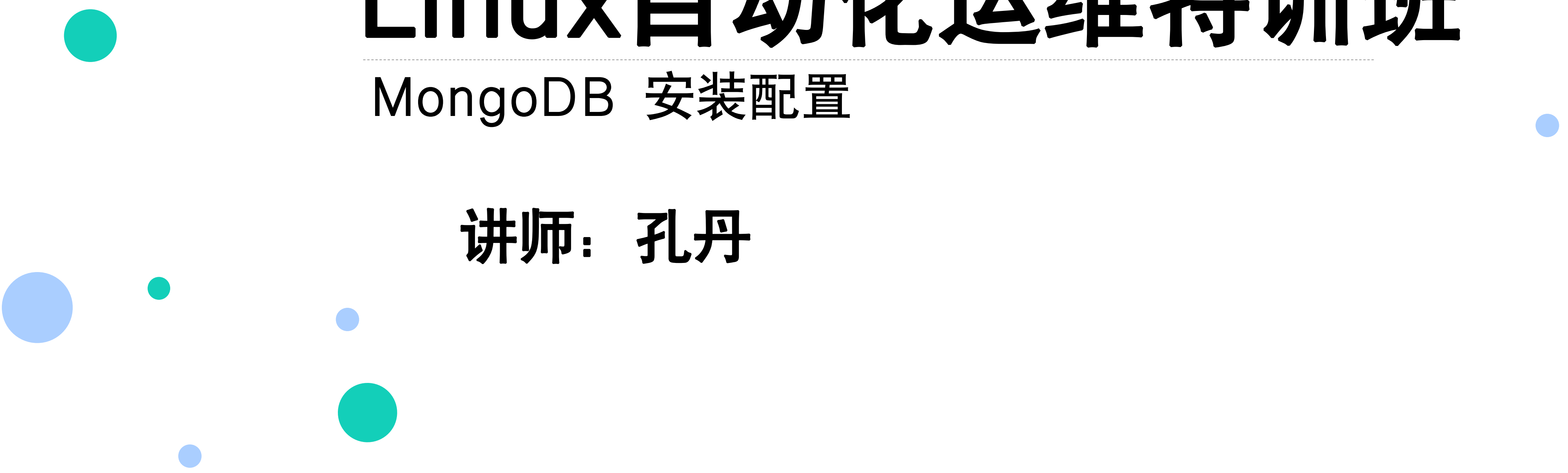




Linux自动化运维特训班

基于Jenkins实现持续集成

讲师：孔丹



大纲

- 持续集成简介
- Jenkins简介
- Jenkins配置
- Jenkins使用

持续集成简介

□ 什么是持续集成？

Continuous integration (CI)

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽快地发现集成错误。许多团队发现这个过程可以大大减少集成的问题，让团队能够更快的开发内聚的软件。

□ 没有持续集成

项目做模块集成的时候，发现很多接口都不通。

- 浪费大量时间

需要人手动去编译打包最新的代码。

- 构建过程不透明

发布代码，上线，基本靠手。

- 脚本乱飞

持续集成简介

□ 持续集成最佳实践

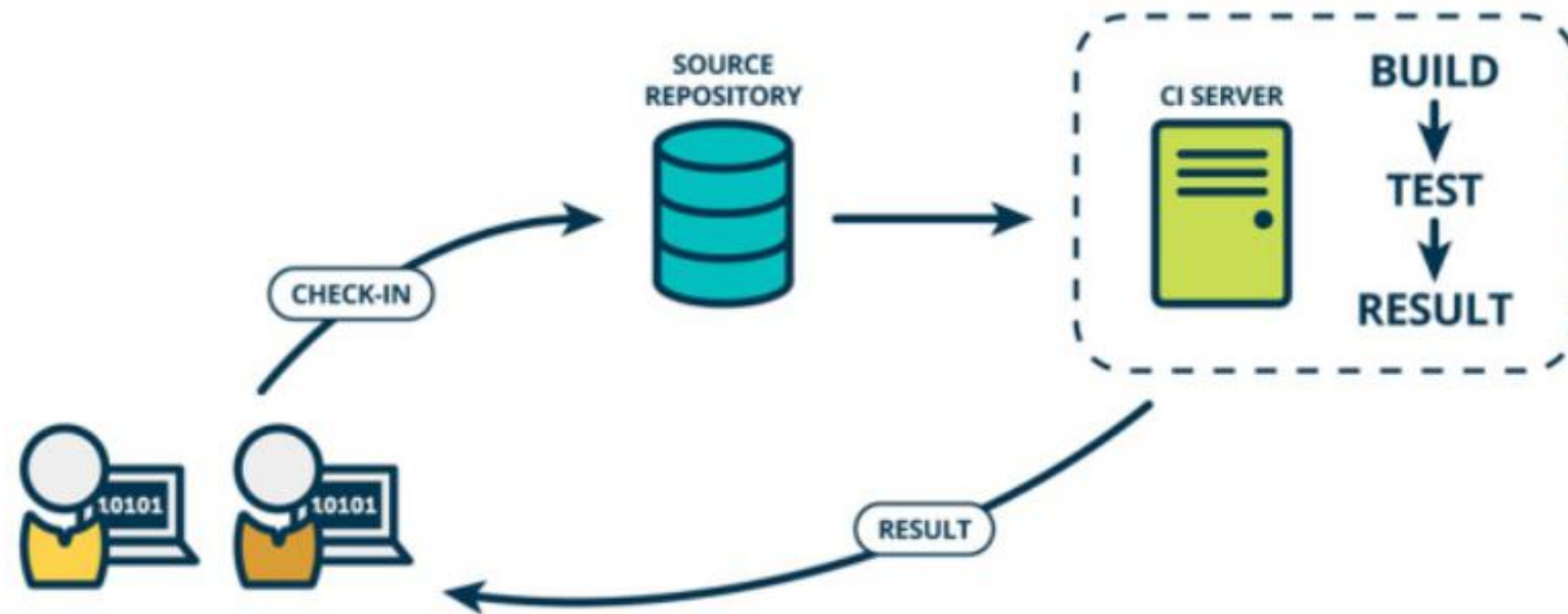
1. 维护一个单一的代码库 【不要使用多个代码库，避免每个组使用单独得git/svn，减轻权限管理】
2. 使构建自动化
3. 执行测试是构建的一部分 【构建一个项目得时候，测试就包含在里面了，可以及时得发现问题】
4. 集成日志及历史记录
5. 使用统一的依赖包管理库 【很容易出错，所以要统一得管理】
6. 每天至少集成一次 【持续集成得时候，是最容易发现问题得时候】

