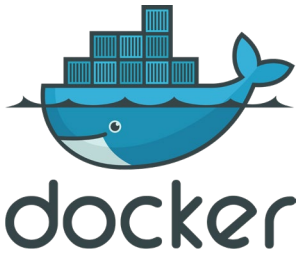
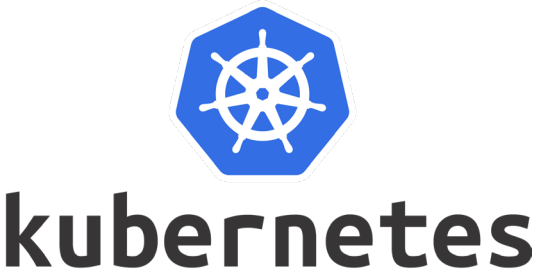
kubernetes

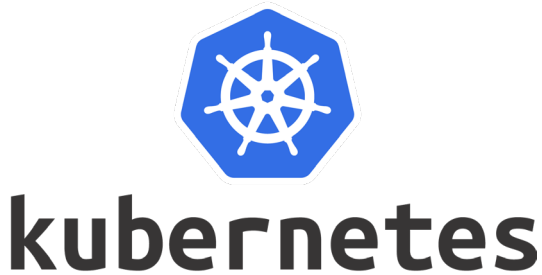
Kubernetes training Itenium

Jan Wielemans
jan.wielemans@devgem.be



Evening 1
24/11/2022





Evening 2
22/12/2022

Debugging

Specific questions day 1

- ## Docker logs

  Local File logging driver | Docker Documentation
  https://tecadmin.net/truncate-docker-container-logfile/

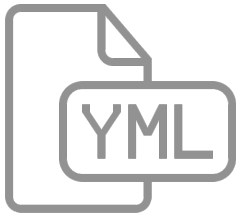
- ## Docker image labeling schema

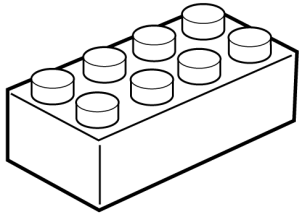
  Label Schema | Bringing Structure To Labels (label-schema.org)

Recap

kubernetes

- Docker
  - CLI `docker`
  - Pull and create images
  - Run and configure containers
  - docker-compose
- Kubernetes
  - Lens
  - CLI `kubectl`
  - Pod





Service

kubernetes

Service
- abstraction
- loadbalancing
- type
- based on selectors

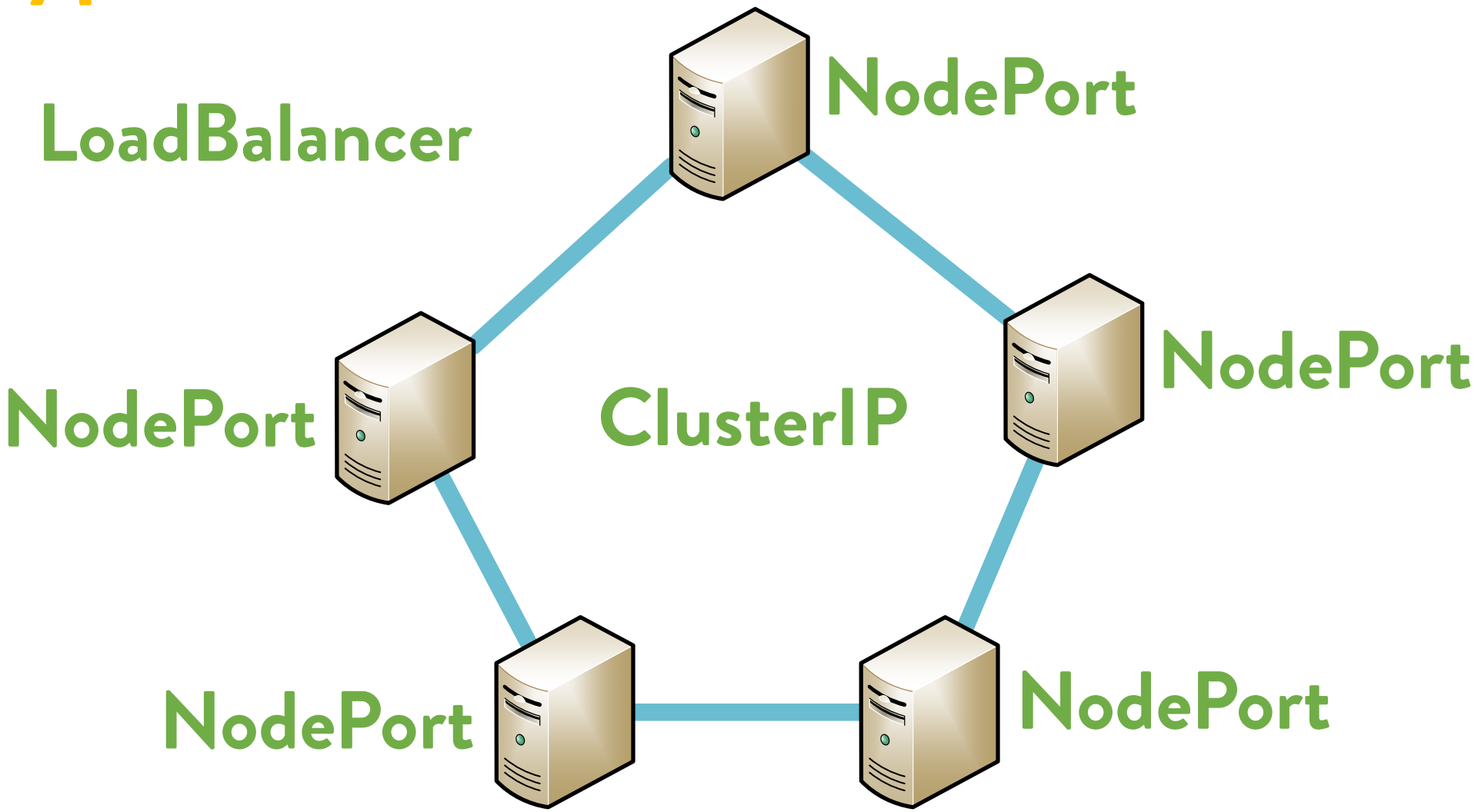
Pod

my-app

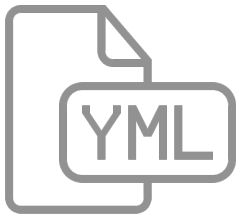
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app-pod
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

