



# OneCloud V3 容器化之路

# 目录

1. 为什么容器化？
2. 容器化改造过程
3. 容器化遇到的挑战
4. Q&A

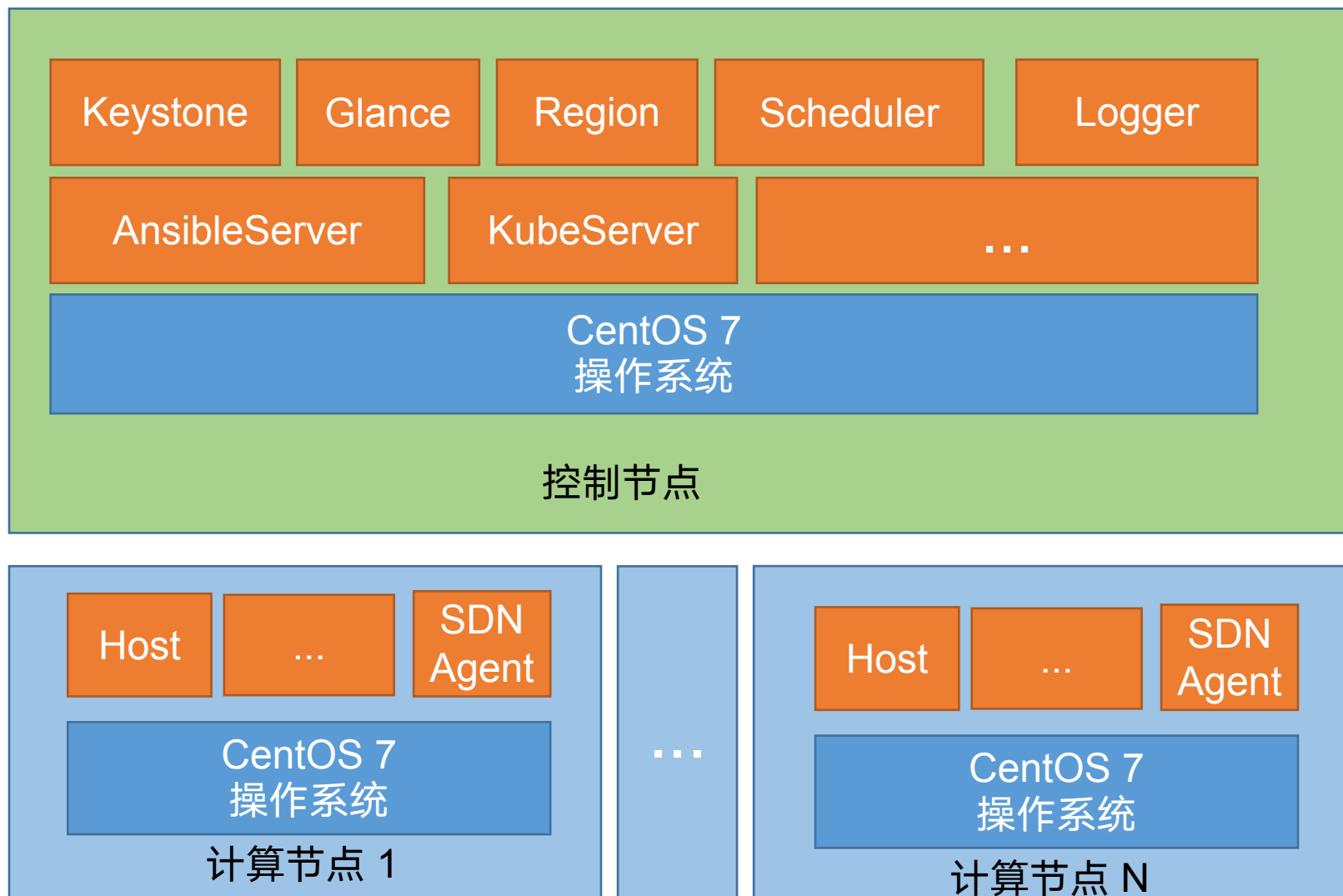
1

# 为什么容器化？

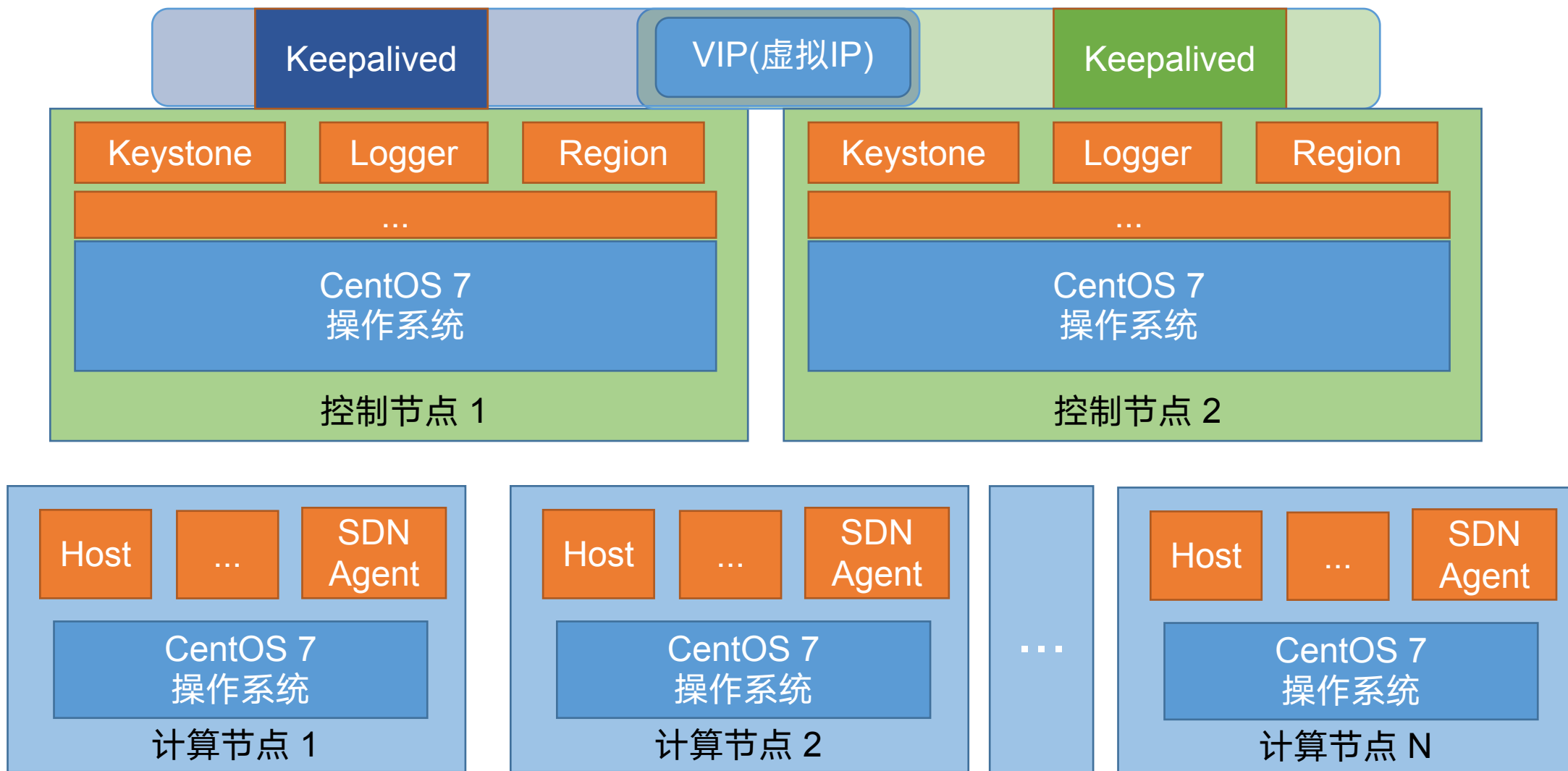


# OneCloud V2 部署方式


- 只做了 CentOS 7 发行版适配
- 控制节点部署运行 30+ 服务
- 多个计算节点部署相同服务



# OneCloud V2 高可用部署方式



## V2 部署方式的不足

- 控制节点部署服务方式固定，30+服务在同一节点
  - 服务无法分布式运行
- 私有云**计算节点服务部署升级麻烦**
  - SSH 远程逐个登录计算节点
- 服务配置管理麻烦
- 在其它发行版上运行适配工作量大
  - 比如：客户要求**在 OpenSUSE ** 上部署 OneCloud

# 开发 OneCloud V3

- 解决 V2 部署方式的不足
- 全新的部署方式: 服务容器化、运行在 K8S(Kubernetes) 之上

# 什么是 K8S (Kubernetes)



- 生产可用的主流容器管理系统
- 支持容器化应用分布式运行
- 提供配置管理、升级回滚等功能
- 丰富的资源对象模型: Deployment, Daemonset, ConfigMap 等等



# 常用 K8S 资源模型简介

## 应用类

资源名称	作用
Deployment	抽象无状态服务，可多副本运行，比如: (nginx, tomcat)
