Jenkins 入门

作者： **itech**

Blog： http://www.cnblogs.com/itech

Contents

Jenkins 入门 1

作者： itech 1

Blog： http://www.cnblogs.com/itech 1

Jenkins 入门总结 3

Jenkins 安装 4

Jenkins 构建 JavaHelloWorld 7

Jenkins 的配置 13

Jenkins 的 Windows Slave 的配置 16

Jenkins 的 Linux 的 Slave 的配置 23

Jenkins Master/Slave 架构 26

Jenkins 最佳实践 29

Jenkins 中执行 batch 和 Python 30

Jenkins 的授权和访问控制 32

Jenkins 插件之 Perforce 访问 37

Jenkins 插件之 trigger 43

Jenkins 插件之构建与 MSBuild 44

Jenkins 插件之环境变量插件 EnvInject 47

Jenkins 插件之 Workspace cleanup + Copy to slave 51

Jenkins 插件之 VShpere Cloud 53

Jenkins 插件之 Publish Over SSH/CIFS/FTP 55

Jenkins 插件之 Deploy 57

Jenkins 插件之 Dashboard 和 wall display 60

Jenkins 插件之有用 62

Jenkins 入门总结

在网上貌似没有找到 Jenkins 的中文的太多的文档，有 的都是关于 Hudson 的一些零零散散的，所以自己边学习边实践总结了以下系列文章，希望有助于大家对于 Jenkins 的使用。本系列文章是基于我 3 年多的 SCM+build release 经验，总结了最常用的最基本功能，文章基本上来自于 Jenkins 官方网站的英文帮助的翻译和实际的例子和操作的截图，让大家能够快速直观 地学习 Jenkins。

Jenkins 是 Continous Integration Server，是 Hadson 的继续。 类似的产品： CruiseControl，BuildBot，TeamCity，BuildForge 等。

Jenkins 的突出特点：
开源免费；
跨平台，支持所有的平台；
master/slave 支持分布式的 build；
web 形式的可视化的管理页面；

安装配置超级简单；
tips 及时快速的帮助；
已有的 200 多个插件；

Jenkins 的入门教程：
Jenkins 安装
Jenkins 构建 JavaHelloWorld
Jenkins 的配置
Jenkins 的 Windows Slave 的配置
Jenkins 的 Linux 的 Slave 的配置
Jenkins Master/Slave 架构
Jenkins 最佳实践
Jenkins 中执行 batch 和 Python
Jenkins 的授权和访问控制
Jenkins 插件之 Perforce 访问
Jenkins 插件之 trigger
Jenkins 插件之构建与 MSBuild
Jenkins 插件之环境变量插件 EnvInject
Jenkins 插件之 Workspace cleanup + Copy to slave
Jenkins 插件之 VShpere Cloud
Jenkins 插件之 Publish Over SSH/CIFS/FTP
Jenkins 插件之 Deploy
Jenkins 插件之 Dashboard 和 wall display
Jenkins 插件之有用

Jenkins 安装

1 Jenkins
Jenkins 由以前的 hudson 更名而来。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。

具体地：

* 软件的持续构建和测试，此时 Jenkins 与 CruiseControl 或 DamageControl 相似。本质上提供了一个易于使用的持续集成系统，使 得开发人员更容易地将改变集成到工程中，使得用户更容易获得一个新的 build。自动化，持续的构建提高了软件开发的效率。
*监视外部运行 的 job 的执行，例如 cron jobs 或 procmail jobs，即使这些 jobs 是运行在远程的机器上。例如，对于 cron，你将会收到 email 包含 job 的 output，你需要检查 email 来确认是否 job broke。Jenkins 将保持这些 outputs 且使得你更加容易地注意到 job 的 broke。

