## 任务8：Identtityserver4登录中心

项目(IdentityServerSample)



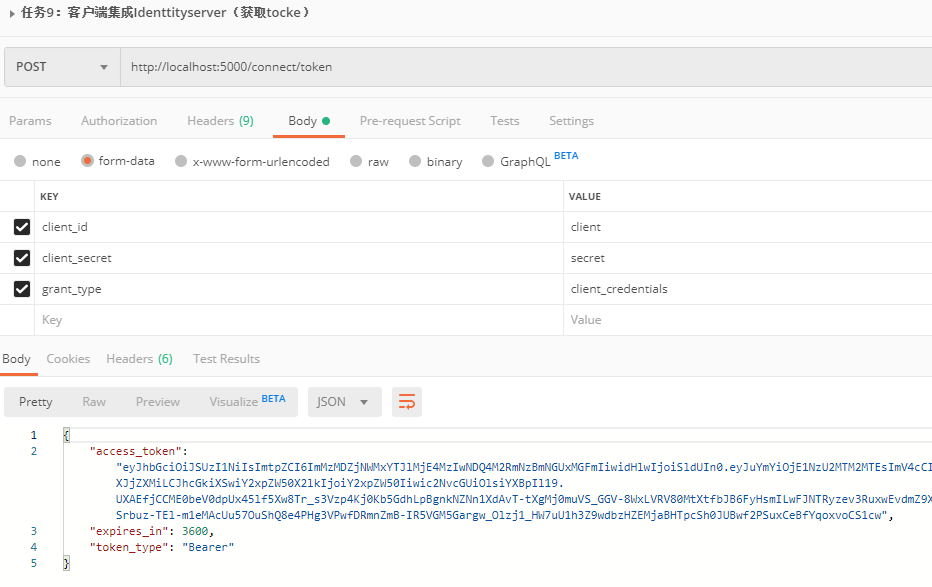
## 任务9：客户端集成Identtityserver

项目（ClientCredentialApi）

先在服务端拿到tocke，再去请求客户端

通过地址<http://localhost:5000/.well-known/openid-configuration>得到获取tocke的请求地址

需要开启服务端先拿到tocke



## 任务10：第三方ClientCredential模式调用

项目（ThirdPartyDemo）

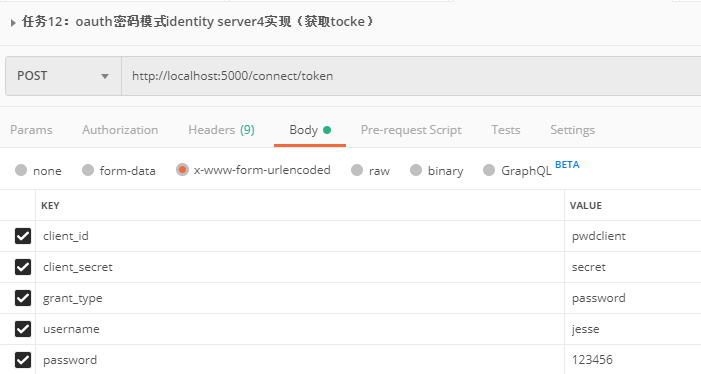
第三方应用去请求api，先要去服务端拿到tocke，再去请求客户端api，需要带上tocke

