Basic Concepts in Kubernetes and Istio

Kubernetes

- Kubernetes (aka. K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

Kubernetes

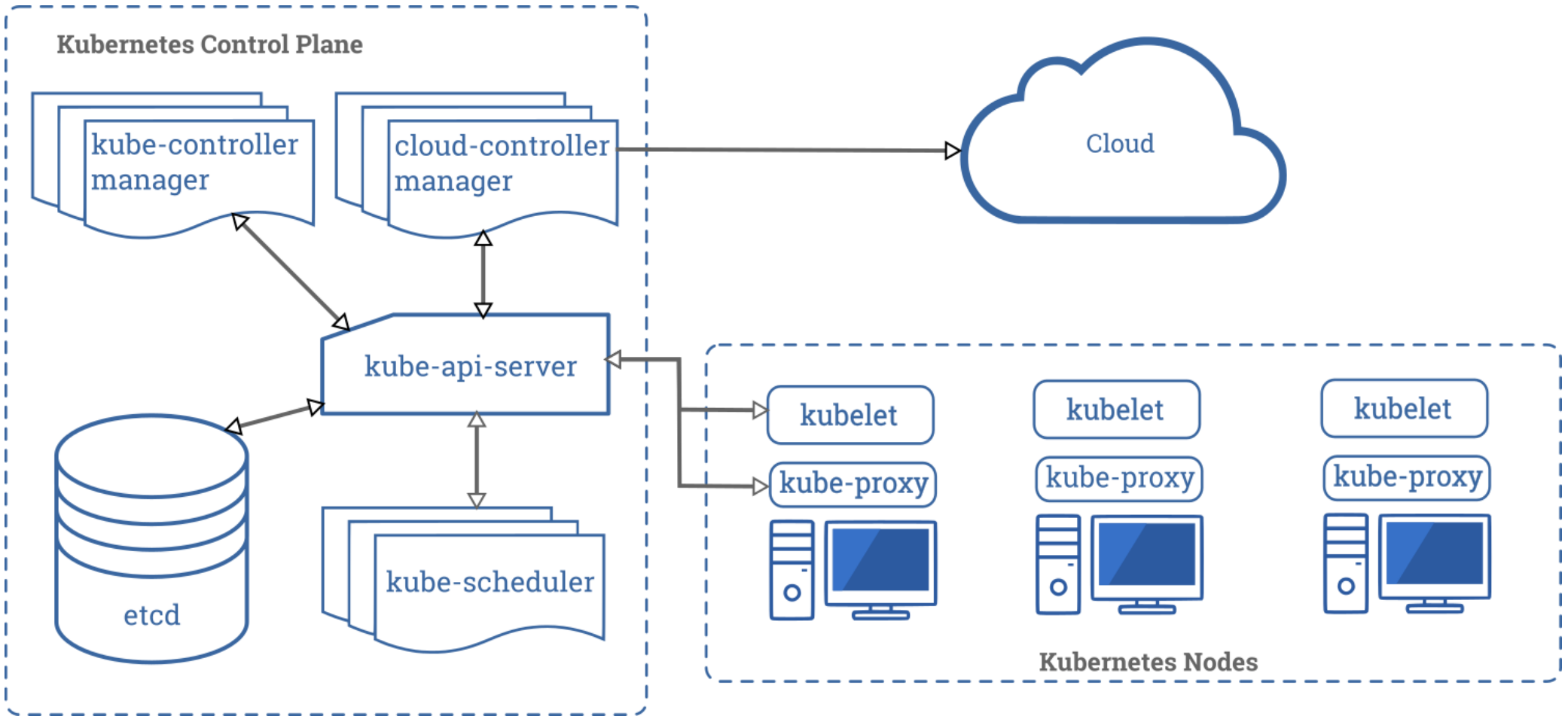
Features

- Orchestrate containers across multiple hosts
- Make better use of hardware
- Control and automate application deployments and updates
- Mount and add storage to run stateful apps
- Scale containerized applications and their resources
- Declaratively manage services
- Reliability guarantee of applications

Kubernetes

Core Concepts

- **Kubernetes cluster**: k8s system runs as a cluster, every cluster has at least one node
- **Master**: the machine that controls Kubernetes nodes
- **Node**: a node is a worker machine in k8s, that run containerized applications
  - **Kubelet**: an agent makes sure that containers are running in a Pod
  - **kube-proxy**
  - **Container Runtime**

