

CICD - 持续集成与持续交付

持续集成 CI

什么是持续集成？（CI：Continuous Integration）

软件开发中，集成是一个很可能发生未知错误的过程。持续集成是一种软件开发实践，希望团队中的成员频繁提交代码到代码仓库，且每次提交都能通过自动化测试进行验证，从而使问题尽早暴露和解决。

持续集成的好处是什么？

持续集成可以使问题尽早暴露，从而也降低了解决问题的难度，正如老马所说，持续集成无法消除bug，但却能大大降低修复的难度和时间。

持续交付 CD

什么是持续交付？（CD：Continuous Delivery）

持续交付是持续集成的扩展，指的是将通过自动化测试的软件部署到产品环境。持续交付的本质是把每个构建成功的应用更新交付给用户使用。在持续交付的世界里，我们对完成的定义不是测试完成，而是交付到客户手中。这里需要注意的是，CD代表持续交付（Continuous Delivery）而不是持续部署（Continuous Deploy），因为部署也包括部署到测试环境，而持续交付代表的是功能的上线，交付给用户使用。

持续交付的好处是什么？

持续交付的好处在于快速获取用户反馈；适应市场变化和商业策略的变化。开发团队保证每次提交的修改都是可上线的修改，那么决定何时上线，上线哪部分功能则完全由产品业务团队决定。

虽然持续交付有显著的优点，但也有不成立的时候，比如对于嵌入式系统的开发，往往需要软硬件的配合。

基于Docker+Jenkins+Gitlab搭建持续集成环境

主机	服务
server1	GitLab (至少需要4G内存) 一般需要独立的服务器。下面教程用GitHub
server2	Jenkins+Docker

一、搭建GitLab服务器

gitlab是一款类似于github的源码管理平台，而且是开源的，一般公司的内部的代码处于保密都不会放到第三方平台上，这时候就需要公司内部搭建源码管理平台，gitlab的功能堪称完整和强大也具备CI/CD的功能

一、指定目录文件

```
mkdir -p /opt/data/gitlab && cd /opt/data/gitlab
```

二、我们使用 Docker 来安装和运行 GitLab 中文版， docker-compose.yml 配置如下：

```
version: '3'
services:
  gitlab:
    image: 'twang2218/gitlab-ce-zh:11.1.4' # gitlab/gitlab-ce:latest
    restart: always
    container_name: "gitlab"
    #privileged: true #特权模式
    hostname: 'gitlab'
    environment:
      TZ: 'Asia/Shanghai'
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://192.168.20.139'
        gitlab_rails['time_zone'] = 'Asia/Shanghai'
        gitlab_rails['gitlab_shell_ssh_port'] = 58522
        gitlab_rails['smtp_enable'] = true
    ports:
      - '80:80' #阿里云服务器端口改为 8086
      - '443:443'
      - '58522:22'
    volumes:
      - /opt/gitlab/config:/etc/gitlab
      - /opt/gitlab/data:/var/opt/gitlab
      - /opt/gitlab/logs:/var/log/gitlab
```

三、执行docker-compose命令进行安装

```
docker-compose up -d
```

四、查看输出的日志，需要等待几分钟就可以安装完成了

```
docker-compose logs -f
```

五、启动页面并设置用户名密码

请为您的新帐户创建密码。

GitLab 中文社区版

用于代码协作的开源软件

细粒度访问控制管理 git 仓库以保证代码安全。使用合并请求进行代码审查并加强团体合作。每个项目均有自己的问题跟踪和维基页面。

修改密码

新密码

.....

确认新密码

.....

密码统一设定

修改密码

没有收到确认邮件？[重新发送](#)

已有账号和密码？[登录](#)



