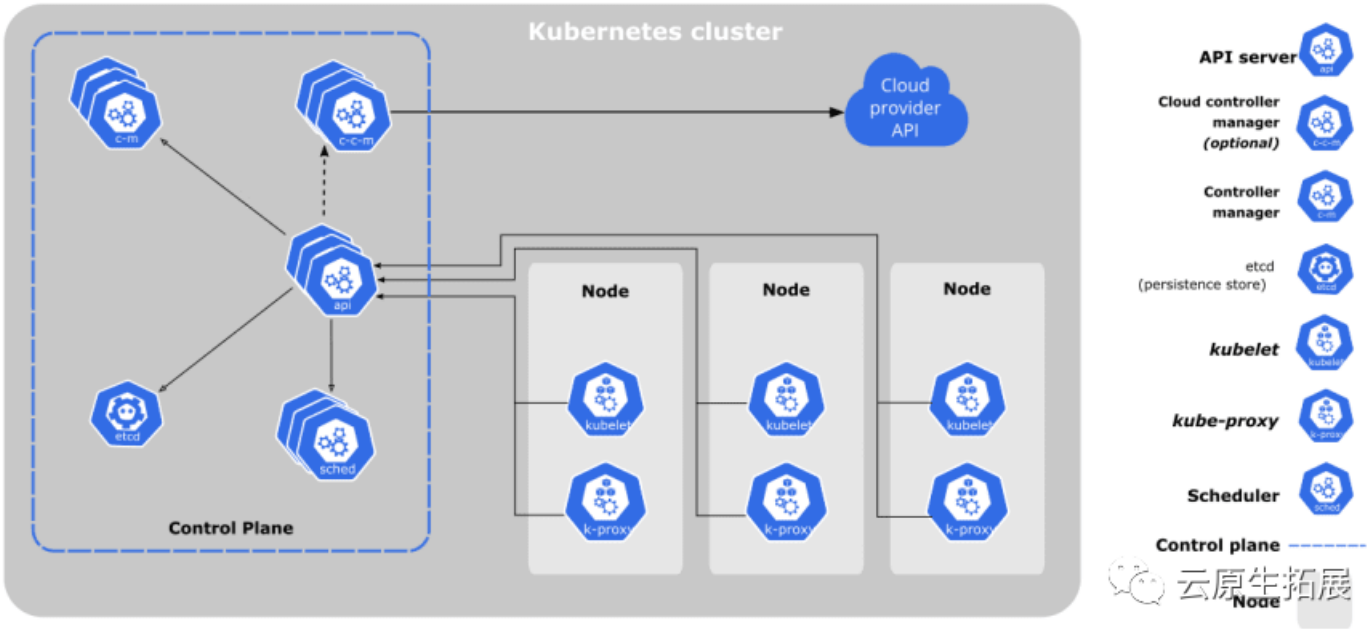
51Kubernetes 系列（四十八）冥王星 pluto - 检测 Kubernetes 过时 API

Kubernetes 系列（四十八）冥王星 pluto - 检测 Kubernetes 过时 API

Kubernetes API 一直在变化，随着这些变化，API 的一些部分被弃用并最终被移除。为了能够保持与最新的Kubernetes集群版本一致，我们必须识别已弃用的api并更新它们。在拥有数百个资源的大型集群中，这可能会变得单调乏味，但像 pluto 这样的工具可以提供帮助。

API 弃用意味着什么?

管理所有Kubernetes资源生命周期的操作是由Kubernetes API服务通过RESTful API端点提供的。换句话说，Kubernetes API是Kubernetes控制平面的前端。



资源 API 与 URI 相关联，比如

- `/apis/GROUP/VERSION/*` 用于集群类型的资源
- `/apis/GROUP/VERSION/namespaces/NAMESPACE/*` 用于命名空间管辖的资源

某些资源也被分组到核心组(或遗留组)中。这些可以通过特殊的API端点 `/api/{version}` 获得(参见 API组)。例如，Pod 是核心组的一部分。列出给定名称空间 `my-namespace` 中所有pod的请求映射到下面的HTTP GET请求。

```
GET /api/v1/namespaces/my-namespace/pods
```

(可参考 API documentation for the Pod resource .)

Kubernetes API弃用的问题

Kubernetes 指定了一个弃用策略，该策略定义了API的某些部分被弃用时的意义。本质上，弃用意味着Kubernetes API服务器的相关端点被标记为删除，然后后续将被删除。由于API Server 控制资源生命周期，因此使用已删除 API 版本的资源将阻止该资源的部署。因此，如果我们不能更新我们的资源API版本，我们要么会被一个过时的Kubernetes版本绊住；或者，更新到新的Kubernetes版本将阻止某些资源的部署。两者都是不受欢迎的状态，因此我们要么

