled   9m30s             default-scheduler Successfully assigned default/b
usybox-pod to controlplane
  Normal     Pulling     8m8s (x4 over 9m29s) kubelet            Pulling image "busybox123"
  Warning    Failed      8m8s (x4 over 9m28s) kubelet            Failed to pull image "busybox12
3": rpc error: code = Unknown desc = failed to pull and unpack image "docker.io/library/busybox1
23:latest": failed to resolve reference "docker.io/library/busybox123:latest": pull access denie
d, repository does not exist or may require authorization: server message: insufficient_scope: a
uthorization failed
  Warning    Failed      8m8s (x4 over 9m28s) kubelet            Error: ErrImagePull
  Warning    Failed      7m38s (x6 over 9m27s) kubelet            Error: ImagePullBackOff
  Normal     BackOff     4m28s (x20 over 9m27s) kubelet            Back-off pulling image "busybox
123" 云原生拓展
```


3. **kubectl get pod <pod name> -o yaml** :用于提取存储在Kubernetes中的pod的yaml定义。

```
controlplane ~ → kubectl get pod busybox-pod -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-10-21T07:02:50Z"
  labels:
    run: busybox-pod
  name: busybox-pod
  namespace: default
  resourceVersion: "1255"
  uid: 8abfc278-c533-45d0-9ff8-4d90570b5f6f
spec:
  containers:
  - image: busybox123
    imagePullPolicy: Always
    name: busybox-pod
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-bhskk
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: controlplane
```

 云原生拓展

4. **kubectl exec -it <pod name> bash** :用于在pod的一个容器中运行交互式命令。

```
controlplane ~ → kubectl exec -it nginx -- bash
root@nginx:/# pwd
/
root@nginx:/# ls
bin  dev  docker-entrypoint.sh  home  lib64  mnt  proc  run  srv  tmp  var
boot  docker-entrypoint.d  etc  lib  media  opt  root  sbin  sys  usr
root@nginx:/#
```

 云原生拓展

常见Pods错误

错误有两种类型,

1. 启动错误
2. 运行时错误

启动错误包括以下几种:

- ImagePullBackoff
- ImageInspectError
- ErrImagePull
- ErrImageNeverPull
- RegistryUnavailable

- InvalidImageName

运行时错误包括:

- CrashLoopBackOff
- RunContainerError
- KillContainerError
- VerifyNonRootError
- RunInitContainerError
- CreatePodSandboxError
- ConfigPodSandboxError
- KillPodSandboxError
- SetupNetworkError
- TeardownNetworkError

最常见的错误和如何修复它们

ImagePullBackOff :

这个错误意味着 K8s 无法为Pod中的一个容器拉取镜像。

常见的错误原因可能是以下之一，

1. 镜像名称无效
2. 您为镜像指定了不存在的标记
3. 您试图拉取的镜像属于私有注册中心，k8s没有访问它的凭据。

前两种情况可以通过纠正镜像名称和标记来解决。

最后，您应该在Secret中将凭证添加到您的私有注册表中，并在您的Pods中引用它。

CrossLoopBackOff:

如果容器不能启动，那么K8s将显示 CrashLoopBackOff 消息作为状态。

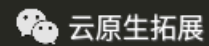
通常，容器在以下情况下不能启动:

1. 应用程序中有一个错误，阻止它启动。
2. 您错误地配置了容器。
3. alive探测失败了太多次。

您应该尝试从该容器检索日志，以调查它失败的原因。

如果因为容器重启太快而看不到日志，可以使用以下命令：

```
$ kubectl logs <pod-name> --previous _
```



它打印来自前一次容器的错误信息。

RunContainerError:

当容器无法启动时出现该错误。

这甚至是在容器内的应用程序启动之前。

该问题通常是由错误配置造成的，例如：

- 挂载不存在的卷，如ConfigMap、Secrets等。
- 将只读卷挂载为可读写。

你应该使用 `kubectl describe pod <pod-name>` 来检查和分析错误。

pod 处于挂起(Pending)状态:

当你创建一个Pod时，Pod保持在 *Pending* 状态。

为什么？

假设您的调度程序组件运行良好，以下是可能原因：

1. 集群没有足够的资源(如CPU和内存)来运行Pod。
2. 当前的命名空间有一个ResourceQuota对象，创建Pod将使命名空间超过配额。
3. Pod被绑定到 *Pending* PersistentVolumeClaim。

你最好的选择是检查 `kubectl describe` 命令中的 *Events* 部分：

```
$ kubectl describe pod <pod name> _
```

对于由于 ResourceQuotas 而产生的错误，您可以使用以下方法检查集群的日志：

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

Pods 处于未就绪(not Ready)状态:

如果一个Pod是Running 但不是 Ready(Deployment 等控制类型状态), 这意味着 Readiness探测失败。

当Readiness探测失败时, Pod不会附加到Service, 并且没有流量被转发到该实例。

Readiness 探测失败是应用程序特有的错误, 因此您应该检查 `kubectl describe` 中的 *Events* 部分来识别错误。

欢迎关注我的公众号“云原生拓展”, 原创技术文章第一时间推送。