项目

欢迎来到 GitLab

集代码，测试和部署于一体。



创建一个项目

项目是存储你的代码的地方，里面还包含问题列表、维基文档以及其它一些 GitLab 的功能。



创建一个群组

使用群组管理项目和人员是非常好的方式。



添加人员

添加你的团队成员或者其它人到 GitLab。



配置 GitLab

对你的 GitLab 服务器的设置进行调整。

GitLab应用

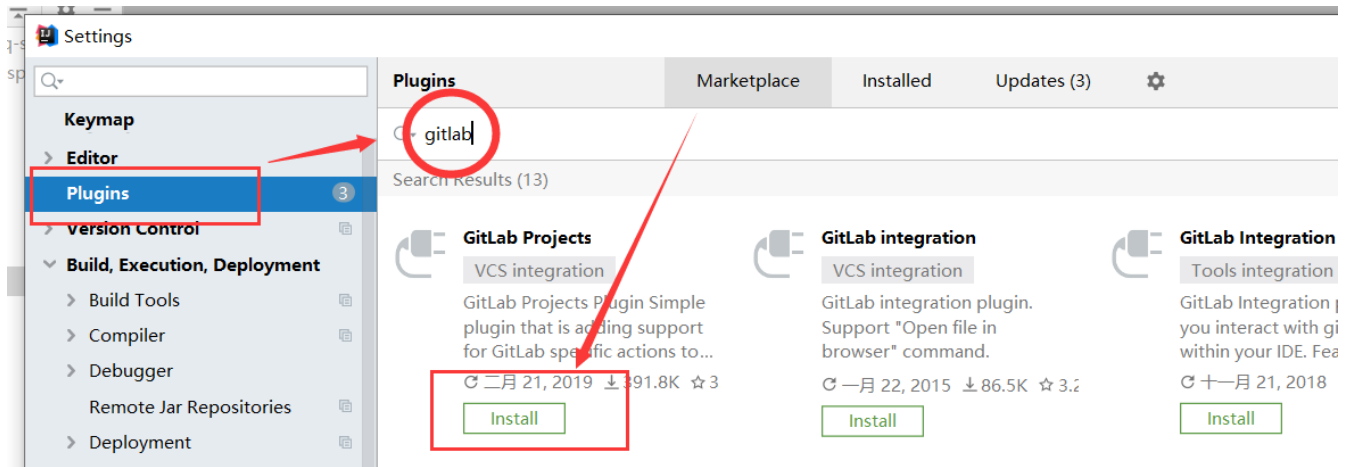
GitLab的应用操作和GitHub的应用一样。

GitLab集成IDEA

首先打开IntelliJ IDEA，点击左上角菜单File，然后弹出的菜单中选择Settings

然后在左边的菜单中选择Plugins，这时候右边会有一个搜索框，输入gitlab,然后回车搜索

找到GitLab Project，点击install，这时候会有一个二次确认框，点击Accept，然后等待安装完成

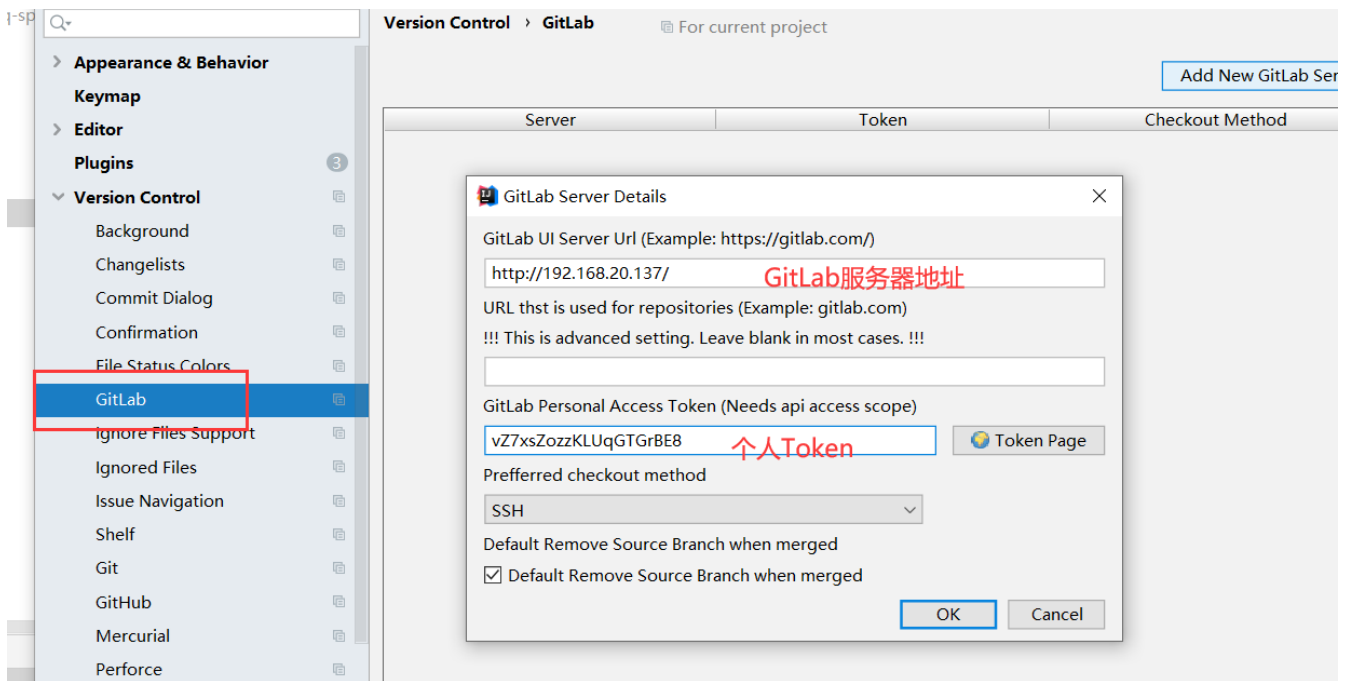


安装完成之后如下，点击Restart IDE，这时候也会有一个二次确认框，点击Restart，重启

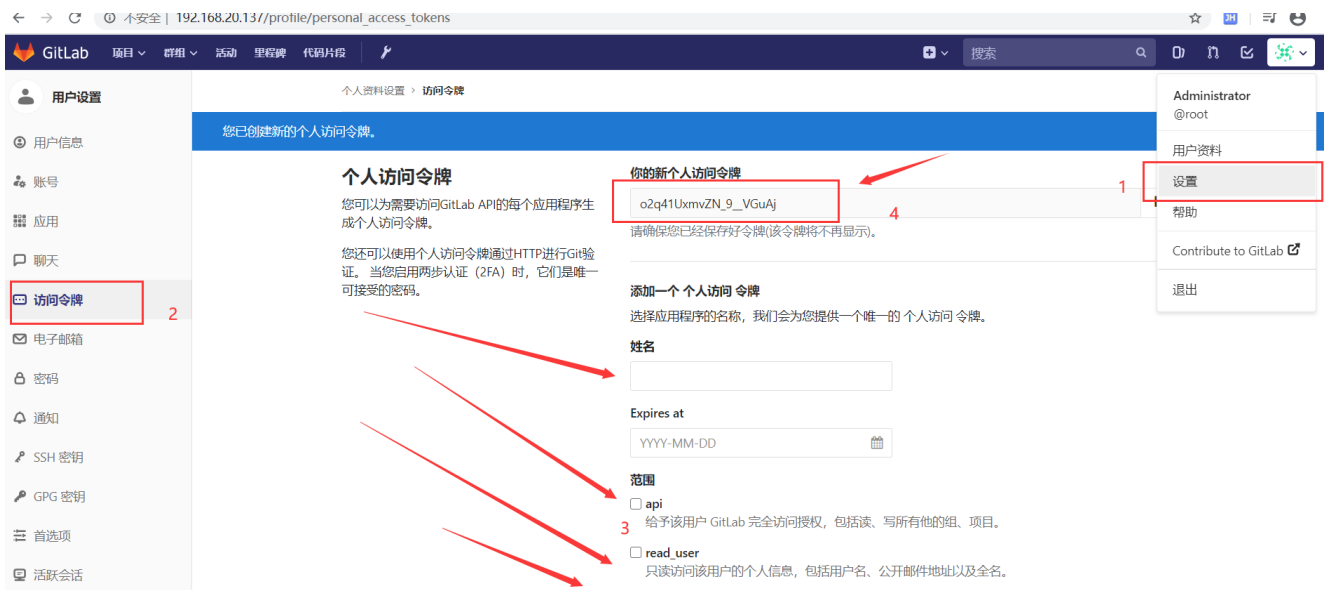
配置GitLab

点击File => Settings => Version Control => GitLab

这里面主要配置GitLab Server Url和个人的私有访问token，如下：



GitLab中获得的Token



二、Jenkins安装

Jenkins是一个**开源软件**项目，是基于**Java**开发的一种**持续集成**工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

官网

<https://www.jenkins.io/zh/>

Docker 安装

1 安装

```
docker pull jenkins
```

2 运行

```
docker run -d -p 8002:8080 -v ~/jenkins:/var/jenkins_home --name jenkins --restart=always jenkins
```

Docker-compose 安装（此处安装）

#1 指定目录

```
mkdir -p /opt/docker_jenkins
```

#2 编写 docker-compose.yml

```
version: '3.0'
services:
  jenkins:
    image: jenkins/jenkins
    restart: always
    container_name: jenkins
    ports:
      - 8888:8080
      - 50000:50000
    volumes:
      - ./data:/var/jenkins_home
```