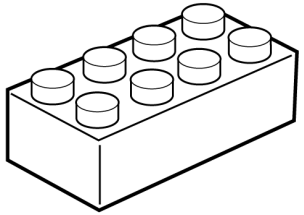
Service types

kubernetes



Demo time
Service





Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-app-ingress
spec:
  rules:
    - host: myapp.goca-training.be
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              service:
                name: my-app-service
                port:
                  number: 80
  ingressClassName: nginx
```

kubernetes

*.goca-training.be
aaa.bbb.ccc.ddd

Loadbalancer

Ingress Service

NGINX

Ingress Controller

myapp.goca-training.be

app1.goca-training.be

app2.nogietsanders.com

Service

Service

Service

Deployment

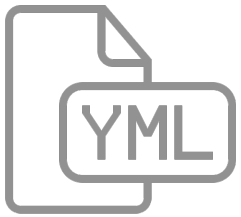
my-app

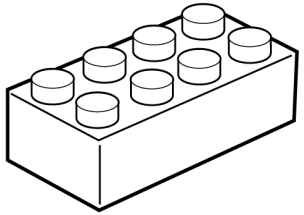
Deployment

app2

Deployment

app3





Namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: demo
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: demo
  name: demo-backend
  labels:
    app: demo-backend
spec:
  ...
```

kubernetes

What?

Logical division of the cluster in virtual sub-clusters (that can communicate internally and can have RBAC)

Why?

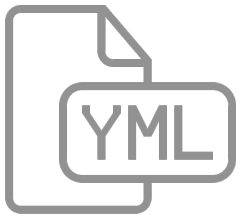
Division by application or environment (e.g. test, staging) or whatever you want

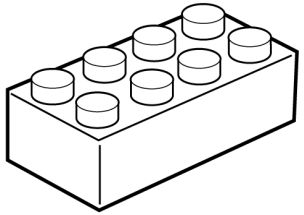
How?

Assign each resource to a namespace (except nodes and persistent volumes, default is default, don't use kube-system, kube-public or kube-node-lease)

Demo time

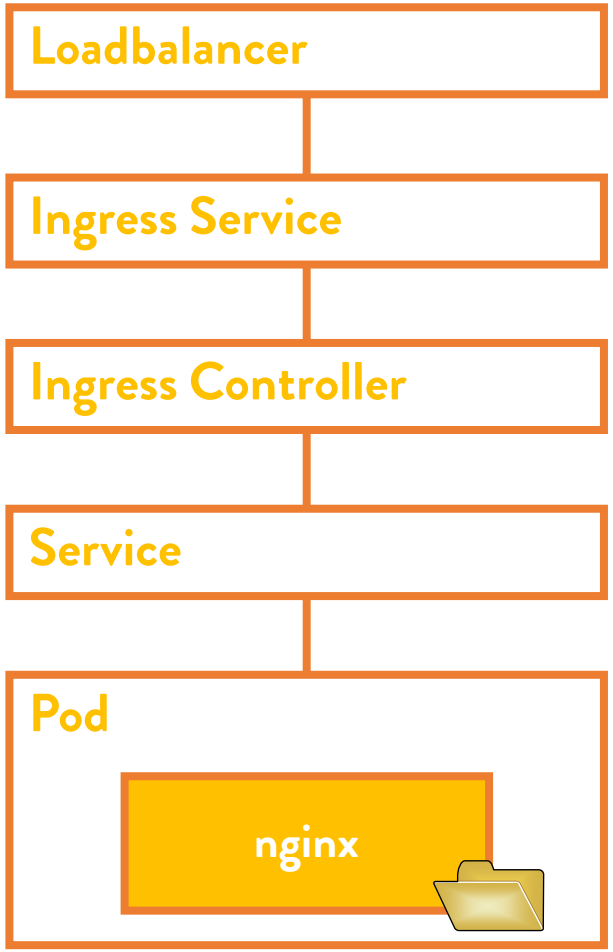
- Ingress
- Multi api resource
- Namespaces
- TLS

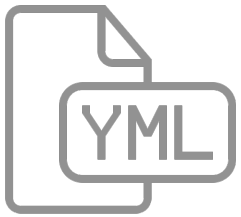


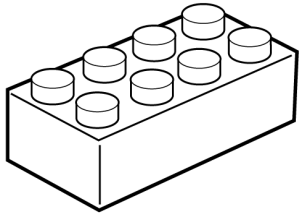


Volumes

kubernetes







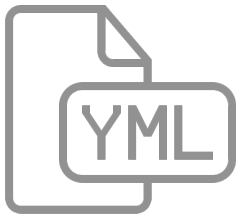
Volumes

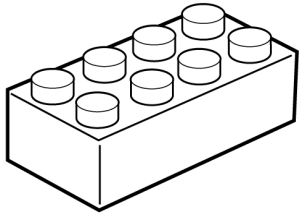
kubernetes

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nginx-pvc
spec:
  storageClassName: do-block-storage
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx-pod
spec:
  containers:
    - name: nginx-container
      image: nginx:1.23.1
      volumeMounts:
        - mountPath: /usr/share/nginx/html
          name: nginx-volume
  volumes:
    - name: nginx-volume
      persistentVolumeClaim:
        claimName: nginx-pvc
```

Demo time
Volumes





Deployment

Deployment

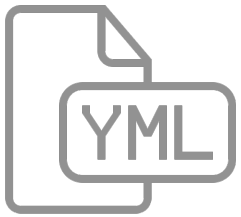
- metadata (e.g. labels)
- specs (replicas, selectors)
- pod template
- scaling
- rolling updates

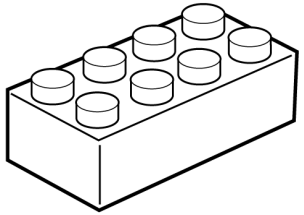
demo-db-mongodb

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: demo
  name: demo-db-mongodb
  labels:
    app: demo-db-mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-db-mongodb
  template:
    metadata:
      namespace: demo
      labels:
        app: demo-db-mongodb
    spec:
      containers:
        - name: demo-db-mongodb
          image: mongo:latest
```

