

# 源于社区 服务社区

 中国DevOps社区峰会 2023 · 广州



## 海量K8S集群的应用部署和管理实践

邓宇星 SUSE 架构师





# 邓宇星

## SUSE 软件架构师 云原生/多集群

- 在Rancher/SUSE工作，负责产品研发/项目交付
- 多集群管理/CICD/监控日志告警
- RFO(Rancher/RKE2 for openEuler)发行版维护





# 目录

- 1 DC与Edge 集群共同管理
- 2 海量K3s 边缘集群管理
- 3 基于fleet的海量集群应用部署分享
- 4 门户整合以及平台推广
- 5 Q&A





# DC与Edge 集群共同管理

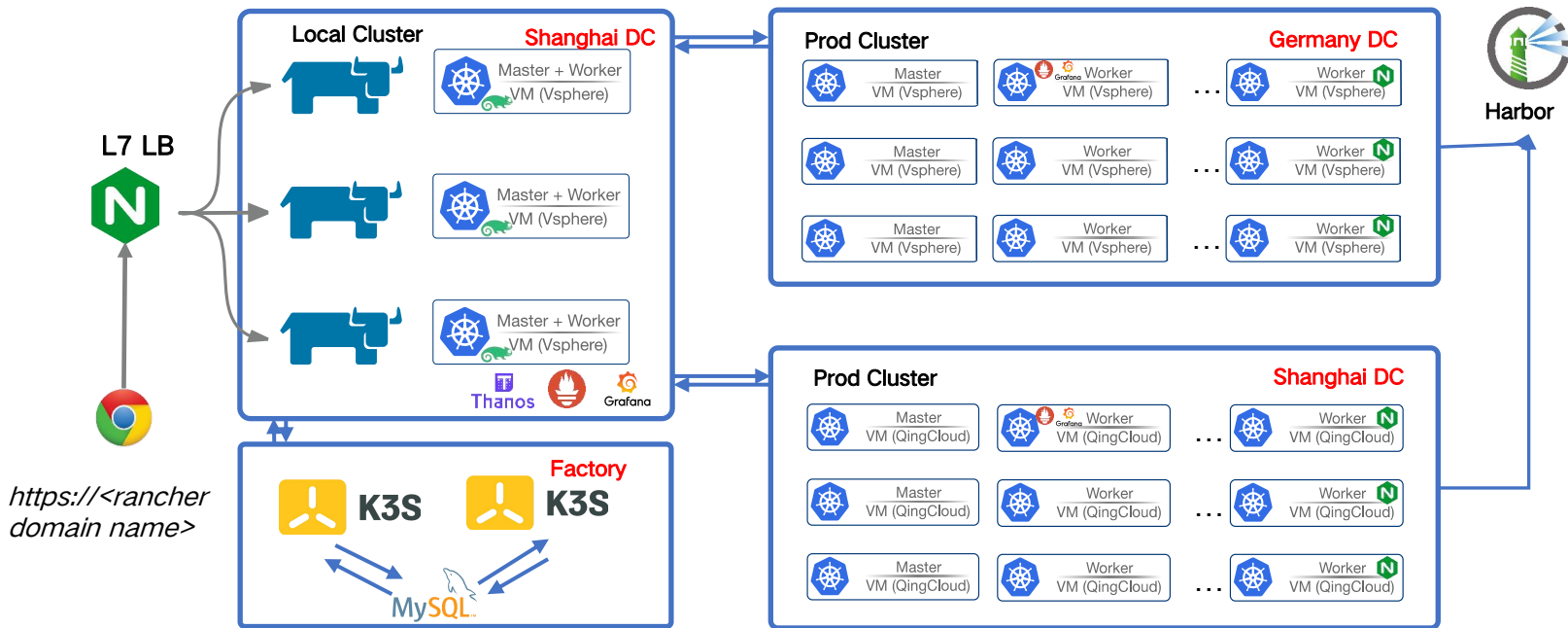


# 汽车制造业客户 - 云边协同/边缘自治



- 转向Java技术栈，工厂应用重构并容器化后进行了微服务改造。解决业务快速发展与创新时面临的系统弹性可扩展、敏捷迭代、技术驱动业务创新等难题。
- 基于 K3S 实现了 MES 、WMS等系统在节省约 35% 硬件成本的前提下的高可用，工厂网络在线时可批量管理、更新、发版、维护，工厂网络离线时 K3S 本地自治，服务不中断，产线不停产。
- 通过 Rancher 纳管全球工厂 K3S及数据中心K8S，统一平台、统一权限、统一管理。立足上海，管理全球。

