

Jenkins Phabricator 持续集成

一、	简介	2
1.	持续集成.....	2
2.	Jenkins.....	2
3.	Phabricator	2
4.	目标	2
二、	搭建	2
1.	所需的软件	2
2.	环境搭建.....	3
	安装 Jenkins 和 Phabricator	3
	Phabricator 配置	3
	Jenkins 配置.....	4
三、	Jenkins 和 Phabricator 集成.....	15
1.	安装配置.....	15
2.	使用	16
四、	Reference	24

一、 简介

1. 持续集成

一种软件开发实践，即团队的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，这导致每天发生多次集成。每次集成都通过自动化的构建(包括测试)来验证，从而尽快地检测出集成错误。许多团队发现这个过程会大大减少集成问题，让团队能够更快地开发内聚的软件。本质上说，**持续集成就是通过快速的反馈降低风险**。

2. Jenkins

Jenkins 是一个用 java 编写的持续集成工具。它为软件开发提供持续集成服务。它基于服务器，并运行于 servlet 容器，例如 Apache Tomcat。它支持版本控制工具，包括 CVS, Subversion, Git, Mercurial, Perforce, Clearcase 和 RTC, 并能执行基于 Apache Ant 和 Apache Maven 的项目。另外，它能够通过多种方法开始构建，例如，由版本控制系统触发，由 cron-like 机制调度，由远程命令触发，或者通过一个 URL。

3. Phabricator

Phabricator 是一套基于网络的软件开发协作工具，包括代码审查工具、库浏览器、Herald 变化检测工具、Bug 跟踪、Phriction Wiki。Phabricator 集成了 Git、Mercurial、和 Subversion 版本控制工具。它在 Apache License 2 协议下发布的自由软件。

4. 目标

我们的目标是采用 Git + Phabricator + Jenkins 模式，实现 Git 托管代码，Phabricator 进行代码审查、Bug 跟踪，Jenkins 集成构建的流程。通过 Phabricator 和 Jenkins 的强大功能，完善代码管理和编译，测试以及部署，提高软件质量并加速软件开发。

二、 搭建

1. 所需的软件

为了搭建一个完整的持续集成系统，我们需要如下软件：

Apache: Internet 上最流行的 Web 服务器软件，仅支持静态网页

PHP: 在服务器端执行的嵌入 HTML 文档的脚本语言

MariaDB/MySQL: 数据库管理系统

Git: 版本控制系统的一个免费开源客户端

Jenkins: 持续构建工具

Phabricator: Code review 工具

以下分别是这些软件的安装配置和使用方法的简要介绍，这里都是基于 CentOS 7 64 位系统介绍，其它版本的操作系统暂不介绍。

2. 环境搭建

安装 Jenkins 和 Phabricator

具体内容查看安装脚本 [install-jenkins-phabricator.sh](#)。

Phabricator 配置

安装完成后，打开 <http://<your-server>>，首先注册一个管理员账户。

按照官方配置指南 [Configuration Guide](#)，需要进行多项配置，这里只介绍几个比较重要的，其余可以省略或自行参考官方指南。

➤ [Accounts and Registration](#)

打开 <http://<your-server>/auth>，添加一个或多个登录系统。登录方法称为 Authentication Provider，例如以下：

- Username/Password: 用户使用用户名和密码进行登录或注册
- LDAP: 用户使用 LDAP 认证进行登录或注册
- OAuth: 用户使用支持 OAuth2 的提供商账户（例如 Github、Facebook 或 Google）进行登录或注册。

- 恢复管理员账户

如果你意外的锁住了你的管理员用户，可以使用 `bin/auth` 脚本进行恢复，要恢复管理员账户，运行：

```
phabricator/$ ./bin/auth recover <username>
```

- 使用 Web 控制台管理账户

打开 <http://<your-server>/people/> 管理账户

- 手动创建新账户

这里有两种方式创建用户，第一种是通过 web 用户界面

<http://<your-server>/people/>，另外一种是通过 CLI 使用 `accountadmin` 可执行文件：

```
Phabricator/$ ./bin/accountadmin
```

➤ [Configuring Outbound Mail](#)

Phabricator 可以通过多种方式发送邮件，称为 “Adapter”。这里使用 External SMTP，配置已经在安装脚本中写好，只需要设置好对应的参数，不再需要手动配置。如果需要使用其它邮件发送方式，可以查看官方文档。

➤ [Diffusion User Guide: Repository Hosting](#)

Phabricator 支持 Git、Subversion 等版本控制工具，可以通过 HTTP 和 SSH 对其托管的库进行读写访问验证。这里我们使用 Git 版本控制工具并通过 SSH 进行访问。配置已经在安装脚本中完成，但每个 Phabricator 用户还需要在 Phabricator 中添加公钥。步骤如下：

- 生成密钥

运行命令 `ssh-keygen -t rsa`

- 上传到 Phabricator

登录到 Phabricator，打开 <http://<your-server>/settings/panel/ssh>，点击 Upload Public Key，将刚才生成的密钥公钥内容 `~/.ssh/id_rsa.pub` 复制进去。

Jenkins 配置

Jenkins 安装完成后，进入 <http://<your-server>:8080/manage> 配置 Jenkins，在这个页面可以完成配置系统，全局安全设置，从磁盘重新加载配置信息，管理插件，显示系统信息，显示系统日志，显示统计信息，命令行工具，脚本控制台，管理节点，管理认证，关于 Jenkins，管理旧数据，管理用户和关闭 jenkins 等任务。以下是各个任务的简要介绍。

在配置系统中，可能由于没有安装某些插件而缺少相应的配置选项，只需要在 [管理插件](#) 部分安装插件即可。对于我们的目标，我们首先需要安装 `git plugin`, `git client plugin`, `git server plugin`, `git parameter plugin`, `build authorization token root plugin`, `build with parameters`, `post build script plugin`, `phabricator differential plugin` 和 `cobertura plugin`。安装方法可以查看 [插件管理](#) 部分

Jenkins

search

Jenkins

New Job

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Manage Jenkins

New version of Jenkins (1.535) is available for [download](#) ([changelog](#)).

