

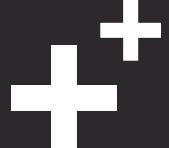
# Python开发

NSD DEVOPS

DAY05

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	Jenkins基础
	10:30 ~ 11:20	准备git仓库
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	应用jenkins
	15:00 ~ 15:50	管理应用服务器
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



# Jenkins基础

Jenkins基础

Jenkins简介

Jenkins概述

持续集成

Jenkins特点

安装Jenkins

下载Jenkins

安装Jenkins

初始化Jenkins

安装插件

管理用户

完成安装

修改管理员密码

CI/CD流程

# Jenkins简介

# Jenkins概述

- Jenkins是由java编写的一款开源软件
- 作为一款非常流行的CI（持续集成）工作，用于构建和测试各种项目
- Jenkins 的主要功能是监视重复工作的执行，例如软件工程的构建或在 cron下设置的 jobs



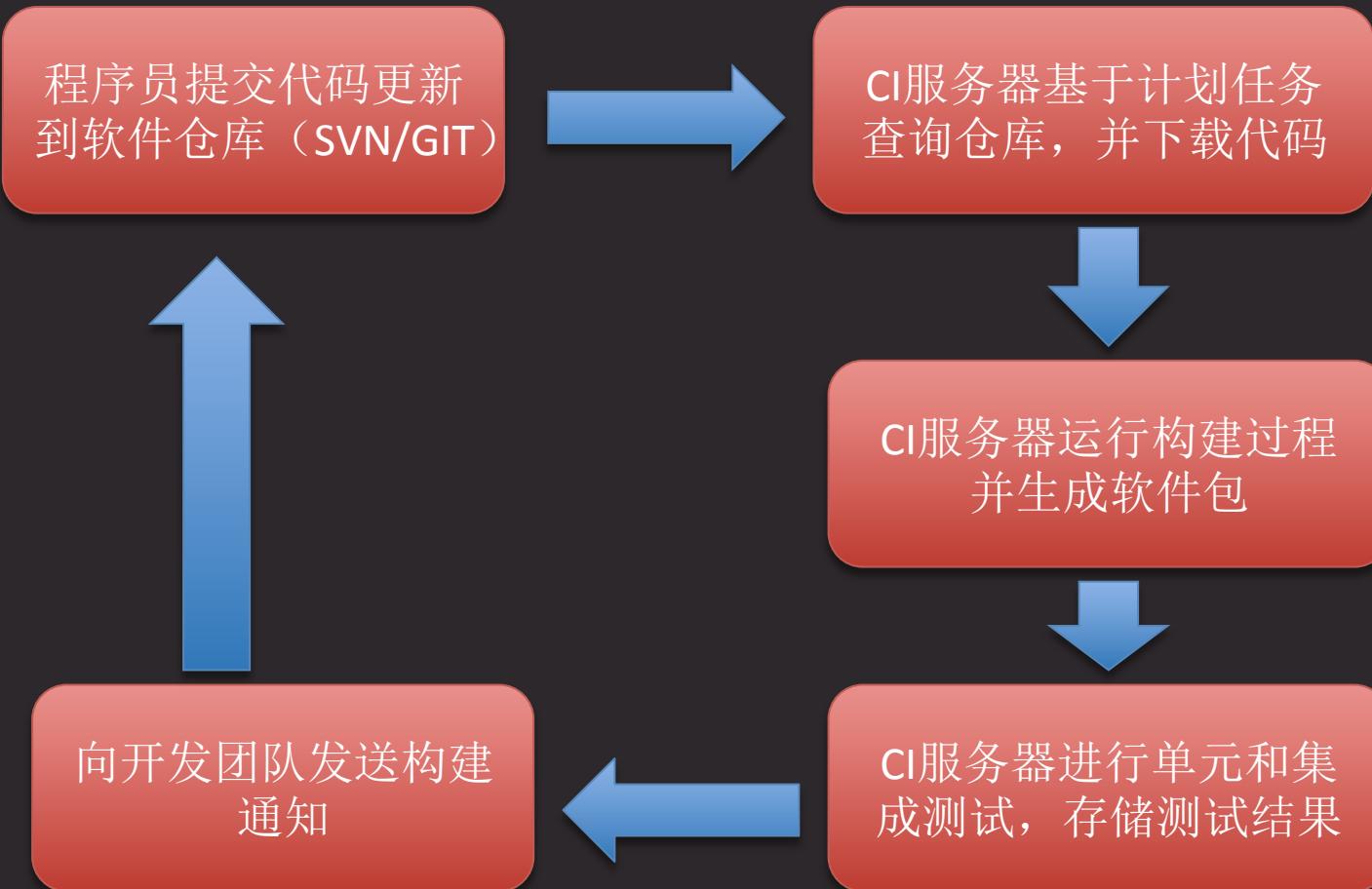
# 持续集成

- 持续集成（CI）是当下最为流行的应用程序开发实践方式
- 程序员在代码仓库中集成了修复bug、新特性开发或是功能革新
- CI工具通过自动构建和自动测试来验证结果。这可以检测到当前程序代码的问题，迅速提供反馈



# 持续集成（续1）

知识讲解



# Jenkins特点

- 简单、可扩展、用户界面友好
- 支持各种SCM（软件配置管理）工具，如SVN、GIT、CVS等
- 能够构建各种风格的项目
- 可以选择安装多种插件
- 跨平台，几乎可以支持所有的平台



# 安装Jenkins



# 下载Jenkins

The screenshot shows the Jenkins official website at jenkins.io. The header includes navigation links for Blog, Documentation, Plugins, Use-cases, Participate, Sub-projects, Resources, About, and a prominent Download button. The main content features the Jenkins mascot (a smiling man in a tuxedo) on the left and a large "Jenkins" title with a subtitle "Build great things at any scale". Below this is a description of Jenkins as "The leading open source automation server" that provides many plugins for project management. At the bottom, there's a "Say ‘hello’ to Blue Ocean" section and a preview of the Blue Ocean user interface.