#3 启动运行

```
docker-compose up -d
```

#4 查看日志

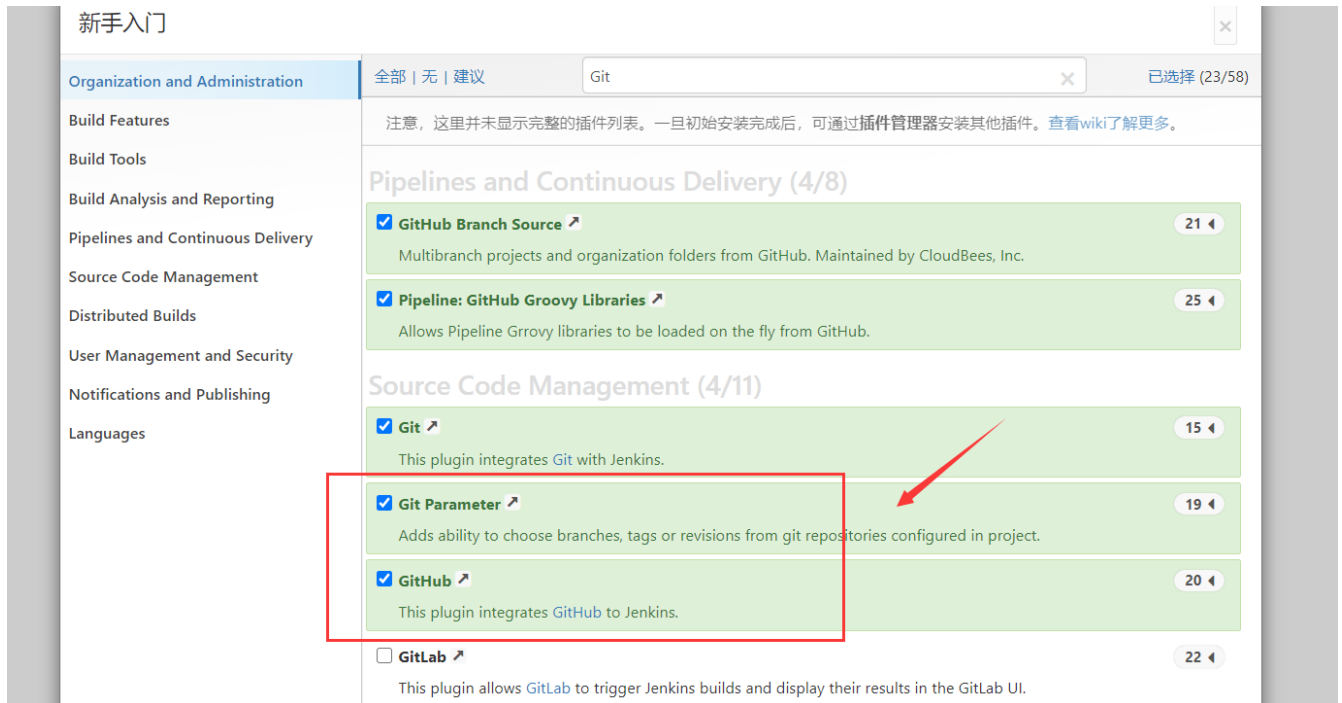
```
docker-compose logs -f
# 第一次启动, 无法继续执行, 原因是 data 没有权限
chmod 777 data
#重启
docker-compose restart #重启
#日志查看
docker-compose logs -f
```