Configure System

Configure global settings and paths.

Configure Global Security

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

Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)

System Information

Displays various environmental information to assist trouble-shooting.

System Log

System log captures output from java.util.logging output related to Jenkins.

Load Statistics

Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI

Access/manage Jenkins from your shell, or from your script.

Script Console

Executes arbitrary script for administration/trouble-shooting/diagnostics.

Manage Nodes

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Manage Credentials

Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.

About Jenkins

See the version and license information.

Manage Old Data

Scrub configuration files to remove remnants from old plugins and earlier versions.

Manage Users

Create/delete/modify users that can log in to this Jenkins

Prepare for Shutdown

Stops executing new builds, so that the system can be eventually shut down safely.

Help us localize this page

Page generated: Oct 18, 2013 9:58:36 AM

REST API

1) Configure System

Jenkins 的一些基本配置内容可以查看文件 `/etc/sysconfig/jenkins`

系统配置最上面显示了 Jenkins 所在目录

点击后面的 `advances` 可以配置工作目录和日志保存目录

Workspace root Directory: 设置每次构建工作区的目录

Build Record Root Directory: 设置每次构建日志的目录

Home directory	/var/lib/jenkins
Workspace Root Directory	<input type="text" value="\${JENKINS_HOME}/workspace/\${ITEM_FULLNAME}"/>
Build Record Root Directory	<input type="text" value="\${ITEM_ROOTDIR}/builds"/>
System Message	<div><div></div><div>[Escaped HTML] Preview</div></div>
# of executors	<input type="text" value="2"/>
Labels	<input type="text"/>
Usage	<input type="text" value="Utilize this node as much as possible"/>
Quiet period	<input type="text" value="5"/>
SCM checkout retry count	<input type="text" value="0"/>
<input checked="" type="checkbox"/> Restrict project naming	
Naming Strategy	<input type="text" value="Strategy"/>

工程名称设置，可以要求工程名称符合某种格式

☒ Restrict project naming
Naming Strategy

Strategy

☐ Default

☒ Pattern

Name Pattern

*

Description

force existing ☐

全局属性：这里可以添加一些环境变量，之后在任务中可以引用，还可以添加一些工具，只需要填写工具的名称和所在目录。

Global properties

☒ Environment variables

List of key-value pairs

Add

☒ Tool Locations

List of tool locations

Name

Home

Add

Delete

Maven：可以设置所使用的 Maven 来源，默认选项或者自己设置 maven 所在路径

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

Use default maven global settings

Global settings file on filesystem

JDK

如果我们之前已经安装了 JDK，那么这里使用默认选项即可；这里我们指定名称和已经安装的 JDK 位置。

JDK

JDK installations

JDK

Name

JDK8

JAVA_HOME

/usr/java/jdk1.8.0_60/

☐

Install automatically

?

Delete JDK

Add JDK

List of JDK installations on this system

Git

Git installations

Git

Name

Default

Path to Git executable

git

☐

Install automatically

?

Delete Git

Add Git

description

上面由于可执行程序 `git` 已经在系统 `PATH` 路径中，所以可以不使用完整路径。
后面的 `ANT` 和 `Maven` 安装类似，这里不再讲述。截图如下：

Ant

Ant installations

Ant

Name

ant

ANT_HOME

/usr/share/ant

☐

Install automatically

?

Delete Ant

Add Ant

List of Ant installations on this system

Maven

Maven installations

Maven

Name

Maven3

MAVEN_HOME

/usr/share/maven

☐

Install automatically

?

Delete Maven

Add Maven

List of Maven installations on this system

Maven 工程配置

Maven Project Configuration

Global MAVEN_OPTS

Local Maven Repository

Default (~/.m2/repository)

☒

Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

Jenkins URL: 设置 Jenkins 的网络地址

Email Add: 设置 jenkins 的 email 地址。Jenkins 发送的邮件会以该用户名和邮件地址显示。

Jenkins Location

Jenkins URL	<input type="text" value="http://10.61.3.157:8080/"/>
<small>Optionally specify the HTTP address of the Jenkins installation, such as <code>http://yourhost.yourdomain/jenkins/</code>. This value is used to let Jenkins know how to refer to itself, ie. to display images or to create links in emails. This is necessary because Jenkins cannot reliably detect such a URL from within itself.</small>	
System Admin e-mail address	<input type="text" value="Jenkins Server <jenkins@nobida.ict.cn>"/>
<small>Notification e-mails from Jenkins to project owners will be sent with this address in the from header. This can be just "foo@acme.org" or it could be something like "Jenkins Daemon <foo@acme.org>"</small>	