Jenkins

Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

Documentation   Download

Say “hello” to Blue Ocean

Blue Ocean is a new user experience that puts

...  
✓ dropdown / whimsy #5  
Pipeline   Changes   Tests   Artifacts  
Branch: Jenkinsfile   ① 12s   No changes



# 安装Jenkins

- 安装Jenkins

```
[root@localhost 下载]# rpm -ivh jenkins-2.121-1.1.noarch.rpm
```

警告: jenkins-2.121-1.1.noarch.rpm: 头V4 DSA/SHA1 Signature, 密钥 ID d50582e6: NOKEY

准备中...

##### [100%]

正在升级/安装...

1:jenkins-2.121-1.1

##### [100%]

- 启动服务

```
[root@localhost 下载]# systemctl start jenkins
```

```
[root@localhost 下载]# systemctl enable jenkins
```



# 初始化Jenkins

- Jenkins默认运行在8080端口



# 安装插件

- 如果网速较快，选择推荐插件，否则选择自定义，只选中GIT即可



# 管理用户

- 可以直接使用Admin登陆

新手入门

## 创建第一个管理员用户

用户名:

密码:

确认密码:

全名:

电子邮件地址:

Jenkins 2.121

[使用admin账户继续](#)

[保存并完成](#)



# 完成安装

- 安装成功如下图所示：



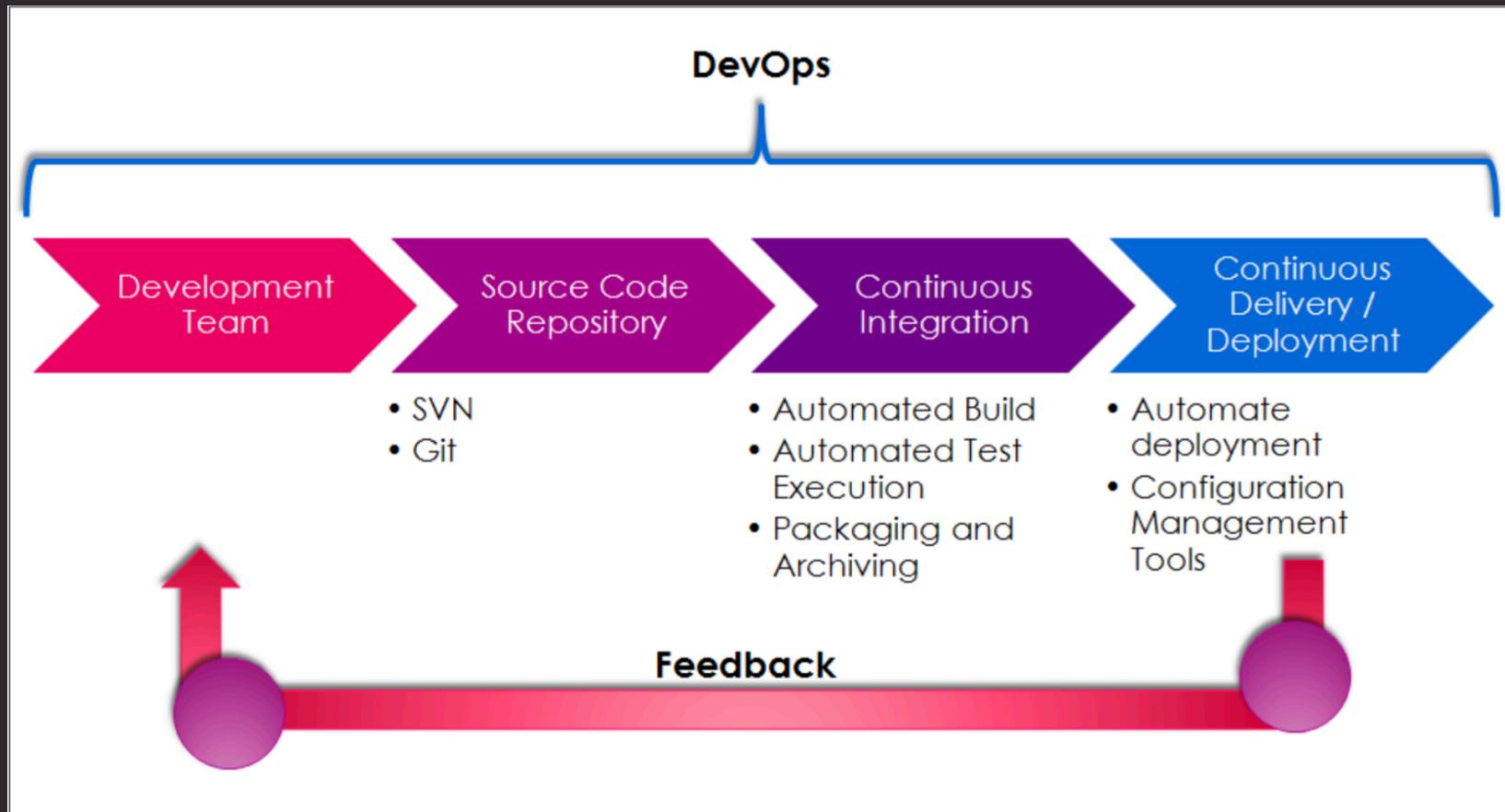
# 修改管理员密码

- 修改管理员密码 : admin->设置



# CI/CD流程

- 程序开发的生命周期内，需要程序员和运维共同协作



# 案例1：安装Jenkins

1. 运行虚拟机，将第一块网络的连接方式改为NAT
2. 安装Jenkins
3. 初始化Jenkins



# 准备git仓库

准备git仓库

本地仓库

初始化wordpress项目

tag标签

升级wordpress

更新git仓库

添加标签

管理标签

远程仓库

创建群组

创建wordpress项目

创建用户

上传wordpress代码

在网页中查看项目

# 本地仓库



# 初始化wordpress项目

- 解压wordpress 4.8版本

```
[root@localhost ~]# unzip wordpress-4.8-zh_CN.zip
```

- 初始化git仓库

```
[root@localhost ~]# cd wordpress/  
[root@localhost wordpress]# git init  
[root@localhost wordpress]# git add .
```

```
[root@localhost wordpress]# git commit -m "wordpress init"
```

```
[root@localhost wordpress]# git status
```