1. 继续使用不稳定的 Kubernetes 版本，或者

2. 我们在 Kubernetes 中无法正常完成服务部署

使用已删除的 API 版本部署资源

为了更清楚地了解情况，让我们看看第二个问题，看看如果我们尝试使用已删除的API版本部署资源，Kubernetes会如何响应。为此，我们使用 `[k3d](https://k3d.io/)` 启动了一个本地 Kubernetes 集群

```
$ k3d cluster create
INFO[0000] Prep: Network
INFO[0004] Created network 'k3d-k3s-default'
INFO[0004] Created image volume k3d-k3s-default-images
INFO[0004] Starting new tools node...
INFO[0005] Pulling image 'ghcr.io/k3d-io/k3d-tools:5.4.3'
INFO[0007] Starting Node 'k3d-k3s-default-tools'
INFO[0007] Creating node 'k3d-k3s-default-server-0'
INFO[0009] Pulling image 'docker.io/rancher/k3s:v1.23.6-k3s1'
INFO[0027] Creating LoadBalancer 'k3d-k3s-default-serverlb'
INFO[0028] Pulling image 'ghcr.io/k3d-io/k3d-proxy:5.4.3'
INFO[0030] Using the k3d-tools node to gather environment information
INFO[0030] Starting new tools node...
INFO[0030] Starting Node 'k3d-k3s-default-tools'
INFO[0033] Starting cluster 'k3s-default'
INFO[0033] Starting servers...
INFO[0033] Starting Node 'k3d-k3s-default-server-0'
INFO[0040] All agents already running.
INFO[0040] Starting helpers...
INFO[0040] Starting Node 'k3d-k3s-default-serverlb'
INFO[0049] Injecting records for hostAliases (incl. host.k3d.internal) and for 3 network members into Core
INFO[0051] Cluster 'k3s-default' created successfully!
INFO[0051] You can now use it like this:
kubectl cluster-info
```

现在，我们通过执行 `kubectl version` 来查看Kubernetes API Server 正在运行哪个版本

```
$ kubectl version
WARNING: This version information is deprecated and will be replaced with the output from kubectl version
Client Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.2", GitCommit:"f66044f4361b9f1f96f00
Kustomize Version: v4.5.4
Server Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.6+k3s1", GitCommit:"418c3fa858b69b12
```

从 Server Version：输出中可以看到，我们的k3d集群正在运行Kubernetes v1.23。

通过查看API弃用指南(https://kubernetes.io/docs/reference/using-api/deprecation-guide/#v1-22)，我们可以看到 `In`

`gress` API的两个版本，extensions/v1beta1和networking.k8s.io/v1beta1 在Kubernetes v1.22中被删除。因此，让我们尝试使用该API版本部署Ingress资源，看看会发生什么。下面我们有一个示例清单，我无耻地从Kubernetes官方文档的* The Ingress Resource *部分窃取，并将其放在文件 `ingress-pre-1.22.yaml` 中。

```
# ingress-pre-1.22.yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

然后让我们尝试将其部署到我们的v1.23 Kubernetes集群中。

```
$ kubectl apply -f ingress-pre-1.22.yaml
error: resource mapping not found for name: "minimal-ingress" namespace: "" from "ingress-pre-1.22.yaml": 
ensure CRDs are installed first
```

如我们所见，API返回一个错误，指示 `networking.k8s.io/v1beta1` 不再包含Ingress类型。如果我们将API版本更改为 *networking.k8s.io/v1* ，

```
# ingress-1.22.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

我们的 `Ingress` 将毫无阻碍地创建：

```
$ kubectl apply -f ingress-1.22.yaml
ingress.networking.k8s.io/minimal-ingress created
```

使用 pluto 检测 API 弃用

在一个更现实的场景中，我们已经将资源部署到集群中，并希望保持其API版本最新，以便我们可以安全地更新集群版本。因此，问题是，我们如何发现已弃用且即将删除的API版本的资源？这个问题的一个答案是查看前面提到的弃用指南，并检查即将发布的Kubernetes更新中将删除哪些资源API版本。但是，需要注意的是，如果我们跳过几个版本，我们也必须对当前和目标Kubernetes版本之间的所有版本重复此检查。