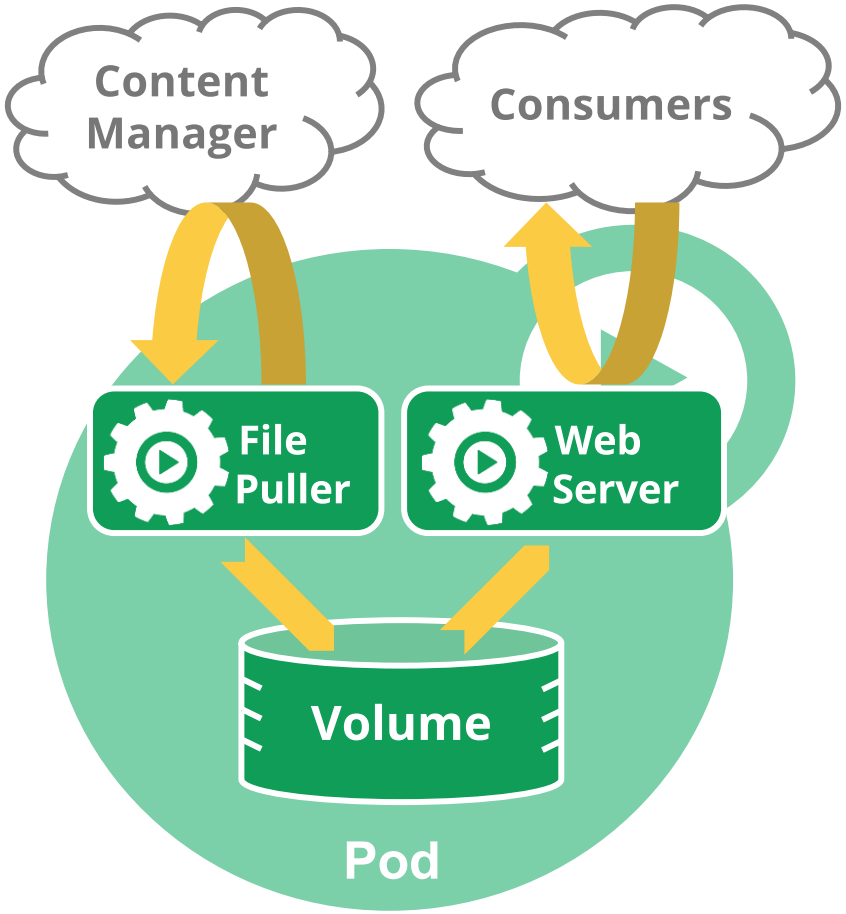


Kubernetes Cluster Components

Kubernetes

Core Concepts

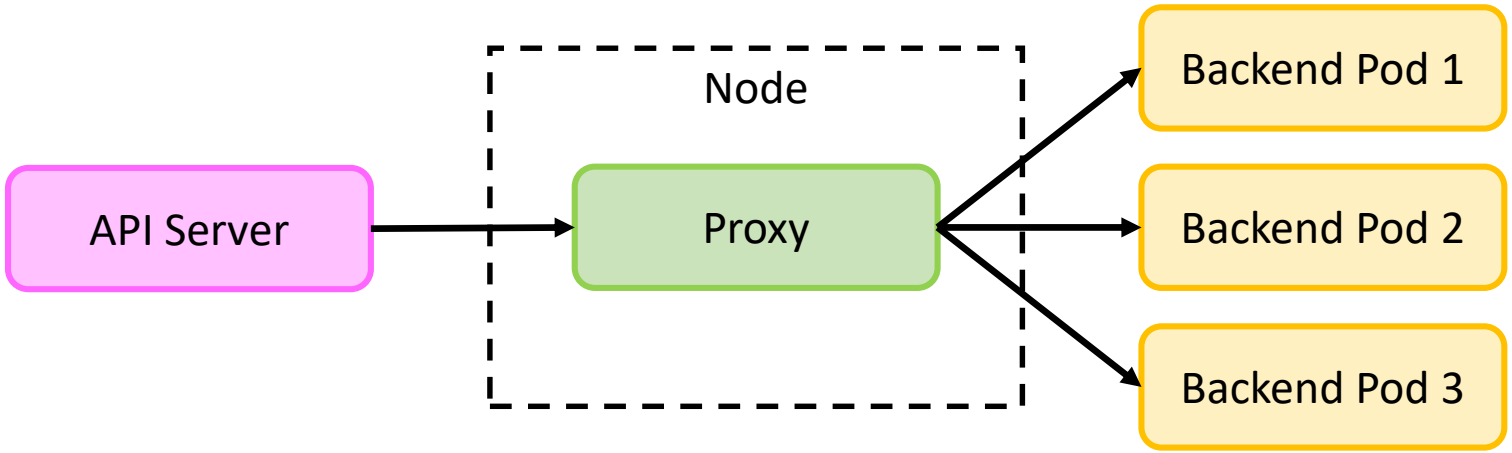
- **Pod**: the basic execution unit of k8s application
  - Pod is a **group** of containers
  - Each Pod is meant to run a single instance of a given application
  - All containers in a pod share the same resources *(network, storage)*
  - Stateful *(StatefulSets)* & stateless



