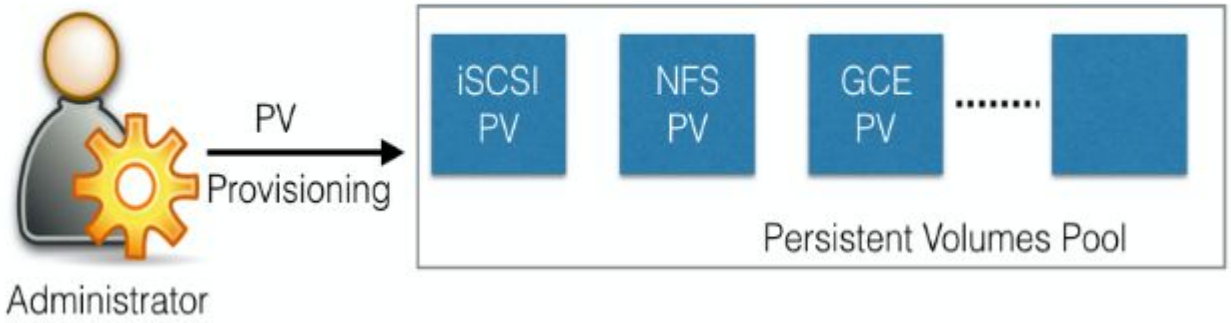
Storage Classes and Ingress

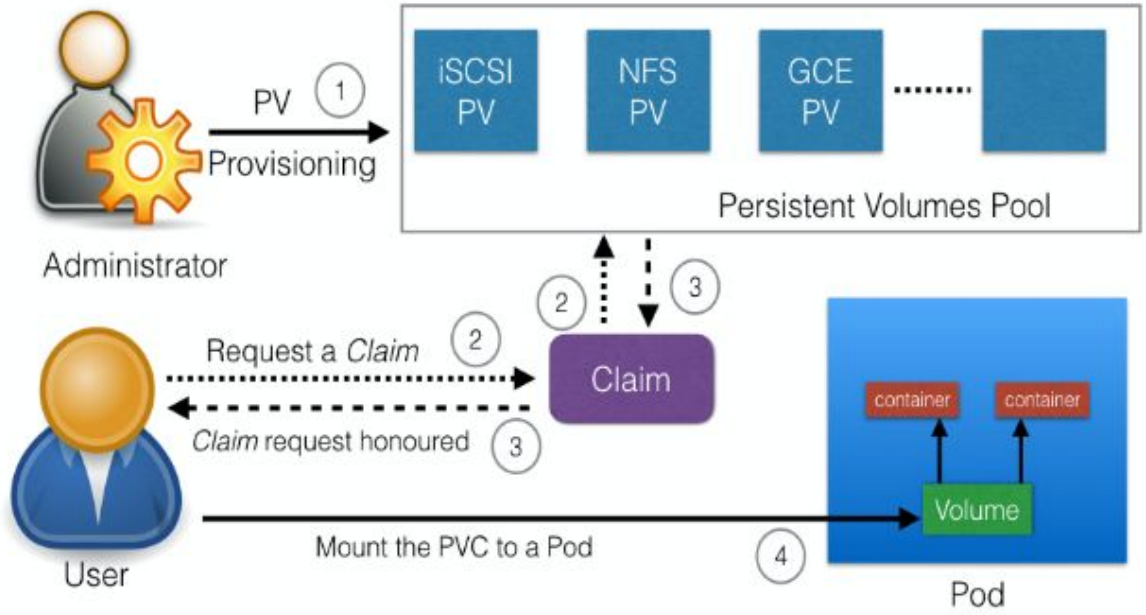
Table of Contents

1 Storage Class

Storage Class

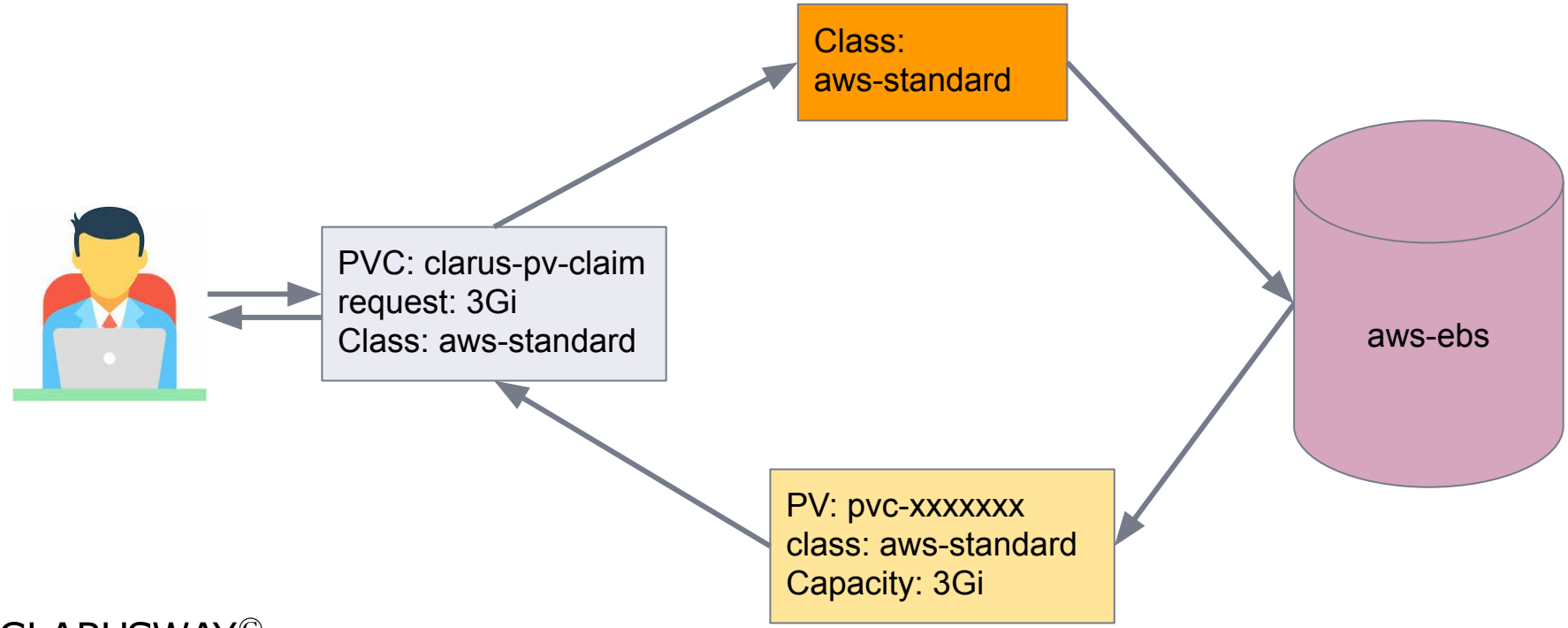


Storage Class



Storage Class

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