# 位于分支 master

无文件要提交，干净的工作区



# tag标签

- 如果达到一个重要的阶段，并希望永远记住那个特别的提交快照，可以使用 git tag 给它打上标签
- 将初始化完毕的wordpress打标签v1.0

```
[root@localhost wordpress]# git tag v1.0
```



# 升级wordpress

- 将wordpress新版本解压到项目中

```
[root@localhost wordpress]# cd ..  
[root@localhost ~]# unzip wordpress-4.9-zh_CN.zip  
Archive: wordpress-4.9-zh_CN.zip  
replace wordpress/wp-mail.php? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
```



# 更新git仓库

- 检查git状态

```
[root@localhost ~]# cd wordpress/  
[root@localhost wordpress]# git status
```

- 将更新/增加的文件确认至仓库

```
[root@localhost wordpress]# git add .  
[root@localhost wordpress]# git commit -m "upgrade to new version"
```



# 添加标签

- 确认所有项目已提交

```
[root@localhost wordpress]# git status
```

# 位于分支 master

无文件要提交，干净的工作区

- 为升级后的wordpress项目增加v2.0标签

```
[root@localhost wordpress]# git tag v2.0
```



# 管理标签

- 查看所有标签

```
[root@localhost wordpress]# git tag
```

```
v1.0
```

```
v2.0
```

- 切换至v1.0标签

```
[root@localhost wordpress]# git checkout v1.0
```

```
Note: checking out 'v1.0'.
```

- 切换到最新状态

```
[root@localhost wordpress]# git checkout master
```

之前的 HEAD 位置是 50984b1... wordpress init

切换到分支 'master'



## 案例2：设置本地仓库

1. 解压wordpress4.8
2. 将解压目录配置为git仓库
3. 为当前代码标记为v1.0
4. 更新wordpress到4.9
5. 将代码标记为v2.0



# 远程仓库

# 创建群组

- 创建名为devops的群组



# 创建群组（续1）

- 创建名为devops的群组

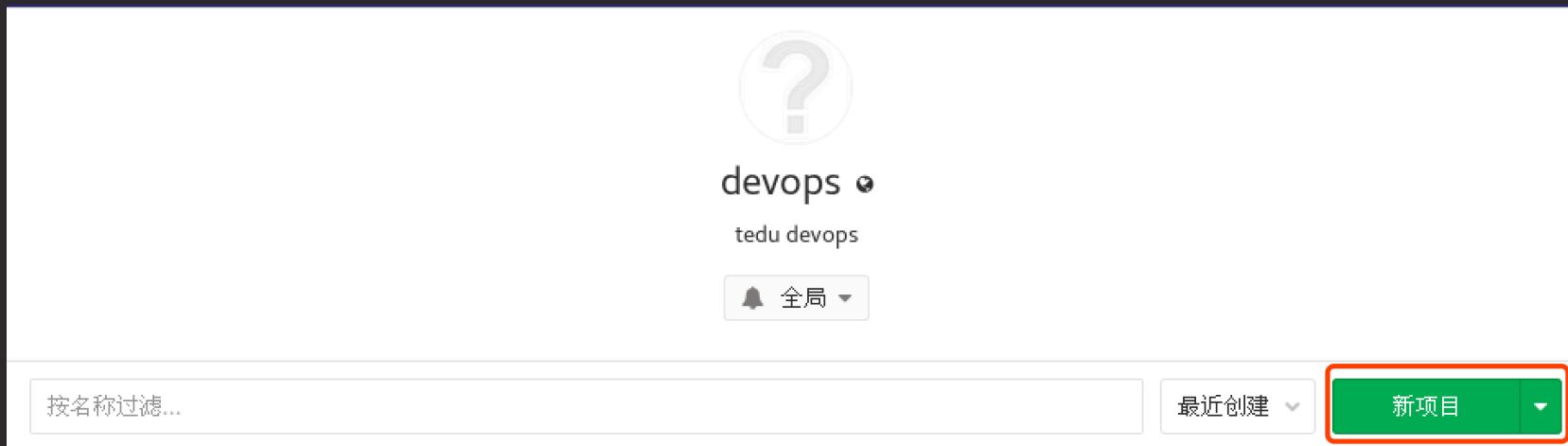
知识讲解

+

群组路径	http://192.168.113.139/	devops
群组名称	devops	
描述	tedu devops	
群组图标	Choose File ...	No file chosen The maximum file size allowed is 200KB.
可见等级	<input type="radio"/>  私有 该群组和其项目只有其成员能以看到。	
	<input type="radio"/>  内部 该群组和其内部项目只有已登录用户能看到。	
	<input checked="" type="radio"/>  公开 该群组和其公开项目可以被任何授权的用户看到。	

# 创建wordpress项目

- 创建名为wordpress的项目



# 创建wordpress项目（续1）

- 创建名为wordpress的项目

空白项目 从模板创建 导入项目

项目路径

希望将几个相关联的项目放置于同一个命名空间下？[创建群组](#)

项目名称

项目描述 (可选)

wordpress project for jinkens

可见等级

 私有  
项目访问权限必须明确授权给每个用户。

 内部  
该项目允许已登录的用户访问。

 公开  
该项目允许任何人访问。

[创建项目](#) [取消](#)



# 创建用户

- 为WordPress项目创建主程序员用户

The screenshot shows the GitLab dashboard with several key components:

- Top Bar:** Includes links for "GitLab", "项目", "群组", "更多", a user icon (highlighted with a red box), a plus sign for creating new items, a search bar, and navigation icons.
- Dashboard Summary:** Shows "Projects: 1", "Users: 1", and "Groups: 1".
- Project Card:** Contains a "New project" button.
- User Card:** Contains a "New user" button, which is highlighted with a red box.
- Group Card:** Contains a "New group" button.
- Metrics (统计):** Shows "派生项目数" (0) and "问题数" (0).
- Features (特性):** Shows "注册" (enabled) and "LDAP" (disabled).
- Components (组件):** Shows "GitLab" (version 10.5.4), "GitLab Shell" (version 6.0.3), and a red button labeled "update asap".