Kubernetes

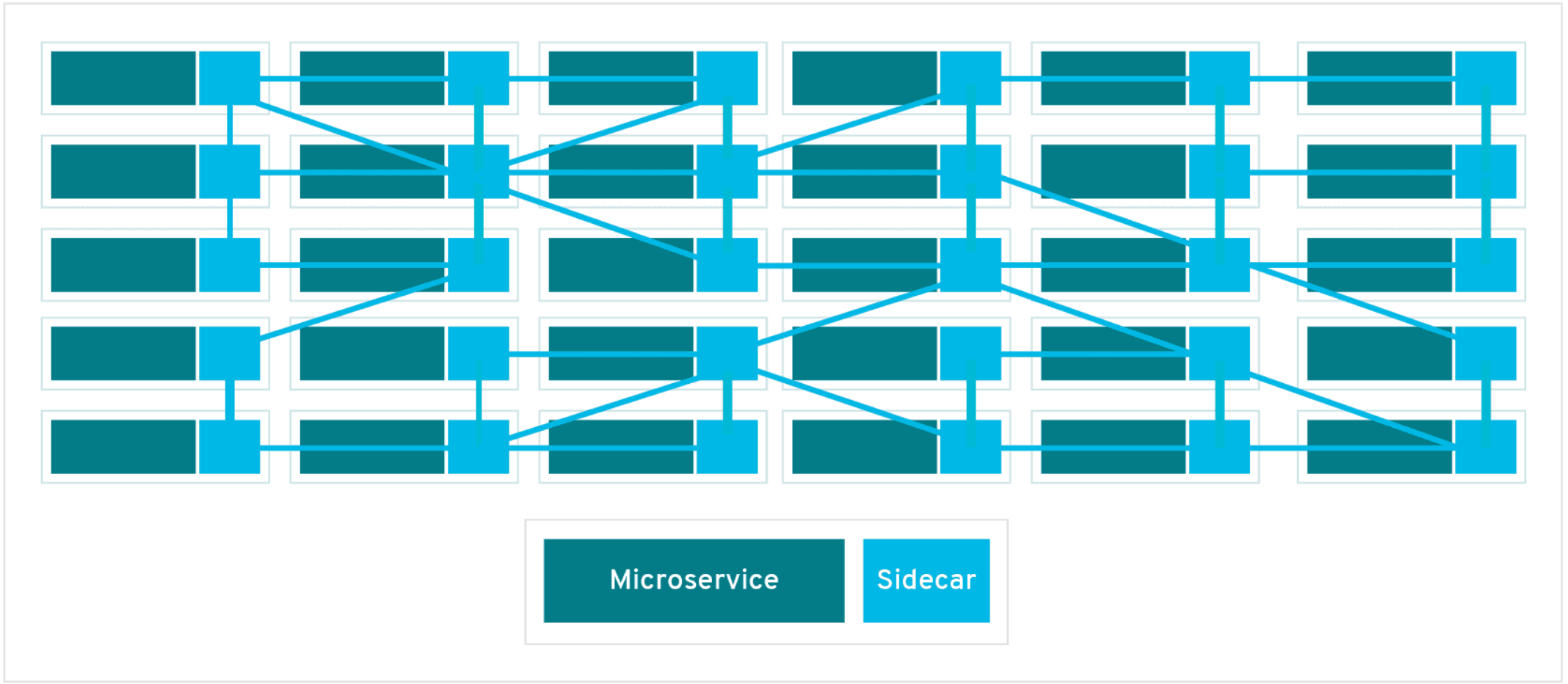
Core Concepts

- **Service**: an abstraction of a set of application runs on pods
  - Other applications can’t access pods directly
  - Every pods in a service is the same, no matter where it is
  - Kubernetes service proxies automatically get service requests to the right pod