Statefulset	抽象有状态服务，可多副本运行，比如: (zookeeper, kafka)
Daemonset	每个节点都要运行的服务，比如：每个节点都要部署运行监控 agent
CronJob	周期性运行的服务，不需要长期后端运行

# 常用 K8S 资源模型简介

## 其他类型

资源名称	作用
ConfigMap	抽象服务的配置，配置可以通过挂载的方式注入容器应用
PersistentVolumeClaim	对应需要持久化存储的数据卷，比如 mysql 容器服务需要保存数据到 /var/lib/mysql
Service	实现容器之间的服务发现和负载均衡

# OneCloud V3 与 K8S 资源对应表

K8S 应用类资源名称	OneCloud V3 使用方式
Deployment	运行 keystone, region, scheduler 等无状态控制层面服务
Daemonset	运行 Host, SDN Agent 等需要在每个计算节点都运行的服务
CronJob	运行定期从公有云拉取监控数据的服务
ConfigMap	保存各个服务的配置，实现配置统一管理
PersistentVolumeClaim	用于一些需要保存持久化数据的服务，比如 glance 服务需要保存虚拟机的镜像
Service	用于暴露服务，比如 keystone 认证服务需要暴露端口提供集群外部访问

# 为什么基于 K8S 运行 OneCloud 服务？

## 发展趋势

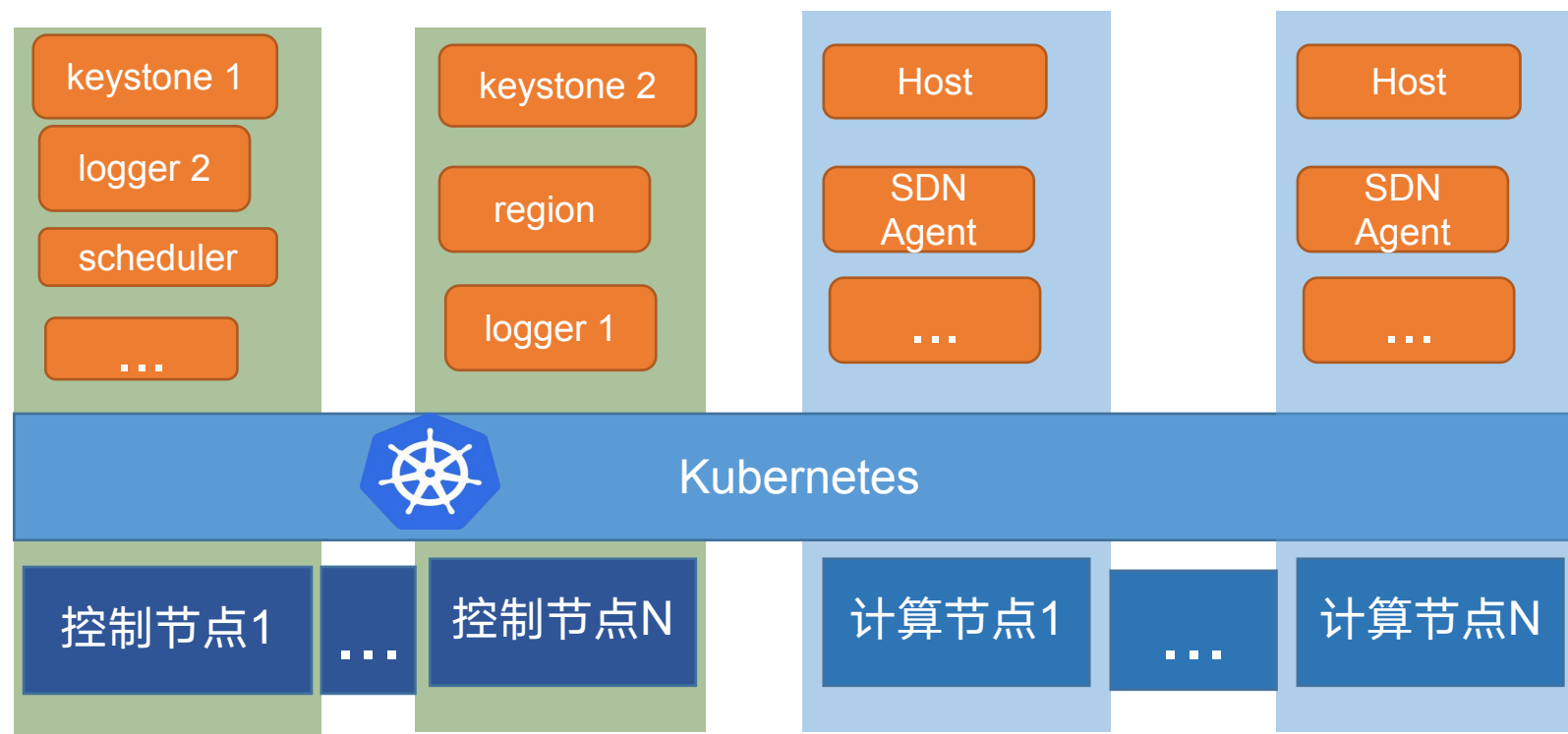
- 服务容器化运行是主流方式
- K8S 已经是容器管理平台的事实标准
- 主流服务都有在 K8S 上运行的方案

