

# Microservices and Deploying Methodologies

Author: Nho Luong  
Skill: DevOps Engineer Lead



**Certification status**

**NHO LUONG**  
AWS04440050

**HOME**

**PROFILE**

**EXAM REGISTRATION**

**EXAM HISTORY**

**CERTIFICATIONS**

**Certification status**

**Agreement**

**BENEFITS**

**DIGITAL BADGES**

**SUPPORT AND FAQS**

**Certification status**

**Activity**

**Training**

**Plans**

**Challenges**

**Credentials**

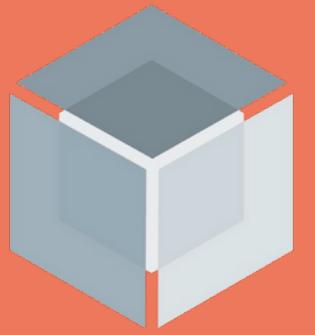
**Q&A**

**Achievements**

**Collections**

**Transcript**





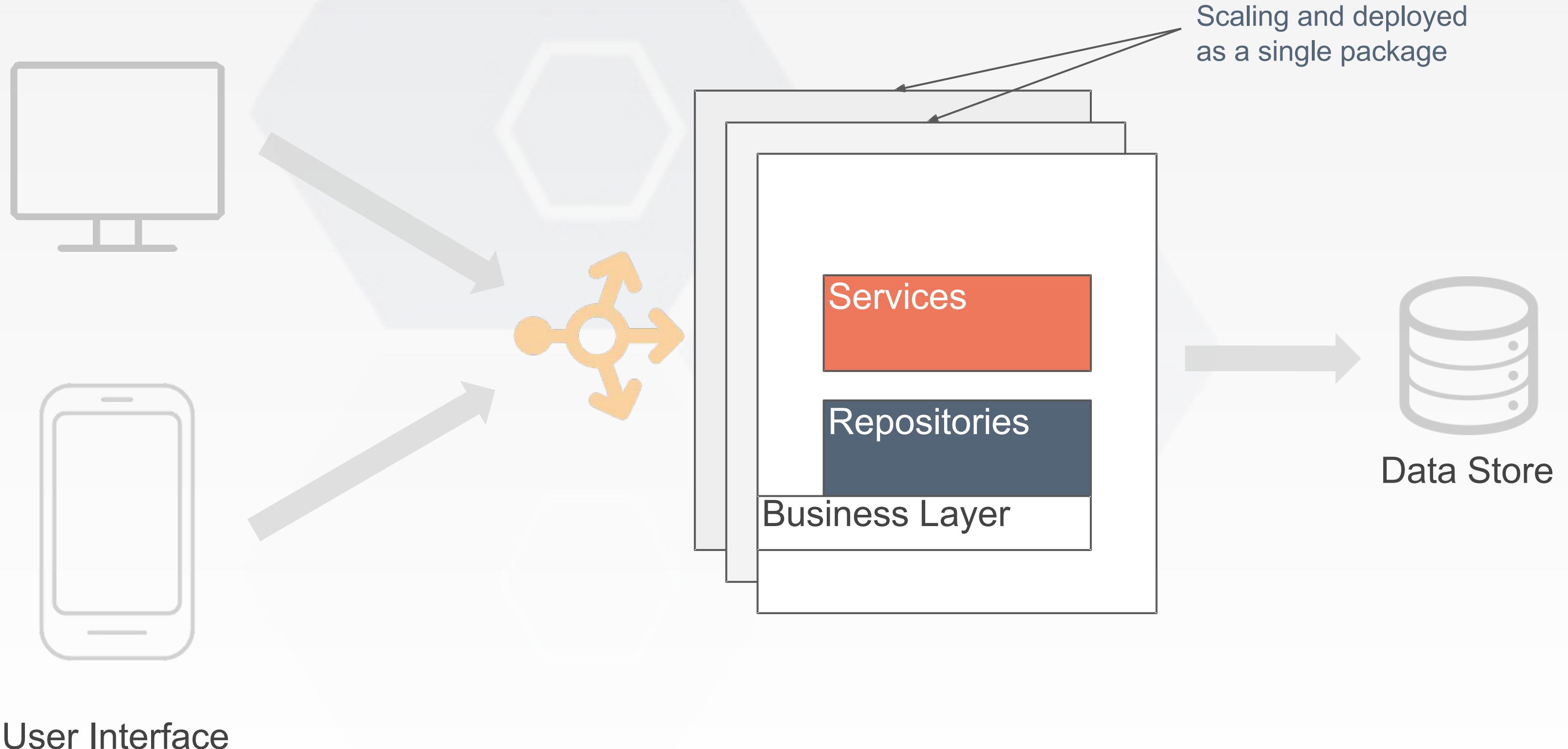
# Monolithic Applications

Traditional way to build enterprise app, and deployed and scaled as a single package

## Main Components:

- Client-side user interface
- Server-side application
  - Design Pattern(Services & Repositories)
- A database server

# Monolithic Application



# Monolithic Application: Example

	<b>Markets</b> <ul style="list-style-type: none"><li>• market_id</li><li>• market_name</li></ul>

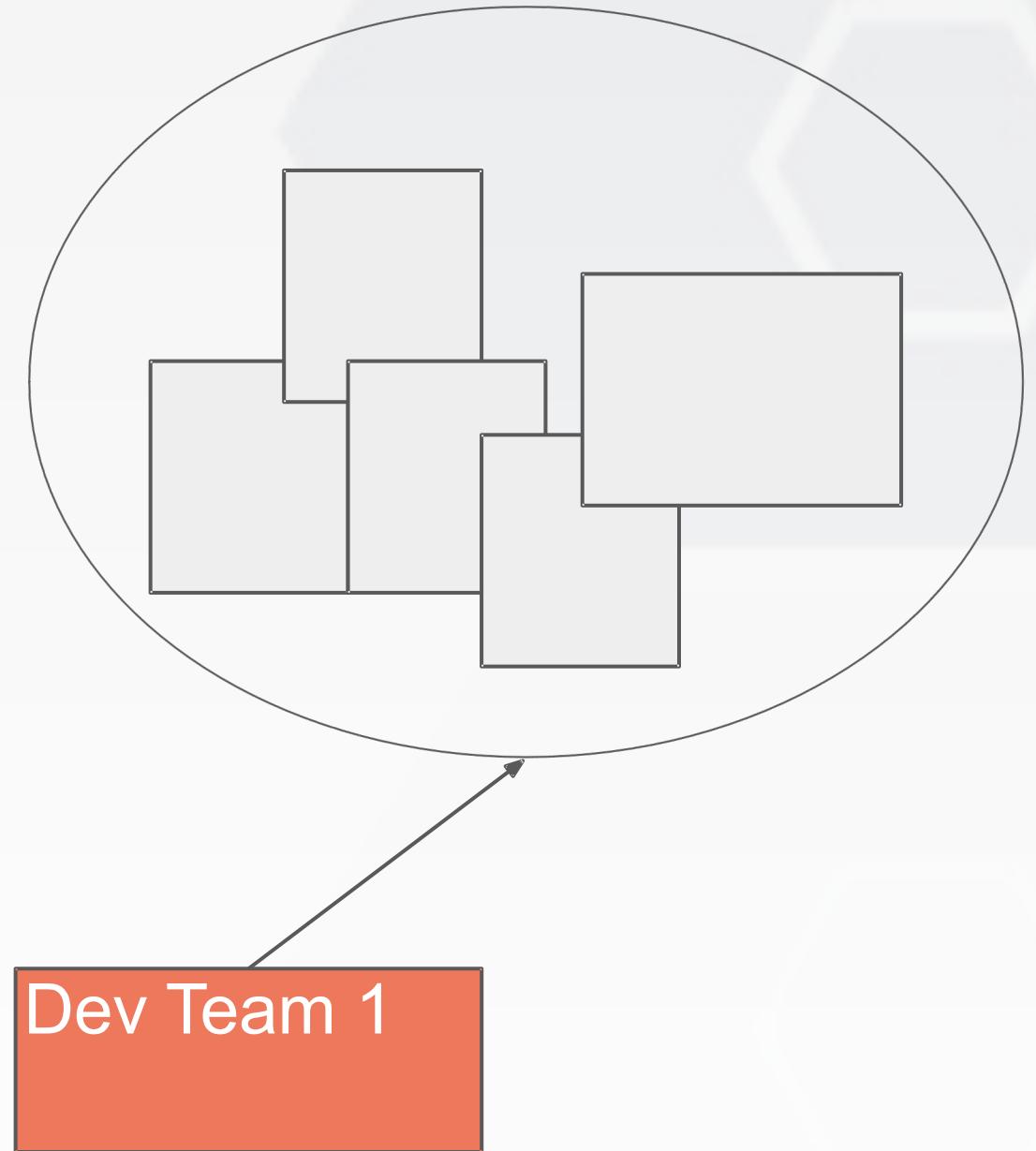
	<b>Products</b> <ul style="list-style-type: none"><li>• product_id</li><li>• product_name</li><li>• product_price</li><li>• market_id</li></ul>

	<b>Users</b> <ul style="list-style-type: none"><li>• user_id</li><li>• user_name</li><li>• market_id</li></ul>

A query to get the list of products and market name of the logged in user say with id=40 under price 300?

