



# Storage Classes and Ingress



## Table of Contents



- ▶ Ingress
- ▶ StorageClass



1

# Ingress



## Ingress

CLARUSSHOP

StoreFrontUI

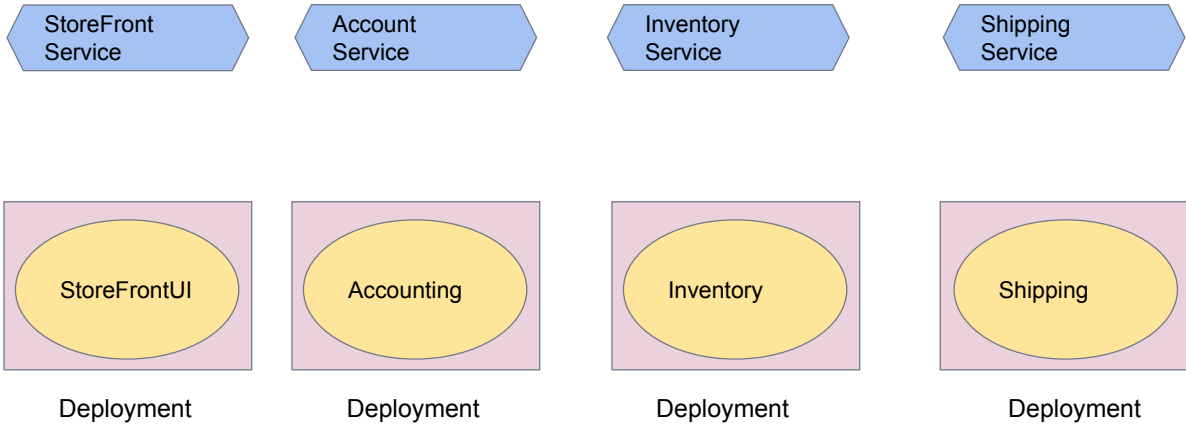