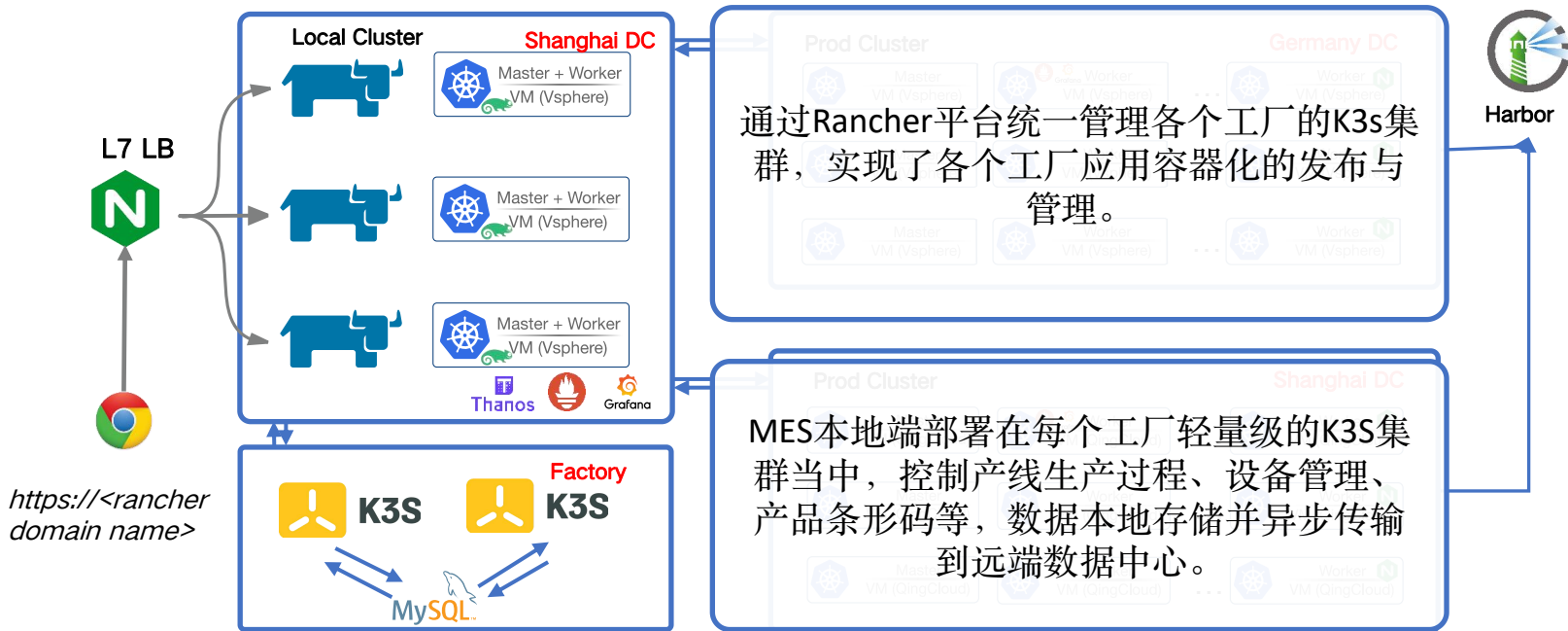
# 汽车制造业客户部署案例



# 汽车制造业客户部署案例



# 汽车制造业客户部署案例







# 海量K3s 边缘集群管理





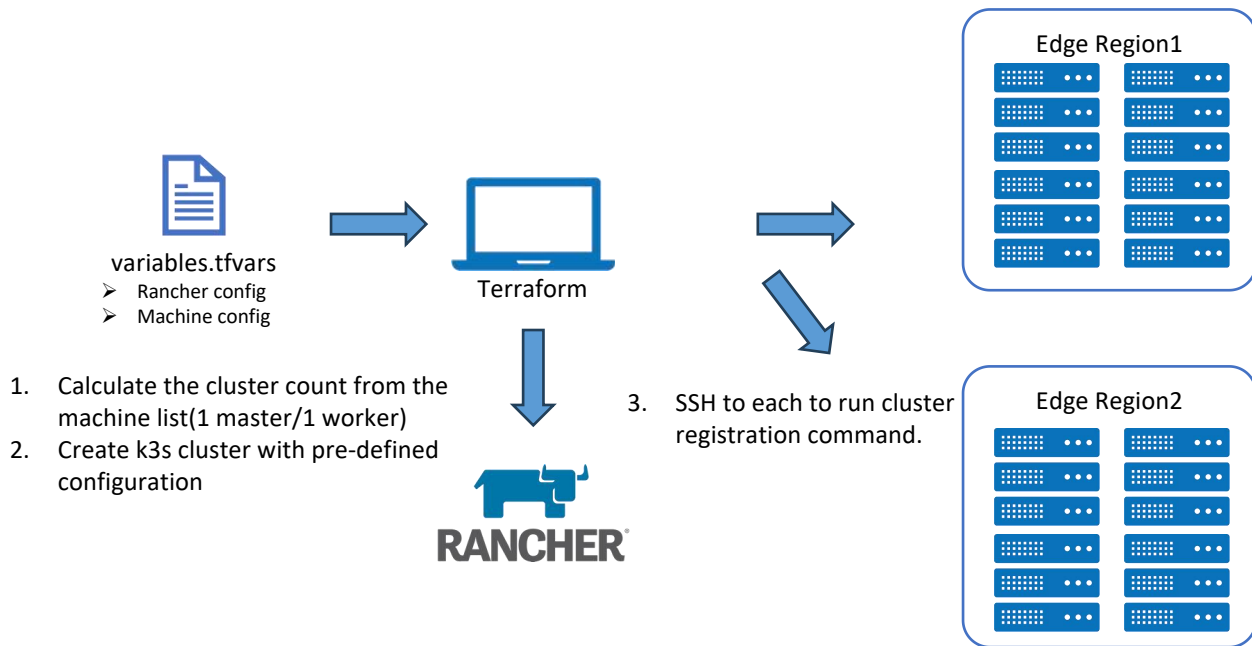
# 海量 Managed K3s管理实践

- IAC(Infrastructure As Code) Tools
  - Puppet – Server/Agent Mode with state file
  - Ansible – CLI tools with script like configuration
  - Terraform – CLI tools with state file





# Terraform Demo



配置文件例子: <https://github.com/orangedeng/devops-demo/tree/master/test-machine-terraform>



# Terraform Demo

```
machine_config = {
  ssh_user      = "root"
  machines      = ["192.168.31.229", "192.168.31.95", "192.168.31.36"]
  ssh_key_path  = "/Users/dengyuxing/keys/id_rsa"
}
```

```
rancher_config = {
  api_url      = "https://192.168.31.40:9443"
  access_key   = "token-dj9mw"
  secret_key   = "nkp1w9p9htjgmlfvzhvt4pczt4jhgkh8x2cn542gk6zx8f95q77rfz"
  insecure     = true
}
```

```
resource "rancher2_cluster_v2" "testk3s" {
  count      = ceil(length(var.machine_config.machines) / 2)
  provider   = rancher2.admin
```

```
  name              = "testk3s-${count.index}"
  kubernetes_version = "v1.26.8+k3s1"
```

Yuxing Deng, 5天前 | 1 author (Yuxing Deng)

```
  rke_config {
```

Yuxing Deng, 5天前 | 1 author (Yuxing Deng)

```
    registries {
```

Yuxing Deng, 5天前 | 1 author (Yuxing Deng)

```
      mirrors {
```

```
        hostname = "docker.io"
```

```
        endpoints = ["https://docker.nju.edu.cn"]
      }
    }
  }
}
```

```
→ test-machine-terraform git:(master) x terraform init
```

Initializing the backend...

Initializing provider plugins...

