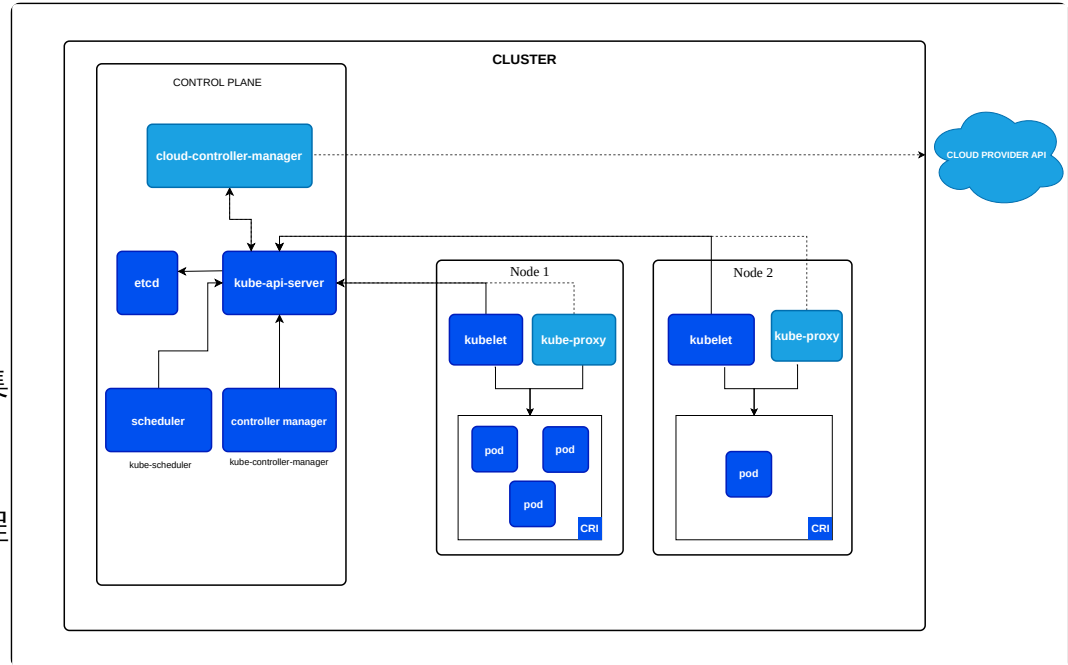
Kubernetes 零基础入门教程

从核心概念到实战部署的全面解析

什么是 Kubernetes？

自动化容器编排引擎

简而言之：如果把容器比作一个个装着货物的集装箱，那么 Kubernetes 就是负责管理、调度和搬运这些集装箱的“自动化港口系统”。它是一个用于自动化部署、扩缩和管理容器化应用程序的开源引擎。



为什么需要 K8s？

解决生产环境痛点

- **自动化自我修复**：自动重启或替换失败的容器。
- **弹性伸缩**：根据流量或资源使用率，自动增减应用实例。
- **服务发现**：提供统一的网络访问入口和自动负载均衡。
- **存储编排**：自动挂载您所选择的存储系统。

