



GitLab StatefulSet 設定



我修改gitlab上repo的statefulset設定，讓我指定的2個harbor上的image，運行在同一個pod，我應該怎麼做，要注意什麼



在 GitLab 上修改 StatefulSet 的設定，讓兩個 Harbor 上的 image 運行在同一個 Pod，以下是具體步驟和需要注意的事項：

步驟

1. **確認 StatefulSet 定義文件** 在 GitLab 上找到相關的 StatefulSet YAML 文件，這通常位於專案的 `k8s` 或 `deployment` 資料夾內。
2. **修改 StatefulSet 的 `spec.template.spec.containers` 部分** 將你的兩個 Harbor image 定義為兩個 container，並配置在同一個 `pod` 中。例如：

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: example-statefulset
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  serviceName: "example-service"
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: container1
          image: harbor.example.com/repo1/image1:tag1
          ports:
            - containerPort: 8080
        - name: container2
          image: harbor.example.com/repo2/image2:tag2
          ports:
            - containerPort: 9090
```

3. **配置通信方式** 如果兩個 container 需要相互通信，建議使用以下方法：
 - ****共享文件系統****：可以通過 `emptyDir` 或 `volume` 的方式共享數據。
 - ****localhost 通信****：容器之間可以直接使用 `localhost` 和相應的 port。
4. **檢查資源需求** 確保兩個容器的資源需求（如 CPU 和內存）合理配置，避免因為資源不足導致 Pod 無法啟動。





```
resources:
  requests:
    memory: "256Mi"
    cpu: "500m"
  limits:
    memory: "512Mi"
    cpu: "1"
```

5. **更新 StatefulSet** 使用 `kubectl` 或 GitLab CI/CD 部署流程將修改後的 StatefulSet 應用到 Kubernetes 集群：

```
kubectl apply -f statefulset.yaml
```

6. 驗證

- 使用 `kubectl get pods` 確認 Pod 是否成功啟動。
- 使用 `kubectl logs` 檢查兩個容器的日誌。
- 測試兩個容器的通信和服務是否正常。

注意事項

1. **Harbor 的 Registry 配置** 確保你的 Kubernetes 節點能夠正確訪問 Harbor registry：

- 確保 Image URL 是正確的。
- 如果需要身份驗證，配置 Kubernetes secret，並在 StatefulSet 中引用：

```
kubectl create secret docker-registry harbor-secret --docker-server=harbor.example.com --docker-username=your-username
```

在 StatefulSet 中指定：

```
imagePullSecrets:
- name: harbor-secret
```

2. **Networking** 如果兩個容器需要網路通信，檢查是否需要開放相關的端口，並避免端口衝突。
3. **持久化存儲** 如果需要共享數據，確保使用正確的 Persistent Volume 或 `emptyDir` 配置，避免數據丟失。
4. **資源調整** 如果容器資源需求過高，可能導致 Pod 無法調度到節點上，應根據實際需求調整 `resources` 配置。
5. **調試** 如果出現問題，可以通過以下命令進行調試：
- 檢查 Pod 狀態：`kubectl describe pod <pod-name>`
 - 查看容器日誌：`kubectl logs <pod-name> -c <container-name>`
6. **滾動更新** 若在生產環境中進行變更，確保 StatefulSet 配置了正確的滾動更新策略，以避免服務中斷。



如果有更多細節需求或遇到具體問題，隨時可以提供進一步協助！



畫布



</> Statefulset Harbor Config

已生成了一份配置文件，展示了如何將兩個 Harbor 的映像部署在同一個 Pod 中，並通過共享文件系統（`emptyDir`）實現數據共享。如果需要進一步修改或優化，請告訴我！



我原先pod裡有2個image，怎麼增加第3個image



</> Statefulset Harbor Config

1
2
3
4
5
6
7
8
9
10
11
12
13
14

進行了 1 次編輯

我已經將第3個 image 添加到您的 Pod 中，並命名為 `container3`，分配了相應的端口 `7070` 並共享相同的數據卷。如果還有其他需求或需要進一步修改，請隨時告訴我！



不要用畫布



好的！以下是如何在現有的 Pod 中添加第 3 個容器的配置方法：

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: example-statefulset
spec:
  replicas: 1
```





```
selector:
  matchLabels:
    app: my-app
serviceName: "example-service"
template:
  metadata:
    labels:
      app: my-app
  spec:
    containers:
      - name: container1
        image: harbor.example.com/repo1/image1:tag1
        ports:
          - containerPort: 8080
        volumeMounts:
          - name: shared-data
            mountPath: /shared
      - name: container2
        image: harbor.example.com/repo2/image2:tag2
        ports:
          - containerPort: 9090
        volumeMounts:
          - name: shared-data
            mountPath: /shared
      - name: container3
        image: harbor.example.com/repo3/image3:tag3
        ports:
          - containerPort: 7070
        volumeMounts:
          - name: shared-data
            mountPath: /shared
    volumes:
      - name: shared-data
        emptyDir: {}
    imagePullSecrets:
      - name: harbor-secret
```

