中小团队的K8S落地之路

元年科技 云事业部架构师

王海峰

UCLOUD 优刻得

消费

记账

发票

支付

费用报销/对公审批付款

元年 YUANIAN | 云快报 | UCAN 下午茶

功能

商旅

- 全网比价
- 贴身客服
- 实时对账

采购

- 企业商城
- 自主高效
- 实时对账

报销

- 移动审批
- 拍票识别
- 分析管控

预算

- 敏捷编制
- 预算管控
- 多维分析

目录

01 架构总览

- 镜像库

- 镜像构建服务
- 基础镜像

- 代码仓库

- Kubernetes集群

- ykb_docker_build
- jenkins
- git用户账户凭据
- 浏览器/远程桌
- win-server

01

架构总览

- Nexus3
- 镜像库

- 镜像构建服务
- 基础镜像

- coding

- Rancher
- Kubernetes集群

- ykb_docker_build
- jenkins
- git用户账户凭据
- 浏览器/远程桌
- win-server

01

架构总览

- UCloud
- 镜像库

- 镜像构建服务
- 基础镜像

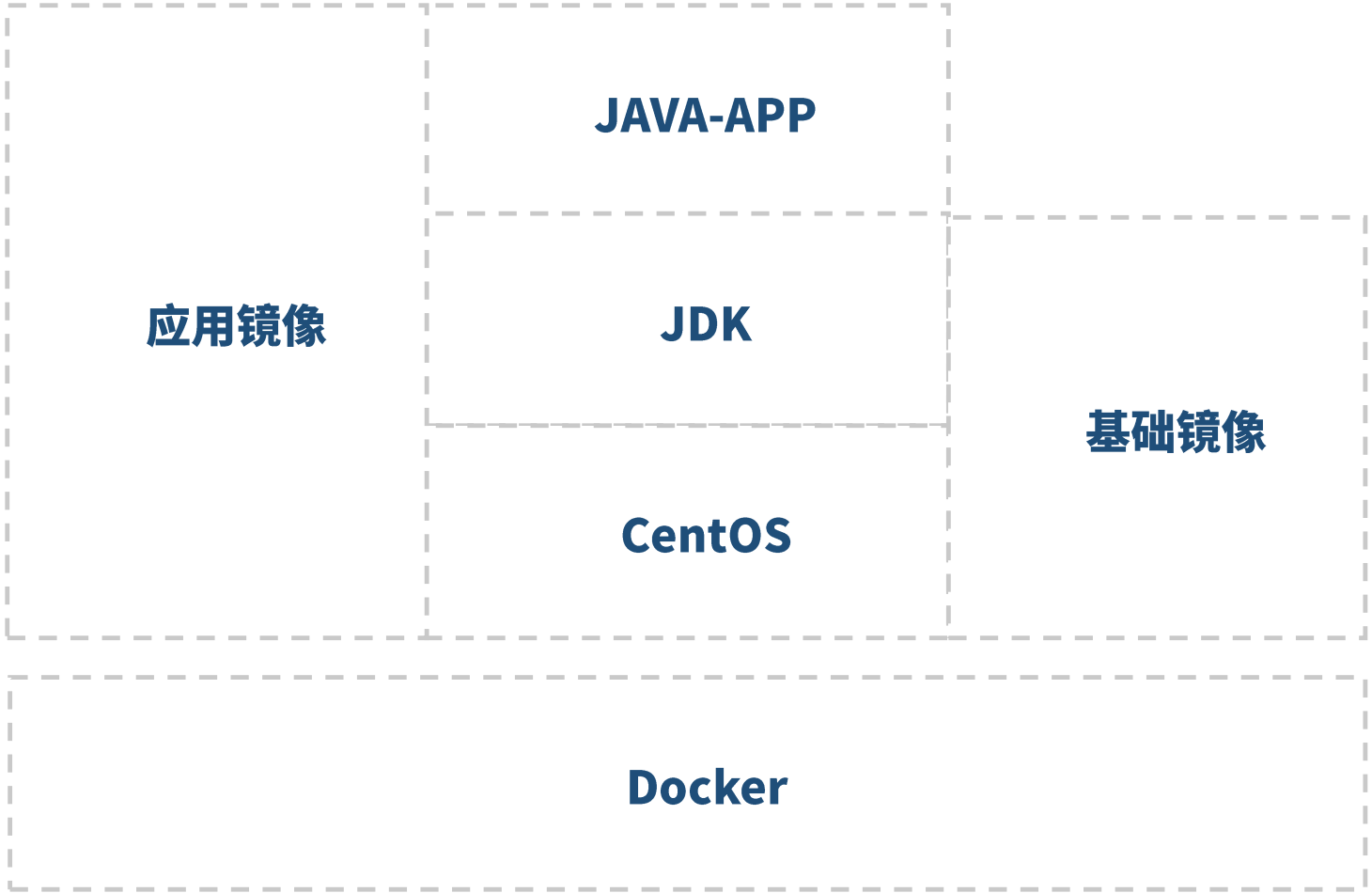
- coding

- UCloud Kubernetes集群
- API

- ykb_docker_build
- jenkins
- git用户账户凭据
- 浏览器/远程桌
- win-server

02

构建应用



02

构建应用

doc	dev	com.app.conf.info.Conf.java	Ant=>
	test	com.app.conf.info.Conf.java	
	release	com.app.conf.info.Conf.java	
src	配置	com\app\conf\info\Conf.java	
	源码	****	
	xml配置	log4j2.xml	
pub	静态文件	**	
lib	依赖jar	**.**	
webdefault.xml	Jetty配置文件		

