# 127Kubernetes 系列（一二一）使用replace和force命令在Kubernetes中手动调度pod

当我们使用 Kubernetes 时，我们通常会接受 Kubernetes 架构所授予的大部分流程。将 Pod 放置在合适的节点中就是其中之一。我们要么运行命令式 `kubectl run` 命令，要么简单地提供部署或副本集 yaml 文件，然后等待 Pod 被放置在集群内的节点中。然而，了解事物背后的工作原理至关重要。事实上，了解如何在集群内手动调度 Pod 是非常有必要的，以便将来需要时可以进行手动调度。

## Kube-scheduler

Kube-scheduler 是将 pod 调度到集群内节点的组件。在此过程中，kube-scheduler 会考虑参数，例如每个节点上的可用资源、pod 所需的资源、亲和性和反亲和性规则、节点污染和容忍、pod 优先级和抢占策略，以及许多其他因素。根据这些参数，kube-scheduler 决定集群中的哪个节点最适合 pod，并将其分配给该节点。

当我们为 pod 准备 yaml 清单文件时，我们通常会省略一个名为 nodeName 的字段。在 Pod 创建过程中，kube-scheduler 会扫描所有 Pod 并检查该字段。如果该字段为空，kube-scheduler 会将 pod 分配给合适的节点，并且字段名称会获取值。换句话说，如果我们不添加 nodeName 字段，Kubernetes 会在 kube-scheduler 创建并调度 pod 后为我们添加它。

## 如果没有 kube-scheduler 怎么办

当没有 kube-scheduler 工作或者 kube-scheduler 无法正常工作时会发生什么？让我们看看实际效果。

为了查看 kube-scheduler 是否工作，我们可以简单地检查 kube-system 命名空间中的 kube-scheduler pod。

```
kubectl get pods -n kube-system
```

```
controlplane $ kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
calico-kube-controllers-5f94594857-smxtv   1/1     Running   3          3d1h
canal-f6j2b                            2/2     Running   0          5m52s
canal-sxfpf                            2/2     Running   0          5m52s
coredns-68dc769db8-6cwk7               1/1     Running   0          3d1h
coredns-68dc769db8-9h8gn               1/1     Running   0          3d1h
etcd-controlplane                      1/1     Running   0          3d1h
kube-apiserver-controlplane            1/1     Running   2          3d1h
kube-controller-manager-controlplane   1/1     Running   2          3d1h
kube-proxy-ldwqn                       1/1     Running   0          3d1h
kube-proxy-pftdf                       1/1     Running   0          3d1h
kube-scheduler-controlplane            1/1     Running   2          3d1h
```



如您所见，kube-scheduler 正在工作。让我们删除 kube-scheduler pod 清单文件并再次检查。

```
mv /etc/kubernetes/manifests/kube-scheduler.yaml kube-scheduler
kubectl get pods -n kube-system
```

```
controlplane $ kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
calico-kube-controllers-5f94594857-smxtv   1/1     Running   3          3d1h
canal-f6j2b                            2/2     Running   0          6m40s
canal-sxfpf                            2/2     Running   0          6m40s
coredns-68dc769db8-6cwk7               1/1     Running   0          3d1h
coredns-68dc769db8-9h8gn               1/1     Running   0          3d1h
etcd-controlplane                      1/1     Running   0          3d1h
kube-apiserver-controlplane            1/1     Running   2          3d1h
kube-controller-manager-controlplane   1/1     Running   2          3d1h
kube-proxy-ldwqn                       1/1     Running   0          3d1h
kube-proxy-pftdf                       1/1     Running   0          3d1h
```



现在，没有 kube-scheduler 来调度 Pod。让我们用一个简单的 nginx 镜像创建一个测试 Pod，看看会发生什么。

```
kubectl run testpod --image=nginx
kubectl get pods -o wide
```

```
controlplane $ kubectl run testpod --image=nginx
pod/testpod created
controlplane $ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
testpod   0/1     Pending   0          7s    <none>   <none>   <none>           <none>
```



如果 kube-scheduler 不工作，pod 就会陷入挂起状态。

## 人工干预

当 kube-scheduler 不工作时，手动干预可以将 pod 放置在节点中。怎么做？其实一开始我就已经暗示过了。我们所要做的就是在清单文件中添加一个nodeName字段并手动添加合适的节点。

```
Editor  Tab 1  Tab 2  +
apiVersion: v1
kind: Pod
metadata:
  name: manualpod
  labels:
    name: nginx
spec:
  nodeName: node01
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```



如您所见，我们添加了node01作为nodeName并保存了yaml文件。现在，让我们运行 kubectl apply 命令来创建 pod，看看会发生什么。

