Java架构师必备技能之

kubernetes与微服务的完美结合

第 1 部分

势不可挡的云原生

什么是云原生（cloud native）架构？

容器化：作为应用包装的载体

持续交付：利用容器的轻便的特性，构建持续集成和持续发布的流水线

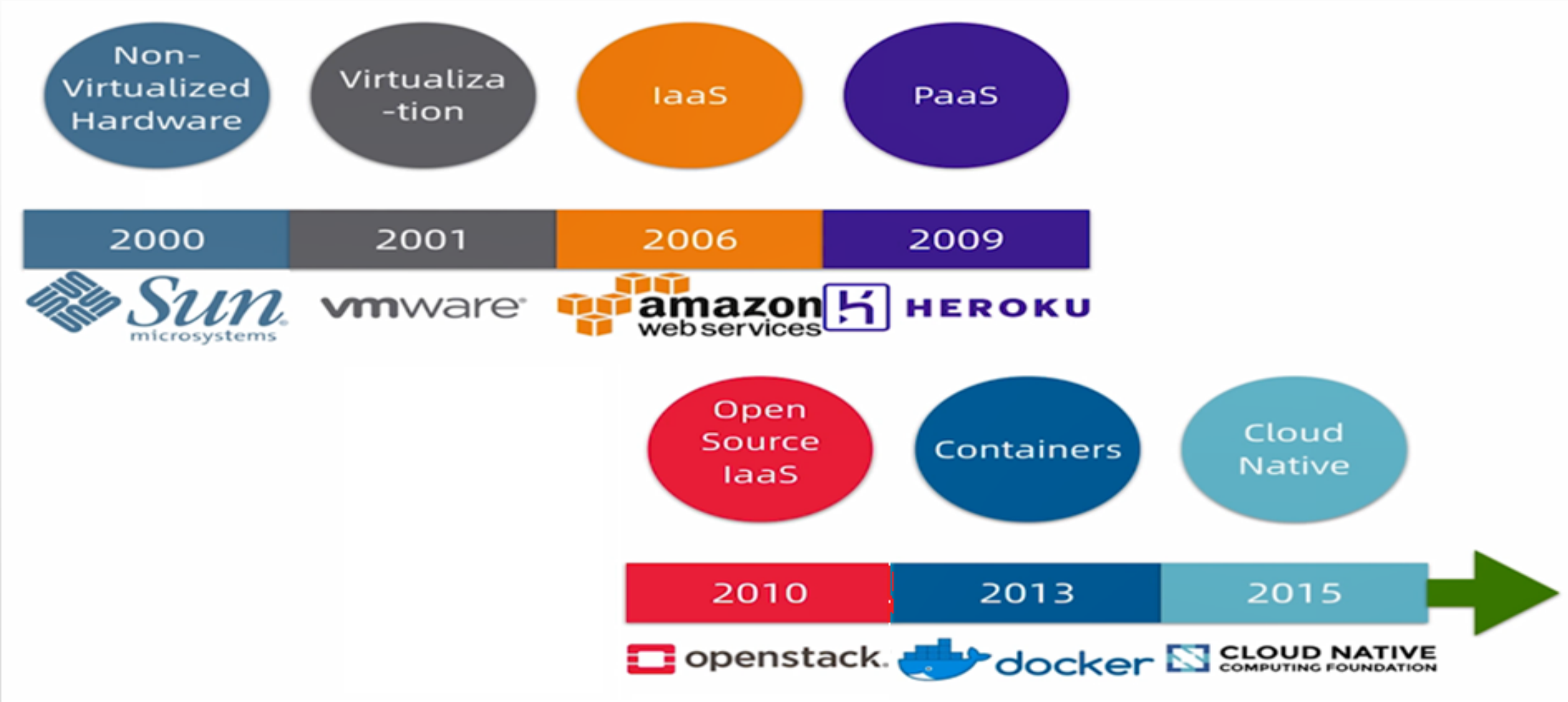
DevOps：开发与运维之间的协同，上升到一种文化的层次，能够让应用快速的部署和发布

