

Docker+K8S+DevOps 微服务架构师

学神 IT 教育：从零基础到实战，从入门到精通！

版权声明：

本系列文档为《学神 IT 教育》内部使用教材和教案，只允许 VIP 学员个人使用，禁止私自传播。否则将取消其 VIP 资格，追究其法律责任，请知晓！

免责声明：

本课程设计目的只用于教学，切勿使用课程中的技术进行违法活动，学员利用课程中的技术进行违法活动，造成的后果与讲师本人及讲师所属机构无关。倡导维护网络安全人人有责，共同维护网络文明和谐。

联系方式：

学神 IT 教育官方网站: <http://www.xuegod.cn>

学神 K8S 精英学习 11 群 QQ 群: 957231097



学习顾问：小语老师

学习顾问：边边老师

学神微信公众号

微信扫码添加学习顾问微信，同时扫码关注学神公众号了解最新动态，获取更多学习资料及答疑就业服务！

第 5 章 在 k8s 平台部署电商项目-安装 Ingress、harbor、MySQL

本节所讲内容:

- 5.1 Ingress 和 Ingress Controller 概述
- 5.2 准备安装 harbor 需要的实验环境
- 5.3 安装 harbor
- 5.4 为 harbor 签发证书
- 5.5 安装 harbor
- 5.6 harbor 图形化界面使用说明
- 5.7 上传镜像到 harbor 仓库
- 5.8 从 harbor 仓库下载镜像
- 5.9 安装和配置数据存储仓库 MySQL

实验环境: k8s 集群: **k8s 的 master 节点**

ip: 192.168.1.63

主机名: xuegod63

配置: 6vCPU/6Gi 内存

k8s 的工作节点:

ip: 192.168.1.64

主机名: xuegod64

配置: 12vCPU/8Gi 内存

harbor 机器:

ip: 192.168.1.62

主机名: harbor

配置: 4vCPU/6Gi 内存

5.1 Ingress 和 Ingress Controller 概述

5.1.1 回顾 service 四层代理

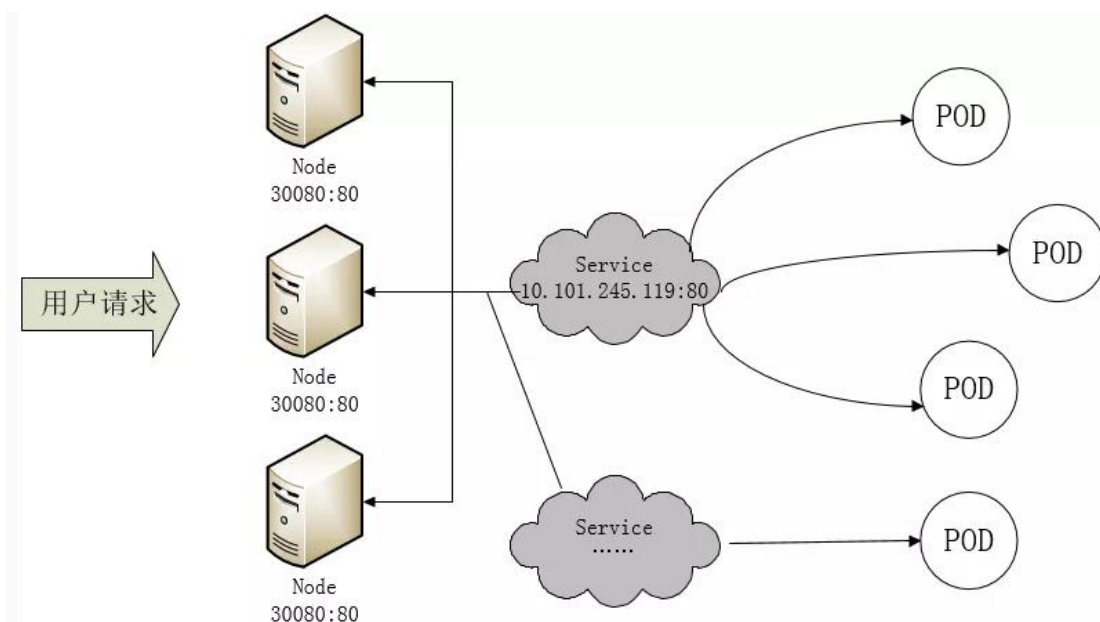
互动 1: 在 k8s 中为什么会有 service 这个概念?

1、Pod 漂移问题

Kubernetes 具有强大的副本控制能力, 能保证在任意副本 (Pod) 挂掉时自动从其他机器启动一个新的, 还可以动态扩容等, 通俗地说, 这个 Pod 可能在任何时刻出现在任何节点上, 也可能在任何时刻死在任何节点上; 那么自然随着 Pod 的创建和销毁, Pod IP 肯定会动态变化; 那么如何把这个动态的 Pod IP 暴露出去? 这里借助于 Kubernetes 的 Service 机制, Service 可以以标签的形式选定一组带有指定标签的 Pod, 并监控和自动负载他们的 Pod IP, 那么我们向外暴露只暴露 Service IP 就行了; 这就是 NodePort 模式: 即在每个节点上开起一个端口, 然后转发到内部 Pod IP 上:

此时的访问方式: `http://nodeip:nodeport/`, 即数据包流向如下:

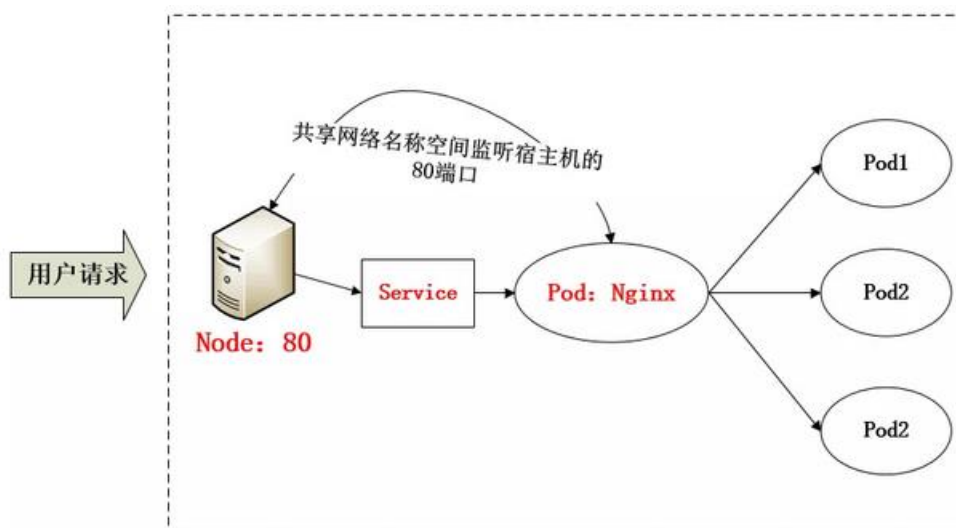
客户端请求-->node 节点的 ip:端口--->service 的 ip:端口--->pod 的 ip:端口



互动 2: Service 存在哪些问题?

