Technical Guide

Kubernetes 애플리케이션 트러블슈팅

September 2021

SAMSUNG SDS

Contents

1. 개요

이 문서는 Kubernetes 클러스터에 워크로드를 배포하는 과정 또는 워크로드 운영 도중 발생할 수 있는 문제점에 대한 해결 방법을 소개합니다.

워크로드 유형 중에서 주로 사용되는 웹 애플리케이션 유형을 다루고 있으며 바닐라 Kubernetes, Public Managed Kubernetes (GKE, AKS, EKS 등)를 포함하여 SDS Cloud 상품인 Kubernetes Engine 과 Kubernetes Apps 에도 이 문서를 참고하여 문제를 해결할 수 있습니다.

2. 애플리케이션 일반 구조

Kubernetes 에서 웹 애플리케이션을 외부에서 접근을 위한 방식으로는 hostNetwork, nodePort, type: LoadBalancer 등 여러가지 방법이 있지만, 이 중 http/https 기반 도메인 호출 방식 접근에 용이한 Ingress 방식의 워크로드를 다루겠습니다.

다음은 nginx 웹 애플리케이션을 Kubernetes 워크로드로 배포하기 위한 Kubernetes resource 조합의 예 입니다.

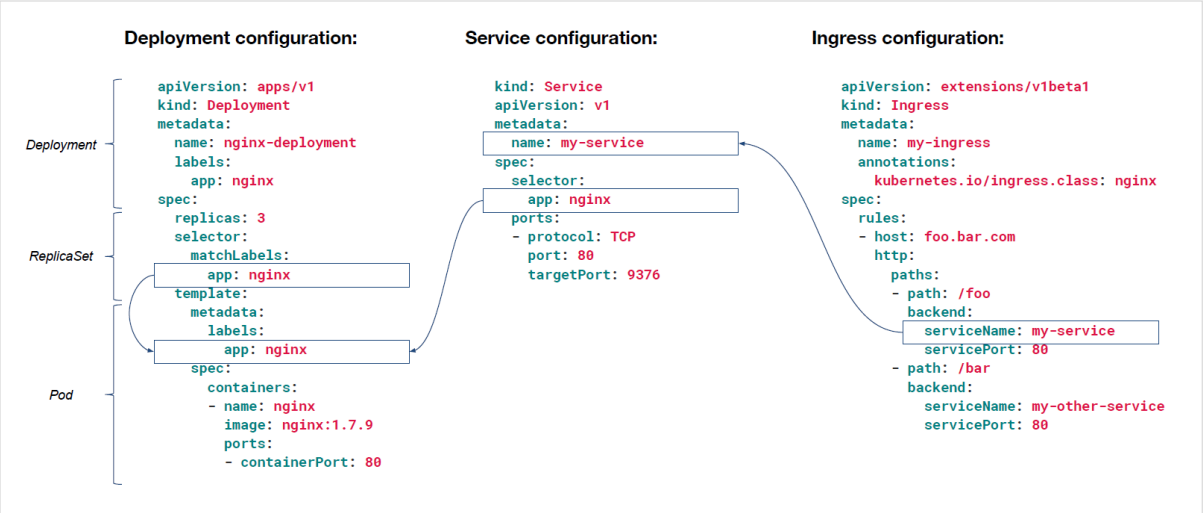

Figure 1. Deployment strategies on Kubernetes, CNCF

이 외에도 persistent volume 또는 network policy 등 다양한 resource 들이 존재할 수 있지만, 위 예시가 Ingress 를 사용하는 가장 기본적인 Kubernetes resource 들 입니다.

여기서 중요한 점은 각 resource 들의 참조 관계입니다. Deployment 와 Service 의 selector 는 모두 Pod template 의 label 을 참조하고, Ingress 에서 serviceName 은 Service resource 의 name 을 참조한다는 점을 반드시 알아야 합니다.

다음부터 다룰 내용은 각 resource 들이 배포되었을 때 정상적인 상태 점검 방법과 문제가 발생했을 때 어떻게 대처해야 하는지 알아보겠습니다.

3. Pod 상태 점검

3.1 기본 명령어

Pod 의 상태를 점검하기 위해 필요한 kubectl 명령어 몇가지를 소개합니다.