微服务：这是应用开发的一种理念，将单体应用拆分为微服务才能更好的实现云原生，才能独立的部署、扩展和更新

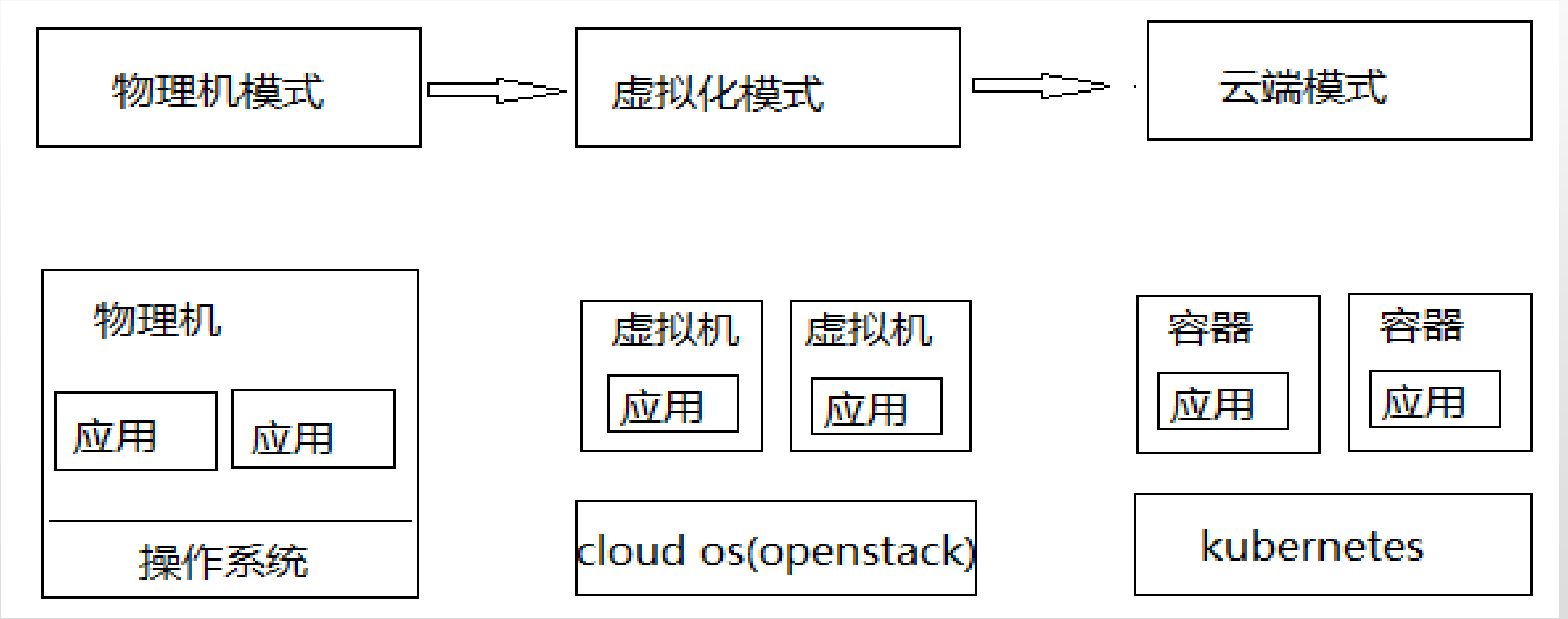
一句话解释什么是云原生应用：云原生应用就是为了在云上运行而开发的应用

云原生是一条最佳路径或者最佳实践。更详细的说，***云原生为用户指定了一条低心智负担的、敏捷的、能够以可扩展、可复制的方式最大化地利用云的能力、发挥云的价值的最佳路径。***