持续集成简介

□ 持续集成概览

持续集成得过程：

先把代码放到git、Jenkins从git获取代码进行构建、测试、生成结果再返回给客户端。



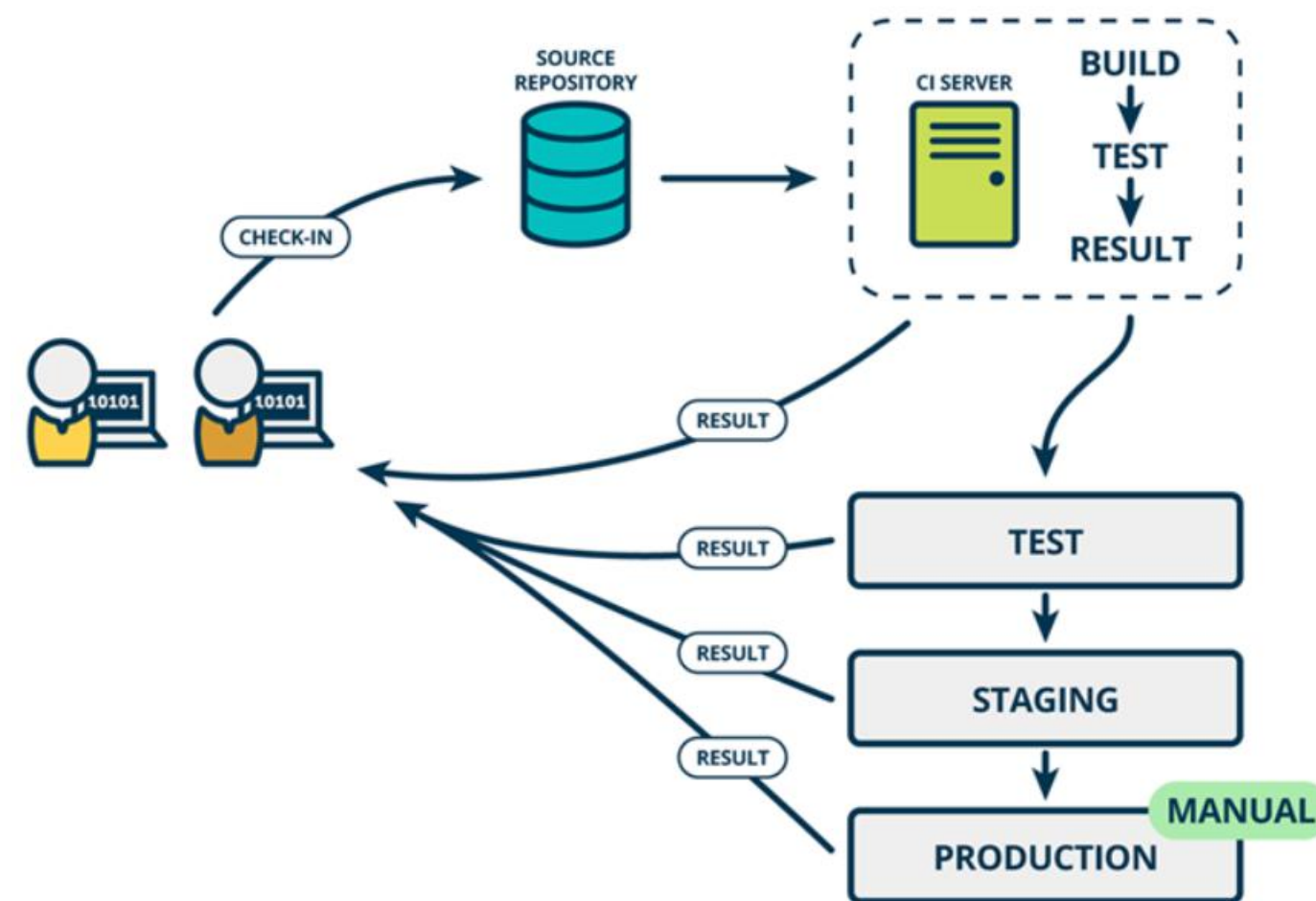
持续交付

□ 持续交付

持续交付（英语：Continuous delivery，缩写为 CD），是一种软件工程手法，让软件产品的产出过程在一个短周期内完成，以保证软件可以稳定、持续的保持在随时可以释出的状况。

它的目标在于让软件的建置、测试与释出变得更快以及更频繁。这种方式可以减少软件开发的成本与时间，减少风险。

持续交付在持续集成的基础上，将集成后的代码部署到更贴近真实运行环境的「类生产环境」（production-like environments）中。比如，我们完成单元测试后，可以把代码部署到连接数据库的 Staging 环境中更多的测试。如果代码没有问题，可以继续手动部署到生产环境中。