Demo time

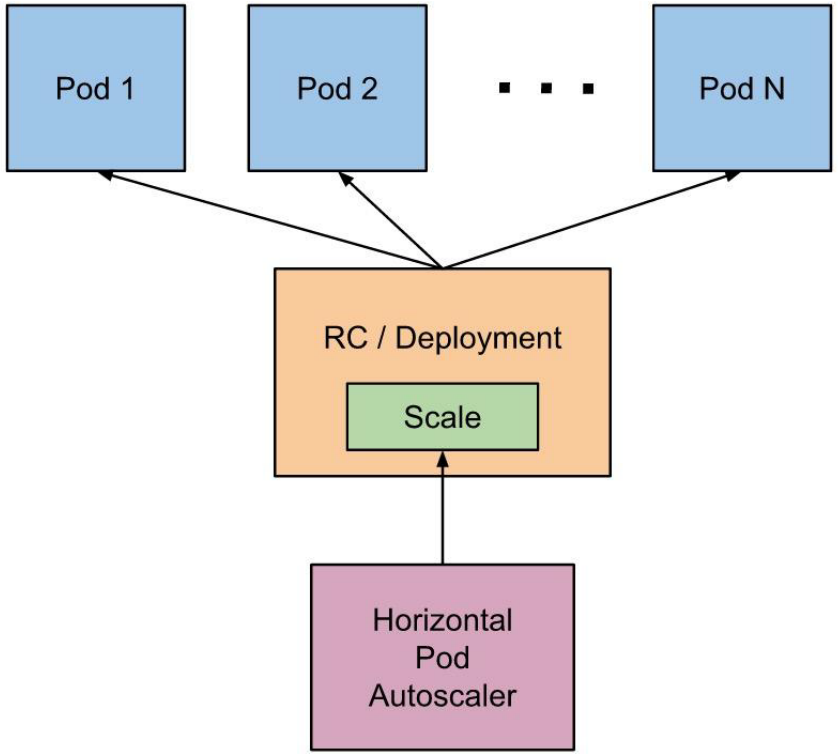
- Deployment
- Scale & loadbalance
- Rolling updates
- Probe
- Bringing it all together
- Intro debugging

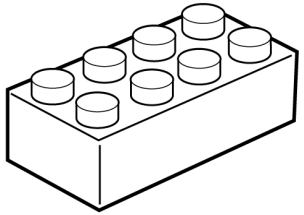




Horizontal Pod Autoscaler

kubernetes





Cluster Autoscaler

kubernetes

autoscaler/README.md

GitHub, Inc. [US] | https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/README.md

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

kubernetes / autoscaler

Watch 61 Star 592 Fork 276

Code Issues 50 Pull requests 5 Projects 0 Insights

Branch: master ▾ autoscaler / cluster-autoscaler / README.md Find file Copy path

aleksandra-malinowska Add 1.2 as recommended version with Kubernetes 1.10 d10fea7 16 days ago

10 contributors

126 lines (97 sloc) 6.86 KB Raw Blame History

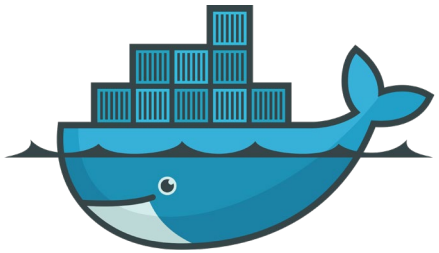
Cluster Autoscaler

Introduction

Cluster Autoscaler is a tool that automatically adjusts the size of the Kubernetes cluster when:

- there are pods that failed to run in the cluster due to insufficient resources.
- some nodes in the cluster are so underutilized, for an extended period of time, that they can be deleted and their pods will be easily placed on some other, existing nodes.

FAQ/Documentation



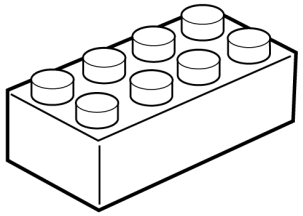
docker

kubernetes

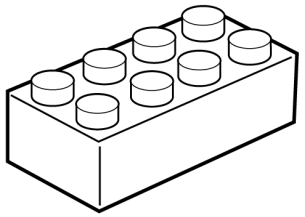
kubernetes

CLI´s:

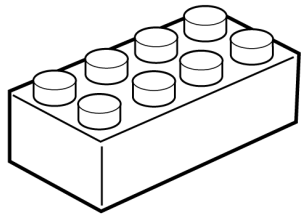
- docker
- kubectl



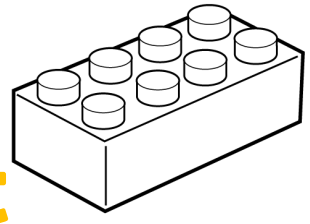
Pod



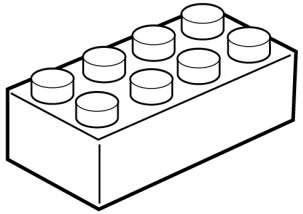
Ingress



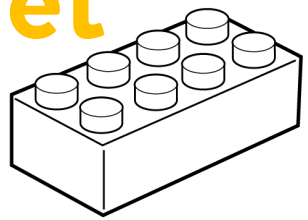
Volumes



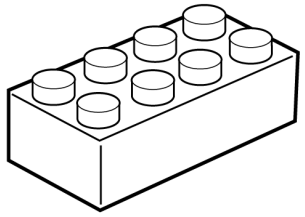
Secret



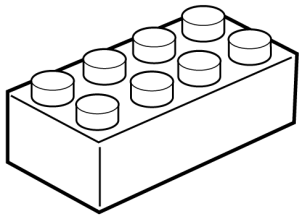
Deployment



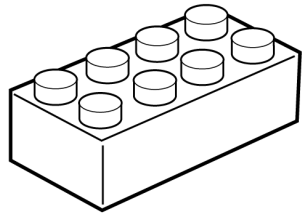
Namespace



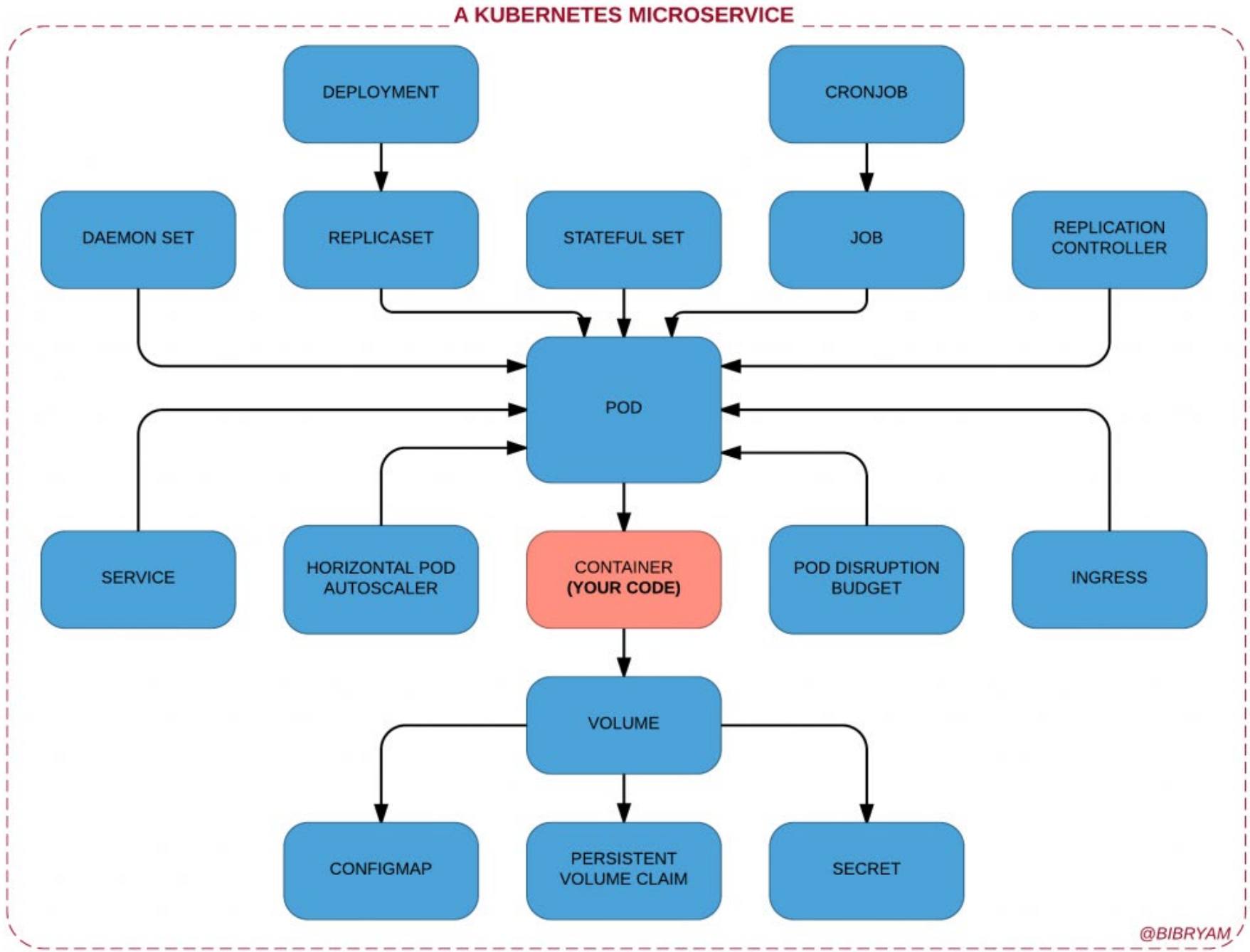
Horizontal Pod Autoscaler



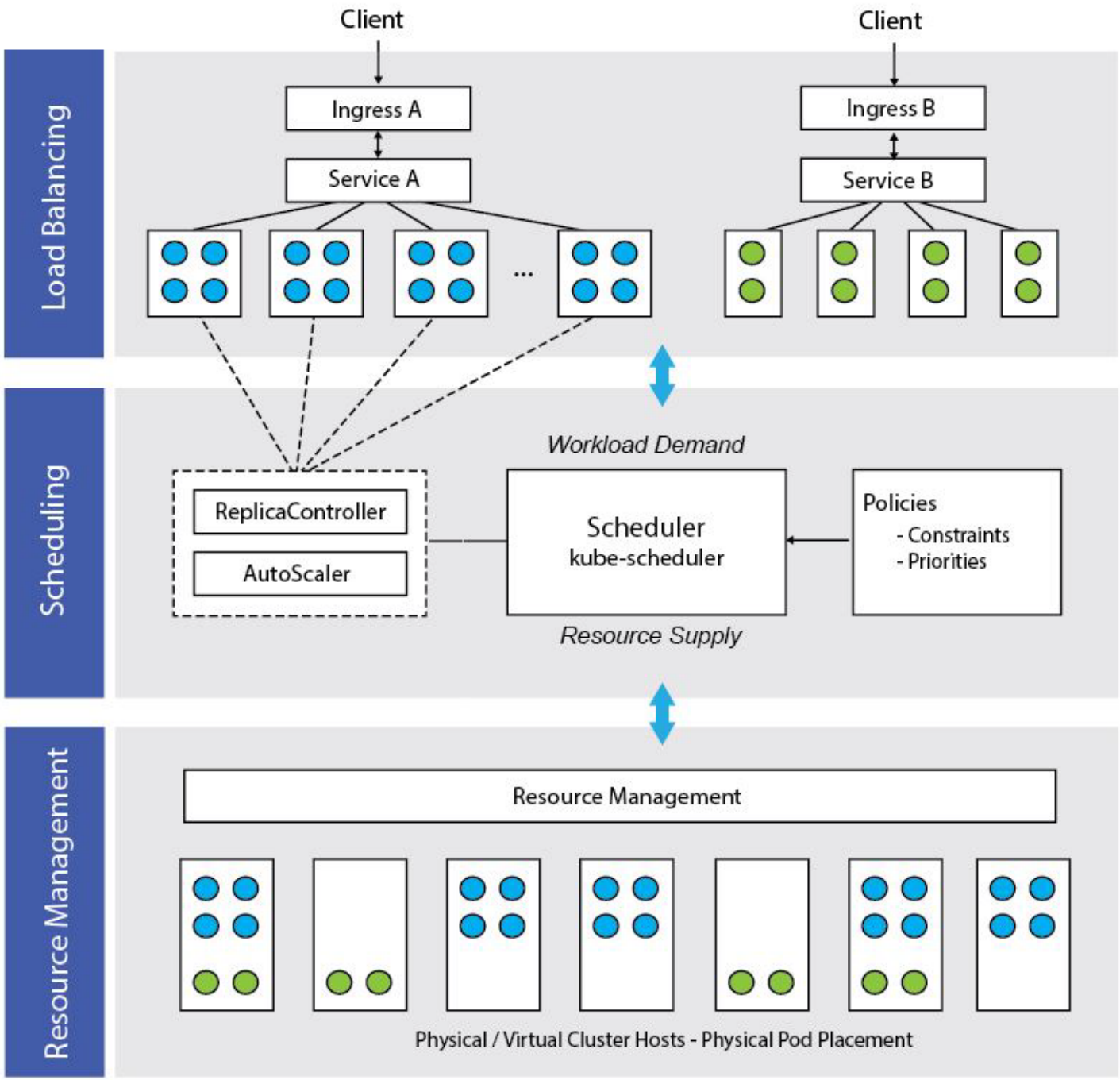
Service



Cluster Autoscaler


A KUBERNETES MICROSERVICE
DEPLOYMENT
CRONJOB
DAEMON SET
REPLICASET
STATEFUL SET
JOB
REPLICATION CONTROLLER
POD
SERVICE
HORIZONTAL POD AUTOSCALER
CONTAINER (YOUR CODE)
POD DISRUPTION BUDGET
INGRESS
VOLUME
CONFIGMAP
PERSISTENT VOLUME CLAIM
SECRET
@BIBRYAM

kubernetes



kubernetes

Microsoft Azure

Google Cloud

DigitalOcean

amazon web services

On Premise

kubernetes

Helm is a package manager for Kubernetes

Isn't the Docker image the package?
Yes, but typically when deploying an entire application it can consist of several containers (the container landscape of your application) with specific properties and "wiring".

Aren't the Kubernetes .yaml files then the way to package the application?
Yes, but typically you would want to have different sets of files with different configurations for different environments (e.g. for test and production environments you would have differen namespaces, different url's, different db connectionstrings, different scaling, etc.).

So how does Helm help?
Helm is

- a CLI
- a convention for how to organize .yaml files (a folder structure that can be zipped called a "chart")
- a templating language to make your .yaml files configurable