## 解决V2问题

- K8S 天生支持容器应用多副本、分布式运行
- Daemonset 模型解决多计算节点运行相同服务的需求
- K8S 默认提供配置统一管理、服务升级回滚等操作
- K8S 屏蔽操作系统发版细节

# OneCloud V3 架构图

- 多节点组成集群
- 控制节点服务多副本分布式运行
- 计算节点保证运行相同服务



## V2/V3 关键操作对比

操作	V2 传统方式	V3 使用 K8S
部署计算节点服务	SSH 登录到各个计算节点依次部署	使用 Daemonset 资源自动在多个计算节点部署服务
服务配置变更	登录服务所在节点，修改配置文件，重启服务	修改服务配置对应的 ConfigMap，重启对应的容器
升级服务	登录到服务对应节点，下载升级安装包升级	直接修改对应 Deployment (Daemonset、CronJob) 的 image 属性，K8S 会自动升级

# 2

## 容器化改造过程

2.1 源代码到 Docker 镜像

2.2 部署 K8S 集群

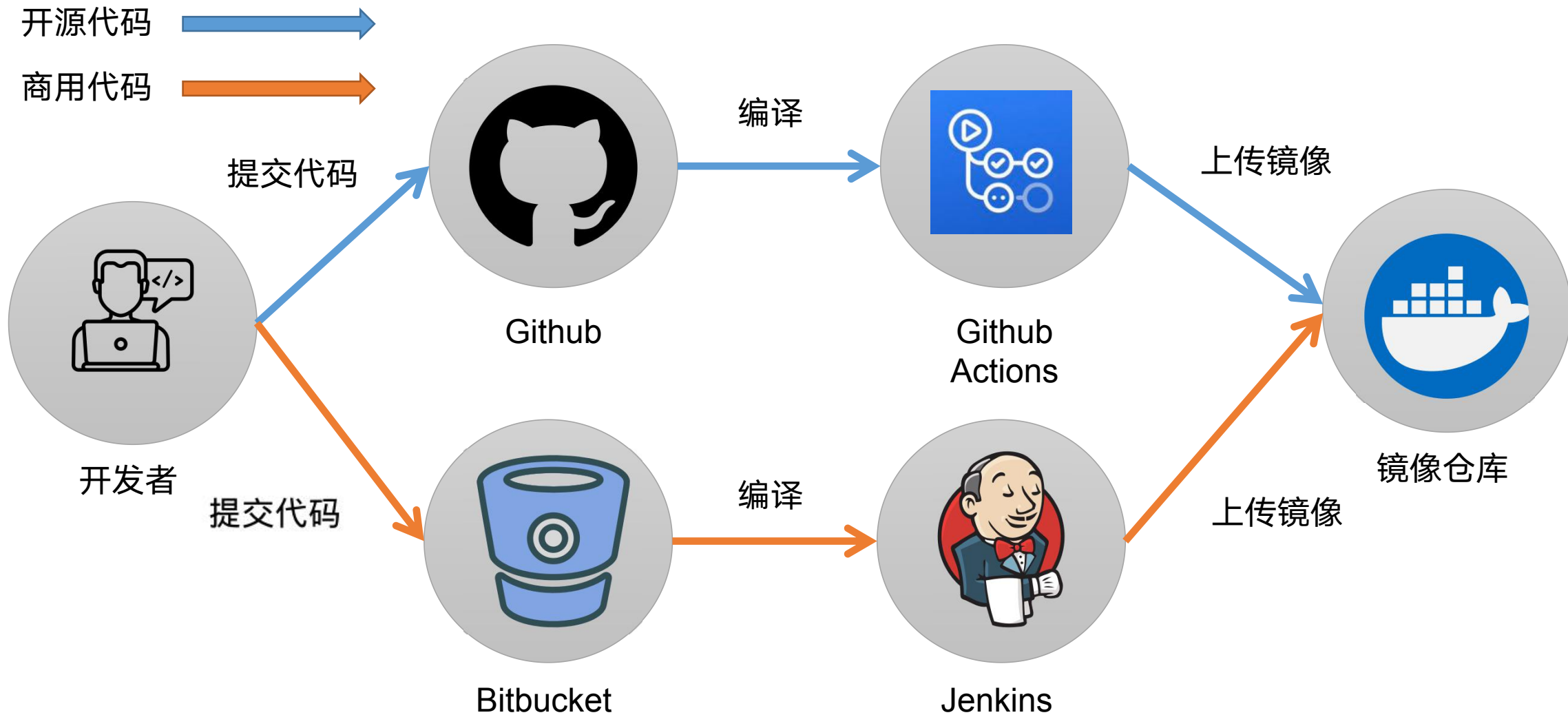
2.3 K8S 上运行 OneCloud



## 2.1 源代码到 Docker 镜像



# 源代码到 Docker 镜像流程



# Github 开源代码结合 Actions

## 各服务的 Dockerfile

Branch: master ▾	onecloud / build / docker /
Qiu Jian feature: climc docker image add vim	
..	
<a href="#">Dockerfile.ansibleserver</a>	dockerfile: component use base image
<a href="#">Dockerfile.ansibleserver-base</a>	dockerfile: component use base image
<a href="#">Dockerfile.apigateway</a>	fix: 整理改进Dockerfile
<a href="#">Dockerfile.baremetal-agent</a>	host deployer base version up
<a href="#">Dockerfile.baremetal-base</a>	fix baremetal base dockerfile
<a href="#">Dockerfile.climc</a>	feature: climc docker image add vim
<a href="#">Dockerfile.climc-base</a>	feature: climc docker image add vim
<a href="#">Dockerfile.cloudevent</a>	fix: 整理改进Dockerfile
<a href="#">Dockerfile.cloudnet</a>	fix: 整理改进Dockerfile
<a href="#">Dockerfile.devtool</a>	feature: add devtool dockerfile
<a href="#">Dockerfile.esxi-agent</a>	feature(esxi-agent): Esxi Agent Golang