Accounting

Inventory

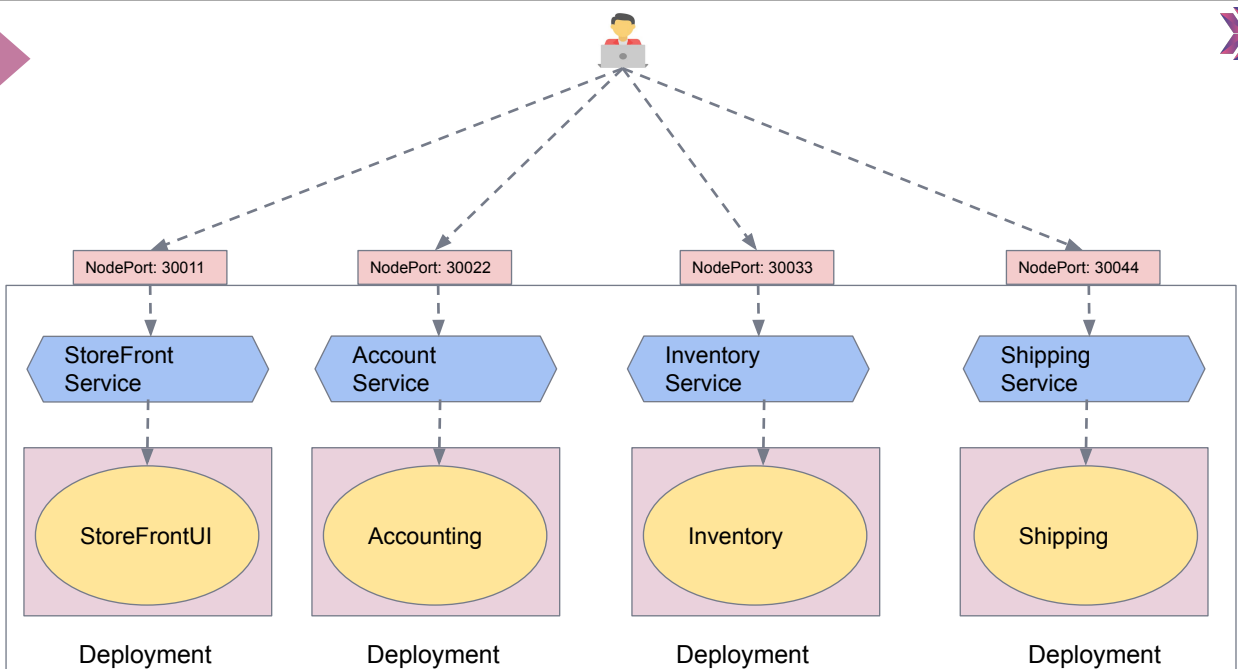
Shipping



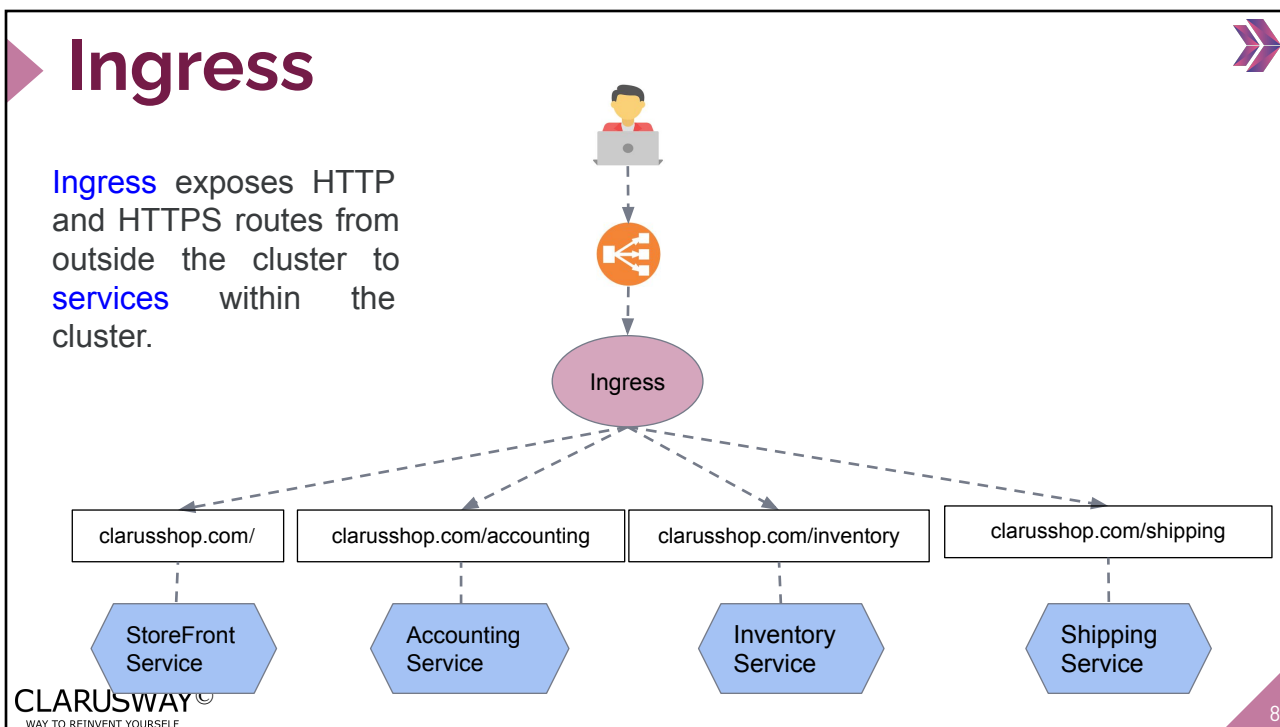
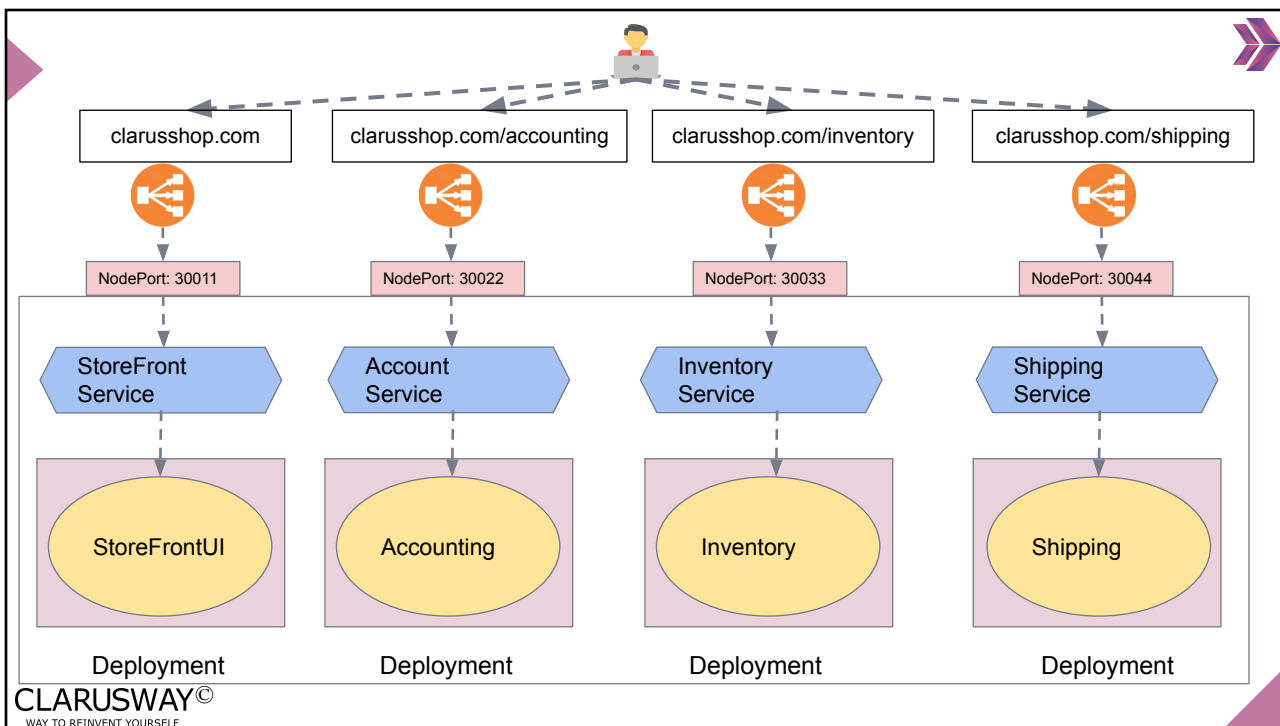
# Ingress