1、端口管理问题

采用 NodePort 方式暴露服务面临的问题是，服务一旦多起来，NodePort 在每个节点上开启的端口会及其庞大，而且难以维护；这时，我们能否使用一个 Nginx 直接对内进行转发呢？众所周知的是，Pod 与 Pod 之间是可以互相通信的，而 Pod 是可以共享宿主机的网络名称空间的，也就是说当在共享网络名称空间时，Pod 上所监听的就是 Node 上的端口。那么这又该如何实现呢？简单的实现就是使用 DaemonSet 在每个 Node 上监听 80，然后写好规则，因为 Nginx 外面绑定了宿主机 80 端口（就像 NodePort），本身又在集群内，那么向后直接转发到相应 Service IP 就行了，如下图所示：



2、域名分配及动态更新问题

从上面的方法，采用 Nginx-Pod 似乎已经解决了问题，但是其实这里面有一个很大缺陷：当每次有新服务加入又该如何修改 Nginx 配置呢？我们知道使用 Nginx 可以通过虚拟主机域名进行区分不同的服务，而每个服务通过 upstream 进行定义不同的负载均衡池，再加上 location 进行负载均衡的反向代理，在日常使用中只需要修改 nginx.conf 即可实现，那在 K8S 中又该如何实现这种方式的调度呢？假设后端的初始服务只有 ecshop，后面增加了 bbs 和 member 服务，那么又该如何将这 2 个服务加

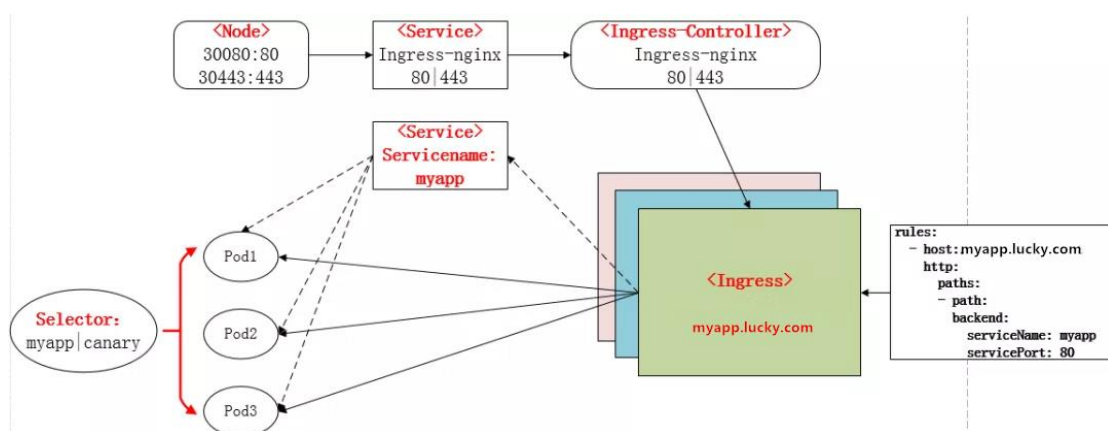
入到 Nginx-Pod 进行调度呢? 总不能每次手动改或者 Rolling Update 前端 Nginx Pod 吧! 此时 Ingress 出现了, 如果不算上面的 Nginx, Ingress 包含两大组件: Ingress Controller 和 Ingress。

5.1.2 Ingress 介绍

Ingress 官网定义: Ingress 可以把进入到集群内部的请求转发到集群中的一些服务上, 从而可以把服务映射到集群外部。Ingress 能把集群内 Service 配置成外网能够访问的 URL, 流量负载均衡, 提供基于域名访问的虚拟主机等。

Ingress 简单的理解就是你原来需要改 Nginx 配置, 然后配置各种域名对应哪个 Service, 现在把这个动作抽象出来, 变成一个 Ingress 对象, 你可以用 yaml 创建, 每次不要去改 Nginx 了, 直接改 yaml 然后创建/更新就行了; 那么问题来了: “Nginx 该怎么处理?”

Ingress Controller 这东西就是解决 “Nginx 的处理方式” 的; Ingress Controller 通过与 Kubernetes API 交互, 动态的去感知集群中 Ingress 规则变化, 然后读取他, 按照他自己模板生成一段 Nginx 配置, 再写到 Nginx Pod 里, 最后 reload 一下, 工作流程如下图:

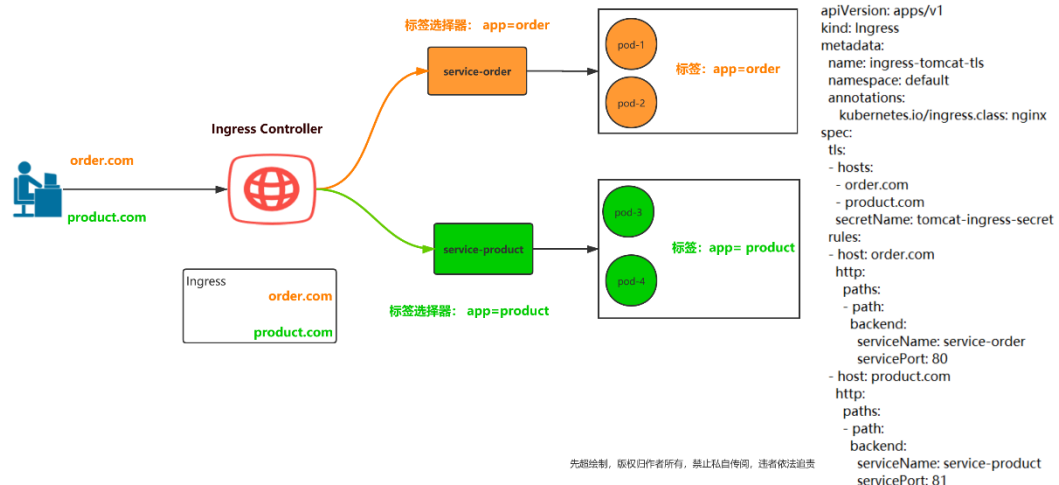


实际上 Ingress 也是 Kubernetes API 的标准资源类型之一, 它其实就是一组基于 DNS 名称 (host) 或 URL 路径把请求转发到指定的 Service 资源的规则。用于将集群外部的请求流量转发到集群内部完成的服务发布。我们需要明白的是, Ingress 资源自身不能进行“流量穿透”, 仅仅是一组规则的集合, 这些集合规则还需要其他功能的辅助, 比如监听某套接字, 然后根据这些规则的匹配进行路由转发, 这些能够为 Ingress 资源监听套接字并将流量转发的组件就是 Ingress Controller。