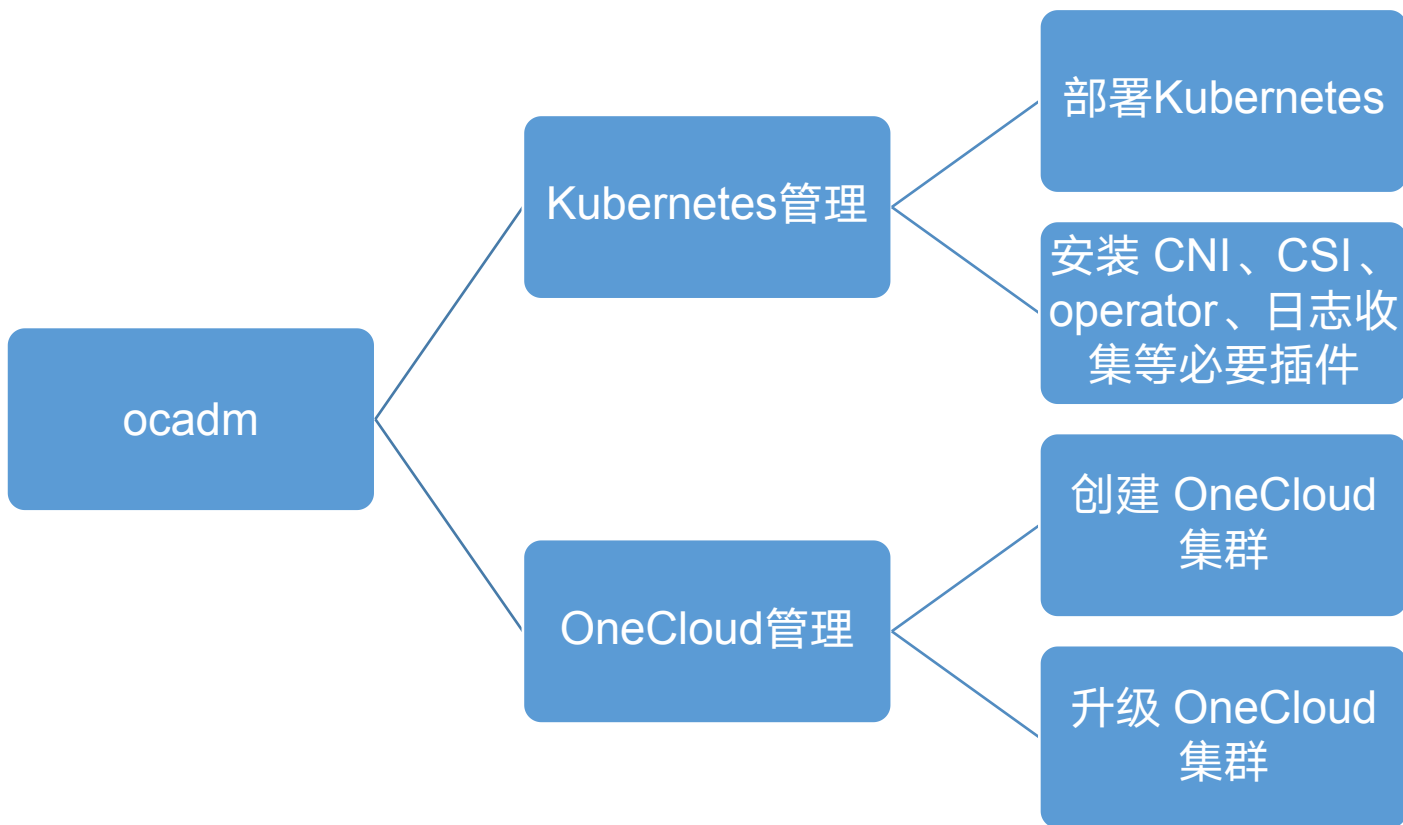
## Github Actions 编译上传镜像

```
✔ Image baremetal-agent
✔ Image esxi-agent
✔ Image vpcagent
✔ Image climc
23 a63597a6b288: Already exists
24 3db94142d0e2: Already exists
25 b31edacf7615: Pulling fs layer
26 658481e367b4: Pulling fs layer
27 658481e367b4: Verifying Checksum
28 658481e367b4: Download complete
29 b31edacf7615: Verifying Checksum
30 b31edacf7615: Download complete
31 b31edacf7615: Pull complete
32 658481e367b4: Pull complete
33 Digest: sha256:24f8edaeabd474a0e9a78cdec3ac39d47eb88156d5c53167ebd4482924f48390
34 Status: Downloaded newer image for registry.cn-beijing.aliyuncs.com/yunionio/climc-base:v20200308
35 ---> aed4bbd47a31
36 Step 2/3 : ADD ./build/climc/root/opt /opt
37 ---> 553f1bcd73b4
38 Step 3/3 : ADD ./_output/bin/climc ./_output/bin/*cli /opt/yunion/bin/
39 ---> e463311f8839
40 Successfully built e463311f8839
41 Successfully tagged registry.cn-beijing.aliyuncs.com/yunionio/climc:latest
42 Successfully tagged registry.cn-beijing.aliyuncs.com/yunionio/climc:20200324133953e0b1b1
43 The push refers to repository [registry.cn-beijing.aliyuncs.com/yunionio/climc]
44 30351a5d65dd: Preparing
45 6ec5fa161f37: Preparing
46 0017b07446da: Preparing
```

## 2.2 如何部署 Kubernetes 集群

# 如何部署 Kubernetes 集群

- 开发 ocadm 集群部署管理工具
  - 基于官方的 kubeadm 部署工具进行包装
  - 代码仓库: <https://github.com/yunionio/ocadm>



## ocadm 使用举例

```
# 1. 创建 Kubernetes 集群
ocadm init \
    --mysql-host "$MYSQL_HOST" \
    --mysql-user root \
    --mysql-password "$MYSQL_PASSWORD"

# 2. 创建 v3.1.1 OneCloud 集群
ocadm cluster create --version v3.1.1

# 3. 升级 OneCloud 集群到 v3.1.2
ocadm cluster update --version v3.1.2
```

## 2.3 K8S 之上运行 OneCloud

# K8S 运行 OneCloud 面临的问题

- OneCloud 总共有 30+ 个服务，如何统一部署到 K8S？
- 部署到 K8S 里面的 OneCloud 服务怎么统一升级回滚？

# K8S 部署服务的方式 - kubectl

- kubectl: K8S 原生的命令行工具

优点	缺点
开箱即用，小巧轻量，能够创建 K8S 原生资源	没有对原生资源对象再做封装，要创建一个服务往往需要编写大量的 yaml 配置文件
	OneCloud 后端要部署 30+ 个服务，如果用 kubectl 部署手写 yaml 的方式将不可能维护