新增的容器名为 `container3`，对应的镜像为 `harbor.example.com/repo3/image3:tag3`，暴露端口为 `7070`，并共享数据卷 `/shared`。

如果需要进一步调整或添加其他配置，请告诉我！



為什麼我的k8s使用了3個image的statefulset檔，結果還是只有2個image佈署上去？



如果 Kubernetes 使用了包含 3 个容器镜像的 StatefulSet 配置文件，但结果只有 2 个容器被部署，可能的原因包括以下几点：

1. YAML 配置问题





- **未保存更改的 YAML 文件：**确保应用的 StatefulSet YAML 文件已经包含 3 个容器配置，并且是最新的。
- **格式错误：**检查 YAML 文件是否存在缩进错误或格式问题，可能导致 Kubernetes 忽略某些容器的配置。
- **没有更新资源：**如果没有正确应用更改，Kubernetes 可能仍使用旧的配置。使用以下命令强制更新：

```
kubectl apply -f statefulset.yaml
```

2. 更新策略问题

- 如果 StatefulSet 的更新策略为 `OnDelete`，更改的配置不会自动应用到现有 Pod。需要手动删除旧的 Pod，让 Kubernetes 根据新配置重新创建。

```
kubectl delete pod <pod-name>
```

3. 资源限制

- **Node 资源不足：**如果 Kubernetes 集群的节点资源不足（如 CPU 或内存），可能会限制新的容器启动。检查节点资源情况：

```
kubectl describe node <node-name>
```

- **Pod 调度问题：**如果 Pod 无法满足资源请求（`requests`）或限制（`limits`），则无法启动全部容器。

4. 镜像拉取失败

- 确保第三个镜像可以正常拉取。如果拉取失败，可能会导致该容器未启动：

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

查看事件日志，是否有镜像拉取错误（如 `ImagePullBackOff`）。

- 验证镜像地址和凭据是否正确：

```
kubectl get secret harbor-secret -o yaml
```

5. 调度策略或限制

- 如果设置了特定的调度规则（如 Node Selector、Taints/Tolerations），检查是否导致第三个容器未能正常调度到节点。
- 查看 Pod 的详细信息：

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

6. 控制器问题

- StatefulSet Controller 是否正常工作，确保 Kubernetes 的控制平面没有报错：



```
kubectl get events
```

解决步骤

1. 检查 StatefulSet 配置是否正确，特别是 `spec.template.spec.containers` 是否包含 3 个容器。
2. 强制更新 StatefulSet：

```
kubectl apply -f statefulset.yaml
```