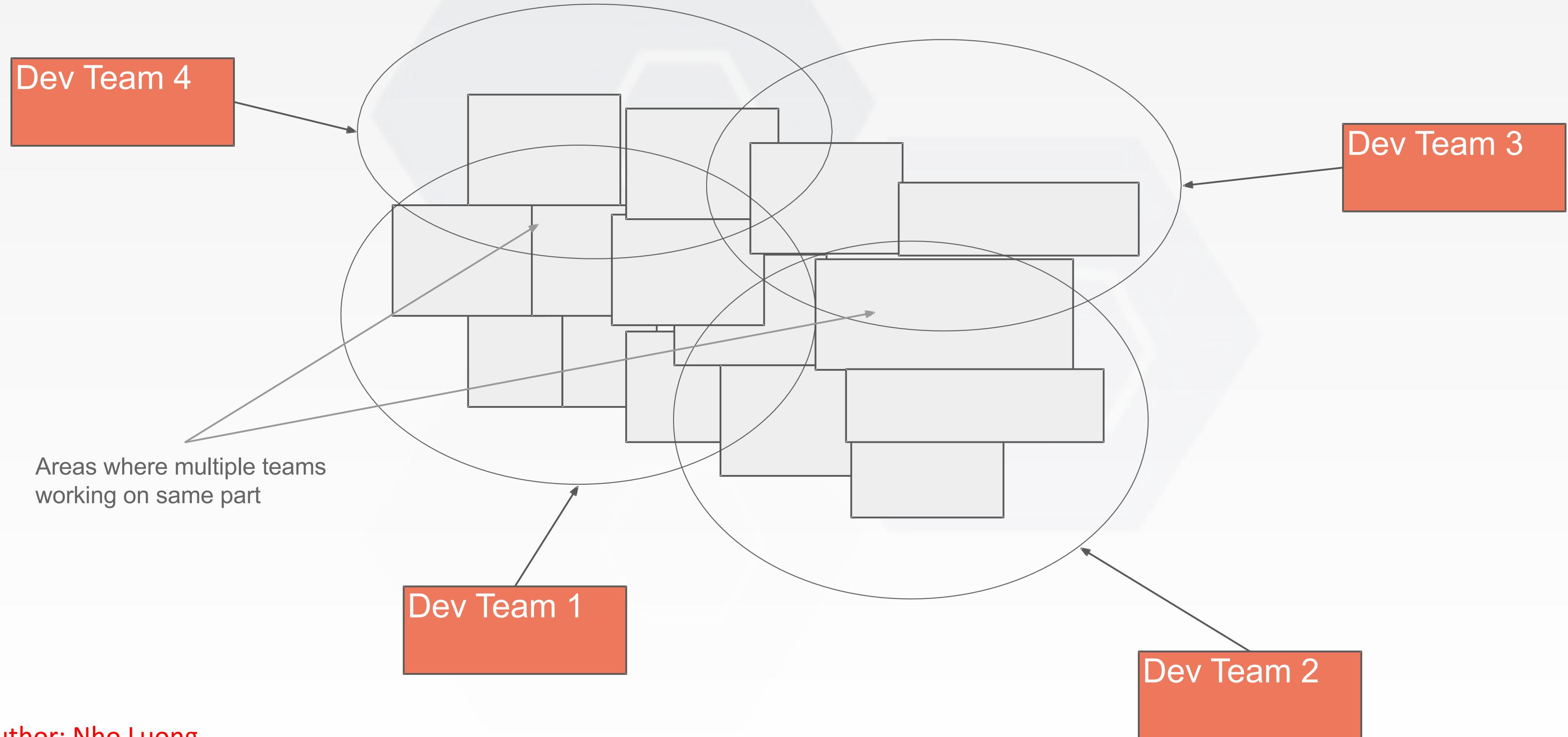
```
SELECT product_name, market_name FROM products
INNER JOIN markets on products.maket_id = markets.id INNER JOIN
users on users.market_id = markets.id WHERE users.user_id = 40
AND products.product_price < 300
```

# Monolithic Application - MVP version

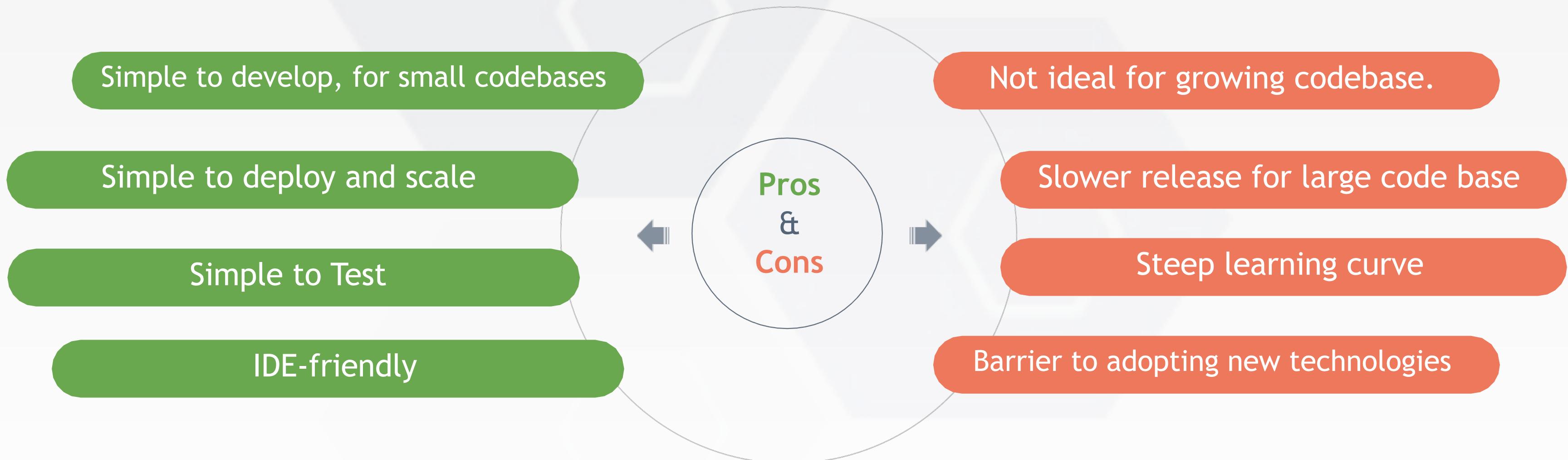


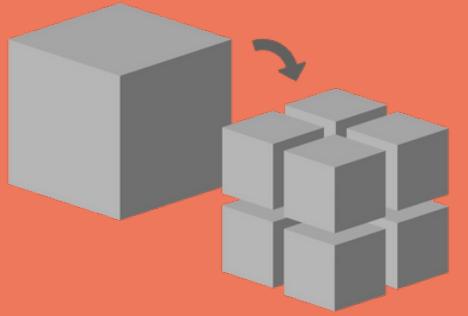
What if a big hit?