Jenkins运行

<http://192.168.20.135:8888/>



自定义插件准备

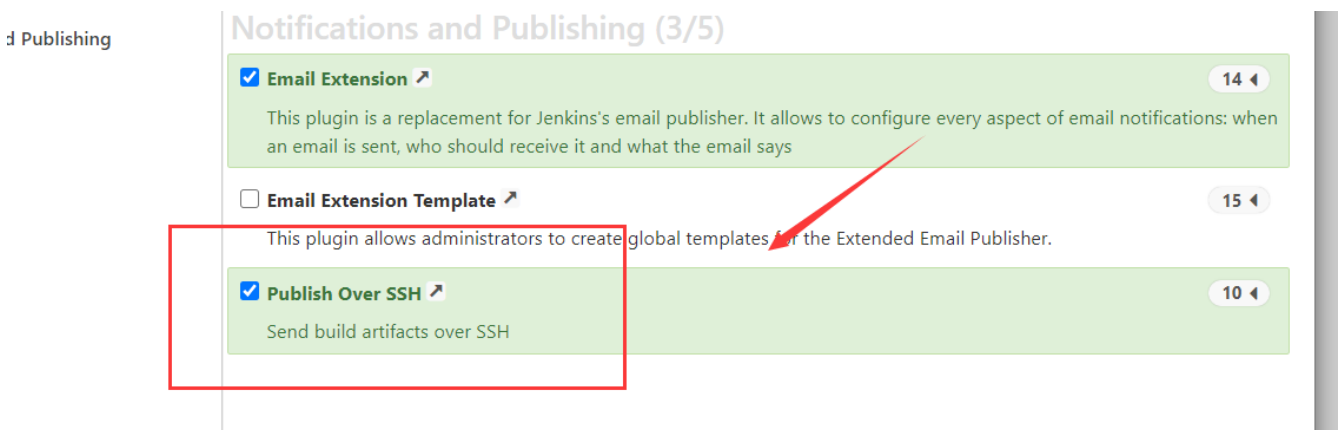


安装 GitHub 插件

首先，需要连接 GitHub 有一个基本的插件要安装，可以在插件管理中搜索 GitHub，然后找到 **GitHub** 这个插件进行安装即可。

安装 Git Parameter

安装了 **GitHub** 插件就已经实现了连接 GitHub，虽然这个基本的插件本身也有选择分支的参数，但是分支参数没有限制，无法做到根据实际的分支和 Tag 名称去选择，所以最好另外安装一个可以支持选择分支和 Tag 的插件，这个支持分支的插件的名字是 **Git Parameter**，这个插件可以实现在拉取 GitHub 的代码的时候选择分支和 Tag 并通过参数的形式传入到拉取过程中。



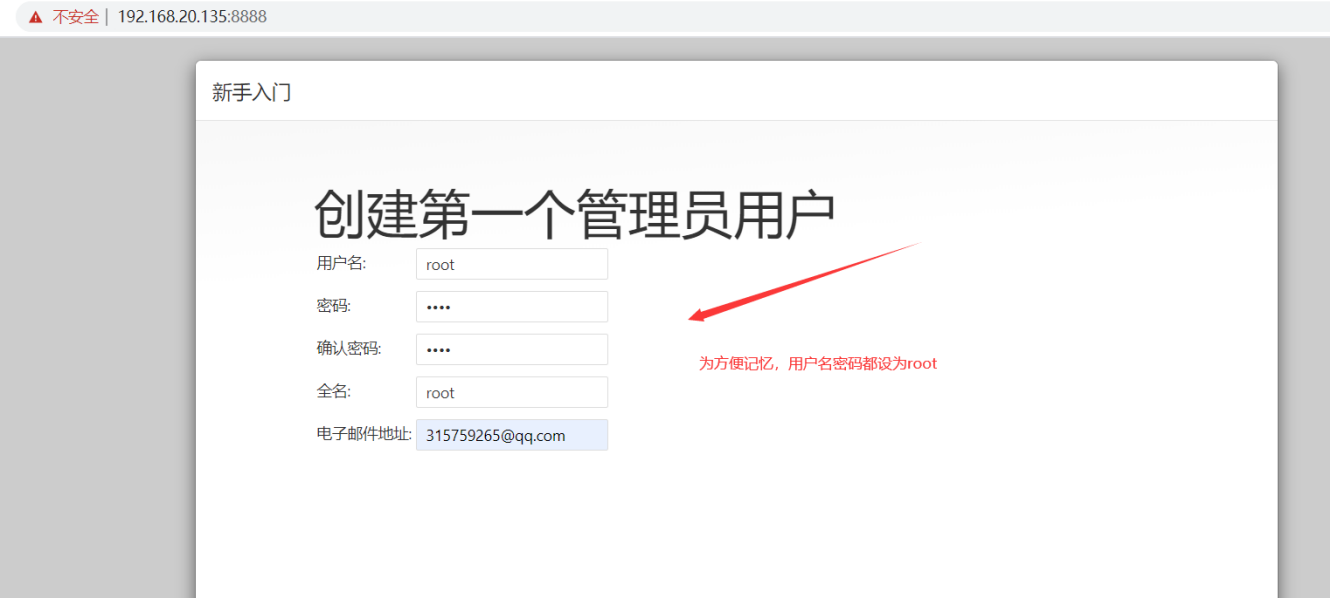
安装 Publish over SSH

Jenkins通过Publish over SSH插件实现远程部署（后续可以在如下进行安装）

主界面-->系统管理-->管理插件-->可选插件-->右上角过滤框中输入“Publish over SSH”-->勾选安装

等待漫头的安装....