Jenkins 的主要特点：
* 容易安装，只需要执行 Java -jar jenkins.war， 或者直接部署到一个 servlet container 中，例如 tomcat。不需要安装，不需要数据库的支持。
* 容易配置，jenkins 可以完全地通过友好的 web GUI 来配置，且配置页面支持配置项的错误检查和很好的在线帮助。不需要手动地编辑 xml 的配置文件，但是 jenkins 也支持手动修改 xml 配置文件。
* 项目源码修改的检测，jenkins 能够从项目的 Subversion/CVS 生成最近修改的集合列表，且改方式非常有效，不会增加 Subversion/CVS Repository 的负载。
* 可读的永久的链接生成，jenkins 对于大部分 pages 都生成清楚的可读的永久的链接，例如"latest build"/"latest successful build",因此可以容易地在其他的地方引用 jenkins 的生成的 pages。
* RSS/EMail/IM 集成，可以通过 RSS，EMail 或 IM 来实时地监视 build 的失败。
* Build 完成后仍然可以 tag，支持在 build 完成后 tag 或重 tag。
* Junit/TestNG 测试报告，能够很好地显示各种测试的报告，且可以生成失败的趋向图。
* 分布式 build，jenkins 能够分发 build/test 的负载到多台机器，能够更好地利用硬件资源，提高 build 的时间。
* 文件标识，jenkins 可以标识 build 产生的文件，例如 jars。
* 插件支持，jenkins 可以通过第三方的插件来扩展。
* 跨平台，支持几乎所有的平台，例如 Windows,Ubuntu/Debian,Red Hat/Fedora/CentOS,Mac OS X,openSUSE,FreeBSD,OpenBSD,Solaris/OpenIndiana.Gentoo。

2 jenkins 的安装
下载 jenkins.war, 拷贝到 c:\jenkins 下，然后运行 java -jar jenkins.war. (注意需要先安装 JDK，然后设置 JAVA_HOME 环境变量且将%JAVA_HOME%\bin 加入到 PATH 环境变量中)

运行如下：
```
c:\jenkins>java -jar jenkins.war
Running from: C:\jenkins\jenkins.war
webroot: $user.home/.jenkins
[Winstone 2011/11/02 17:11:27] - Beginning extraction from war file
Jenkins home directory: C:\Users\AAA\.jenkins found at: $user.home/.jenkins
[Winstone 2011/11/02 17:12:57] - HTTP Listener started: port=8080
[Winstone 2011/11/02 17:12:57] - AJP13 Listener started: port=8009
[Winstone 2011/11/02 17:12:58] - Winstone Servlet Engine v0.9.10 running:
controlPort=disabled
Nov 02, 2011 5:12:58 PM jenkins.model.Jenkins$6 onAttained
INFO: Started initialization
Nov 02, 2011 5:13:02 PM jenkins.model.Jenkins$6 onAttained
INFO: Listed all plugins
Nov 02, 2011 5:13:02 PM jenkins.model.Jenkins$6 onAttained
INFO: Prepared all plugins
Nov 02, 2011 5:13:02 PM jenkins.model.Jenkins$6 onAttained
```

INFO: Started all plugins
Nov 02, 2011 5:13:02 PM jenkins.model.Jenkins$6 onAttained
INFO: Augmented all extensions
Nov 02, 2011 5:13:02 PM jenkins.model.Jenkins$6 onAttained
INFO: Loaded all jobs
Nov 02, 2011 5:13:04 PM jenkins.model.Jenkins$6 onAttained
INFO: Completed initialization
Nov 02, 2011 5:13:04 PM hudson.TcpSlaveAgentListener <init>
INFO: JNLP slave agent listener started on TCP port 37157
Nov 02, 2011 5:13:14 PM hudson.WebAppMain$2 run
INFO: Jenkins is fully up and running

访问 http://localhost:8080 , jenkins 的主界面如下：

jenkins 的问题的回答 ： http://jenkins.361315.n4.nabble.com/

Jenkins 构建 JavaHelloWorld

注意：我们知道 Jenkins 通过 master/slave 来支持分布式的 job 运行，这里的 JavaHelloworld 运行在 master，即 Jenkins 所在的机器。

一 Java 的 HelloWorld 程序

Ant 构建脚本：c:\JavaHelloWorld\build.xml

```
<project name="HelloWorld" basedir="." default="main">

    <property name="src.dir"     value="src"/>

    <property name="build.dir"   value="build"/>
    <property name="classes.dir" value="${build.dir}/classes"/>
    <property name="jar.dir"     value="${build.dir}/jar"/>

    <property name="main-class"  value="oata.HelloWorld"/>


    <target name="clean">
        <delete dir="${build.dir}"/>
    </target>

    <target name="compile">
        <mkdir dir="${classes.dir}"/>
        <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
    </target>
```