기본적인 Pod 의 상태를 확인하거나 IP 및 할당된 노드의 정보를 포함한 확장 명령어(-owide), yaml manifest 형태의 rawdata 를 확인(-oyaml)하는 가장 기본적인 조회 명령어입니다.

```
$ kubectl get pod
$ kubectl get pod -owide
$ kubectl get pod -oyaml
```

다음은 주로 문제가 발생 한 경우 Pod 의 상세한 정보 및 Event 를 확인하기 위해 사용되는 명령어입니다.

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

구동되고 있는 애플리케이션에 문제가 생겼을 경우 로그를 확인할 수 있는 명령어며, Pod 안에 다중 Container 가 구동되고 있는 경우 -c 옵션을 통해 특정 Container 를 지정할 수 있고, -f 를 통한 실시간 tailing, --previous 를 통한 이전 Pod 에 대한 로그도 확인이 가능합니다.

```
$ kubectl logs <pod-name> [-c <container-name>]
$ kubectl logs <pod-name> [-c <container-name>] -f
$ kubectl logs <pod-name> [-c <container-name>] --previous
```

아래는 Namespace 내에 발생한 Event 들을 보여주는 명령어고, --sort-by 옵션을 통해 발생한 시간순으로 정렬이 가능합니다.

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

3.2 정상적인 Pod Lifecycle

Pod 를 배포했을 때 정상적인 상태로 변경되기까지의 상태 변화 흐름을 알아야 합니다.

일반적인 Pod 는 다음과 같은 Lifecycle 을 갖습니다.

- **Pending**: Pod 가 스케줄 될 노드가 결정되기 전 또는 스케줄 가능한 노드가 없는 경우 표시되는 상태입니다.
- **ContainerCreating**: Pod 가 특정 노드에 스케줄된 후에 해당 노드 Container Runtime 이 Container 를 생성하는 단계이며, Image Pulling, Configmap & Persistent Volume 등 연계 자원을 마운트하는 과정에 표시될 수 있습니다.
- **Running**: 정상적으로 Pod 내 Process 가 구동중인 상태입니다.
- **Ready**: Pod 가 Running 된 이후, 해당 Pod 에 설정된 Readiness Probe 에 의해 Pod 가 서비스할 준비가 되었음을 의미합니다.
- **Terminating**: Pod 가 Delete 되어 해당 노드에서 Pod 를 정리하는 과정에서 표시될 수 있습니다.

아래 kubectl get po 명령어에 –w (watch) 옵션을 통해 Pod Lifecycle 변화를 확인 할 수 있습니다. 이 중 Pod 의 정상 상태를 확인하는 가장 기본적인 컬럼은 READY 와 STATUS 입니다.

```
$ kubectl get po -owide -w
NAME                     READY   STATUS              AGE   IP          NODE
nginx-5d796fc999-qkgpp   0/1     Pending             0s    <none>      <none>
nginx-5d796fc999-qkgpp   0/1     Pending             0s    <none>      node1
nginx-5d796fc999-qkgpp   0/1     ContainerCreating   1s    <none>      node1
nginx-5d796fc999-qkgpp   0/1     Running             3s    10.44.0.2   node1
nginx-5d796fc999-qkgpp   1/1     Running             9s    10.44.0.2   node1
```

다음부터는 비정상 Pod 상태가 발생하는 사례 별로 원인과 조치방안에 대해 알아보겠습니다.

3.3 Pending 상태가 지속되는 경우

Pod 의 상태가 Pending 에서 장시간 머무는 경우가 있습니다.

```
$ kubectl get po
NAME                     READY   STATUS    RESTARTS   AGE
nginx-6dcd8d4dff-njsrh   0/1     Pending   0          10m
```

이런 경우엔 먼저 kubectl describe pod 명령어를 통해 Events 메시지를 확인해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason            Age                From
  Message
  ----     ------            ----               ----               -----
  Warning  FailedScheduling  14s (x2 over 14s)  default-scheduler  0/3 nodes are
available: 3 Insufficient cpu/memory.
```

위의 경우 클러스터에 자원이 부족한 경우 입니다. kubectl describe node 명령어를 통해 각 노드에 할당된 Allocated resources: 를 확인하여야 하며, 필요한 경우 클러스터 노드 자원을 증설하거나 Pod 자원을 감량해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason            Age              From               Message
  ----     ------            ----             ----               -------
  Warning  FailedScheduling  5s (x2 over 5s)  default-scheduler  0/3 nodes are available:
3 node(s) didn't match node selector.
```