# Monolithic Application - After few releases



# Monolithic - Pros / Cons





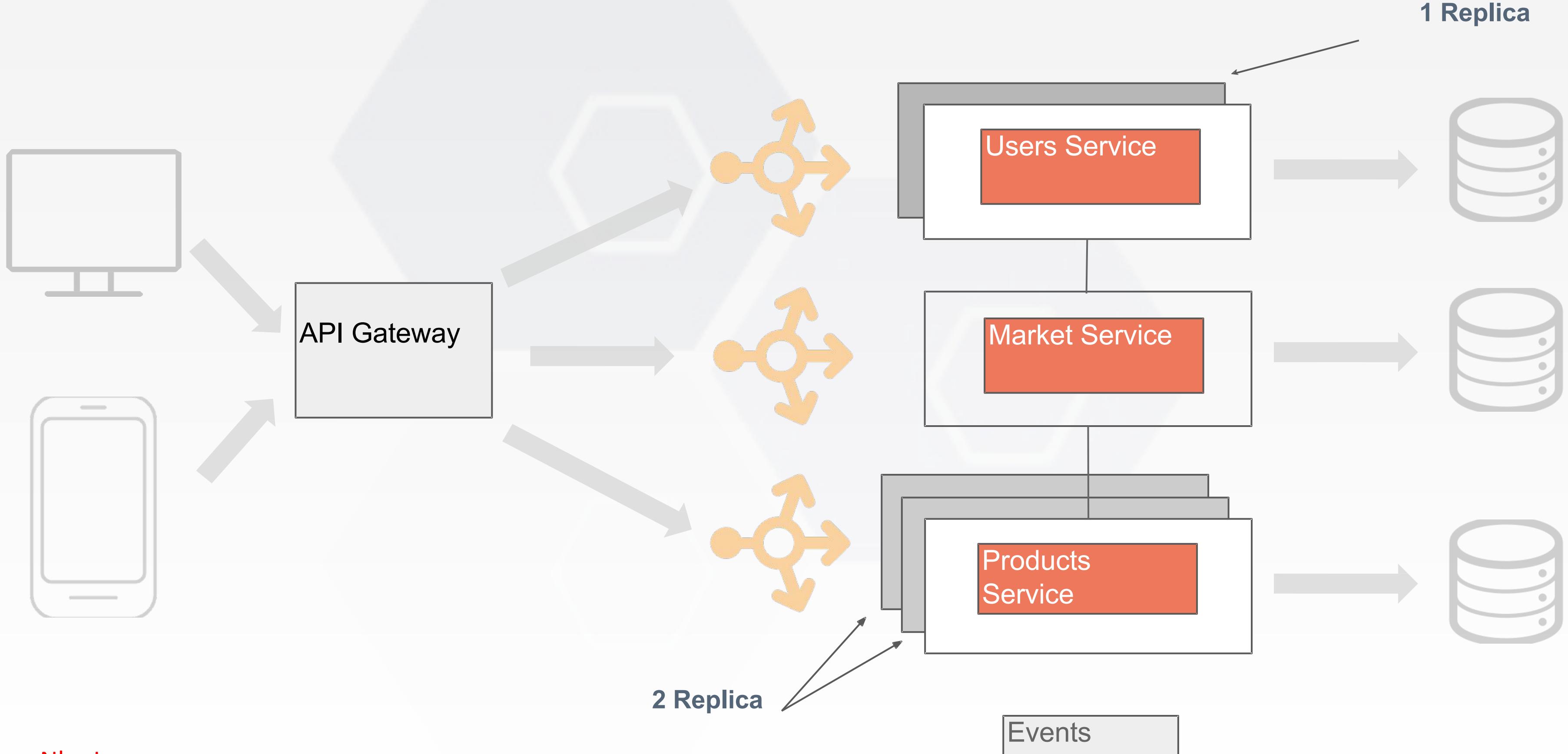
# Micro Services

Suites of independently deployable services

## Design Considerations

- **Organized around business capability**
- **Decentralized Database**
- **Loosely Coupled and Highly Cohesive**
- **Independently deployed**

# Microservice Application



# Microservice Application: Example

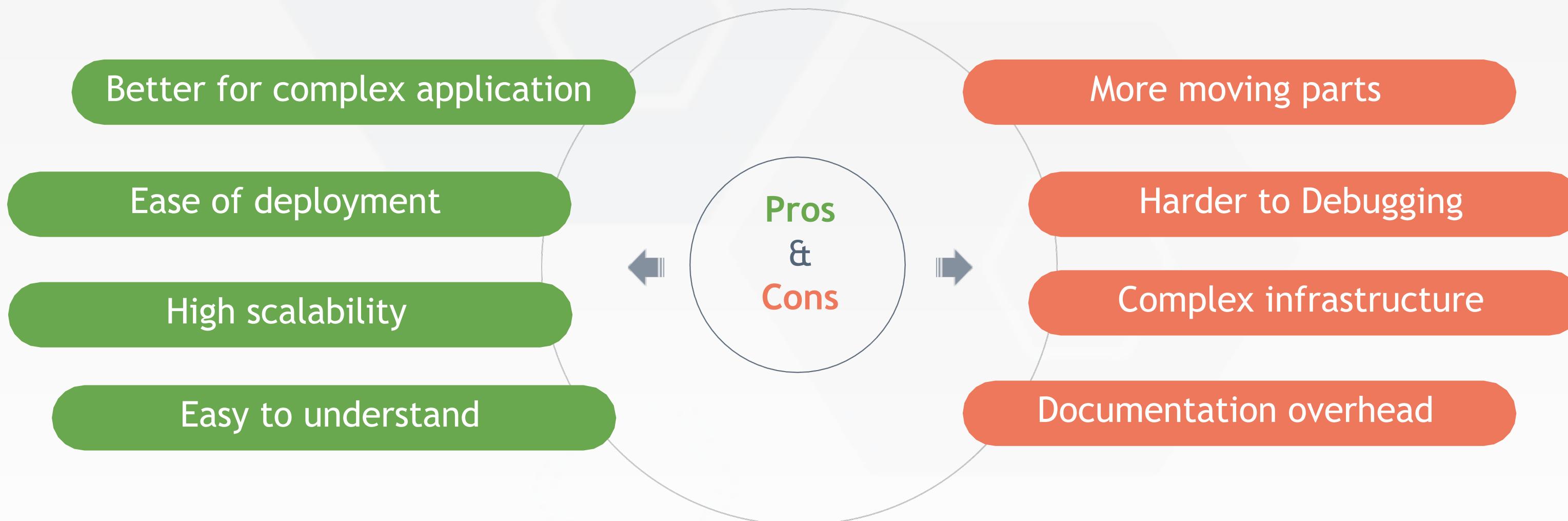
```
markets
{
  id: 1, name: US,
  currency: '$',
  created_at: '123',
  updated_at: 'abc'
}
```

```
products
{
  id: 1,
  name: 'AWS' price:
  300, market: {
    id: 1,
    name: US
  }
}
```

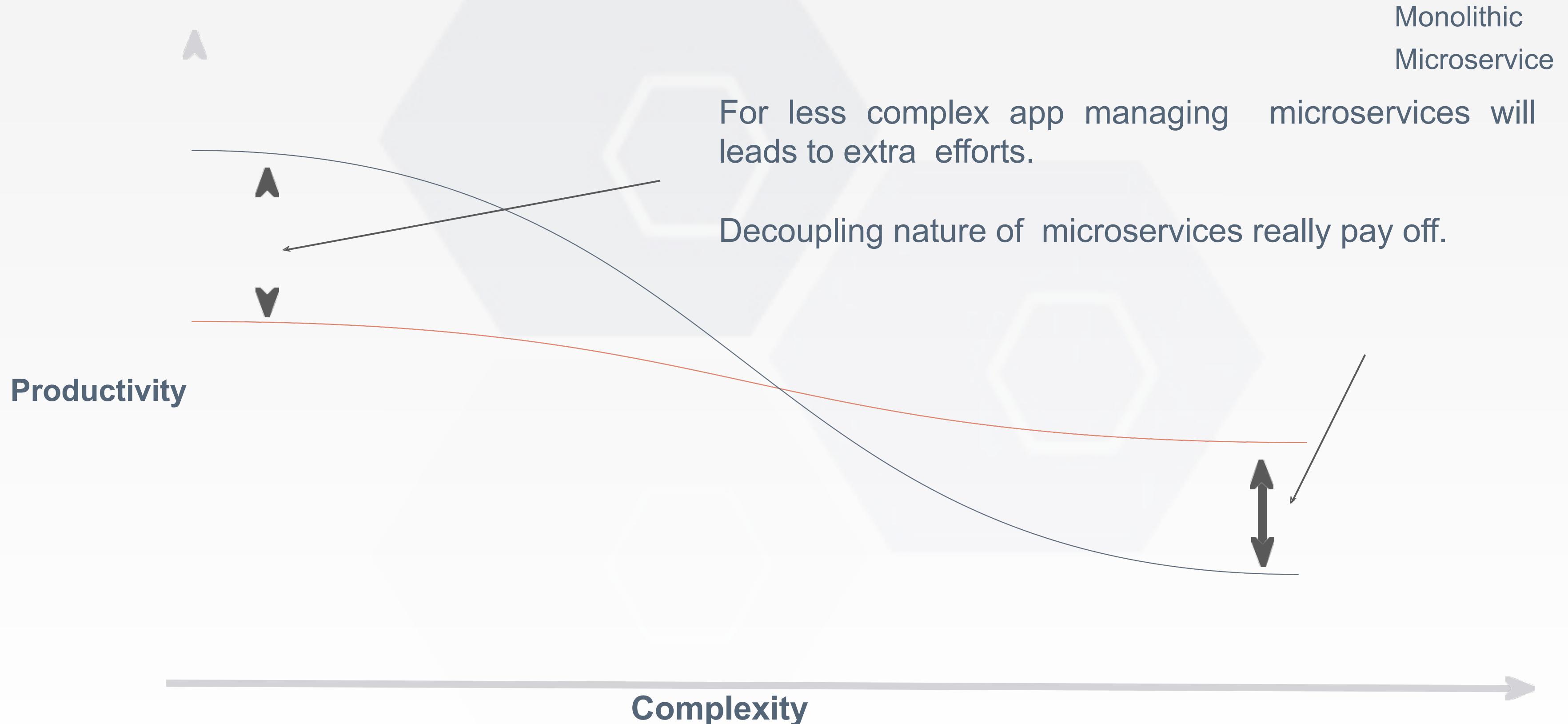
```
Users
{
  id: 1,
  name: 'John Smith'
  market: {
    id: 1,
    name: US
  }
}
```

- Separate database per service as per business domain
- Maintain Duplication of data with single source of truth
- Eventual Consistency among services