指定用户名和密码



安装完成



三、Jenkins连接GitLab服务器

为方便Jenkins拉取GitLab服务器的密码，需要 连接GitLab服务器

Jenkins中点击 系统管理 --》全局配置

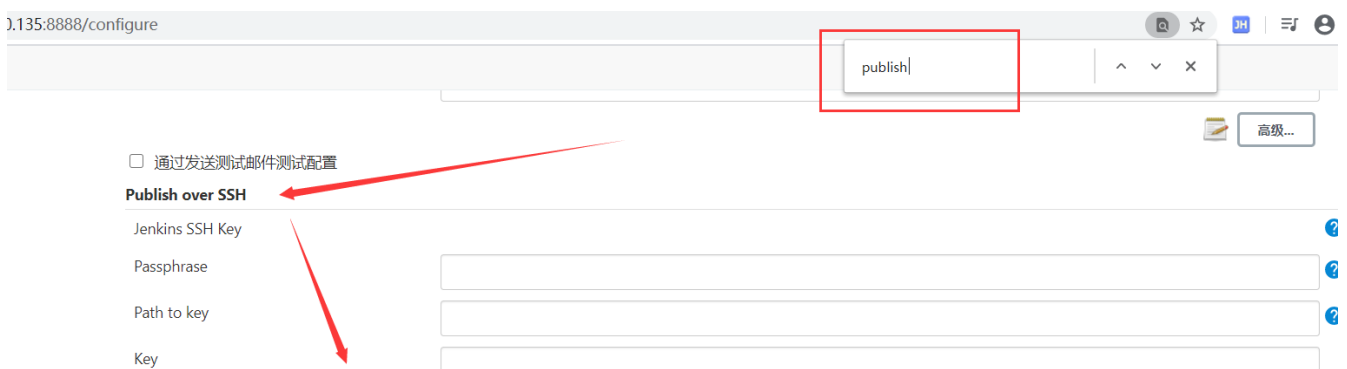


The image shows the Jenkins 'System Management' (系统管理) interface. On the left is a sidebar with links: '新建任务' (New Task), '用户列表' (User List), '构建历史' (Build History), '系统管理' (System Management), '我的视图' (My Views), 'Lockable Resources', '凭据' (Credentials), and '新建视图' (New View). The '系统管理' link is highlighted with a red box. A red arrow points from this box to the '系统配置' (System Configuration) option in the main content area. The main content area is titled '管理Jenkins 系统配置' (Manage Jenkins System Configuration) and contains two main sections: '系统配置' (System Configuration) with the subtitle '配置全局设置和路径' (Configure global settings and paths), and '节点管理' (Node Management) with the subtitle '添加、删除、控制和监视系统运行任务的节点。' (Add, delete, control and monitor system task nodes). A '全局工具' (Global Tools) section is also visible on the right. At the bottom left, there is a '构建队列' (Build Queue) section.

找到 Publish over SSH(如果没有，需要下载该插件)



The image shows a browser window displaying the Jenkins configuration page for the 'publish' plugin. The address bar shows '192.168.20.135:8888/configure'. The page title is 'publish'. A red box highlights the 'Publish over SSH' option under the '通过发送测试邮件测试配置' (Test configuration by sending test email) section. Below this option are input fields for 'Jenkins SSH Key', 'Passphrase', and 'Path to key'.



The image shows the same Jenkins configuration page for the 'publish' plugin, but with additional red arrows. One red arrow points from the 'publish' tab in the browser's tab bar to the 'Publish over SSH' option. Another red arrow points from the 'Publish over SSH' option to the 'Jenkins SSH Key' input field. A third red arrow points from the 'Jenkins SSH Key' input field to the 'Key' input field at the bottom of the page.

Disable exec

SSH Servers

SSH Server

Namejenkins-gitlab服务的名称

Hostname192.168.20.137GitLab服务器的IP地址

UsernamerootGitLab服务器的登录名

Remote Directory/usr/local/jenkins此处指定的GitLab服务器的目录必须存在

☒ Use password authentication, or use a different key

Passphrase / Password.....登录GitLab服务器的密码

Path to key

Jump host

Port22暴露的端口号

Timeout (ms)300000

Disable exec☐

Proxy type

Proxy host

Proxy port

Proxy user

Proxy password

Success

Test Configuration

删除

四、Jenkins免密钥登录GitLab

- 1、登录Jenkins容器内部
- 2、输入SSH生成密钥命令
- 3、复制到GitLab配置项目

1 找到Jenkins 容器ID, 并进入容器内容

```
docker ps
```

```
docker exec -it 容器ID bash
```

2输入SSH生成密钥命令 (如下图)

```
ssh-keygen -t rsa -C "315759265@qq.com" #输入3次确定回车即可 邮箱随意
```

```

nutes      0.0.0.0:3306->3306/tcp      mysql-3306
[root@192 docker_jenkins]# docker exec -it ca bash
jenkins@cale60ea751c:/$ ssh-keygen -t rsa -C "315759265@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa):
Created directory '/var/jenkins_home/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/jenkins_home/.ssh/id_rsa.
Your public key has been saved in /var/jenkins_home/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AlCWFFERJaT/ajq2+bdU1QLFPR9ZJWfmB7CEi+b02R4 315759265@qq.com
The key's randomart image is:
+---[RSA 2048]---+
| .+O=+o.. .o+..= |
| +.o o. +oo. B |
| . o . ..O+.. o |
| . o +o. . . |
| . .+S. o |
| . o. o E |