이 경우는 Pod 에 설정된 nodeSelector/nodeAffinity 와 매치되는 노드가 없는 경우 입니다. Pod 의 nodeSelector/nodeAffinity 를 매치되는 노드 정보로 수정하거나 노드에 label 을 추가하여 Pod 을 정상적으로 예약할 수 있습니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason            Age              From               Message
  ----     ------            ----             ----               -------
  Warning  FailedScheduling  6s (x2 over 6s)  default-scheduler  0/3 nodes are available:
3 node(s) were unschedulable.
```

이 경우는 노드가 예약이 불가능한 상태입니다. 노드가 모두 Cordon 되었거나 NotReady 상태가 되어 Taint 가 걸려있는 경우 입니다. 노드를 Uncordon 하거나 NotReady 상태의 노드를 조치해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  53s (x2 over 60s)  default-scheduler  error while running
"VolumeBinding" filter plugin for pod "nginx-9d8d9896-k7gzj": pod has unbound immediate
PersistentVolumeClaims
```

PVC (PersistentVolumeClaim)가 Pending 상태가 아닌지 다음과 같이 확인해야 합니다.

```
$ kubectl get pvc
NAME        STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS
nginx-pvc   Pending                                       standard
```

정적 PV (PersistentVolume)를 사용하는 경우에는 Available 상태의 PV 가 먼저 정상적으로 생성 된 후, PVC 의 volumeName 에 해당 PV name 이 제대로 일치하는지 확인해야 합니다.
동적 PV 를 사용하는 경우는 PVC 에 설정한 storageClassName 으로 생성된 StorageClass 가 있는지 확인 후 해당 Volume Provisioner 가 제대로 동작하고 있는지 확인해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  21s   default-scheduler  Successfully assigned default/nginx-
5d796fc999-6ksbj to node1
$ kubectl get po -owide
NAME                     READY   STATUS    RESTARTS   AGE   NODE    nginx-
5d796fc999-6ksbj   0/1     Pending   0          37s   node1
```

이 경우는 Pod 이 정상적으로 노드에 할당되었다는 이벤트만 나오고 Pod 이 계속 Pending 인 경우에는 -owide 조회 시 나오는 노드의 kubelet 이 정상적으로 동작하는지 확인 후 조치해야 합니다.

```
$ kubectl describe po <pod-name>
Events: <none>
$ kubectl get po -n kube-system
NAME                          READY   STATUS             RESTARTS   AGE
kube-scheduler-master         0/1     CrashLoopBackOff   9          16m
```

Event 에 메시지가 기록되지 않으면 kube-scheduler 가 정상적으로 스케줄 작업을 수행하지 않는 것이므로 조치해야 합니다.

3.4 ContainerCreating 상태가 지속되는 경우

다음은 Pod 가 ContainerCreating 상태에 머물러 있는 경우 입니다.

```
$ kubectl get po
NAME                        READY   STATUS              RESTARTS   AGE
nginx-6dcd8d4dff-njsrh      0/1     ContainerCreating   0          1m
```

이런 경우에도 먼저 kubectl describe pod 명령어를 통해 Events 메시지를 확인해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason       Age          From                Message
  ----     ------       ----         ----                -------
  Normal   Scheduled    <unknown>    default-scheduler   Successfully assigned
default/nginx-58cd5b85f7-cmb6c to node1
  Normal   Pulling      12s          kubelet, node1      Pulling image "nginx:1.13.9"
```

Events 메시지가 Pulling image 에서 멈춰 있는 단계라면 아직 Image 를 Pulling 하고 있는 상태이니 조금 더 기다려볼 필요가 있습니다. Image 용량이 크거나 Network 성능이 낮은 경우 Pulling 하는 시간이 오래 걸릴 수 있으니 Image Pulling 이 완료되면 자연스럽게 해결됩니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason       Age          From                Message
  ----     ------       ----         ----                -------
  Warning  FailedMount  5s (x6 over 21s)  kubelet, node2      MountVolume.SetUp failed
for volume "config-volume" : configmap/secret "nginx-cm" not found
```

Pod 이 사용하고 있는 ConfigMap/Secret 이 생성되어 있지 않거나 잘못된 이름을 명시한 경우 입니다. Pod Specification 의 volumes 필드에 지정된 ConfigMap/Secret 등을 실제로 존재하는지 확인해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason       Age          From                Message
  ----     ------       ----         ----                -------
  Warning  FailedMount        18s          kubelet, node2      MountVolume.SetUp failed
for volume "pvc-d2def29a-d503-443e-9615-886713983216" : mount failed: exit status 32
```

