82Kubernetes 系列（七十五）揭开 Ingress 的神秘面纱：高效服务通信的网关层

介绍

随着容器化成为构建和部署应用程序的实际方法，有效地编排这些容器至关重要。Kubernetes 是领先的容器编排平台，提供了广泛的功能来管理和扩展应用程序。这些基本功能之一是 Kubernetes Ingress。在这篇博文中，我们将深入探讨 Kubernetes Ingress 的世界、它的目的以及它如何在 Kubernetes 集群内实现高效的服务通信。

了解 Ingress

在 Kubernetes 中，Ingress 是一个 API 对象，充当前端，将外部流量路由到集群内运行的服务。它充当外部客户端访问群集节点上运行的服务的网关或入口点。简单来说，Ingress 公开 HTTP 和 HTTPS 路由，并根据一组预定义的规则管理流向不同后端服务的流量。

使用 Kubernetes Ingress 的好处

1. **路由灵活性：** Kubernetes Ingress 提供高级路由功能，允许流量根据各种条件（如路径、主机、HTTP 标头等）定向到不同的服务。这种灵活性使开发人员能够构建复杂的应用程序架构，将流量路由到不同的微服务，并无缝实施蓝绿或金丝雀部署。

2. **负载均衡：** Ingress 控制器是负责管理 Ingress 资源的组件，提供内置负载均衡功能。它们在服务的多个实例之间分配传入流量，确保高可用性和高效的资源利用率。此负载平衡功能增强了应用程序性能和可伸缩性。

3. **TLS 终止：** Ingress 支持 SSL/TLS 终止，实现客户端和后端服务之间的安全通信。它允许使用 SSL 证书来加密流量，确保数据的机密性和完整性。此功能消除了每个服务管理自己的 SSL 证书的需要，并简化了集群级别的加密管理。

4. **命名空间隔离：** Ingress 资源是 Kubernetes 中的命名空间级对象，这意味着它们可以限定为特定的命名空间。这种隔离使群集中的不同团队或应用程序能够拥有自己的专用 Ingress 配置，从而防止干扰并保持安全边界。

Ingress 控制器

为了利用 Ingress 资源，Kubernetes 需要一个 Ingress 控制器，作为 Ingress 规则的实现。有几个 Ingress 控制器可用，每个控制器都有自己的一组功能和配置选项。流行的例子包括NGINX Ingress Controller，Traefik，HAProxy Ingress和Istio Gateway。

Ingress 控制器在 Kubernetes 集群中作为 Pod 运行，并侦听 Ingress 资源中的更改。创建或修改新的 Ingress 对象时，控制器会动态更新其配置以反映所需的路由规则。这种动态特性允许根据集群的需求轻松扩展和重新配置 Ingress 控制器。

设置 Kubernetes Ingress

要设置 Kubernetes Ingress，集群必须部署正在运行的 Ingress 控制器。部署 Ingress 控制器的具体步骤因所选实现而异。通常，它涉及将控制器部署为 Pod 并配置必要的 RBAC（基于角色的访问控制）权限，以便控制器在群集中运行。

Ingress 控制器启动并运行后，创建 Ingress 资源就像使用 Ingress API 定义所需的路由规则一样简单。这些规则指定要将流量转发到的后端服务，以及所需的任何其他条件或设置。应用后，Ingress 控制器将选取更改并开始相应地路由流量。

让我们来看看一个如何配置基本 Kubernetes Ingress 的示例。

假设您设置了一个 Kubernetes 集群并部署了一个 Ingress 控制器，我们将创建一个简单的示例，其中外部流量根据不同的 URL 路径路由到两个不同的后端服务。

1. 部署后端服务

让我们首先部署两个示例后端服务来演示路由。可以使用以下 YAML 清单创建服务：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: service-a
spec:
  selector:
    app: service-a
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: service-b
spec:
  selector:
    app: service-b
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```

