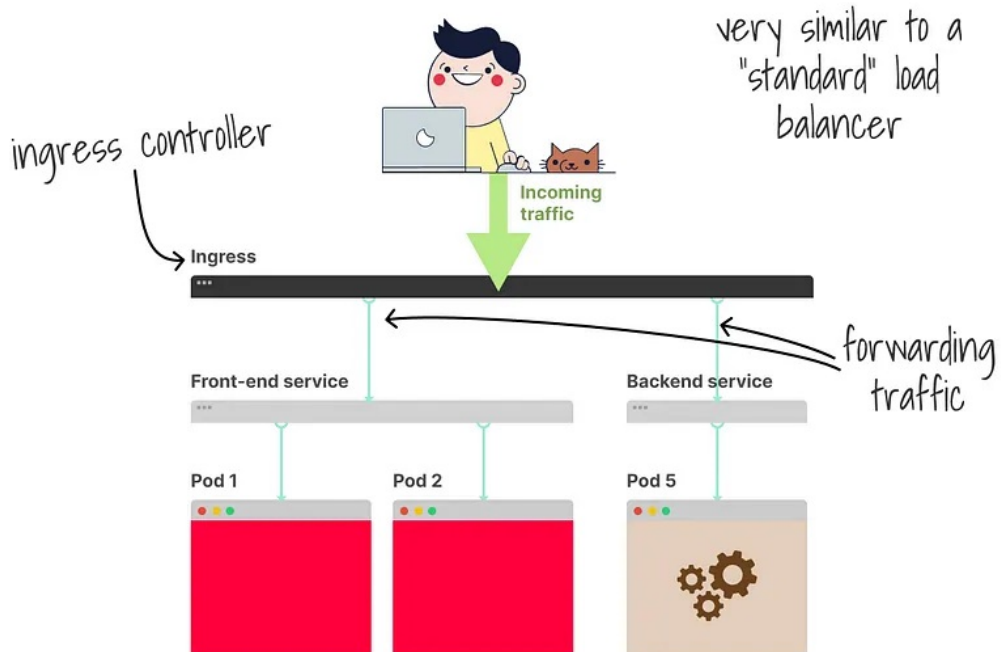


74Kubernetes 系列（六十七）通过在 bash 中构建 Ingress 控制器来学习它是如何工作的

在本文中，您将通过在 bash 中从头开始构建一个控制器来了解 Ingress 控制器在 Kubernetes 中的工作原理。

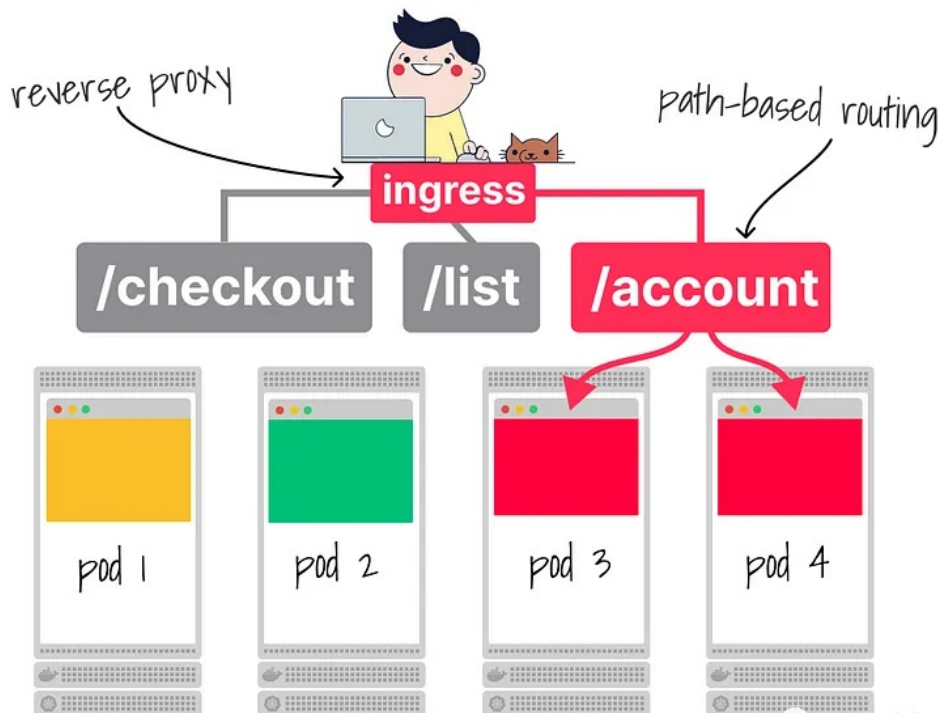
在深入研究代码之前，让我们回顾一下 Ingress 控制器的工作原理。

您可以将 Ingress 控制器视为将流量转发到正确 Pod 的路由器。



云原生拓展

更具体地说，Ingress 控制器是一个反向代理，（主要）在 L7 上工作，允许您根据域名、路径等路由流量。

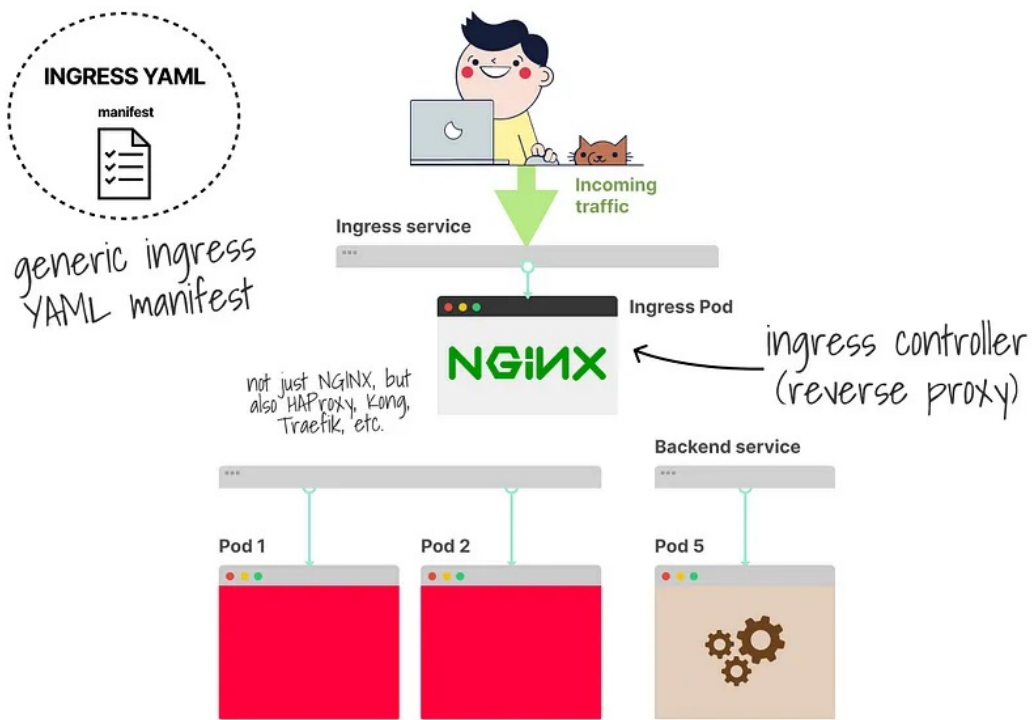


云原生拓展

默认情况下，Kubernetes 不附带 Ingress 控制器。

因此，您必须在集群中安装和配置所选的 Ingress 控制器。

但 Kubernetes 提供了一个 Ingress 清单（YAML）定义。



云原生拓展

无论您使用哪种 Ingress 控制器，确切的 YAML 定义都应有效。

该文件中的关键字段是：

1. Path (`spec.rules[*].http.paths[*].path`)
2. Backend (`spec.rules[*].http.paths[*].backend.service`)

```
~$ cat ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app
spec:
  rules:
    - http:
        paths:
          - backend:
              service:
                name: app
                port:
                  number: 80
            path: /
            pathType: Prefix
```