- terraform.io/builtin/terraform is built in to Terraform
- Reusing previous version of rancher/rancher2 from the dependency lock file
- Using previously-installed rancher/rancher2 v3.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

- ```
→ test-machine-terraform git:(master) x terraform apply -auto-approve -
→ test-machine-terraform git:(master) x terraform apply -auto-approve -var-file=./variables.tfvars
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

```
+ create
```

配置文件例子: <https://github.com/orangedeng/devops-demo/tree/master/test-machine-terraform>



# Terraform Demo

Cluster Management

Clusters3

Cloud Credentials

Drivers

Global Monitoring

Image Repo

RKE1 Configuration

Advanced

Clusters

Download KubeConfig

Take Snapshot

Download YAML

Delete

Filter

| <input type="checkbox"/> | State  | Name      | Version      | Provider   | Machines | Age     |                    |
|--------------------------|--------|-----------|--------------|------------|----------|---------|--------------------|
| <input type="checkbox"/> | Active | local     | v1.26.4+k3s1 | Local K3s  | 1        | 7 days  | <div>Explore</div> |
| <input type="checkbox"/> | Active | testk3s-0 | v1.26.8+k3s1 | Custom K3s | 2        | 10 mins | <div>Explore</div> |
| <input type="checkbox"/> | Active | testk3s-1 | v1.26.8+k3s1 | Custom K3s | 1        | 10 mins | <div>Explore</div> |

v2.7.9-ent

配置文件例子: <https://github.com/orangedeng/devops-demo/tree/master/test-machine-terraform>





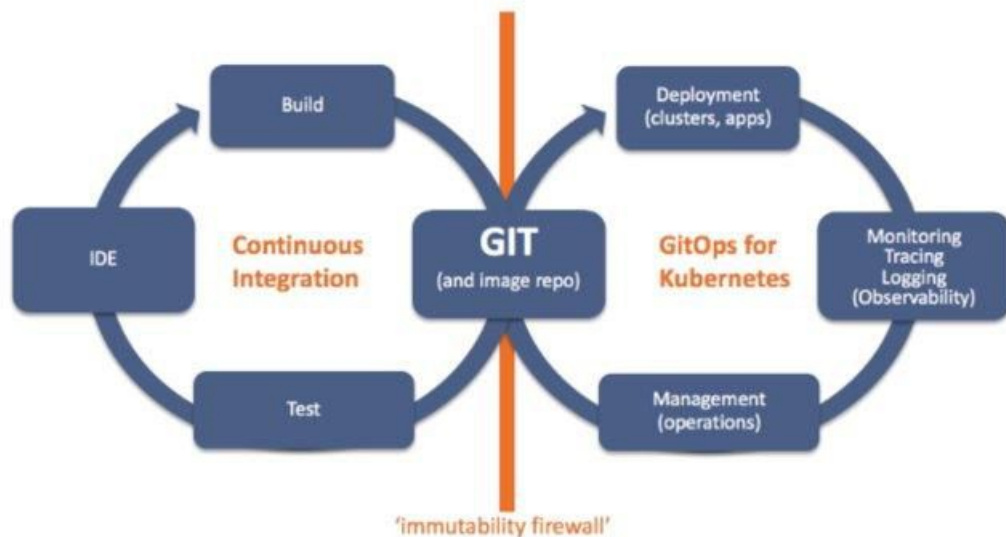
# Fleet的应用部署分享

- GitOps背景
- Fleet架构
- Fleet应用管理/部署实践



# GitOps背景

GitOps 是一种持续交付的方式，是 DevOps 中的一部分，它的核心思想是将应用系统的声明性基础架构存放在 Git 版本库中。