# Microservice - Pros / Cons



# Comparison (Productivity vs Complexity)



# Deployment Challenges with Microservices

Deployment and configuration of high number of services

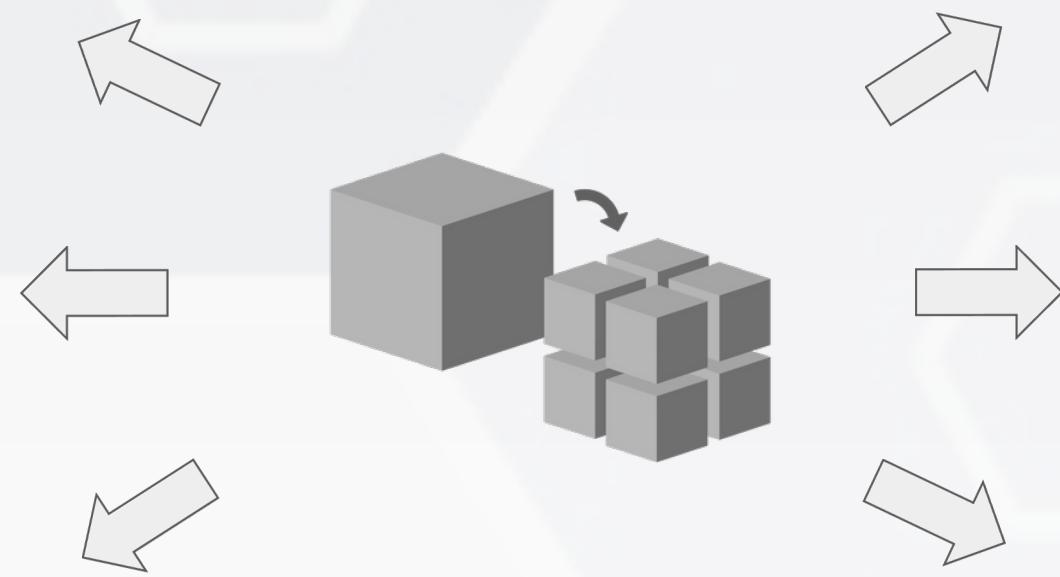
Scaling of particular services and distribution of traffic

Monitoring of services for node failures

Update strategy with zero downtime

Rollback in case of any trouble

Service Logs and Debugging





# Kubernetes (K8s)

An open-source system for automating deployment, scaling, and management of containerized apps

## Why to Choose?

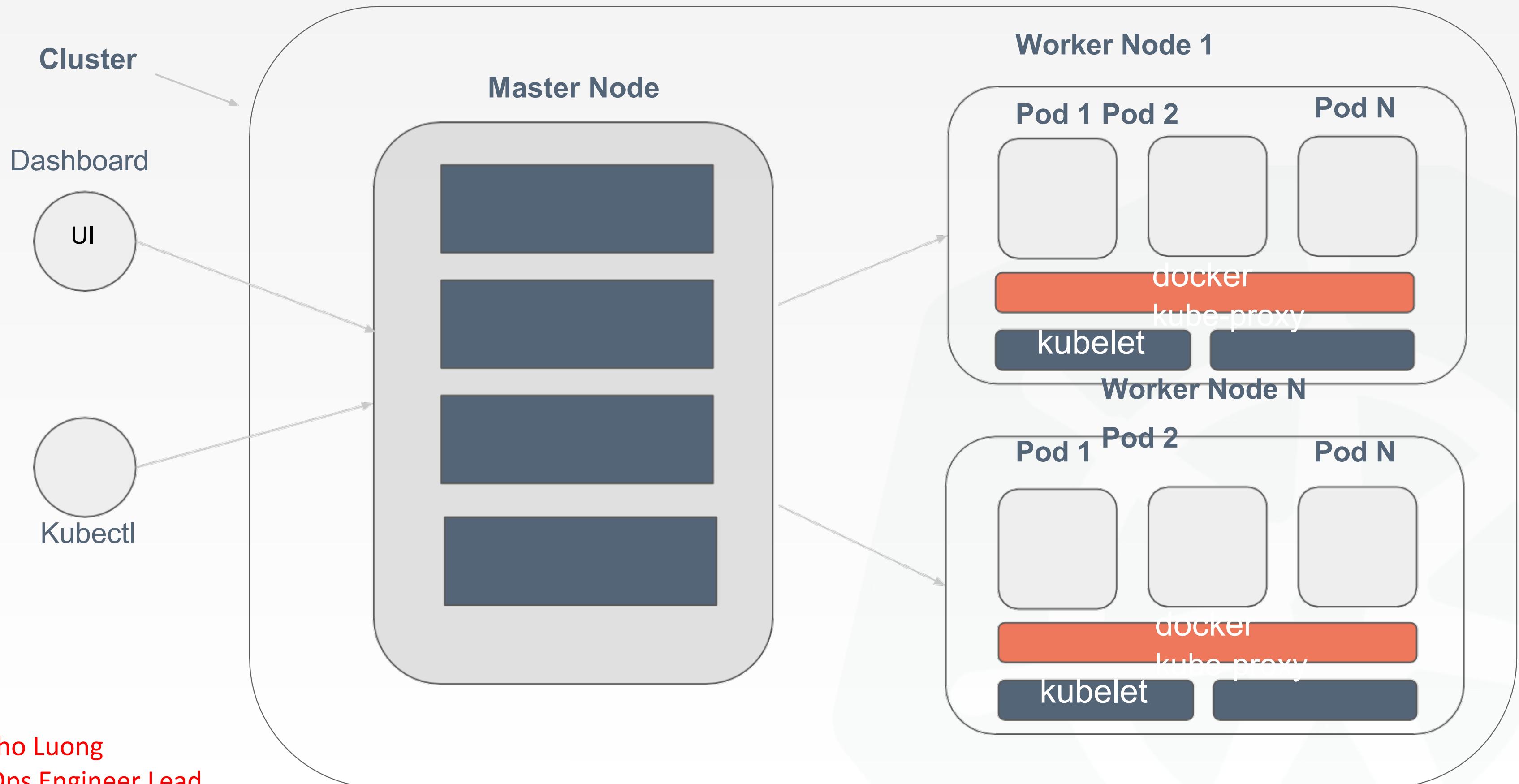
- Invented by Google
- Managed by open source community
- Adopted by Microsoft and Amazon
- Run Anywhere

