Introducing the Ingress API Object and Controller

Nigel Brown

@n_brownuk www.windsock.io

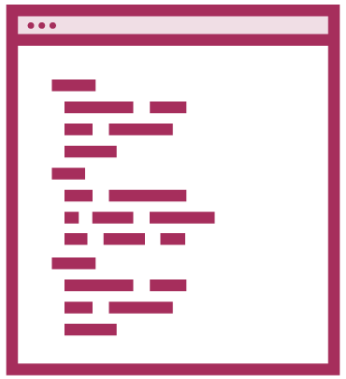
Module Outline

- Introduce the Ingress concept and API
- Discuss the nature of ingress controllers
- Differentiate between host-based and path-based routing
- Learn how to define ingress objects to route requests to backends

Ingress

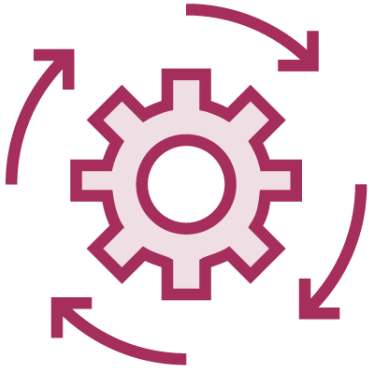
A Kubernetes API object that manages the routing of external HTTP/S traffic to services running in a cluster

Ingress Objects and Controllers



Ingress Object Definition

+



Ingress Controller Deployment

Why Third-party Controllers?

Part of the ingress controller's function, is to act as a reverse proxy for cluster workloads

Best of breed is subjective, instead we have the ability to choose a solution that works for us

Infrastructure providers can create ingress controllers optimized for their environments

Proxy-based Ingress Controllers

Nginx

Maintained by the k8s community

https://git.io/fh4UC

Traefik

Based on a cloud native edge router

https://bit.ly/2GiweUA

Contour

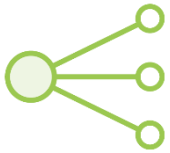
Uses the popular Envoy service proxy

https://git.io/fh4Ua

Characteristics of the Ingress API

Defines traffic routes between external clients and services

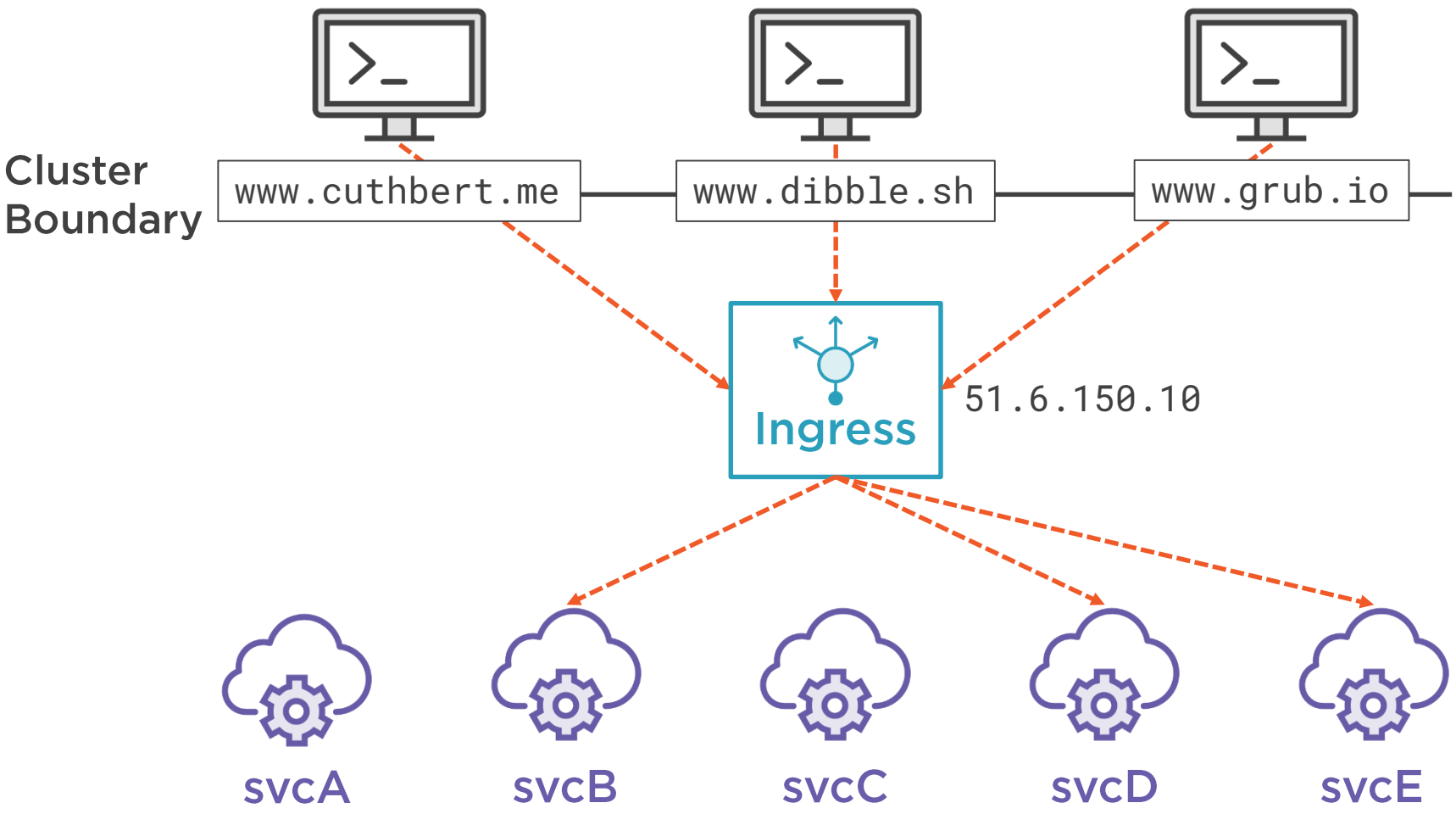
Allows encrypted communication using Transport Layer Security