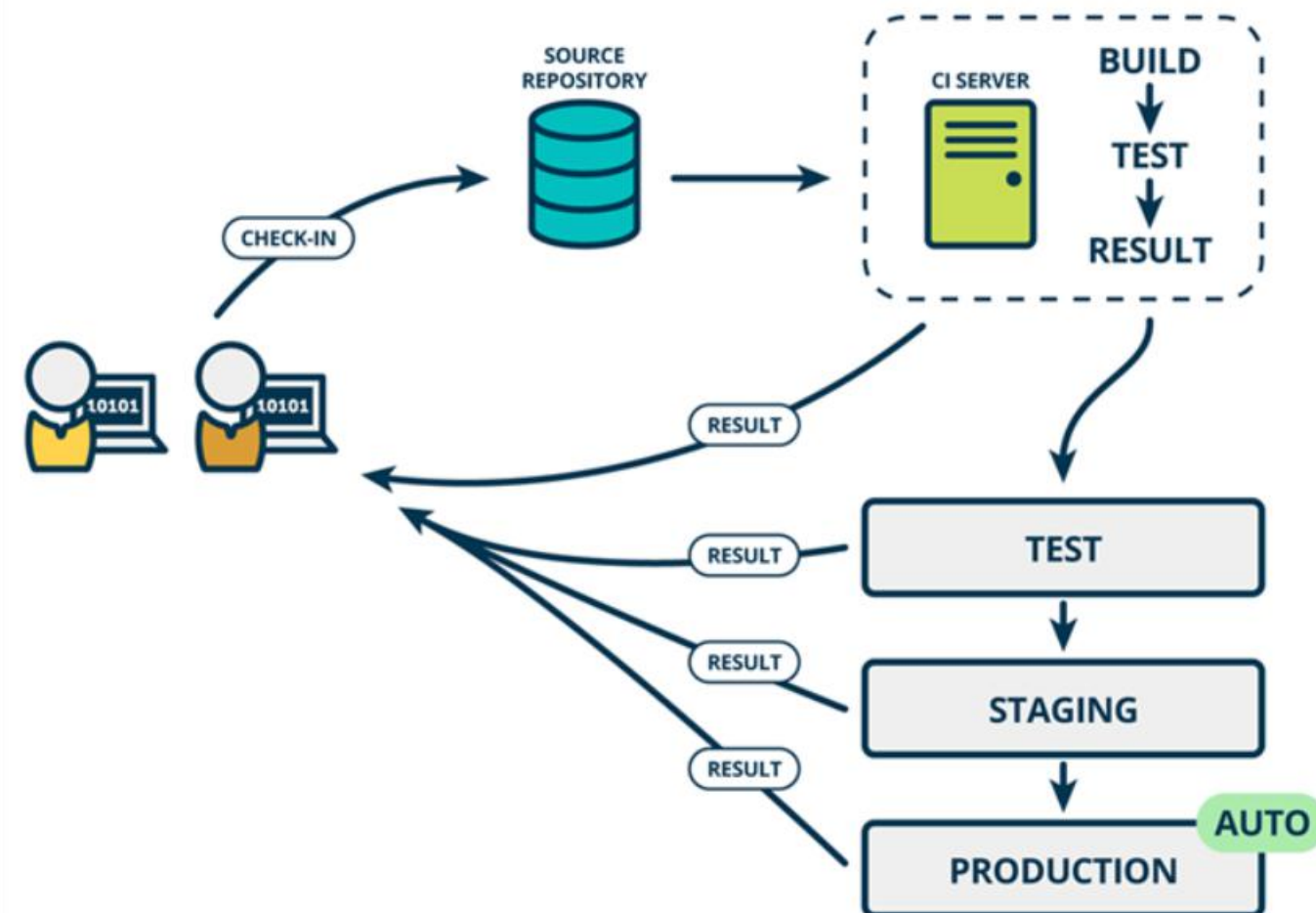
持续部署

□ 持续部署

持续部署（英语：Continuous Deployment，缩写为 CD），是持续交付的下一步，指的是代码通过评审以后，自动部署到生产环境。

有时候，持续部署也与持续交付混淆。持续部署意味着所有的变更都会被自动部署到生产环境中。持续交付意味着所有的变更都可以被部署到生产环境中，但是出于业务考虑，可以选择不部署。如果要实施持续部署，必须先实施持续交付。

持续部署即在持续交付的基础上，把部署到生产环境的过程自动化。



Jenkins简介

□ 什么是Jenkins:

Jenkins is an automation engine with an unparalleled plugin ecosystem to support all of your favorite tools in your delivery pipelines, whether your goal is continuous integration, automated testing, or continuous delivery.

持续集成、自动测试、持续部署的超级引擎，支持自定义工具集、多种交付通道。

Jenkins的缺点:

Jenkins是开发人员开发的，没有照顾到运维人员得感受。CMDB对接是Jenkins得短板。

Jenkins安装

□ 安装:

安装方式有多种，包括rpm、war包启动、dockerfile。
本文为rpm包。

推荐的硬件配置

1 GB的RAM

50 GB的驱动器空间

软件要求

Java 8--无论是Java运行时环境（JRE）还是Java开发工具包（JDK）都可以。

Jenkins安装

□ 安装依赖:

```
yum localinstall jdk-8u144-linux-x64.rpm
```

□ rpm安装

软件下载

官方仓库 <https://pkg.jenkins.io/redhat-stable/>

清华大学开源软件镜像站

<https://mirrors.tuna.tsinghua.edu.cn/jenkins/redhat/>

```
# wget
```

```
https://mirrors.tuna.tsinghua.edu.cn/jenkins/redhat/jenkins-2.206-1.1.noarch.rpm
```

```
# yum localinstall jenkins-2.206-1.1.noarch.rpm
```

Jenkins安装

□ rpm安装内容

# rpm -ql jenkins	
/etc/init.d/jenkins	# 启动文件
/etc/logrotate.d/jenkins	# 日志分割配置文件
/etc/sysconfig/jenkins	# jenkins主配置文件
/usr/lib/jenkins	# 存放war包目录
/usr/lib/jenkins/jenkins.war	# war 包
/usr/sbin/rcjenkins	# 命令
/var/cache/jenkins	# war包解压目录 jenkins网页代码目录
/var/lib/jenkins	# jenkins 工作目录
/var/log/jenkins	# 日志