# 创建用户（续1）

- 为WordPress项目创建主程序员用户

管理区域 > 新用户

## 新用户

### 账号

姓名  \* 必须填写

用户名  \* 必须填写

电子邮箱  \* 必须填写

# 创建用户（续2）

- 为wordpress项目创建主程序员用户

The screenshot shows a user management interface with a sidebar on the left containing icons for help, teams, projects, members, and settings. A search bar at the top has the text 'devops' with a '详细信息' (Detailed Information) link. The main area displays a user profile for 'devops' with a question mark icon, the name 'devops', the organization 'tedu devops', and a notification bell labeled '全局' (Global). Below the profile, there is a search bar with the placeholder '按名称过滤...' (Filter by name...), a dropdown menu set to '名称' (Name), and a green button labeled '新项目' (New Project). At the bottom, a project card for 'wordpress' is shown, which is a 'wordpress project for jinkens'. The card includes a star rating of 0, a comment icon, and the text '10 minutes ago'.

# 创建用户（续3）

- 为WordPress项目创建主程序员用户

devops > 成员

## 成员

添加成员到 devops

Zhangzhg \*

搜索已存在的成员或者使用他们的邮箱地址邀请。

主程序员

访问到期日期

点击这里 了解更多关于角色权限的介绍。

到此日期，该成员将自动失去对此群组及其所有项目的访问权限。

添加到群组

通过名字查找已存在的成员

名称, 升序排列

The screenshot shows a user interface for managing group members. At the top left, it says 'devops > 成员'. Below that is a section titled '成员' (Members). A search bar contains the text 'Zhangzhg \*'. To its right is a dropdown menu set to '主程序员' (Programmer-in-Charge). Further right is a field for '访问到期日期' (Access expiration date) with a note about automatic member removal if reached. A large green button on the far right says '添加到群组' (Add to group). Below these controls are two smaller input fields: one for searching existing members by name and another for sorting by name in ascending order.

# 创建用户（续4）

- 用户生成ssh密钥

```
[root@localhost ~]# ssh-keygen -t rsa -C "zhangzg@tedu.cn" -b 4096
[root@localhost ~]# cat ~/.ssh/id_rsa.pub
ssh-rsa
AAAAAB3NzaC1yc2EAAAQABAAQACQC4iidWslzdFWQM3mvbFCC5SL
RSqXnoT2wFodo0FkdbbcSOeJ1RX6CgZfhW+PTDjsu7OfiCw
+ZOSIaeY0xQWcl1ExVn2aDMNKr7Lr2VyHEpo7cJZoGIOc6vQBN83VZKcY
dJzEbaxsHbRg2MKHN85cdxVWAQOqaHs105thHBCI3Um6+sAvhAt9Use
QQOQYBIIHO02QJ .....
```



# 创建用户（续4）

- 新用户登陆重置密码，然后设置ssh密钥

The screenshot shows the 'Add SSH Key' page on the GitLab interface. On the left, there's a sidebar with various icons. A red box highlights the 'key' icon, which is currently selected. The main content area has a title '增加 SSH 密钥' and a note: '在增加 SSH 密钥之前需要先 [生成密钥](#)。' Below this is a '密钥' section containing a large block of text representing an SSH public key. At the bottom of this section is a '增加密钥' button. To the right of the main content is a vertical sidebar with a user profile for 'Zhangzhg @zhangzg'. This sidebar includes links for '个人资料', '设置' (which is highlighted with a red box), '帮助', and '退出'.



# 上传wordpress代码

- 因为本地wordpress已经是git版本库了，所以采用以下方式进行上传：

```
[root@localhost ~]# cd wordpress/  
[root@localhost wordpress]# git remote rename origin old-origin  
error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'  
上述错误可忽略  
[root@localhost wordpress]# git remote add origin  
git@192.168.113.139:devops/wordpress.git  
[root@localhost wordpress]# git push -u origin --all  
[root@localhost wordpress]# git push -u origin --tags
```



# 在网页中查看项目

- 所有的tag可以通过<http://192.168.113.139/devops/wordpress/tags>访问

The screenshot shows a web browser window displaying the GitLab interface. The URL in the address bar is <http://192.168.113.139/devops/wordpress/tags>. The page title is "在网页中查看项目". The left sidebar has a "tags" section selected. The main content area shows a list of tags for a repository named "wordpress". There are two tags listed: "v2.0" and "v1.0". Each tag entry includes a small icon, the tag name, and a timestamp indicating when it was created.

## 案例3：创建远程仓库

1. 在gitlab的devops群组下创建wordpress项目
2. 通过devops的主程序员用户将代码上传至wordpress项目
3. 通过web查看项目



# 应用jenkins

应用jenkins

构建项目

下载git插件

创建自由风格项目

设置参数

源码管理

构建工程

检验结果

查看本地结果

分发服务器管理

优化构建工程

配置分发服务器

修改工程构建工程

构建测试

创建版本文件

创建live\_version文件

执行构建版本工程

# 构建项目

# 下载git插件

- 为了使得Jenkins可以使用git的tag，需要下载git parameter插件
- 点击“系统管理”->“管理插件”



# 创建自由风格项目

- 新建任务

The screenshot shows the Jenkins dashboard with a dark theme. At the top left is the Jenkins logo and the word "Jenkins". On the left side, there is a vertical sidebar with the following items:

- 新建任务** (New Item) - This item is highlighted with a red border.
- 用户 (User)
- 构建历史 (Build History)
- 系统管理 (System Management)
- 我的视图 (My Views)
- Credentials
- 新建视图 (New View)

The main content area features a large "欢迎使用 Jenkins!" (Welcome to Jenkins!) message. Below it is a teal-colored button with the text "开始创建一个新任务." (Start creating a new item).