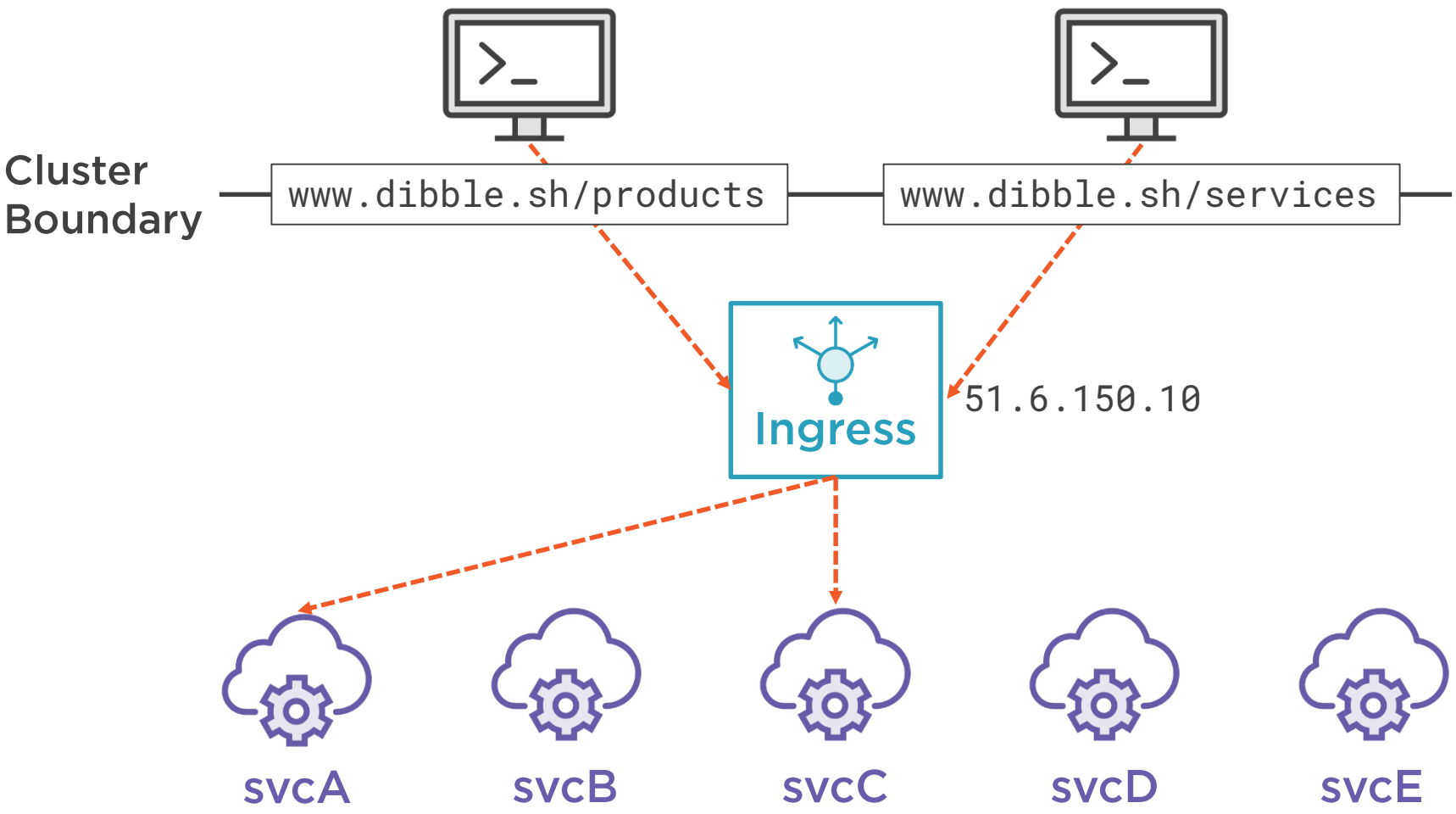
Load balances client traffic across a service's endpoints (pods)