CLARUSWAY©  
WAY TO REINVENT YOURSELF



CLARUSWAY©  
WAY TO REINVENT YOURSELF





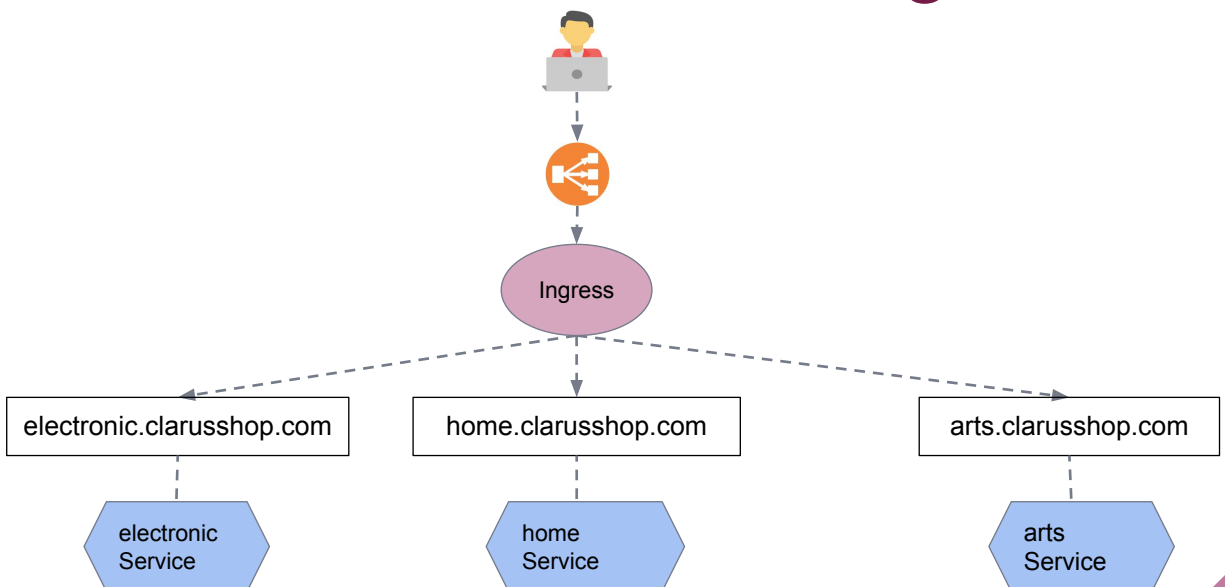
# Ingress

With Ingress, users do not connect directly to a Service. Users reach the Ingress endpoint, and, from there, the request is forwarded to the desired Service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: clarusshop-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: clarusshop-svc
                port:
                  number: 80
          - path: /account
            pathType: Prefix
            backend:
              service:
                name: account-svc
                port:
                  number: 80
```



# Name based virtual hosting



# Name based virtual hosting

Name-based virtual hosts support routing HTTP traffic to multiple host names at the same IP address.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: clarusshop-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - host: home.clarusshop.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: clarusshop-svc
                port:
                  number: 80
    - host: art.clarusshop.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: account-svc
                port:
                  number: 80
```

2

## Storage Class



## The interaction between PVs and PVCs

### Provisioning

There are two ways PVs may be provisioned: statically or dynamically.