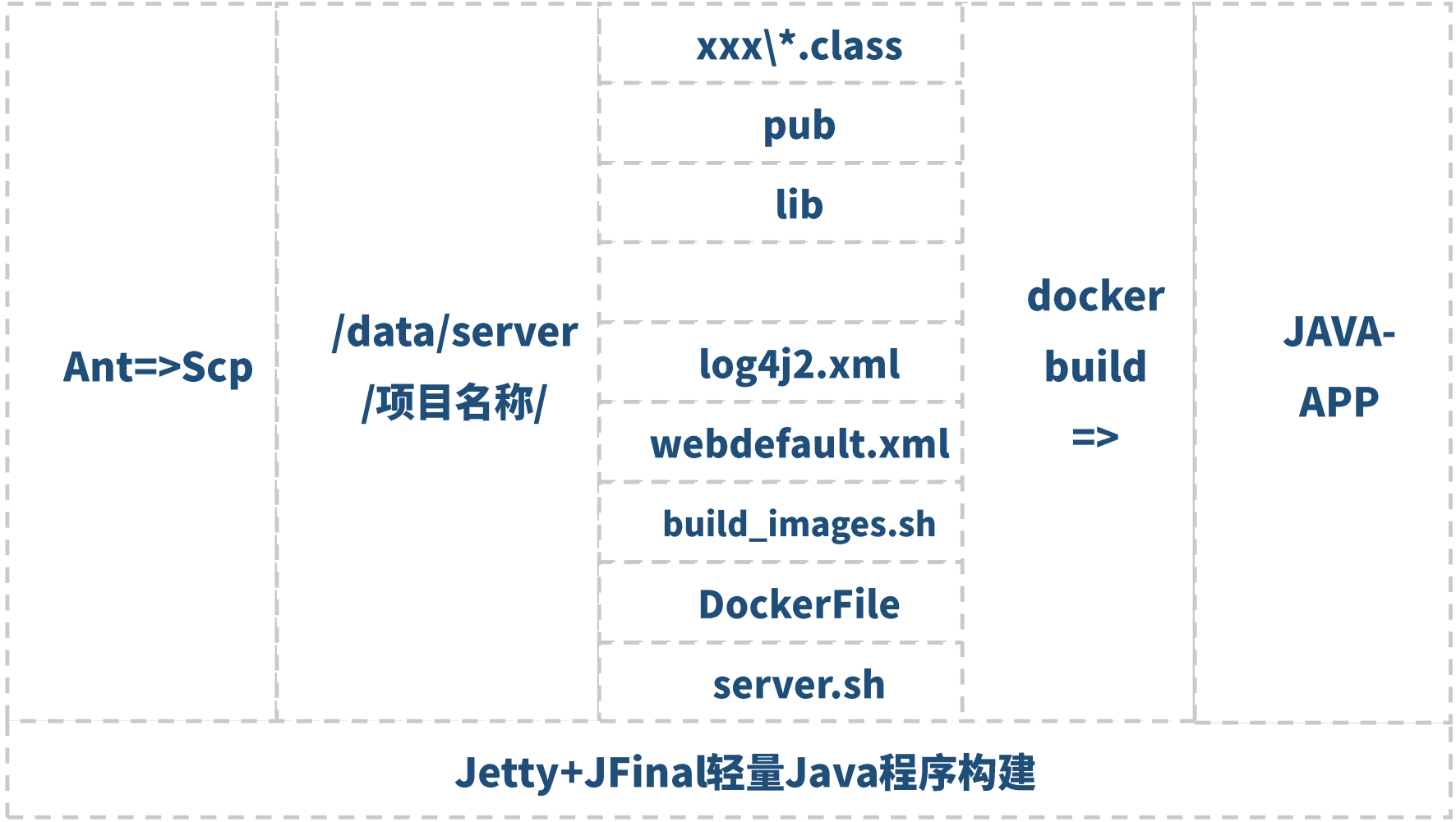
03

构建应用

Ant=>	WebContent	发布文件	xxx*.class	Ant=>Scp
			pub	
			lib	
			log4j2.xml	
			webdefault.xml	
			build_images.sh	
			DockerFile	
			server.sh	
Jetty+JFinal轻量Java程序构建				

03

构建应用


Ant=>Scp
/data/server
/项目名称/
xxx*.class
pub
lib
log4j2.xml
webdefault.xml
build_images.sh
DockerFile
server.sh
docker
build
=>
JAVA-
APP
Jetty+JFinal轻量Java程序构建