The Ingress API is still beta, and is likely to evolve over time

Name-based Virtual Hosts



Path-based Routing



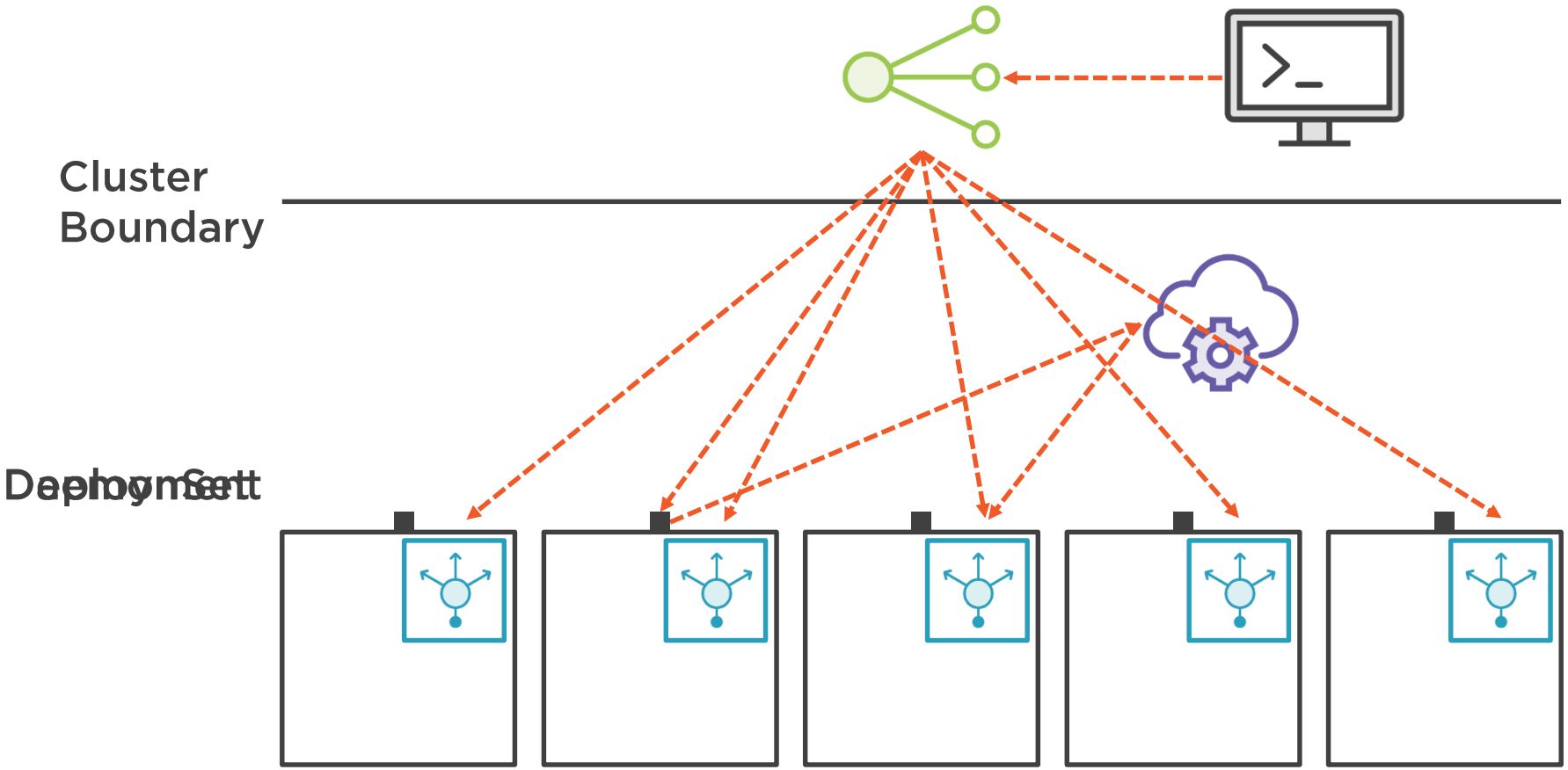
Demo

