

软件测试教程 持续集成jenkins篇

本课程主要讲解持续集成工具jenkins的使用。

持续集成的核心价值在于：

- 1、持续集成中的任何一个环节都是自动完成的，无需太多的人工干预，有利于减少重复过程以节省时间、费用和工作量；
- 2、持续集成保障了每个时间点上团队成员提交的代码是能成功集成的。换言之，任何时间点都能第一时间发现软件的集成问题，使任意时间发布可部署的软件成为了可能；
- 3、持续集成还能利于软件本身的发展趋势，这点在需求不明确或是频繁性变更的情景中尤其重要，持续集成的质量能帮助团队进行有效决策，同时建立团队对开发产品的信心。

在敏捷时代，持续集成的作用越来越突出。本节主要讲解如下部分：

jenkins的特点

jenkins的安装与启动

jenkins的使用

jenkins与sonarqube的集成

jenkins的特点

Jenkins是一个软件界非常流行的开源CI服务器，jenkins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，功能包括：

- 持续的软件版本发布/测试项目。
- 监控外部调用执行的工作

Jenkins能实时监控集成中存在的错误，提供详细的日志文件和提醒功能，还能用图表的形式形象地展示项目构建的趋势和稳定性。

使用Jenkins的益处有：

-易安装、易配置

-基于Web访问，用户界面非常友好、直观和灵活

-Jenkins是基于Java开发的，但它不仅限于构建基于Java的软件

-变更支持：Jenkins能从代码仓库（Subversion/GIT）中获取并产生代码更新列表并输出到编译输出信息中

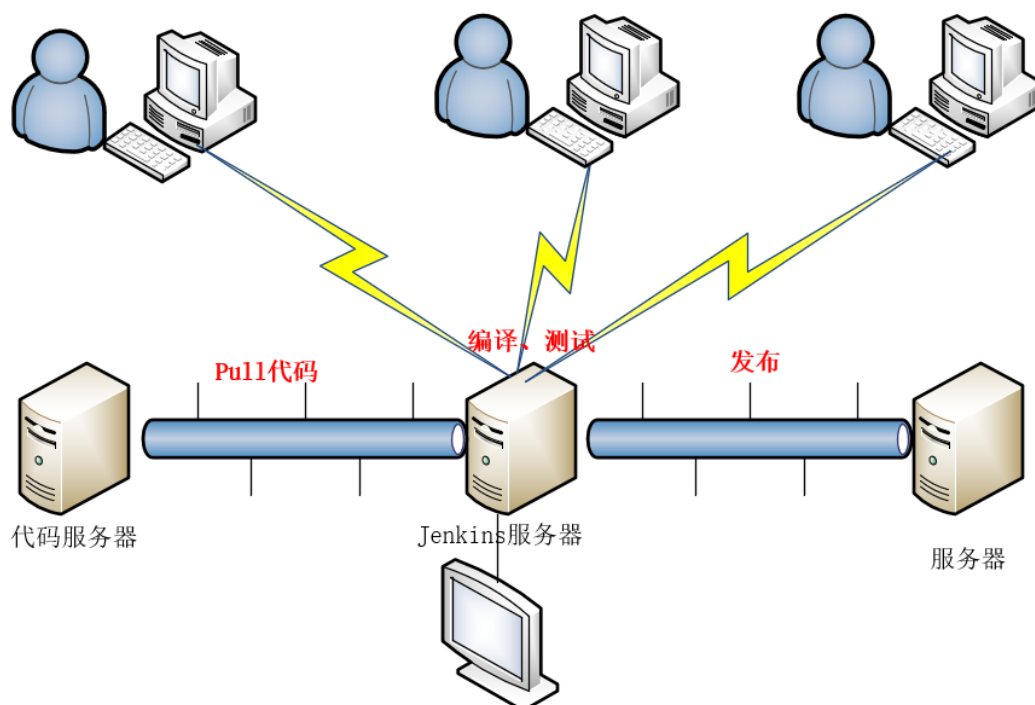
-测试报告：也就是用以图表等形式提供详细的测试报表功能

-拥有大量的插件：这些插件极大的扩展了Jenkins的功能

典型的工作流：

1. pull

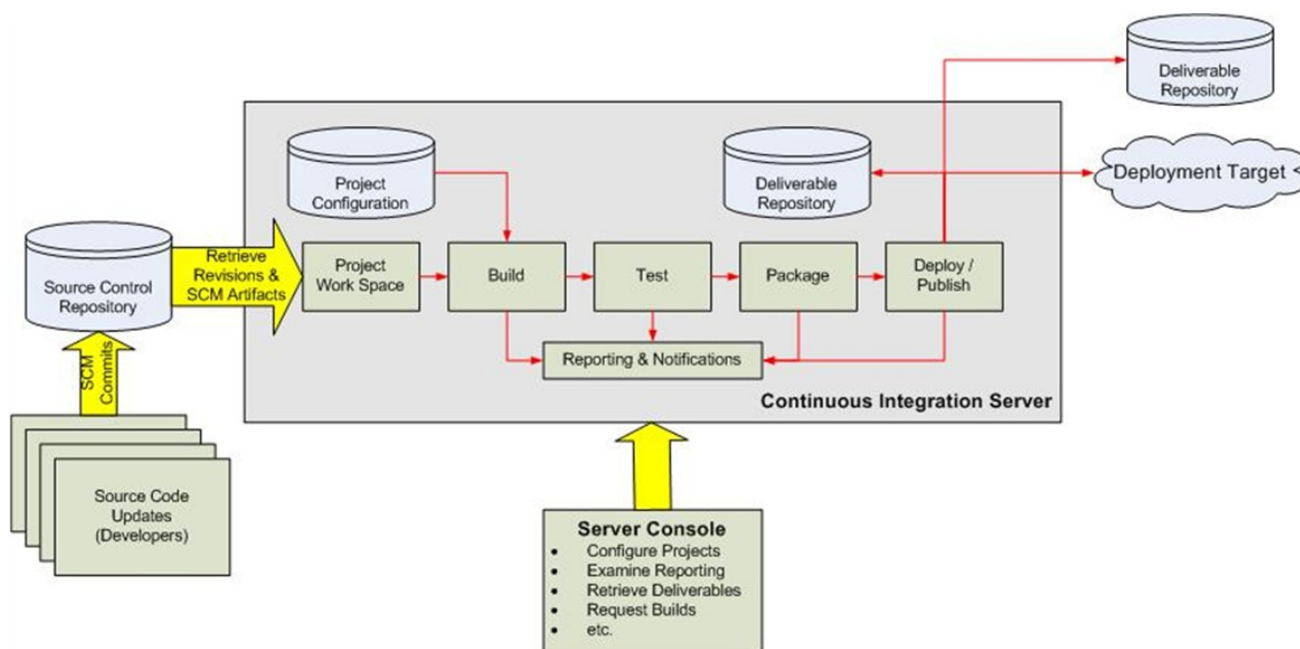
2. 编译
3. 测试
4. 发布



Jenkins会定时获取最新的代码，自动运行你的编译脚本，编译成功后，再运行你的测试脚本，这一步成功后，接着它会帮你把新程序发布出去，特别的，在最后一步，你可以选择手动发布，或自动发布，毕竟发布这件事情，还是需要人为的确认一下比较好。简而言之