# K8S 部署服务的方式 - Helm

- Helm: K8S 包管理工具



优点	缺点
能够组合 K8S 多种任意资源，一键创建、升级、回滚对应资源	需要编写较多模板，复杂业务不易编写
	不能实现更复杂的服务运维逻辑，比如实现服务的定时更新

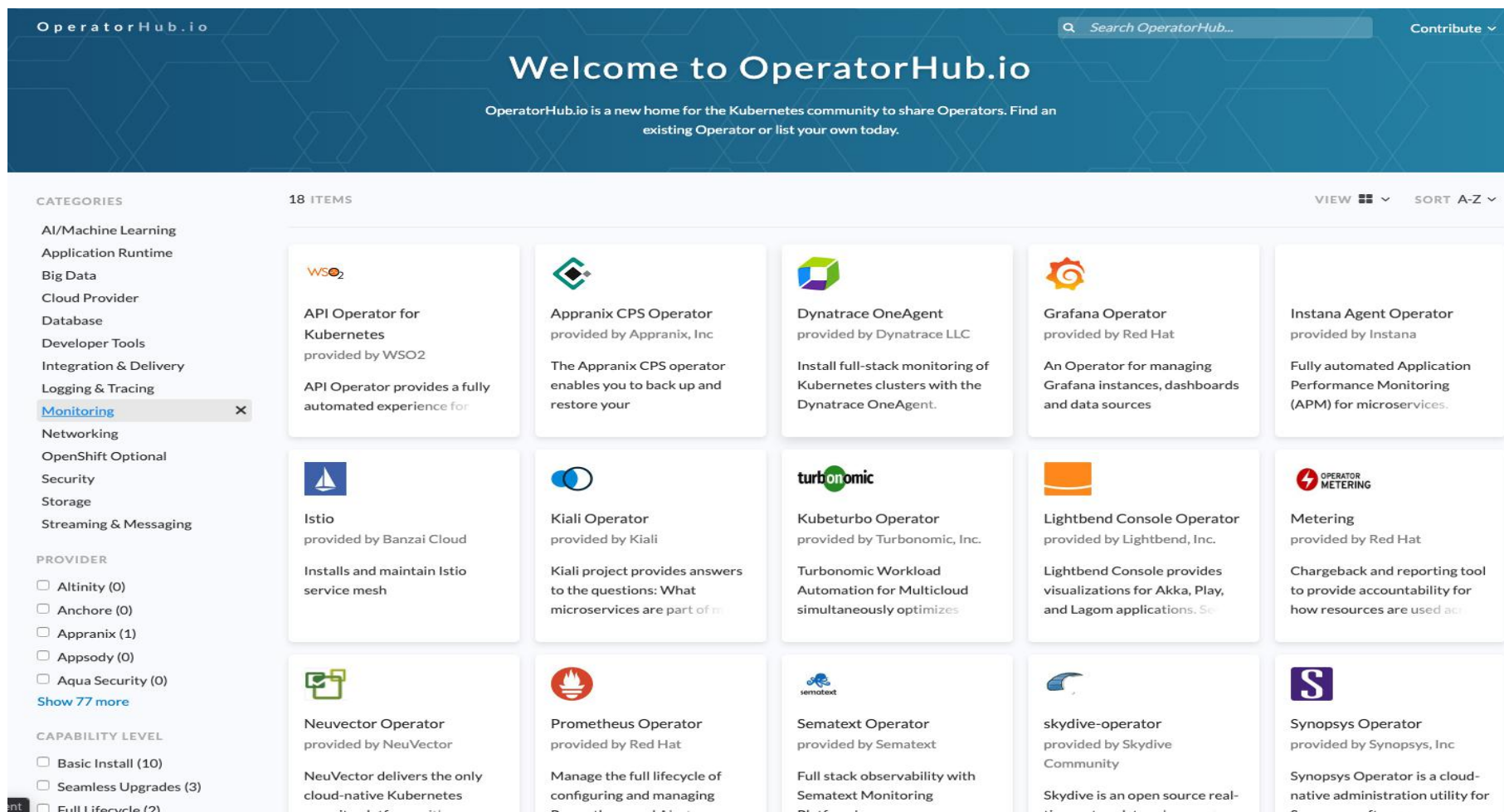
# K8S 部署服务的方式 - Operator

- Operator: K8S 自定义资源操作定制机制

优点	缺点
足够灵活，通过自定义资源的机制能够实现任意服务的部署、升级等操作	需要额外的代码开发
业内运维部署复杂业务的推荐做法，已成为趋势	

# K8S 部署服务的方式 - Operator

OperatorHub: <https://operatorhub.io/>



The screenshot displays the OperatorHub.io homepage. At the top, a dark blue header contains the site name, a search bar, and a 'Contribute' link. Below the header, a large banner reads 'Welcome to OperatorHub.io' with a subtitle: 'OperatorHub.io is a new home for the Kubernetes community to share Operators. Find an existing Operator or list your own today.'

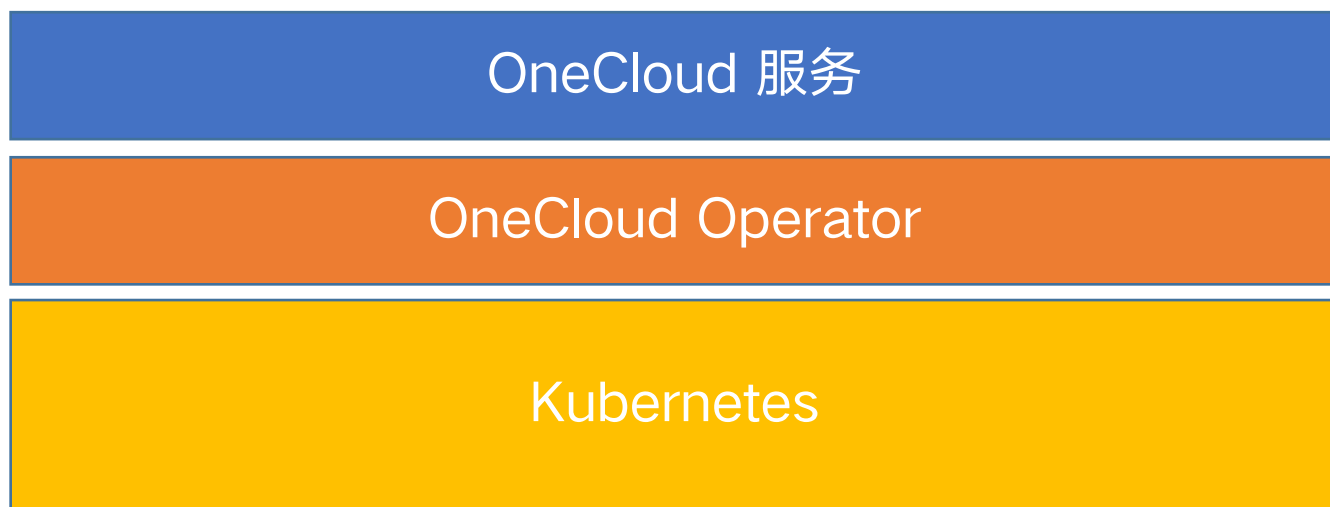
The main content area is divided into a left sidebar and a central grid of 18 operator cards. The sidebar includes filters for 'CATEGORIES' (e.g., AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Integration & Delivery, Logging & Tracing, Monitoring, Networking, OpenShift Optional, Security, Storage, Streaming & Messaging) and 'PROVIDER' (a list of providers with counts, such as Altinity (0), Anchore (0), Appratrix (1), Appsody (0), Aqua Security (0), and a 'Show 77 more' link). There is also a 'CAPABILITY LEVEL' section with filters like Basic Install (10), Seamless Upgrades (3), and Full Lifecycle (2).