3. 检查 Pod 的状态：

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

4. 如果问题仍未解决，删除现有 Pod 强制重新创建：

```
kubectl delete pod <pod-name>
```

通过这些步骤，您应该可以找到问题所在。如果问题持续，您可以将 Pod 的描述信息或事件日志分享给我，我可以进一步分析问题原因。



****格式错误、资源限制、镜像拉取失败、调度策略或限制**



针对您的问题描述，这里对 **格式错误**、**资源限制**、**镜像拉取失败** 和 **调度策略或限制** 进行详细说明和排查建议：

1. 格式错误

- **问题描述：**YAML 文件中格式问题（如缩进、语法错误）会导致 Kubernetes 忽略部分配置。
- **排查方法：**

1. 使用 YAML 校验工具检查格式是否正确：

```
yamllint statefulset.yaml
```

2. 确认 `containers` 字段是否正确定义，缩进是否一致。
3. 确认第三个容器的 `name` 和 `image` 字段没有拼写错误，且不重复。

2. 资源限制





- **问题描述：** Kubernetes 集群节点的资源不足可能导致部分容器无法启动。
- **排查方法：**

1. 检查 Pod 的状态，确保没有 `Pending` 或 `Evicted` 状态：

```
kubectl get pods
```

2. 查看 Pod 描述，检查资源是否超出限制：

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

重点查看事件日志中是否有类似 `Insufficient CPU` 或 `Insufficient Memory` 的报错。

3. 确认 StatefulSet 配置的 `resources` 部分是否合理。例如：

```
resources:
  requests:
    memory: "256Mi"
    cpu: "500m"
  limits:
    memory: "512Mi"
    cpu: "1"
```

- **解决方案：**
 - 减少资源请求或限制，确保容器能够被调度。
 - 增加集群节点的资源（CPU、内存）。

3. 镜像拉取失败

- **问题描述：** 第三个容器的镜像可能因地址错误、网络问题或凭据问题无法拉取。
- **排查方法：**

1. 检查 Pod 的状态，是否有 `ImagePullBackOff` 或 `ErrImagePull` 错误：

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

2. 确认第三个镜像的 URL 是否正确（如 `harbor.example.com/repo3/image3:tag3`）。
3. 如果需要身份验证，检查 `imagePullSecrets` 是否正确配置：

```
kubectl get secret harbor-secret -o yaml
```

- **解决方案：**
 - 确保镜像仓库地址和凭据正确。
 - 测试手动拉取镜像是否成功：



```
docker pull harbor.example.com/repo3/image3:tag3
```

- 如果镜像无法拉取，检查网络连通性或仓库设置。

4. 调度策略或限制

- **问题描述：** 调度策略可能限制了 Pod 被分配到合适的节点上，导致部分容器未运行。
- **排查方法：**
 1. 检查是否设置了 Node Selector、Taints/Tolerations 或 Affinity：

```
spec:
  nodeSelector:
    disktype: ssd
  tolerations:
  - key: "key1"
    operator: "Exists"
    effect: "NoSchedule"
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
```

2. 查看节点状态，确保节点可用：

```
kubect1 get nodes
```

重点检查节点的 `STATUS` 是否为 `Ready`。

- **解决方案：**
 - 修改 StatefulSet 配置，移除过于严格的调度限制。
 - 确保节点处于 `Ready` 状态，且资源充足。

总结

逐步检查以下几方面：

1. **格式问题：** 使用 YAML 校验工具检查。
2. **资源限制：** 确保集群节点有足够资源，调整 StatefulSet 中的资源请求。
3. **镜像拉取：** 确认镜像地址和凭据正确，检查 Pod 状态日志中的错误信息。
4. **调度限制：** 查看 Node 状态和调度策略，确保没有调度冲突。



如果问题仍然存在，请分享具体的 Pod 状态描述或事件日志，我将帮助您进一步分析和解决！



以上内容幫我生成word檔



已生成一份包含详细排查说明的 Word 文档，您可以通过以下链接下载：

[Kubernetes_StatefulSet_Troubleshooting.docx](#)

如果还有其他需要，请随时告诉我！ [~]



以上内容幫我生成.md檔



已生成一份 Markdown 文件，包含详细的排查说明。您可以通过以下链接下载：

[Kubernetes_StatefulSet_Troubleshooting.md](#)

如果还有其他需求，请随时告诉我！ [~]