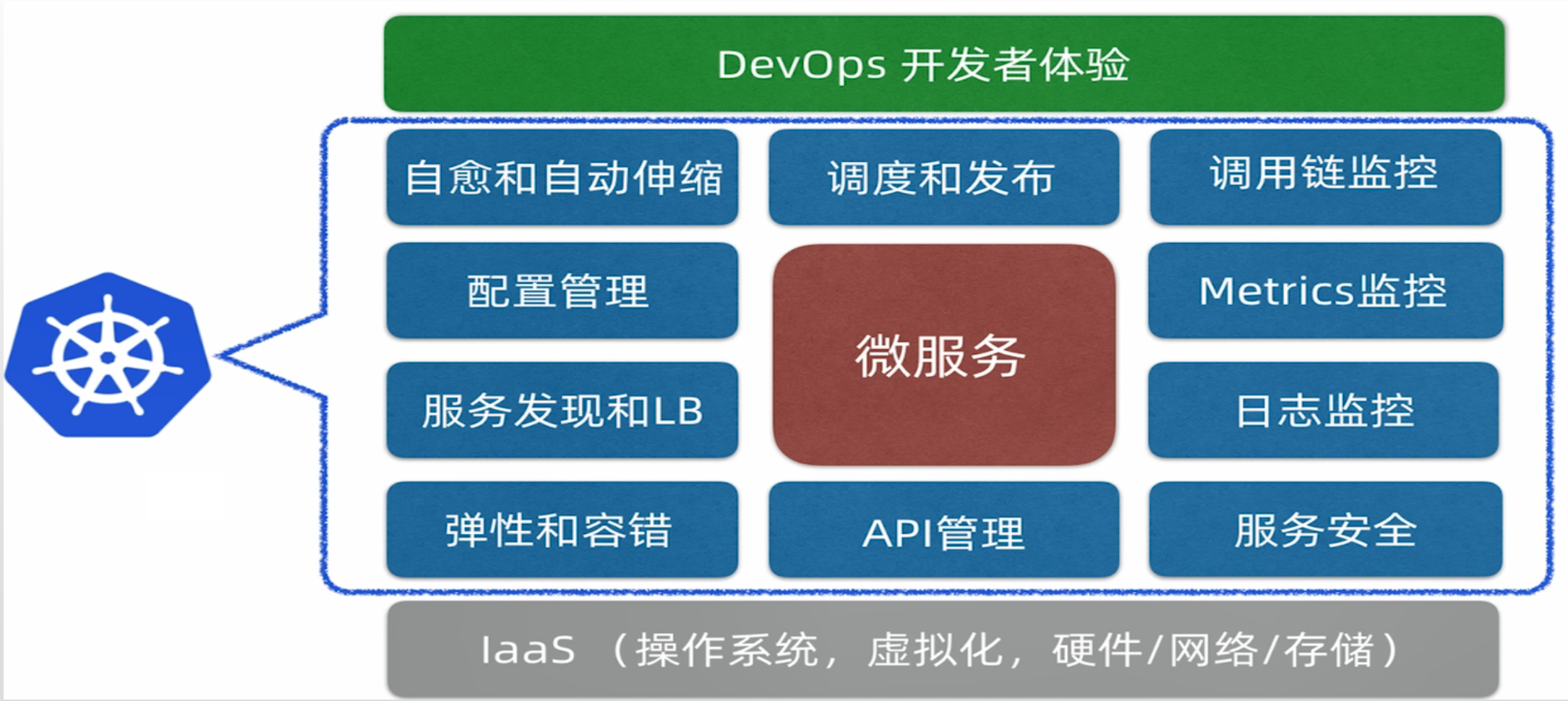
云架构演进发展


Non-Virtualized Hardware
Virtualiza-tion
IaaS
PaaS
2000
2001
2006
2009
Sun microsystems
vmware
amazon web services
HEROKU
Open Source IaaS
Containers
Cloud Native
2010
2013
2015
openstack.
docker
CLOUD NATIVE COMPUTING FOUNDATION