Jenkins安装

□ 配置文件

注意：修改监听地址，默认本地访问

```
# grep "[a-Z]" /etc/sysconfig/jenkins
JENKINS_HOME="/var/lib/jenkins" #jenkins工作目录
JENKINS_JAVA_CMD=""
JENKINS_USER="jenkins" # jenkins启动用户
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"
JENKINS_PORT="8080" # 端口
JENKINS_LISTEN_ADDRESS="" # 监听地址
JENKINS_HTTPS_PORT=""
JENKINS_HTTPS_KEYSTORE=""
JENKINS_HTTPS_KEYSTORE_PASSWORD=""
JENKINS_HTTPS_LISTEN_ADDRESS=""
JENKINS_DEBUG_LEVEL="5"
JENKINS_ENABLE_ACCESS_LOG="no"
JENKINS_HANDLER_MAX="100" # 最大连接
JENKINS_HANDLER_IDLE="20"
JENKINS_ARGS=""
```


Jenkins安装

□ web界面安装

```
# systemctl start jenkins
```

```
# systemctl enable jenkins
```

访问Jenkins

http://IP:8080

查找admin默认密码

入门

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里?](#)）该文件在服务器上：

```
/var/lib/jenkins/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

继续

Jenkins安装

- web界面安装
显示为离线，解放方案见下页。



Jenkins安装

□ web界面安装

显示为离线，解放方案如下。

1、修改/var/lib/jenkins/updates/default.json

jenkins在下载插件之前会先检查网络连接，其会读取这个文件中的网址。默认是谷歌：修改成baidu

2、修改/var/lib/jenkins/hudson.model.UpdateCenter.xml

改文件为插件地址，默认是通过

<https://updates.jenkins.io/update-center.json>进行更新的。此处，将https改为http，重启Jenkins

Jenkins安装

□ web界面安装 安装推荐插件



Jenkins安装

□ web界面安装

安装推荐插件，由于网络原因，部分插件可能安装不成功，后面可以下载插件使用离线安装。清华Jenkins插件镜像：
<https://mirrors.tuna.tsinghua.edu.cn/jenkins/plugins/>

新手入门

新手入门

✖ SSH Credentials	✖ Credentials Binding	✖ Pipeline: Declarative Extension Points API	✖ JSch dependency	Pipeline: GitHub Groovy Libraries
✖ Git client	✖ GIT server	✖ Pipeline: Shared Groovy Libraries	✖ Docker Commons	** MapDB API
✖ Docker Pipeline	✖ Pipeline: Declarative Agent API	✖ Pipeline: Declarative	✖ Pipeline	Subversion
✖ Git	🔄 GitHub	✖ GitHub Branch Source	✖ Pipeline: GitHub Groovy Libraries	SSH Slaves
✖ Subversion	✖ SSH Slaves			** Matrix Authorization Strategy

SSH Credentials
Credentials Binding
Pipeline: Declarative Extension Points API
JSch dependency
Git client
GIT server
Pipeline: Shared Groovy Libraries
Docker Commons
Docker Pipeline
Pipeline: Declarative Agent API
Pipeline: Declarative
Pipeline
Git
GitHub
** - 需要依赖

Jenkins 2.206

Jenkins安装

- web界面安装
- 创建管理员用户

新手入门

创建第一个管理员用户

用户名:

admin

密码:

.....

确认密码:

.....

全名:

admin

电子邮件地址:

admin@123.com

Jenkins 2.206

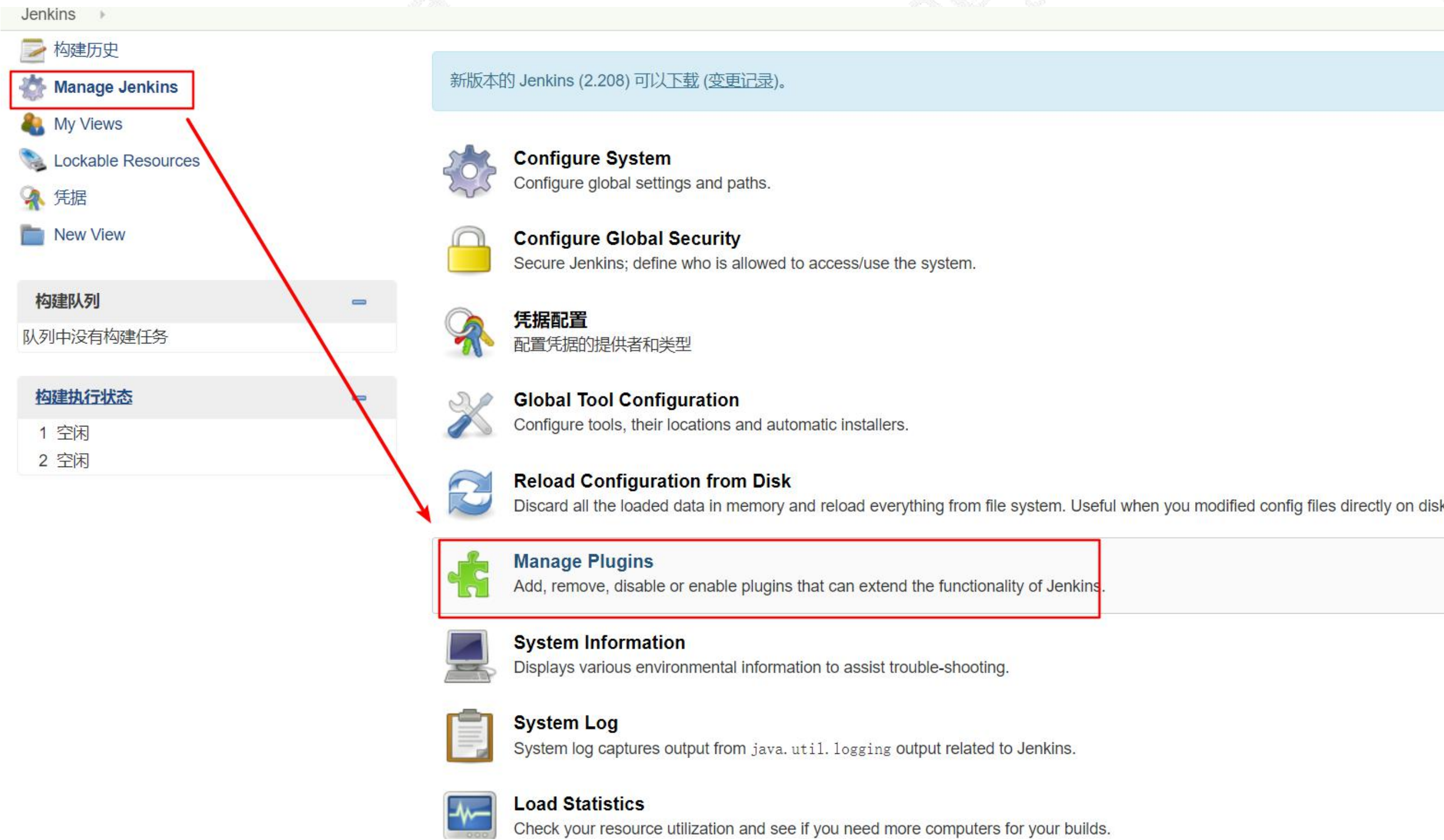
使用admin账户继续

保存并完成

Jenkins安装

□ 插件安装

可以在线安装或离线安装， 离线需要事先下载。



Jenkins安装

□ 插件安装--在线安装
示例：在线安装gitlab



Jenkins安装

- ❑ 插件安装--离线安装
- 示例： 离线安装git

Jenkins ▸ 插件管理

用户名

密码

不通过代理的主机

高级...

提交

上传插件

您可以通过上传一个.hpi文件来安装插件。

文件:

选择文件

 git.hpi

上传

git.hpi

Maven安装

□ 安装

Maven的作用：Maven负责java语言的编译和打包。
换个角度来说，Maven相当于Java中的make命令。
同类型的工具有Ant，但是Ant不提供解决依赖关系的功能

官方地址：

<http://maven.apache.org/>

Maven安装

□ 安装

获取软件包:

```
# wget -c
```

```
https://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz
```

解压tar包

```
# tar xf apache-maven-3.6.2-bin.tar.gz -C /bin/
```

```
# ln -sv /bin/apache-maven-3.6.2/ /bin/maven
```

设置环境变量

```
# vim /etc/profile.d/maven.sh
```

```
export M2_HOME=/bin/maven
```

```
export MAVEN_HOME=/bin/maven
```

```
PATH=$M2_HOME/bin:$PATH
```

测试: # source /etc/profile.d/maven.sh

```
# mvn -v
```

Jenkins配置

□ Jenkins的功能


- 1.通过git命令从Gitlab中拉取源码
- 2.让maven进行编译并打包
- 3.把打好的包发送到目标主机
- 4.执行部署脚本。

□ Jenkins目录

/var/lib/jenkins 主目录
/etc/init.d/jenkins 启动文件
/var/cache/Jenkins 程序文件
/var/log/Jenkins 日志文件

Jenkins配置

配置jenkins并发执行数量，提高执行效率

**Jenkins**

Jenkins > 配置

新建Item

用户列表

构建历史

Manage Jenkins

My Views

Lockable Resources

凭据

New View

构建队列

队列中没有构建任务

主目录

系统消息

执行者数量

标签

用法

/var/lib/jenkins

[Plain text] 预览

5

Use this node as much as possible

Jenkins配置

□ 设置邮件，能够在测试完成后，主动发邮件告知测试情况

1、安装Email Extension Plugin插件



Jenkins配置

- 设置邮件，能够在测试完成后，主动发邮件告知测试情况
- ## 2、设置Extended E-mail Notification

Extended E-mail Notification

SMTP server

smtp.163.com

邮件服务器

Default user E-mail suffix

163.com

邮箱后缀

☒ Use SMTP Authentication

User Name

esen_monitor@163.com

发件人

Password

.....

邮箱授权密码

Advanced Email Properties

Use SSL

☒

SMTP port

465

加密端口

Charset

UTF-8

Additional accounts

增加

Default Content Type

HTML (text/html)

邮件内容格式

☐ Use List-ID Email Header

☐ Add 'Precedence: bulk' Email Header

Default Recipients

esen_monitor@163.com,esen_monitor@tom.com

默认接收人列表

Buttons

保存应用

Extended E-mail Notification

SMTP server

smtp.163.com

邮件服务器

Default user E-mail suffix

163.com

邮箱后缀

☒ Use SMTP Authentication

User Name

esen_monitor@163.com

发件人

Password

.....

邮箱授权密码

Advanced Email Properties

Use SSL

☒

SMTP port

465

加密端口

Charset

UTF-8

Additional accounts

增加

Default Content Type

HTML (text/html)

邮件内容格式

☐ Use List-ID Email Header

☐ Add 'Precedence: bulk' Email Header

Default Recipients

esen_monitor@163.com,esen_monitor@tom.com

默认接收人列表

Buttons

保存应用

Jenkins配置

□ 设置邮件，能够在测试完成后，主动发邮件告知测试情况

3、设置邮件通知

在"Jenkins Location"设置系统管理员地址（重要：不能省略！）

Jenkins Location

Jenkins URL	<input type="text" value="http://192.168.95.12:8080/"/>
系统管理员邮件地址	<input type="text" value="esen_monitor@163.com"/>

邮件通知

SMTP服务器	<input type="text" value="smtp.163.com"/>
用户默认邮件后缀	<input type="text" value="@163.com"/>
<input checked="" type="checkbox"/> 使用SMTP认证	
用户名	<input type="text" value="esen_monitor@163.com"/>
密码	<input type="password" value="*****"/>
<input checked="" type="checkbox"/> 使用SSL协议	
SMTP端口	<input type="text" value="465"/>
Reply-To Address	<input type="text"/>
字符集	<input type="text" value="UTF-8"/>
<input checked="" type="checkbox"/> 通过发送测试邮件测试配置	
Test e-mail recipient	<input type="text" value="esen_monitor@tom.com"/>

Email was successfully sent

Test configuration



Jenkins配置

□ 安装常用插件

需要添加的插件：Gitlab Hook、Build Authorization Token
Root、Gitlab Authentication、Gitlab
Gitlab Plugin

#安装之后才可以在系统配置中指定gitlab的IP地址

Git Plugin

Git Client Plugin

#用于jenkins在gitlab中拉取源码

Publish Over SSH

#用于通过ssh部署应用

Maven Integration plugin

#用于新建maven项目

注意：安装了插件之后，在全局配置才能详细配置。

Jenkins配置

安装常用插件

JenkinsUpdate Center

返回工作台

管理 Jenkins

插件管理

安装/更新 插件中

准备

build-token-root

准备

Infrastructure plugin for Publish Over X

准备

publish-over-ssh

WMI Windows Agents

ruby-runtime

准备

gitlab-hook

Gitlab Authentication

Gitlab API

Loading plugin extensions

Javadoc

Maven Integration

Loading plugin extensions

准备

gitlab-plugin

准备

git

准备

git-client

完成

Checking internet connectivity

Checking update center connectivity

Success

完成

完成

完成

安装中

等待

等待

等待

Pending

等待

等待

Pending

等待

等待

等待

等待

Jenkins配置

指定Maven、Git、JDK的安装路径

Jenkins

构建执行状态

1 空闲

2 空闲

3 空闲

4 空闲

5 空闲

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

凭据配置
配置凭据的提供者和类型

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Usefu

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
可更新

JDK

JDK 安装

新增 JDK

JDK

别名

jdk-1.8

JAVA_HOME

/usr/java/jdk1.8.0_144/

☐ Install automatically

Git

Git installations

Git

Name

git-1.8.3.1

Path to Git executable

/usr/bin/git

☐ Install automatically

Maven

Maven 安装

新增 Maven

Maven

Name

maven-3.6.2

MAVEN_HOME

/usr/bin/maven

☐ Install automatically

持续集成示例

□ 1. Jenkins机器设置