Helm illustration

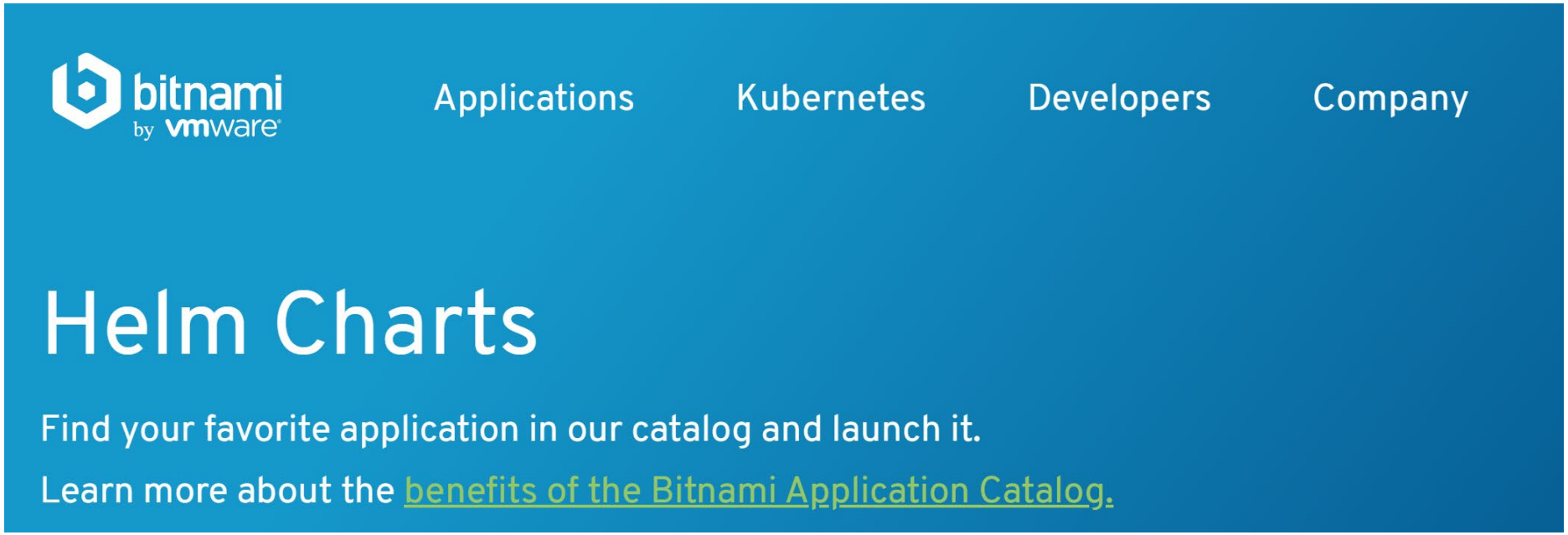
kubernetes

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  values.schema.json  # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/             # A directory containing any charts upon which this chart depends.
  crds/               # Custom Resource Definitions
  templates/          # A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

```yaml
demo:
  environment: staging
  url: demo-api-staging.goca-training.be
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: { .Values.demo.environment }
  name: demo-backend
spec:
  rules:
    - host:  { .Values.demo.url }
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              service:
                name: demo-backend
                port:
                  number: 80
  ingressClassName: nginx
```

```
helm install charts/demo --set demo.environment=production --set demo.url=demo-api.goca-training.be
```


bitnami
by vmware
Applications
Kubernetes
Developers
Company
Helm Charts
Find your favorite application in our catalog and launch it.
Learn more about the benefits of the Bitnami Application Catalog.

Demo time

- Deploying a MongoDB cluster in your Kubernetes cluster with Helm

Istio

kubernetes

Istio is a Service Mesh

kubernetes

Why

A service mesh enables developers to separate and manage service-to-service communications in a dedicated infrastructure layer. As the number of microservices involved with an application increases, so do the benefits of using a service mesh to manage and monitor them.

Concepts

Traffic Management

Deploy capabilities like inter-service routing, failure recovery and load balancing.

Observability

Provide an end-to-end view of traffic flow and service performance.

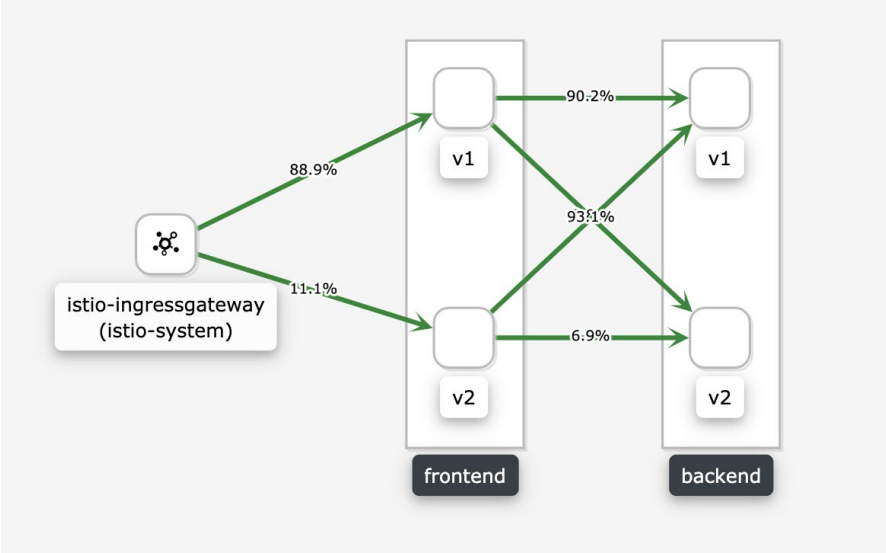
Security

Engage encryption, role-based access, and authentication across services.

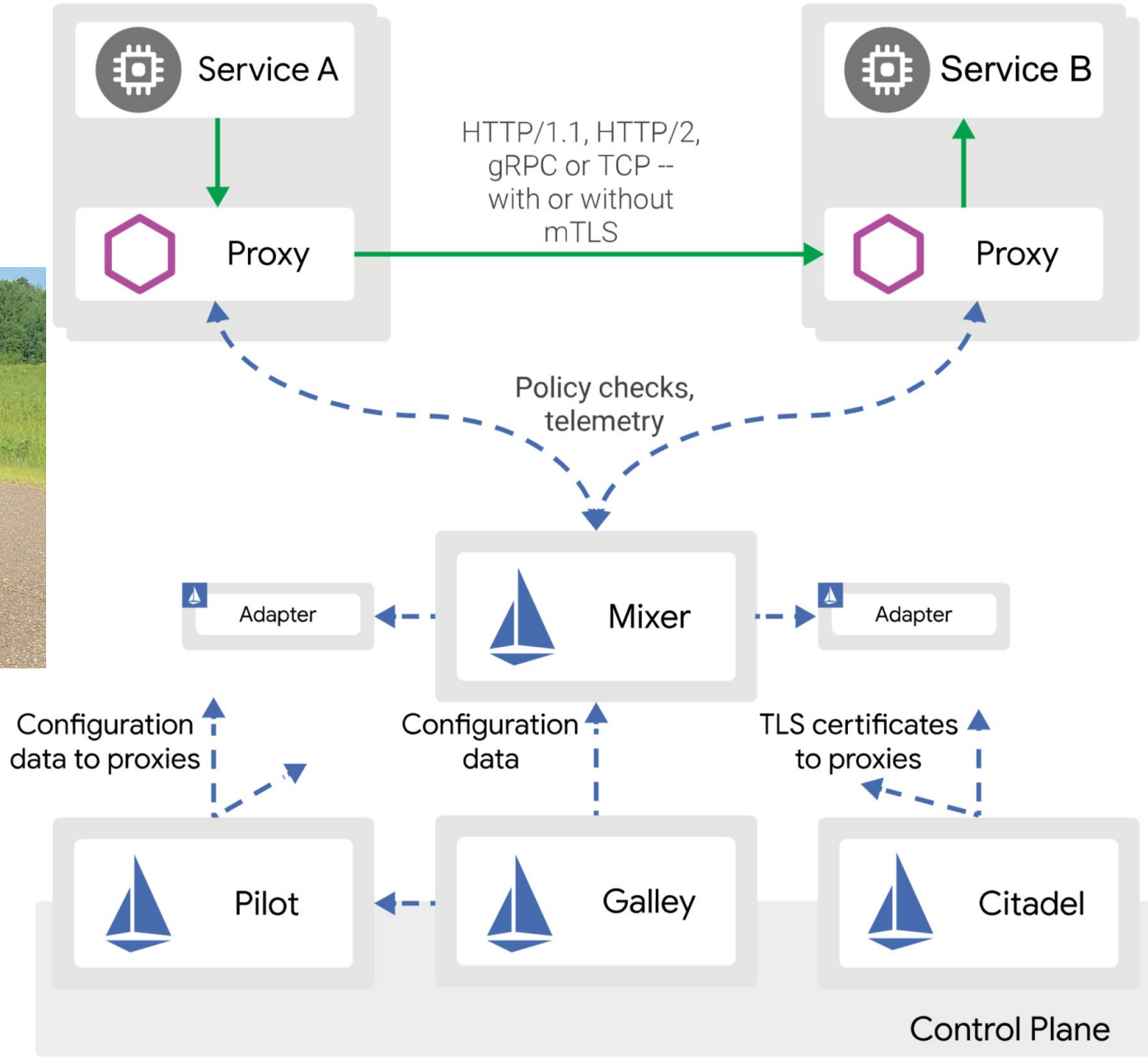
A service mesh tackles microservice challenges