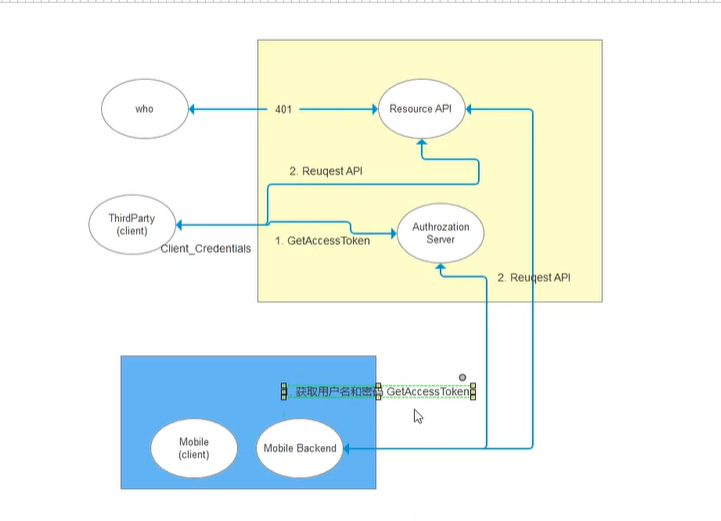
## 任务11：ClientCredential模式总结

去访问api，先要去Identtityserver4中心拿到tocke

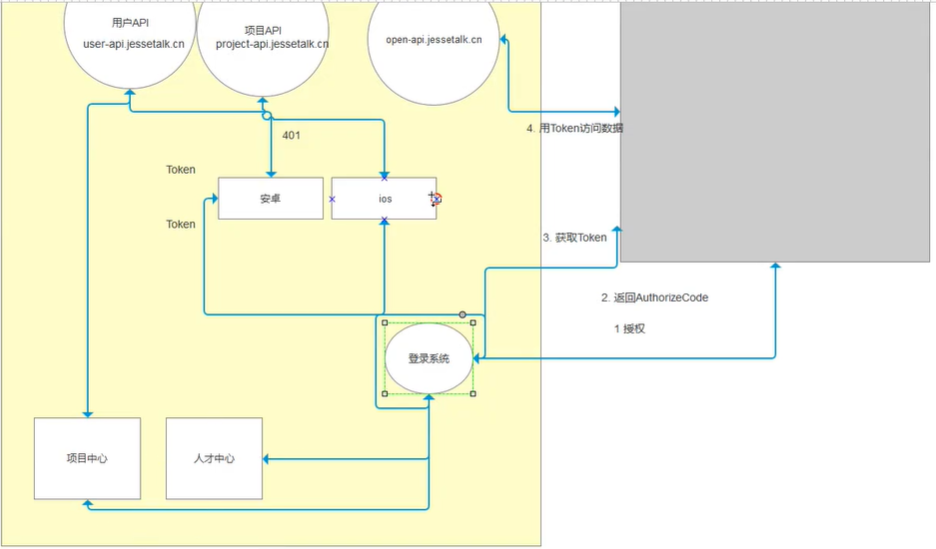
## 任务12：oauth密码模式identity server4实现

项目（IdentityServerSample2，ThirdPartyDemo2）

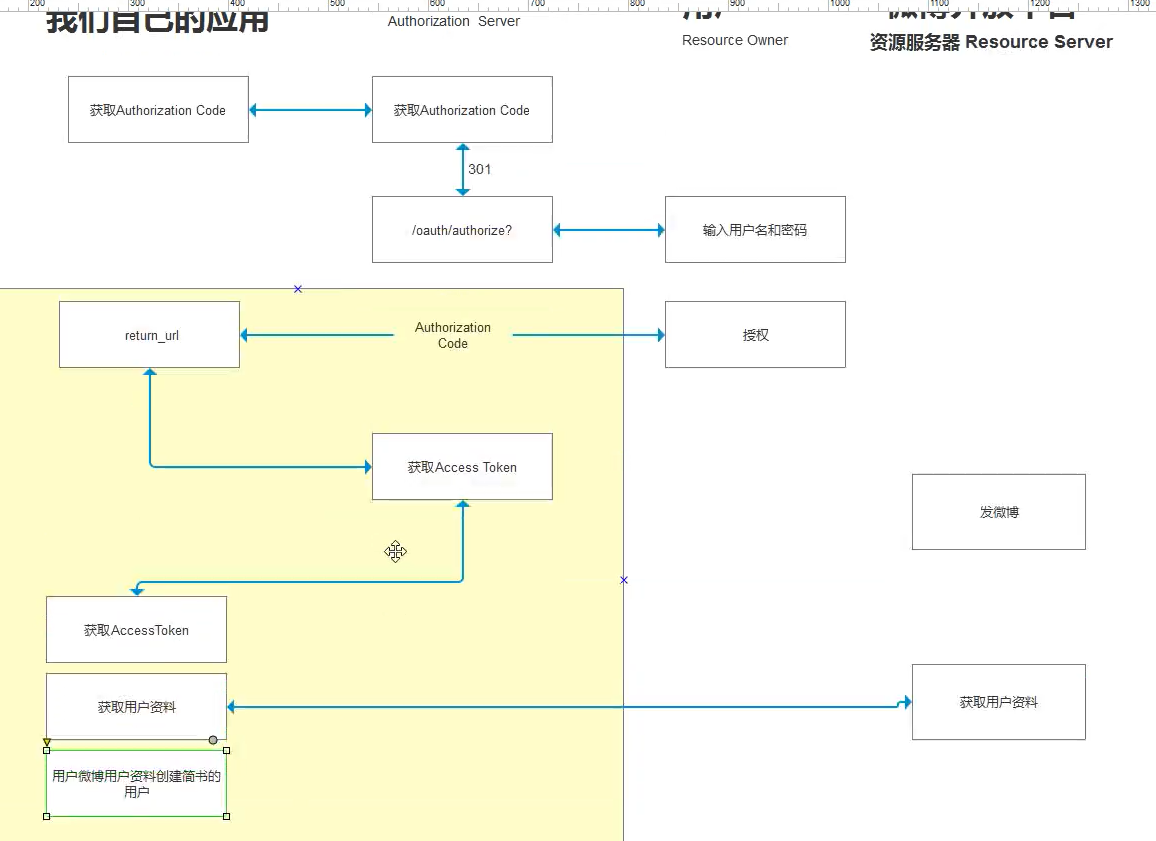


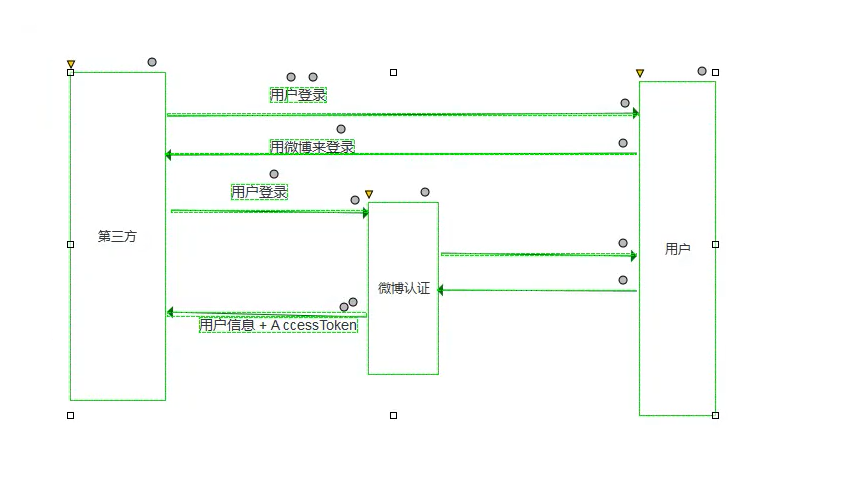


## 任务13：详解oauth2授权码流程



## 任务14：oauth2与open id connect 对比





## 任务15：oauth2 + oidc 实现 server部分

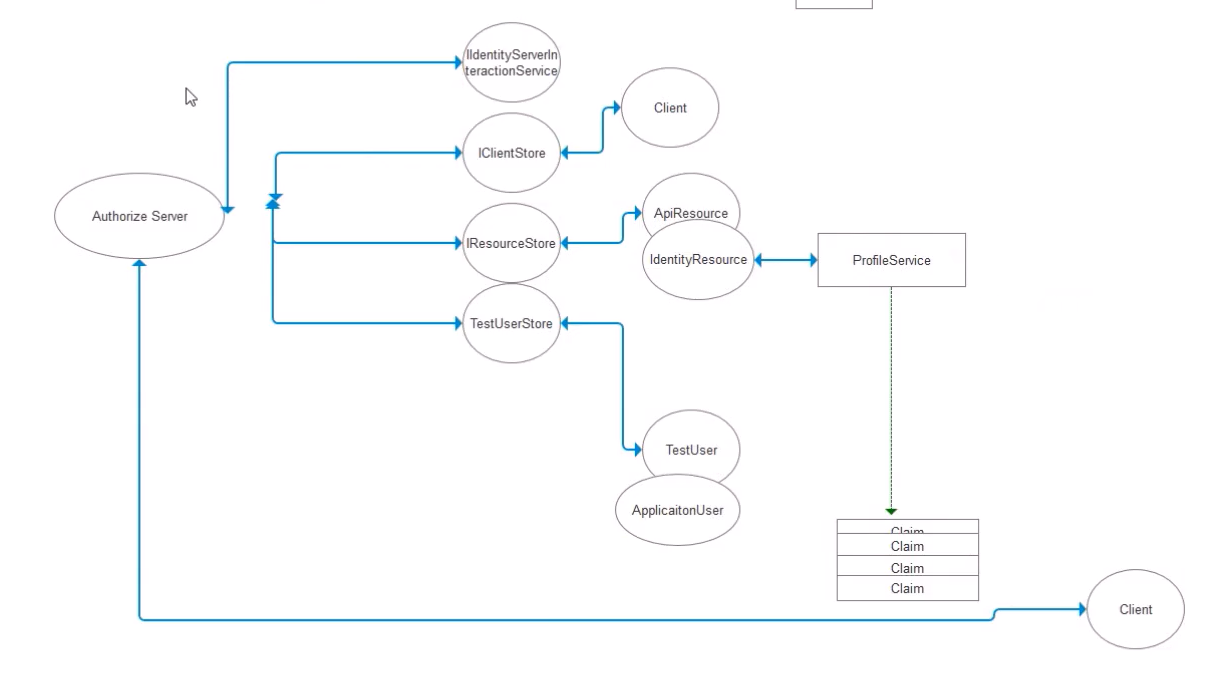
项目（MvcCookieAuthSample3）

## 任务16：oauth2 + oidc 实现 client部分

项目（MvcClient）