```
# git config --global user.name tom
# git config --global user.email tom@126.com
# ssh-keygen -C "tom@126.com" -f ~/.ssh/id_rsa -P "" -q
```

□ 2. Gitlab服务器添加ssh key

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDdkbrYsfP+Z15Ct77BbKq2lobUSqbu/P4toZQc4h1sSr6/Y
Xpqkmls2/8gozYcGGRA3FyMCgdNF33QTfYsY65mXJoTJeXjEt/QypRGXMqN+nHDXylBGfgBeyPWN
kwb8Gtp+inzmBeAbP9Zqky8Z94AuwlyAP03y8jffXJVgiSvrHs9lbXvYn/+ +zSy3FyCZ8hiw3TDozlZaNS
e1Tg053N0YNfCURONbdoJlPaFX0qw0yvs1gqtM8PfNDpf4AnO0q4kH6SxTVmt3vOSGsgag3XFFY5d
ulKWxXOYw4QYCnDWExuLnDeLu9xaFXPzD6n3DQ8cNIMQx/X/rfS9zj6cEmUv tom@126.com
```

Title

Name your individual key via a title

Add key

持续集成示例

❑ 3. Jenkins机器测试

```
# git clone git@192.168.150.16:dev/app1.git
```


❑ 4. 进入主页--创建一个新任务

输入项目名称---构建一个自由风格的软件项目---确认

输入一个任务名称

auto-deploy

» 必填项

 **Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

General--填写项目描述

General

源码管理

构建触发器

构建环境

构建

构建后操作

描述

测试

持续集成示例

- 4. 进入主页--创建一个新任务
- 源码管理--安装git插件--授权认证
- Credentials—选择root，后面添加的凭据，设计源码库浏览器
- gitlab

General

源码管理

构建触发器

构建环境

构建

构建后操作

无

Git

Repositories

Repository URL

git@192.168.150.16:dev/app1.git

Credentials

root

添加

高级...

Add Repository

Branches to build

指定分支 (为空代表any)

*/master

增加分支

源码库浏览器

gitlab

URL

http://192.168.150.11/

Version

12.5

持续集成示例

- 4. 进入主页--创建一个新任务
源码管理--安装git插件--授权认证
添加凭据

添加凭据

Domain

全局凭据 (unrestricted)

类型

SSH Username with private key

范围

全局 (Jenkins, nodes, items, all child items, etc)

ID

描述

Username

root

Private Key

Enter directly

Key

M6neHdpK8XPzcYRXhx2rKm3Y9f4NiXYztGNgWKPpZ2LkwNMzfLV14oh9Kq3BkpHT
1x9NY21XUA3mbVeCcFHsJgiNMnm7KnC1BLNO/kCug5bDer8MqFfmsVw=
-----END RSA PRIVATE KEY-----