where the traffic should be forwarded to **1**

path to map **2**

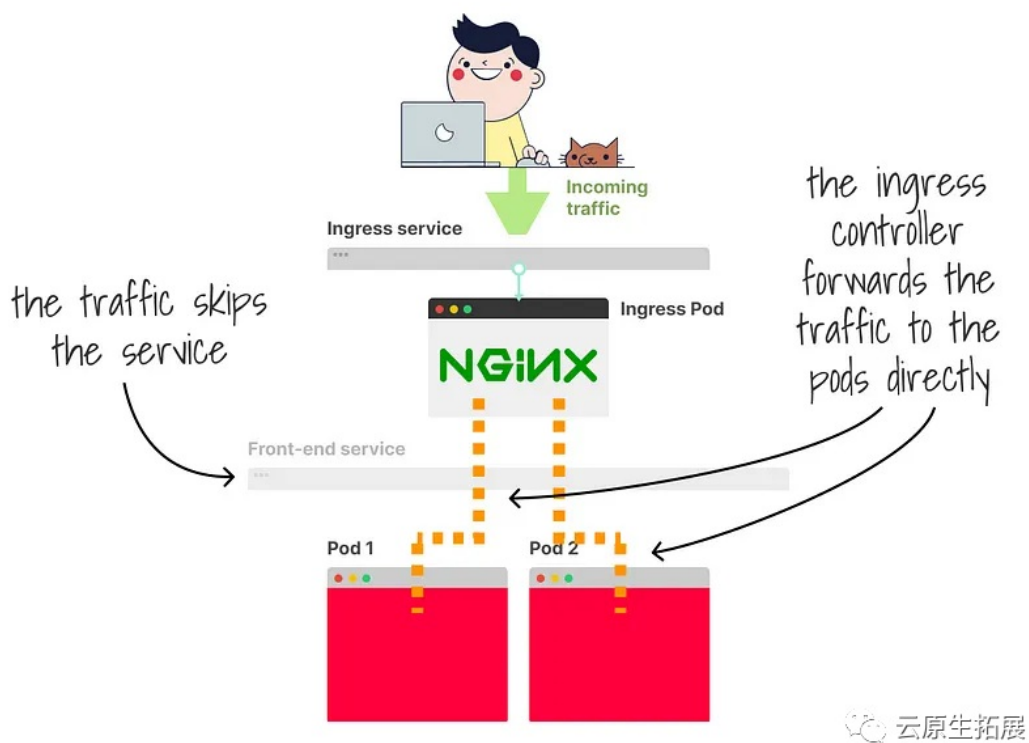
云原生拓展

`backend` 字段描述哪个服务应接收转发的流量。

但是，有趣的是，流量永远不会到达它。

这是因为控制器使用 `endpoint` 而不是 `Service` 来路由流量。

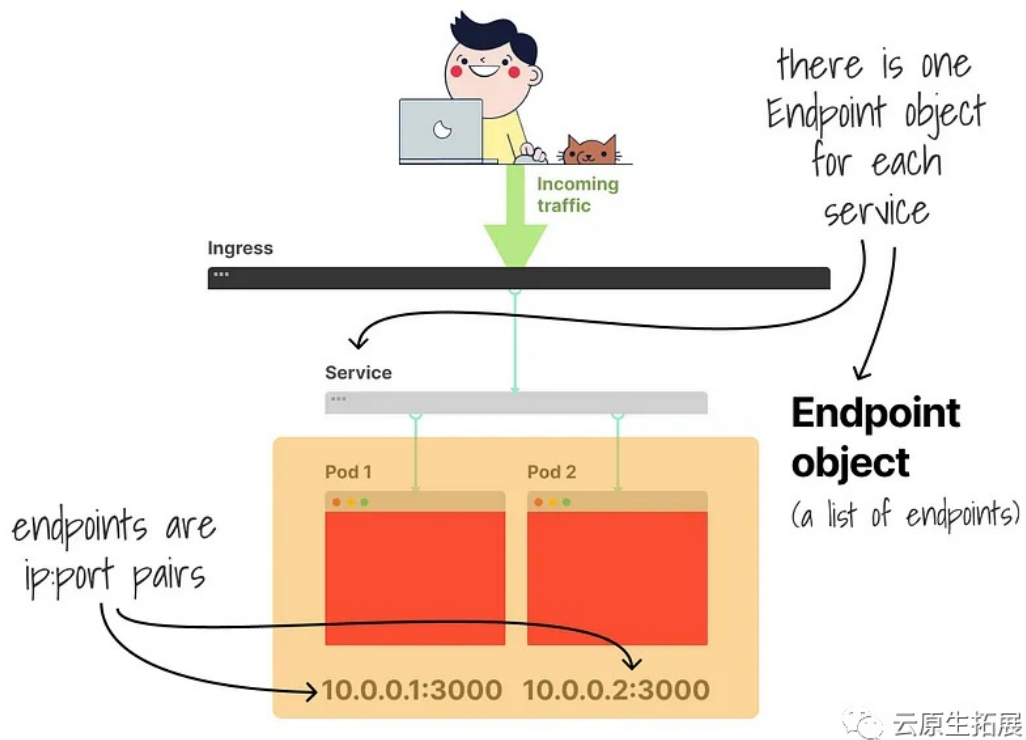
什么是 `endpoint`？



当您创建 `Service` 时，Kubernetes 会创建一个配套的 `endpoint` 对象。

`endpoint` 对象包含 `endpoint`（`ip:port` 对）的列表。

`IP` 和端口属于 `Pod`。



如果您想构建自己的控制器，这在实践中如何工作？

分为两部分：

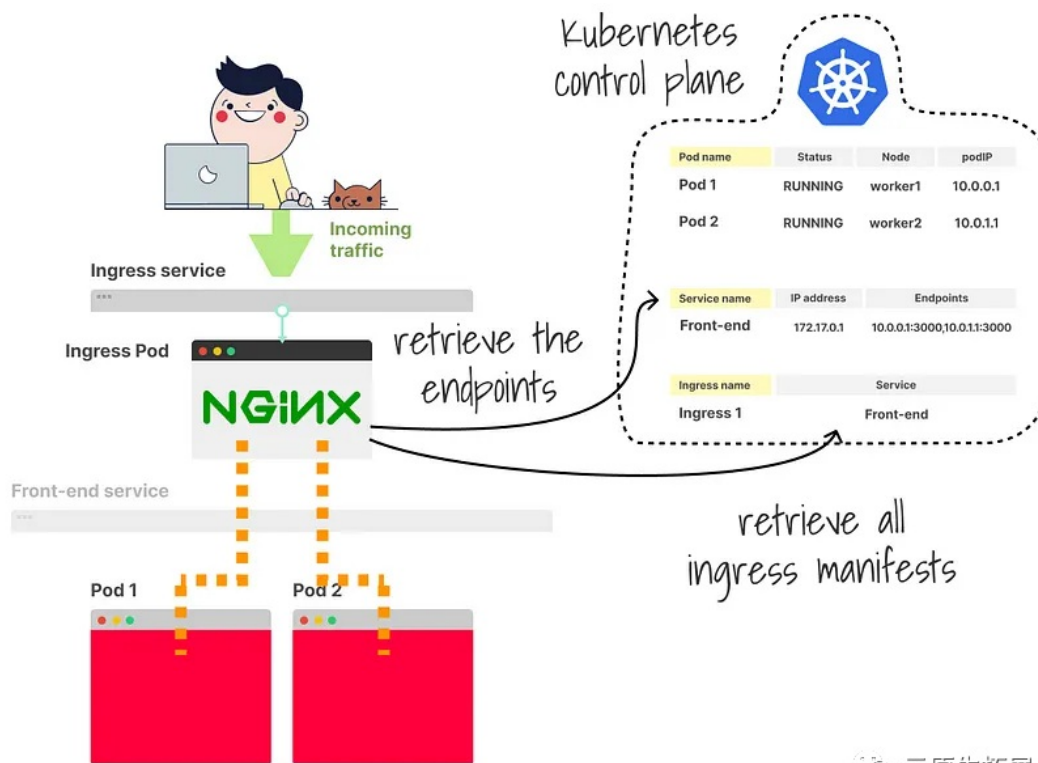