Jenkins可以帮你在写完代码后，一键完成开发过程中的一系列工作

使用Jenkins的好处显而易见，它减少了你的重复劳动。更重要的是，一个团队的开发流程一开始是不一致的，不一致往往会带来各种各样的问题，最终体现在软件的质量或开发效率不够高，而Jenkins会帮你规范大家的行为，从而避免一系列的问题。



jenkins的安装和启动

jenkins官网地址: <https://jenkins.io/>

建议下载rpm包程序

注意: 不要在中文目录下运行

JDK安装

1.运行java-version, 会发现Linux自带的OpenJDK, 运行java -version查看系统自带的JDK:

```
java version "1.7.0"
```

```
OpenJDK Runtime Environment (build 1.7.0-b09)
```

```
OpenJDK 64-Bit Server VM (build 1.7.0-b09, mixed mode)
```

2.通过 rpm -qa | grep java查看JDK相关的包,

然后通过命令删除下面三个相关的包: rpm -e --nodeps java-1.7.0-openjdk-1.7.0.75-2.5.4.2.el7_0.x86_64

```
rpm -e --nodeps java-1.7.0-openjdk-headless-1.7.0.75-2.5.4.2.el7_0.x86_64
```

```
rpm -e --nodeps tzdata-java-2015a-1.el7.noarch
```

3.上传jdk-8u20-linux-x64.rpm到服务器;

4.运行rpm -ivh jdk-8u20-linux-x64.rpm;

5.运行vi ~/.bash_profile(不要修改~/profile,该文件为系统配置文件), 在文件末尾输入以下几行:

```
export JAVA_HOME=/usr/java/jdk1.8.0_171-amd64
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bin
```

保存, 退出;

6.运行source ~/.bash_profile, 使~/.bash_profile文件生效, 或者重启;

7.运行java -version, 返回结果如下结果表示安装成功

```
java version "1.8.0_171"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
```

安装rar

1.tar -zxf rarlinux-x64-5.4.0.tar.gz

2.cd rar

make

安装jenkins

1. 下载jenkins

<https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions#InstallingJenkinsonRedHatdistributions-ImportantNoteonCentOSJava>

2. linux服务器下执行如下命令

Installation

Add the Jenkins repository to the yum repos, and install Jenkins from here.

- `sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo`
- `sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key`
- `sudo yum install jenkins`

Installation of a stable version

There is also a LTS YUM repository for the LTS Release Line

- `sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo`
- `sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key`
- `sudo yum install jenkins`

`sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo`

`sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key`

`sudo yum install jenkins`

另一种安装方式：

上传安装包jenkins-2.121.1-1.1.noarch.rpm到服务器上,执行下边命令：

`rpm -ivh jenkins-2.121.1-1.1.noarch.rpm`

3. 接下来配置jenkins端口

`vi /etc/sysconfig/jenkins`

查找JENKINS_PORT, 修改JENKINS_PORT="8080", 默认为"8080"

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# Options to pass to java when running Jenkins.
#
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"

## Type:          integer(0:65535)
## Default:       8080
## ServiceRestart: jenkins
#
# Port Jenkins is listening on.
# Set to -1 to disable
#
JENKINS_PORT="8080"

## Type:          string
## Default:       ""
## ServiceRestart: jenkins
#
# IP address Jenkins listens on for HTTP requests.
# Default is all interfaces (0.0.0.0).
#
JENKINS_LISTEN_ADDRESS=""

## Type:          integer(0:65535)
```

4.启动jenkins

service (systemctl) jenkins restart

默认目录: /var/lib/jenkins

5.登录jenkins 登录地址: <http://192.168.1.x:8080>

1. 提示输入初始密码
2. 在/.../.jenkins/secrets/initialAdminPassword中查取密码输入

vi /.../.jenkins/secrets/initialAdminPassword

3. 复制密码输入
4. 提示创建用户名和密码, 输入admin/admin
5. 根据提示安装插件

备注: 如果在虚拟机上部署, 需要关闭防火墙, 如果在云服务器上需要配置安全组

查看CentOS防火墙信息:

/etc/init.d/iptables status

关闭CentOS防火墙服务:

/etc/init.d/iptables stop

永久关闭防火墙:

chkconfig --level 35 iptables off

CentOS7防火墙的关闭

查看状态

systemctl status firewalld.service

关闭防火墙

systemctl stop firewalld.service

永久关闭防火墙

systemctl disable firewalld.service

maven安装

1. 下载，下载地址：<http://maven.apache.org/download.html>
2. 解压到/home/zhangxiaoli/目录下

命令：tar -zxvf apache-maven-3.5.0-bin.tar.gz，解压后的目录为：apache-maven-3.5.0。

3. 建立链接，方便升级：ln -s /home/zhangxiaoli/apache-maven-3.5.0/ maven
4. 在/etc/profile 添加下面两行：

```
export M3_HOME=/home/apache-maven-3.5.0
export PATH=${M3_HOME}/bin:${PATH}
```

5. 执行下面命令，使环境参数生效：

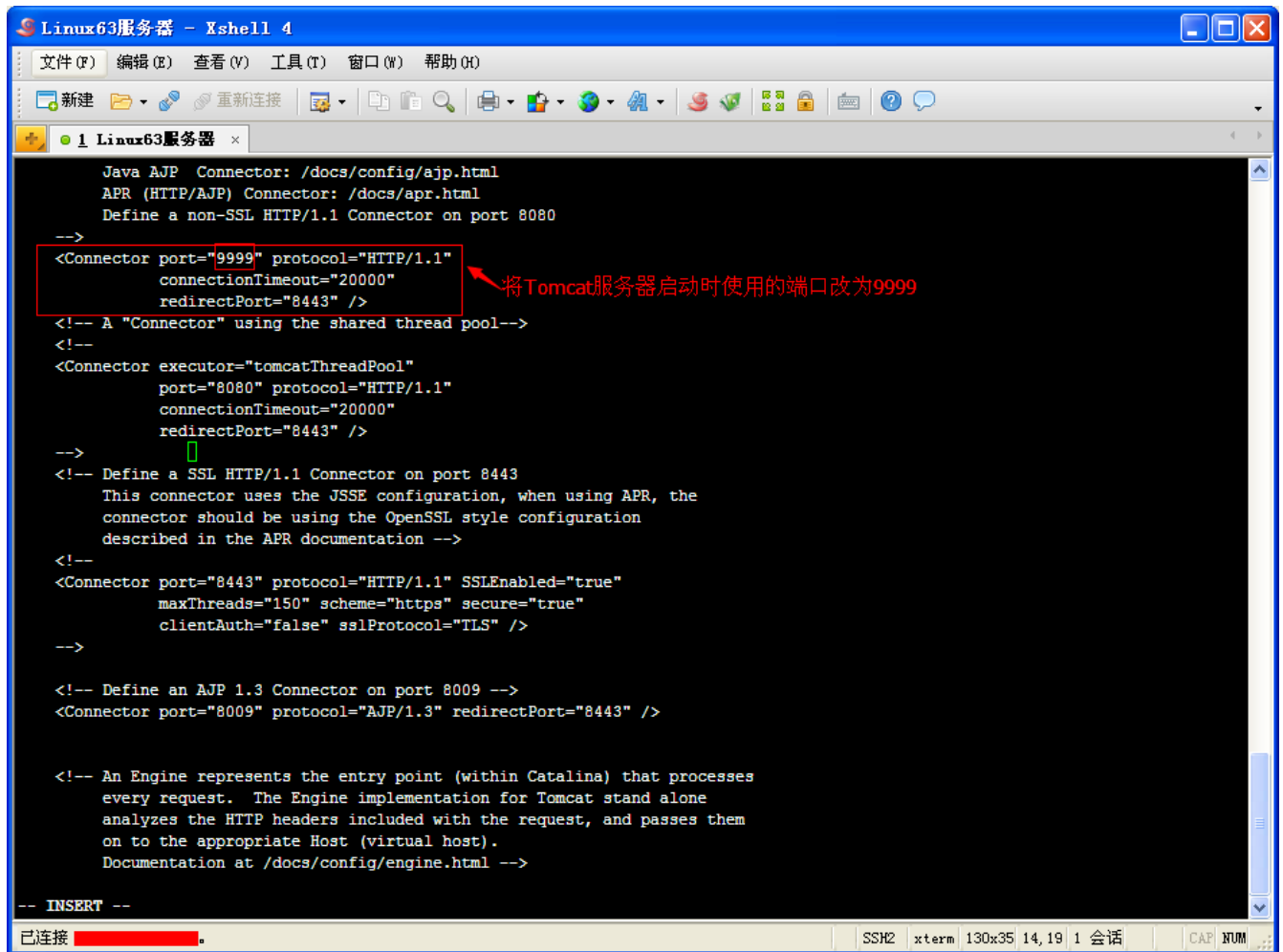
source /etc/profile 6.查看mvn版本，看是否配置正确。

命令：mvn -version

安装TOMCAT

tomcat上传到要安装的服务器上

1. mkdir tomcat
2. cp apache-tomcat-8.0.30.rar tomcat
3. cd tomcat
4. unrar x apache-tomcat-8.0.30.rar
5. vi server.xml 修改tomcat端口8081（默认8080）