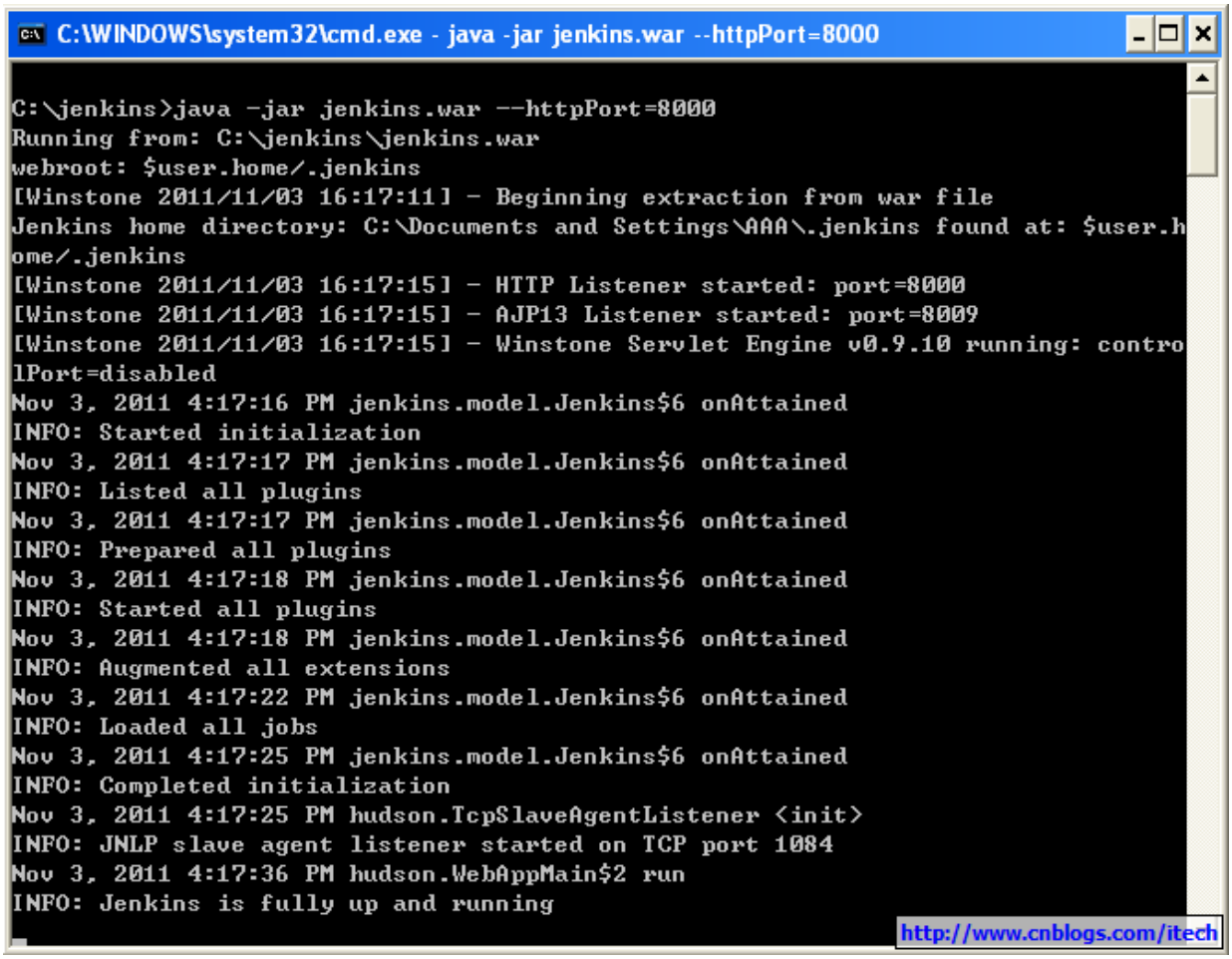
```
    <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}"/>
        <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
            <manifest>
                <attribute name="Main-Class" value="${main-class}"/>
            </manifest>
        </jar>
    </target>

    <target name="run" depends="jar">
        <java jar="${jar.dir}/${ant.project.name}.jar" fork="true"/>
    </target>

    <target name="clean-build" depends="clean,jar"/>

    <target name="main" depends="clean,run"/>

</project>
```