kubernetes

- Security
- Canary deployments
- A/B testing
- Retries
- Rate limiting
- Fault injection
- Policy management
- Telemetry



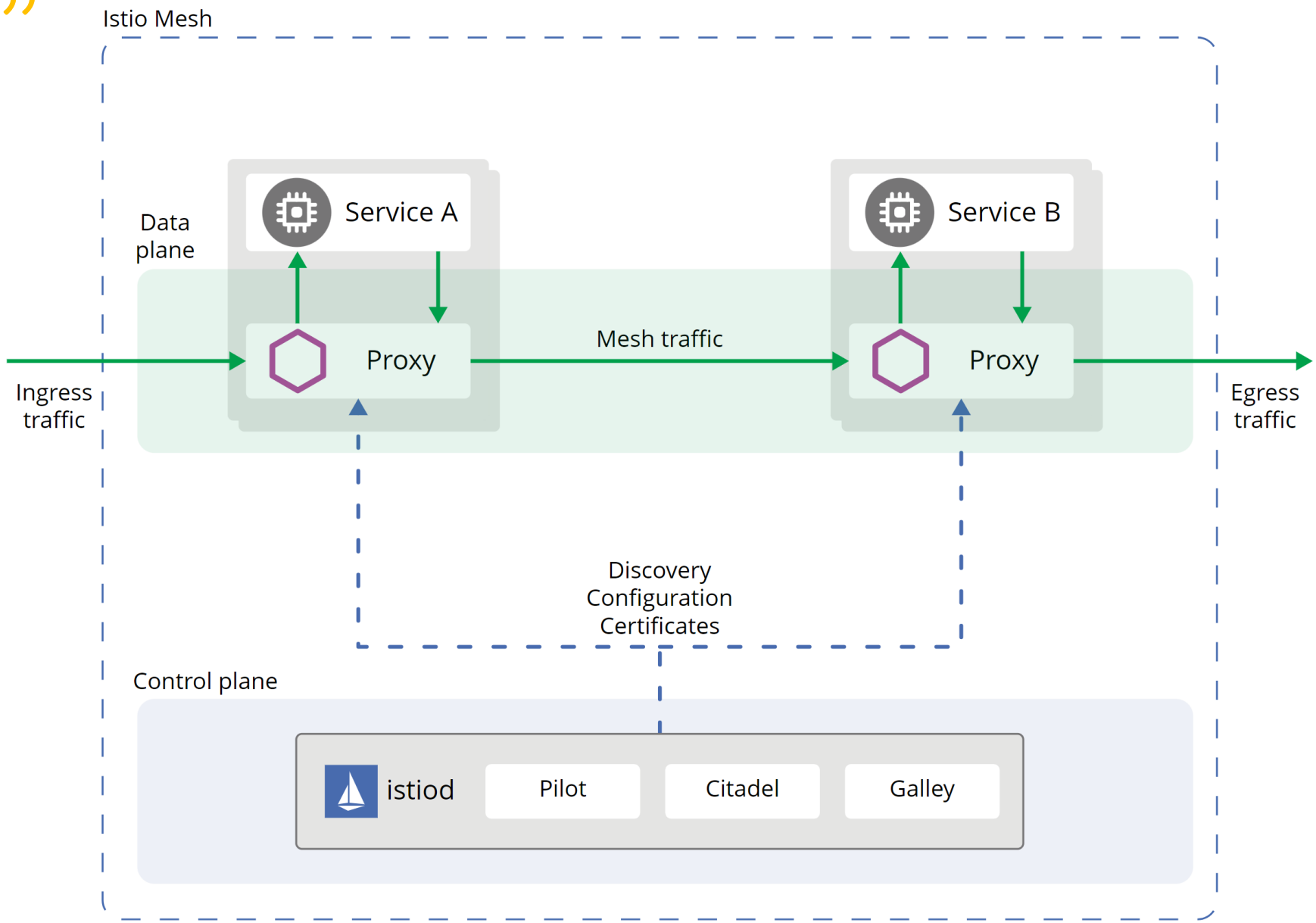
“Classic” Istio



kubernetes

“New” Istio

kubernetes



Istio Architecture

Scratch the surface...