Git plugin 和 CVS 使用默认配置

Git plugin

Global Config user.name Value	<input type="text"/>
Global Config user.email Value	<input type="text"/>
Create new accounts base on author/committer's email	<input type="checkbox"/>

CVS

Default Compression Level	<input type="text" value="3 (Recommended)"/>
Private Key Location	<input type="text" value="/var/jenkins_home/.ssh/id_rsa"/>
Private Key Password	<input type="password" value="....."/>
Known Hosts Location	<input type="text" value="/var/jenkins_home/.ssh/known_hosts"/>
Authentication	<input type="button" value="Add"/>

Email 配置: 设置发送邮件的 smtp 服务器, 应特别注意, 缺省邮箱后缀

点击 “Advanced” 可以配置 smtp 认证, 包括用户名, 用户密码, 缺省编码等。

Test configuration by sending test e-mail: 通过发送邮件测试配置是否成功, 点击 Test Configuration, 如果成功会显示 Email was successfully sent, 否则下面会报错, 出错时请认真看下面的出错提示, 可能错误是 SMTP server 没有配置对, 默认是 localhost, 或者是某些服务没有打开, 或者是端口已经被其他程序占用, 默认端口是 25, 可以不填写, 另外重启一下也可能就成功。

E-mail Notification

SMTP server	<input type="text" value="mail.software.ict.ac.cn"/>
Default user e-mail suffix	<input type="text" value="@software.ict.ac.cn"/>
<input checked="" type="checkbox"/> Use SMTP Authentication	
User Name	<input type="text" value="luoyuanhao@software.ict.ac.cn"/>
Password	<input type="password" value="....."/>
Use SSL	<input type="checkbox"/>
SMTP Port	<input type="text" value="25"/>
Reply-To Address	<input type="text"/>
Charset	<input type="text" value="UTF-8"/>
<input checked="" type="checkbox"/> Test configuration by sending test e-mail	
Test e-mail recipient	<input type="text" value="1102781439@qq.com"/>

Email was successfully sent



This is test email #7 sent from Jenkins

到这里之后点击 **apply** 或者 **save** 保存设置，如果出现错误，根据错误提示进行更正即可。

2) Configure global security

这里主要完成一些关于系统安全和访问权限的配置，勾选 **Enable security** 才可以配置，否则不启用其机制。前面的一般使用默认配置，如下图



Configure Global Security

☒ Enable security

TCP port for JNLP slave agents ☐ Fixed : ☒ Random ☐ Disable

Disable remember me ☐

Markup Formatter

Raw HTML

Treat the text as HTML and use it as is without any translation

☐ Disable syntax highlighting

Access Control

Security Realm

☐ Active Directory

☐ Delegate to servlet container

☒ Jenkins's own user database

☒ Allow users to sign up

☐ LDAP

Authorization

☒ Anyone can do anything

☐ Legacy mode

☐ Logged-in users can do anything

☐ Matrix-based security

☐ Project-based Matrix Authorization Strategy

☐ Prevent Cross Site Request Forgery exploits

Disable remember me: 如果勾选，每次登陆都需要输入用户名和密码

Disable syntax highlighting: 是否启用 html 语法高亮

Active Directory: 如果没有这个选项，则需要先安装 Jenkins Active Directory Plugin 插件

<input checked="" type="checkbox"/>	Jenkins Active Directory plugin This plugin enables authentication through Active Directory on Windows environment.
-------------------------------------	--

勾选这个选项，可以使用公司的域账号登陆

Delegate to servlet container: 使用容器的用户，tomcat 也可以添加用户，这里不讨论。

☒ Delegate to servlet container

Unprotected URLs

These URLs (and URLs starting with these prefixes plus a /) should require no authentication. If possible, configure your container to pass these requests straight to Jenkins without requiring login.

- [git](#)
- [inlpJars](#)
- [subversion](#)
- [whoAmI](#)

Jenkins's own user database: 使用 jenkins 自身的账户，需要勾选 Allow users to sign up 以允许注册

☒ Jenkins's own user database

☒ Allow users to sign up

Authorization

- ☒ Anyone can do anything
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☐ Matrix-based security
- ☐ Project-based Matrix Authorization Strategy

上面主要是用户访问权限设置，有下面一些选项：

Anyone can do anything: 任何人可以做任何事

Legacy mode: 如果是管理员，则拥有所有权限，其他用户或匿名用户都只有读权限

Logged-in users can do anything: 登录用户可以做任何事

Matrix-based security: 基于矩阵的安全，功能看点击后面蓝色问号后的提示，见下图，不翻译

Matrix-based security

In this scheme, you can configure who can do what by using a big table.

Each column represents a permission. Hover the mouse over the permission names to get more information about what they represent.

Each row represents a user or a group (often called 'role', depending on the security realm.) This includes a special user 'anonymous', which represents unauthenticated users, as well as 'authenticated', which represents authenticated users. Use the text box below the table to add new users/groups/roles to the table, and click the [x] icon to remove it from the table.

Permissions are additive. That is, if a user X is in group A, B, and C, then the permissions that this user actually has are the union of all permissions given to X, A, B, C, and anonymous.