Storage Class

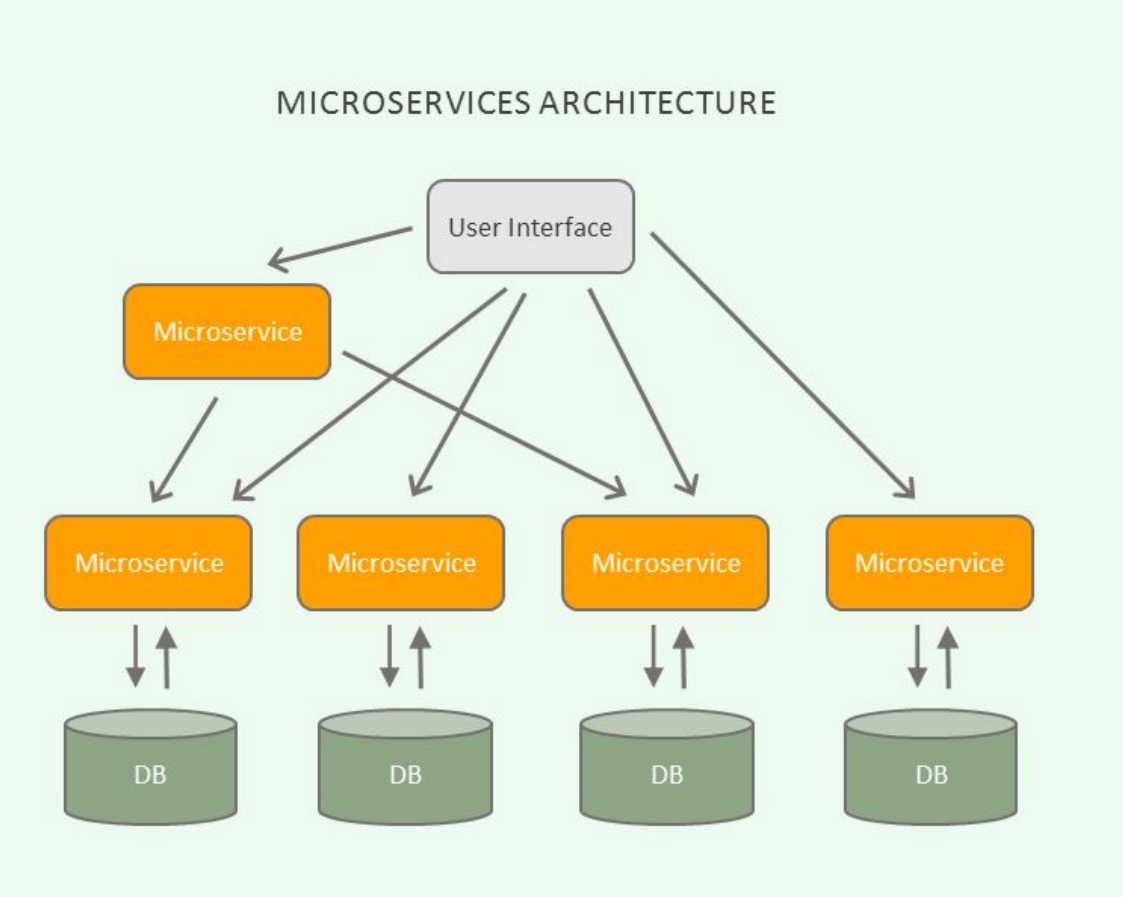
```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: aws-standard
 annotations:
   storageclass.kubernetes.io/is-default-class:
"true"
provisioner: kubernetes.io/aws-ebs
parameters:
 type: gp2
 fsType: ext4
```