Author: Nho Luong

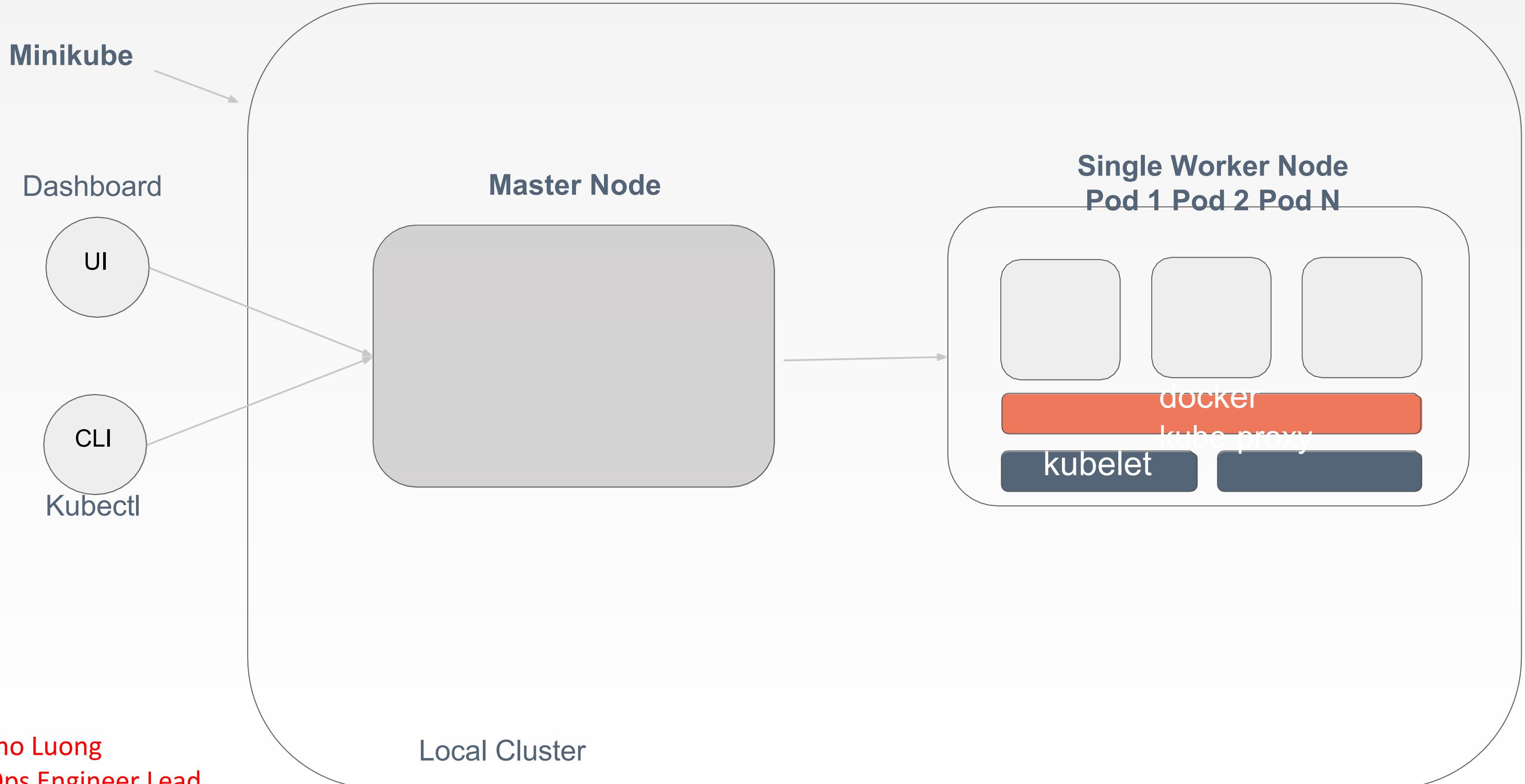
Skill: DevOps Engineer Lead

# Overview of K8s

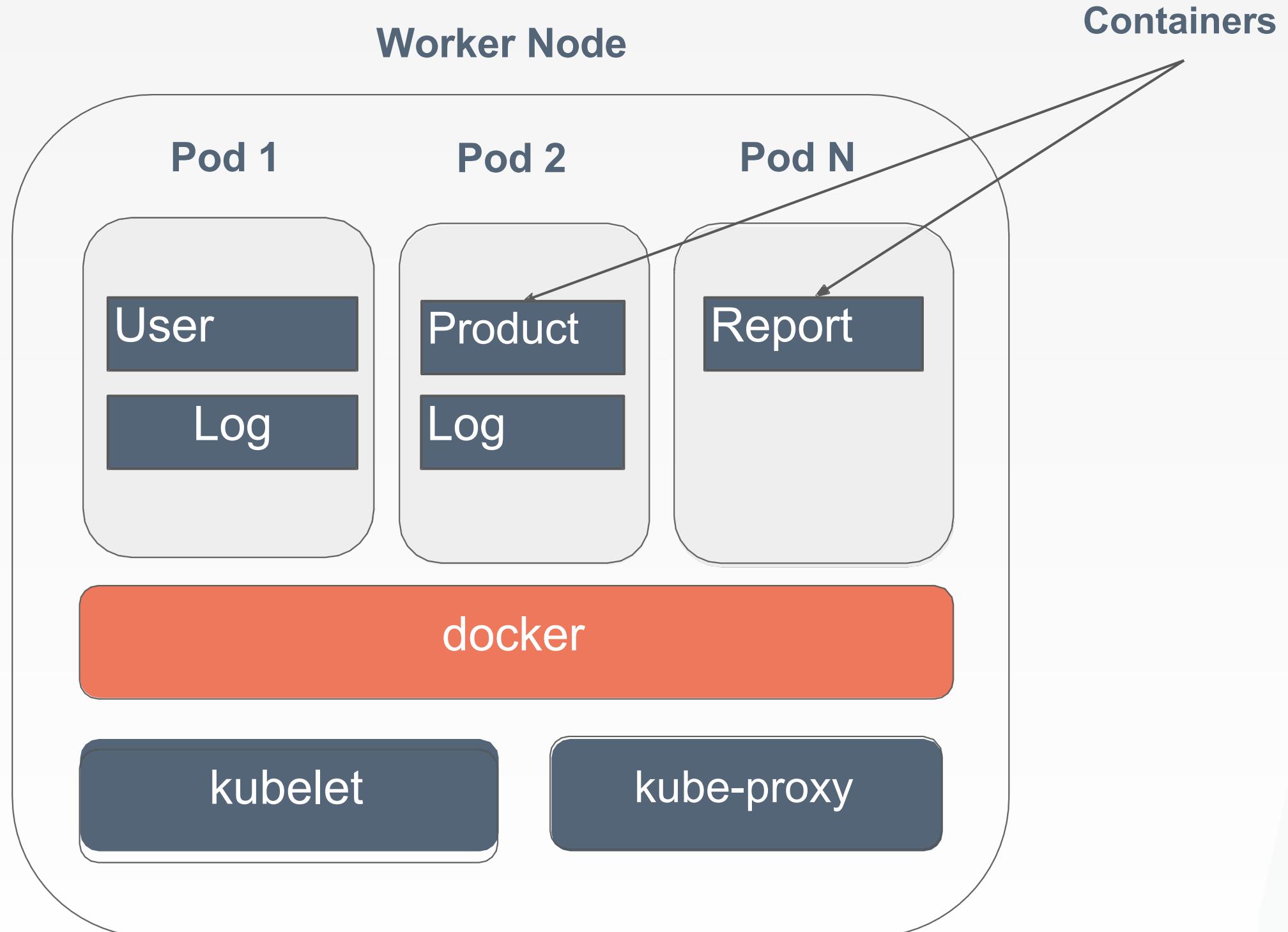
# Basic components of Kubernetes (Cloud)