Pod 가 사용하고 있는 PV 와 마운트 되지 않은 경우에 발생합니다. 해당 PV 를 사용하기 위한 CSI(Container Storage Interface) Driver 또는 NFS 의 경우 nfs-utils 및 rpcbind 등의 client 가 설치되어 있는지 확인해야 합니다. 추가로 노드에서 해당 PV 에 대한 접근제어 및 방화벽은 확인하여 마운트가 가능하도록 조치를 해야 합니다.

3.5 ImagePullBackOff 상태로 표시되는 경우

이번에는 ImagePullBackOff 가 발생하는 상황에 대해서 알아보겠습니다.

```
$ kubectl get po
NAME                     READY   STATUS             RESTARTS   AGE
nginx-6dcd8d4dff-njsrh   0/1     ImagePullBackOff   0          6s
```

마찬가지로 describe 명령어를 통해 확인이 가능합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Warning  Failed     23s (x2 over 44s)  kubelet, node1     Failed to pull image
"nginx:invalid-tag": rpc error: code = Unknown desc = Error response from daemon:
manifest for nginx:test not found
```

해당 Image:Tag 를 받아오기 위한 Image Repository 에 Image:Tag 가 없는 경우입니다. Pod specification 에 올바른 Image:Tag 로 수정하거나 해당 Image:Tag 를 Image Repository 에 Push 해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Warning  Failed     9s                 kubelet, node1     Failed to pull image
"myregistry.io/nginx:1.17.8": rpc error: code = Unknown desc = Error response from
daemon: Get https://myregistry.io/v2/nginx/manifests/1.17.8: no basic auth credentials
```

Private Registry 를 사용하는 경우 Credential 정보가 필요한 경우가 있습니다. 이런 경우 올바른 Credential 정보로 imagePullSecrets 을 만든 후, Pod specification 에 추가해야 합니다. 이미 imagePullSecrets 을 사용 중인 경우라면 권한이 만료되지는 않았는지 추가로 확인이 필요합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason            Age                From                Message
  ----     ------            ----               ----                -------
  Warning  Failed            12s    kubelet, node1  Failed to pull image "nginx:1.17.8": rpc error:
code = Unknown desc = Error response from daemon: Get https://registry-1.docker.io/v2/:
net/http: request canceled while waiting for connection (Client.Timeout exceeded while
awaiting headers)
```

노드에서 해당 Image Repository 와 통신이 실패한 경우 입니다. 노드에서 Image Repository 간 방화벽 및 Image Repository 서비스가 정상 운영 중인지 확인해야 합니다.

3.6 CrashLoopBackOff, Running 상태가 주기적으로 반복되는 경우

Pod 이 CrashLoopBackOff 와 Running 상태가 주기적으로 반복되는 경우가 있습니다.

```
$ kubectl get po -w
NAME                    READY   STATUS             RESTARTS   AGE
test-7f74c45f58-jm629   0/1     CrashLoopBackOff   8          16m
test-7f74c45f58-jm629   0/1     Running            9          16m
```

이런 경우 아래 명령어를 통해 현재 Pod 또는 이전 Pod 로그를 확인하여, 로그가 확인되는 경우 애플리케이션의 문제를 해결 해야 합니다.

```
$ kubectl logs po <pod-name> -f
$ kubectl logs po <pod-name> --previous
```

```
$ kubectl describe po <pod-name>
 Last State:      Terminated
      Reason:          ContainerCannotRun
      Message:         OCI runtime create failed: container_linux.go:344: starting container
process caused "exec: ₩"ping₩": executable file not found in $PATH": unknown
      Exit Code:       127
```

주로 발생하는 애플리케이션 이슈로는 DB 연결 실패, 비정상 command 사용, 애플리케이션 설정 이슈 등이 있습니다.

현재 Pod 로그 또는 이전 Pod 로그를 볼 수 없는 경우라면 describe 명령어를 통해 Last State: 내용을 확인하여 조치 해야 합니다.

3.7 CrashLoopBackOff, Completed 상태가 주기적으로 반복되는 경우

Pod 이 CrashLoopBackOff 와 Completed 를 반복하는 경우도 생길 수 있습니다.