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

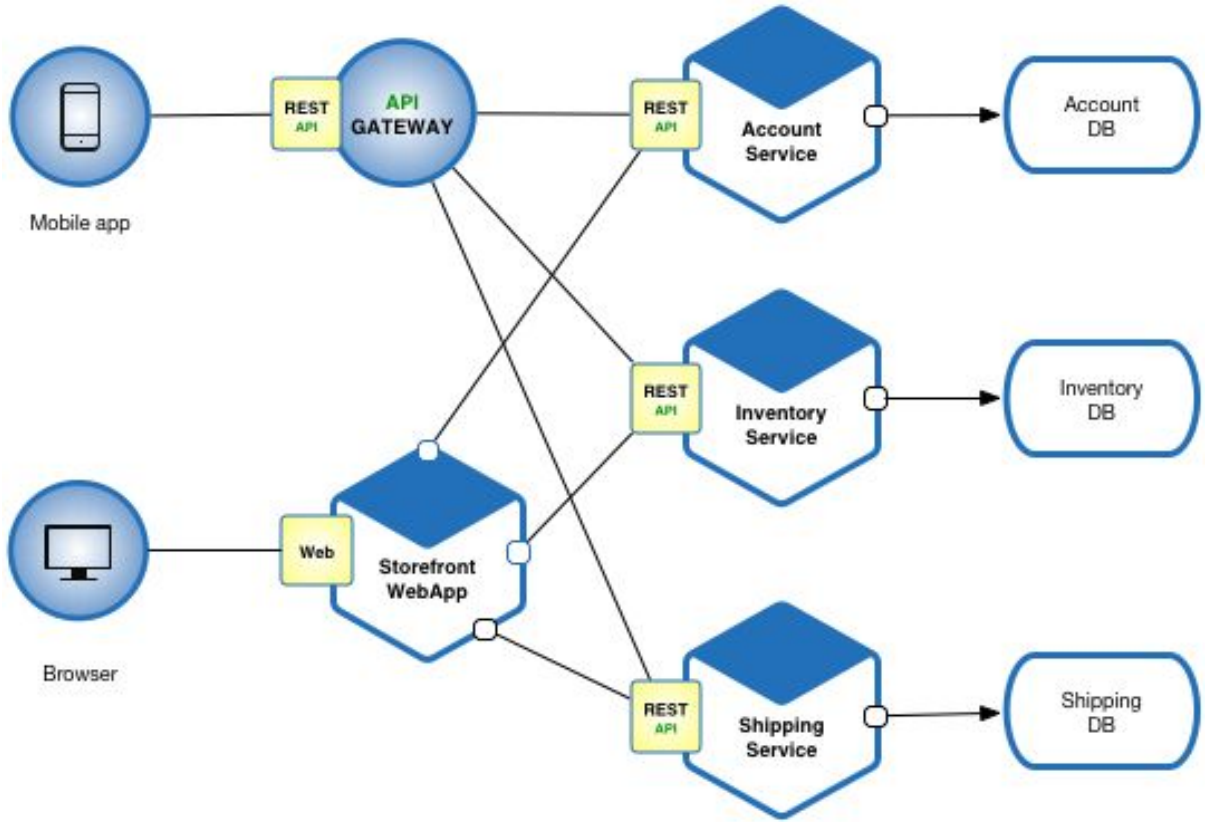
Parameters: Storage Classes have parameters that describe volumes belonging to the storage class. Different parameters may be accepted depending on the provisioner