Project-based Matrix Authorization strategy: 基于工程的矩阵授权策略，和上面基于矩阵的区别不大，只是基于工程的可以对各个工程进行不同的权限配置，而上面的是对所有 jobs 都一样。

3) Reload Configuration from Disk

重新从磁盘加载配置信息。

4) Manage plugins

管理各种插件，包括安装，升级，删除等等。

Updates				
Available				
Installed				
Advanced				
Install	Name ↓	Version	Installed	
<input type="checkbox"/>	Credentials Plugin This plugin allows you to store credentials in Jenkins.	1.9.1	1.4	
<input type="checkbox"/>	CVS Plugin This bundled plugin integrates Jenkins with CVS version control system.	2.9	2.8	
<input type="checkbox"/>	LDAP Plugin Security realm based on LDAP authentication.	1.6	1.2	
<input type="checkbox"/>	Monitoring external jobs Adds the ability to monitor the result of externally executed jobs.	1.2	1.1	
<input type="checkbox"/>	PAM Authentication Plugin Adds Unix Pluggable Authentication Module (PAM) support to Jenkins.	1.1	1.0	
<input type="checkbox"/>	SSH Credentials Plugin This plugin allows you to store SSH credentials in Jenkins. Warning: New version not compatible with installed version. Jobs using this plugin may need to be reconfigured.	1.5.1	0.3	
<input type="checkbox"/>	SSH Slaves plugin This plugin allows you to manage slaves running on *nix machines over SSH. Warning: New version not compatible with installed version. Jobs using this plugin may need to be reconfigured. Warning: This plugin requires dependent plugins be upgraded and some of these dependent plugins are not compatible with the current installed version. Jobs using these dependent plugins may need to be reconfigured.	1.5	0.25	
<input type="checkbox"/>	Subversion Plugin This plugin adds the Subversion support (via SVNKit) to Jenkins.	1.53	1.45	

如上图所示, Updates 表示已安装可升级插件, Available 表示可用但未安装插件, Installed 表示已安装插件, Advanced 里面可以从本地安装插件。如果 Available 界面没有任何可用更新, 可能是更新网址不正确。进入 Advanced 界面更改更新网址为:
<https://updates.jenkins-ci.org/current/update-center.json>

Update Site

URL

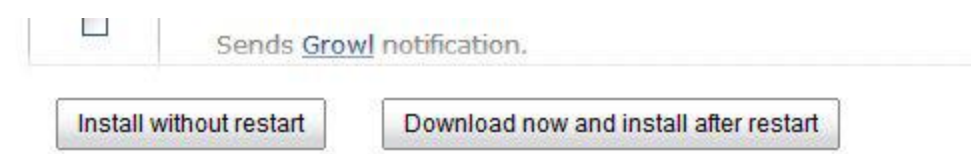
Update information obtained: 1 day 20 hr ago

点击 Check Now 检查是否有可用更新和可用插件。

Available 列表里有很多可选插件, 插件是按照类型分类, 按字母序排序的。类型包括有 Artifact Uploaders 上传文件插件, Authentication and User Management 认证和用户管理类插件, Build Notifiers 构建提示插件, 构建报告插件等等。如下图所示, 点击类型名可以展开此类型下所有可用的插件, 每个插件都有简要的功能介绍。

Updates	Available	Installed	Advanced
Install ↓			
Name			
Artifact Uploaders			
Authentication and User Management			
Build Notifiers			
Build Reports			
Build Tools			
Build Triggers			
Build Wrappers			
Cluster Management and Distributed Build			
Command Line Interface			
External Site/Tool Integrations			
List view columns			
Maven			

勾选所有想要安装的插件，下拉到最后，可以点击安装，安装完成后重启 jenkins 即可，如下图：



5) System information

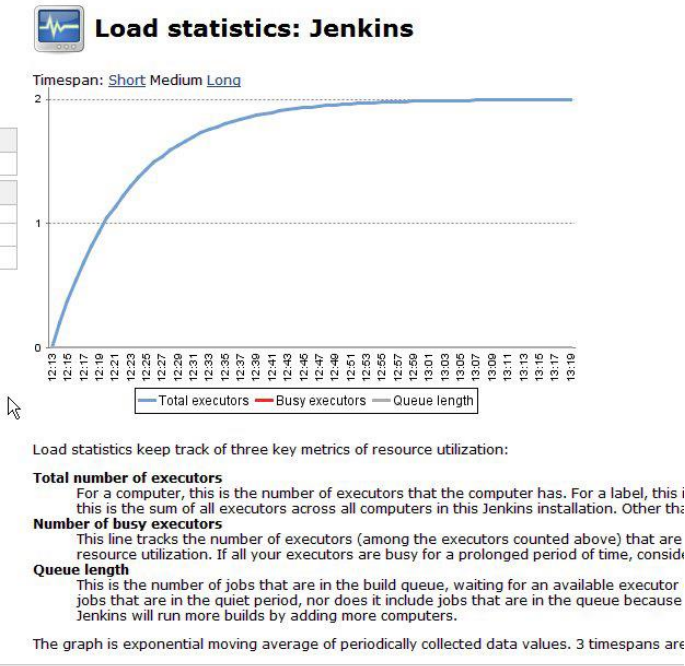
这里主要显示了一些系统属性，环境变量，已安装的插件汇总等信息。

6) System log

主要显示了系统的日志

7) Load statistics

一些统计数据，例如节点数目，正在运行的节点数，构建队列长度等信息



8) Jenkins CLI

命令行接口，需要下载 jenkins-cli.jar 文件。将在 [jenkins cli 命令行使用简介.doc](#) 中详细介绍。



Jenkins CLI

You can access various features in Jenkins through a command-line tool. See [the Wiki](#) for more details follows:

```
java -jar jenkins-cli.jar -s http://localhost:8080/jenkins/ help
```