# Basic components of Kubernetes (local)

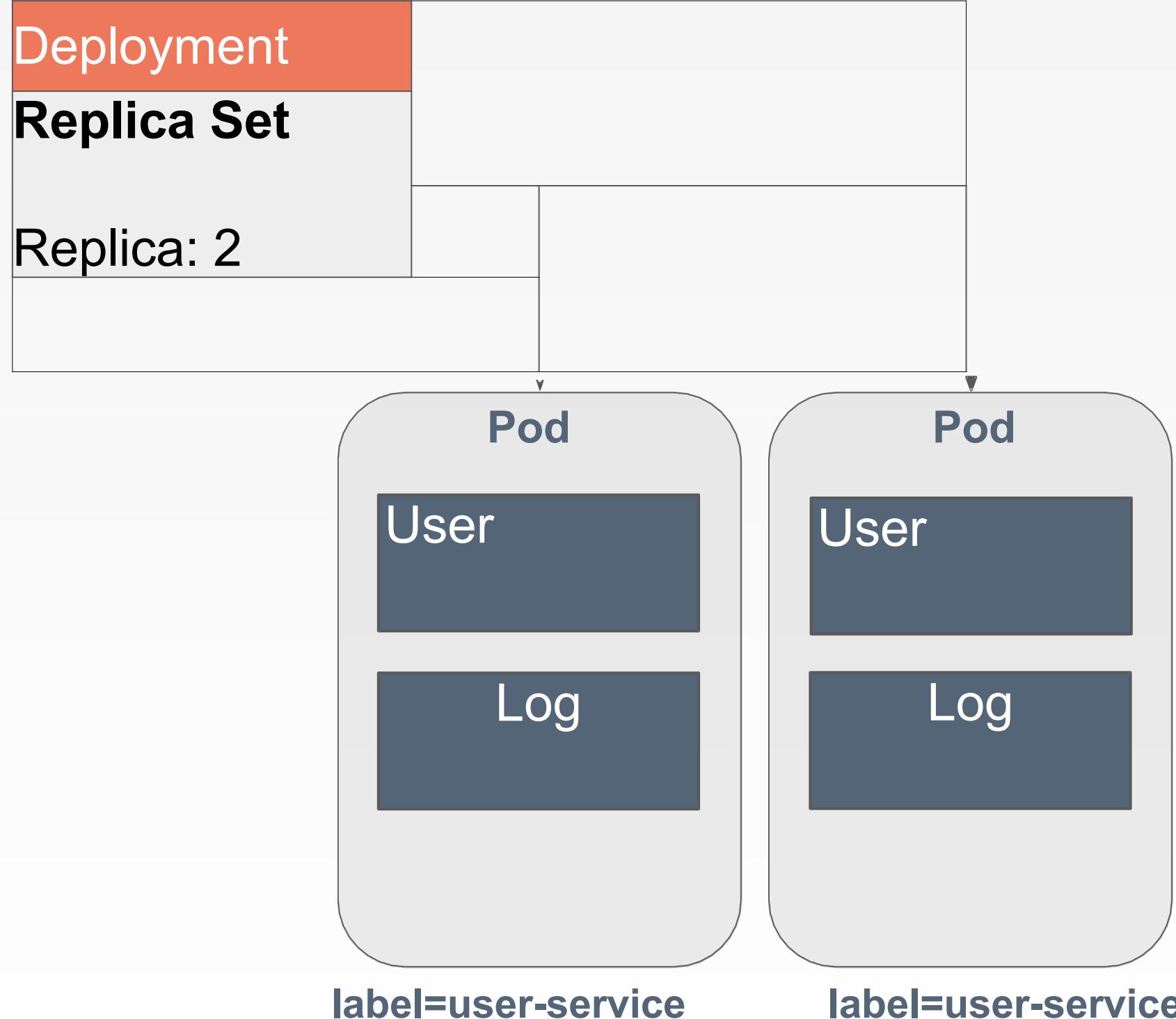


# Basic components: Pods



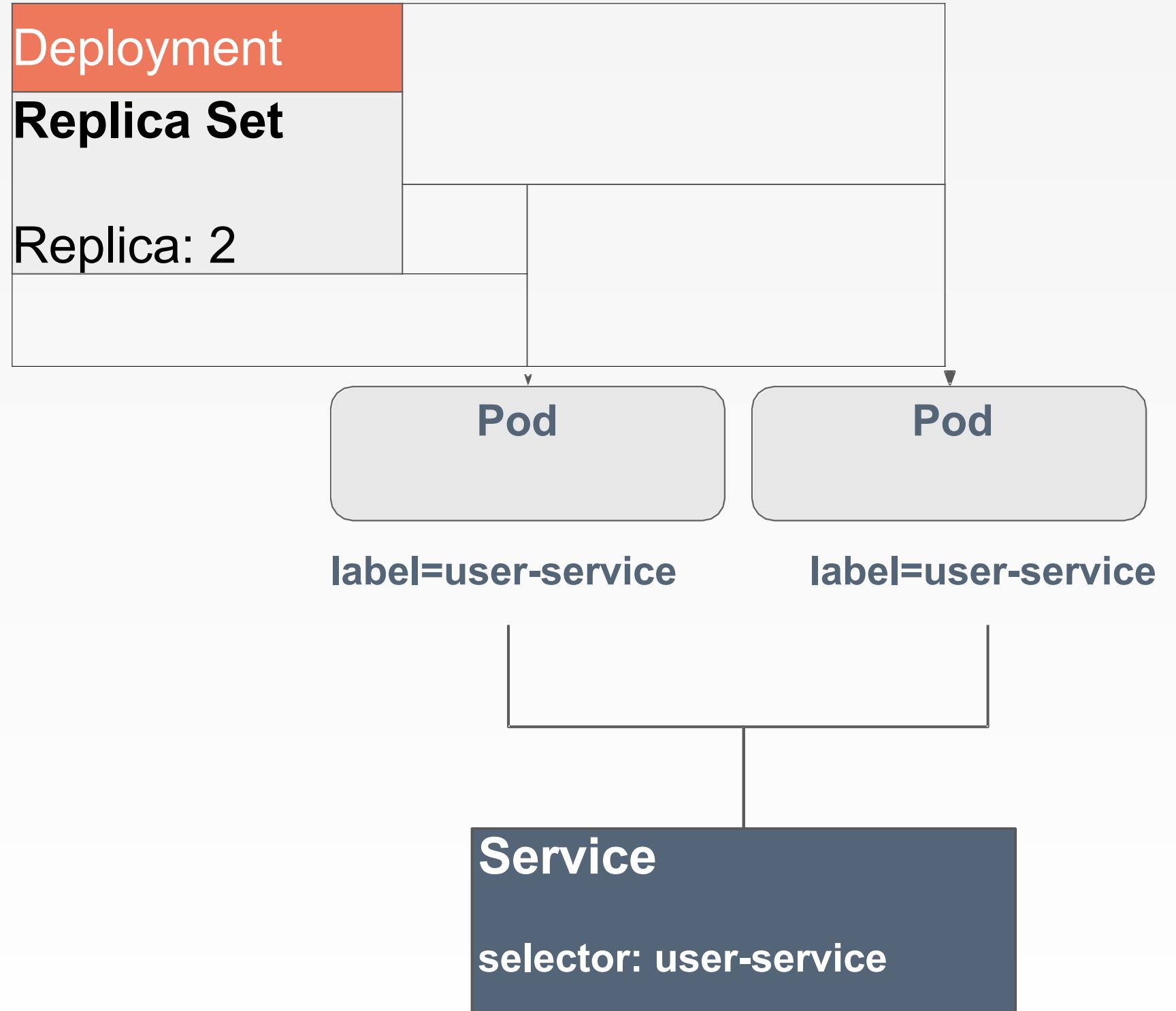
- Basic and smallest building block.
- All the connected containers deployed to a single pod
- Containers inside pod will live and die together
- All containers have common IP, and resources.
- Each POD has unique IP.

# Basic components: Deployments



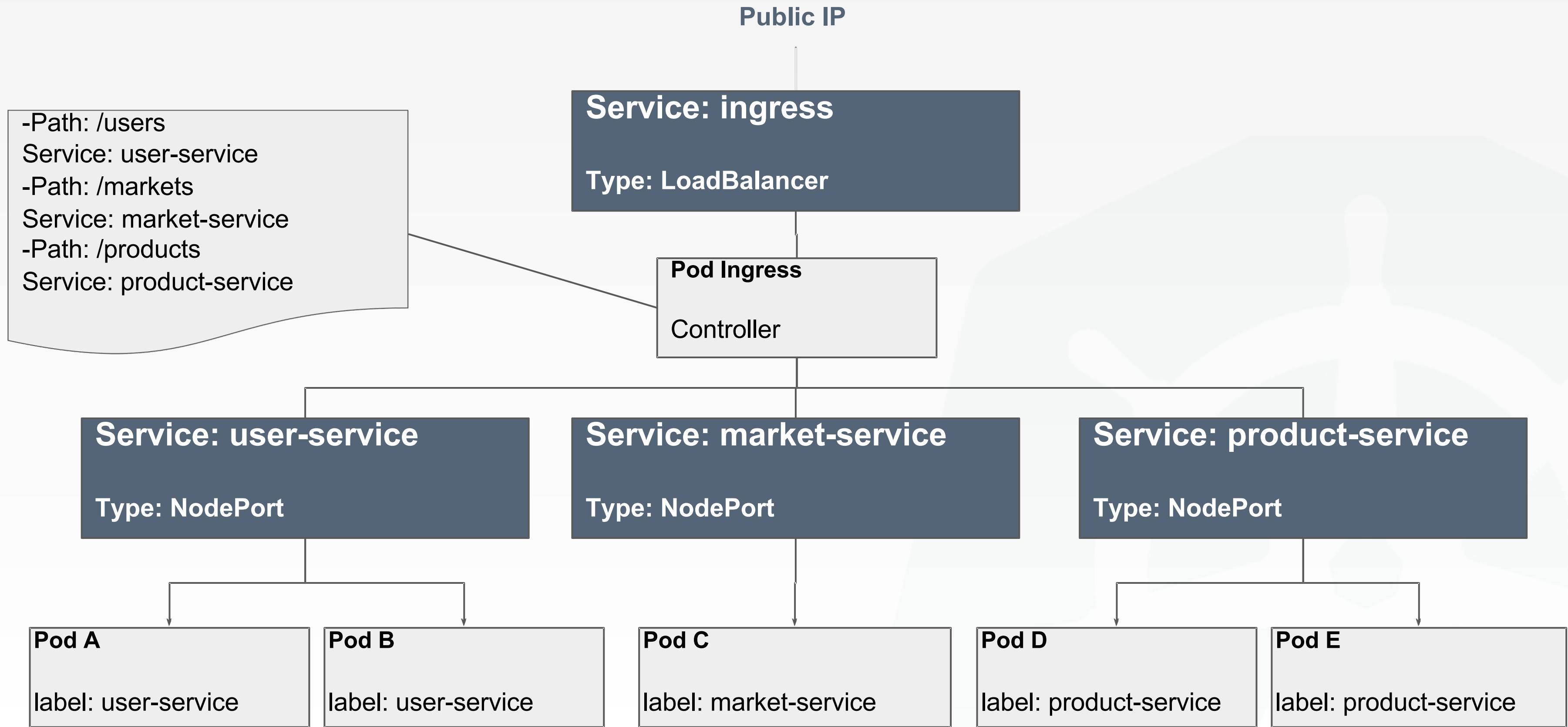
- Scaling up and down of the pods.
- Manages releases and rollback
- Self Healing of system
- Assign a unique label to every set of pods, and used by services for discovery.

# Basic components: Service



- Stable endpoint that load balances traffic among pods with similar label
- Discover the respective pods by the selectors
- With Service you don't have to remember how many pods are running or where

# Basic components: Ingress



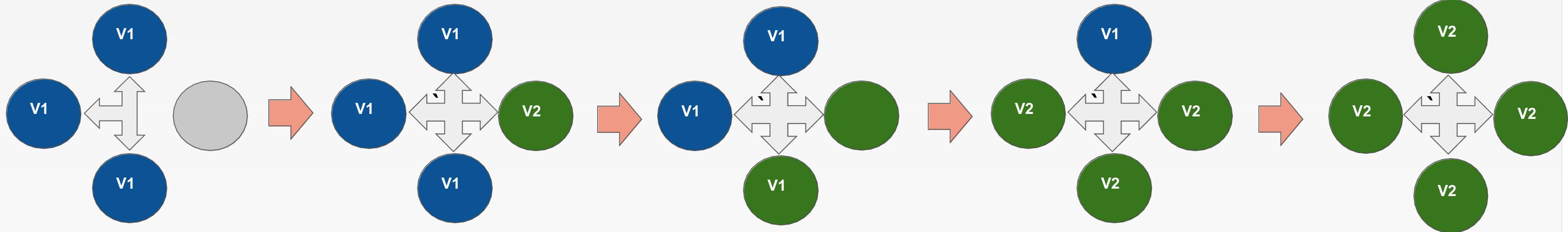
# K8s: Rollbacks

- Rolling out your application to previous version
- Set the revision that you want
- Kubernetes will scale up the corresponding ReplicaSet, and scaled down the current one

## Handy Commands:

- `kubectl rollout history deployments products`
- `kubectl rollout status deployments products`
- `kubectl rollout undo deployments products`