# 创建自由风格项目（续1）

- 选择自由风格

输入一个任务名称

wpbuild

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.

GitHub Organization

选择一个GitHub organization (or user account) for all repositories matching some defined markers.

确定

Multibranch Pipeline



# 设置参数

- 添加Git Parameter参数

The screenshot shows the Jenkins 'General' configuration page. At the top, there are tabs for General, Source Management, Build Triggers, Build Environment, Build, and Post-build Actions. The General tab is selected. Below the tabs, there is a 'Preview' section with a 'Plain Text' link. Under the preview, several checkboxes are listed: 'GitHub project', 'Throttle builds', 'Discard old builds', and 'Parameterized build process'. The fourth checkbox, 'Parameterized build process', is checked and highlighted with a red box. Below these checkboxes is a 'Add Parameter' dropdown menu. The menu has a 'Credentials Parameter' item, which is also highlighted with a red box. Other items in the dropdown include 'Git Parameter', 'List Subversion tags (and more)', 'Character parameter', and 'Password parameter'. A 'Advanced...' button is located at the bottom right of the dropdown menu.



# 设置参数（续1）

- 添加Git Parameter参数

The screenshot shows the Jenkins 'General' configuration page for a job. The 'Git Parameter' section is highlighted. A parameter named 'wptag' has been defined with a description 'wordpress tag' and a type set to 'Branch or Tag'. The 'Parameter Type' dropdown is circled in red.

**General** 源码管理 构建触发器 构建环境 构建 构建后操作

参数化构建过程

**Git Parameter**

Name	wptag
Description	wordpress tag
Parameter Type	Branch or Tag

**高级...**

**添加参数 ▾**

关闭构建

在必要的时候并发构建

**高级...**

# 源码管理

- 源码采用git

The screenshot shows the 'Source Management' tab of a Jenkins job configuration. The 'Git' option is selected. The 'Repository URL' field contains the value `http://192.168.113.135/devops/wordpress.git`, which is highlighted with a red box. The 'Branch Specifier' field contains the value `$(wptag)`, also highlighted with a red box. Other visible fields include 'Credentials' set to '- none -', an 'Add' button, a 'Advanced...' button, an 'Add Repository' button, and a 'Branches to build' section with an 'Add Branch' button. At the bottom, there are 'Additional Behaviours' with an 'Add' button, and 'Save' and 'Apply' buttons.



# 源码管理（续1）

- 将源码checkout到子目录

The screenshot shows the Jenkins 'Source Management' configuration page. At the top, there are tabs: General, 源码管理 (highlighted), 构建触发器, 构建环境, 构建, and 构建后操作. Below the tabs, there's a section for 'Branches to build' with a 'Branch Specifier' field containing '\${wptag}' and a 'Add Branch' button. Under 'Source Code Browser' is a dropdown set to '(自动)'. In the 'Additional Behaviours' section, a red box highlights the 'Add' button, which opens a dropdown menu. This menu contains several options: Subversion (radio button), Advanced checkout behaviours, Advanced clone behaviours, Advanced sub-modules behaviours, Calculate changelog against a specific branch, Check out to a sub-directory (highlighted with a red box), Check out to specific local branch, Clean after checkout, and Clean before checkout. To the right of each menu item is a help icon (blue question mark). On the left side of the main configuration area, there's a sidebar with sections like '构建触发器' (Build Triggers) and checkboxes for '触发远程构建 (例如, 使用 GitHub hook trigger for G' and '其他工程构建后触发'.



# 源码管理（续2）

- 将源码checkout到子目录

The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for General, 源码管理 (Source Management), 构建触发器 (Build Triggers), 构建环境 (Build Environment), 构建 (Build), and 构建后操作 (Build Post Actions). The 源码管理 tab is selected.

In the Source Management section, there is a 'Branches to build' configuration. It includes a 'Branch Specifier (blank for 'any')' field containing the value \${wptag} and a 'Add Branch' button.

Below this, under 'Additional Behaviours', there is a 'Check out to a sub-directory' configuration. It includes a 'Local subdirectory for repo' field containing the value wpress\_\${wptag}, which is highlighted with a red rectangle. There is also an 'Add' button and a 'Subversion' checkbox.



# 构建工程

- 构建工程

Jenkins ➤ wpbuild ➤

返回面板 状态 修改记录 工作空间 **Build with Parameters** 删除 工程 配置 重命名

## 工程 wpbuild

工作区 最新修改记录

相关链接

This screenshot shows the Jenkins interface for the 'wpbuild' project. The top navigation bar includes 'Jenkins' and 'wpbuild'. On the left, there's a sidebar with '知识讲解' (Knowledge Lecture). The main content area has a title '工程 wpbuild' and several management options: '返回面板' (Return Panel), '状态' (Status), '修改记录' (Change Log), '工作空间' (Workspace), 'Build with Parameters' (highlighted with a red box), '删除 工程' (Delete Project), '配置' (Configure), and '重命名' (Rename). To the right, there are two sections: '工作区' (Workspace) and '最新修改记录' (Latest Change Log), each with its respective icon.



# 构建工程（续1）

- 选择指定的标签

知识讲解

The screenshot shows the Jenkins interface for the 'wpbuild' project. On the left, there's a sidebar with various options: '返回面板', '状态', '修改记录', '工作空间', 'Build with Parameters', '删除 工程', '配置', and '重命名'. Below this is a 'Build History' section with a sun icon and a plus sign icon. The main content area is titled '工程 wpbuild' and asks for parameters to build the project. A dropdown menu for 'wptag' contains three options: 'v2.0', 'v1.0' (which is highlighted with a red border), and 'origin/master'. At the bottom right is a large blue '开始构建' (Start Build) button, also highlighted with a red border. A note at the bottom says: 'If you selected revisions as a type of presented contents. This may take some time if you have wordpress tag'.