```

密钥生成的目录文件下

```

# 3 /var/jenkins_home 在安装时已和宿主机 /data目录完成映射。退出容器，找到宿主机对应目录
cd /opt/docker_jenkins/data #进入宿主机 /data目录
# 4 .ssh文件默认为隐藏文件，进入文件内部（如下图）
ls -a
cd .ssh
ll

```

```

workflow-libs
[root@192 data]# cd .ssh
[root@192 .ssh]# ll
总用量 8
-rw-----. 1 guoweixin guoweixin 1675 5月 24 17:37 id_rsa
-rw-r--r--. 1 guoweixin guoweixin 398 5月 24 17:37 id_rsa.pub
[root@192 .ssh]#

```

```

# 5 进入 id_rsa.pub 文件。将内容复制
cat id_rsa.pub

```

打开GitLab主页服务器，进行密钥配置

The screenshot shows the GitLab web interface for configuring SSH keys. The left sidebar contains navigation links: 用户设置 (User Settings), 用户信息 (User Information), 账号 (Account), 应用 (Applications), 聊天 (Chat), 访问令牌 (Access Tokens), 电子邮箱 (Email), 密码 (Password), 通知 (Notifications), SSH 密钥 (SSH Keys), GPG 密钥 (GPG Keys), 首选项 (Preferences), and 语言设置 (Language Settings). The main content area is titled 'SSH 密钥' and shows a list of keys. A red arrow points to the 'Add key' button. Another red arrow points to the 'id_rsa.pub' file in the terminal output above. A third red arrow points to the 'id_rsa.pub' file in the terminal output above. A fourth red arrow points to the 'id_rsa.pub' file in the terminal output above.

拉取GitLab项目

使用SSH名密钥连接时，首次需要手动确定操作

```
# 1、进入Jenkins容器内部
docker exec -it jenkins容器ID bash
# 2、执行 拉取 clone
git clone (gitlab ssh连接地址) //如果无权限，执行3操作
# 3、进入指定目录（操作）
cd /var/jenkins_home/
# 4、再次执行 git clone
git clone (gitlab ssh连接地址)
# 5、手动选择yes 即可完成首次需要确定操作
# 6、拉取完成后，可以直接删除。
rm -rf 文件夹名
```

五、Jenkins配置JDK和MAVEN

Jenkins会从GitHub/GitLab 中拉取代码。会在Jenkins中进行打包等操作。此处需要JDK和MAVEN配置

安装文件准备

官网直接搜索下载即可

```
# 1 复制本地的 JDK和MAVEN 到Linux桌面。并移动 /opt/docker_jenkins/data目录下
mv jdk-8u141-linux-x64.tar.gz /opt/docker_jenkins/data
mv apache-maven-3.6.3-bin.tar.gz /opt/docker_jenkins/data
```

```
# 2 依次在该目录下完成解压
cd /opt/docker_jenkins/data
tar -zxvf apache-maven-3.6.3-bin.tar.gz
tar -zxvf jdk-8u141-linux-x64.tar.gz
```

Jenkins中配置

1、配置

系统配置--》全局工具配置



2、配置JDK

JDK

JDK 安装

新增 JDK



JDK

别名

jdk 起的别名

JAVA_HOME

/var/jenkins_home/jdk1.8.0_141 解压的目录

☐ 自动安装

3、配置MAVEN

Maven

Maven 安装

新增 Maven



Maven

Name

maven

MAVEN_HOME

/var/jenkins_home/apache-maven-3.6.3

☐ 自动安装

删除 Maven

新增 Maven

保存 即可

六、测试

让Jenkins实行一个任务，拉取GitLab中的代码。并在Jenkins中 用Maven打成一个jar包

- 1、创建一个项目 推送到GitLab服务器
- 2、Jenkins页面中创建任务（拉取GitLab代码）
- 3、运行Maven命令，进行打包

Jenkins创建任务

Jenkins首页，新建任务

← → ↻ 不安全 | 192.168.20.135:8888



Jenkins

Jenkins ▶

- 新建任务
- 用户列表
- 构建历史
- 系统管理
- 我的视图
- Lockable Resources
- 凭据
- 新建视图

欢迎来到 Jenkins!

[Create an agent](#) or [configure a cloud](#) to set up distributed builds. [Learn more](#)

开始[创建一个新任务](#)。

输入任务名称，（自由风格）。 点击确定

输入一个任务名称

» 必填项



构建一个自由风格的软件项目

这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.



流水线

精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。



构建一个多配置项目

适用于多配置项目,例如多环境测试,平台指定构建,等等.