Discuss the available deployment options for ingress controllers

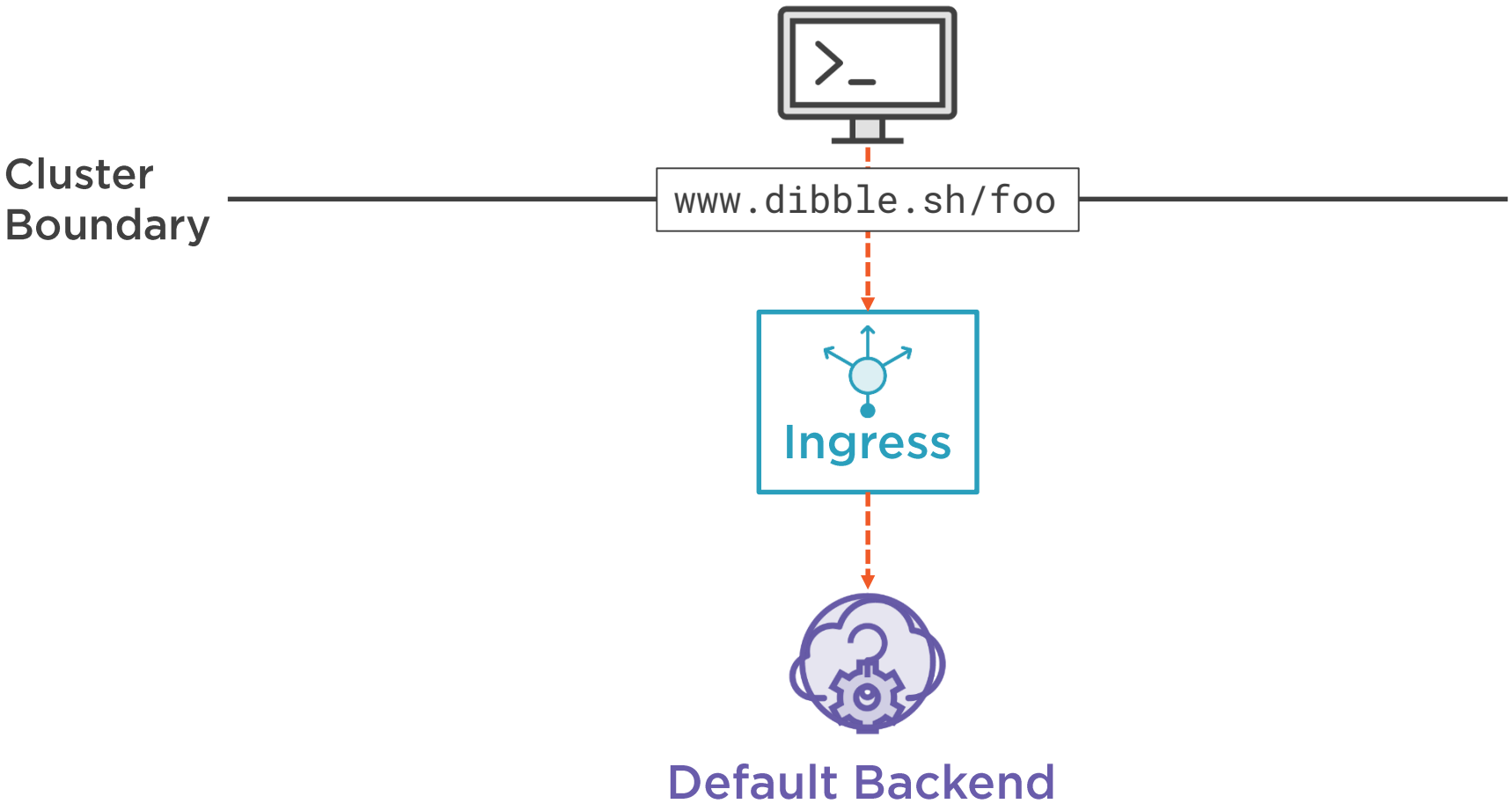
Deploy the open source Traefik ingress controller