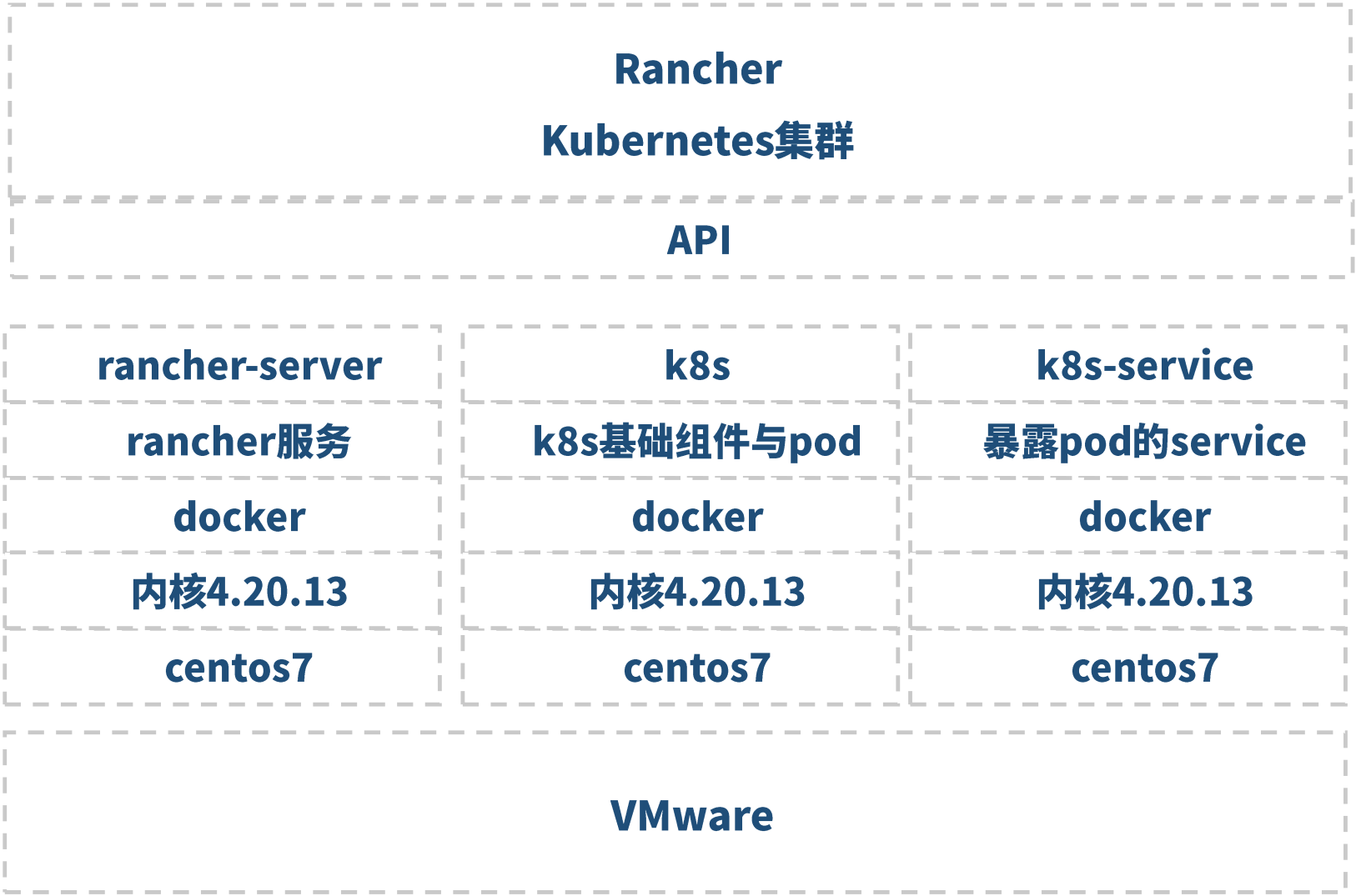
03

构建应用

https://gitee.com/aisao/full_container_station_scheme

https://gitee.com/aisao/k8s_java_project_demo

03 构建线下K8S集群



04 K8S-API-删除应用

```
curl -k -X DELETE -H "Authorization:Bearer %ApiKey%"
-H "Content-Type: application/yaml"
-T "del.yml"
%ApiUrl%/apis/apps/v1beta1/namespaces/%k8sNamespace%/deployments/%k8sNam
e%
curl\curl -k -X DELETE -H "Authorization:Bearer %rancherApiKey%"
-H "Content-Type: application/yaml"
-T "del.yml"
%ApiUrl%/api/v1/namespaces/%k8sNamespace%/services/%k8sName%

#删除-del.yml
gracePeriodSeconds: 0
orphanDependents: false
```

04 K8S-API-部署应用

```
curl -k -X POST -H "Authorization:Bearer %ApiKey%"
-H "Content-Type: application/yaml"
-T "%k8sNamespace%_%k8sName%.yml"
%ApiUrl%/apis/extensions/v1beta1/namespaces/%k8sNamespace%/deployments
```

```
curl -k -X POST -H "Authorization:Bearer %ApiKey%"
-H "Content-Type: application/yaml"
-T "%k8sNamespace%_%k8sName%.yml"
%ApiUrl%/apis/extensions/v1beta1/namespaces/%k8sNamespace%/services
```