Service Mesh

- *Background: Hard to maintain service-to-service communication between microservices*
- Service Mesh is a way to control how different parts of an application interaction with each another
- Service Mesh's requirements can include discovery, load balancing, failure recovery, metrics, and monitoring
- Service mesh also often has more complex operational requirements, like A/B testing, canary rollouts, rate limiting, access control, and end-to-end authentication



How microservices in Service Mesh communicate with each other

Istio

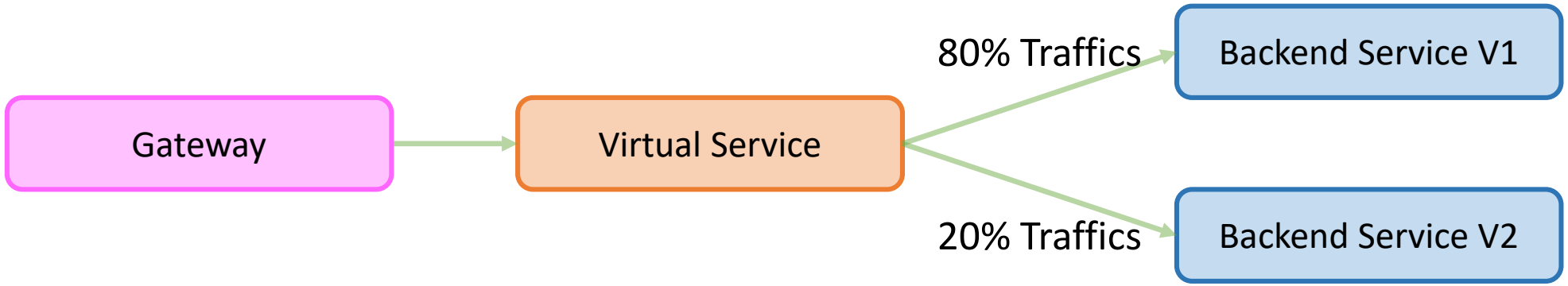
- Traffic management
- Security
- Observability

Istio

- **Traffic management**
  - Relies on Envoy proxies *(sidecar)*
  - Istio will connect to service discovery system, and detects the services and endpoints in that cluster
  - Each service workload has a load balancing pool
  - Envoy proxies will distribute traffic to instance in the pool