```
kubectl apply -f test2.yaml
kubectl get pods -o wide
```

```
controlplane $ kubectl apply -f test2.yaml
pod/manualpod created
controlplane $ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
manualpod   1/1     Running   0          94s   192.168.1.3  node01   <none>           <none>
testpod     0/1     Pending   0          29m   <none>       <none>   <none>           <none>
controlplane $
```

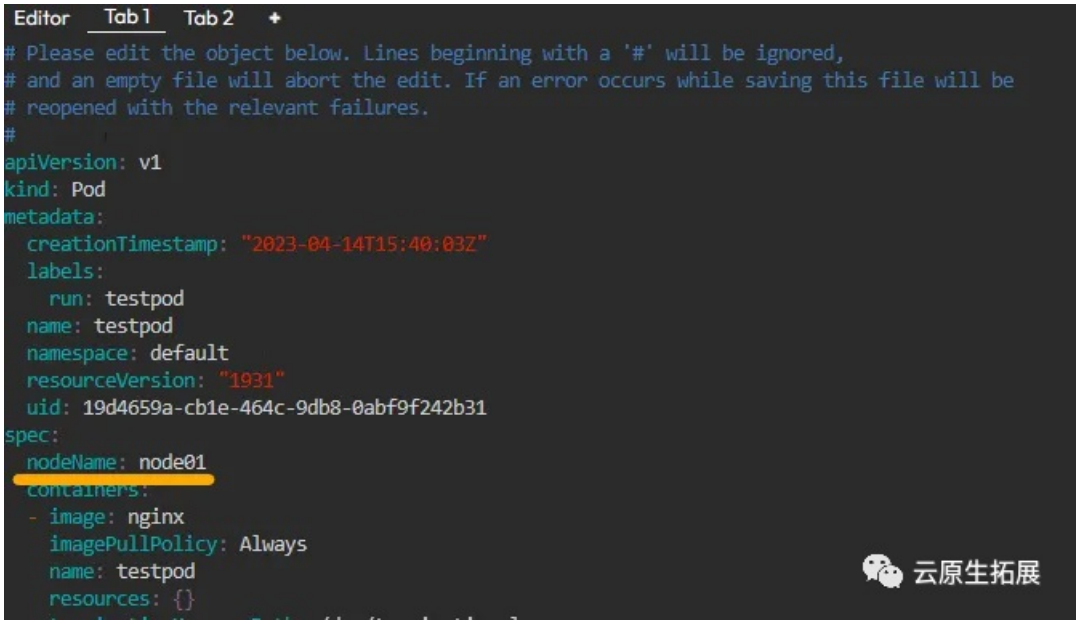
好吧，我们成功地将 pod 调度到 node01，而无需使用 kube-scheduler。但我们仍然有第一个 pod 处于待处理状态！

首先，我们始终可以删除 Pod，然后通过添加 NodeName 来编辑清单文件，然后重新创建 Pod。如果我们不想删除 pod，那么我们可以通过添加 nodeName 字段来编辑清单文件，然后使用 replace 和 force 命令。让我们这样做吧。

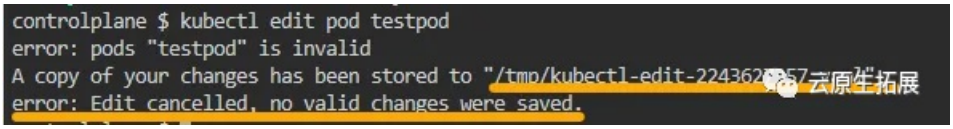
1. 首先，我们运行以下命令来访问 pod 清单文件（该清单文件会比通常的清单文件有更多字段，但这是正常的，请继续）。

```
kubectl edit pod testpod
```

2.在spec下添加nodeName字段并输入节点。



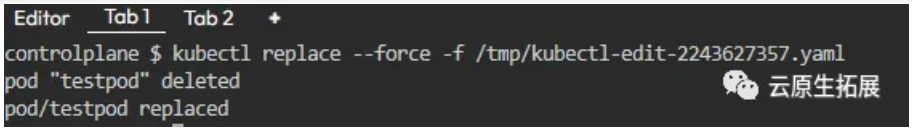

3. 保存并退出清单文件 ( :wq )。




4. 您将收到一条错误消息，指出编辑已取消，因为您无法正常编辑 Pod 清单文件。但是，输出还将在 tmp 目录下提供一个新文件路径，其中包含已编辑的清单文件。可以将其视为使用新名称和位置保存为清单文件。复制该文件路径。

5. 运行以下命令替换 pod 清单文件并添加刚刚复制的文件位置。

```
kubectl replace --force -f /tmp/path-to-new-manifest-file-copied-above
```




6. 让我们获取 pod，看看是否可以将 pod 分配给节点

```

```

```
kubectl get pods -o wide
```

```
controlplane $ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
manualpod   1/1     Running   0          21m   192.168.1.3   node01   <none>           <none>
testpod     1/1     Running   0          37s   192.168.1.4   node01   <none>           <none>
```



我们做到了。两个 Pod 均通过手动干预分配给节点。

> 重要提示：如果您使用清单文件创建了 pod，则不需要输入 kubectl edit 命令，可以直接使用编辑器（例如 vim）编辑清单文件。但您仍然需要运行 kubectl replace --force 命令来进行更改。