Enter New Secret Below

Passphrase

添加

取消

持续集成示例

□ 5. 构建测试 点击控制台输出查看



Jenkins备份恢复

- 进行有效的Jenkins数据备份，首先要理解Jenkins的数据存储结构，然后根据业务场景选择合适的粒度进行备份

典型的jenkins实例包含以下文件和目录：

*.xml	需要备份
config-history	需要备份
fingerprints	需要备份
global-build-stats	需要备份
.key	需要备份
jobs	jobs配置需要备份（config.xml, nextBuildNumber），builds目录（build logs等）根据需求而定
logs	插件logs，根据需求而定，可以不备份
monitoring	可以不备份，插件会实时生成监控数据
nodes	需要备份
plugins	需要备份 *.jpi及 *.hpi，可以不备份每个插件子目录，jenkins启动后会更新插件子目录
secrets	需要备份
updates	需要备份
userContent	用户上传内容，可以根据需要备份
users	用户缓存信息，最好备份

Jenkins备份恢复

□ 使用插件备份

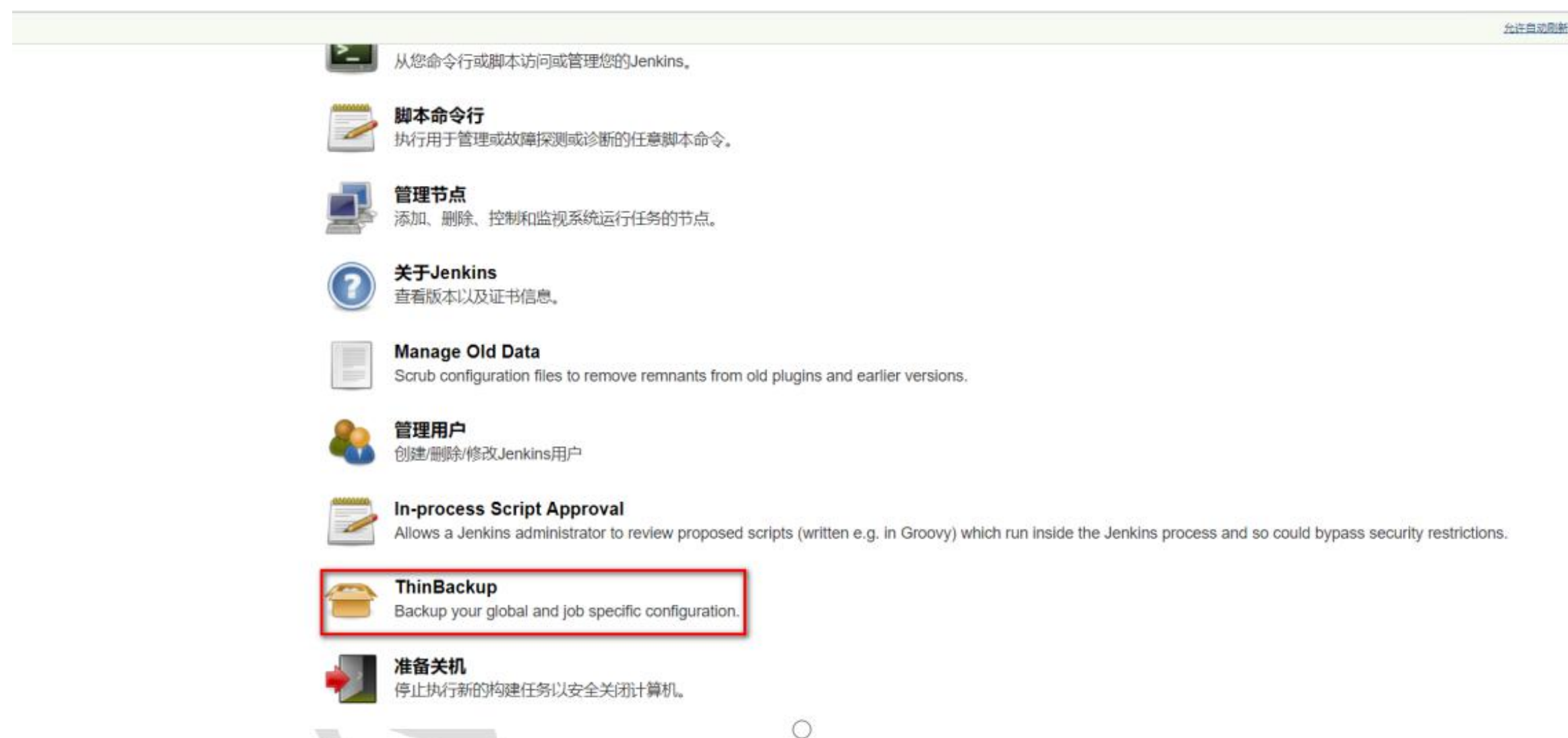
系统管理-管理插件-找到ThinBackup并安装

创建备份目录

```
# mkdir /backup
```

```
# chown -R jenkins.jenkins /backup
```

系统管理-找到ThinBackup-点击Setting进行设置
设置备份,周一到周五的凌晨1点进行完整备份



Jenkins备份恢复

□ 使用插件备份

系统管理-管理插件-找到ThinBackup并安装

thinBackup Configuration

Backup settings

Backup directory: E:\ProgramFiles\Jenkins\backup

Backup schedule for full backups:

Backup schedule for differential backups:

Max number of backup sets: -1

Files excluded from backup (regular expression):

☒ Wait until Jenkins/Hudson is idle to perform a backup

Force Jenkins to quiet mode after specified minutes: 120

☒ Backup build results

☐ Backup build archive

☐ Backup only builds marked to keep

☐ Backup 'userContent' folder

☐ Backup next build number file

☐ Clean up differential backups

☐ Move old backups to ZIP files

Save

thinBackup Configuration

Backup settings

Backup directory: /backup

Backup schedule for full backups: 0 1 * * 1-5

Backup schedule for differential backups:

Max number of backup sets: -1

Files excluded from backup (regular expression):