注: Ingress 控制器不同于 Deployment 控制器的是, Ingress 控制器不直接运行于 kube-controller-manager 的一部分, 它仅仅是 Kubernetes 集群的一个附件, 类似于 CoreDNS, 需要在集群上单独部署。

5.1.3 Ingress Controller 介绍

Ingress Controller 是一个七层负载均衡调度器, 客户端的请求先到达这个七层负载均衡调度器, 由七层负载均衡器在反向代理到后端 pod, 常见的七层负载均衡器有 nginx、traefik, 以我们熟悉的 nginx 为例, 假如请求到达 nginx, 会通过 upstream 反向代理到后端 pod 应用, 但是后端 pod 的 ip 地址是一直在变化的, 因此在后端 pod 前需要加一个 service, 这个 service 只是起到分组的作用, 那么我们 upstream 只需要填写 service 地址即可



5.1.4 Ingress 和 Ingress Controller 总结

Ingress Controller

Ingress Controller 可以理解为控制器，它通过不断的跟 Kubernetes API 交互，实时获取后端 Service、Pod 的变化，比如新增、删除等，结合 Ingress 定义的规则生成配置，然后动态更新上边的 Nginx 或者 trafik 负载均衡器，并刷新使配置生效，来达到服务自动发现的作用。

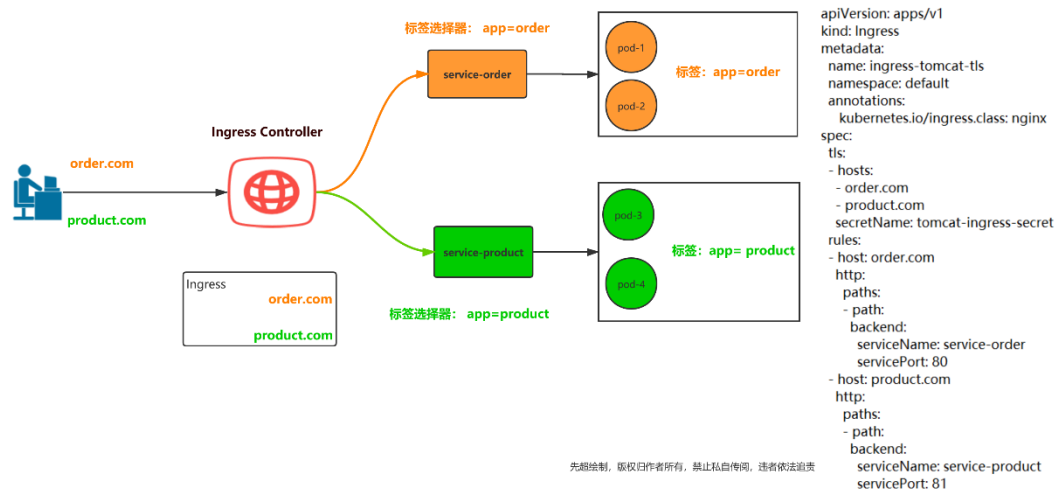
Ingress 则是定义规则，通过它定义某个域名的请求过来之后转发到集群中指定的 Service。它可以通过 Yaml 文件定义，可以给一个或多个 Service 定义一个或多个 Ingress 规则。

5.1.5 Ingress Controller 代理 k8s 内部应用的流程

- (1) 部署 Ingress controller，我们 ingress controller 使用的是 nginx
- (2) 创建 Service，用来分组 pod
- (3) 创建 Pod 应用，可以通过控制器创建 pod
- (4) 创建 Ingress http，测试通过 http 访问应用
- (5) 创建 Ingress https，测试通过 https 访问应用

客户端通过七层调度器访问后端 pod 的方式

使用七层负载均衡调度器 ingress controller 时，当客户端访问 kubernetes 集群内部的应用时，数据包走向如下图流程所示：



5.1.6 安装 Nginx Ingress Controller

安装需要的 yaml 文件和镜像在课件, 可自行下载

把 yaml 文件上传到 K8s 的 xuegod63 节点上, 把 defaultbackend.tar.gz 和 nginx-ingress-controller.tar.gz 镜像上传到 k8s 的 xuegod64 节点, 手动解压镜像:

```
[root@xuegod64 ~]# docker load -i nginx-ingress-controller.tar.gz
```

```
[root@xuegod64 ~]# docker load -i defaultbackend.tar.gz
```

更新 yaml 文件:

```
[root@xuegod63 ~]# mkdir /root/ingress
```

```
[root@xuegod63 ~]# cd /root/ingress/
```

```
[root@xuegod63 ingress]# kubectl apply -f nginx-ingress-controller-rbac.yaml
```

```
[root@xuegod63 ingress]# kubectl apply -f default-backend.yaml
```

```
[root@xuegod63 ingress]# kubectl apply -f nginx-ingress-controller.yaml
```

#验证 pod 是否部署成功:

```
[root@xuegod63 ingress]# kubectl get pods -n kube-system
```

显示如下, 说明部署成功了:

nginx-ingress-controller-74cf657846-qrvdm	1/1	Running	0	30s
default-http-backend-6b9b667c96-qw29n	1/1	Running	0	25s

注意:

default-backend.yaml 和 nginx-ingress-controller.yaml 文件指定了 **nodeName:**

xuegod64, 表示 default 和 nginx-ingress-controller 部署在 xuegod64 节点, 大家的 node 节点

如果主机名不是 xuegod64, 需要自行修改成自己的主机名, 这样才会调度成功, 一定要让 default-http-backend 和 nginx-ingress-controller 这两个 pod 在一个节点上。

5.1.7 测试 Ingress HTTP 代理 tomcat

#把 tomcat-8-5.tar.gz 上传到 xuegod64 机器, 手动解压:

```
[root@xuegod64 ~]# docker load -i tomcat-8-5.tar.gz
```

1.部署后端 tomcat 服务

```
[root@xuegod63 ~]# mkdir ingress-tomcat
```

```
[root@xuegod63 ~]# cd ingress-tomcat/
```

```
[root@xuegod63 ingress-tomcat]# cat deploy-demo.yaml
```

apiVersion: v1

kind: Service

metadata:

name: tomcat

namespace: default

spec:

selector:

app: tomcat

release: canary

ports:

- name: http

targetPort: 8080

port: 8080

- name: ajp


```
targetPort: 8009
port: 8009
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-deploy
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: tomcat
      release: canary
  template:
    metadata:
      labels:
        app: tomcat
        release: canary
    spec:
      containers:
        - name: tomcat
          image: tomcat:8.5.34-jre8-alpine
          ports:
            - name: http
              containerPort: 8080
              name: ajp
              containerPort: 8009
```