Inspect the API objects created in the cluster

Ingress Controller Deployment



Invalid Requests



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: single-default-backend-ingress
```

Defining an Ingress API Object

An ingress object is defined using the Ingress resource kind

```
spec:
  backend:
    serviceName: default-backend-svc
    servicePort: 8080
```

Defining a Default Backend Service

The backend field defines the default backend service

Limitations of the Default Backend Service

Perfect design for routing ingress traffic to a single backend service

Works with ingress definitions containing rules for HTTP requests

Not so good when there are multiple default backend definitions

Ingress controllers use different mechanisms for resolving the issue

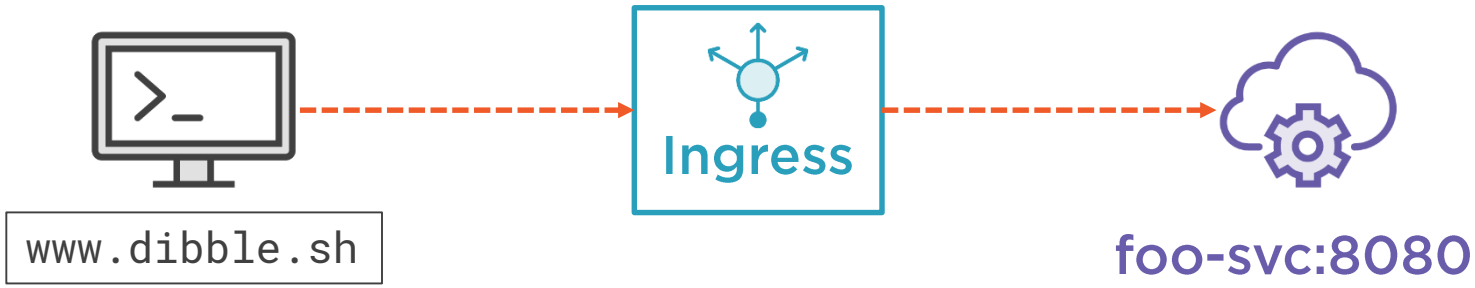
```
# Community Nginx Ingress Controller
--default-backend-service