应用自己比较信用的应用

## 任务17：Identity Server 4回顾



## 任务18：Consent 实现思路介绍

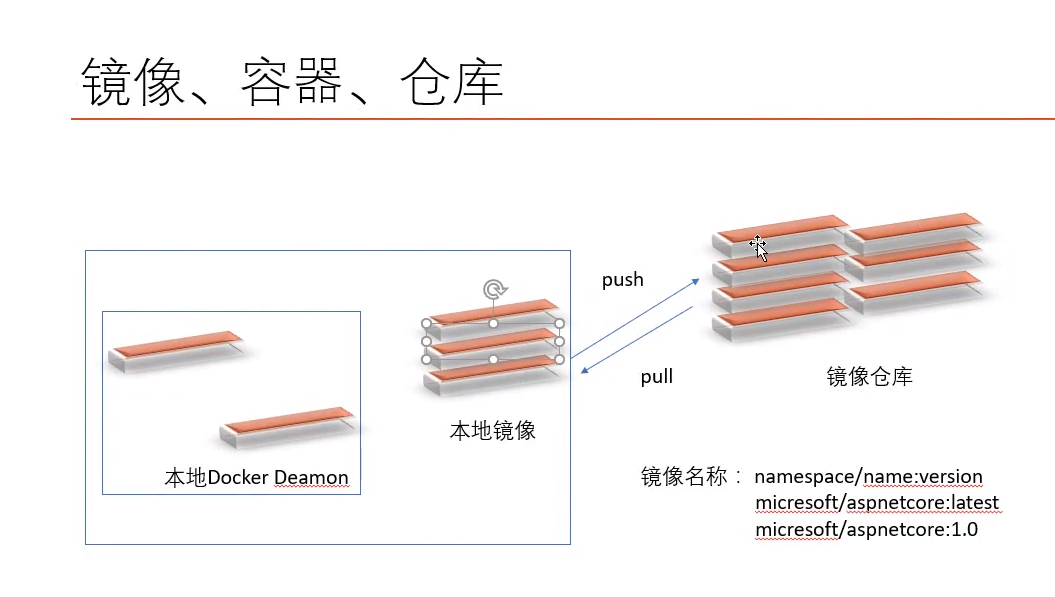
## 任务19：Consent视图制作

## 任务28：Docker介绍与目录

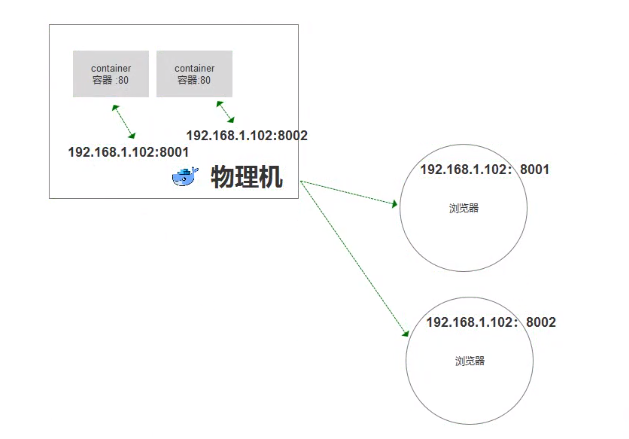




## 任务29：镜像容器与仓库



任务30：构建单机多容器环境-故障





任务32：Docker安装MongoDb

查看使用的端口：netstat

查看端口是否被使用：netstat -aon|findstr “27016”

删除容器：docker rn mymongo

创建mongo：docker run -d -p 27016:27016 –name mymongo mongo

查看本地镜像 ：docker ps -a

查看运行的镜像 docker ps

任务33：Docker安装Mysql及用户配置

Mysql 镜像拉取：docker pull mysql/mysql-server