更新 yaml 文件:

```
[root@xuegod63 ingress-tomcat]# kubectl apply -f deploy-demo.yaml
```

查看 pod 是否部署成功:

```
[root@xuegod63 ingress-tomcat]# kubectl get pods
```

显示如下, 说明 pod 创建成功:

tomcat-deploy-66b67fcf7b-h582w	1/1	Running	0	50s
tomcat-deploy-66b67fcf7b-rm82h	1/1	Running	0	50s

2、部署 ingress

(1) 编写 ingress 的配置清单

```
[root@xuegod63 ingress-tomcat]# cat ingress-myapp.yaml
```

```
apiVersion: extensions/v1beta1      #api 版本
kind: Ingress                        #清单类型
metadata:                            #元数据
  name: ingress-myapp                #ingress 的名称
  namespace: default                 #所属名称空间
  annotations:                       #注解信息
```

```
kubernetes.io/ingress.class: "nginx"
spec:      #规格
  rules:    #定义后端转发的规则
  - host: tomcat.lucky.com    #通过域名进行转发
    http:
      paths:
      - path:    #配置访问路径, 如果通过 url 进行转发, 需要修改; 空默认为访问的路径为 "/"
        backend: #配置后端服务
          serviceName: tomcat
          servicePort: 8080
```

更新 yaml 文件:

```
[root@xuegod63 ingress-tomcat]# kubectl apply -f ingress-myapp.yaml
```

查看 ingress-myapp 的详细信息

```
[root@xuegod63 ingress-tomcat]# kubectl describe ingress ingress-myapp
```

Name: ingress-myapp

Namespace: default

Address:

Default backend: default-http-backend:80 (10.244.209.134:8080)

Rules:

Host	Path	Backends
------	------	----------

tomcat.lucky.com		
------------------	--	--

		tomcat:8080 (10.244.209.135:8080,10.244.209.136:8080)
--	--	---

Annotations: kubernetes.io/ingress.class: nginx

修改电脑本地的 host 文件, 增加如下一行, 下面的 ip 是 k8s 的 xuegod64 节点 ip


192.168.1.64 tomcat.lucky.com

浏览器访问 tomcat.lucky.com, 出现如下:

Apache Tomcat/8.5.34



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:
[Security Considerations HOW-TO](#)
[Manager Application HOW-TO](#)
[Clustering/Session Replication HOW-TO](#)

[Server Status](#)
[Manager App](#)
[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#) [Realms & AAA](#) [Examples](#) [Servlet Specifications](#)
[First Web Application](#) [JDBC DataSources](#) [Tomcat Versions](#)

Managing Tomcat

For security, access to the `manager webapp` is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 8.5 access to the manager application is split between different users.
[Read more...](#)

[Release Notes](#)
[Changelog](#)
[Migration Guide](#)
[Security Notices](#)

Documentation

[Tomcat 8.5 Documentation](#)
[Tomcat 8.5 Configuration](#)
[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

[Tomcat 8.5 Bug Database](#)
[Tomcat 8.5 JavaDocs](#)
[Tomcat 8.5 SVN Repository](#)

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)
User support and discussion

[taglibs-user](#)
User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)
Development mailing list, including commit messages

把上面资源清单文件删除，防止干扰后面的实验

```
[root@xuegod63 ingress-tomcat]# kubectl delete -f deploy-demo.yaml
```

```
[root@xuegod63 ingress-tomcat]# kubectl delete -f ingress-myapp.yaml
```

5.2 准备安装 harbor 需要的实验环境

新创建一台虚拟机安装 harbor，配置如下：

主机名	ip	配置	网络
harbor	192.168.1.62	4vCPU/6G 内存/60G 硬盘	NAT

注：新机器的初始化只需要按照上面课程步骤进行初始化即可

5.2.1 harbor 简介

harbor 是私有镜像仓库，用来存储和分发镜像的

docker 还有一个官方的镜像仓库 docker hub，免费用户只能简单的使用，创建一个私有镜像仓库，存储镜像，付费用户才可以拥有更多权限，默认 docker pull 拉取镜像就是从 docker hub 上拉取，速度极慢，不利于生产环境使用。

harbor 私有镜像仓库拉取镜像速度极快，属于内网传输，功能也很强大。

互动：有什么功能？

1.基于角色访问控制：用户与 Docker 镜像仓库通过“项目”进行组织管理，一个用户可以对多个镜像仓库在同一命名空间（project）里有不同的权限。

2.镜像复制：镜像可以在多个 Registry 实例中复制（同步）。尤其适合于负载均衡，高可用，混合云和多云的场景。

3.图形化界面：用户可以通过浏览器来浏览，检索当前 Docker 镜像仓库，管理项目和命名空间。

4.部署简单: 提供在线和离线两种安装工具

5.LDAP: Harbor 可以集成企业内部已有的 AD/LDAP, 用于鉴权认证管理

5.2.2 配置静态 IP

把虚拟机或者物理机配置成静态 ip 地址, 这样机器重新启动后 ip 地址也不会发生改变。

在 harbor 节点配置网络

修改/etc/sysconfig/network-scripts/ifcfg-ens33 文件, 变成如下:

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=192.168.1.62
NETMASK=255.255.255.0
GATEWAY=192.168.42.2
DNS1=192.168.42.2
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
DEVICE=ens33
ONBOOT=yes
```

修改配置文件之后需要重启网络服务才能使配置生效, 重启网络服务命令如下:

```
[root@xuegod62 ~]# systemctl restart network
```

5.2.3 修改 yum 源

下面的步骤在 harbor 节点操作

```
[root@xuegod62 ~]# mv /etc/yum.repos.d/CentOS-Base.repo
/etc/yum.repos.d/CentOS-Base.repo.backup
```

下载阿里的 yum 源

```
[root@xuegod62 ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
```

配置安装 k8s 需要的 yum 源

```
[root@xuegod62 ~]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
EOF
#添加 docker 需要的 yum 源
```

```
[root@xuegod62 ~]# yum-config-manager --add-repo
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
清理 yum 缓存
[root@xuegod62 ~]# yum clean all
生成新的 yum 缓存
[root@xuegod62 ~]# yum makecache fast
更新 yum 源
[root@xuegod62 ~]# yum -y update
安装软件包
[root@xuegod62 ~]# yum -y install wget net-tools nfs-utils lrzsz gcc gcc-c++ make
cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel wget
vim ncurses-devel autoconf automake zlib-devel python-devel epel-release openssh-
server socat ipvsadm conntrack ntpdate yum-utils device-mapper-persistent-data lvm2
telnet
```

5.2.3 配置防火墙

关闭 firewalld 防火墙, centos7 系统默认使用的是 firewalld 防火墙, 停止 firewalld 防火墙, 并禁用这个服务。