1. 从 Kubernetes 检索数据。
2. 重新配置反向代理。

让我们从检索数据开始。

在此步骤中，控制器必须监视 Ingress 清单和 endpoint 的更改。

如果创建了 Ingress YAML，则应配置控制器。

当 Service 发生变化（例如，添加新的 Pod）时也会发生同样的情况。

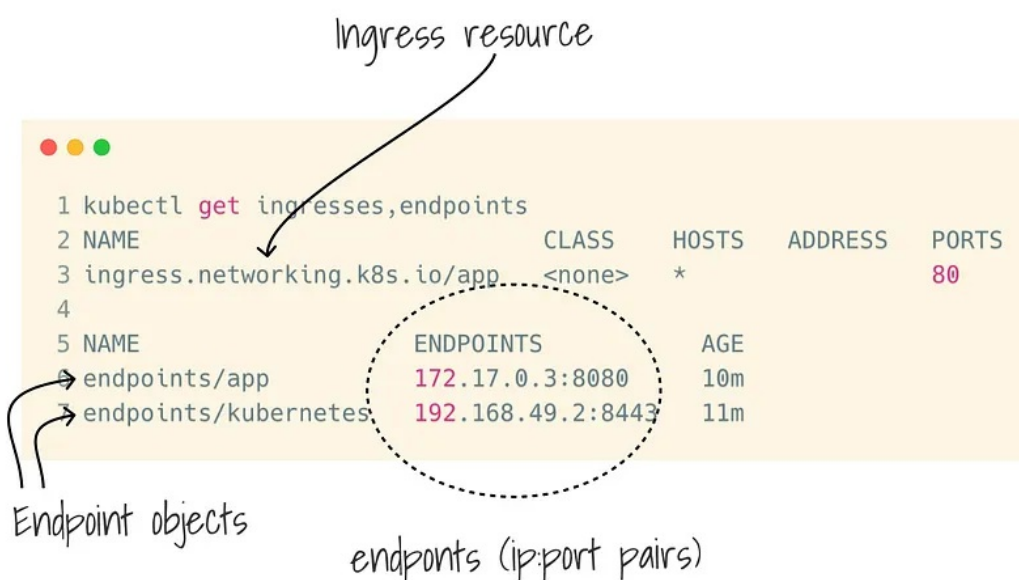


云原生拓展

实际上，这可以像 `kubectl get ingresses` 和 `kubectl get endpoints <service-name>` 一样简单。

使用此数据，您将获得以下信息：

- Ingress 清单中的 Path。
- 应接收流量的所有 endpoint。



云原生拓展

使用 `kubectl get ingresses`，可以获取所有 Ingress 清单并循环访问它们。

我使用 `-o jsonpath` 来过滤规则并检索：path 和 endpoint 服务。

For all ingress resources

```
1 IFS=' ' read -ra INGRESSES <<<"$(kubectl get ingress -o name)"
2
3 for i in "${INGRESSES[@]"; do
4     SERVICE_NAME=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].backend.service.name}')
5     PORT=$(kubectl get endpoints "$SERVICE_NAME" -o jsonpath='{.subsets[0].ports[0].port}')
6     HTTP_PATH=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].path}')
7 done
```