```
Linux63服务器 - Xshell 4
文件(F) 编辑(E) 查看(V) 工具(T) 窗口(W) 帮助(H)
新建 重新连接
Linux63服务器 x
Java AJP Connector: /docs/config/ajp.html
APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="9999" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
  port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
-->
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS" />
-->

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

<!-- An Engine represents the entry point (within Catalina) that processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html -->

-- INSERT --
已连接
```

将Tomcat服务器启动时使用的端口改为9999

6. 配置/etc下的profile文件 vi /etc/profile

```
JAVA_HOME=/usr/java/jdk1.8.0_171-amd64
JRE_HOME=/usr/java/jdk1.8.0_171-amd64/jre
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin
```

7. Tomcat服务器启动和关闭

7.1 启动Tomcat服务器

进入tomcat服务器的bin目录，然后执行"./startup.sh"命令启动Tomcat服务器，如下图所示：



```
[root@rh6 apache-tomcat-6.0.36]# cd bin
[root@rh6 bin]# ls
bootstrap.jar  catalina-tasks.xml  cpappend.bat  setclasspath.bat  shutdown.sh  tomcat-juli.jar  tool-wrapper.sh
catalina.bat  commons-daemon.jar  digest.bat    setclasspath.sh   startup.bat   tomcat-native.tar.gz  version.bat
catalina.sh    commons-daemon-native.tar.gz  digest.sh     shutdown.bat      startup.sh    tool-wrapper.bat     version.sh
[root@rh6 bin]# ./startup.sh
Using CATALINA_BASE: /home/apache-tomcat-6.0.36
Using CATALINA_HOME: /home/apache-tomcat-6.0.36
Using CATALINA_TMPDIR: /home/apache-tomcat-6.0.36/temp
Using JRE_HOME: /usr/java/jdk1.7.0_60/jre
Using CLASSPATH: /home/apache-tomcat-6.0.36/bin/bootstrap.jar
```

进入tomcat服务器的bin目录

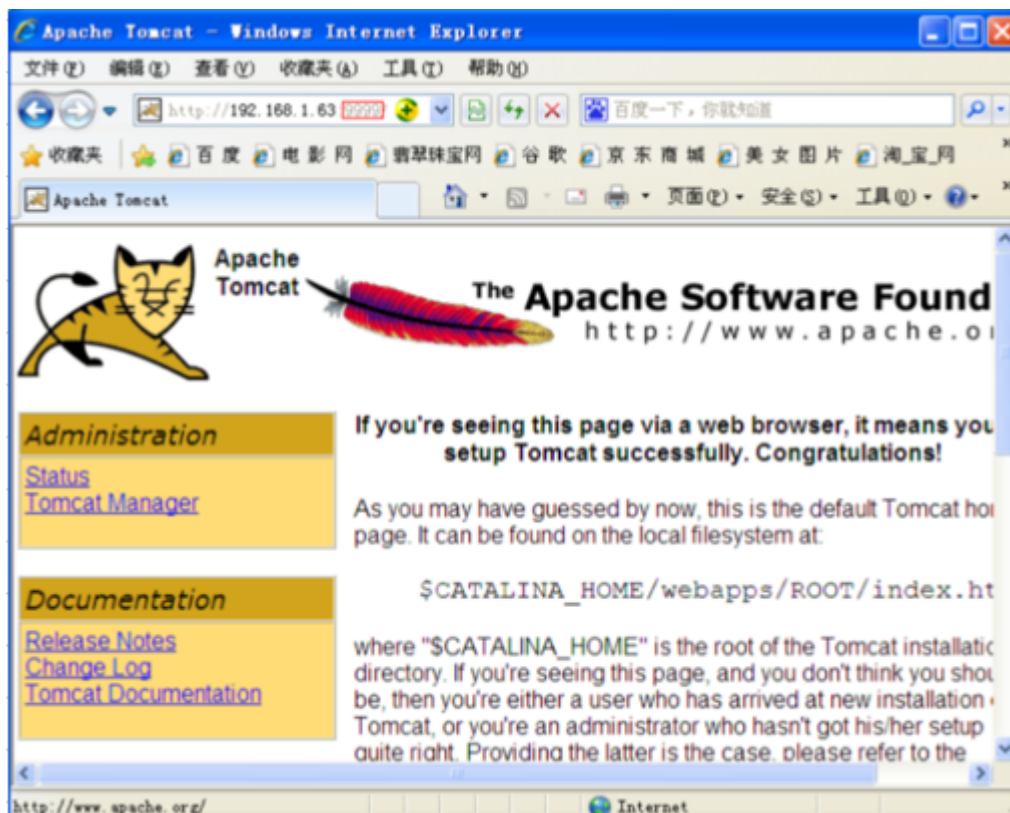
启动Tomcat服务器

7.2 查看tomcat服务器的Log信息，看看tomcat服务器是否已经正常启动，进入tomcat服务器下的logs目录，打开catalina.out文件进行查看，如下图所示：