```
[root@xuegod62 ~]# systemctl stop firewalld && systemctl disable firewalld
```

5.2.4 时间同步

```
[root@xuegod62 ~]# ntpdate cn.pool.ntp.org
```

编辑计划任务, 每小时做一次同步

```
1) [root@xuegod62 ~]# crontab -e
```

```
**/*1 * * * /usr/sbin/ntpdate cn.pool.ntp.org
```

2) 重启 crond 服务:

```
[root@xuegod62 ~]# systemctl restart crond
```

5.2.5 关闭 selinux

关闭 selinux, 设置永久关闭, 这样重启机器 selinux 也处于关闭状态

可用下面方式修改:

```
[root@xuegod62 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/'
```

/etc/sysconfig/selinux

```
[root@xuegod62 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
```

/etc/selinux/config

上面文件修改之后, 需要重启虚拟机, 如果测试环境可以用如下命令强制重启:

```
[root@xuegod62 ~]# reboot -f
```

注: 生产环境不要 reboot -f, 要正常关机重启

查看 selinux 是否修改成功

重启之后登录到机器上用如下命令:

```
[root@xuegod62 ~]# getenforce
```

显示 Disabled 说明 selinux 已经处于关闭状态

5.2.6 修改内核参数

```
[root@xuegod62 ~]# cat <<EOF >/etc/sysctl.d/k8s.conf
```

```
net.bridge.bridge-nf-call-ip6tables = 1
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
EOF
```

```
[root@xuegod62 ~]# systemctl --system
```

注: `systemctl --system` 这个会加载所有的 `systemd` 配置

5.2.7 修改主机名

在 192.168.1.62 上:

```
[root@xuegod62 ~]# hostnamectl set-hostname harbor
```

5.2.8 配置 hosts 文件

xuegod63、xuegod64、harbor 主机的 hosts 文件保持一致, 可按如下方法修改:

在 `/etc/hosts` 文件增加如下几行:

```
192.168.1.63 xuegod63
```

```
192.168.1.64 xuegod64
```

```
192.168.1.62 harbor
```

5.2.9 配置主机之间无密码登陆

配置 xuegod63 到 harbor 无密码登陆

在 xuegod63 上操作

```
[root@xuegod63 ~]# cd /root && ssh-copy-id -i .ssh/id_rsa.pub root@harbor
```

#上面需要输入 yes 之后, 输入密码, 输入 harbor 物理机密码即可

5.3 安装 docker

在 harbor 机器继续操作下面步骤

5.3.1 查看 docker 版本

```
[root@harbor ~]# yum list docker-ce --showduplicates |sort -r
```

5.3.2 安装 docker

```
[root@harbor ~]# yum install -y docker-ce-19.03.7-3.el7
```

```
[root@harbor ~]# systemctl enable docker && systemctl start docker
```

#查看 docker 状态, 如果状态是 active (running), 说明 docker 是正常运行状态

```
[root@harbor ~]# systemctl status docker
```

5.3.3 修改 docker 配置文件

```
[root@harbor ~]# cat > /etc/docker/daemon.json <<EOF
```

```
{  
  "registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-  
cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hu  
b-mirror.c.163.com","http://qtid6917.mirror.aliyuncs.com"]  
}
```

```
EOF
```

注:

```
"registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-  
cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hub  
-mirror.c.163.com","http://qtid6917.mirror.aliyuncs.com"]
```

上面配置的是 docker 镜像加速器

5.3.4 重启 docker 使配置生效

```
[root@harbor ~]# systemctl daemon-reload && systemctl restart docker && systemctl  
status docker
```

5.3.5 开启机器的 bridge 模式

#临时生效

```
[root@harbor ~]# echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
```

```
[root@harbor ~]# echo 1 > /proc/sys/net/bridge/bridge-nf-call-ip6tables
```

#永久生效

```
[root@harbor ~]# echo ""
```

```
vm.swappiness = 0
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
net.ipv4.ip_forward = 1
```

```
net.bridge.bridge-nf-call-ip6tables = 1
```

```
"" > /etc/sysctl.conf
```

```
[root@harbor ~]# sysctl -p
```

5.4 为 harbor 签发证书

```
[root@harbor ~]# mkdir /data/ssl -p
```

```
[root@harbor ~]# cd /data/ssl/
```

生成 ca 证书:

```
[root@harbor ~]# openssl genrsa -out ca.key 3072
```

#生成一个 3072 位的 key, 也就是私钥

```
[root@harbor ~]# openssl req -new -x509 -days 3650 -key ca.key -out ca.pem
```

#生成一个数字证书 ca.pem, 3650 表示证书的有效时间是 3 年, 按箭头提示填写即可, 没有箭头标注的为空:

```
[root@harbor ssl]# openssl req -new -x509 -days 3650 -key ca.key -out ca.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CH
State or Province Name (full name) []:BJ
Locality Name (eg, city) [Default City]:BJ
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
```

生成域名的证书:

```
[root@harbor ~]# openssl genrsa -out harbor.key 3072
```

#生成一个 3072 位的 key, 也就是私钥

```
[root@harbor ~]# openssl req -new -key harbor.key -out harbor.csr
```

#生成一个证书请求, 一会签发证书时需要的, 标箭头的按提示填写, 没有箭头标注的为空:

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CH
State or Province Name (full name) []:BJ
Locality Name (eg, city) [Default City]:BJ
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:harbor
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

主机名

签发证书:

```
[root@harbor ~]# openssl x509 -req -in harbor.csr -CA ca.pem -CAkey ca.key -
CAcreateserial -out harbor.pem -days 3650
```

显示如下, 说明证书签发好了:

```
Signature ok
subject=/C=CH/ST=BJ/L=BJ/O=Default Company Ltd/CN=harbor
Getting CA Private Key
```

查看证书是否有效:

```
[root@harbor ~]# openssl x509 -noout -text -in harbor.pem
```

显示如下, 说明有效:

```
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number:
      cd:21:3c:44:64:17:65:40
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=CH, ST=BJ, L=BJ, O=Default Company Ltd
    Validity
      Not Before: Dec 26 09:29:19 2020 GMT
      Not After : Dec 24 09:29:19 2030 GMT
    Subject: C=CH, ST=BJ, L=BJ, O=Default Company Ltd, CN=harbor
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (3072 bit)
      Modulus:
        00:b0:60:c3:e6:35:70:11:c8:73:83:38:9a:7e:b8:
        4b:f6:b7:53:ae:ab:30:00:48:da:4b:43:15:8b:41:
        e5:c4:ba:4b:50:c1:89:9e:71:cc:1b:1b:bb:6c:ed:
        ec:6c:22:19:96:9b:b8:be:60:5a:ff:50:13:0a:bd:
        8c:4a:63:b9:c5:d5:24:f5:21:72:68:9f:39:d9:38:
        18:c7:dc:a7:83:ea:67:81:b8:92:d3:fd:31:99:bb:
```

a6:0f:47:74:68:ff:90:a0:4c:56:85:32:95:5b:c9:
e6:53:3d:46:c9:2a:7f:fb:1d:a6:b6:b3:cc:ad:3b:
4b:bc:42:3a:13:7d:a8:5e:c8:ab:96:fd:06:50:4d:
26:2b:b2:6b:34:b5:f4:d1:9b:49:36:c7:6d:0f:fd:
a4:17:92:4d:96:80:f8:2f:b5:60:fd:2b:01:27:54:
80:60:09:40:bf:c0:56:a6:d1:2e:09:23:4c:84:0e:
d9:bb:f4:39:5f:5d:e7:ce:b3:22:51:f8:53:ad:95:
fe:52:8c:b8:87:e3:67:be:02:3c:18:73:80:23:7e:
37:2a:2c:de:9d:1c:b4:ab:b4:79:90:09:d9:3f:f5:
ac:b2:08:39:28:56:1e:20:96:e5:35:34:b3:c9:ae:
ef:eb:d4:68:14:7d:ac:cb:ec:a6:61:7c:27:fb:87:
fa:4f:a8:e1:c9:b0:22:b3:7a:c5:f3:ec:58:97:61:
de:b5:fb:b5:89:a5:91:76:04:eb:c7:ee:db:62:97:
14:50:04:0d:6d:6c:fb:1b:52:b5:0b:04:bb:e8:4c:
d8:c5:19:c9:f1:3f:5e:bf:b0:3b:33:28:72:47:35:
c3:56:98:32:4f:0b:e3:0c:4e:d5:31:cb:e4:1a:37:
a5:ec:ef:97:a5:da:cc:72:3c:0a:e2:0d:a8:e7:b1:
fa:aa:d9:cb:54:b6:d2:85:09:85:a2:1b:93:73:27:
86:41:76:e2:cf:d3:77:91:d9:b2:19:08:53:f8:0c:
dd:51:b1:cf:2a:64:02:35:c6:11

Exponent: 65537 (0x10001)

Signature Algorithm: sha256WithRSAEncryption

e6:3d:15:29:a8:a4:bf:79:c7:bd:65:82:49:99:d0:71:5f:60:
0f:e0:2d:4f:f1:1c:22:73:4f:29:b6:cb:7e:29:e2:6c:bc:b3:
9e:e4:b0:20:52:24:c5:e5:33:9f:dc:38:29:55:91:3d:84:d5:
80:f3:7a:78:86:ce:80:5d:8b:70:ac:d6:84:18:3a:ad:b0:06:
c3:49:ba:24:13:75:cd:37:91:81:6d:80:b0:20:91:f6:a8:0c:
d1:77:3d:8f:2e:e8:66:ba:29:1d:d7:6e:ec:81:15:75:70:dd:
7b:9e:55:85:6e:d9:89:04:c8:5e:db:bf:e2:f7:ec:93:03:8a:
73:34:5a:fb:23:03:1d:d3:26:aa:80:1a:a9:65:21:61:f6:16:
38:14:fe:3a:43:a2:46:60:be:32:b7:b1:aa:47:44:40:01:ca:
b9:13:10:03:fe:c1:95:6d:aa:71:57:73:ba:76:30:f0:b2:53:
0f:7e:db:6b:ed:03:d9:8f:64:07:5e:e9:3a:b1:6f:46:d3:67:
2c:ea:c7:aa:02:de:48:b8:e3:d3:88:ad:52:24:d5:c2:e4:c0:
c5:0a:fe:eb:ea:8e:a4:8b:f8:d7:36:15:ff:0b:c7:41:79:7c:
7e:16:75:61:52:fa:b4:29:fb:6b:07:c2:1d:cd:4d:4e:6c:29:
b4:ef:9a:ed:86:c6:14:75:a6:41:fa:bb:11:d4:d7:00:82:46:
58:8b:15:54:76:63:84:94:ef:9f:41:a7:c9:4f:63:00:3a:36:
f0:27:14:4c:16:1b:22:6e:1e:db:bc:83:bb:e5:ab:1d:5c:88:
9e:b6:4d:f0:b3:10:73:84:c9:6a:eb:4d:f3:f0:cc:a5:80:3f:
e5:b4:ba:91:ef:c8:8f:be:45:eb:d8:ad:3f:18:8e:dd:f7:c0:
d0:92:80:21:83:c6:86:e6:c0:af:42:09:49:f4:ce:7b:a2:35:
5e:3e:a4:5b:e4:9c:49:52:9a:ee:b5:fb:fb:b6:d5:e3:c7:0c:
50:a6:65:54:28:92

5.5 安装 harbor

创建安装目录

```
[root@harbor ~]# mkdir /data/install -p
```

```
[root@harbor ~]# cd /data/install/
```

安装 harbor

/data/ssl 目录下有如下文件:

```
[root@harbor ~]# ls
```

```
ca.key  ca.pem  ca.srl  harbor.csr  harbor.key  harbor.pem
```

```
[root@harbor ~]# cd /data/install/
```

#把 harbor 的离线包 harbor-offline-installer-v1.5.0.tgz 上传到这个目录, 离线包在课件里提供了, 可自行下载:

解压:

```
[root@harbor ~]# tar zxvf harbor-offline-installer-v1.5.0.tgz
```

```
[root@harbor ~]# cd harbor
```

#common 目录: 存放模板配置

#ha 目录: 做 harbor 高可用的

修改配置文件:

```
[root@harbor harbor]# vim harbor.cfg
```

```
hostname = harbor
```

#修改 hostname, 跟上面签发的证书域名保持一致

```
ui_url_protocol = https
```

#协议用 https

```
ssl_cert = /data/ssl/harbor.pem
```

```
ssl_cert_key = /data/ssl/harbor.key
```

邮件和 ldap 不需要配置, 在 harbor 的 web 界面可以配置

其他配置采用默认即可

harbor 默认的账号密码: admin/Harbor12345

安装 docker-compose

```
[root@harbor ~]# yum install docker-compose -y
```

#按装 harbor

#在执行下面命令之前, 需要先把 docker-harbor.tar.gz 上传到 harbor 机器, 手动解压, 解压出来的镜像就是安装 harbor 需要的镜像:

```
[root@harbor ~]# docker load -i docker-harbor.tar.gz
```

```
[root@harbor ~]# cd /data/install/harbor
```

```
[root@harbor harbor]# ./install.sh --with-notary --with-clair
```

#clair 开启镜像的漏洞扫描

[Step 0]: checking installation environment ...

Note: docker version: 18.09.0

Note: docker-compose version: 1.18.0

[Step 1]: loading Harbor images ...