2 Ingress

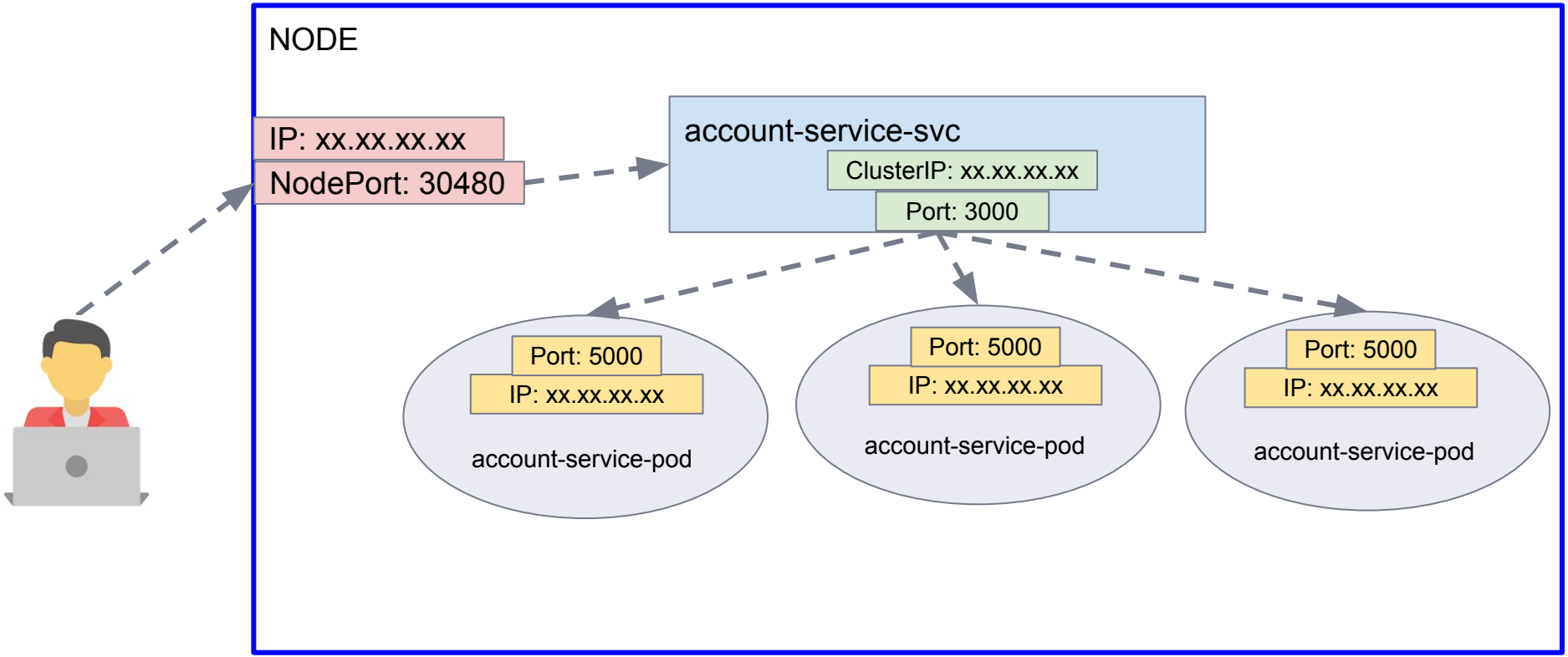
Ingress


MICROSERVICES ARCHITECTURE
User Interface
Microservice
Microservice
Microservice
Microservice
Microservice
DB
DB
DB
DB

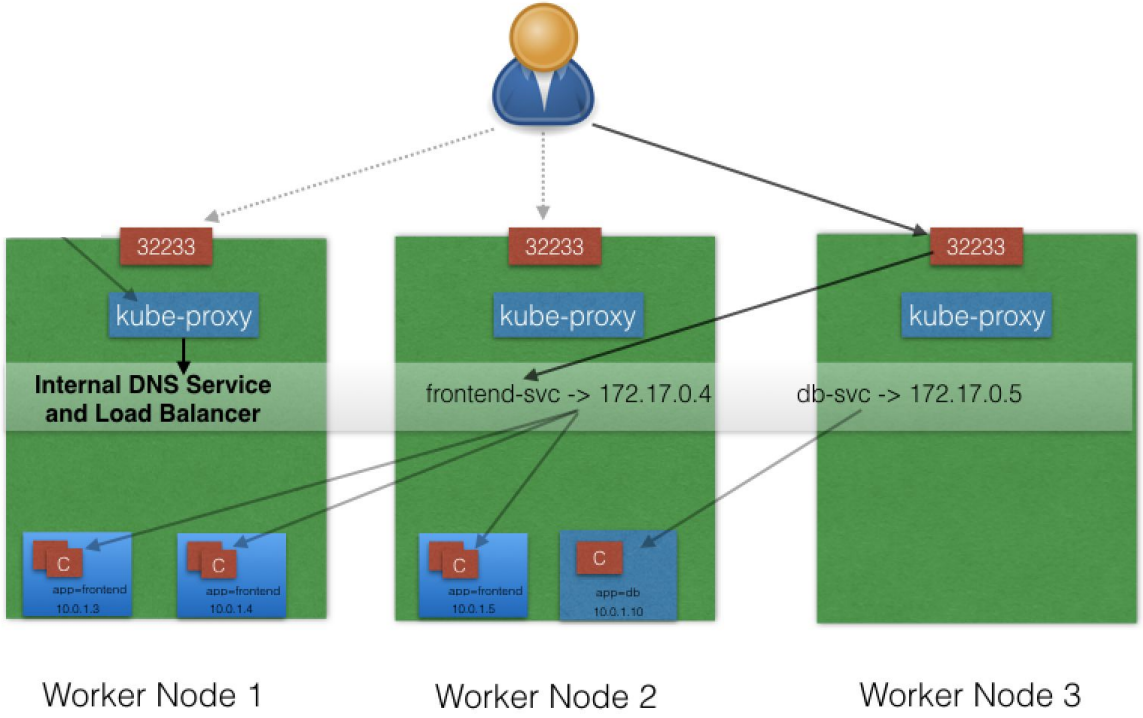
Ingress


Mobile app
REST API
API GATEWAY
REST API
Account Service
Account DB
REST API
Inventory Service
Inventory DB
Browser
Web
Storefront WebApp
REST API
Shipping Service
Shipping DB