Available Commands


- **build**: Builds a job, and optionally waits until its completion.
- **cancel-quiet-down**: Cancel the effect of the "quiet-down" command.
- **clear-queue**: Clears the build queue
- **connect-node**: Reconnect to a node
- **console**: Retrieves console output of a build
- **copy-job**: Copies a job.
- **create-job**: Creates a new job by reading stdin as a configuration XML file.
- **create-node**: Creates a new node by reading stdin as a XML configuration.
- **delete-builds**: Deletes build record(s).
- **delete-job**: Deletes a job
- **delete-node**: Deletes a node
- **disable-job**: Disables a job
- **disconnect-node**: Disconnects from a node
- **enable-job**: Enables a job
- **get-job**: Dumps the job definition XML to stdout
- **get-node**: Dumps the node definition XML to stdout
- **groovy**: Executes the specified Groovy script.
- **groovysh**: Runs an interactive groovy shell.
- **help**: Lists all the available commands.
- **install-plugin**: Installs a plugin either from a file, an URL, or from update center.
- **install-tool**: Performs automatic tool installation, and print its location to stdout. Can be only
- **keep-build**: Mark the build to keep the build forever.
- **list-changes**: Dumps the changelog for the specified build(s).
- **list-jobs**: Lists all jobs in a specific view or item group.
- **list-plugins**: Outputs a list of installed plugins.
- **login**: Saves the current credential to allow future commands to run without explicit credenti
- **logout**: Deletes the credential stored with the login command.
- **mail**: Reads stdin and sends that out as an e-mail.
- **offline-node**: Stop using a node for performing builds temporarily, until the next "online-nod
- **online-node**: Resume using a node for performing builds, to cancel out the earlier "offline-nod
- **quiet-down**: Quiet down Jenkins, in preparation for a restart. Don't start any builds.
- **reload-configuration**: Discard all the loaded data in memory and reload everything from file
- **restart**: Restart Jenkins

9) Script console

脚本控制台，通过执行脚本和 jenkins 交互。

10) Manage Nodes

节点管理，我们暂时只有一个 master 节点，不介绍多节点的使用。

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	78.46 GB	3.87 GB	78.46 GB	0ms
Data obtained		55 min	55 min	55 min	55 min	55 min	55 min

11) Manager credential

增加，修改或者删除认证，可用于 jenkins 和 jenkins job 和第三方服务的验证。

12) About jenkins


介绍了 jenkins 的版本信息，还有所依赖的第三方库列表和 license。

13) Manager Old Data

管理旧数据，具体介绍看 jenkins 的解释

14) Manager Users

管理用户，这个要在 **Configure Global security** 中选中允许用户注册才可以使用



Configure Global Security

☒ Enable security

TCP port for JNLP slave agents ☐ Fixed : ☒ Random ☐ Disable

Disable remember me ☐

Markup Formatter

☐ Treat the text as HTML and use it as is without any translation

☐ Disable syntax highlighting

Access Control

Security Realm

☐ Delegate to servlet container






☒ Jenkins's own user database

☒ Allow users to sign up

这里显示了已经注册的用户，已登录的用户后面没有红色删除标记，但可以删除其他用户，点击后面蓝色的配置标记可以进入相应用户的配置。

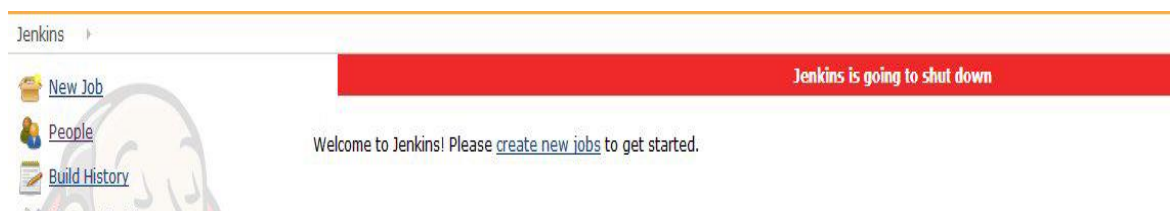
Users

These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User Id	Name	
 amosbird	Amos Bird	 
 yhluo	luoyuanhao	

➤ Prepare for shutdown

关掉 jenkins



三、 Jenkins 和 Phabricator 集成

1. 安装配置

集成 Jenkins 和 Phabricator 需要在 Jenkins 中安装 [Phabricator Differential Plugin](#) 插件，配置过程如下：

在 Phabricator 中新建 robot 用户 Jenkins

Create New User

Choose the type of user account to create. For a detailed explanation of us

- Account Type
- ☐ Create Standard User
Create a standard user account.
 - ☒ Create Bot User
Create a new user for use with automated scripts.
 - ☐ Create Mailing List User
Create a mailing list user to represent an existing, external mailing list like a C