```
$ kubectl get po -w
NAME                    READY   STATUS             RESTARTS   AGE
test-6fd77b68b9-44hxs   0/1     CrashLoopBackOff   8          16m
test-6fd77b68b9-44hxs   0/1     Completed          9          16m
```

이 경우는 Container 내부에 Foreground Process 가 없는 경우 입니다. Dockerfile 에 CMD 또는 ENTRYPOINT 로 Foreground 로 동작할 수 있는 Process 를 명시하거나 Deploymenet 에 commnad 를 추가해야 합니다. 만약 Batch 성 Process 인 경우에는 Job 또는 CronJob 으로 생성해야 합니다.

3.8 Ready 상태로 변경이 안되는 경우

Pod 가 Running 은 되었지만 READY 로 상태가 변경되지 않는 경우 입니다.

```
$ kubectl get po
NAME                     READY   STATUS    RESTARTS   AGE
nginx-659484b897-mnq4l   0/1     Running   0          2m14s
```

이 경우에 describe 명령어를 수행합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason     Age               From               Message
  ----     ------     ----              ----               -------
  Warning  Unhealthy  5s (x7 over 65s)  kubelet, node2     Readiness probe failed: HTTP
probe failed with statuscode: 404
```

Events 메시지에서 Readiness probe 가 실패하지 않았는지를 확인한 후 조치를 해야하며, 애플리케이션 기동시간이 긴 경우 initailDelaySeconds 를 늘려야 합니다.

3.9 Terminating 상태가 지속되는 경우

이번에는 Terminating 상태가 지속되는 경우에 대해 알아보겠습니다.

```
$ kubectl get po -owide
NAME                    READY   STATUS        RESTARTS   AGE   NODE
nginx-6dcd8d4dff-njsrh  0/1     Terminating   0          10m   node2
nginx-6dcd8d4dff-fvlmf  1/1     Running       0          43s   node1
```

먼저 Pod 할당된 노드가 NotReady 가 아닌지 확인해야 합니다. NotReady 가 된 상태에서 워크로드 종류에 따른 Pod 의 Eviction 정책이 다르다는 점 또한 알아야 합니다.

```
$ kubectl get no
NAME      STATUS      ROLES     AGE    VERSION
master    Ready       master    39d    v1.21.2
node1     Ready       worker    39d    v1.21.2
node2     NotReady    worker    39d    v1.21.2
```

- Deployment: 다른 노드로 Failover 되었는지 확인
- StatefulSet: 다른 노드로 이동되지 않음 (강제 삭제 시 이동 가능)
- DaemonSet: 다른 노드로 이동되지 않음

만약 Pod 이 할당된 노드가 Ready 라면 Pod 종료 과정에서 PV 가 unmount 또는 detach 가 제대로 동작하지 않는 경우일 수 있어 스토리지 연결상태를 확인해야 합니다. 다른 원인으로는 해당 노드에 좀비(defunct) 프로세스가 생성되지 않았는지 확인 후 조치해야 합니다.

3.10 Evicted 된 경우

Pod 이 Evicted 된 경우 입니다.

```
$ kubectl get po -owide
NAME                     READY   STATUS    RESTARTS   AGE   NODE
nginx-58cd5b85f7-4fz88   0/1     Evicted   0          1h    node1
nginx-58cd5b85f7-ts7ps   1/1     Running   0          12m   node2
```

해당 노드를 describe 하여 노드의 상태를 확인해야 합니다.

```
$ kubectl describe no node1
Taints:             node.kubernetes.io/disk-pressure:NoSchedule
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime
Reason                       Message
  ----                 ------  -----------------                 ------------------
------                       -------
  DiskPressure         True    Tue, 20 Oct 2020 12:22:14 +0900   Wed, 14 Oct 2020
20:37:51 +0900   KubeletHasDiskPressure       kubelet has disk pressure
```

노드에 Disk 공간이 부족하여 DistPressure 가 발생했는지 확인한 후 부족한 공간을 확보해야 합니다.

```
$ df -h /var/lib/kubelet
Filesystem                  Size  Used Avail Use% Mounted on
/dev/mapper/VGROOT-LV_root   50G   46G  4.7G  91% /
```