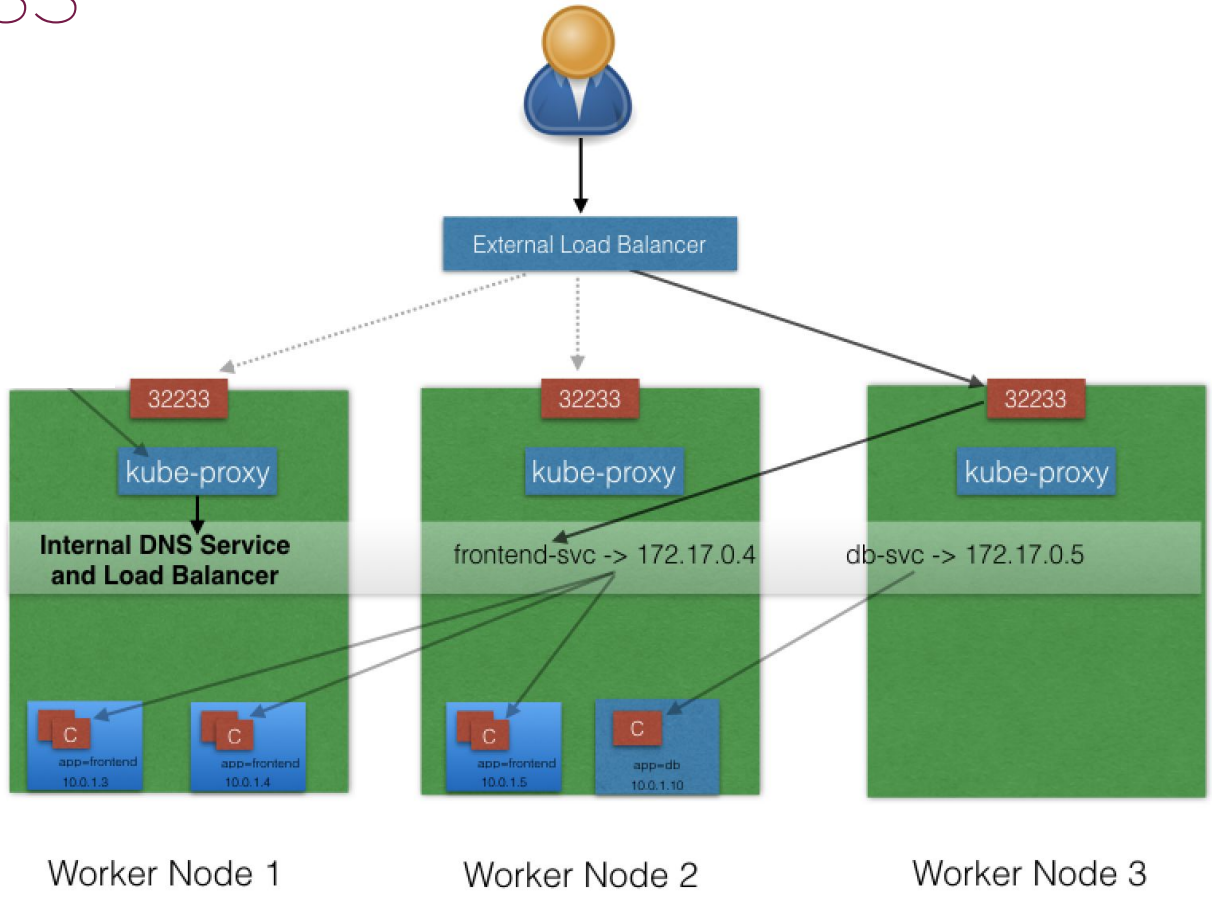
Ingress


NODE
IP: xx.xx.xx.xx
NodePort: 30480
account-service-svc
ClusterIP: xx.xx.xx.xx
Port: 3000
Port: 5000
IP: xx.xx.xx.xx
account-service-pod
Port: 5000
IP: xx.xx.xx.xx
account-service-pod
Port: 5000
IP: xx.xx.xx.xx
account-service-pod

Ingress


32233
kube-proxy
Internal DNS Service
and Load Balancer
32233
kube-proxy
frontend-svc -> 172.17.0.4
32233
kube-proxy
db-svc -> 172.17.0.5
app=frontend
10.0.1.3
app=frontend
10.0.1.4
app=frontend
10.0.1.5
app=db
10.0.1.10
Worker Node 1
Worker Node 2
Worker Node 3

Ingress


External Load Balancer
32233
kube-proxy
Internal DNS Service and Load Balancer
C
app=frontend
10.0.1.3
C
app=frontend
10.0.1.4
Worker Node 1
32233
kube-proxy
frontend-svc -> 172.17.0.4
C
app=frontend
10.0.1.5
C
app=db
10.0.1.10
Worker Node 2
32233
kube-proxy
db-svc -> 172.17.0.5
Worker Node 3