```
[root@rh6 apache-tomcat-6.0.36]# cd logs
[root@rh6 logs]# ls
catalina.2014-11-14.log catalina.out host-manager.2014-11-14.log localhost.2014-11-14.log manager.2014-11-14.log
[root@rh6 logs]# cat catalina.out
十一月 14, 2014 2:43:04 下午 org.apache.catalina.core.AprLifecycleListener init
信息: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the
java.library.path: /usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
十一月 14, 2014 2:43:05 下午 org.apache.coyote.http11.Http11Protocol init
信息: Initializing Coyote HTTP/1.1 on http-9999
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.Catalina load
信息: Initialization processed in 505 ms
十一月 14, 2014 2:43:05 下午 org.apache.catalina.core.StandardService start
信息: Starting service Catalina
十一月 14, 2014 2:43:05 下午 org.apache.catalina.core.StandardEngine start
信息: Starting Servlet Engine: Apache Tomcat/6.0.36
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory examples
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory manager
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory host-manager
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory docs
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory ROOT
十一月 14, 2014 2:43:05 下午 org.apache.coyote.http11.Http11Protocol start
信息: Starting Coyote HTTP/1.1 on http-9999
十一月 14, 2014 2:43:05 下午 org.apache.jk.common.ChannelSocket init
信息: JK: ajp13 listening on /0.0.0.0:8009
十一月 14, 2014 2:43:05 下午 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/17 config=null
十一月 14, 2014 2:43:05 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 580 ms
```

进入到tomcat服务器目录下的logs目录查看tomcat服务器的运行log
查看 catalina.out 这个文件，里面记录了 tomcat 服务器的启动 log
tomcat 服务器启动时使用的端口号
tomcat 服务器启动完成

7.3 访问Tomcat服务器，如下图所示：



jenkins的使用

系统配置

1) 基本设置

工作空间根目录：job文件存放的目录

构建记录根目录：每一次构建日志目录


执行者数量：同时执行的任务数

Subversion：配置svn版本

Extended E-mail Notification&&邮件通知：配置邮件通知服务器

2) 全局工具配置

配置jdk, ant, maven, git等版本,此处可以配置多个，以支持不同项目使用

全局工具配置

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK 安装

新增 JDK

JDK

别名

JDK1.8.0

JAVA_HOME

/usr/java/jdk1.8.0_171-amd64

☐ 自动安装

删除 JDK

新增 JDK

系统下JDK 安装列表

Git

系统下SonarQube Scanner for IDE安装列表

SonarQube Scanner

SonarQube Scanner 安装

新增 SonarQube Scanner

SonarQube Scanner

Name

sonarscanner

SONAR_RUNNER_HOME

/home/sonar-scanner-3.0.3.778-linux

☐ 自动安装

删除 SonarQube Scanner

新增 SonarQube Scanner

系统下SonarQube Scanner 安装列表

Ant

Maven

Maven 安装

新增 Maven

Maven

Name

maven

MAVEN_HOME

/home/apache-maven-3.5.0

☐ 自动安装

删除 Maven

新增 Maven

系统下Maven 安装列表

3) 管理插件

安装、卸载、升级插件

4) 全局安全配置

配置授权策略

建议采用Jenkins专用数据库，安全策略采用安全矩阵方式，已经能够基本满足现有的模式。

安全域：

1、Jenkins专用户数据库

2、LDAP

授权策略：

1、安全矩阵

| Overall | Credentials | Agent | Job | Run | View | SCM |

Agents：

当需要设置slave时，TCP port for JNLP agents需要启用，否则不同通过java web启动代理

5) 管理用户

增加jenkins新用户

6) 系统信息

查看系统环境信息

7) 管理节点

Jenkins的分布式构建，在Jenkins的配置中叫做节点。

当我们使用多台服务器时，可通过jenkins的节点配置，将jenkins项目发布在不同服务器上编译、部署，这就形成了jenkins的分布式

简单解释一下配置：

名字：该节点的名字。

描述：说明这个节点的用途。

of executors：允许在这个节点上并发执行任务的数量，一般设置为 cpu 支持的线程数。

远程工作目录：节点上 Jenkins 的根目录。

标签：分配给这个节点的标签。

用法：节点的使用策略。

启动方法：启动 agent 的方式，对于 windows 平台，最好选择 "通过Java Web启动代理"。

对于linux平台，选择**Launch slave agents via SSH**

Availability：Jenkins 控制 slave 是否在线的策略。

Node Properties：设置工具列表和环境变量

8) 读取设置

放弃当前内存中所有的设置信息并从配置文件中重新读取 仅用于当您手动修改配置文件时重新读取设置。

###视图

用于分类管理，使job分类清晰

新建视图

- 1、在Jenkins主界面中点击图示的【+】开始执行视图创建工作
- 2、在【新建视图】页面，按照图示填写“视图名称”，选择“List View”点击【OK】按钮
- 3、在【视图配置】页面中，我们可以给当前的视图添加描述性信息，添加完成之后，点击【保存】按钮

修改视图

- 1、点击【编辑视图】按钮，进入【编辑视图】页面
- 2、在【编辑视图】页面，将我们所修改的编写完之后，点击【保存】按钮，即可实现编辑视图的操作

删除视图

- 1、点击【删除视图】按钮，即可实现视图的删除操作，删除视图不会影响job

job

一个JOB就是一个任务。

在jenkins主页面，选择新建，可以新建不同种类的job

输入一个任务名称

» 该字段不能为空，请输入一个合法的名称



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。



构建一个maven项目

构建一个maven项目。Jenkins利用你的POM文件,这样可以大大减轻构建配置。



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。



External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.



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。



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

构建一个自由风格的软件项目：jenkins最常用的类型，一般没有特殊需求可以选择它

构建一个maven项目：当安装了maven插件之后，会出现该项目，仅针对maven构建方式的项目

流水线：jenkins2.0+之后推广的方式，通过编写yaml文件的方式来实现自动构建、部署等操作

构建一个多配置项目：当需要多环境时，必须多环境编译、部署时可以采用该配置

Mutibranch Pipeline:Multibranch Pipeline类型的job好处就是可以自动扫描git工程所有分支，并创建对应的job(前提是分支根目录中包含有Jenkinsfile文件)

新建一个自由风格的项目后job的页面如下：

General | 源码管理 | 构建触发器 | 构建环境 | 构建 | 构建后操作