노드에 /var/lib/kubelet 영역이 속한 디스크가 90% 이상 찬 경우 일반적으로 Evicted 가 발생 됩니다. (default kubelet 정책 사용 시)

3.11 Restart 가 자주 발생하는 경우

Pod 가 현재는 정상적으로 구동 중인데 restart 가 자주 발생하는 경우가 있습니다.

```
$ kubectl get po -n kube-system
NAME                READY   STATUS    RESTARTS   AGE
weave-net-kwzvz     2/2     Running   5          5d1h
```

--previous 옵션을 통해 이전 Pod 로그 를 확인하여 문제 해결이 가능 한 경우에는 애플리케이션에서 발생한 문제를 해결해야 합니다.

```
$ kubectl logs weave-net-kwzvz -n kube-system -c weave --previous
FATA: 2020/10/15 04:25:03.297881 [kube-peers] Could not get peers: Get
https://172.24.0.1:443/api/v1/nodes: dial tcp 172.24.0.1:443: i/o timeout
Failed to get peers
```

주로 애플리케이션과 연계되는 외부 DB 서비스 또는 내부 Pod 과의 통신이 실패했을 수 있으며, 정상적인 노드 작업, 네트워크 PM 작업 등으로 인해 restart 가 발생될 수 있습니다. 이전 Pod 로그로 확인이 안되는 경우 describe 명령어를 통해 Last Stated: 확인이 필요합니다.

```
$ kubectl describe po <pod-name>
    Last State:     Terminated
      Reason:       OOMKilled
      Exit Code:    137
```

OOM (OutOfMemory)로 인해 종료된 경우, 서비스 가용성을 고려한 Pod resources 를 설정해야 합니다.

```
$ kubectl describe po <pod-name>
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Warning  Unhealthy  48m (x6 over 49m)    kubelet, node2     Liveness probe failed:
localhost:5432 - no response
```

Liveness probe 가 실패한 경우라면 부하 상황 및 Connection lease 등을 고려한 Liveness probe threshold 를 설정해야 합니다.

3.12 Pod 자체가 조회되지 않는 경우

Deployment 등의 워크로드를 생성 했지만, Pod 가 조회되지 않는 경우 입니다.

이런 경우 Event 객체 조회가 먼저 필요합니다.

```
$ kubectl get ev
LAST SEEN   TYPE      REASON          OBJECT                          MESSAGE
3m1s        Warning   FailedCreate    replicaset/nginx-847f85d779     Error creating:
pods "nginx-847f85d779-j484f" is forbidden: exceeded quota: resourcequota, requested:
requests.cpu=100m,requests.memory=2Gi, used: requests.cpu=4,requests.memory=128Mi,
limited: requests.cpu=2,requests.memory=1Gi
```

위의 경우는 Namespace 에 ResourceQuota 를 설정한 경우 해당 Pod 이 이를 초과하였거나 LimitRange 를 벗어난 경우 입니다. ResourceQuota/LimitRange 를 조정하거나 Pod resources 를 조정해야 합니다.

```
$ kubectl get ev
LAST SEEN   TYPE      REASON          OBJECT                    MESSAGE
3s          Warning   FailedCreate    replicaset/nginx-56b5449445   Error creating:
pods "nginx-56b5449445-" is forbidden: error looking up service account default/my-sa:
serviceaccount "my-sa" not found
```

Pod 구동에 필요한 resource 를 만들지 않은 경우 입니다. 이런 경우 Pod specification 을 조회하여 필요한 resource 를 만들어야 합니다.

```
$ kubectl get po -n kube-system
NAME                                READY   STATUS             RESTARTS   AGE
kube-controller-manager-master 0/1  CrashLoopBackOff   6          10m
```

kube-controller-manager 가 정상이 아닌 경우에 kube-controller-manager 의 log 를 확인하여 조치해야 합니다.

4. Service 상태 점검

Pod 구동 상태에 문제가 없다면 Service 에 대한 상태를 점검해야 하는데, 우선은 Pod 로의 통신에 문제가 없는지를 먼저 확인해야 합니다.