Jenkins ➤ wpbuild ➤

返回面板 状态 修改记录 工作空间 Build with Parameters 删除 工程 配置 重命名

Build History 构建历史 -

工程 wpbuild

需要如下参数用于构建项目：

wptag v2.0  
v1.0  
origin/master

If you selected revisions as a type of presented contents. This may take some time if you have wordpress tag

开始构建

# 检验结果

- 选择指定的标签

知识讲解

返回面板

状态

修改记录

工作空间

Build with Parameters

删除 工程

配置

重命名

## 工程 wpbuild

工作区

最新修改记录

### 相关链接

Build History 构建历史

find X

#1 2018-6-25 下午7:54

RSS 全部 RSS 失败

# 检验结果（续1）

- 查看日志输出

知识讲解

回到工程 状态集 变更记录 控制台输出 编辑编译信息 删 除本次生成 参数 Git Build Data No Tags

构建 #1 (2018-6-25 19:54:33)

No changes.

启动用户 admin

Revision: 50984b10ab0ea24baa947754aa352e8e87fb210b

• v1.0

# 检验结果（续2）

- 查看日志输出

知识讲解



The screenshot shows the Jenkins control console output for a build. The left sidebar lists various build-related actions: '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', '删除本次生成', '参数', 'Git Build Data', 'No Tags', and '前一次构建'. The main area is titled '控制台输出' and displays the following log output:

```
由用户 admin 启动
构建中 在工作空间 /var/lib/jenkins/workspace/wpbuild 中
Cloning the remote Git repository
Cloning repository http://192.168.113.135/devops/wordpress.git
> git init /var/lib/jenkins/workspace/wpbuild/wprefs_v1.0 # timeout=10
Fetching upstream changes from http://192.168.113.135/devops/wordpress.git
> git --version # timeout=10
> git fetch --tags --progress http://192.168.113.135/devops/wordpress.git +refs/heads/*:refs/remotes/origin/*
> git config remote.origin.url http://192.168.113.135/devops/wordpress.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url http://192.168.113.135/devops/wordpress.git # timeout=10
Fetching upstream changes from http://192.168.113.135/devops/wordpress.git
> git fetch --tags --progress http://192.168.113.135/devops/wordpress.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse origin/v1.0^{commit} # timeout=10
> git rev-parse v1.0^{commit} # timeout=10
Checking out Revision 50984b10ab0ea24baa947754aa352e8e87fb210b (v1.0)
> git config core.sparsecheckout # timeout=10
> git checkout -f 50984b10ab0ea24baa947754aa352e8e87fb210b
Commit message: "wordpress init"
> git rev-list --no-walk 50984b10ab0ea24baa947754aa352e8e87fb210b # timeout=10
Finished: SUCCESS
```

# 查看本地结果

- 构建好的项目出现在/var/lib/jenkins目录下

```
[root@localhost ~]# cd /var/lib/jenkins/workspace/
```

```
[root@localhost workspace]# ls  
wpbuild
```

```
[root@localhost workspace]# ls wpbuild/  
wpress_v1.0
```



## 案例4：构建工程

- 创建一个自由风格的项目
- 源码管理采用git
- 构建时可以指定tag
- 构建tag为v1.0的源码



# 分发服务器管理

# 优化构建工程

- 在Jenkins服务器上安装apache，用于分发应用程序
- 为了方便应用服务器下载，Jenkins构建的工程应该打包成为一个文件
- 为了应用服务器可以获知下载的程序文件是没有损坏的，应该为其生成md5值



# 配置分发服务器

- 通过web服务为应用服务器提供应用程序
- 下载目录为/var/www/deploy/packages

```
[root@localhost ~]# yum install -y httpd
[root@localhost ~]# mkdir -pv /var/www/html/deploy/packages
[root@localhost ~]# chown -R jenkins:jenkins /var/www/html/deploy/
[root@localhost ~]# systemctl start httpd
[root@localhost ~]# systemctl enable httpd
```



# 修改工程构建工程

- 为下载的应用打包，以及生成md5可以能过在工程中增加构建步骤完成

知识讲解

The screenshot shows the Jenkins interface for the 'wpbuild' project. At the top, there's a navigation bar with the Jenkins logo and the path 'Jenkins > wpbuild'. Below the navigation bar, there's a sidebar on the left with several options: '返回面板' (Back to Panel), '状态' (Status), '修改记录' (Change History), '工作空间' (Workspace), 'Build with Parameters', '删除 工程' (Delete Project), '配置' (Configure) which is highlighted with a red rectangle, and '重命名' (Rename). To the right of the sidebar, the main content area has a title '工程 wpbuild' and two sections: '工作区' (Workspace) with a folder icon and '最新修改记录' (Latest Change History) with a document icon. At the bottom, there's a section titled '相关链接' (Related Links).