Create New User

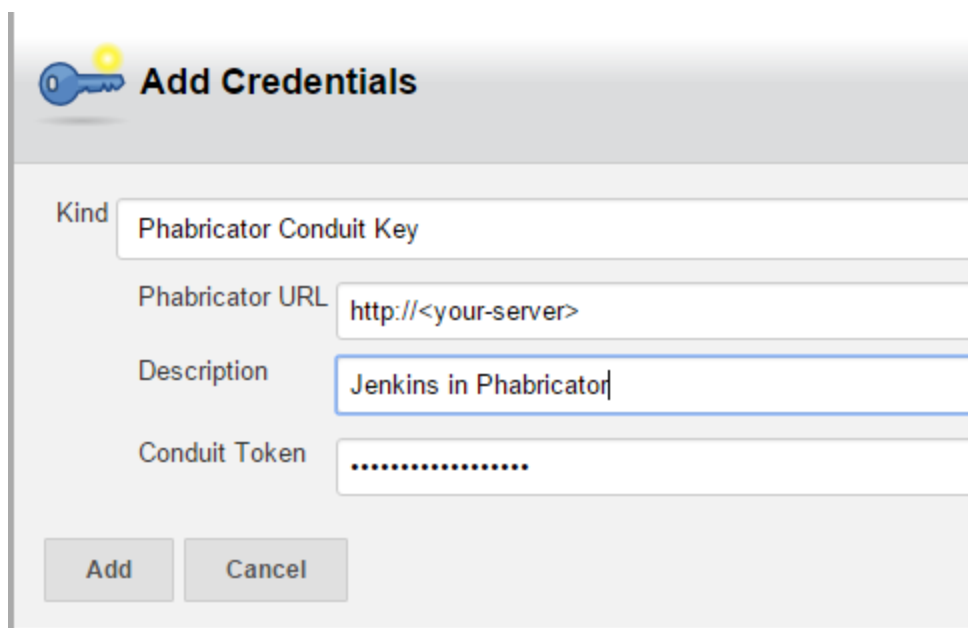
You are creating a new **bot** user account.

Username jenkins

Real Name Jenkins

Email jenkins@nobida.ict.ac.cn

新建用户完成后，进入 Jenkins 用户界面 <http://<your-server>/p/jenkins>，点击右边的 Edit Settings，再点击 Conduit API Tokens，点击 Generate API Token，复制生成的 Token。进入到 Jenkins 系统配置页面 <http://<your-server>:8080/configure>，找到 Phabricator 部分，添加 Credentials，填写 Phabricator URL 并将之前生成的 Token 粘贴到此处。



Add Credentials

Kind: Phabricator Conduit Key

Phabricator URL: http://<your-server>

Description: Jenkins in Phabricator

Conduit Token:

Add Cancel

2. 使用

在 Phabricator 新建 Repository 处 <http://<your-server>/diffusion/new> 新建一个 Repository。

Choose a human-readable name for this repository, like "Core". You can change this later.

Name

Human-readable repository name.

Choose a "Callsign" for the repository. This is a short, unique identifier. For example, you might use "M" for your mobile app repository and "W" for your web service repository.

Callsigns must be UPPERCASE, and can not be edited after the repository is created.

Callsign

Short UPPERCASE identifier.

1. Clone 该库至本地，执行 `git commit --allow-empty -m "initial empty commit"`
2. `git push origin master`
在 Jenkins <http://<your-server>:8080/view/All/newJob> 中新建对应该 Diffusion 的 Job，名称为 hello-world，free-style

Project name	<input type="text" value="hello-world"/>
Description	<input type="text" value="test project hello-world in phabricator"/>

由于需要将构建结果返回给 Phabricator，这里需要添加两个 String 参数 DIFF_ID 和 PHID：

☒ This build is parameterized

String Parameter

Name	<input type="text" value="DIFF_ID"/>
Default Value	<input type="text"/>
Description	<input type="text" value="The differential diff ID"/>
[Escaped HTML] Preview	

String Parameter

Name	<input type="text" value="PHID"/>
Default Value	<input type="text"/>
Description	<input type="text" value="The PHID of the current build target."/>

在源代码管理中，选择 Git，Repository URL 填写 git clone Phabricator Diffusion 时的后半部分，由于 Git 已经设置了所有用户可以 Clone，这里不需要认证，否则可以添加用户名密码或者密钥认证。

Source Code Management

- ☐ None
☐ CVS
☐ CVS Projectset
☒ Git
Repositories

Repository URL

Credentials

Name

ID of the repository, such as origin, to uniquely identify this repository among other remote repositories. This is the same "name" that you use in your git remote command. If left empty, Jenkins will generate unique names for you.
You normally want to specify this when you have multiple remote repositories.
[\(from Jenkins GIT plugin\)](#)

Refspec

上面的 name 表示库的 ID，例如 origin，用于和其它远端库区分。Refspec 用于指定获取远端的哪些 refs 以及和本地 refs 对应。如果留空，则和 git fetch 效果相同。

Branches to build Branch Specifier (blank for 'any')

Specify the branches if you'd like to track a specific branch in a repository. If left blank, all branches will be examined for changes and built.