在具有数十种资源类型和版本的大型集群中，这可能会变得乏味且容易出错。幸运的是，有一些像 pluto 这样的工具可以帮助我们发现已弃用和即将删除的资源API版本。

让我们假设将 `Ingress` 资源部署到一个API服务器版本早于v1.22（例如v1.19）的Kubernetes集群。使用非当前Kubernetes API版本创建一个k3d集群是可能的，方法是将 `--image` 选项传递给k3d，为我们所需的Kubernete版本指定一个k3s图像（例如v1.196-k3s1，Dockerhub上提供了完整的镜像列表）：

```
$ k3d cluster delete
INFO[0000] Deleting cluster 'k3s-default'
INFO[0002] Deleting cluster network 'k3d-k3s-default'
INFO[0005] Deleting 2 attached volumes...
WARN[0005] Failed to delete volume 'k3d-k3s-default-images' of cluster 'k3s-default': failed to find volum
INFO[0005] Removing cluster details from default kubeconfig...
INFO[0005] Removing standalone kubeconfig file (if there is one)...
INFO[0005] Successfully deleted cluster k3s-default!
$ k3d cluster create --image rancher/k3s:v1.19.16-k3s1
INFO[0000] Prep: Network
INFO[0003] Created network 'k3d-k3s-default'
INFO[0003] Created image volume k3d-k3s-default-images
INFO[0003] Starting new tools node...
INFO[0003] Starting Node 'k3d-k3s-default-tools'
INFO[0006] Creating node 'k3d-k3s-default-server-0'
INFO[0009] Pulling image 'rancher/k3s:v1.19.16-k3s1'
INFO[0018] Creating LoadBalancer 'k3d-k3s-default-serverlb'
INFO[0018] Using the k3d-tools node to gather environment information
INFO[0018] Starting new tools node...
INFO[0018] Starting Node 'k3d-k3s-default-tools'
INFO[0021] Starting cluster 'k3s-default'
INFO[0021] Starting servers...
INFO[0021] Starting Node 'k3d-k3s-default-server-0'
INFO[0028] All agents already running.
INFO[0028] Starting helpers...
INFO[0028] Starting Node 'k3d-k3s-default-serverlb'
INFO[0037] Injecting records for hostAliases (incl. host.k3d.internal) and for 3 network members into Core
INFO[0039] Cluster 'k3s-default' created successfully!
INFO[0039] You can now use it like this:
kubectl cluster-info
```

我们再次通过 `kubectl version` 确认 API Server 正在运行Kubernetes v1.19 版本

```
$ kubectl version
WARNING: This version information is deprecated and will be replaced with the output from kubectl version
Client Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.2", GitCommit:"f66044f4361b9f1f96f00
Kustomize Version: v4.5.4
Server Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.16+k3s1", GitCommit:"da16869555775cf
WARNING: version difference between client (1.24) and server (1.19) exceeds the supported minor version sk
```

现在，让我们再次应用Ingress和指定不推荐的API版本的清单。

```
$ kubectl apply -f ingress-pre-1.22.yaml
Warning: networking.k8s.io/v1beta1 Ingress is deprecated in v1.19+, unavailable in v1.22+; use networking.
ingress.networking.k8s.io/minimal-ingress created
```

正如我们所看到的，我们已经得到了一个有用的弃用警告，建议使用 `networking.k8s.io/v1` 而不是 `networking.k8s.io`

`/v1beta1`。然而，目前，我们假设我们的资源已经部署到正在运行的集群，我们希望检测这些资源（而不再次应用它们）。

有一个像`[pluto](https://pluto.docs.fairwinds.com/installation/)`这样的工具可以方便地检测不推荐使用的资源API版本。 Pluto可以在多个平台上使用，也可以通过各种软件包管理器使用。我们将使用〔binenv〕(https://github.com/devops-works/binenv)来安装它。

```
$ binenv install pluto
2022-07-21T19:47:18+02:00 WRN version for "pluto" not specified; using "5.8.0"
fetching pluto version 5.8.0 100% |
2022-07-21T19:47:22+02:00 INF "pluto" (5.8.0) installed
```

有几种方法可以将资源清单移交给pluto，以检测不推荐的API版本：例如，通过目录扫描或直接输入。此外，还有一个方便的与Helm的集成，它可以检查我们的版本是否弃用。

目录扫描

使用 **directory scan** 需要让集群知道在哪里可以找到集群的Kubernetes 清单文件。在我们的例子中，这非常简单，因为我们只有两个简单的清单，它们都在当前目录中。以下命令使pluto运行目录扫描并检测我们使用到的不推荐的api版本。

```
$ pluto detect-files --directory . --target-versions k8s=v1.22.0
NAME              KIND      VERSION                  REPLACEMENT           REMOVED   DEPRECATED
minimal-ingress   Ingress   networking.k8s.io/v1beta1   networking.k8s.io/v1   true      true
```

注意，我们可以使用`--target-versions`选项指定目标版本。如果我们以Kubernetes v1.15为目标，pluto将返回空列表，因为我们的CRD API版本在v1.16.0之后才被弃用。

```
$ pluto detect-files --directory . --target-versions k8s=v1.15.0
No output to display
```

目录输入

使用 **direct input** 我们可以直接将资源清单通过管道发送到pluto。如果我们想扫描已经部署到集群的资源（并且查看清单太复杂），这特别有用。由于CRD已经部署，我们可以使用`kubectl get`获取资源清单把它交给 pluto:

```
$ kubectl get ingress minimal-ingress -o yaml | pluto detect -
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/
NAME              KIND      VERSION              REPLACEMENT           REMOVED   DEPRECATED
minimal-ingress   Ingress   extensions/v1beta1   networking.k8s.io/v1   true      true
```

Helm releases

如果我们使用 Helm 部署资源，Pluto还会提供一个`detect-helm`子命令，用于检查我们的发行版中是否存在不推荐的API版本。

总结

Kubernetes API不断发展。为了使我们的集群保持最新状态，我们必须不断关注已弃用和即将删除的资源API版本。由于Kubernetes弃用指南的存在，手动检查是可能的，但它们可能会变得非常乏味和容易出错。像pluto这样的工具允许我们自动化API弃用检查，并简化维护最新资源API版本的工作。

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。