4.1 Pod 통신 확인

Kubernetes 노드에서는 <Pod IP>:<Port> 로 직접 통신 요청을 보내어 이를 확인 할 수 있습니다.

```
$ kubectl get po -owide
NAME                     READY   STATUS    RESTARTS   AGE     IP
nginx-58cd5b85f7-sn5n9   1/1     Running   0          2m15s   10.36.0.1
$ curl 10.36.0.1:80
<title>Welcome to nginx!</title>
```

만약 Kubernetes 노드가 아니라면 kubectl 의 port-forward 를 통해 아래와 같이 통신 확인이 가능합니다. (TCP 만 가능)

```
$ kubectl port-forward <pod-name> <local-port>:<pod-port>
Forwarding from 127.0.0.1:8888 -> 80
Forwarding from [::1]:8888 -> 80
$ curl localhost:8888
<title>Welcome to nginx!</title>
```

이렇게 Pod 통신이 확인 된 이후에는 Service 통신을 점검하면 됩니다.

4.2 Pod 통신 실패

하지만, Pod 과의 통신이 안되는 경우라면 우선 Pod 이 해당 Port 를 Listen 하고 있는지를 확인해야 하는데, Pod 내부에서 먼저 netstat, ss, curl 등 명령어 사용이 가능한 경우 내부에서 통신 여부를 먼저 체크해 볼 수 있습니다.

정상적으로 Port Listen 을 하는 경우라면 Network Policy 에 의해 통신이 차단된 경우를 생각해 볼 수 있습니다. Network Policy 가 Accept 되어있는 다른 Pod 에서 통신을 확인해 봐야 합니다.

```
$ kubectl get netpol
NAME           POD-SELECTOR   AGE
default-deny   <none>         1d
```

또는 CNI(Container Network Interface) Plugin 이 정상동작 하고 있는지 확인 후 이를 조치해야 합니다.

```
$ kubectl get po -n kube-system | grep weave
NAME                READY   STATUS             RESTARTS   AGE
weave-net-97gcr     1/2     CrashLoopBackOff   8          16m
```

4.3 Service 통신 확인

Pod 와 마찬가지로 Kubernetes 노드인 경우 <Service IP>:<Port>로 직접 통신 확인을 할 수 있습니다.

```
$ kubectl get svc
NAME          TYPE          CLUSTER-IP        EXTERNAL-IP    PORT(S)    AGE
nginx         ClusterIP     10.100.215.120    <none>         80/TCP     54m
$ curl 10.100.215.120:80
<title>Welcome to nginx!</title>
```

Kubernetes 노드가 아닌 경우는 kubectl port-forward 를 통해 통신을 확인 할 수 있습니다. (TCP 만 가능)

```
$ kubectl port-forward service/<service-name> <local-port>:<pod-port>
Forwarding from 127.0.0.1:8888 -> 80
Forwarding from [::1]:8888 -> 80
$ curl localhost:8888
<title>Welcome to nginx!</title>
```

여기까지 Service 통신이 확인 된 경우에는 Ingress 통신 점검으로 넘어가면 됩니다.

4.4 Service 통신 실패

Service 로의 통신이 실패한 경우 다음과 같은 점검 항목을 수행해볼 수 있습니다.

```
$ kubectl describe svc <service-name>
Selector:          name=nginx
Port:              http  80/TCP
TargetPort:        8080/TCP
Endpoints:         <none>
```

Service 를 describe 했을 때 Endpoints 여부를 확인해야 하는데, <none>인 경우 Service selector 와 Pod label 이 mismatch 된 경우가 있을 수 있으며, 이를 일치하도록 수정해야 합니다.

Endpoint 가 있는데 통신이 안되는 경우입니다.

```
$ kubectl describe svc <service-name>
Selector:          name=nginx
Port:              http  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.36.0.1:8080
```

Service 의 targetPort 와 Pod 의 containerPort 가 mismatch 된 경우에는 Port 가 일치하도록 수정해야 합니다. 그런 경우가 아니라면 Network Policy 에 의해 통신이 안되는 경우를 확인해야 하고, kube-proxy 가 비정상인지를 점검한 후에 정상 조치해야 합니다.

5. Ingress 상태 점검

Service 까지 점검이 끝났다면 Ingress 와의 통신을 통해 Kubernetes 와 관련된 워크로드 점검을 마무리 할 수 있습니다.