**Git as the single source of truth** of a system's desired state

**GitOps Diffs** compare desired state with observed state (eg Kubediff, Terradiff, Canary..)

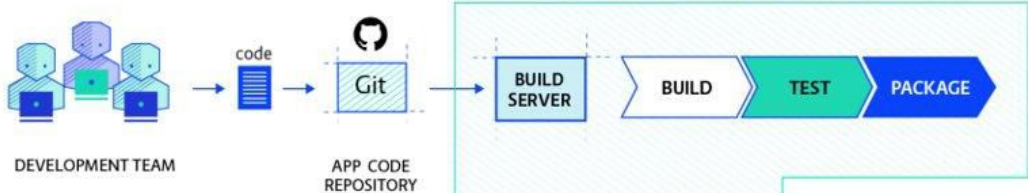
**ALL** intended operations are committed by pull request, for all environments

**ALL** diffs between GIT and observed state lead to (auto) convergence using tools like K8s

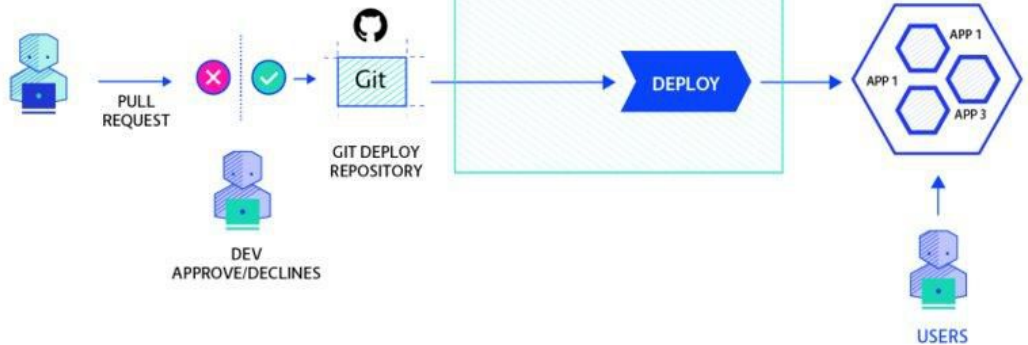
**ALL** changes are observable, verifiable and audited indisputably, with rollback & D/R

# GitOps背景 - 架构

## Build Pipeline

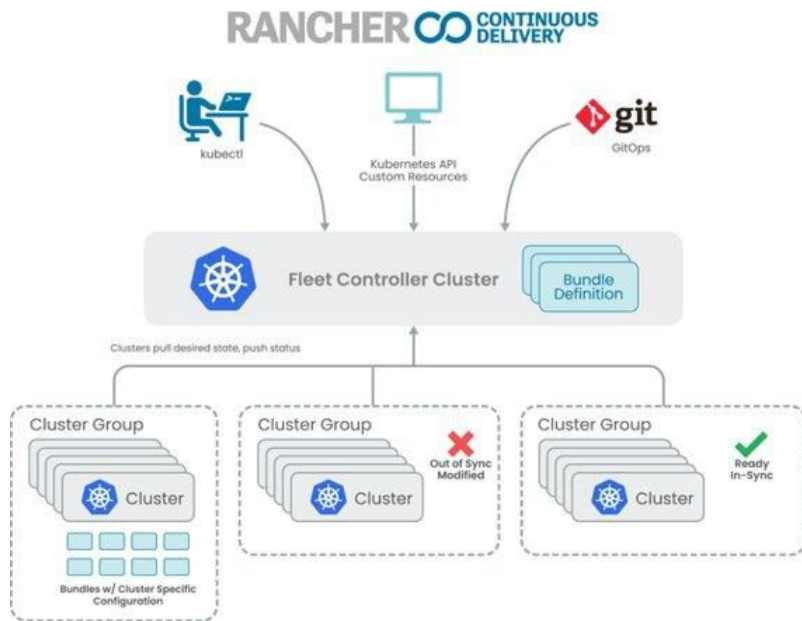


## GitOps Workflow





# Fleet 架构



- Fleet Manager (gitjob 组件 polling mode) 从 github pull 代码
- Fleet Manager 运行 “fleet apply” 通过 k8s resource 文件生成 bundles 资源对象
- 为每个下游 cluster 生成 bundleDeployment 资源对象
- 下游 agent 从 Fleet Manager 中 pull bundleDeployment 资源对象
- 生成 helm charts/manifests
- 运行 helm/manifest 部署
- 更新状态信息

# Fleet应用管理/部署实践 – Cluster/Group

Continuous Delivery

fleet-default

Clusters

|| Pause    ↻ Force Update    📁 Change workspace    ⚙️ Actions    Filter

| <input type="checkbox"/> State  | Name        | Nodes Ready | Repos Ready | Resources | Last Seen   | Age     |   |
|---------------------------------|-------------|-------------|-------------|-----------|-------------|---------|---|
| <input type="checkbox"/> Active | dc-k3s      | 1           | 1           | —         | 52 secs ago | 16 mins | ⋮ |
| <input type="checkbox"/> Active | k3s-region1 | 1           | 1           | —         | 13 mins ago | 14 mins | ⋮ |
| <input type="checkbox"/> Active | k3s-region2 | 1           | 1           | —         | 13 mins ago | 13 mins | ⋮ |

v2.7.9-ent

Cluster Group: edge Active

Workspace: fleet-default    Age: 6 days