```
651f69aef02c: Loading layer [=====>] 135.8MB/135.8MB
40a1aad64343: Loading layer [=====>] 23.24MB/23.24MB
3fe2713e4072: Loading layer [=====>] 12.16MB/12.16MB
ba3a1eb0e375: Loading layer [=====>] 17.3MB/17.3MB
447427ec5e1a: Loading layer [=====>] 15.87kB/15.87kB
4ccb4026663c: Loading layer [=====>] 3.072kB/3.072kB
16faa95946a1: Loading layer [=====>] 29.46MB/29.46MB
Loaded image: vmware/notary-server-photon:v0.5.1-v1.4.0
fa7ba9fd42c9: Loading layer [=====>] 10.95MB/10.95MB
4e400f9ae23e: Loading layer [=====>] 17.3MB/17.3MB
2802fb27c88b: Loading layer [=====>] 15.87kB/15.87kB
e6367a4e1e1e: Loading layer [=====>] 3.072kB/3.072kB
8ece8dfcdd98: Loading layer [=====>] 28.24MB/28.24MB
Loaded image: vmware/notary-signer-photon:v0.5.1-v1.4.0
a7dd1a8afcaf: Loading layer [=====>] 12.26MB/396.7MB
```

Creating nginx ...

Creating notary-server ...

□ ----Harbor has been installed and started successfully.----

Now you should be able to visit the admin portal at <https://img.rulin.me>.
For more details, please visit <https://github.com/vmware/harbor> .

安装过程会出现上面的界面, 说明安装正常, docker ps 显示如下, 说明容器启动正常

```
[root@localhost harbor]# docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS              PORTS
32b0a2f3e926       vmware/notary-server-photon:v0.5.1-v1.4.0  "/bin/server-start.sh"  About a minute ago  Up About a minute  8080/tcp
d56549318c4c       vmware/harbor-jobservice:v1.4.0          "/harbor/start.sh"      About a minute ago  Up About a minute  8080/tcp
1d417a16578b       vmware/nginx-photon:v1.4.0              "nginx -g 'daemon of..." About a minute ago  Up About a minute  0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
8f7ca3675f97       vmware/notary-signer-photon:v0.5.1-v1.4.0  "/bin/signer-start.sh"  About a minute ago  Up About a minute  8080/tcp
7c610e120463       vmware/clair-photon:v2.0.1-v1.4.0        "/docker-entrypoint..." About a minute ago  Up 53 seconds      6060-6061/tcp
0db8a3d75f1f       vmware/harbor-ui:v1.4.0                  "/harbor/start.sh"      About a minute ago  Up About a minute  8080/tcp
bf0ea5da19a3       vmware/mariadb-photon:v1.4.0             "/usr/local/bin/dock..." About a minute ago  Up About a minute  3306/tcp
6aa875f91882       vmware/registry-photon:v2.6.2-v1.4.0     "/entrypoint.sh serv..." About a minute ago  Up About a minute  5000/tcp
ab912c4ac596       vmware/harbor-db:v1.4.0                  "/usr/local/bin/dock..." About a minute ago  Up About a minute  3306/tcp
746c09783df2       vmware/postgresql-photon:v1.4.0          "/entrypoint.sh post..." About a minute ago  Up About a minute  5432/tcp
e8abc52bf7a2       vmware/harbor-adminserver:v1.4.0         "/harbor/start.sh"      About a minute ago  Up About a minute  8080/tcp
c19e91f3e7c3       vmware/harbor-log:v1.4.0                 "/bin/sh -c /usr/loc..." About a minute ago  Up About a minute  127.0.0.1:1514->10514/tcp
```

在自己电脑修改 hosts 文件, 添加如下一行, 然后保存即可

192.168.1.62 harbor

注意:

如何停掉 harbor:

```
[root@harbor ~]# docker-compose stop
```

如何启动 harbor:

```
[root@harbor ~]# docker-compose start
```

5.6 harbor 图像化界面使用说明

在浏览器输入:

<https://harbor/>



出现如下界面, 说明登陆正常



账号: admin

密码: Harbor12345

输入账号密码出现如下:



所有基础镜像都会放在 library 里面, 这是一个公开的镜像仓库

新建项目->起个项目名字 test (把访问级别公开那个选中, 让项目才可以被公开使用)



在 harbor 机器上修改 docker 配置, 修改之后的配置如下

```
[root@harbor ~]# cat > /etc/docker/daemon.json <<EOF
{
  "registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-
    cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hu
    b-mirror.c.163.com","http://qtid6917.mirror.aliyuncs.com"],
  "insecure-registries":["192.168.1.62"]
}
EOF
```

修改配置之后使配置生效:

```
[root@harbor ~]# systemctl daemon-reload && systemctl restart docker
```

注意:

配置新增加了一行内容如下:

```
"insecure-registries":["192.168.1.62"],
```

上面增加的内容表示我们内网访问 harbor 的时候走的是 http, 192.168.1.62 是安装 harbor 机器的 ip

登录 harbor:

```
docker login 192.168.1.62
```

Username: admin

Password: Harbor12345

输入账号密码之后看到如下, 说明登录成功了:

Login Succeeded

5.7 上传镜像到 harbor 仓库

在 harbor 机器上操作:

把 busybox.tar.gz 压缩包上传到 harbor 机器, 手动解压:

```
[root@harbor ~]# docker load -i busybox.tar.gz
```

```
[root@harbor ~]# docker tag busybox:latest 192.168.1.62/test/busybox:v1
```

```
[root@harbor ~]# docker push 192.168.1.62/test/busybox:v1
```

执行上面命令就会把 192.168.1.62/test/tomcat:v1 上传到 harbor 里的 test 项目下



5.8 从 harbor 仓库下载镜像

在 harbor 机器上删除镜像