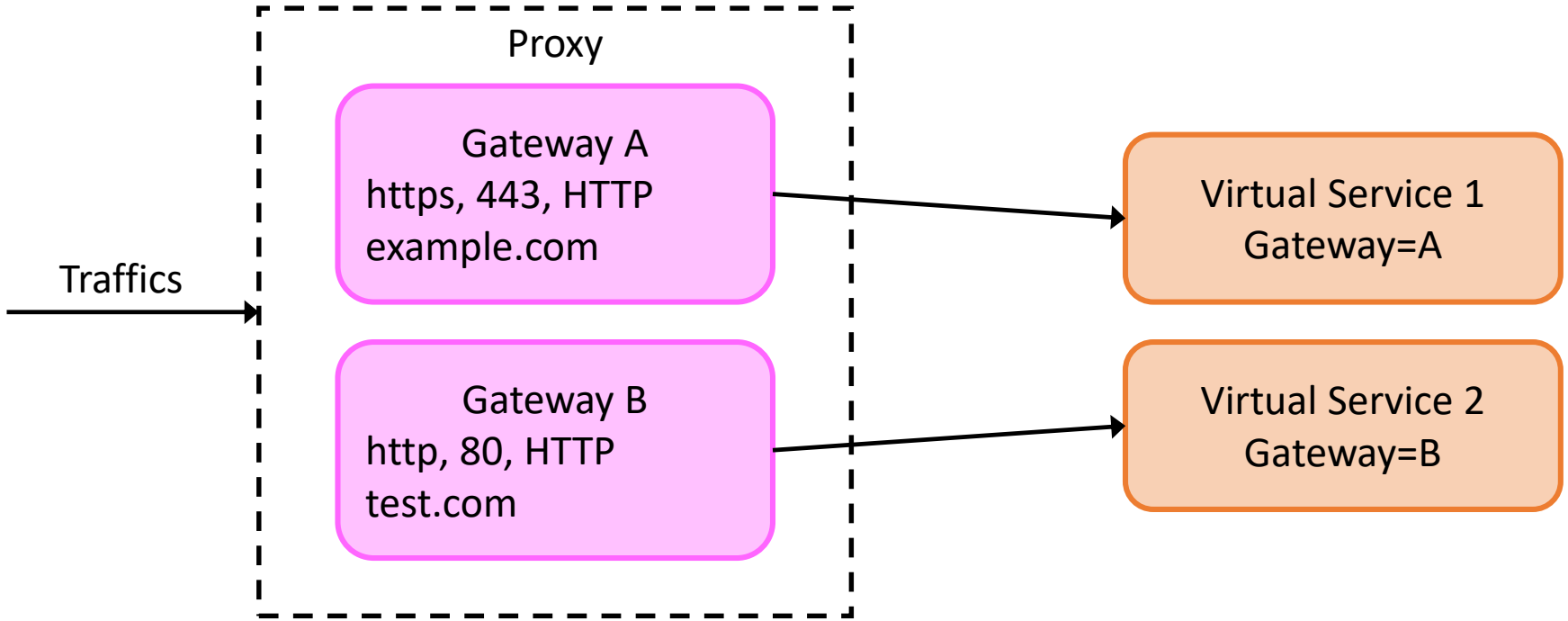
Istio

- **Traffic management: Virtual Services**
  - Configure how requests are routed to a service within an Istio service mesh
  - Each virtual service consists of a set of routing rules
  - Making Istio's traffic management flexible and powerful



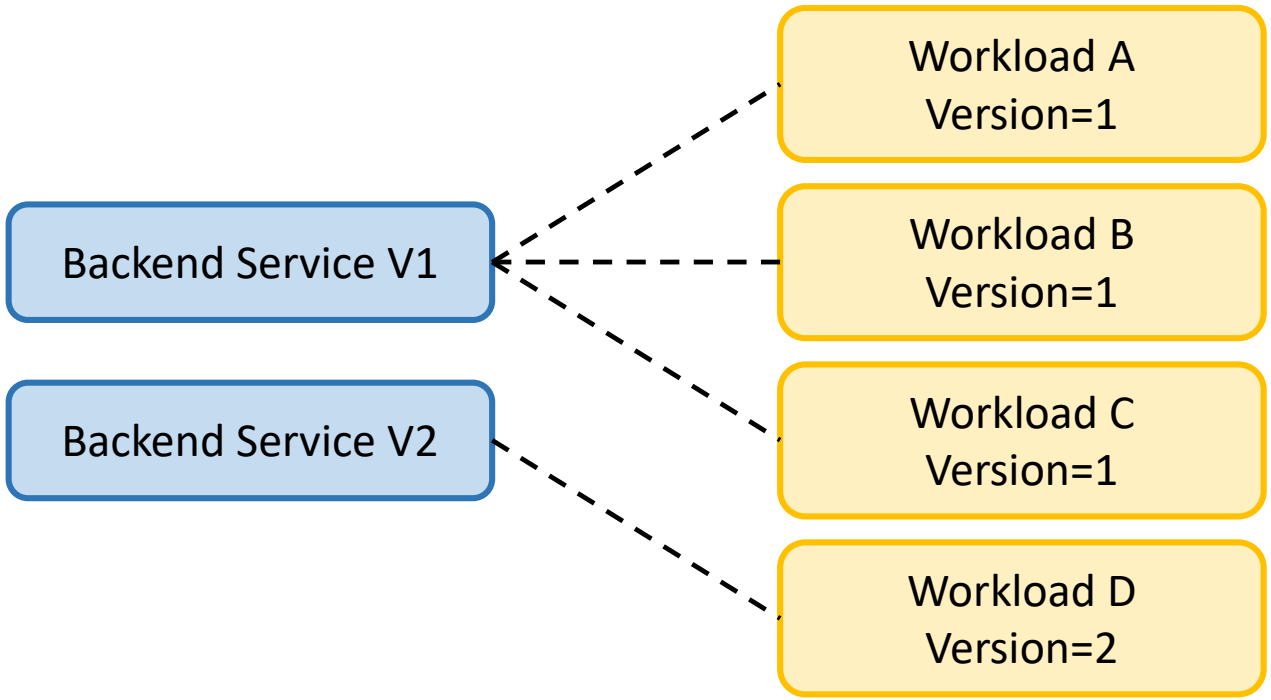
Istio

- **Traffic management: Gateway**
  - Manage inbound and outbound traffic for your mesh
  - Gateway configurations are applied to standalone Envoy proxies



Istio

- **Traffic management: Destination Rules**
  - Configure what happens to traffic **for** the destination
  - Define how traffic corresponds and load balance to real services or pods



Istio

- **Traffic management: Service entries**
  - Add an entry to the service registry that Istio maintains internally
  - Allows you to manage traffic for services running outside of the mesh