filtering json

get service name, port and path

云原生拓展

使用 `kubectl get endpoint <service-name>`，可以检索服务的所有 endpoint (ip: port 对)。

即使在这种情况下，我也使用 `-o jsonpath` 来过滤它们并将它们保存在 bash 数组中。

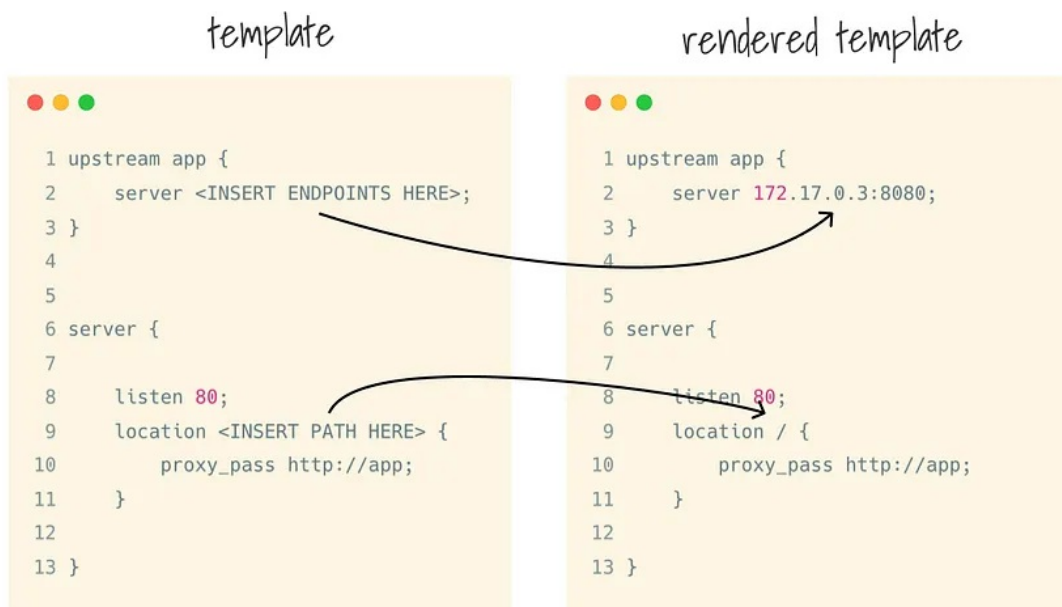
```
1 IFS=' ' read -ra INGRESSES <<<"$(kubectl get ingress -o name)"
2
3 for i in "${INGRESSES[@]"; do
4     SERVICE_NAME=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].backend.service.name}')
5     PORT=$(kubectl get endpoints "$SERVICE_NAME" -o jsonpath='{.subsets[0].ports[0].port}')
6     HTTP_PATH=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].path}')
7
8     IFS=' ' read -ra ENDPOINTS <<<"$(kubectl get endpoints "$SERVICE_NAME" -o
9     jsonpath='{.subsets[0].addresses[*].ip}')"
10 done
```