☒ Wait until Jenkins/Hudson is idle to perform a backup

Force Jenkins to quiet mode after specified minutes: 120

☒ Backup build results

常用的定时构建举例

每隔5分钟构建一次

H/5 * * * *

每两小时构建一次

H H/2 * * *

每天中午12点定时构建一次

0 12 * * *

每天下午6点下班前定时构建一次

0 18 * * *

工作日（周一到周五）早上9点定时构建一次

0 9 * * 1-5

代码上线方案

□ 早期手动部署代码

纯手动scp上传代码。

纯手动登陆，Git pull 或者SVN update。

纯手动xftp上传代码。

开发发送压缩包，rz上传，解压部署代码。

缺点：

全程运维参与，占用大量时间。

如果节点多，上线速度慢。

人为失误多，目录管理混乱。

回滚不及时，或者难以回退。

代码上线方案

□ 合理化上线方案

1、开发人员需在个人电脑搭建LAMP环境测试开发好的网站代码，并且在办公室或 IDC机房的测试环境测试通过，最好有专职测试人员。

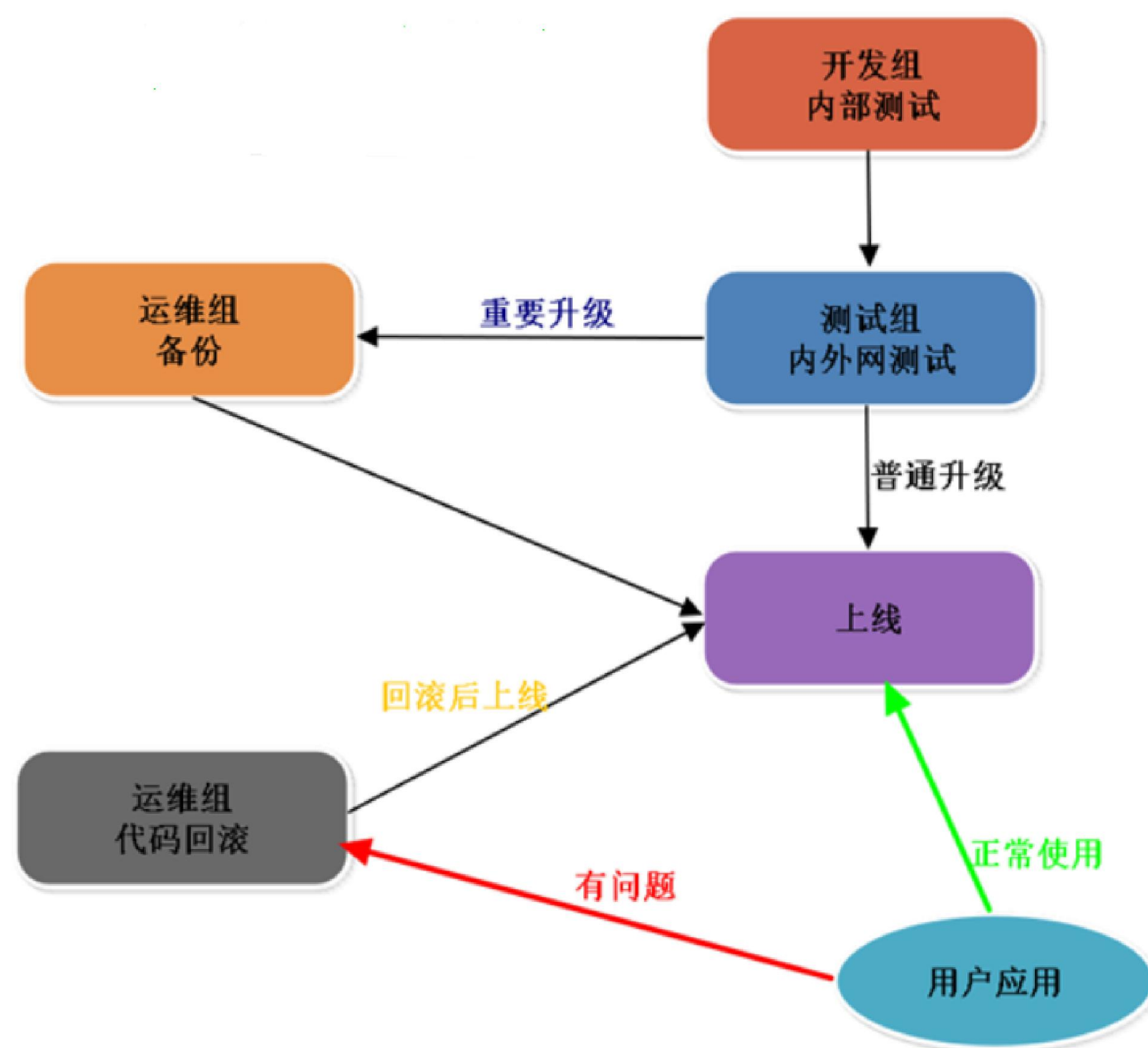
2、程序代码上线要规定时间，例如：三天上线一次，如网站需经常更新可每天下午 17点上线，这个看网站业务性质而定，原则就是影响用户体验最小。

3、代码上线之前需备份，网站程序出了问题方便回退，另外，从上线技巧上讲，上传代码时尽可能先传到服务器网站临时目录，传完整后一步mv过去，或者通过ln做软链接— 线上更新代码的思路。如果严格更新，把应用服务器从集群节点平滑下线，然后更新。

4、尽量由运维人员管理上线，对于代码的功能性，开发人员更在意，而对于代码的性能优化和上线后服务器的稳定，运维更在意服务器的稳定，因此，如果网站宕机问题归运维管，就要让运维上线，这样更规范科学。否则，开发随意更新，出了问题运维负责，这样就错了，运维永远无法抬头。

代码上线方案

□ 合理化上线方案



总结

- 持续集成简介
- Jenkins简介
- Jenkins配置
- Jenkins使用



谢谢观看

更多好课，请关注[万门大学APP](#)