项目名称: empty

描述: [Plain text] 预览

☐ Throttle builds

☐ 丢弃旧的构建

☒ 参数化构建过程

String Parameter

名字: server

默认值: 127.0.0.1

描述: Git server address

[Plain text] 预览

JOB主要包含以下部分：

- 1) 项目名称和描述
- 2) 项目安全
- 3) 参数化构建过程：是否升级db、选择部署环境等
- 4) 源码管理：从SVN、GIT等获取代码
- 5) 构建触发器：是否与其他JOB有依赖关系，是否使用定时器等
- 6) 构建：调用ant、maven、shell等脚本
- 7) 日志查看，点击console output可以查看日志信息

job执行过程如图：



常见配置项解释：

项目名称与描述：名称不建议为中文，项目（任务）名称不能重复

丢弃旧的构建：设置构建历史的保存策略，可以节省空间，可以按天数或者个数来设置

参数化构建过程：某些构建过程，需要一些输入参数，常用的有choice parameter，string parameter等

Restrict where this project can be run：当jenkins配置有slave时，可以设置job在该slave上运行

github project：github使用，里面配置响应的url和需要显示的名称就可以了

throttle builds: 节流构建, 通过设置时间段内允许并发的次数来实现构建的控制

参数化构建过程: 里面可以配置不同的参数, 便于在构建时引用这些参数

windows引用方式%paraname%,shell为\$paraname

关闭构建: 项目无法进行构建

安静期: 设置一个时间来间隔每次构建的间隔

重试次数: 拉取源码重试的次数

该项目的上游项目正在构建时阻止该项目构建与该项目的下游项目正在构建时阻止该项目构建: 用于上下游项目有关联的构建策略

使用自定义的工作空间: 指定当前任务的workspace, 否则默认为JENKINS_HOME的工作目录

保留构建的依赖日志

源码管理: 选择使用git或者svn, 以git为例:

以svn为例时

repository url:填写仓库的地址

Credentials: 这里需要配置拉取svn源码的用户名和密码

Local module directory: 具体的项目的路径, 默认从根目录拉取

Additional Credentials: 增加额外认证

Check-out Strategy: 代码检出策略

Use 'svn update' as much as possible: 相当于svn update

Always check out a fresh copy: checkout之前先清除工作区文件

源码库浏览器: 这里默认就可以了

构建触发器:

- 1、触发远程构建 (例如,使用脚本): 这里使用于自动化构建, 拼接url后写入代码中可以实现在脚本或者工具执行构建
- 2、Build after other projects are built:构建与其他项目构建后, 用于上下游项目有关联的时候
- 3、Build periodically: 定时执行构建

日程表的参数:

第一个参数代表的是分钟 minute, 取值 0~59;

第二个参数代表的是小时 hour, 取值 0~23;

第三个参数代表的是天 day, 取值 1~31;

第四个参数代表的是月 month, 取值 1~12;

最后一个参数代表的是星期 week, 取值 0~7, 0 和 7 都是表示星期天。

4、Build when a change is pushed to GitHub:这个是github项目的触发规则

5、Poll SCM：设置定时检查代码仓库是否有变更，有变更则构建

构建环境：

1、Delete workspace before build starts：在构建之前清空工作空间

2、Abort the build if it's stuck：如果构建出现问题则终止构建

3、Add timestamps to the Console Output：给控制台输出增加时间戳

4、Use secret text(s) or file(s)：使用加密文件或者文本

构建

1、execute windows batch command：执行windows的cmd

2、execute shell：执行shell命令

3、invoke ant：调用ant ,调用ant的执行脚本来进行构建

备注：

- Targets：主要是执行ant脚本中哪几个部分，可以添加多个；
- Build File:需要指定Ant脚本的物理位置；
- Properties：添加Ant指定的属性；
- Java Options：设置运行java时的属性，例如内存、堆大小等；

4、invoke gradle script：调用gradle脚本，来帮助我们自动打包

5、invoke top-level maven targets：调用maven

不同的语言可能会采用不同的构建方式

构建后操作

1、build other projects:构建其他项目

2、e-mail notification:发送邮件

3、delete workspace when build is done:构建后删除工作空间

运行并监控构建作业

当配置完成一个任务后，回到主控制面板：

1、列表列举现在已经配置的任务已经任务当前的状态





2、左边有构建队列，当有构建时，会把当前正在构建的队列在上面进行列举

3、当一个任务配置完成后，可以采用手动构建和触发器构建两种方式，在项目验证阶段，可以通过手动触发方式，点击任务区的“立即构建”，会在Build History中出现进度条






4、点击进度条，可以进入到具体的编译过程

任务构建状态

当前构建状态分为以下几种：

	项目构建完成，同时被认为是稳定的
	项目构建完成，但被认定为不稳定
	构建失败
	作业已经禁止

构建稳定行，Jenkins会基于一些后处理器任务为构建发布一个稳健指数(从0-100)，越高越稳定

	构建成功率>80%
	构建成功率 60%-79%
	构建成功率 40%-59%
	构建成功率 20%-39%
	构建成功率 0-19%

实践案例一

案例：TEST.py

以一个python为样例(代码已上传到码云)

- 1、安装Python,findbugs插件
- 2、新建-构建一个自由风格的软件项目
- 3、源码管理选择git，没有的话就不选择
- 4、构建触发器，选择poll scm，设置触发时间，例如H 5 * * *，表示每晚5点开始运行Job

poll scm的触发条件是：晚上5点，查看scm是否有更新，有更新则开始构建

- 5、构建：选择构建方式

选择执行shell

- 6、构建后操作：选择发布到远程服务器上

- 7、进行构建

8、查看构建日志

- 1.通过本地git客户端将python代码上传到码云 https://gitee.com/sunraylily/test_for_jenkins.git
- 2.创建一个自由项目并进行配置

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。

**构建一个maven项目**

构建一个maven项目,Jenkins利用你的POM文件,这样可以大大减轻构建配置。

**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**

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**文件夹**

可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器,而文件夹则是一个独立的命名空间。因此你可以有多个相同名称的文件夹。只要它们在不同的文件夹里即可。

配置general

General

源码管理构建触发器构建环境构建构建后操作

描述

python

[纯文本] 预览

☐ GitHub project

☐ Throttle builds

☒ 丢弃旧的构建

策略

Log Rotation

保持构建的天数

10

如果非空，构建记录将保存此天数

保持构建的最大个数

30

如果非空，最多此数目的构建记录将被保存

高级...

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

保存

应用

高级...

配置源码管理

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

https://gitee.com/sunraylily/test_for_jenkins

Credentials

- 无 -

添加

高级...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Add Branch

源码库浏览器

(自动)

Additional Behaviours

新增

构建触发器、构建环境配置

General

源码管理

构建触发器

构建环境

构建

构建后操作

构建触发器

☐ 触发远程构建 (例如,使用脚本)

☐ GitHub hook trigger for GITScm polling

☐ 其他工程构建后触发

☐ 定时构建

☒ 轮询 SCM

日程表

H 5 ***

上次运行的时间 2018年7月3日 星期二 上午05时15分12秒 CST; 下次运行的时间 2018年7月4日 星期三 上午05时15分12秒 CST.

忽略钩子 post-commit ☐

构建环境

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Prepare SonarQube Scanner environment

构建配置

执行 shell

命令