04 K8S-API-部署模板

```yaml
#应用
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  #名称
  name: {项目名称}
  #命名空间
  namespace: {命名空间}
spec:
  #启动应用数量
  replicas: 1
  #从容器启动到应用正常提供服务
  minReadySeconds: 10
  #策略
  strategy:
    #更新类型
    type: RollingUpdate
    #滚动更新
    rollingUpdate:
      #更新时允许最大激增的容器数
      maxSurge: 1
      #更新时允许最大 unavailable 容器数
      maxUnavailable: 0
  #模板
  template:
    #数据描述[元数据]
    metadata:
      #标签
      labels:
        #应用
        app: {项目名称}
        #版本
        version: {自定义版本号}
```

04 K8S-API-部署模板

```
    #规格
    spec:
      #调度规则,指定部署在指定标签节点中  调度key: 调度值 默认为rancher compute=true标签主机
      nodeSelector:
        {标签名称}: {标签值}
      #容器拉取权限
      imagePullSecrets:
        - name : {拉取镜像密码字典}
      #容器
      containers:
        #名称
      - name: {项目名称}
        #镜像
        image: {镜像库地址}
        #镜像挂载目录
        volumeMounts:
        - name: time
          mountPath: /etc/localtime
        #计算资源配置
        resources:
          #启动限制
          requests:
            #初始化CPU用量
            cpu: "100m"
            #初始化内存用量
            memory: "128Mi"
          #最大限制
          limits:
            #限制CPU最大用量
            cpu: "100m"
            #限制内存最大用量
            memory: "256Mi"
```

04 K8S-API-部署模板

```
66          #环境变量
67          env:
68          #堆内存
69          - name: heap
70          #堆内存值,例子128m
71            value: "{heap}"
72          #存活探针
73          livenessProbe:
74            #get方式
75            httpGet:
76              #检测链接
77              path: {k8sWorkingCheck}
78              #端口
79              port: 80
80              #方式
81              scheme: HTTP
82            #初始化
83            initialDelaySeconds: 60
84            #检测间隔
85            periodSeconds: 5
86            #超时时间
87            timeoutSeconds: 5
88            #应用错误判定次数
89            failureThreshold: 3
90        #实际目录挂载
91        volumes:
92        - name: time
93          hostPath:
94            path: /etc/localtime
```

04 K8S-API-部署模板

```yaml
#服务
kind: Service
apiVersion: v1
#数据描述[元数据]
metadata:
  #名称
  name: {项目名称}
  #命名空间
  namespace: {命名空间}
  #标签
  labels:
    #对应的应用标签
    app: {项目名称}
    #对应的应用版本
    version: {自定义版本号}
#规格
spec:
  #容器
  selector:
    #应用
    app: {项目名称}
    #对应的应用版本
    version: {自定义版本号}
  #端口
  ports:
    #名称
    - name: http
      #协议
      protocol: TCP
      #对外开放端口
      port: {K8s外部访问端口}
      #容器开放端口
      targetPort: 80
```

04 K8S-API-部署模板

```
34  ··#类型
35  ··type:·ClusterIP
36  ··#对外开放的IP地址
37  ··externalIPs:
38  ····#地址
39  ····-·{K8s外部访问地址}
```

04 K8S-API-部署模板


```
curl -k -X PATCH -H "Authorization:Bearer %ApiKey%"
-H "Content-Type: application/strategic-merge-patch+json"
-T "%k8sNamespace%_%k8sName%.json"
%ApiUrl%/apis/apps/v1beta1/namespaces/%k8sNamespace%/deployments/%k8sName%


#更新%k8sNamespace%_%k8sName%.json
{
  "spec": {
    "template": {
      "spec": {
        "containers": [{
          "name": "%k8sName%",
          "image": "%imagesRegistryIp%:%imagesRegistryPort%/%name%:%tag%"
        }]
      }
    }
  }
}
```

THANK YOU

UCLOUD 优刻得