# Deployment Strategies: Rolling Updates

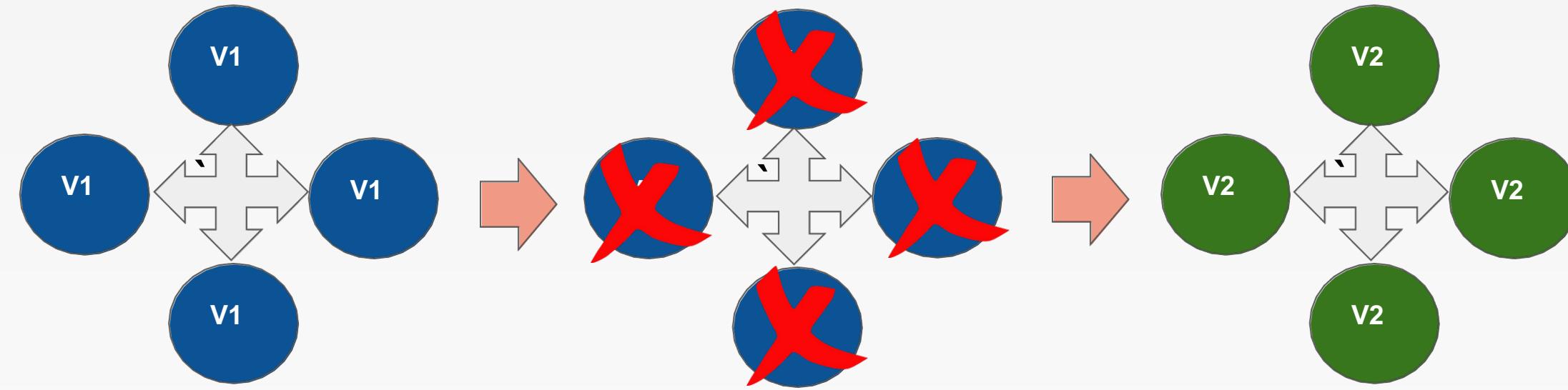


- Replace each pod in the deployment with a new one
- Backwards compatibility
- New ReplicaSet created and old ones gets decreased

# Deployment Strategies: Rolling Updates (example)

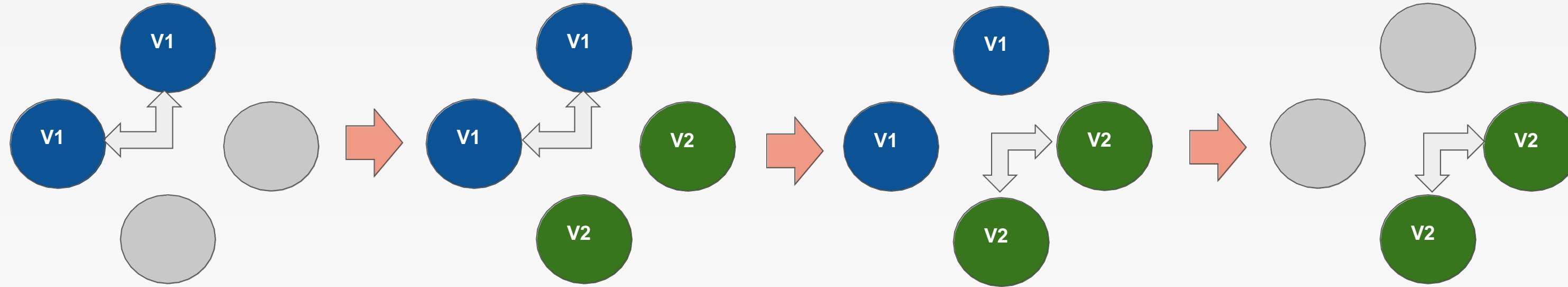
```
spec:  
  replicas: 3  
  strategy:  
    type: RollingUpdate  
    rollingUpdate:  
      maxSurge: 2          # how many pods we can add at a time  
      maxUnavailable: 0   # maxUnavailable define how many pods can be unavailable during the rolling update
```

# Deployment Strategies: Recreate Strategy



Terminate the old version and release the new one

# Deployment Strategies: Blue/Green

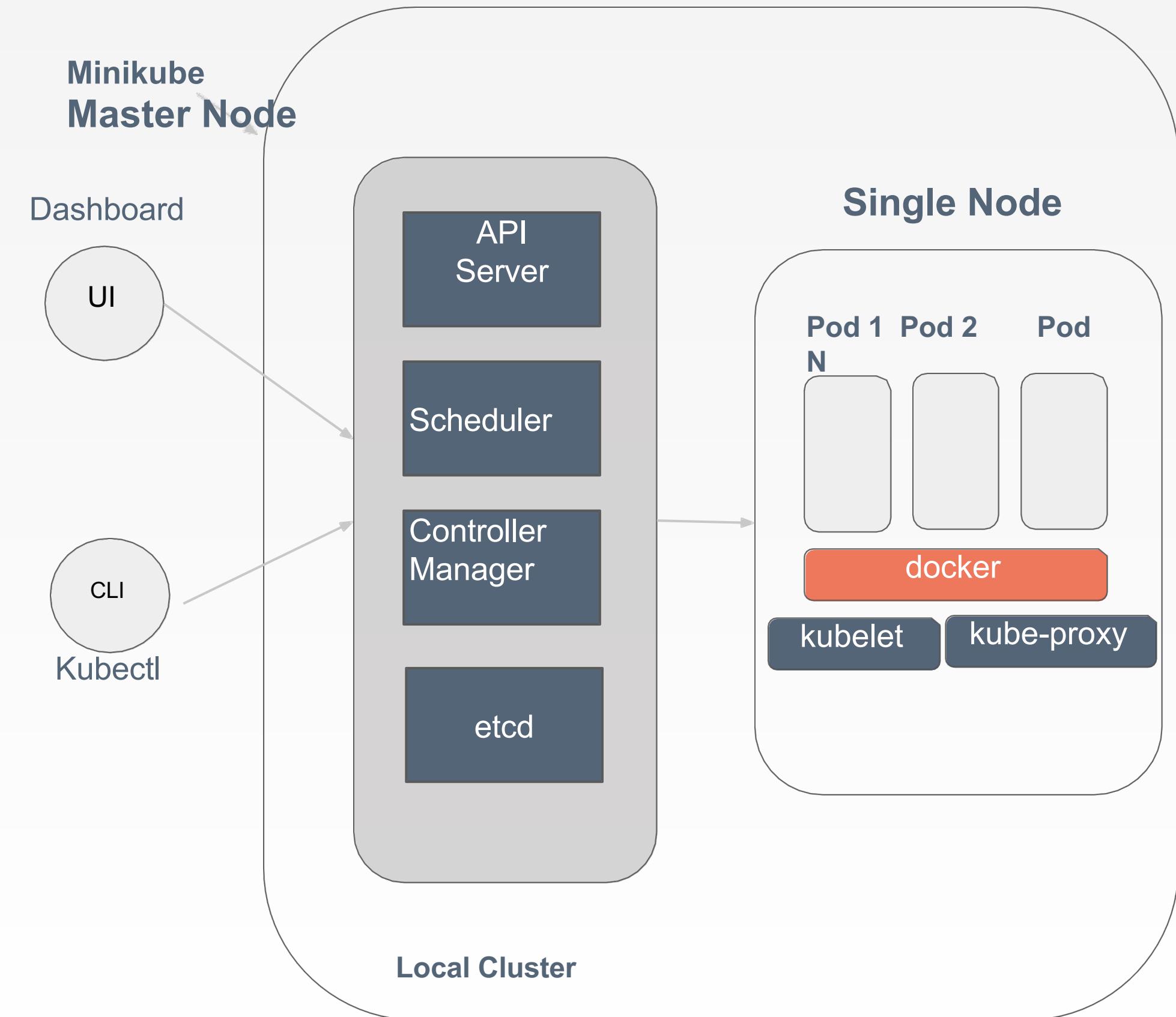


- Run two complete deployments of your application
- “green” version of the application is deployed alongside the “blue” version

# Deployment Strategies: Blue/Green (example)

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: teapot
5  spec:
6    type: ClusterIP
7    selector:
8      app: teapot
9      deployment: ${DEPLOYMENT}
10   ports:
11     - containerPort: 80
12       port: 80
```

# Minikube: K8s cluster on Local



# Minikube: Prerequisites

- VM Driver
  - macOS: VirtualBox, VMware Fusion, HyperKit
  - Linux: VirtualBox, KVM
  - Windows: VirtualBox, Hyper-V
- Install [Minikube](#)
  - brew cask install minikube  
(macOS)
- Install [kubectl](#)
- VT-x/AMD-v virtualization must be enabled in BIOS
- Internet connection on first run

Few handy commands:

- minikube start (pass VM option), default is VirtualBox
- minikube addons enable ingress

# Minikube: K8s Cluster

```
[As-MacBook-Pro:openvpn sapna$ kubectl config get-contexts
CURRENT      NAME
connectors-aks-cluster-admin
ds-governance-aks-cluster-admin
ds-management-aks-cluster-admin
ds-taxonomy-aks-cluster-admin
ds-taxonomy-prd-aks-cluster-admin
ds-visualization-aks-cluster-admin
enablers-aks-cluster-admin
ent-integration-aks-cluster
ent-integration-aks-cluster-admin
gdp-brprint-aks-cluster-admin
gdpclient-aks-cluster
gdpclient-aks-cluster-admin
gdpiintegration1-aks-cluster-admin
gke_civic-ripsaw-176319_us-central1-f_dev
gke_civic-ripsaw-176319_us-central1-f_preprod
gke_civic-ripsaw-176319_us-central1-f_prod
gke_civic-ripsaw-176319_us-central1-f_test
*
gke_devops-230413_us-central1-a_demo-poc
minikube
tvstack-acr-rgmgmt
```

Hurrah! Now we have K8s cluster running named as minikube

- kubectl config get-contexts

# Minikube: K8s Cluster