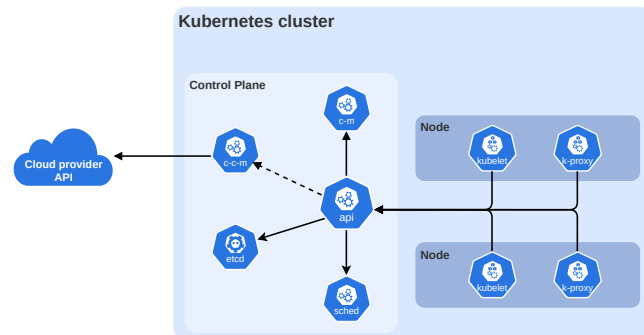
第一部分：Kubernetes 集群架构

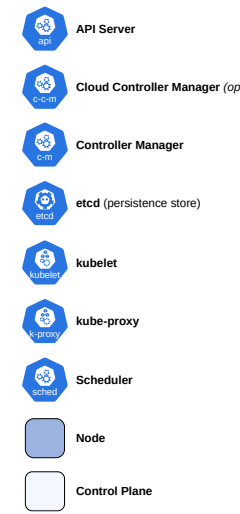
了解 K8s 的运转基础

核心架构：控制平面与节点

大脑与工作机器

- **控制平面 (Control Plane)**：集群的“大脑”。负责做出全局决策，例如根据资源分配将应用调度到哪台机器，以及响应集群的突发事件。
- **节点 (Nodes)**：集群的“工作机器”。每个节点上实际运行着您的容器化应用，并听从控制平面的指挥。