#### Static

A cluster administrator creates a number of PVs. They carry the details of the real storage, which is available for use by cluster users. They exist in the Kubernetes API and are available for consumption.

#### Dynamic

When none of the static PVs the administrator created match a user's PersistentVolumeClaim, the cluster may try to dynamically provision a volume specially for the PVC. This provisioning is based on **StorageClasses**.

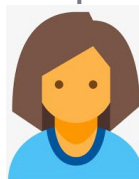


## The interaction between PVs and PVCs

### Static

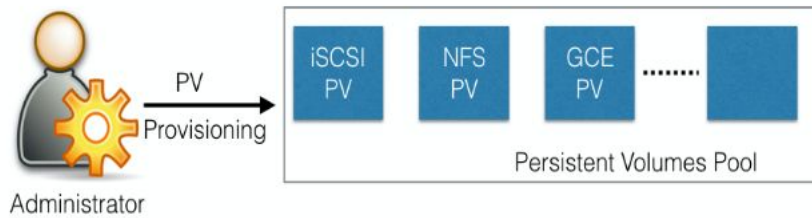


### Dynamic

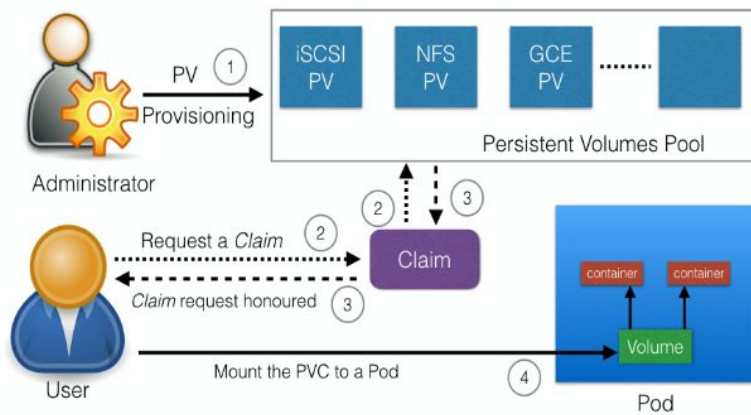




## ► Static PV Provisioning



## ► Static PV Provisioning

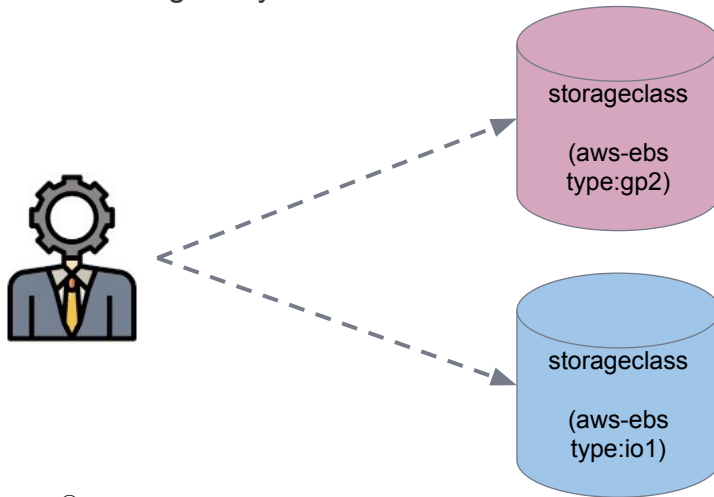




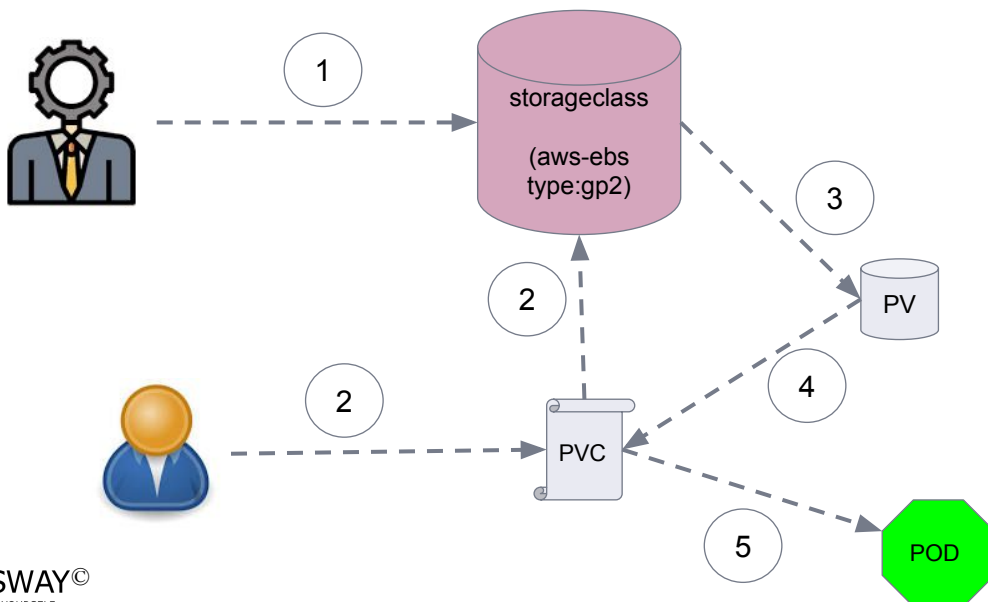