依次完成()

General 源码管理 构建触发器 构建环境 构建 构建后操作

描述

[纯文本] 预览

☐ GitHub 项目

☐ This build requires lockable resources

☐ Throttle builds

☒ 丢弃旧的构建

策略 Log Rotation

保持构建的天数

如果非空，构建记录将保存此天数

保持构建的最大个数

如果非空，最多此数目的构建记录将被保存

高级...

源码管理Git

General **源码管理** 构建触发器 构建环境 构建 构建后操作

源码管理

☐ 无

☒ Git

Repositories

Repository URL GitLab项目 SSH地址

Credentials 添加

高级...

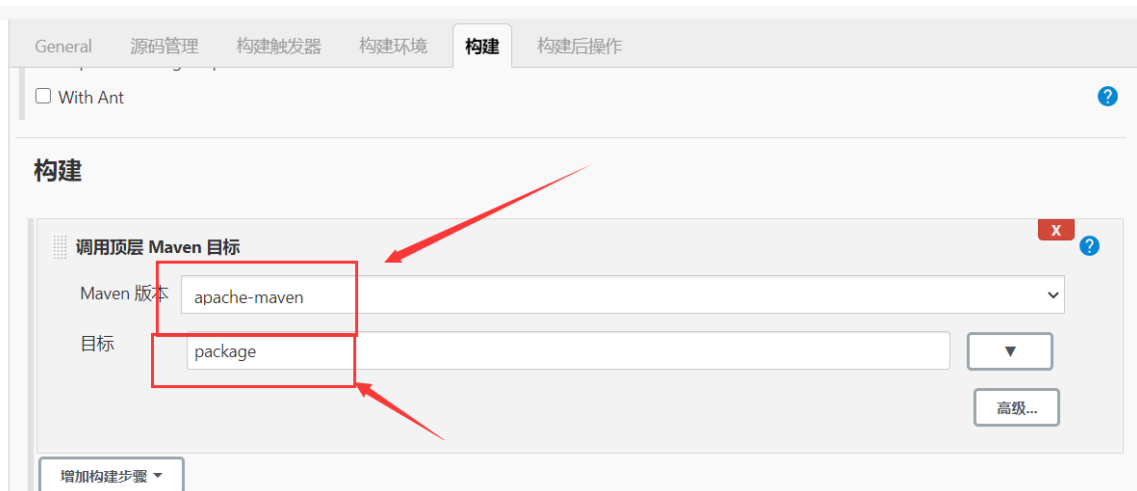
Add Repository

Branches to build

指定分支 (为空时代表any)

增加分支

构建 (选择Maven)



保存即可。

执行任务

立即构建--》点击查看构建详情

- 返回面板
- 状态
- 修改记录
- 工作空间
- 立即构建**
- 删除工程
- 配置
- 重命名

工程 test_demo

- 工作区
- 最新修改记录

相关链接



可点击查看构建详情

首次构建比较慢。控制台会输出 构建的全过程

Jenkins

test_demo

#1

返回到工程

状态集

变更记录

控制台输出

文本方式查看

编辑编译信息

Git Build Data

No Tags

控制台输出

Started by user root
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/test_demo
No credentials specified
Cloning the remote Git repository
Cloning repository ssh://git@192.168.20.137:58522/root/demo.git
> git init /var/jenkins_home/workspace/test_demo # timeout=10
Fetching upstream changes from ssh://git@192.168.20.137:58522/root/demo.git
> git --version # timeout=10
> git fetch --tags --progress -- ssh://git@192.168.20.137:58522/root/demo.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url ssh://git@192.168.20.137:58522/root/demo.git # timeout=10
Fetching upstream changes from ssh://git@192.168.20.137:58522/root/demo.git
> git fetch --tags --progress -- ssh://git@192.168.20.137:58522/root/demo.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master {commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master {commit} # timeout=10
Checking out Revision 6eaf7769b83d12f421a72d2571fa449d391221e9 (refs/remotes/origin/master)

查看执行结果

在构建 控制台日志最下方 可看到构建的 jar包目录地址。找进去查看是否 maven package成功

- 1、 进入 数据卷映射的目录位置
cd /opt/docker_jenkins/data
- 2、进入/workspace目录
cd workspace
- 3、在下方即可找到打好的 jar包

IDEA中提交代码--》GitLab--》Jenkins运行任务--》完成镜像构建

要用换行来表示依次的命令

General

源码管理

构建触发器

构建环境

构建

构建后操作

☐ Inspect build log for published Gradle build scans

☐ With Ant

构建

调用顶层 Maven 目标

Maven 版本

apache-maven

目标

clean
package
docker:build

高级...

七、实战测试 持续交付和持续部署

135 是 Jenkins

137是 GitLab