Ingress

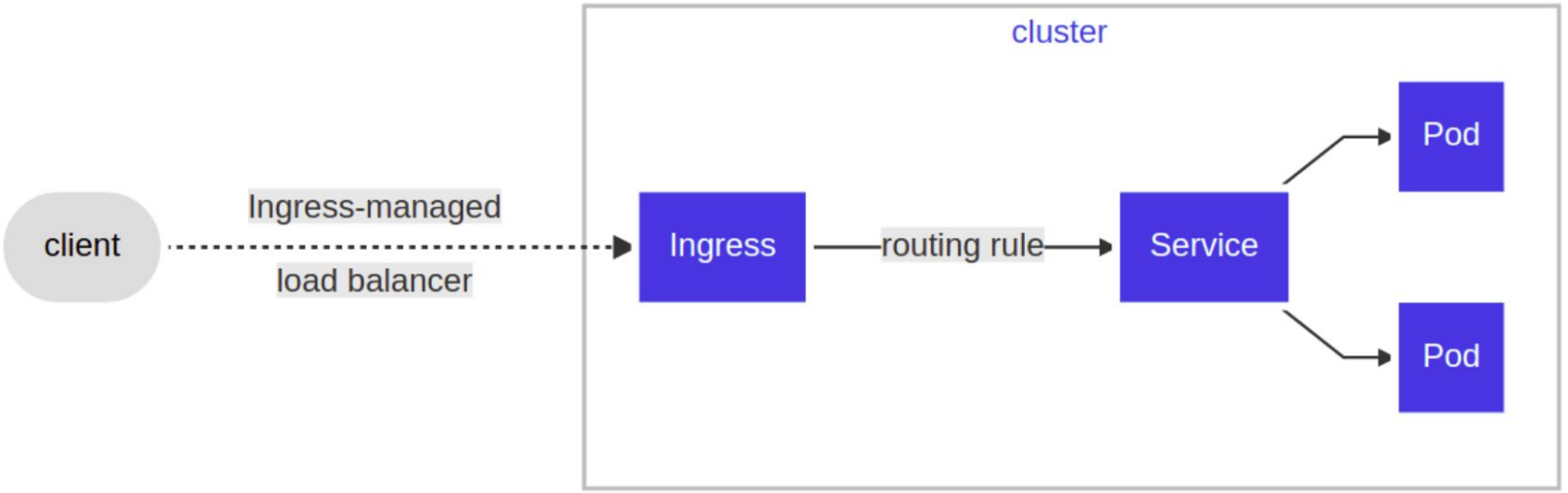
With Services, routing rules are associated with a given Service. They exist for as long as the Service exists, and there are many rules because there are many Services in the cluster. If we can somehow decouple the routing rules from the application and centralize the rules management, we can then update our application without worrying about its external access.

www.clarus-commerce.com

www.clarus-commerce.com/account
www.clarus-commerce.com/inventory
www.clarus-commerce.com/shipping