```
[root@harbor ~]# docker rmi -f 192.168.1.62/test/busybox:v1
```

拉取镜像

```
[root@harbor ~]# docker pull 192.168.1.62/test/busybox:v1
```

5.9 安装和配置数据存储仓库 MySQL

5.9.1 MySQL 简介

MySQL 是一款安全、跨平台、高效的, 并与 PHP、Java 等主流编程语言紧密结合的数据库系统。该数据库系统是由瑞典的 MySQL AB 公司开发、发布并支持, 由 MySQL 的初始开发人员 David Axmark 和 Michael Monty Widenius 于 1995 年建立的。MySQL 的象征符号是一只名为 Sakila 的海豚, 代表着 MySQL 数据库的速度、能力、精确和优秀本质。

MySQL logo:



目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低,尤其是开放源码这一特点,使得很多公司都采用 MySQL 数据库以降低成本。

MySQL 数据库可以称得上是目前运行速度最快的 SQL 语言数据库之一。除了具有许多其他数据库所不具备的功能外,MySQL 数据库还是一种完全免费的产品,用户可以直接通过网络下载 MySQL 数据库,而不必支付任何费用。

5.9.2 MySQL 特点

1) 功能强大

MySQL 中提供了多种数据库存储引擎,各引擎各有所长,适用于不同的应用场合,用户可以选择最合适的引擎以得到最高性能,可以处理每天访问量超过数亿的高强度的搜索 Web 站点。MySQL5 支持事务、视图、存储过程、触发器等。

2) 支持跨平台

MySQL 支持至少 20 种以上的开发平台,包括 Linux、Windows、FreeBSD、IBMAIX、AIX、FreeBSD 等。这使得在任何平台下编写的程序都可以进行移植,而不需要对程序做任何修改。

3) 运行速度快

高速是 MySQL 的显著特性。在 MySQL 中,使用了极快的 B 树磁盘表(MyISAM)和索引压缩;通过使用优化的单扫描多连接,能够极快地实现连接;SQL 函数使用高度优化的类库实现,运行速度极快。

4) 支持面向对象

PHP 支持混合编程方式。编程方式可分为纯粹面向对象、纯粹面向过程、面句对象与面向过程混合 3 种方式。

5) 安全性高

灵活和安全的权限与密码系统,允许基本主机的验证。连接到服务器时,所有的密码传输均采用加密形式,从而保证了密码的安全。

6) 成本低

MySQL 数据库是一种完全免费的产品,用户可以直接通过网络下载。

7) 支持各种开发语言

MySQL 为各种流行的程序设计语言提供支持,为它们提供了很多的 API 函数,包括 PHP、ASP.NET、Java、Eiffel、Python、Ruby、Tcl、C、C++、Perl 语言等。

8) 数据库存储容量大

MySQL 数据库的最大有效表尺寸通常是由操作系统对文件大小的限制决定的,而不是由 MySQL 内部限制决定的。InnoDB 存储引擎将 InnoDB 表保存在一个表空间内,该表空间可由数个文件创建,表空间的最大容量为 64TB,可以轻松处理拥有上千万条记录的大型数据库。

9) 支持强大的内置函数

PHP 中提供了大量内置函数,几乎涵盖了 Web 应用开发中的所有功能。它内置了数据库连接、文件上传等功能,MySQL 支持大量的扩展库,如 MySQLi 等,可以为快速开发 Web 应用提供便利。

5.9.3 安装 MySQL

用 xuegod63 机器即可:

在 xuegod63 上操作:

把 mysql-community-release-el7-5.noarch.rpm 上传到 xuegod63 上:

```
[root@xuegod63 ~]# rpm -ivh mysql-community-release-el7-5.noarch.rpm
```

```
[root@xuegod63 ~]# yum install mysql-server -y
```

权限设置

```
[root@xuegod63 ~]# chown mysql:mysql -R /var/lib/mysql
```

初始化 MySQL

```
[root@xuegod63 ~]# mysqld --initialize
```

如有下图报错:

```
2021-06-08 10:36:03 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_timestamp server option (see documentation for more details).
2021-06-08 10:36:03 0 [Note] mysqld (mysqld 5.6.51) starting as process 84113 ...
2021-06-08 10:36:03 84113 [ERROR] Fatal error: Please read "Security" section of the manual to find out how to run mysqld as root!
2021-06-08 10:36:03 84113 [ERROR] Aborting
2021-06-08 10:36:03 84113 [Note] Binlog end
2021-06-08 10:36:03 84113 [Note] mysqld: Shutdown complete
```

解决:

```
[root@xuegod63 ~]# mysqld --initialize --explicit_defaults_for_timestamp --user=root
```

启动 MySQL

```
[root@xuegod63 ~]# systemctl start mysqld
```

查看 MySQL 运行状态

```
[root@xuegod63 ~]# systemctl status mysqld
```

mysql 安装成功后, 默认的 root 用户密码为空, 你可以使用以下命令来创建 root 用户的密码, 密码设置成 111111

```
[root@xuegod63 ~]# mysqladmin -u root password "111111"
```

登陆数据库

```
[root@xuegod63 ~]# mysql -uroot -p111111
```

创建数据库 tb_order、tb_product、tb_stock

```
mysql> create database tb_product;
```

```
mysql> create database tb_stock;
```

```
mysql> create database tb_order;
```

5.9.4 在 Mysql 数据库导入数据

把相应的 sql 语句上传到 mysql 机器的 root 目录下, sql 文件分别是 order.sql、product.sql、stock.sql, 这些 sql 文件在课件里大家可以搜索, 按如下方法导入:

```
mysql> use tb_order
```

```
mysql> source /root/order.sql
```

```
mysql> use tb_stock
```

```
mysql> source /root/stock.sql
```

```
mysql> use tb_product
```

```
mysql> source /root/product.sql
```


5.9.5 对 MySQL 数据库授权

```
mysql> grant all on *.* to 'root'@'10.244.%.%' identified by '111111';
mysql> grant all on *.* to 'root'@'192.168.%.%' identified by '111111';
mysql> grant all on *.* to 'root'@'%' identified by '111111';
mysql> flush privileges;
mysql> exit
```

5.9.6 验证数据库导入数据是否正确

```
mysql> use tb_stock
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> select
```

Display all 754 possibilities? (y or n)

```
mysql> select * from stock
```

```
-> ;
```

```
+----+-----+-----+-----+
| id | prod_id | sales_stock | real_stock |
+----+-----+-----+-----+
|  1 |      1 |          99 |          99 |
|  2 |      2 |          88 |          88 |
|  3 |      3 |          77 |          77 |
|  4 |      4 |          66 |          66 |
+----+-----+-----+-----+
```

4 rows in set (0.01 sec)

```
mysql> use tb_product
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> select * from product;
```

```
+----+-----+-----+
| id | product_name | price |
+----+-----+-----+
|  1 | 手机         | 99.990 |
|  2 | 大彩电       | 999.000 |
|  3 | 洗衣机       | 100.000 |
|  4 | 超级大冰箱   | 9999.000 |
+----+-----+-----+
```

4 rows in set (0.01 sec)

```
mysql> use tb_order
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

```
mysql> select * from orders;
```

Empty set (0.00 sec)

总结:

- 5.1 Ingress 和 Ingress Controller 概述
- 5.2 准备安装 harbor 需要的实验环境
- 5.3 安装 harbor
- 5.4 为 harbor 签发证书
- 5.5 安装 harbor
- 5.6 harbor 图形化界面使用说明
- 5.7 上传镜像到 harbor 仓库
- 5.8 从 harbor 仓库下载镜像
- 5.9 安装和配置数据存储仓库 MySQL