安装：docker run -d -p 3306:3306 –name mysql101 mysql/mysql-server

任务38：Mysql EF Core 快速构建 web api

--安装mysql

docker run -d -p 3306:3306 -e MYSQL\_USER="jesse" -e MYSQL\_PASSWORD="123456" -e MYSQL\_ROOT\_PASSWORD="123456" --name mysql02 mysql:5.6 --character-set-server=utf8 --collation-server=utf8\_general\_ci

--mysql命令

docker exec -it mysql01 bash

连接：mysql -ujesse -p

pwd123456

ALTER USER 'root'@' localhost' IDENTIFIED BY '123456' 修改密码

GRANT ALL PRIVILEGES ON \*.\* TO 'jesse'@'%' WITH GRANT OPTION; 给权限

退出：exit

## 任务39：ASPNETCORE WEB API与MYSQL互联(构建镜像:不被推荐)

构建镜像

docker run -t jesse/aspnetcore:prod .

运行容器

docker run -d -p 8002:80 --name aspnetcore --link mysql02:db jesse/aspnetcore:prod

docker exec -it aspnetcore bash



查看端口



Docker inspect mysql02

## 任务40：Docker Network（构建镜像运行）

Bridge(单机)



1 Docker network create -d bridge mybridge

Docker network ls



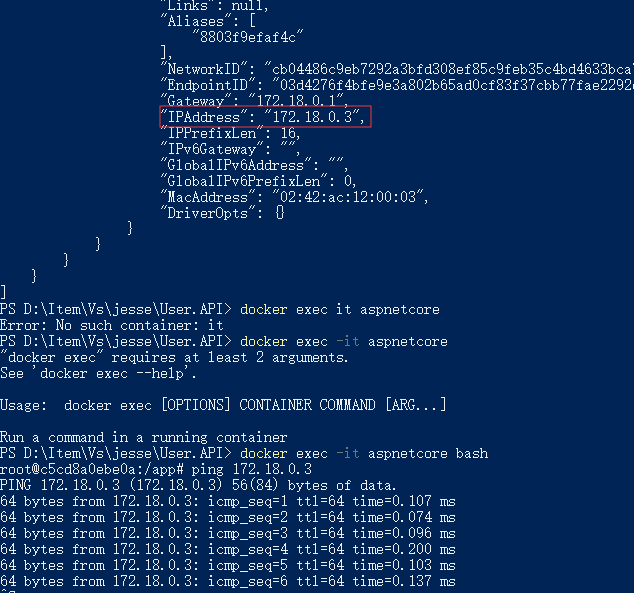
2 docker run -d -p 8002:80 --net mybridge --name aspnetcore jesse/aspnetcore:prod