```
As-MacBook-Pro:openvpn sapna$ kubectl config use-context minikube
Switched to context "minikube".
As-MacBook-Pro:openvpn sapna$ k config get-contexts
CURRENT  NAME                                     CLUSTER
connectors-aks-cluster-admin                   connectors-a
ds-governance-aks-cluster-admin                ds-governanc
ds-management-aks-cluster-admin                ds-managemen
ds-taxonomy-aks-cluster-admin                 ds-taxonomy-
ds-taxonomy-prd-aks-cluster-admin              ds-taxonomy-
ds-visualization-aks-cluster-admin            ds-visualiza
enablers-aks-cluster-admin                    enablers-aks
ent-integration-aks-cluster                   ent-integrat
ent-integration-aks-cluster-admin             ent-integrat
gdp-brprint-aks-cluster-admin                 gdp-brprint-a
gdpclient-aks-cluster                        gdpclient-ak
gdpclient-aks-cluster-admin                  gdpclient-ak
gdpiintegration1-aks-cluster-admin           gdpiintegrati
gke_civic-ripsaw-176319_us-central1-f_dev    gke_civic-ri
gke_civic-ripsaw-176319_us-central1-f_preprod gke_civic-ri
gke_civic-ripsaw-176319_us-central1-f_prod   gke_civic-ri
gke_civic-ripsaw-176319_us-central1-f_test   gke_civic-ri
gke_devops-230413_us-central1-a_demo-poc    gke_devops-2
*      minikube                                minikube
        tvstack-ac-s-rgmgt                      tvstack-ac-
```

As-MacBook-Pro:openvpn sapna\$

## Change context to minikube

- kubectl config use-context minikube
- kubectl config get-contexts

# Minikube: Deployment (Markets Service)

```
! markets-deployment.yaml ×  
1  apiVersion: apps/v1beta1  
2  kind: Deployment  
3  metadata:  
4    name: markets  
5  spec:  
6    replicas: 2  
7    template:  
8      metadata:  
9        labels:  
10       app: demo-markets  
11    spec:  
12      containers:  
13        - name: demo-markets  
14          image: nodexperts/demo-markets:v1.4.0  
15          imagePullPolicy: Always  
16          env:  
17            - name: PORT  
18              value: "10000"  
19            - name: API_PREFIX  
20              value: "/markets/api"  
21          ports:  
22            - containerPort: 10000  
23          readinessProbe:  
24            httpGet:  
25              port: 10000  
26              path: /markets/api/health-check  
27            initialDelaySeconds: 1  
28            periodSeconds: 5  
29            timeoutSeconds: 4  
30            successThreshold: 2  
31            failureThreshold: 3
```

Container Port is the port that the application is running on.

Readiness probes: when a Container is ready to start accepting traffic.

A Pod is considered ready when all of its Containers are ready

# Minikube: Deployment (Products Service)

! products-deployment.yaml ×

```
1 apiVersion: apps/v1beta1
2 kind: Deployment
3 metadata:
4   name: products
5 spec:
6   replicas: 4
7   template:
8     metadata:
9       labels:
10      app: demo-products
11      version: "1.0"
12   spec:
13     containers:
14       - name: demo-products
15         image: nodexperts/demo-products:v1.4.0
16         imagePullPolicy: Always
17         env:
18           - name: PORT
19             value: "9000"
20           - name: API_PREFIX
21             value: "/products/api"
22         ports:
23           - containerPort: 9000
24         readinessProbe:
25           httpGet:
26             port: 9000
27             path: /products/api/health-check
28           initialDelaySeconds: 1
29           periodSeconds: 5
30           timeoutSeconds: 4
31           successThreshold: 2
32           failureThreshold: 3
```

- A Deployment named demo-product is created
- four replicated Pods
- Pods are labeled as demo-products
- Image from docker
- Environment variables

# Minikube: Deployment (Users Service)

```
! users-deployment.yaml x
1  apiVersion: apps/v1beta1
2  kind: Deployment
3  metadata:
4    name: users
5  spec:
6    replicas: 2
7    template:
8      metadata:
9        labels:
10       |   app: demo-users
11    spec:
12      containers:
13      - name: demo-users
14        image: nodexperts/demo-users:v1.0.0
15        imagePullPolicy: Always
16        env:
17        - name: PORT
18          value: "11000"
19        - name: API_PREFIX
20          value: "/users/api"
21        ports:
22        - containerPort: 11000
23        readinessProbe:
24          httpGet:
25            port: 11000
26            path: /users/api/health-check
27          initialDelaySeconds: 1
28          periodSeconds: 5
29          timeoutSeconds: 4
30          successThreshold: 2
31          failureThreshold: 3
```

- A Deployment named demo-users is created
- Two replicated Pods
- Pods are labeled as demo-users
- Image from docker
- Environment variables

# Minikube: Service (Markets Service)

! markets-service.yaml ✘

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: demo-markets
5   labels:
6     app: demo-markets
7 spec:
8   selector:
9     app: demo-markets
10  ports:
11    - port: 80
12      protocol: TCP
13      targetPort: 10000
14      # nodePort: 10000
15      type: NodePort
```

address, those IPs are not exposed outside the cluster

ts demo-markets Pods

# Minikube: Service (Products Service)

! products-service.yaml ×

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: demo-products
5   labels:
6     app: demo-products
7 spec:
8   selector:
9     app: demo-products
10  ports:
11    - port: 80
12      protocol: TCP
13      targetPort: 9000
14      name: http
15      # nodePort: 9000
16      type: NodePort
```

These services, those IPs are not exposed outside the cluster

demo-products Pods

# Minikube: Service (Users Service)

```
! users-service.yaml ✘  
1  apiVersion: v1  
2  kind: Service  
3  metadata:  
4    name: demo-users  
5    labels:  
6      app: demo-users  
7  spec:  
8    selector:  
9      app: demo-users  
10   ports:  
11     - port: 80  
12       protocol: TCP  
13       targetPort: 11000  
14       name: http  
15       # nodePort: 11000  
16       type: NodePort
```

address, those IPs are not exposed outside the cluster

:s demo-users Pod

# Minikube: Ingress

! ingress.yaml ✘

```
1  apiVersion: extensions/v1beta1
2  kind: Ingress
3  metadata:
4    name: demo-ingress
5    annotations:
6      ingress.kubernetes.io/rewrite-target: /
7  spec:
8    rules:
9      - http:
10        paths:
11          - path: /products/api/*
12            backend:
13              serviceName: demo-products
14              servicePort: 80
15          - path: /markets/api/*
16            backend:
17              serviceName: demo-markets
18              servicePort: 80
19          - path: /users/api/*
20            backend:
21              serviceName: demo-users
22              servicePort: 80
```

Ingress objects are the rules that define the routes that should exist.

Annotations to configure some options depending on the Ingress controller

Target URI where the traffic must be redirected

# Minikube: UI Deployment and Service

```
3  metadata:
4    name: ui
5  spec:
6    replicas: 1
7    template:
8      metadata:
9        labels:
10       app: demo-ui
11    spec:
12      containers:
13      - name: demo-ui
14        image: nodexperts/demo-app-ui:v1.4.0
15        imagePullPolicy: Always
16      env:
17      - name: PORT
18        value: "5000"
19      # - name: REACT_APP_PRODUCTS_URL
20      #   value: "http://34.96.84.42/products/api"
21      # - name: REACT_APP_MARKETS_URL
22      #   value: "http://34.96.84.42/markets/api"
23      # - name: REACT_APP_USER_URL
24      #   value: "http://34.96.84.42/users/api"
25      - name: REACT_APP_PORT
26        value: "5000"
27      # For minikube
28      - name: REACT_APP_PRODUCTS_URL
29        value: "https://http://34.96.84.42/products/api"
30      - name: REACT_APP_MARKETS_URL
31        value: "https://192.168.99.100/markets/api"
32      - name: REACT_APP_USER_URL
33        value: "https://192.168.99.100/users/api"
34      - name: REACT_APP_PORT
```

```
! demo-app-ui.service.yaml ✘
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: demo-ui
5    labels:
6      app: demo-ui
7  spec:
8    selector:
9      app: demo-ui
10   ports:
11     - port: 80
12       protocol: TCP
13       targetPort: 5000
14       # nodePort: 10000
15     type: LoadBalancer
```

# Clouds and Tools

We at Successive Technologies use following tools for building **Cloud Agnostic Platform**.



Google Cloud Platform



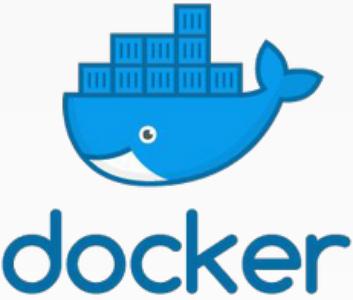
kubernetes



HashiCorp  
Terraform



HashiCorp  
Vault



docker



Grafana



OPENVPN





Thank You