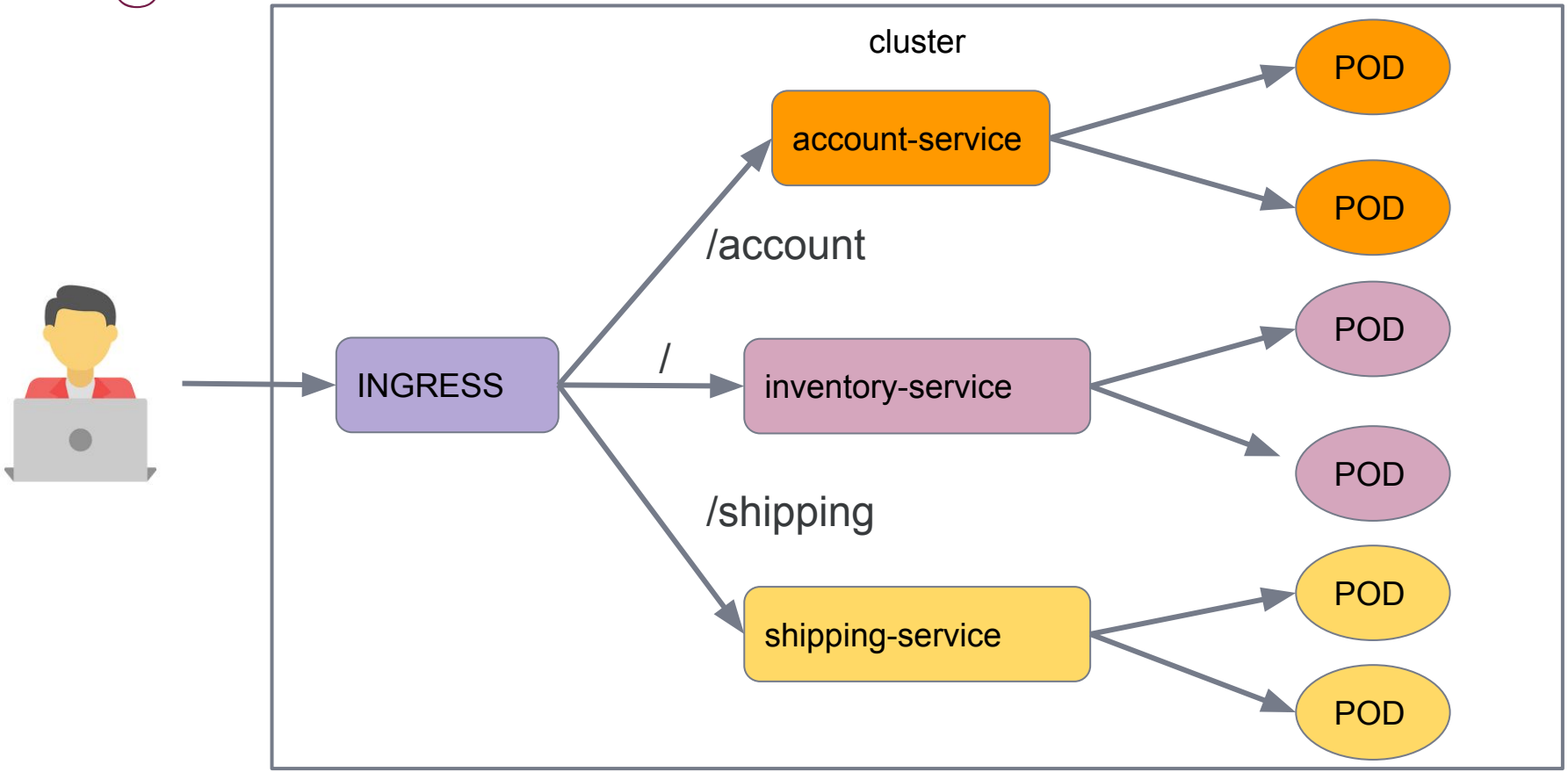
Ingress

"An Ingress is a collection of rules that allow inbound connections to reach the cluster Services."