e.g. $(POD_NAMESPACE)/default-http-backend
```

Configuring the Nginx Default Backend

The config option specifies the service and its namespace

A custom default backend service can be configured to suit you

Host Rule Routing

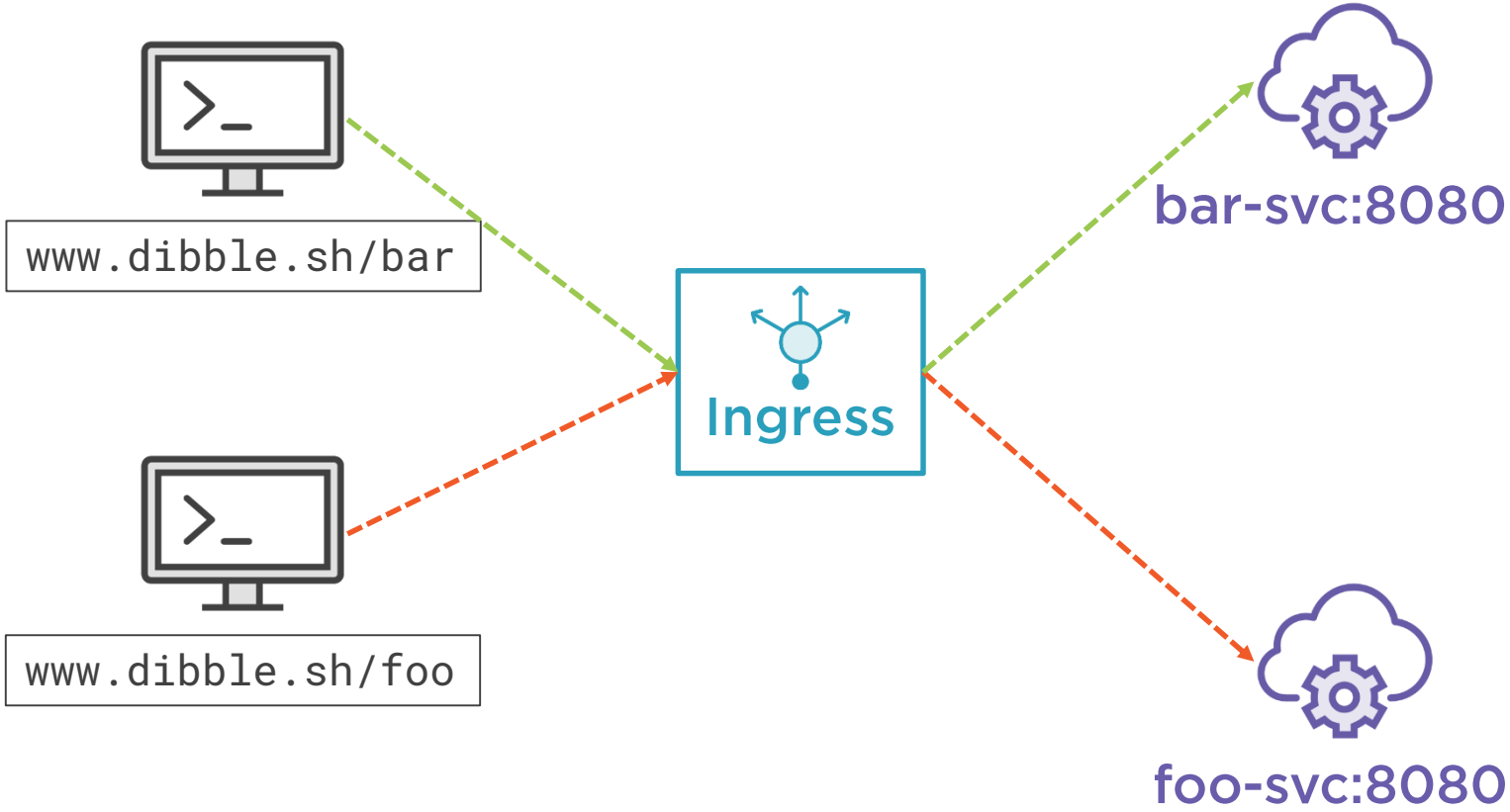


```
spec:
  rules:
  - host: www.dibble.sh
    http:
      paths:
      - backend:
          serviceName: foo-svc
          servicePort: 8080
```

Ingress Host Rule

A host rule requires a host field and a list of HTTP paths

Path Rule Routing



```
spec:
  rules:
  - http:
      paths:
      - path: /foo
        backend:
          serviceName: foo-svc
          servicePort: 8080
      - path: /bar
        backend:
          serviceName: bar-svc
          servicePort: 8080
```

- Path field specifies the required path of the URL request
- Additional paths can be added to the list

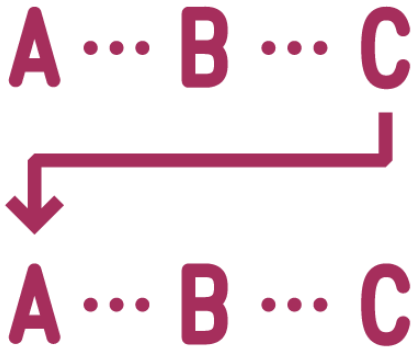
Path Rule Considerations

Regex

Paths are expressed as extended POSIX regex

Rewrite

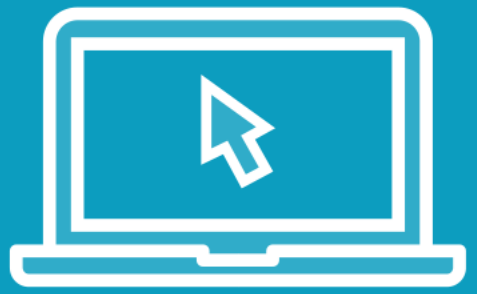
Request paths may need to be rewritten



Priority

It may be necessary to prioritize paths

Demo



Define and deploy an ingress for default backend service

Create an ingress object for host-based routing

Reconfigure the ingress object for path-based routing

Module Summary

The Ingress API allows us to define routes for ingress traffic

Ingress controllers fulfil the routing defined in Ingress objects

Default backends help to manage the response to invalid requests