Java 的 helloworld： c:\JavaHelloWorld\src\oata\helloworld.java

```
package oata;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

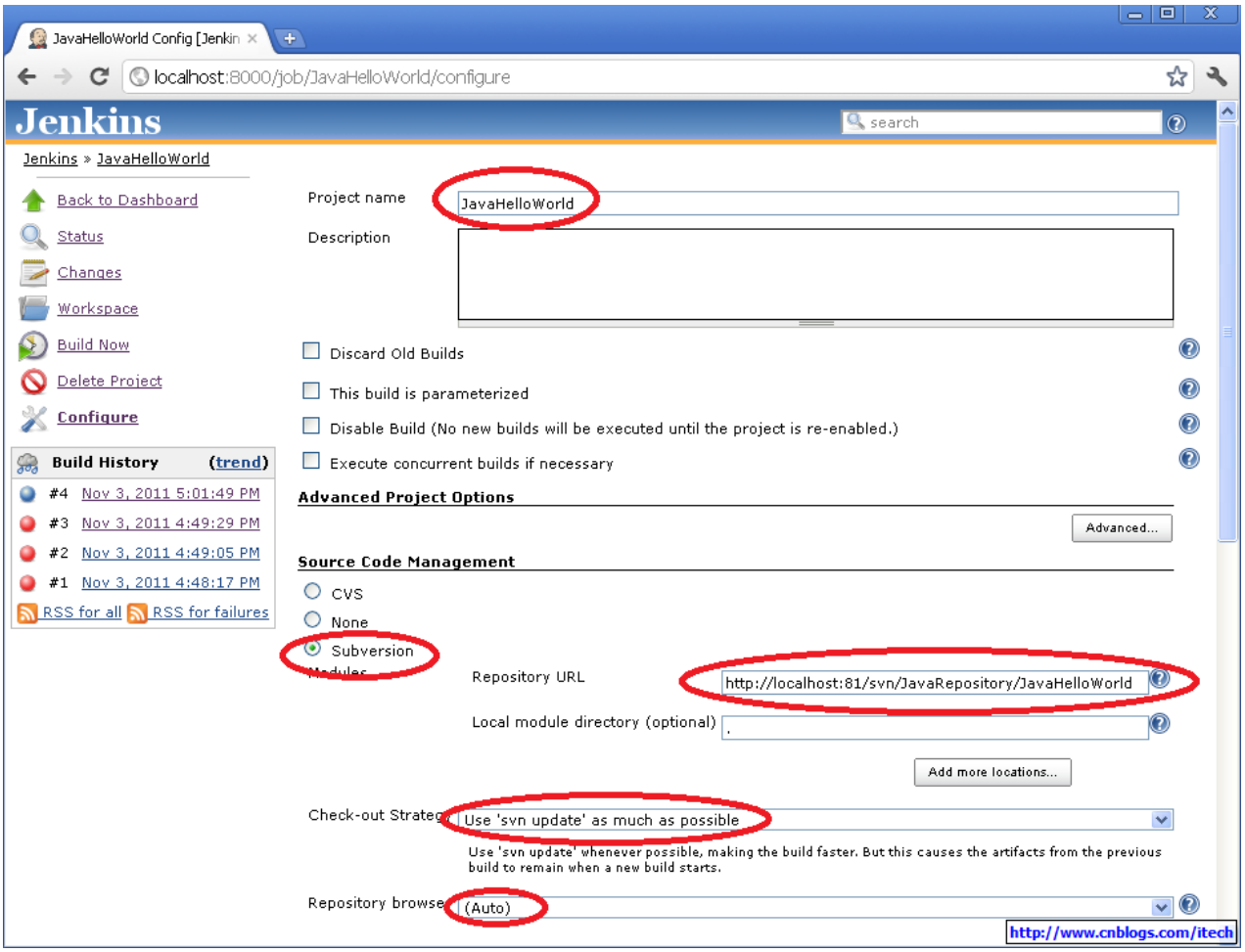
二 启动 Jenkins 且创建 job 来运行 JavaHelloWorld

1） 启动 jenkins 在 8000 端口：



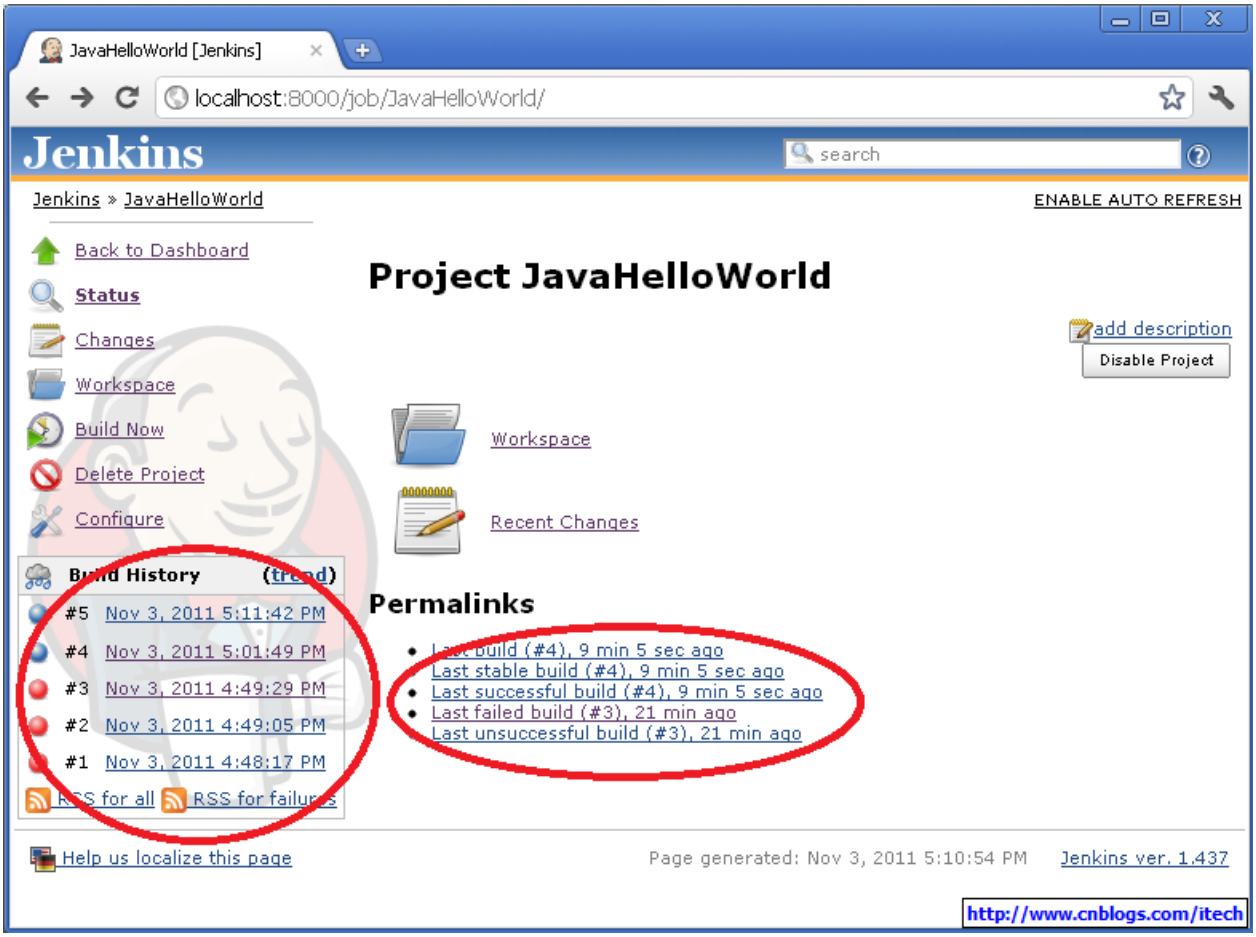
2） 创建 JavaHelloWorld 的 job
在 ie 中打开 http://localhost:8000，
单击 new job 链接，为 javahelloworld 新建 job，且编译 job 的配置如下：

注意 jenkins 默认已经安装了 svn 的 plugin 了。



3） 运行 JavaHelloWorld 的 job

进入 JavaHelloWorld 的主页面，点击 build now 链接进行 build，build 后可以在此主页面上看到所有的 build 历史，如下：



然后还可以点击某个 build 的链接，查看某个 build 的详细日志，如下：

参考：

http://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html

Jenkins 的配置

1 修改 jenkins 的根目录，默认地在 C:\Documents and Settings\AAA\.jenkins 。

.jenkins