使用 `kubectl apply -f backend-services.yaml` 命令应用 YAML 文件。

2. 创建入口资源

接下来，我们将创建一个定义路由规则的 Ingress 资源。创建包含以下内容的 YAML 文件（例如 ingress.yaml ）：

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /service-a
        pathType: Prefix
        backend:
          service:
            name: service-a
            port:
              number: 80
      - path: /service-b
        pathType: Prefix
        backend:
          service:
            name: service-b
            port:
              number: 80
```

请注意，您需要将 `example.com` 替换为可访问 Ingress 控制器的实际域或 IP 地址。

使用 `kubectl apply -f ingress.yaml` 命令应用 YAML 文件。

3. 测试 Ingress

若要测试 Ingress，可以使用 curl 或 Web 浏览器等工具。假设 Ingress 控制器可在 `example.com` 处访问，则可以尝试访问以下 URL：

- `http://example.com/service-a` ：这应该将流量路由到 `service-a` 。
- `http://example.com/service-b` ：这应该将流量路由到 `service-b` 。

确保后端服务正确响应以验证路由。

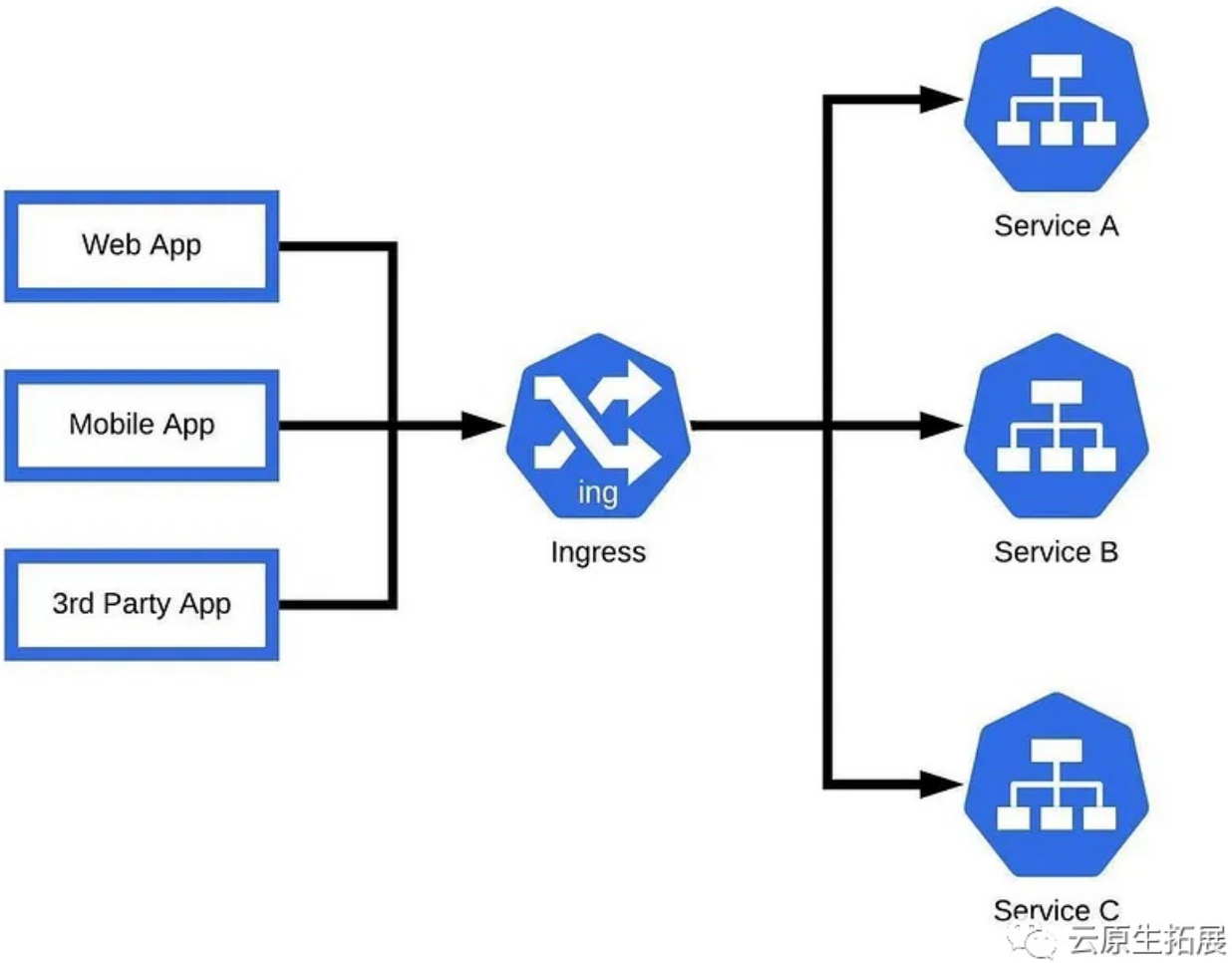
您已成功在 Kubernetes 中创建基本的 Ingress 配置。此示例演示如何根据不同的 URL 路径将流量路由到单独的后端服务。您可以探索 Ingress 的其他功能，例如 TLS 终止、高级路由规则等，以满足您的特定应用程序要求。

请记住根据您的环境和所需的路由规则自定义配置。

总结

Kubernetes Ingress 是管理和路由外部流量到 Kubernetes 集群内服务的关键组件。它提供强大的路由功能、负载平衡、SSL 终止和命名空间隔离。通过利用 Ingress，开发人员可以构建弹性和可扩展的应用程序体系结构，从而提高应用程序的整体

性能和安全性。


Web App
Mobile App
3rd Party App
ing
Ingress
Service A
Service B
Service C
云原生拓展