docker network connect mybridge mysql02

docker exec -it aspnetcore bash

ping 下端口



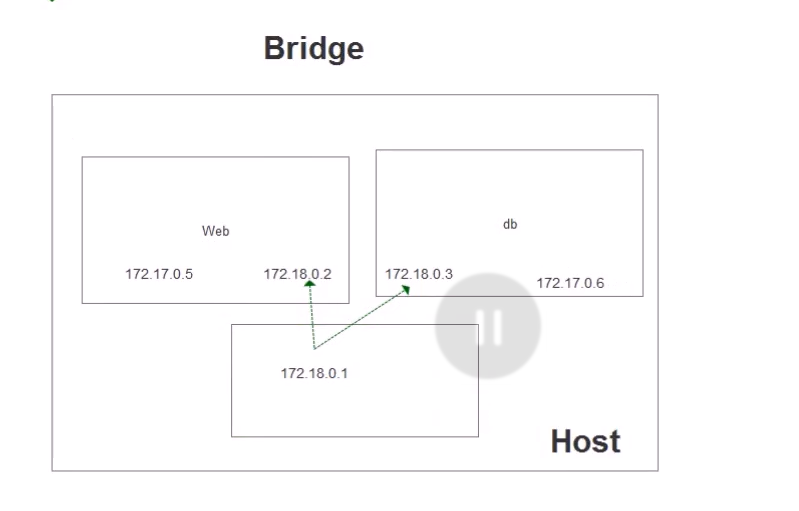
3 docker rename mysql02 db

第二种方式

改配置文件

第二种方式

挂载资料卷（推荐）



Overlay

## 任务41：制作 docker compose

制作镜像

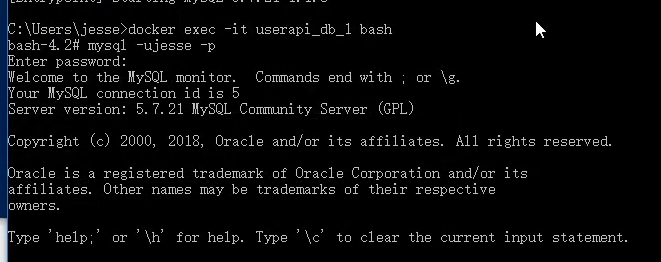
如制作不了，删除其他镜像

Docker-compose down

docker-compose build

docker-compose up

是否能登录成功





## 任务42：docker compose 启动容器(重点)

1 docker-compose up

重启

Docker-compose restart

Docker network ls

Docker inspect mysql02

给用户赋予权限

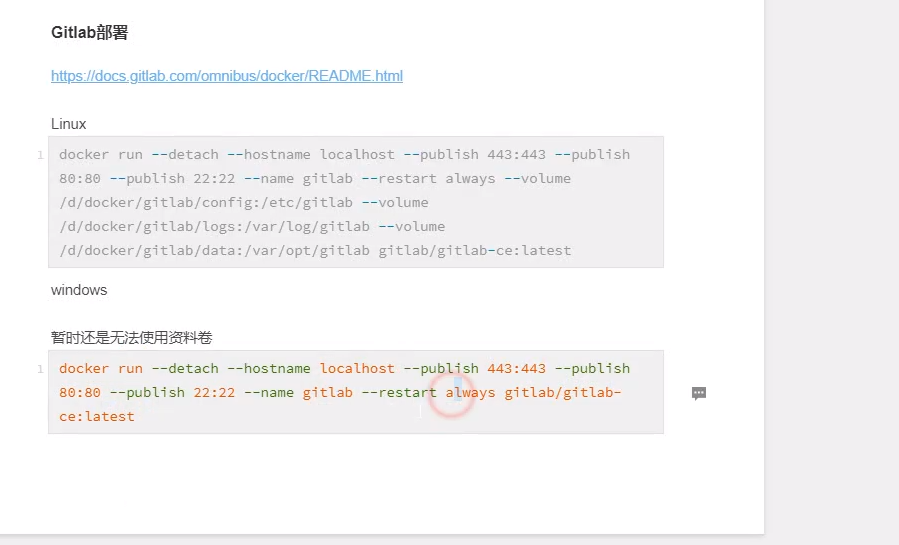
GRANT ALL PRIVILEGES ON \*.\* TO 'jesse'@'%' WITH GRANT OPTION;