- Ingress
- Traffic shaping
- Chaos testing
- Observability

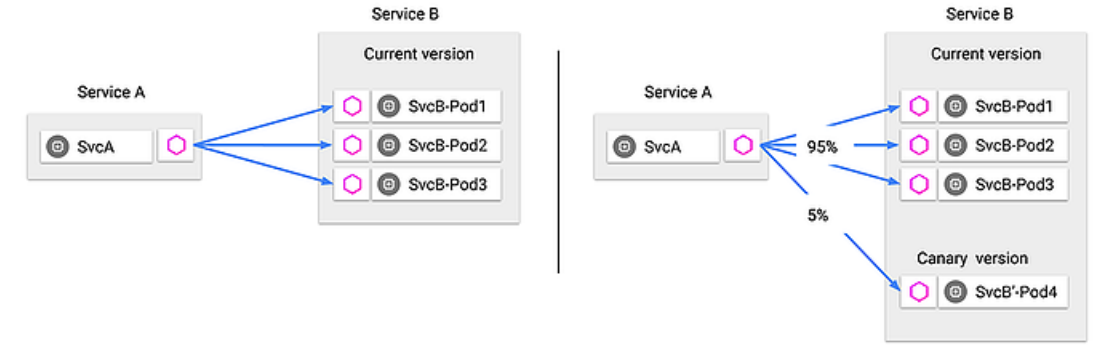
Demo time

- Istio installation
- Istio ingress

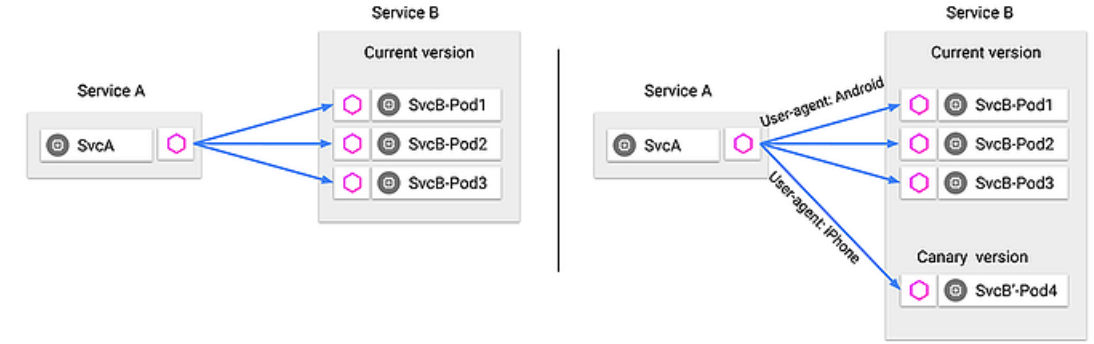
Traffic shaping

kubernetes

Canary release



Traffic splitting decoupled from infrastructure scaling - proportion of traffic routed to a version is independent of number of instances supporting the version



Content-based traffic steering - The content of a request can be used to determine the destination of a request

Dark release

Demo time

- Dark deployment
- Canary deployment

Chaos testing

kubernetes

Benefits of Chaos testing

Below are the key benefits of Chaos testing

Benefits of Chaos Testing	
Five-Nines availability	One of the key benefits of chaos engineering is the very high availability of the system for its end users. Five-Nine availability means the system is up 99.999%. This means there are very less chances of system outages.
Financial profits	Even a very small outage can cause companies to lose millions of dollars. With chaos testing promising to keep the system up, companies are eying at increasing revenues.
Better disaster recovery plan in place	Chaos testing is a way to proactively eliminate, or at least reduce, the frequency and severity of any system disaster. The teams are more equipped to handle those, and therefore have better plans in place. The plans get better with more disasters avoided or recovered
Efficient coding	Since engineers know that their code will be tested for Chaos testing, they are challenged to write better codes to ensure the final system is as resilient as possible. They start thinking out of the box and bring innovative ideas into place.

Demo time

- Fault injection

Observability

Demo time

- Install the observability stack
- Inspect the dashboards

kubernetes

Istio

There's more...

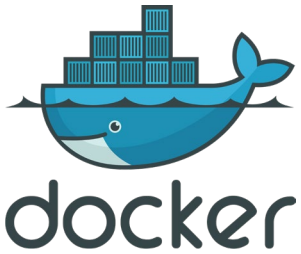
Istio / Tasks

https://microk8s.io/docs/install-raspberry-pi

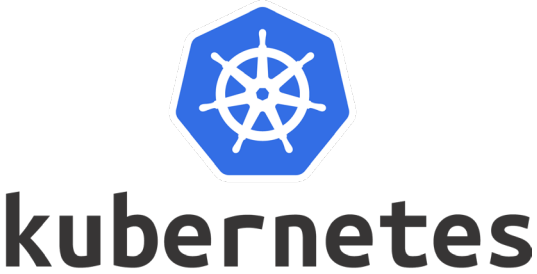
https://docs.k3s.io/advanced#raspberry-pi

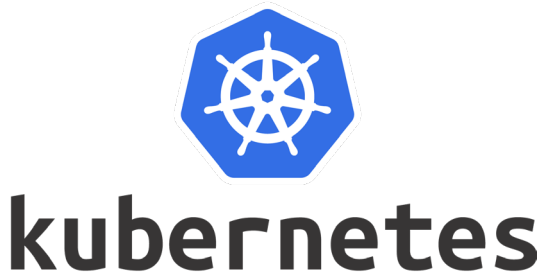
Local Kubernetes DevOps pipeline

Tilt | Kubernetes for Prod, Tilt for Dev



Evening 1
24/11/2022





Evening 2
22/12/2022

Debugging