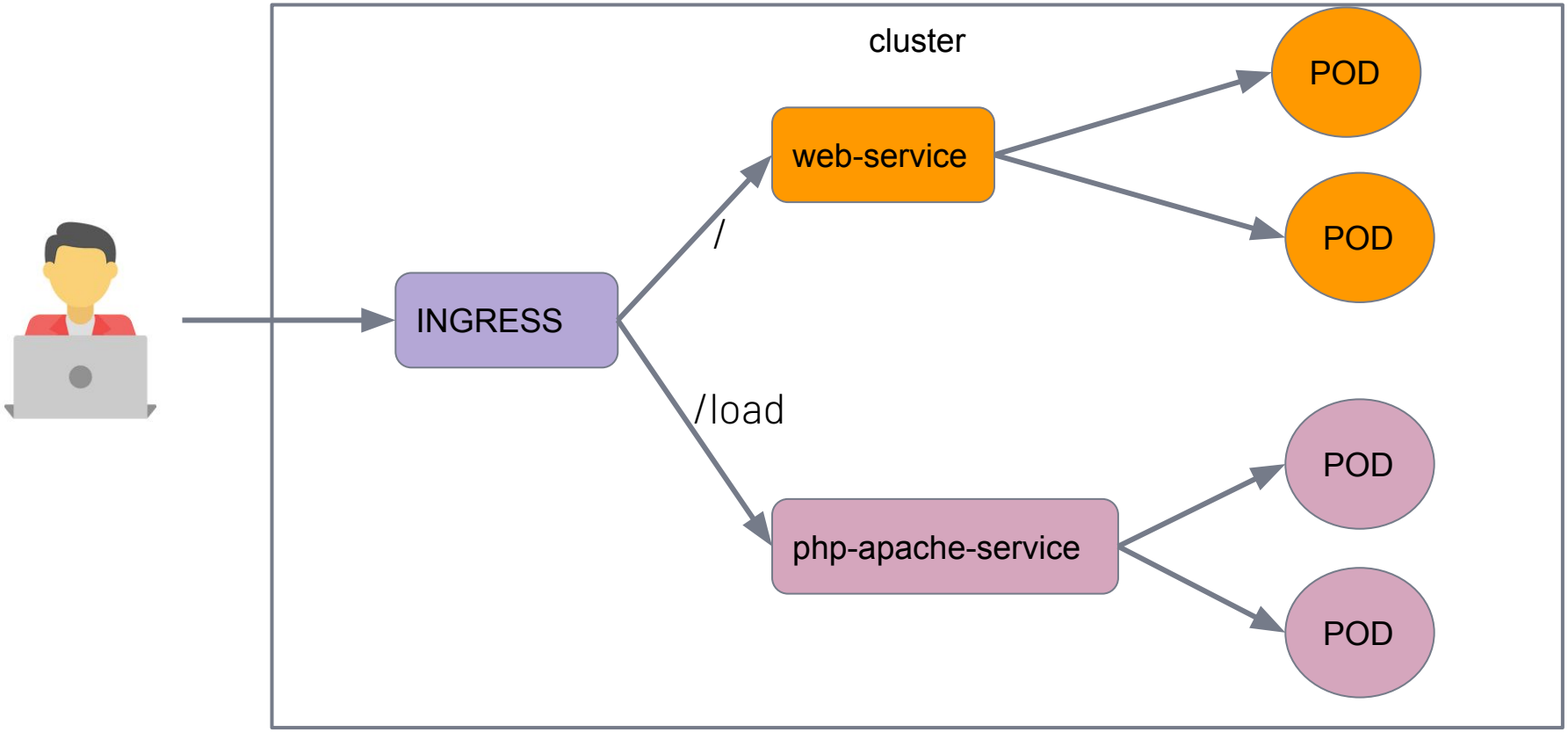


Ingress

www.clarus-commerce.com



Ingress

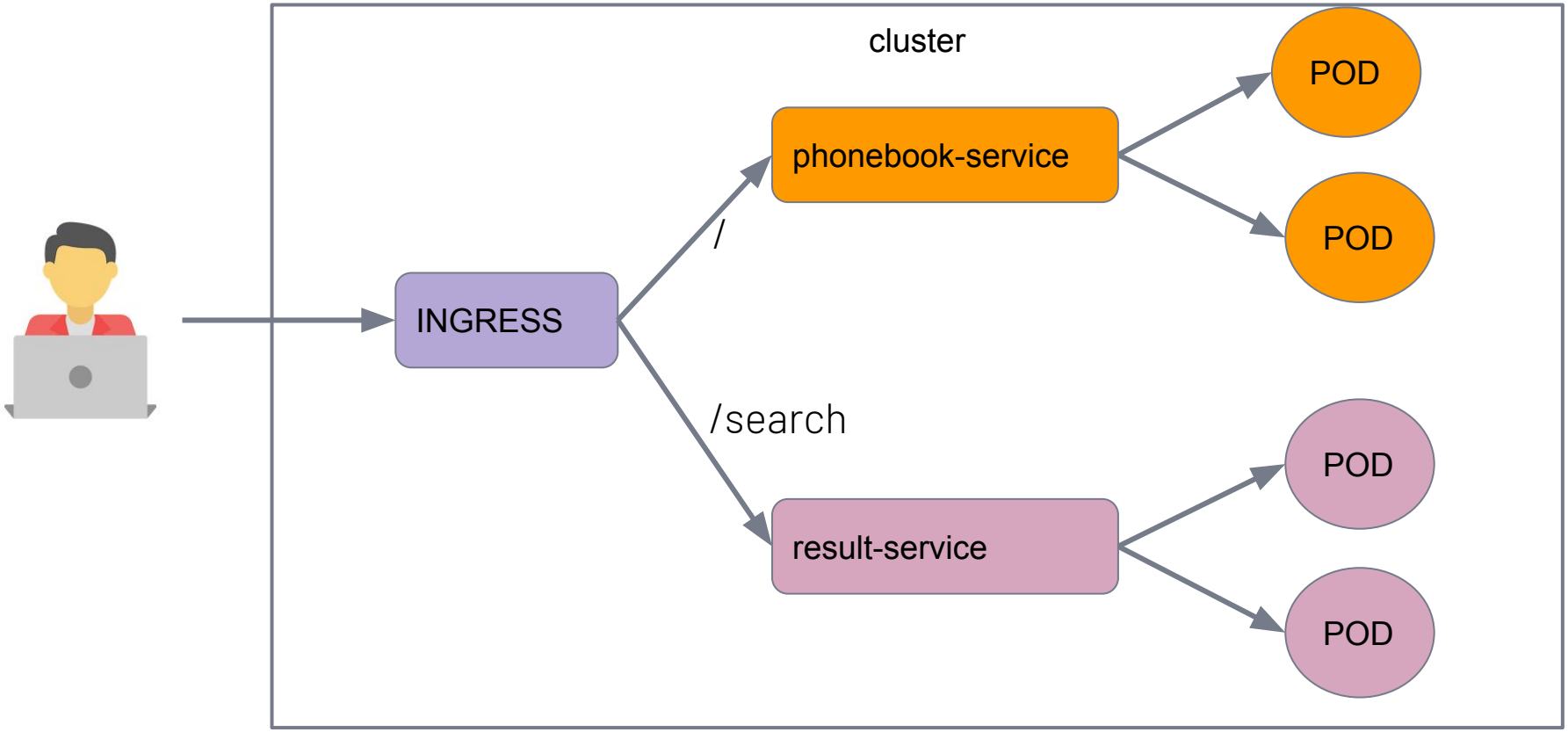

cluster
POD
web-service
POD
/
INGRESS
/load
POD
php-apache-service
POD

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/v1beta1
kind: Ingress
metadata:
 name: ingress-service
 annotations:
   kubernetes.io/ingress.class: 'nginx'
   nginx.ingress.kubernetes.io/use-regex: 'true'
   nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
 rules:
   - http:
       paths:
         - path: /?(.*)
           backend:
             serviceName: web-service
             servicePort: 3000
         - path: /load/?(.*)
           backend:
             serviceName: php-apache-service
             servicePort: 80
```

Ingress


cluster
POD
phonebook-service
POD
/
INGRESS
/search
POD
result-service
POD

THANKS!

Any questions?

You can find me at:

- james@clarusway.com