Jenkins > wpbuild

工程 wpbuild

相关链接

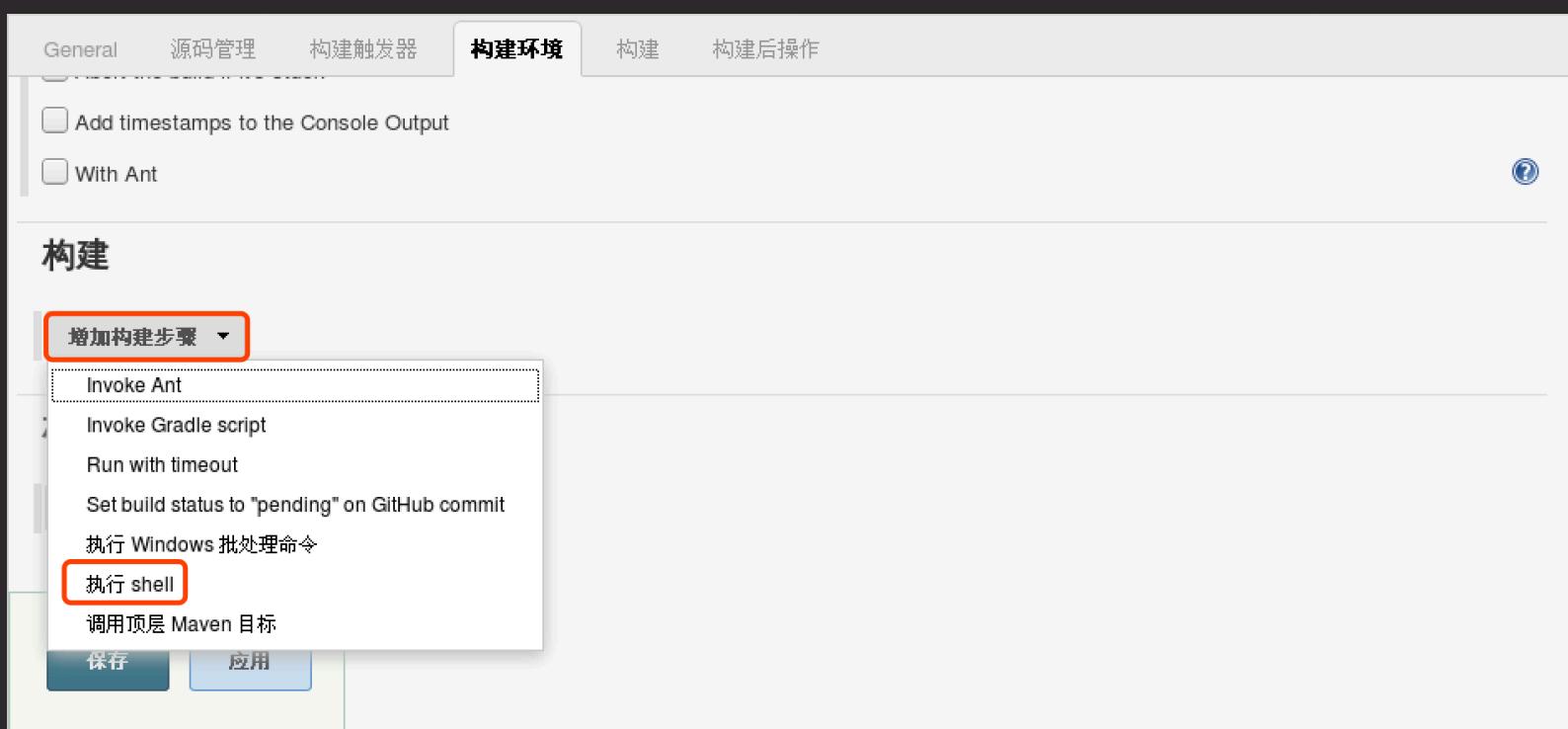
- 返回面板
- 状态
- 修改记录
- 工作空间
- Build with Parameters
- 删除 工程
- 配置
- 重命名

工作区

最新修改记录

# 修改工程构建工程（续1）

- 为下载的应用打包，以及生成md5可以能过在工程中增加构建步骤完成



# 修改工程构建工程（续2）

- 为下载的应用打包，以及生成md5可以通过在工程中增加构建步骤完成

知识讲解

The screenshot shows a build configuration interface with the following details:

- General**, **源码管理**, **构建触发器**, **构建环境**, **构建** (selected), **构建后操作**
- 构建** tab selected
- 执行 shell** section
- 命令**:  
```sh  
deploy\_dir=/var/www/html/deploy/packages/  
cp -r wpress\_\${wptag} \$deploy\_dir  
rm -rf \$deploy\_dir/wpress\_\${wptag}/.git  
cd \$deploy\_dir  
tar czf wpress\_\${wptag}.tar.gz wpress\_\${wptag}  
rm -rf wpress\_\${wptag}  
md5sum wpress\_\${wptag}.tar.gz | awk '{print \$1}' > wpress\_\${wptag}.tar.gz.md5
- 查看 可用的环境变量列表**
- 高级...**
- 增加构建步骤 ▾**



# 构建测试

- 执行构建工程

知识讲解

The screenshot shows the Jenkins interface for the 'wpbuild' project. On the left, there's a sidebar with various options: '返回面板', '状态', '修改记录', '工作空间', 'Build with Parameters' (which is highlighted with a red box), '删除 工程', '配置', and '重命名'. The main area is titled '工程 wpbuild' and asks for parameters to build the project. It shows two dropdown fields: 'wptag' with 'v2.0' and 'v1.0' (which is highlighted with a red box), and 'origin/master'. Below these fields is a note: 'If you selected revisions as a type of presented data an contents. This may take some time if you have a slow wordpress tag'. At the bottom right is a large blue button labeled '开始构建'.

Jenkins > wpbuild >

返回面板 状态 修改记录 工作空间 **Build with Parameters** 删除 工程 配置 重命名

## 工程 wpbuild

需要如下参数用于构建项目：

wptag v2.0  
v1.0

origin/master

If you selected revisions as a type of presented data an contents. This may take some time if you have a slow wordpress tag

**开始构建**

# 构建测试（续1）

- 检查分发服务器的相关目录

```
[root@localhost ~]# ls /var/www/html/deploy/packages/  
wpress_v1.0.tar.gz wpress_v1.0.tar.gz.md5
```