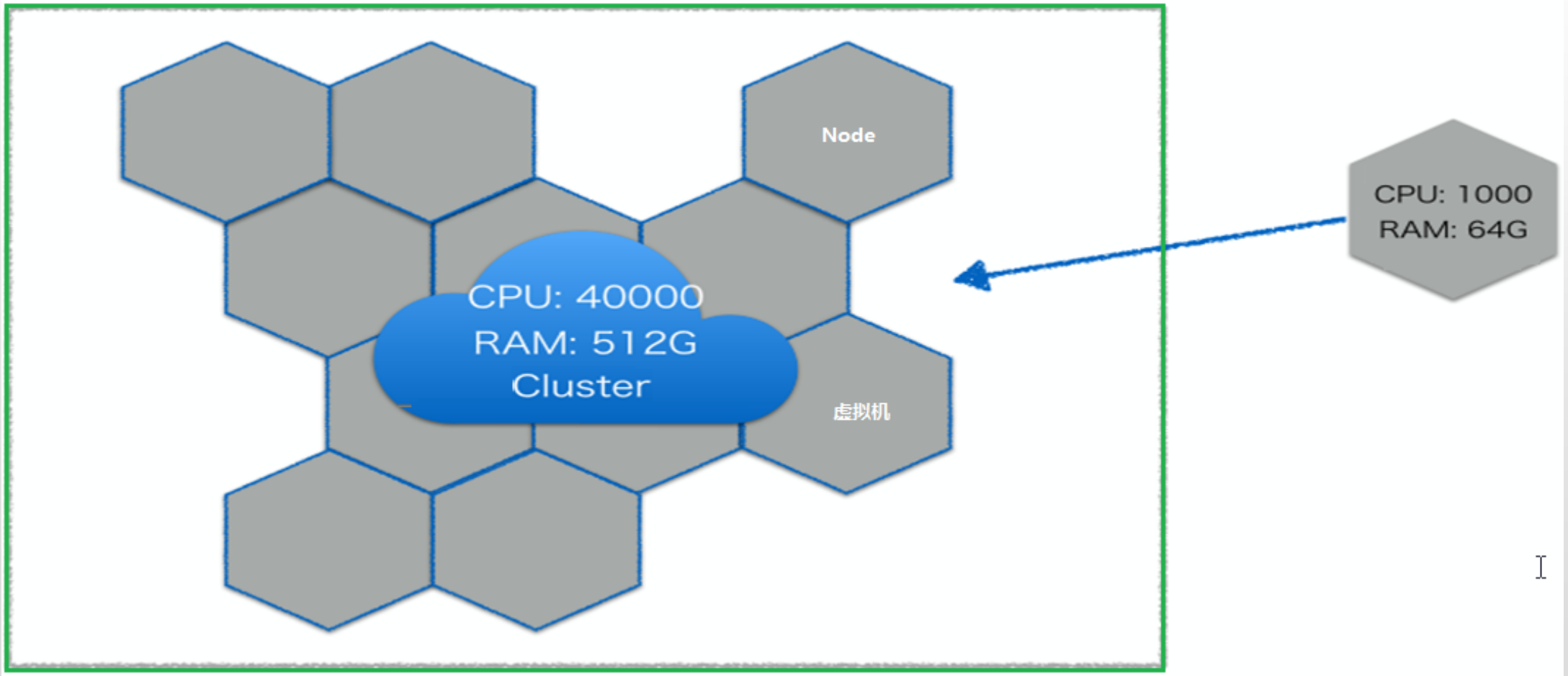
Kubernetes云原生&微服务



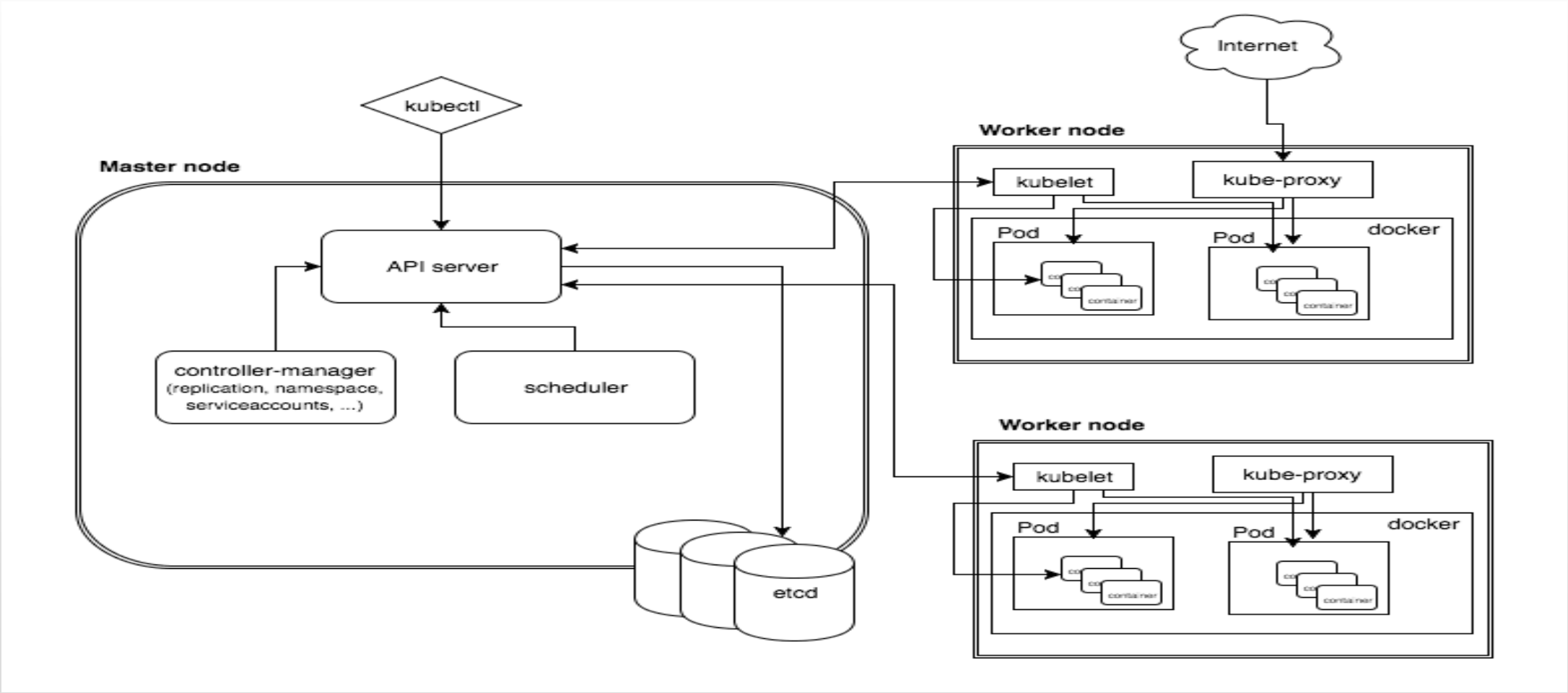
Kubernetes 解决了什么问题？



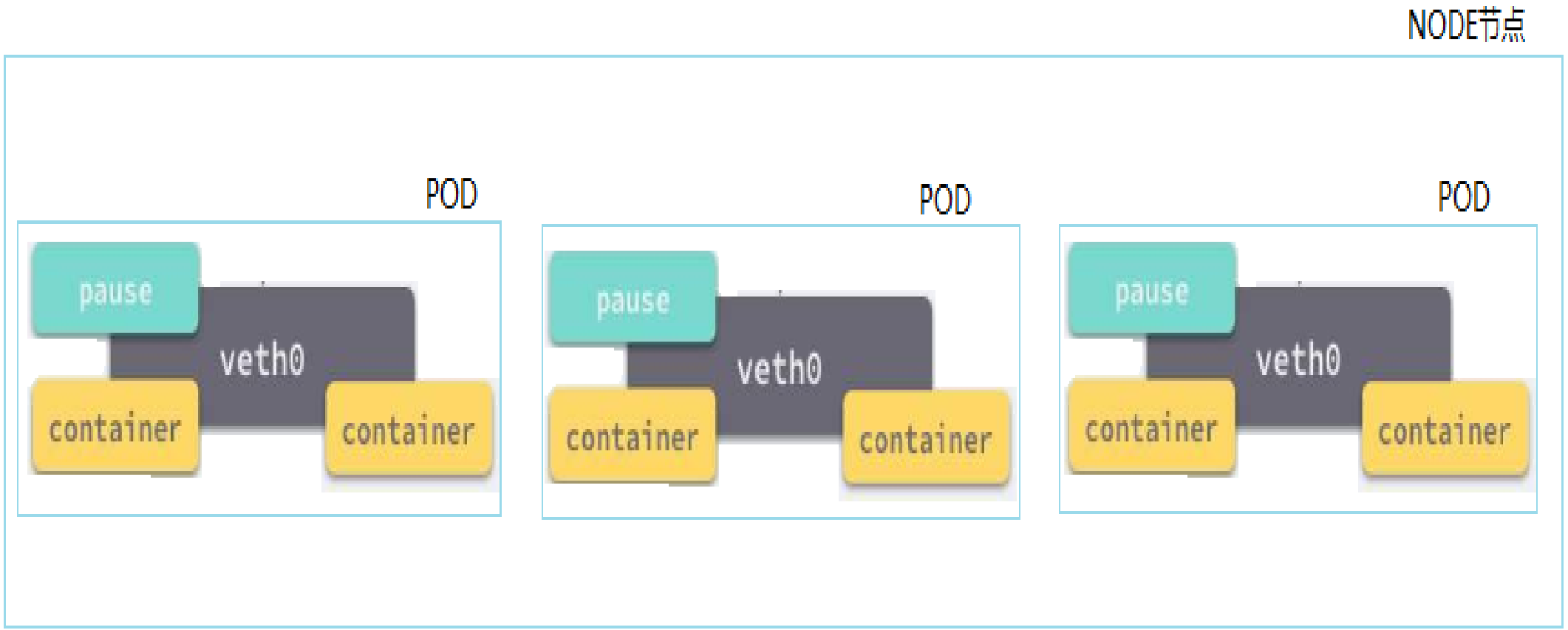