Branches to build: 指定要跟踪的分支，如果留空，则会检查构建所有分支。

Build Triggers

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

Use the following URL to trigger build remotely: `JENKINS_URL/job/hello-world/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`
Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

☐ Build after other projects are built

在 Build Triggers 部分，勾选 Trigger builds remotely，这样可以通过 URL 触发构建。也就是后面 Phabricator 发出一个 URL POST 请求触发构建。这里填写一个验证令牌，用于 Phabricator 或 URL 触发时认证。记住下面 jenkins 提示的 Use the following URL to trigger build remotely:

http://<your-server>:8080/job/hello-world/buildWithParameters?token=TOKEN_NAME¶m1=value1¶m2=value2

Build

Execute shell

Command

```
sh ./build.sh
echo diff-id=${DIFF_ID}
echo phid=${PHID}
```

See [the list of available environment variables](#)

Add build step

- Execute Python script
- Execute Windows batch command
- Execute shell
- HTTP Request
- Invoke Ant
- Invoke top-level Maven targets

在 **Build** 部分，填写需要构建的具体内容，例如编译，单元测试，集成测试等。这里执行 `build.sh` 脚本编译库中的 `c` 和 `java` 文件，然后打印两个参数的值。

Post-build Actions

Post to Phabricator

Comment on Success



Post a differential comment on successful builds

Enable Uberalls



Enable code coverage reporting

Read comment from file

`.phabricator-comment`

Add additional context to Phabricator comment by outputting to this file

Preserve Formatting



Preserve comment formatting

Maximum comment character length

1000

Comment with console link on Failure



Post a differential comment on failed builds with a link to the raw Console Output

Add post-build action

在 **Post-build Actions** 部分，添加 **Post to Phabricator**，将构建结果返回给 **Phabricator**。

3. 在 **Phabricator** 中新建 **Harbormaster build plan**

<http://<your-server>/harbormaster/plan/edit>

New Build Plan

Plan Name

Cancel

Create Build Plan

在该 plan 中添加 build step:

Edit Step: Make HTTP Request

Name

URI

HTTP Method

Credentials [Add Credential](#)

Next Steps

After completing this build step Harbormaster can continue the build normally, or it can pause the build a message. If you are using this build step to trigger some work in an external system, you may want to wait for that system to perform the work and report results back.

If you select **Continue Build Normally**, the build plan will proceed once this step finishes.

If you select **Wait For Message**, the build plan will pause indefinitely once this step finishes. To resume the external system must call `harbormaster.sendMessage` with the build target PHID, and either "pass" or "fail" to indicate the result for this step. After the result is recorded, the build plan will resume.

When Complete

填写名称, 由于该 build step 的 action 就是发送一个 URL POST 请求, 这里使用 Make HTTP Request, 然后在下面的 URI 中填写之前 Jenkins 提示的, 因为 Jenkins 根据 PHID 向 Phabricator 返回构建结果, 这里需要传递 PHID 作为参数:

[http://<your-server>:8080/job/hello-world/buildWithParameters?token=hello-world-token&DIFF_ID=\\${buildable.diff}&PHID=\\${target.phid}](http://<your-server>:8080/job/hello-world/buildWithParameters?token=hello-world-token&DIFF_ID=${buildable.diff}&PHID=${target.phid})

在 HTTP Method 部分选择 POST, 在 When Complete 中选择 Wait For Message, 等待 Jenkins 返回结果再进行下一个 build step。点击 Save Build Step 保存构建步骤。

在下面还列出了其它 Phabricator 可以传递给 Jenkins 的参数以及参数解析。

Build Variables

The following variables can be used in most fields. To reference a variable, use `${name}` in a field.

Variable	Description
<code>build.id</code>	The ID of the current build.
<code>buildable.commit</code>	The commit identifier, if applicable.
<code>buildable.diff</code>	The differential diff ID, if applicable.
<code>buildable.revision</code>	The differential revision ID, if applicable.
<code>repository.callsign</code>	The callsign of the repository in Phabricator.
<code>repository.phid</code>	The PHID of the repository in Phabricator.
<code>repository.staging.ref</code>	The ref name for this change in the staging repository.
<code>repository.staging.uri</code>	The URI of the staging repository.
<code>repository.uri</code>	The URI to clone or checkout the repository from.
<code>repository.vcs</code>	The version control system, either "svn", "hg" or "git".
<code>step.timestamp</code>	The current UNIX timestamp.
<code>target.phid</code>	The PHID of the current build target.

4. 在 Phabricator 中添加 Herald 规则
到 <http://<your-server>/herald/new> 创建 Herald 规则:

Create Herald Rule

- New Rule for
- ☒ **Commits**
React to new commits appearing in tracked repository.
Commit rules can send email, flag commits, trigger a
 - ☐ **Differential Diffs**
React to new diffs being uploaded, before writes occur.
These rules can reject diffs before they are written to other sensitive information.
 - ☐ **Differential Revisions**
React to revisions being created or updated.
Revision rules can send email, flag revisions, add revision
 - ☐ **Manifest Tasks**
React to tasks being created or updated.
 - ☐ **Outbound Mail**
Route outbound email.
 - ☐ **Pholio Mocks**
React to mocks being created or updated.
 - ☐ **Phriction Documents**
React to wiki documents being created or updated.

可以针对多种情况创建 Herald 规则，例如 commit，diff 或者 revision 等。最主要的就是这三个。这里我们为 Differential Revisions 创建一个 Herald 规则，当新建或者更新 revision 时触发 build step 从而发出 HTTP POST 请求让 Jenkins 执行构建。

Rule Name 填写规则名称，在 Condition 部分添加条件，在 Action 部分添加满足上面的条件时需要执行的操作。例如当有 hello-world 库有 revision 创建或者更新，且 reviewer 是 yhluo2 时，执行 build plan，可以按照如下方式创建 Herald 规则：

Edit Herald Rule

Rule Name

This **Global** rule triggers for **Differential Revisions**.