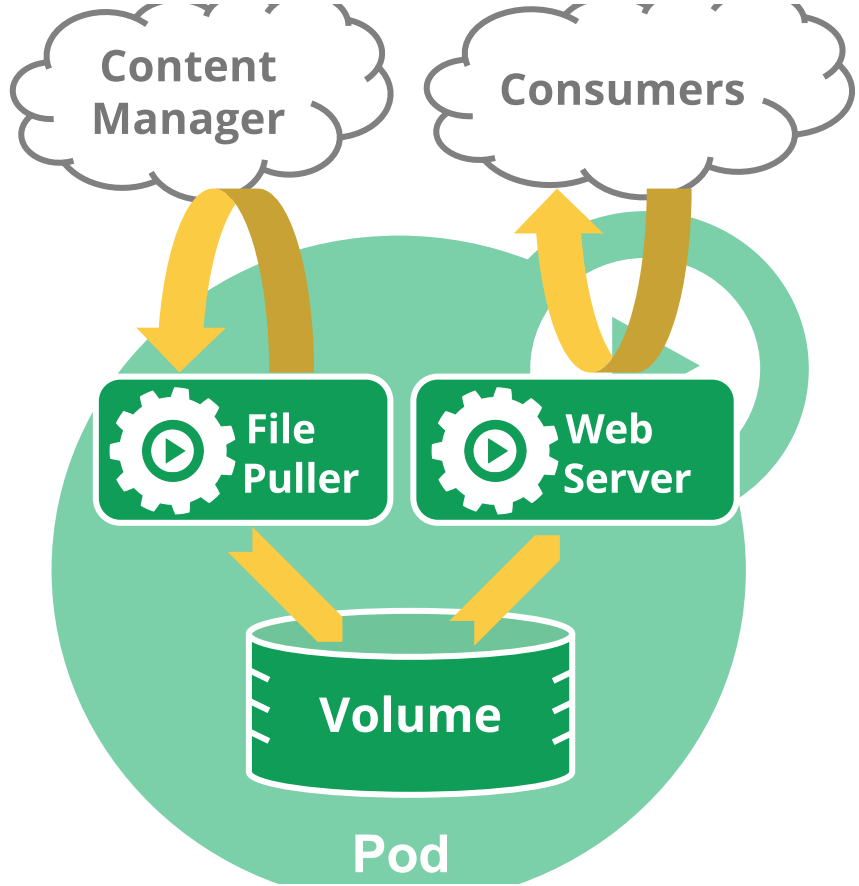
第二部分：核心对象与概念

Pod, Deployment, Service 与 Ingress

Pod（容器组）

最小的可部署计算单元

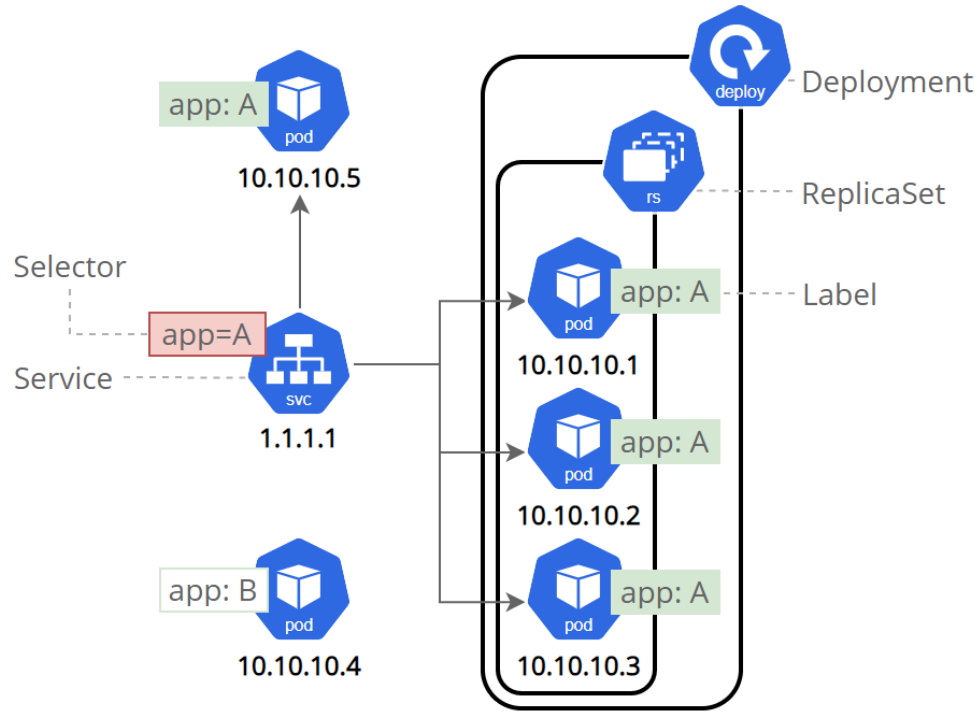
在 K8s 中，我们不直接运行单个容器，而是把容器包装在 Pod 里。一个 Pod 就像是一台“逻辑主机”，通常包含一个核心应用容器。需要注意的是，Pod 的生命周期是短暂的，随时可能被集群销毁和重建。



Deployment (部署)

声明式工作负载管理

- **取代手动管理**：通过它来自动化管理那些脆弱且短暂的 Pod。
- **声明式配置**：您只需告诉它“我需要应用的 3 个副本”，它就会在底层不断监控，确保永远有 3 个容器在运行。
- **平滑升级**：它可以帮助您以“滚动更新”的方式发布新版本代码，实现零停机部署。