retrieve all endpoints for
a service

云原生拓展

此时，您可以使用数据重新配置 Ingress 控制器。

在我的实验中，我使用了 Nginx，所以我只是为 `nginx.conf` 编写了一个模板并热重载了服务。

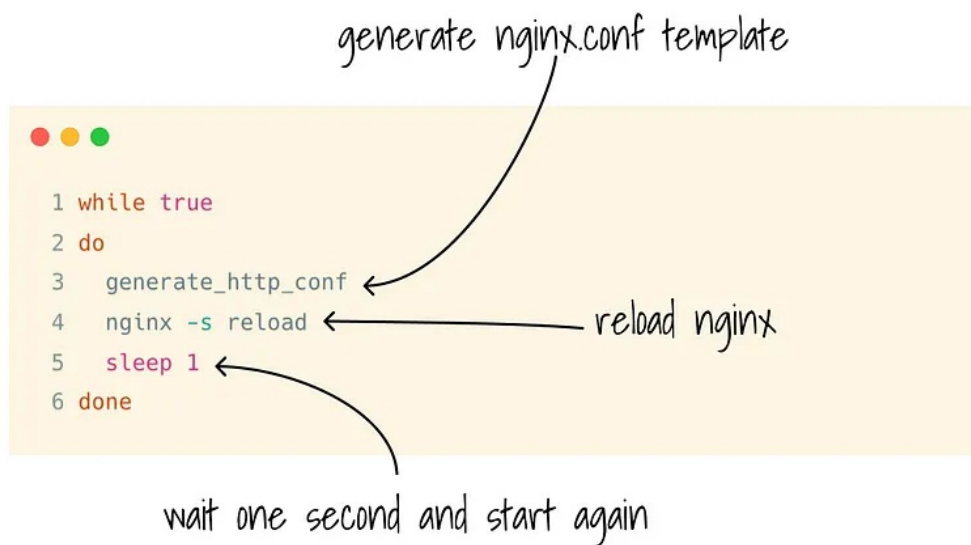


nginx.conf

云原生拓展

在我的脚本中，我没有费心去检测更改。

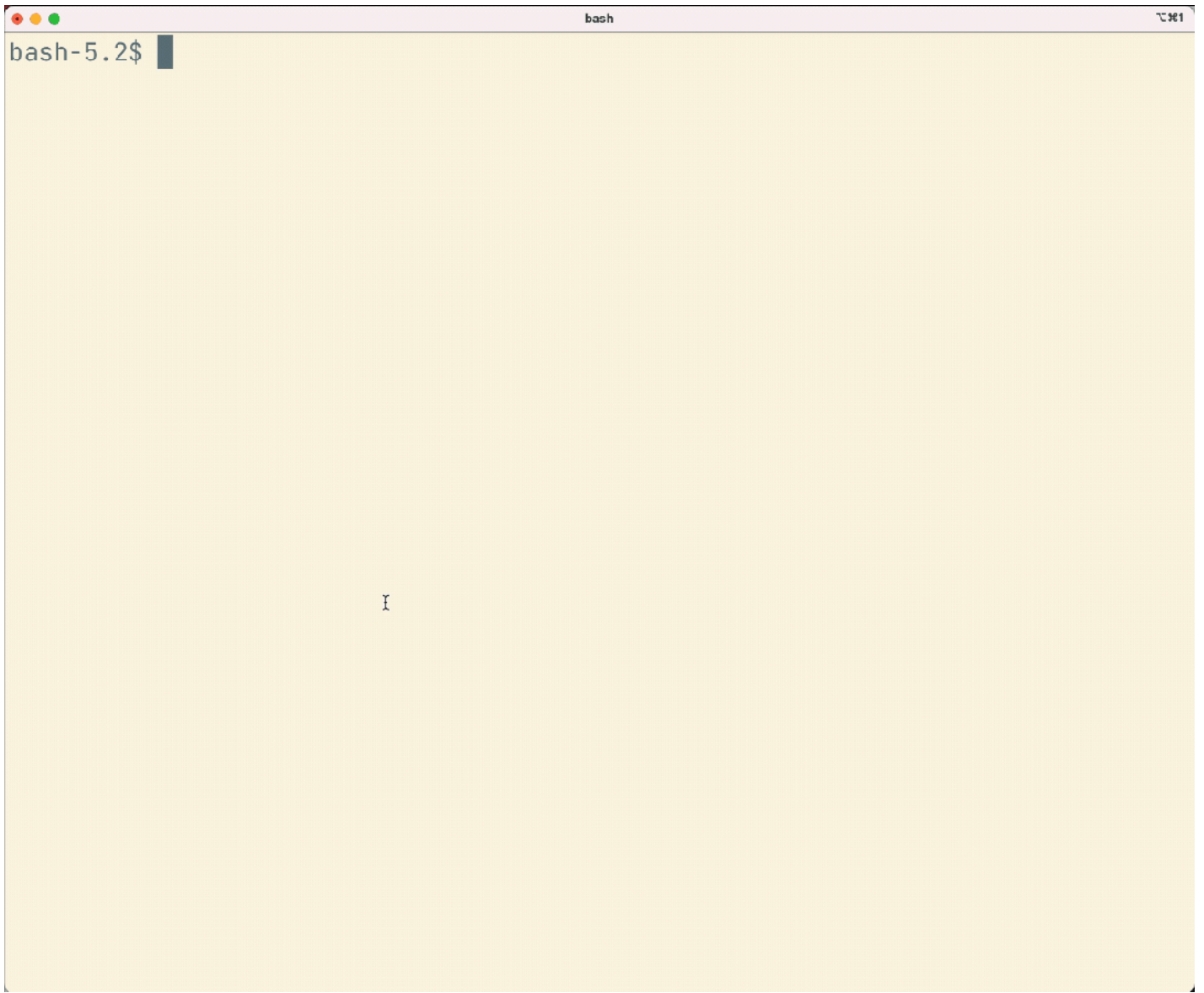
相反，我决定每秒完整地重新创建 `nginx.conf` 。



云原生拓展

最后一步是将脚本打包为容器并设置适当的 RBAC 规则，以便我可以从 API 服务器使用 API endpoint。

在这里，您可以找到它工作的演示：



如果你想玩代码，你可以在这里找到它：<https://github.com/learnk8s/bash-ingress>