```
1 apiVersion: fleet.cattle.io/v1alpha1
2 kind: ClusterGroup
3 metadata:
4   creationTimestamp: '2023-11-02T08:44:23Z'
5   generation: 2
6   managedFields:
49   name: edge
50   namespace: fleet-default
51   resourceVersion: '3598426'
52   uid: fc3c753e-c68b-482f-bcb2-4a40fa4f72c4
53 spec:
54   selector:
55     matchExpressions:
56     - key: edge
57       operator: Exists
58     matchLabels: {}
59 status:
84
```

Create

## Cluster Groups

Download YAML    Delete    Filter

| <input type="checkbox"/> State  | Name     | Clusters Ready | Resources | Age    |   |
|---------------------------------|----------|----------------|-----------|--------|---|
| <input type="checkbox"/> Active | dc-group | 1              | 3         | 6 days | ⋮ |
| <input type="checkbox"/> Active | edge     | 2              | 6         | 6 days | ⋮ |

配置文件例子: <https://github.com/orangedeng/devops-demo/tree/master/fleet-test>



# Fleet应用管理/部署实践 – Repo&Target

Git Repos

|| Pause    ↻ Force Update    ⬇ Download YAML    🗑 Delete    Filter

| <input type="checkbox"/> State  | Name             | Repo                                                                                                              | Target | Clusters Ready | Resources | Age      |   |
|---------------------------------|------------------|-------------------------------------------------------------------------------------------------------------------|--------|----------------|-----------|----------|---|
| <input type="checkbox"/> Active | dc-example       | <a href="http://192.168.31.40:3000/orst">http://192.168.31.40:3000/orst</a><br>ar/devops-demo<br>master @ bb1d8d6 | Group  | 1/1            | 3         | 2.7 days | ⋮ |
| <input type="checkbox"/> Active | edge-exam<br>ple | <a href="http://192.168.31.40:3000/orst">http://192.168.31.40:3000/orst</a><br>ar/devops-demo<br>master @ bb1d8d6 | Group  | 2/2            | 3         | 2.7 days | ⋮ |

```
dc
├── configmap.yaml
├── fleet.yaml
├── nginx.yaml
└── service.yaml
edge
├── configmap.yaml
├── fleet.yaml
├── nginx.yaml
└── overlays
    ├── region1
    │   └── configmap_patch.yaml
    ├── region2
    │   └── configmap_patch.yaml
    └── service.yaml
```

DC 中的应用部署文件  
Edge 中的应用部署文件  
overlays目录下为不同region中对资源文件的patch配置

6 directories, 11 files

配置文件例子: <https://github.com/orangedeng/devops-demo/tree/master/fleet-test>

Git Repo: edge-example Active

Workspace: fleet-default    Age: 2.7 days

Edit: Step 2    Repository Details

Define target details

Deploy To

Target  
edge

No Clusters  
All Clusters in the Workspace  
Advanced

Clusters  
dc-k3s  
k3s-region1  
k3s-region2

Cluster Groups  
dc-group  
edge

Target Name  
Optional:

ing attributes.

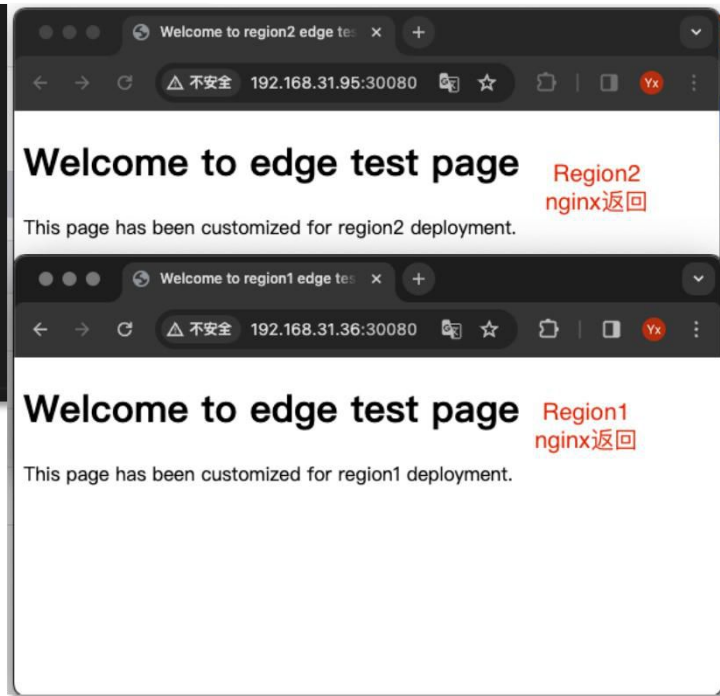


# Fleet应用管理/部署实践 – Fleet配置文件