5.1 Ingress 통신 확인

정상적인 경우라면 describe ing 시 backend 에 Pod IP 정보가 있어야 합니다.

```
$ kubectl describe ing <ingress-name>
Name:             nginx
Address:          172.28.128.11
Rules:
  Host              Path   Backends
  ----              ----   --------
  nginx.example.io
                          nginx:80 (10.36.0.1:80,10.44.0.2:80)
```

Ingress 에 설정된 도메인을 직접 웹브라우저를 통해 접속해서 확인해도 되고, 서버에서 curl 명령어로도 확인이 가능합니다. 해당 도메인이 DNS 서버에 등록되지 않은 경우라면 아래와 같이 Header 에 도메인을 추가하여 확인할 수도 있습니다.

```
$ curl -H 'Host: <host-domain>' http://<ingress-controller-external-ip>
<title>Welcome to nginx!</title>
```

여기까지 통신이 되었다면 Kubernetes 클러스터 내부적으로는 이슈가 없는 것으로 확인가능 합니다.

5.2 Ingress 통신 실패

Ingress 로의 통신이 실패되는 경우에 대해 알아보겠습니다.

먼저, Ingress backend 에 Pod IP 정보가 없는 경우 입니다.

```
$ kubectl describe ing <ingress-name>
Name:             nginx
Address:          172.28.128.11
Rules:
  Host            Path  Backends
  ----            ----  --------
  nginx.example.io
                        nginx:80 (<none>)
```

이런 경우는 Ingress Specification 의 backend 에 serviceName 과 servicePort 를 실제 Service resource 와 제대로 일치하는지 확인 후 조치해야 합니다.

여기부터는 Ingress backend 정보가 있는 통신이 안되는 경우들 입니다.

```
$ curl -H 'Host: nginx.example.io' http://172.28.128.11
<title>504 Gateway Time-out</title>
```

이 경우는 Network Policy 에 의해 통신이 차단되었는지 확인해야 합니다.

```
$ curl -H 'Host: nginx.example.io' http://172.28.128.11
<title>503 Service Temporarily Unavailable</title>
```

애플리케이션 서비스가 응답을 안하고 있는 경우라서 Pod 상태가 정상인지 확인해야 합니다.

```
$ curl -H 'Host: nginx.example.io' http://172.28.128.11
<title>404 Not Found</title>
```

Ingress backend 에 설정된 path 가 실제 애플리케이션이 서비스하고 있는 context path 가 맞는지 확인해야 합니다.

```
$ curl -H 'Host: nginx.example.io' http://172.28.128.11
curl: (7) Failed connect to 172.28.128.11:80; Connection timed out
```

Ingress Controller 와 방화벽이 열려있는지 확인해야 합니다.

```
$ curl -H 'Host: nginx.example.io' http://172.28.128.11
curl: (7) Failed connect to 172.28.128.11:80; Connection refused
```

Ingress Controller 가 Port Listen 을 하지 않는 경우에 발생할 수 있습니다. Ingress Controller 가 정상적으로 구동되고 있는지 확인해야 합니다.

추가로 외부에서 방화벽에 의해 차단된 경우라면 아래 명령어를 통해 통신 여부를 확인한 후 <ingress-controller-external-ip>:80,443 과 방화벽을 열어야 합니다.

```
$ telnet <ingress-controller-external-ip> 80
또는
$ </dev/tcp/<ingress-controller-external-ip>/80
```

외부에서 도메인이 등록되지 않은 경우에는 "Could not resolve host" 에러가 발생할 수 있습니다. 다음 명령어를 통해 도메인을 resolve 할 수 있는지 확인 후 DNS 서버에 해당 도메인을 등록하거나 개별 도메인을 hosts 파일에 등록하여 사용해야 합니다.

```
$ nslookup <host-domain>
또는
$ getent hosts <host-domain>
```

6. Quick 점검 가이드

지금까지 가이드 내용을 토대로 Quick 하게 점검하는 순서입니다.

1. Pod Running & Pod Ready 확인

```
$ kubectl get po
NAME                     READY   STATUS    RESTARTS   AGE
nginx-58cd5b85f7-84xlr   1/1     Running   0          31s
```

2. Pod 통신 확인

```
$ curl <Pod IP>:<port>
```

3. Serivce 통신 확인

```
$ curl <Service IP>:<port>
```

4. Ingress 통신 확인

```
$ curl <Host-Domain>:<port>
```