```
$sudo cd /var/lib/jenkins/workspace/python/  
$sudo python TEST.py  
$sudo scp /var/lib/jenkins/workspace/python/TEST.py /home/testwork/
```

查看 [可用的环境变量列表](#)

高级...

Execute SonarQube Scanner

Task to run

scan

JDK

JDK1.8.0

JDK to be used for this SonarQube analysis

Path to project properties

Analysis properties

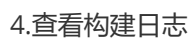
sonar.projectKey=testSonar
sonar.projectName=python
sonar.projectVersion=1.0
sonar.language=python
sonar.java.binaries=.
sonar.sources=.
sonar.sourceEncoding=UTF-8

保存

应用

构建后操作配置

3.进行构建



Jenkins

python

工程 python

pylon

SonarQube

工作区

最新修改记录

Build History

构建历史

Find Bugs Trend

SonarQube Quality Gate

cms OK

server-side processing: Success

相关链接

- 最近一次构建(#6) 47 分之前
- 最近稳定构建(#6) 47 分之前
- 最近成功的构建(#6) 47 分之前
- 最近完成的构建(#6) 47 分之前

Jenkins

python

#6

控制台输出

```
由用户 unknown or anonymous 启动
构建中 在工作空间 /var/lib/jenkins/workspace/python 中
> /usr/local/git/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> /usr/local/git/bin/git config remote.origin.url https://gitee.com/sunraylily/test_for_jenkins # timeout=10
Fetching upstream changes from https://gitee.com/sunraylily/test_for_jenkins
> /usr/local/git/bin/git --version # timeout=10
> /usr/local/git/bin/git fetch --tags --progress https://gitee.com/sunraylily/test_for_jenkins +refs/heads/*:refs/remotes/origin/*
> /usr/local/git/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> /usr/local/git/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 6658cf1f9eb06f7b5f086b944b0f541e164c6d94 (refs/remotes/origin/master)
> /usr/local/git/bin/git config core.sparsecheckout # timeout=10
> /usr/local/git/bin/git checkout -f 6658cf1f9eb06f7b5f086b944b0f541e164c6d94
Commit message: "新建 TEST.py"
> /usr/local/git/bin/git rev-list --no-walk 6658cf1f9eb06f7b5f086b944b0f541e164c6d94 # timeout=10
[python] $ /bin/sh -xe /tmp/jenkins8501006928558409428.sh
+ cd /var/lib/jenkins/workspace/python/
+ python TEST.py
hello world
+ scp /var/lib/jenkins/workspace/python/TEST.py /home/testwork/
[python] $ /home/sonar-scanner-3.0.3.778-linux/bin/sonar-scanner scan -Dsonar.host.url=http://192.168.126.129:9000 ***** -Dsonar.language=java -
Dsonar.projectName=cms -Dsonar.projectVersion=1.0 -Dsonar.sourceEncoding=UTF-8 -Dsonar.projectKey=testSonar -Dsonar.sources=. -Dsonar.java.binaries=. -
Dsonar.projectBaseDir=/var/lib/jenkins/workspace/python
INFO: Scanner configuration file: /home/sonar-scanner-3.0.3.778-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarQube Scanner 3.0.3.778
INFO: Java 1.8.0_121 Oracle Corporation (64-bit)
INFO: Linux 2.6.32-642.el6.x86_64 amd64
```

1. 修改代码为错误的代码，执行查看控制台日志
2. 修改定时脚本，查看定时构建
3. 使用TEST.py先本地运行看效果后，在jenkins运行

实践案例二

文件压缩项目(代码已上传到码云)

- 1、安装需要的插件
- 2、新建-构建一个自由风格的软件项目
- 3、源码管理选择git，没有的话就不选择
- 4、构建触发器，选择poll scm，设置触发时间，例如H 5 * * *，表示每晚5点开始运行Job

poll scm的触发条件是：晚上5点，查看scm是否有更新，有更新则开始构建

5、构建：选择构建方式

调用顶层Maven目标

6、构建后操作：选择发布到远程服务器上

7、进行构建


8、查看构建日志


1.通过本地git客户端将代码上传到码云<https://gitee.com/sunraylily/file>


2.创建一个自由项目并进行配置


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.


**构建一个maven项目**
构建一个maven项目. Jenkins利用你的POM文件, 这样可以大大减轻构建配置.

**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**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**文件夹**
确定 一个可以嵌套存储的容器. 利用它可以进行分组. 视图仅仅是一个过滤器, 而文件夹则是一个独立的命名空间. 因此你可以有多个相同名称的内容, 只要它们在不同的文件夹里即可.

配置General

General

源码管理

构建触发器

构建环境

构建

构建后操作

描述

test289

[纯文本] [预览](#)

☐ GitHub project

☐ Throttle builds

☒ 丢弃旧的构建

策略

Log Rotation

保持构建的天数

10

如果非空，构建记录将保存此天数

保持构建的最大个数

20

如果非空，最多此数目的构建记录将被保存

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

☐ 限制项目的运行节点

保存

应用

高级...

高级...

配置源码

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

https://gitee.com/sunraylily/file

Credentials

- 无 -

添加

高级...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

添加分支

源码库浏览器

(自动)

Additional Behaviours

新增

☐ Subversion

构建触发器

构建触发器

☐ 触发远程构建 (例如,使用脚本)

☐ 其他工程构建后触发

☐ 定时构建

☐ GitHub hook trigger for GITScm polling

☒ 轮询 SCM

日程表

H/2 * * * * *

上次运行的时间 Wednesday, July 11, 2018 10:58:28 PM CST; 下次运行的时间 Wednesday, July 11, 2018 10:58:28 PM CST.

忽略钩子 post-commit

☐

?

?

?

?

?

?

构建环境

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Prepare SonarQube Scanner environment

保存

应用

?

?

?

构建

构建

执行 shell

命令

```
$sudo cd /var/lib/jenkins/workspace/file289
$sudo g++ Test.cpp -o compress
$sudo ./compress
$sudo scp -r /var/lib/jenkins/workspace/file289 /home/testwork
```

查看 [可用的环境变量列表](#)

高级...

Execute Python script

Script

```
#coding utf-8
import os

file1='/var/lib/jenkins/workspace/file289/Input.txt.huffman'
file2='/var/lib/jenkins/workspace/file289/Input.txt.unhuffman'

if os.path.exists(file1) and os.path.exists(file2):

    print "compress success"
else:

    print "compress fail"
```

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

保存

应用

shell脚本

```
$sudo cd /var/lib/jenkins/workspace/file289 $sudo g++ Test.cpp -o compress $sudo ./compress $sudo scp -r /var/lib/jenkins/workspace/file289 /home/testwork
```

python脚本:

```
#coding utf-8
```

```
import os
```

```
file1='/var/lib/jenkins/workspace/file289/Input.txt.huffman'
```

```
file2='/var/lib/jenkins/workspace/file289/Input.txt.unhuffman'
```

```
if os.path.exists(file1) and os.path.exists(file2):
```

```
    print "compress success"
```

```
else:
```

```
    print "compress fail"
```

构建后操作

构建后操作

Publish FindBugs analysis results

FindBugs results

[Fileset includes](#) setting that specifies the generated raw FindBugs XML report files, such as `**/findbugs.xml` or `**/findbugsXml.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/findbugsXml.xml` or `**/findbugs.xml` are used for maven or ant builds, respectively. Be sure not to include any non-report files into this pattern.

Use rank as priority ☐

Uses the bug rank when evaluating the priority of the warnings (otherwise the FindBugs priority is used).

高级...

增加构建后操作步骤

保存

应用

3.立即构建

Jenkins

Jenkins > python >

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

Build History

构建历史

#	时间
#6	2018-7-3 下午7:29
#5	2018-7-3 下午7:27
#4	2018-7-3 下午6:21
#2	2018-7-3 上午11:39
#1	2018-7-3 上午11:34

RSS 全部 RSS 失败

工程 python

python

- SonarQube
- 工作区
- 最新修改记录

SonarQube Quality Gate

cms **OK**

server-side processing: **Success**

相关链接

- 最近一次构建(#6).47 分之前
- 最近稳定构建(#6).47 分之前
- 最近成功的构建(#6).47 分之前
- 最近完成的构建(#6).47 分之前

FindBugs Trend

count

Enlarge Co

4.查看日志

Jenkins

python

工程 python

pylon

SonarQube

工作区

最新修改记录

Build History

构建历史

FindBugs Trend

SonarQube Quality Gate

cms OK

server-side processing: Success

相关链接

- 最近一次构建(#6) 47 分之前
- 最近稳定构建(#6) 47 分之前
- 最近成功的构建(#6) 47 分之前
- 最近完成的构建(#6) 47 分之前

Jenkins

python

#6

控制台输出