The central grid displays 18 operator cards, each with a logo, name, provider, and a brief description. The operators shown include:

- API Operator for Kubernetes (provided by WSO2)
- Appratrix CPS Operator (provided by Appratrix, Inc.)
- Dynatrace OneAgent (provided by Dynatrace LLC)
- Grafana Operator (provided by Red Hat)
- Instana Agent Operator (provided by Instana)
- Istio (provided by Banzai Cloud)
- Kiali Operator (provided by Kiali)
- Kubeturbo Operator (provided by Turbonomic, Inc.)
- Lightbend Console Operator (provided by Lightbend, Inc.)
- Metering (provided by Red Hat)
- Neuvector Operator (provided by NeuVector)
- Prometheus Operator (provided by Red Hat)
- Sematext Operator (provided by Sematext)
- skydrive-operator (provided by Skydrive Community)
- Synopsys Operator (provided by Synopsys, Inc.)

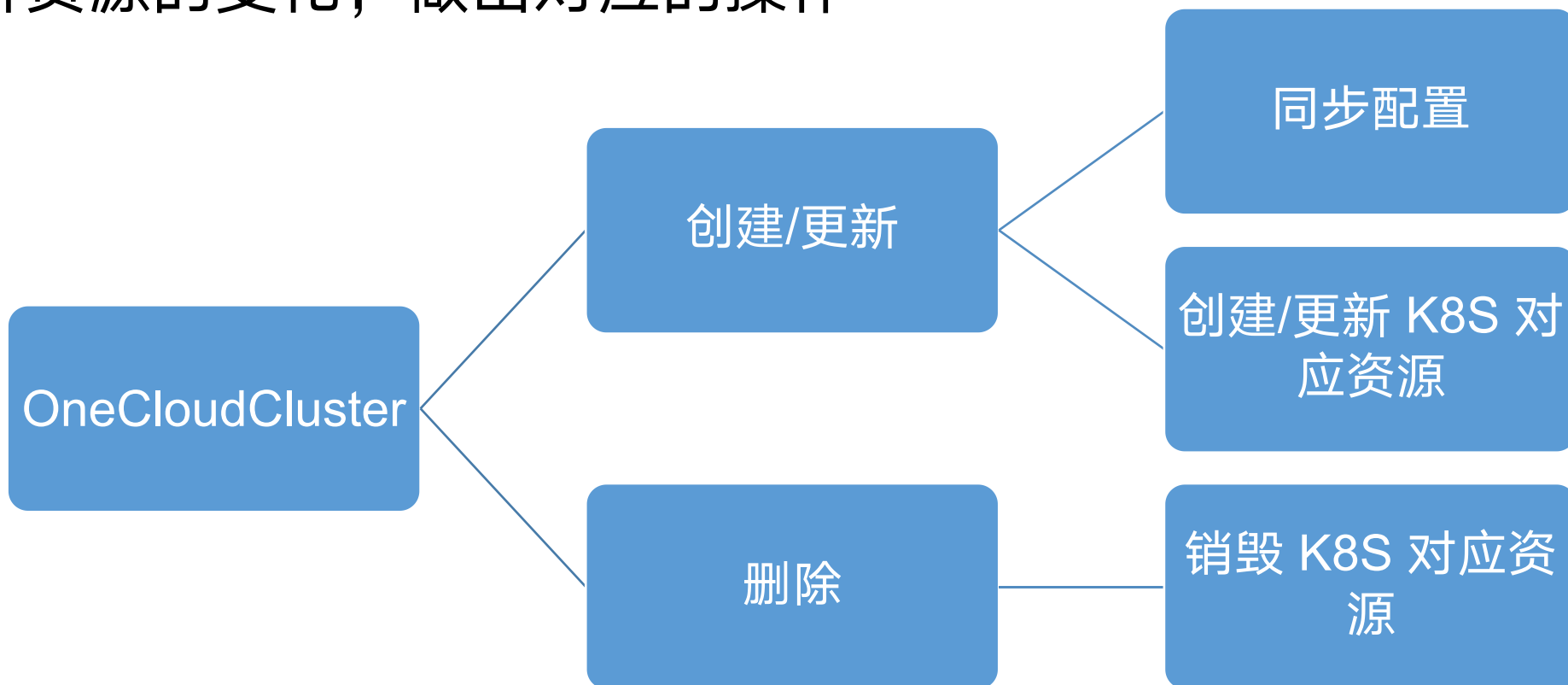
# 如何在 K8S 之上运行 OneCloud

- 开发 OneCloud Operator 工具部署所有服务
  - 参考主流的 K8S Operator 概念实现 OneCloud 服务的部署
  - 代码仓库: <https://github.com/yunionio/onecloud-operator>



# Operator 的原理

- 利用 K8S 自定义资源模型机制，抽象出 OneCloud Cluster 资源
- 监听资源的变化，做出对应的操作



# OneCloud Operator 的作用

## 自动部署 OneCloud 服务

- 初始化配置
- 创建管理对应 K8S 资源

## 升级回滚

- 更新/回滚所有服务版本
- 更新/回滚指定服务版本

## 资源清理

- OneCloud 集群删除，释放对应资源



# Operator 部署 OneCloud 服务

```
[root@lzx-k8s-node1 ~]# cat example-onecloud-cluster.yaml
```

```
apiVersion: "onecloud.yunion.io/v1alpha1"
```

```
kind: OnecloudCluster
```

```
metadata:
```

```
  name: "default"
```

```
  namespace: "onecloud"
```

```
spec:
```

```
  mysql:
```

```
    host: 10.168.26.216
```

```
    username: root
```

```
    password: "your-sql-password"
```

```
  region: "region0"
```

```
  imageRepository: registry.cn-beijing.aliyuncs.com/yunionio
```

```
  version: v3.1.3-20200312.0
```

```
[root@lzx-k8s-node1 ~]# kubectl create -f example-onecloud-cluster.yaml
```

```
onecloudcluster.onecloud.yunion.io/default created
```