Conditions

When these conditions are met:

ActionTake these actions this rule matches:

5. 创建 revision 触发 jenkins 构建

用户 yhluo clone Phabricator 的 hello-world Repository，更新了代码并用 arc diff 创建 revision，而且 review 指定为 yhluo2。

```
[root@localhost hello-world]# git checkout -b br
Switched to a new branch 'br'
[root@localhost hello-world]# ll
total 24
-rw-r--r--. 1 root root 59 Oct 9 17:42 build.sh
-rw-r--r--. 1 root root 79 Oct 9 17:42 hello.c
-rw-r--r--. 1 root root 96 Oct 9 17:42 hello.java
-rw-r--r--. 1 root root 169 Oct 9 17:42 point.java
-rw-r--r--. 1 root root 916 Oct 9 17:42 pom.xml
-rw-r--r--. 1 root root 102 Oct 9 17:46 readme.md
[root@localhost hello-world]# git rm pom.xml
rm 'pom.xml'
[root@localhost hello-world]# git add .
[root@localhost hello-world]# git commit -m "delete pom.xml"
[br 514be15] delete pom.xml
1 file changed, 30 deletions(-)
delete mode 100644 pom.xml
[root@localhost hello-world]# arc diff

delete pom.xml

Summary: delete pom.xml

Test Plan: run jenkins

Reviewers: yhluo2

Subscribers: yhluo|
```

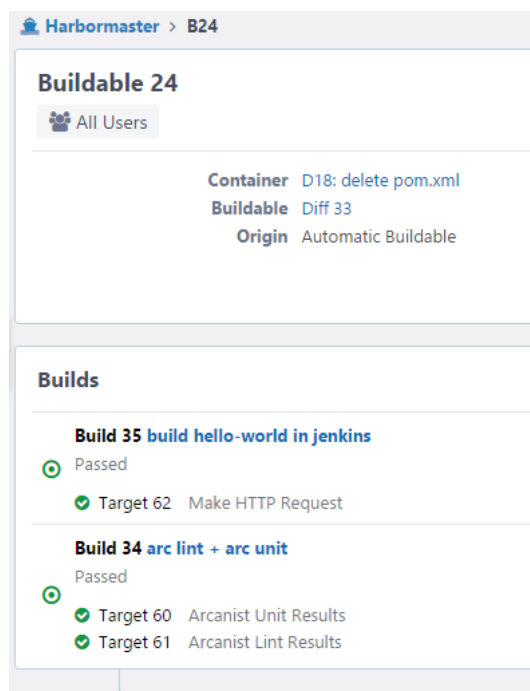
这时满足 Herald 规则 Run jenkins 的条件，就会 run build plan: build hello-world in jenkins。build hello-world in jenkins 其实就是向 Jenkins 发出一个带参数的 HTTP POST

请求。从而激活 Jenkins hello-world job。Jenkins hello-world 就从指定的 Git 库中拉取代码，并逐步执行 Build 部分指定的操作，当所有 Build 部分都成功完成或者某个步骤出错时，进入 Post-build actions 部分，即将 build 结果返回给 Phabricator。可以在 Jenkins Build 控制台输出查看构建的详细步骤。

Console Output

```
Started by remote host 10.61.3.157
Building in workspace /var/lib/jenkins/workspace/hello-world
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url ssh://git@10.61.3.157/diffusion/A/hello-world.git # timeout=10
Fetching upstream changes from ssh://git@10.61.3.157/diffusion/A/hello-world.git
> git --version # timeout=10
> git -c core.askpass=true fetch --tags --progress ssh://git@10.61.3.157/diffusion/A/hello-world.git
+refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision a347d612f41066f8761db6263bf22e45ed6565b2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f a347d612f41066f8761db6263bf22e45ed6565b2
First time build. Skipping changelog.
[hello-world] $ /bin/sh -xe /tmp/hudson5909803496688916237.sh
+ sh ./build.sh
+ echo diff-id=33
diff-id=33
+ echo phid=PHID-HMBT-djrm3im5jc3ewuuhdqd2
phid=PHID-HMBT-djrm3im5jc3ewuuhdqd2
[phabricator:uberalls] No cobertura results found
[phabricator:process-build-result] No unit results available.
[phabricator:process-build-result] No coverage provider available.
[phabricator:harbormaster] Sending Harbormaster BUILD_URL via PHID: PHID-HMBT-djrm3im5jc3ewuuhdqd2
[phabricator:process-build-result] Sending build result to Harbormaster with PHID PHID-HMBT-djrm3im5jc3ew
[phabricator:comment-file] no files found by path: '.phabricator-comment'
Finished: SUCCESS
```

可以在 <http://<your-server>/harbormaster/query/all> 处查看 Jenkins 返回的结果：



The screenshot shows the Harbormaster B24 interface. At the top, it says "Harbormaster > B24". Below that, it displays "Buildable 24" with a "All Users" button. The main section shows build details for "Build 35 build hello-world in jenkins", which is marked as "Passed". It lists "Target 62 Make HTTP Request". Below that, it shows "Build 34 arc lint + arc unit", also marked as "Passed", with "Target 60 Arcanist Unit Results" and "Target 61 Arcanist Lint Results".

四、 Reference

[Phabricator User Documentation](#)

[Use Jenkins](#)

[Jenkins The Definitive Guide](#)

[Jenkins Plugins](#)

[Phabricator Differential Plugin](#)