```
由用户 unknown or anonymous 启动
构建中 在工作空间 /var/lib/jenkins/workspace/python 中
> /usr/local/git/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> /usr/local/git/bin/git config remote.origin.url https://gitee.com/sunraylily/test_for_jenkins # timeout=10
Fetching upstream changes from https://gitee.com/sunraylily/test_for_jenkins
> /usr/local/git/bin/git --version # timeout=10
> /usr/local/git/bin/git fetch --tags --progress https://gitee.com/sunraylily/test_for_jenkins +refs/heads/*:refs/remotes/origin/*
> /usr/local/git/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> /usr/local/git/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 6658cf1f9eb06f7b5f086b944b0f541e164c6d94 (refs/remotes/origin/master)
> /usr/local/git/bin/git config core.sparsecheckout # timeout=10
> /usr/local/git/bin/git checkout -f 6658cf1f9eb06f7b5f086b944b0f541e164c6d94
Commit message: "新建 TEST.py"
> /usr/local/git/bin/git rev-list --no-walk 6658cf1f9eb06f7b5f086b944b0f541e164c6d94 # timeout=10
[python] $ /bin/sh -xe /tmp/jenkins8501006928558409428.sh
+ cd /var/lib/jenkins/workspace/python/
+ python TEST.py
hello world
+ scp /var/lib/jenkins/workspace/python/TEST.py /home/testwork/
[python] $ /home/sonar-scanner-3.0.3.778-linux/bin/sonar-scanner scan -Dsonar.host.url=http://192.168.126.129:9000 ***** -Dsonar.language=java -
-Dsonar.projectName=cms -Dsonar.projectVersion=1.0 -Dsonar.sourceEncoding=UTF-8 -Dsonar.projectKey=testSonar -Dsonar.sources=. -Dsonar.java.binaries=. -
-Dsonar.projectBaseDir=/var/lib/jenkins/workspace/python
INFO: Scanner configuration file: /home/sonar-scanner-3.0.3.778-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarQube Scanner 3.0.3.778
INFO: Java 1.8.0_121 Oracle Corporation (64-bit)
INFO: Linux 2.6.32-642.el6.x86_64 amd64
```

实践案例三

基于MongoDB的分布式图片文件服务器(代码已上传到码云)

- 1、安装maven,Deploy war/ear to a containers插件
- 2、新建-构建一个自由风格的软件项目
- 3、源码管理选择git, 没有的话就不选择
- 4、构建触发器, 选择poll scm, 设置触发时间, 例如H 5 * * *,表示每晚5点开始运行Job

poll scm的触发条件是: 晚上5点, 查看scm是否有更新, 有更新则开始构建

- 5、构建: 选择构建方式

调用顶层Maven目标

6、构建后操作：选择发布到远程服务器上

7、进行构建

8、查看构建日志

1.通过本地git客户端将代码上传到码云<https://gitee.com/secondriver/small-file-server-base-servlet>

2.创建一个自由项目并进行配置

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。

**构建一个maven项目**
构建一个maven项目。Jenkins利用你的POM文件，这样可以大大减轻构建配置。

**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**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**文件夹**
确定
可以放置存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间。因此你可以有多个相同名称的只要它们在不同的文件夹里即可。

配置general

General

源码管理构建触发器构建环境构建构建后操作

描述

java

[纯文本] 预览

☐ GitHub project

☐ Throttle builds

☒ 丢弃旧的构建

策略

Log Rotation

保持构建的天数

10

如果非空，构建记录将保存此天数

保持构建的最大个数

20

如果非空，最多此数目的构建记录将被保存

高级...

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

☐ 限制项目的运行节点

保存

应用

高级...

配置源码

源码管理

☐ 无

☒ Git

Repositories

Repository URL

https://gitee.com/secondriver/small-file-server-base-servlet

Credentials

- 无 -

Add

高级...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/tester

X

Add Branch

源码库浏览器

(自动)

Additional Behaviours

新增

☐ Subversion

构建触发器

构建触发器

- ☐ 触发远程构建 (例如,使用脚本)
- ☐ 其他工程构建后触发
- ☐ 定时构建
- ☐ GitHub hook trigger for GITScm polling
- ☒ 轮询 SCM

日程表

30 20 * * 1-7

⚠ Spread load evenly by using 'H 20 * * 1-7' rather than '30 20 * * 1-7'

上次运行的时间 Wednesday, July 11, 2018 8:30:46 PM CST; 下次运行的时间 Thursday, July 12, 2018 8:30:46 PM CST.

忽略钩子 post-commit ☐

30 20 * * 1-7

构建

构建

Execute SonarQube Scanner

Task to run

scan

JDK

JDK1.8.0

JDK to be used for this SonarQube analysis

Path to project properties

Analysis properties

sonar.projectKey=javaSonar
sonar.projectName=JAVAfile
sonar.projectVersion=1.0
sonar.language=java
sonar.sources=/var/lib/jenkins/workspace/javaproject1
sonar.sourceEncoding=UTF-8

Additional arguments

JVM Options

sonar.projectKey=javaSonar sonar.projectName=JAVAfile sonar.projectVersion=1.0 sonar.language=java
sonar.sources=/var/lib/jenkins/workspace/javaproject1 sonar.sourceEncoding=UTF-8

调用顶层 Maven 目标

X

?

Maven 版本

maven

▼

目标

clean compile findbugs:findbugs install -e

▼

POM

?

属性

maven.test.skip=true

?

Java虚拟机参数

▼

?

注入构建变量

☐

?

使用Maven私有仓库

☐

?

配置文件

Use default maven settings

▼

?

全局配置文件

Use default maven global settings

▼

?

Goals:clean compile findbugs:findbugs install -e

或 POM: /var/lib/jenkins/workspace/javaproject1/pom.xml

Properties:maven.test.skip=true

构建后操作

构建后操作

Publish FindBugs analysis results

FindBugs results

[Fileset includes](#) setting that specifies the generated raw FindBugs XML report files, such as `**/findbugs.xml` or `**/findbugsXml.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/findbugsXml.xml` or `**/findbugs.xml` are used for maven or ant builds, respectively. Be sure not to include any non-report files into this pattern.

Use rank as priority

☐

Uses the bug rank when evaluating the priority of the warnings (otherwise the FindBugs priority is used).

高级...

Deploy war/ear to a container

WAR/EAR files

**/*.war

Context path

Containers

Tomcat 8.x

Credentials

tomcat/*****

▼

Add

Tomcat URL

http://192.168.144.128:8081

Add Container ▼

Deploy war/ear to a container, 用于将构建后生成的war包部署至tomcat服务器:

WAR/EAR files: target/javatest.war,相对于项目我们的路径

Contextpath: 用于配置项目访问路径, 如填/RMS_Server则表示项目的根访问目录为: http://localhost:8080/RMS_Server


tomcat URL就是你希望把war包部署到的tomcat所在IP地址，最后面不需要再加斜杠

add container: 增加容器, 一般选tomcat 8X就可以。这里的username与password需要到tomcat的conf文件夹中的tomcat-users.xml修改。