# Operator 部署 OneCloud 服务

```
[root@lzx-k8s-node1 ~]# kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
default-keystone-7cb57b46cb-hpj49	1/1	Running	0	7s
default-region-6fb6d7bf8f-w6m2r	0/1	Init:0/1	0	2s
onecloud-operator-58794bd46c-hrfv5	1/1	Running	2	77m
default-region-6fb6d7bf8f-w6m2r	0/1	PodInitializing	0	2s
default-region-6fb6d7bf8f-w6m2r	1/1	Running	0	3s
default-scheduler-7f794dc884-k9r45	0/1	Pending	0	0s
default-scheduler-7f794dc884-k9r45	0/1	Pending	0	0s
default-scheduler-7f794dc884-k9r45	0/1	ContainerCreating	0	0s
default-scheduler-7f794dc884-k9r45	0/1	ContainerCreating	0	0s
default-glance-7886766478-4k88r	0/1	Pending	0	0s
default-glance-7886766478-4k88r	0/1	Pending	0	0s
default-glance-7886766478-4k88r	0/1	Init:0/1	0	0s
default-scheduler-7f794dc884-k9r45	1/1	Running	0	1s
default-glance-7886766478-4k88r	0/1	Init:0/1	0	1s
default-ansibleserver-6678fbcdd9-mfw4c	0/1	Pending	0	0s
default-ansibleserver-6678fbcdd9-mfw4c	0/1	Pending	0	0s
default-ansibleserver-6678fbcdd9-mfw4c	0/1	ContainerCreating	0	0s
default-glance-7886766478-4k88r	0/1	PodInitializing	0	2s





# Operator 部署 OneCloud 服务

default-notify-7c446c9787-g6j5q	0/1	Running	0	6s
default-cloudnet-5f647bd5df-m8jnl	0/1	ContainerCreating	0	5s
default-s3gateway-6dc7bd9578-nf6qk	1/1	Running	0	7s
default-host-deployer-gs92x	1/1	Terminating	0	10s
default-vpcagent-6				
default-cloudevent				
default-logger-c79				
default-yunioncon				
default-monitor-6k				
default-autoupdate				
default-influxdb-5				
default-notify-7c4				
default-devtool-84				
default-webconsole				
default-webconsole				
default-host-deplo				
default-host-deplo				
default-host-deplo				
default-host-deplo				
default-host-deplo				
default-cloudnet-5				
default-cloudevent				
default-web-76d8c8				
default-notify-7c4				
default-host-deplo				
default-host-deployer-fzm6k	1/1	Running	0	3s

OneCloud

← → ↻ ⚠ Not secure | 10.168.26.216/auth/login

OneCloud

统一

统一模版/API/调度/账号体系/监控/控制台/计费计量  
实现物理机/虚拟机/容器/公有云资源全面纳管

用户登录

请输入用户名

请输入密码

登录

Made with ♥ Yunion



# Operator 升级 OneCloud 服务

```
apiVersion: onecloud.yunion.io/v1alpha1
kind: OnecloudCluster
metadata:
  generation: 16
  labels:
    app.kubernetes.io/instance: onecloud-cluster-7v96
  name: default
  namespace: onecloud
spec:
  ...
  version: v3.1.3-20200320.0
  vpcAgent:
    disable: false
    image: registry.cn-beijing.aliyuncs.com/yunionio/vpcagent:v3.1.3-20200312.0
    imagePullPolicy: Always
    replicas: 1
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        key: node-role.kubernetes.io/controlplane
```

修改 version 版本号, operator 会自动更新对应 k8s 资源



# Operator 删除 OneCloud 服务

```
[root@lzx-k8s-node1 ~]# kubectl delete -f example-onecloud-cluster.yaml
onecloudcluster.onecloud.yunion.io "default" deleted
```

```
[root@lzx-k8s-node1 ~]# kubectl get pods -n onecloud
```

NAME	READY	STATUS	RESTARTS	AGE
default-ansibleserver-684747ddb-f-b4tf4	1/1	Terminating	0	29m
default-apigateway-7d9f8877cc-2d5lv	1/1	Terminating	0	29m
default-autoupdate-75f9fff5b7-gjvwt	1/1	Terminating	0	28m
default-climc-7995fc6bb4-bdz68	1/1	Terminating	0	28m
default-cloudevent-8bb8d589c-wrx94	1/1	Terminating	0	28m
default-cloudnet-b579548c9-rsgsm	1/1	Terminating	0	28m
default-devtool-5b5f6965f8-9fhrz	1/1	Terminating	0	28m
default-esxi-agent-597d86f748-jkjh6	1/1	Terminating	0	28m
default-glance-7f4dd9d4b4-hx5sn	1/1	Terminating	0	29m
default-influxdb-5ffdb75fc8-b7rrn	1/1	Terminating	0	28m
default-keystone-6f7747bcf8-f4trc	1/1	Terminating	0	29m
default-logger-b88c97b74-ggd2m	1/1	Terminating	0	28m
default-meter-799b7fcd6-8z8pg	1/1	Terminating	0	28m
default-monitor-6cf5fd6c75-cn7cw	1/1	Terminating	0	28m
default-notify-6d54cd4bbf-gwckc	5/5	Terminating	0	28m
default-ovn-north-78b84dfd6f-vxw7j	1/1	Terminating	0	28m
default-region-757bc649dc-ns2g5	1/1	Terminating	0	29m
default-s3gateway-67c8b5d956-h7f82	1/1	Terminating	0	28m
default-scheduler-5b78f9fcd-5vrtj	1/1	Terminating	0	29m
default-vpcagent-6d554b9985-rvsjg	1/1	Terminating	0	28m
default-web-57f8c9b466-fb7kh	1/1	Terminating	2	29m
default-webconsole-5c9784fb94-6vs8j	1/1	Terminating	0	28m
default-yunionconf-7686b6ff9d-7j88b	1/1	Terminating	0	28m



## 举例: Operator 自动部署 glance 虚拟机镜像服务

```
5 apiVersion: onecloud.yunion.io/v1alpha1
6 kind: OnecloudCluster
7 spec:
8   ...
9   glance:
10     disable: false
11     image: registry.cn-beijing.aliyuncs.com/yunionio/glance:v3.1.3-20200312.0
12     imagePullPolicy: Always
13     replicas: 1
14     requests:
15       storage: 100G
16     storageClassName: local-path
17     tolerations:
18     - effect: NoSchedule
19       key: node-role.kubernetes.io/master
20     - effect: NoSchedule
21       key: node-role.kubernetes.io/controlplane
```

## 举例: Operator 部署 glance 服务

根据 OneCloud Cluster 里面的 glance 定义创建/更新以下的 K8S glance 资源:

K8S 资源	作用
ConfigMap	glance 服务需要的配置文件, 通过 volume 的方式挂载到容器的 /etc/yunion/glance.conf
PersistentVolumeClaim	100G 持久化存储挂载到容器中, 虚拟机的镜像会保存在该目录
Deployment	使用镜像 registry.cn-beijing.aliyuncs.com/yunionio/glance:v3.1.3-20200312.0 创建容器, 运行 glance 服务
Service	暴露 glance 容器应用的端口



# 举例: Operator 部署 glance 服务

ConfigMap 保存服务配置