先要把数据库创建好，需要给mysql用户赋予权限

## 任务43：启动问题解决方式

构造数据库的时候延时下

## 任务44：Gitlab安装



# docker pull gitlab/gitlab-ce:latest

# 官方版本是：gitlab/gitlab-ce:latest，为了提升速度我们这里使用阿里云的仓库

docker pull registry.cn-hangzhou.aliyuncs.com/imooc/gitlab-ce:latest

docker tag registry.cn-hangzhou.aliyuncs.com/imooc/gitlab-ce gitlab/gitlab-ce

docker run --detach --hostname localhost --publish 443:443 --publish 80:80 --publish 22:22 --name gitlab --restart always gitlab/gitlab-ce:latest

--查看安装日志

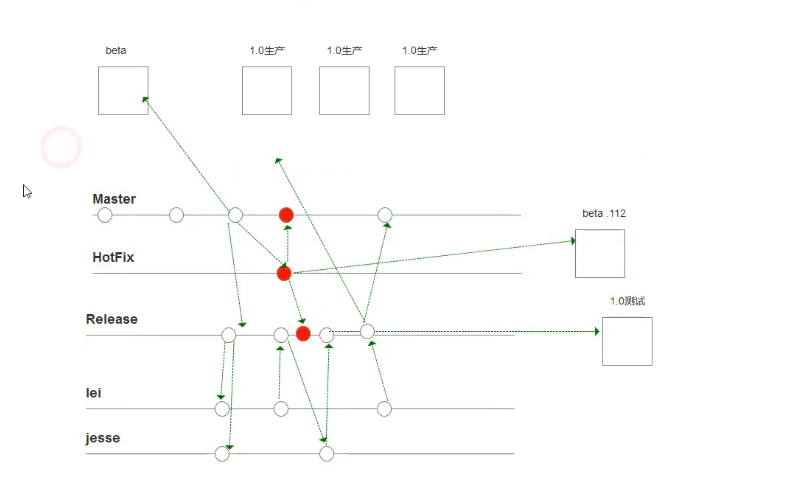
docker logs --follow gitlab

默认账号：root

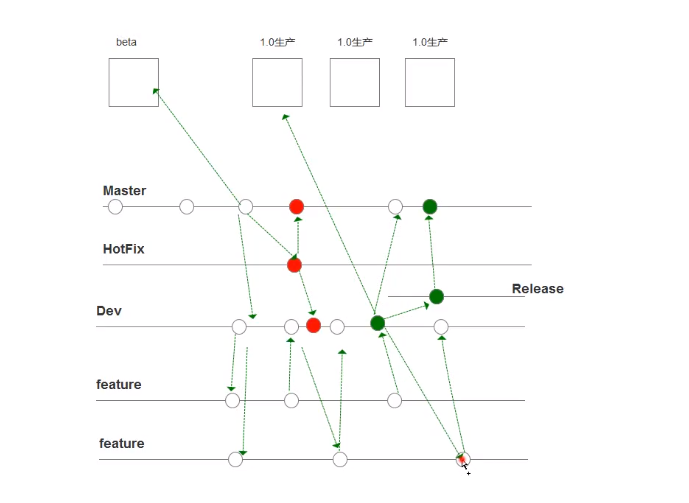
密码：chao411123

## 任务45：GitFlow的三种流程

一个团队的时候



多个团队时候



Feature 代表的是功能分支

## 任务46：GitLab Runner注册和安装

GitLab CI(持续集成)

下载安装

<https://docs.gitlab.com/runner/install/windows.html>

命令

gitlab.exe register

<http://localhost/>

bBD\_xoYxeSwnrP5Syimc

lacal-win

shell