```
<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-status"/>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
<user username="tomcat" password="tomcat" roles="manager-gui,manager-script,manager-
status"/>
```

Deploy on failure: 用于配置当前构建失败时是否仍然部署至tomcat，默认不选

3.进行构建



Jenkins

python

返回面板

状态

修改记录

工作空间

立即构建

删除工程


配置

SonarQube


重命名

工程 python

python

[SonarQube](#)

[工作区](#)

[最新修改记录](#)

SonarQube Quality Gate

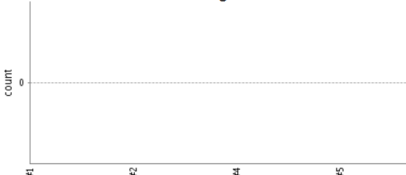
cms OK

server-side processing: Success

相关链接

- 最近一次构建(#6).47分之前
- 最近稳定构建(#6).47分之前
- 最近成功的构建(#6).47分之前
- 最近完成的构建(#6).47分之前

FindBugs Trend



Count

0

10

20

30

40

50

60

70

80

90

100

110

120

130

140

150

160

170

180

190

200

210

220

230

240

250

260

270

280

290

300

310

320

330

340

350

360

370

380

390

400

410

420

430

440

450

460

470

480

490

500

510

520

530

540

550

560

570

580

590

600

610

620

630

640

650

660

670

680

690

700

710

720

730

740

750

760

770

780

790

800

810

820

830

840

850

860

870

880

890

900

910

920

930

940

950

960

970

980

990

1000

1010

1020

1030

1040

1050

1060

1070

1080

1090

1100

1110

1120

1130

1140

1150

1160

1170

1180

1190

1200

1210

1220

1230

1240

1250

1260

1270

1280

1290

1300

1310

1320

1330

1340

1350

1360

1370

1380

1390

1400

1410

1420

1430

1440

1450

1460

1470

1480

1490

1500

1510

1520

1530

1540

1550

1560

1570

1580

1590

1600

1610

1620

1630

1640

1650

1660

1670

1680

1690

1700

1710

1720

1730

1740

1750

1760

1770

1780

1790

1800

1810

1820

1830

1840

1850

1860

1870

1880

1890

1900

1910

1920

1930

1940

1950

1960

1970

1980

1990

2000

2010

2020

2030

2040

2050

2060

2070

2080

2090

2100

2110

2120

2130

2140

2150

2160

2170

2180

2190

2200

2210

2220

2230

2240

2250

2260

2270

2280

2290

2300

2310

2320

2330

2340

2350

2360

2370

2380

2390

2400

2410

2420

2430

2440

2450

2460

2470

2480

2490

2500

2510

2520

2530

2540

2550

2560

2570

2580

2590

2600

2610

2620

2630

2640

2650

2660

2670

2680

2690

2700

2710

2720

2730

2740

2750

2760

2770

2780

2790

2800

2810

2820

2830

2840

2850

2860

2870

2880

2890

2900

2910

2920

2930

2940

2950

2960

2970

2980

2990

3000

3010

3020

3030

3040

3050

3060

3070

3080

3090

3100

3110

3120

3130

3140

3150

3160

3170

3180

3190

3200

3210

3220

3230

3240

3250

3260

3270

3280

3290

3300

3310

3320

3330

3340

3350

3360

3370

3380

3390

3400

3410

3420

3430

3440

3450

3460

3470

3480

3490

3500

3510

3520

3530

3540

3550

3560

3570

3580

3590

3600

3610

3620

3630

3640

3650

3660

3670

3680

3690

3700

3710

3720

3730

3740

3750

3760

3770

3780

3790

3800

3810

3820

3830

3840

3850

3860

3870

3880

3890

3900

3910

3920

3930

3940

3950

3960

3970

3980

3990

4000

4010

4020

4030

4040

4050

4060

4070

4080

4090

4100

4110

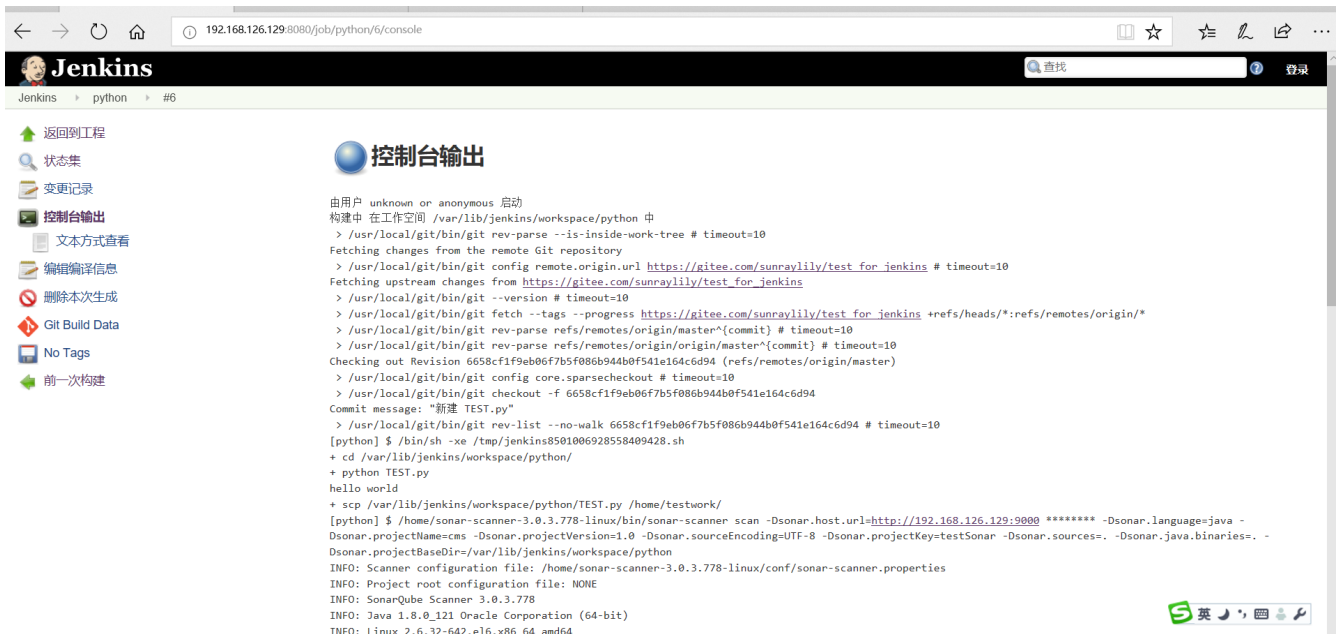
4120

4130

4140

4150</

4.查看构建日志



5.验证部署部的包

- 1.进入/home/apache-tomcat-8.0.30/webapps
- 2.修改ROOT为ROOT1,再修改small-file-server-base-servlet-1.0.0.war 为ROOT.war （没有指定路径）
- 3.启动tomcat
- 4.访问项目：<http://192.168.144.131:8081/>



基于MongoDB的分布式图片文件服务器

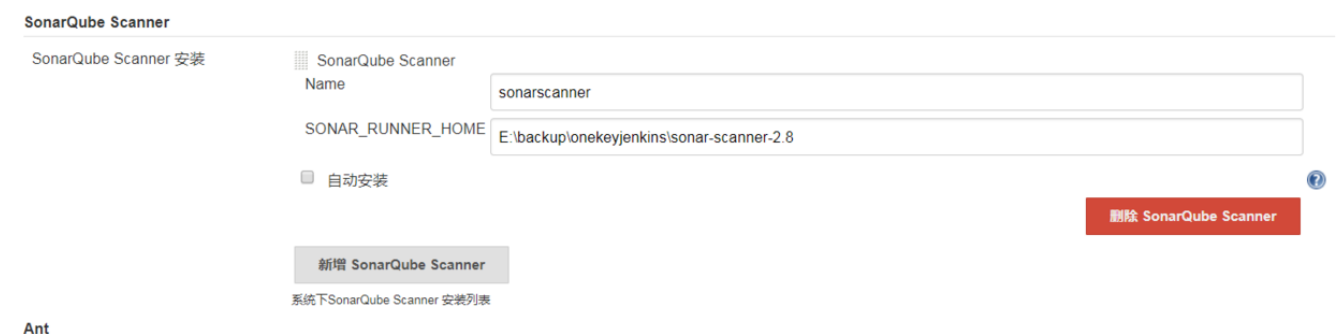
[点击查看开发接口文档](#)

文件上传: 浏览...

Name(名称)	ID (编号)	contentType (内容类型)	size (大小 单位: bytes)	uploadDate (上传时间)	md5 (摘要信息)	操作
三角形1.jpg	5b46ba56cb02882f5b4d120e	image/jpeg	167,040	2018-07-12T10:17:58.938	512ad0dfbb5b154cd27f16e59b851b91	查看 删除

jenkins与sonarqube的集成

- 1、安装[SonarQube Scanner for Jenkins](#)插件
- 2、进入系统设置-全局工具设置，配置SonarQube Scanner路径



3、进入系统设置-系统管理，配置SonarQube servers，增加一个server，配置NAME,以及Server URL

SonarQube servers

Environment variables

SonarQube installations

☐ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Name

sonarserver

Server URL

http://localhost:9000

Server version

Default is http://localhost:9000

5.3 or higher

Server authentication token

SonarQube account login

SonarQube account password

Configuration fields depend on the SonarQube server version.

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

高级...

Delete SonarQube

4、在job的配置页面增加构建步骤：Execute SonarQube scanner

Analysis properties（举例）：

```
sonar.projectKey=Test
sonar.projectName=Test
sonar.projectVersion=1.0
sonar.sources=.
sonar.sourceEncoding=UTF-8
sonar.language=java
sonar.java.binaries=.
```

Execute SonarQube Scanner

X

Task to run

?

JDK

(Inherit From Job)

?

JDK to be used for this SonarQube analysis

Path to project properties

?

Analysis properties

sonar.projectKey=Test
sonar.projectName=Test
sonar.projectVersion=1.0
sonar.sources=.
sonar.sourceEncoding=UTF-8
sonar.language=java

?

Additional arguments

▼

?

JVM Options

▼

?

增加构建步骤 ▼

5、job完成后，可以在项目页面，点击sonarqube链接查看扫描结果