```
[root@localhost ~]# cat /var/www/html/deploy/packages/  
wpress_v1.0.tar.gz.md5  
e4fbaf54a5f580b4e08d5245cc95268
```



## 案例5：修改工程

- 修改构建工程，要求如下：
  - 将下载的应用程序打包放在/var/www/html/deploy/packages目录下
  - 为打包应用程序计算md5值



# 创建版本文件

- 创建两个版本文件
  - live\_version : 表示当前使用版本
  - last\_version : 表示上一个版本
- 应用服务器可以查看live\_version决定是不是要发布新版本
- 如果新版本有问题，应用服务器可以根据last\_version回滚到前一版本



# 创建live\_version文件

- 为了获所正确的版本，通过jenkins工程实现

知识讲解

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with icons for '新建任务' (New Job), '用户' (Users), '构建历史' (Build History), '系统管理' (System Management), '我的视图' (My Views), 'Credentials', and '新建视图' (New View). A red box highlights the '新建任务' button. In the center, there's a search bar with the placeholder '创建live\_version文件'. To the right, a table lists existing Jenkins jobs. The first job is named 'wpbuild', with a blue icon, a cloud icon, and the status '11 分 - #6'. Below the table, it says '图标: S M L'. The top navigation bar has 'Jenkins' and a user icon.

| S | W | 名称                      | 上次成功      |
|---|---|-------------------------|-----------|
|   |   | <a href="#">wpbuild</a> | 11 分 - #6 |

图标: [S](#) [M](#) [L](#)

# 创建live\_version文件（续1）

- 为了获所正确的版本，通过jenkins工程实现

输入一个任务名称

wp\_live\_version

» 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，甚至可以构建多节点构建。

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。

GitHub Organization

选择一个GitHub组织（或用户帐户）为所有匹配某些定义标记的所有仓库构建。

确定

Multibranch Pipeline



# 创建live\_version文件（续2）

- 为了获所正确的版本，通过jenkins工程实现

The screenshot shows the 'General' configuration page for a Jenkins job. At the top, there are tabs for General, Source Management, Build Triggers, Build Environment, Build, and Post-build Actions. The 'General' tab is selected. A checkbox labeled 'Parameterized Build' is checked. Below it, a dropdown menu is open, with the option 'Add Parameter' highlighted by a red box. A sub-menu is displayed, also with a red box around the option 'Character Parameter'. Other options in the sub-menu include 'Credentials Parameter', 'Git Parameter', 'List Subversion tags (and more)', 'String Parameter' (which is also highlighted with a red box), 'Boolean Parameter', 'File Parameter', 'Text Parameter', 'Run-time Parameter', and 'Choice Parameter'. On the left side of the main configuration area, there is a sidebar with sections for 'Source Management' (radio buttons for 'None', 'Git', and 'Subversion', with 'None' selected) and 'Build' (radio buttons for 'None', 'Git', and 'Subversion', with 'None' selected). On the right side, there is a 'Advanced...' button and some help icons.



# 创建live\_version文件（续3）

- 为了获所正确的版本，通过jenkins工程实现

The screenshot shows the 'General' configuration page for a Jenkins job. The 'wptag' parameter is defined under the '字符参数' (Character Parameters) section. The parameter details are as follows:

- 名称**: wptag
- 默认值**: (empty)
- 描述**: wordpress live version

At the bottom of the parameter configuration, there is a preview button labeled "[纯文本] 预览" (Plain Text Preview) and a checkbox for "清除空白字符" (Remove whitespace characters).



# 创建live\_version文件（续4）

- 为了获所正确的版本，通过jenkins工程实现

构建

增加构建步骤 ▾

- Invoke Ant
- Invoke Gradle script
- Run with timeout
- Set build status to "pending" on GitHub commit
- 执行 Windows 批处理命令
- 执行 shell
- 调用顶层 Maven 目标

保存 应用



# 创建live\_version文件（续5）

- 为了获所正确的版本，通过jenkins工程实现

构建

执行 shell

命令 echo \${wptag} > /var/www/html/deploy/live\_version

X ?

查看 可用的环境变量列表

高级...

增加构建步骤 ▾

# 执行构建版本工程

- 创建主机操作与获取信息操作完全一样，只是传递的请求参数不一样而已

Jenkins > wp\_live\_version >

The screenshot shows the Jenkins interface for the 'wp\_live\_version' project. On the left, there's a sidebar with icons for navigating back to the main panel, viewing status, viewing history, viewing workspace, building with parameters, deleting the project, configuring it, and renaming it. The main area is titled '工程 wp\_live\_version' and displays a parameter configuration section. It shows 'wptag' set to 'v1.0' and 'wordpress live version'. A large blue button at the bottom right is labeled '开始构建' (Start Build).

工程 wp\_live\_version

需要如下参数用于构建项目：

wptag v1.0

wordpress live version

开始构建

- 返回面板
- 状态
- 修改记录
- 工作空间
- Build with Parameters
- 删除 工程
- 配置
- 重命名

# 执行构建版本工程（续1）

- 查看构建结果

```
[root@localhost ~]# ls /var/www/html/deploy/  
live_version packages
```

```
[root@localhost ~]# cat /var/www/html/deploy/live_version  
v1.0
```



## 案例6：创建版本文件

- 为了记录应用的当前版本和前一个版本，创建两个工程：
  - 创建live\_version，记录应用程序当前版本
  - 创建last\_version，记录应用程序前一个版本



# 管理应用服务器



# 自动化部署

# 服务器规划

- 为了方便版本的切换，可以规划如下目录
  - /var/www/download用于存储下载的应用
  - /var/www/deploy用于存储解压的应用
- 创建/var/www/html/current软链接，指向需要部署的应用版本



# 下载应用

- 编写下载应用的功能代码
  - 通过位置参数指定要下载的版本
  - 位置参数是live下载当前版本
  - 位置参数是last下载前一个版本
  - 如果已经下载，则不要重复下载



# 校验文件

- 编写校验文件代码
  - 计算指定文件的md5值
  - 将md5值与发布服务器提供的md5值进行比较，以确认下载的文件无误



# 发布应用

- 编写应用发布代码
  - 根据指定的版本，创建/var/www/html/current链接，指向到不同的发布版本



# 案例7：发布应用

- 编写应用发布程序
  - 根据分发服务器的版本，下载对应的应用
  - 校验应用程序
  - 发布指定程序



# 总结和答疑