Kubernetes集群



Kubernetes 架构



Kubernetes POD



第 2 部分

Kubernetes指令服务部署及原理分析

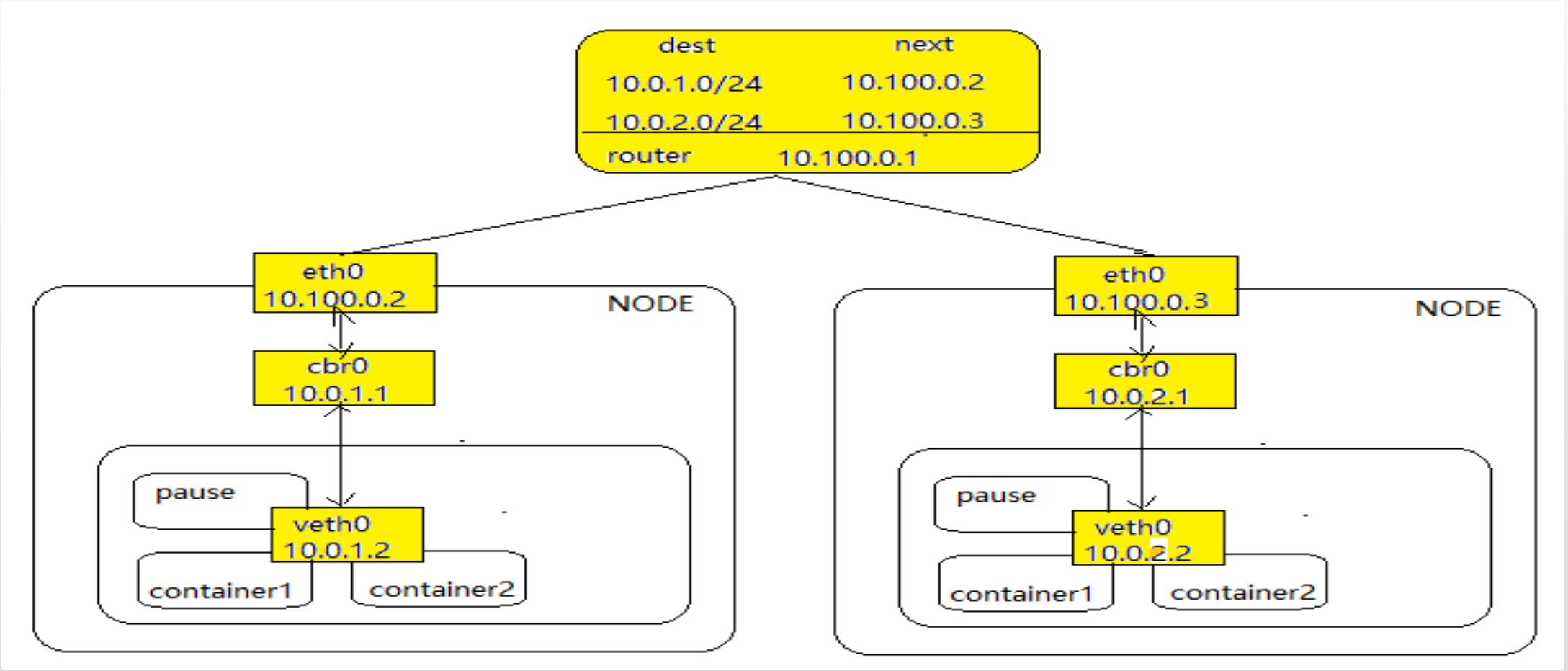
#创建且运行一个pod
#deployment、rs、pod被自动创建
kubectl run my-nginx --image=hub.kaikeba.com/library/myapp:v1 --port=80

#增加创建副本数量
kubectl scale deployment/my-nginx --replicas = 3

#添加service
#kubectl expose将RC、Service、Deployment或Pod作为新的Kubernetes Service公开。
kubectl expose deployment/my-nginx --port=30000 --target-port=80