```
[root@lzx-oc-node ~]# kubectl get configmaps
default-cluster-config      1      14d
default-devtool              1      14d
default-esxi-agent           1      14d
default-glance             1      14d
default-host                 1      14d
default-influxdb             1      14d
default-keystone             1      14d
```

# 举例: Operator 部署 glance 服务

PVC (PersistentVolumeClaim) 对应容器服务需要的存储

```
[root@lzx-oc-node ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
default-baremetal-agent	Pending		
default-esxi-agent	Bound	pvc-db8d6e1c-6442-4222-ad90-ff9a42ec99b1	30G
<u>default-glance</u>	Bound	pvc-ba1dbb5f-5571-4c23-ad49-1e8aaf225b5b	<u>100G</u>
default-influxdb	Bound	pvc-b40ccfc8-5c9c-4ff1-af92-3879e97d1a6d	20G
default-meter	Bound	pvc-4deca586-2e28-4728-9fa2-f8d9d7255bc2	100G
default-notify	Bound	pvc-70ef9d8a-c10e-42ce-897f-c099452ed3b9	1G

# 举例: Operator 部署 glance 服务

Deployment 对应一个容器应用

```
[root@lzx-oc-node ~]# kubectl get deployments -o=jsonpath='{range .items[*]}{"\n"}{.metadata.name}\nec.containers[*]}{.image}{" "}{end}{end}' | grep glance -C 3
default-cloudnet:      registry.cn-beijing.aliyuncs.com/yunionio/cloudnet:v3.1.3-20200312.0
default-devtool:       registry.cn-beijing.aliyuncs.com/yunionio/devtool:v3.1.3-20200312.0
default-esxi-agent:    registry.cn-beijing.aliyuncs.com/yunionio/esxi-agent:v3.1.3-20200312.0
default-glance:        registry.cn-beijing.aliyuncs.com/yunionio/glance:v3.1.3-20200312.0
default-influxdb:      registry.cn-beijing.aliyuncs.com/yunionio/influxdb:1.7.7
default-keystone:      registry.cn-beijing.aliyuncs.com/yunionio/keystone:v3.1.3-20200312.0
default-kubeserver:    registry.cn-beijing.aliyuncs.com/yunionio/kubeserver:dev
```



# 举例: Operator 部署 glance 服务

Service 对应容器服务需要暴露的端口，具备负载均衡作用

```
[root@lzx-oc-node ~]# kubectl get service | grep glance -C 3
default-cloudevent      NodePort    10.101.216.147    <none>          30892:30892/TCP
default-cloudnet        NodePort    10.105.227.166    <none>          30891:30891/TCP
default-devtool         NodePort    10.101.58.104     <none>          30997:30997/TCP
default-glance          NodePort    10.99.249.238     <none>          30292:30292/TCP
default-influxdb        NodePort    10.96.18.4        <none>          30086:30086/TCP
default-keystone        NodePort    10.110.46.218     <none>          30500:30500/TCP,30357:30357/TCP
default-kubeserver      NodePort    10.100.76.25      <none>          30442:30442/TCP
```

# 3

## 容器化遇到的挑战





# 开发方式改变

动作	传统方式	K8S 方式
部署服务	<ol style="list-style-type: none"><li>1. 编译二进制</li><li>2. cp 二进制到开发机</li><li>3. 启动服务<ul style="list-style-type: none"><li>• systemd 管理</li><li>• 或者直接启动</li></ul></li></ol>	<ol style="list-style-type: none"><li>1. 编译二进制</li><li>2. 制作上传 Docker 镜像</li><li>3. 修改对应 K8S 资源的 image</li></ol>
调试	<ul style="list-style-type: none"><li>• 直接启动看输出</li><li>• journactl 查看日志</li><li>• 日志输出到文件</li></ul>	<ul style="list-style-type: none"><li>• 使用 kubectl log 查看日志</li><li>• 使用日志收集服务查看</li></ul>

# 开发方式的改变

- 存在问题：
  - 开发人员熟悉传统开发方式，K8S 有学习成本
  - K8S 中想临时调试一个服务，制作 Docker 镜像再到部署的路线长，对开发不友好



# 解决部署路线长问题

- 编写一些脚本或者 Makefile 规则自动化步骤

```
# lzx @ lzx-t470p in ~/go/src/yunion.io/x/onecloud-operator on git
$ REGISTRY=zexi TAG=dev make image
GO111MODULE=on CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -mod
bin/onecloud-controller-manager cmd/controller-manager/main.go
sudo docker build -f images/onecloud-operator/Dockerfile -t zexi/o
Sending build context to Docker daemon 95.4MB
Step 1/2 : FROM registry.cn-beijing.aliyuncs.com/yunionio/onecloud
---> 3ab3a0b60a2d
Step 2/2 : ADD ./_output/bin/onecloud-controller-manager /bin/onec
---> Using cache
---> b33effb54fba
Successfully built b33effb54fba
Successfully tagged zexi/onecloud-operator:dbcd76eb
sudo docker push zexi/onecloud-operator:dbcd76eb
The push refers to repository [docker.io/zexi/onecloud-operator]
b9a9c9ce55ae: Preparing
1a4126283951: Preparing
3d02aaf87a41: Preparing
2fe71dfc3ce4: Preparing
0061a0e49257: Preparing
f1b5933fe4b5: Waiting
```

- 利用 kubectl cp 命令

- 利用 kubectl exec 命令

- 利用 kubectl debug 命令

[github.com/aylei/kubectl-debug](https://github.com/aylei/kubectl-debug)

# 服务日志收集

- 存在问题：
  - 容器销毁后对应的日志也会删除
- 解决方法：
  - 内部搭建日志收集系统
  - Loki 收集日志，Grafana 展示
- 优点：
  - 开源、轻量、占用资源少



# 底层服务容器化会遇到问题

- 比如 Host 虚拟机管理服务
  - 容器重启，对应的虚拟机也会重启
  - 底层服务有各种动态链接库的依赖
  - 容器里面如何加载内核模块
  - 容器里面如何执行外部宿主机命令

4

Thanks  
Q & A





## Q&A



请填写调查问卷，协助我们把活动  
办得更好、更符合您的需求，谢谢！



欢迎加入OneCloud技术交流群  
加云联小助手ID: yunionio