```
namespace: edge-example
targetCustomizations:
- name: region1
  clusterSelector:
    matchLabels:
      edge: region1
  yaml:
    overlays:
      - "region1"
- name: region2
  clusterSelector:
    matchLabels:
      edge: region2
  yaml:
    overlays:
      - "region2"
```

对edge cluster region1下应用  
进行配置重写

对edge cluster region2下应用  
进行配置重写



参考文档: <https://fleet.rancher.io/ref-fleet-yaml>



# Fleet应用管理/部署实践 – Fleet配置文件

```
kustomize:
  # Use a custom folder for kustomize resources. This folder must contain
  # a kustomization.yaml file.
  dir: ./kustomize
helm:
  ### These options control how "fleet apply" downloads the chart
  chart: ./chart
  repo: https://charts.rancher.io
  version: 0.1.0
  values:
    any-custom: value
  # Path to any values files that need to be passed to helm during install
  valuesFiles:
    - values1.yaml
    - values2.yaml
  releaseName: my-release
# Target customization are used to determine how resources should be modified per target
# Targets are evaluated in order and the first one to match a cluster is used for that cluster.
targetCustomizations:
# The name of target. If not specified a default name of the format "target000"
# will be used. This value is mostly for display
- name: prod
  # Custom kustomize options overriding the options at the root
  kustomize: {}
  # Custom Helm options override the options at the root
  helm: {}
```

- Kustomize 和 Helm 于 Fleet来说是共生关系，可以通过 Fleet 配置文件驱动 Kustomize 与 Helm Chart部署/升级
- 针对不同的部署目标(target)，也支持override操作

参考文档: <https://fleet.rancher.io/ref-fleet-yaml>





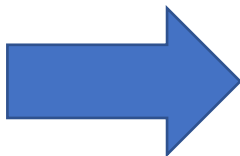
# 门户整合以及平台推广

- 门户集成案例

# 门户集成案例 - 某字母电器制造商

## 项目难点

- 1、如何与当前XX云门户进行统一对接
- 2、已容器化运行的单机Docker应用服务和VM运行的应用服务如何改造上云
- 3、平台运行在openstack上，存储对接CEPH，问题范围划分和职责划分
- 4、以有的Kubernetes集群及部署的业务如何进行管理

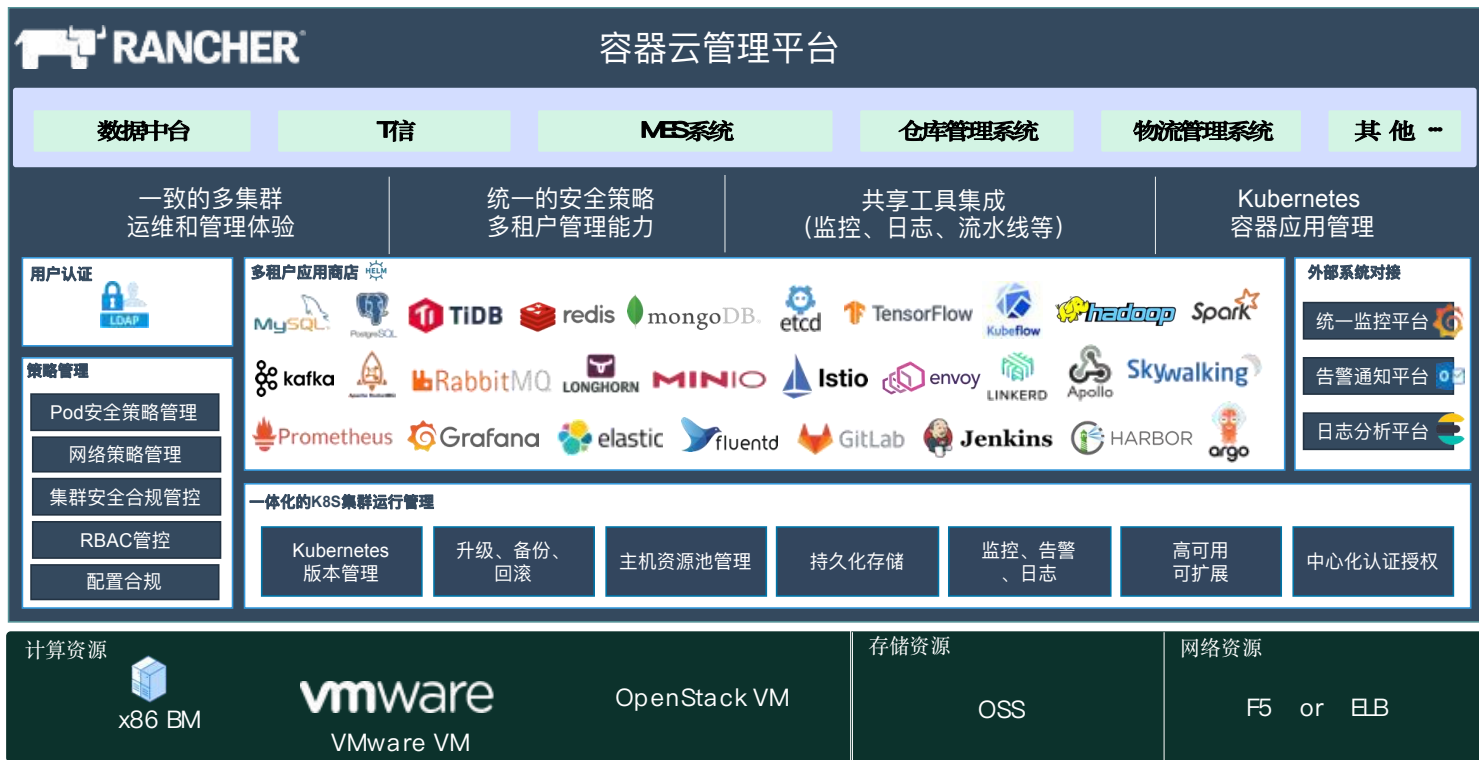


## 处理方式

- 1、确认展示功能，梳理对接接口，配合接口对接调试
- 2、梳理用户需要上云的系统，划分出kubernetes支撑的和VM支撑的，对于kubernetes划分有状态和无状态，制定迁移计划，对于原VM支撑的应用，梳理其技术架构，选择典型进行尝试，积累经验再内部逐步推广
- 3、上线前进行各项功能、性能测试数据归档，提供常见问题处理手册和技术培训指导。
- 4、实现先导入纳管，后迁移统一管理的方案



# 门户集成案例 - 某字母电器制造商







# 门户集成案例 - 某国产计算平台制造商

## 所有集群统一建设和管理

- 业务向kubernetes集群都从Rancher平台创建并进行管理，原集群业务迁移到Rancher集群运行
- 为DevOps平台管理集群提供支撑，DevOps平台调用Rancher接口实现多集群发布



## 成立独立部门后续对内推广

- 为客户提供容器管理标准化经验，在新部门成立的道路上减少障碍
- 配合客户将现容器解决方案，推广到其他应用系统，逐步进行迁移



# 门户集成案例 - 某国产计算平台制造商





## Q&A