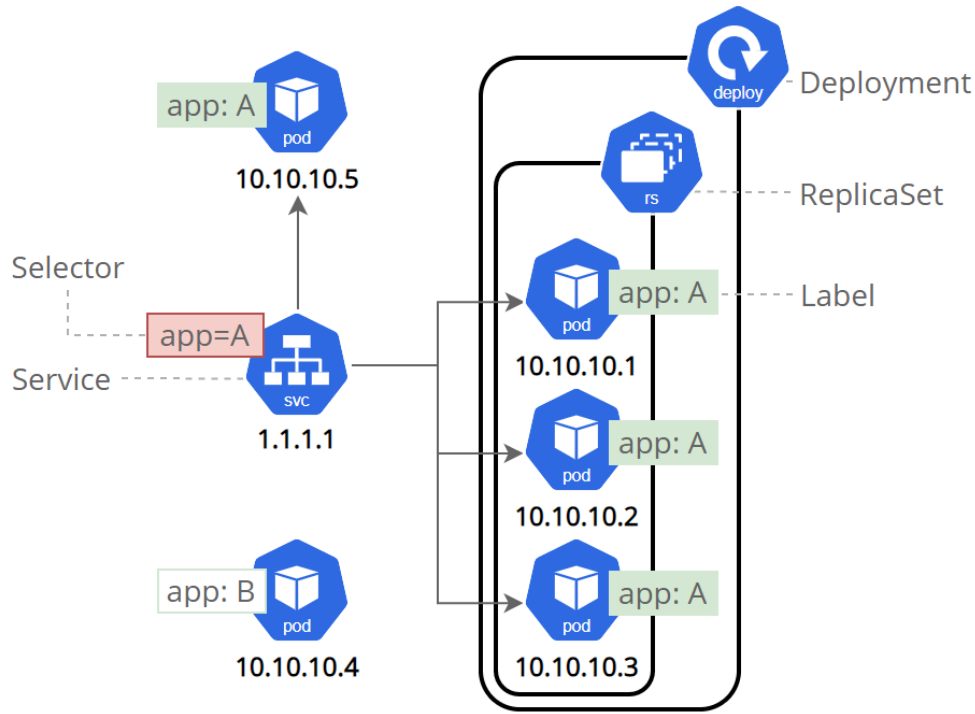


Service（服务）

稳定的内部网络通信

既然 Pod 会不断被创建和销毁导致 IP 变化，前端该如何找到后端的数据库呢？

Service 可以将一组 Pod 抽象封装起来，对集群内部提供一个固定的 IP 地址。无论后端的 Pod 怎么更替，流量都会被自动负载均衡到存活的容器上。

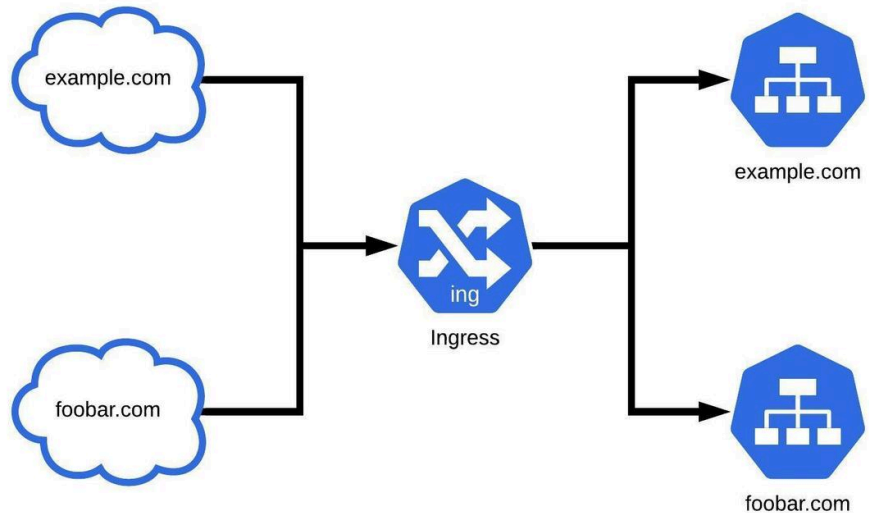


Ingress (入口网关)

统一的外部流量路由

如果您有多个 Service（如前端、API、后台），为每个 Service 暴露外部 IP 非常低效。

Ingress 充当集群的智能路由器。它可以基于域名（如 `api.example.com`）或 URL 路径（如 `/v1/users`），将来自集群外部的 HTTP/HTTPS 流量精准路由到集群内不同的 Service 上。



第三部分：实战演练

namespace, Pod, Service 与 Ingress 综合 Demo

0. 创建 Namespace

```
minikube addons enable ingress # enable Ingress for minikube
minikube tunnel

kubectl create namespace demo
kubectl config set-context --current --namespace=demo
```

1. 部署应用与服务

将以下内容保存为 `app.yaml`。我们使用 Deployment 运行 2 个 Pod 副本：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
  namespace: demo
spec:
  replicas: 2
  selector:
    matchLabels: { app: web }
  template:
    metadata: { labels: { app: web } }
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        ports: [{ containerPort: 80 }]
```

2. 部署服务

将以下内容保存为 `service.yaml`。我们使用 Service 选择到 app: web 的 Pod，并暴露它们：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-service
  namespace: demo
spec:
  selector: { app: web }
  ports: [{ port: 80, targetPort: 80 }]
```

3. 配置路由网关 (Ingress)

将以下内容保存为 `ingress.yaml`。它会将访问 `localhost` 的流量路由到刚刚的 Service：

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
  namespace: demo
spec:
  rules:
  - host: localhost
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: web-service
            port:
              number: 80
```

Refs

- https://kubernetes.io
- minikube
- ...

Q&A

Thank You!