# Dynamic PV Provisioning

A **StorageClass** provides a way for administrators to describe the "classes" of storage they offer.

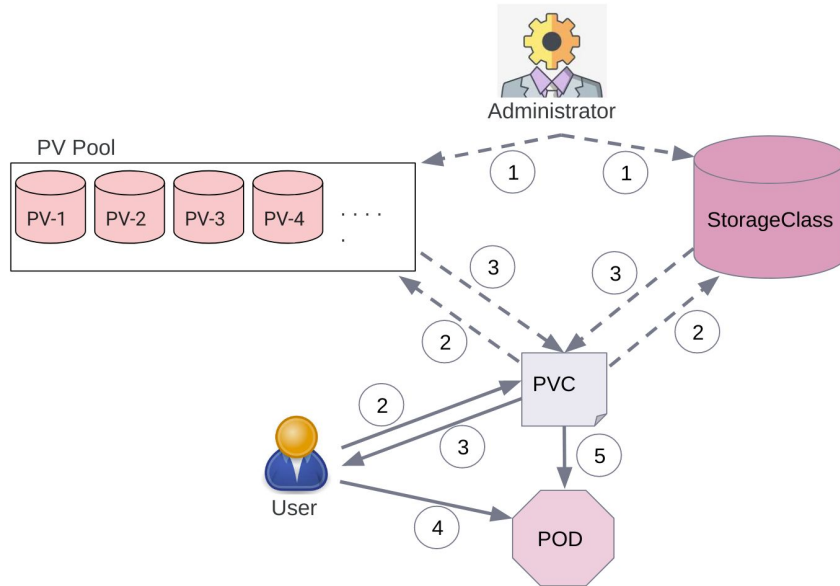


# Dynamic PV Provisioning





# PV Provisioning



# Storage Class

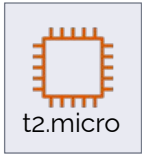
```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: aws-standard
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
  fsType: ext4
```

**Provisioner:** Each StorageClass has a provisioner that determines what volume plugin is used for provisioning PVs.

**Parameters:** Storage Classes have parameters that describe volumes belonging to the storage class. Different parameters may be accepted depending on the provisioner

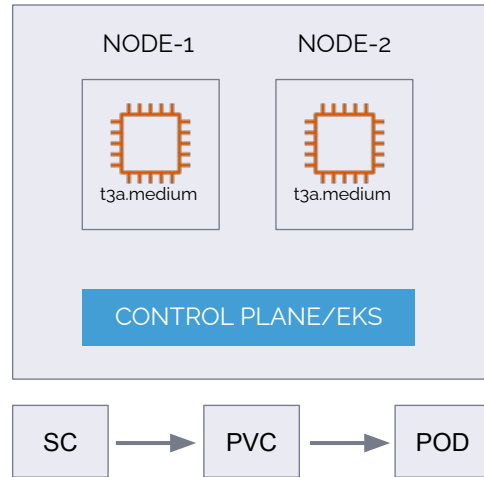


# Hands-on



- kubectl
- eksctl
- aws configure / Role

```
$ eksctl create cluster
```



# THANKS!

## Any questions?

You can find me at:

- james@clarusway.com