#编辑service配置文件
kubectl edit svc/my-nginx

Kubernetes 节点和Pod网络



第 3 部分

Kubernetes之YAML方式部署服务

Nginx部署

参考部署笔记

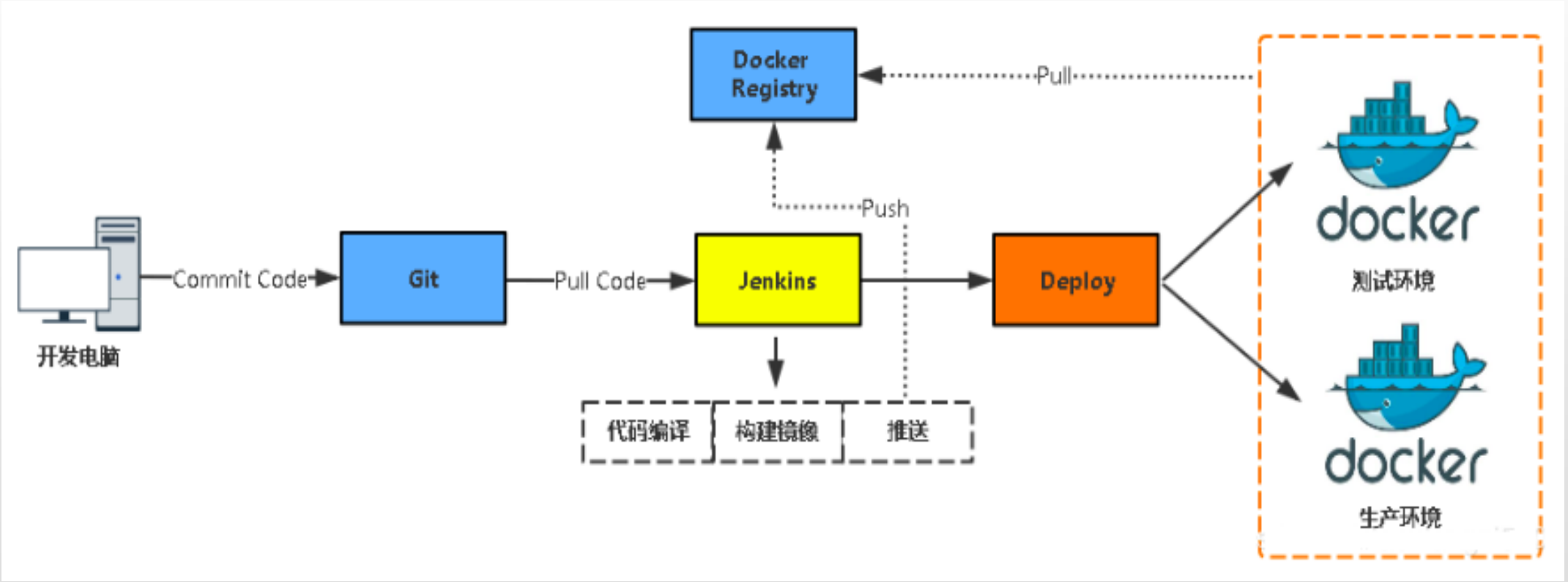
MYSQL部署

参考部署笔记

第 4 部分

Jenkins实现CI/CD

Jenkins


Docker Registry
Pull
Push
开发电脑
Commit Code
Git
Pull Code
Jenkins
Deploy
代码编译
构建镜像
推送
docker
测试环境
docker
生产环境

Project structure

Maven build

S	W	名称 ↓	上次成功	上次失败	上次持续时间
		eureka-one	12 天 - #14	13 天 - #12	1 分 1 秒
		java10-supergo-eureka	没有	3 小时 29 分 - #4	1 分 27 秒
		supergo-eureka	5 分 33 秒 - #3	7 分 20 秒 - #2	18 秒
		supergo-user	26 秒 - #3	2 分 42 秒 - #2	16 秒
		supergo-zuul	22 分 - #8	24 分 - #7	28 秒

Root POM: pom.xml

Goals and options: clean package

Maven Docker

```
<configuration>
    <!--dockerfile 指令：变成插件配置-->
    <!--用于指定镜像名称-->
    <imageName>hub.kaikeba.com/supergo/${project.artifactId}:${project.version}</imageName>
    <!--用于指定基础镜像，相当于Dockerfile中的FROM指令-->
    <!--FROM jdk1.8:v1-->
    <baseImage>hub.kaikeba.com/library/jdk1.8:v1</baseImage>
    <!--指定工作目录-->
    <!--<workdir>/</workdir>-->
    <maintainer>ithubin@163.com</maintainer>
    <cmd>["java","-version"]</cmd>
    <!--相当于Dockerfile的ENTRYPOINT指令-->
    <!--dockerfile : entryPoint-->
    <entryPoint>["java","-jar","/${project.build.finalName}.jar"]</entryPoint>
    <serverId>my-docker-registry</serverId>
    <!--是否跳过docker build-->
    <!--<skipDockerBuild>true</skipDockerBuild>-->
```

Root POM	pom.xml
Goals and options	clean package docker:build -DpushImage

第 5 部分

Dockerfile方式完成部署

Dockerfile

```
FROM hub.kaikeba.com/library/jdk1.8:v1

COPY ./target/app.jar /usr/app

WORKDIR /usr/app

ENTRYPOINT ["java","-jar","app.jar"]
```

小爱AI音箱

Thank you!

第 6 部分

答疑环节