```
├─jobs
|  └─JavaHelloWorld
|     ├─builds
|     |  ├─2011-11-03_16-48-17
|     |  ├─2011-11-03_16-49-05
|     |  ├─2011-11-03_16-49-29
|     |  ├─2011-11-03_17-01-49
|     |  └─2011-11-03_17-11-42
|     └─workspace
|        ├─build
|        |  ├─classes
|        |  |  └─oata
|        |  └─jar
|        └─src
|           └─oata
├─plugins
├─usercontent
├─war
```

可以通过设置环境变量来修改，例如：
set JENKINS_HOME=C:\jenkins
然后重新启动 jenkins。

2 备份和恢复 jenkins
只需要备份 JENKINS_HOME 下的所有文件和文件夹，恢复的时候需要先停止 jenkins。

3 移动，删除或修改 jobs
对于移动或删除 jobs，只需要简单地移动或删除%JENKINS_HOEM%\jobs 目录。
对于修改 jobs 的名字，只需要简单地修改%JENKINS_HOEM%\jobs 下对应 job 的文件夹的名字。
对于不经常使用的 job，只需要对%JENKINS_HOEM%\jobs 下对应的 jobs 的目录 zip 或 tar 后存储到其他的地方。

4 可以在 jenkins 的 url 中执行一些命令来操作 jenkins，如下
http://[jenkins-server]/[command] 命令可以为：

- `exit` shutdown jenkins
- `restart` restart jenkins
- `reload` to reload the configuration

5 Jenkins 启动时的命令行参数
--httpPort=$HTTP_PORT，用来设置 jenkins 运行时的 web 端口。
--httpsPort=$HTTP_PORT，表示使用 https 协议。
--httpListenAddress=$HTTP_HOST，用来指定 jenkins 监听的 ip 范围，默认为所有的 ip 都可以访问此 jenkins server。

6 修改 jenkins 的 timezone
如 果 jenkins 所在的 server 的 timezone 不同于用户的 timezone，这时候需要修改 jenkins 的 timezone，需要在 jenkins 启动的时候增加下列参数-Dorg.apache.commons.jelly.tags.fmt.timeZone=TZ

7 最好通过一个脚本来启动 jenkins，确保 jenkins 每次都运行在相同的环境下，例如

```
startjenkins.bat
set JENKINS_HOME=c:\jenkins
cd /d %JENKINS_HOME%
java -jar %JENKINS_HOME%\jenkins.war --httpPort=8000
```

8 jenkins 在后台运行
如果 jenkins 是部署在 servlet 容器中，例如 apache，tomcat 中。因为 servlet 容器一般都在后台运行了，所以 jenkins 也就已经在后台运行了。
对于 windows 用户需要在 jenkins 的管理页面中点击 insall as windows service 来将 jenkins 部署为 service。 但是感觉比较好的方法还是手动将启动 jenkins 的脚本部署为 windows service，从而可以更灵活地设置更多的参数。

9 jenkins 的系统信息
可以在 jenkins 的管理页面下的系统信息中，查看所有的 jenkins 的信息，例如 jenkins 的启动配置，所依赖的系统的环境变量，所安装的 plugins。

10 jenkins 内置的环境变量
BUILD_NUMBER， 唯一标识一次 build，例如 23；
BUILD_ID，基本上等同于 BUILD_NUMBER，但是是字符串，例如 2011-11-15_16-06-21；
JOB_NAME， job 的名字，例如 JavaHelloWorld；
BUILD_TAG， 作用同 BUILD_ID,BUILD_NUMBER,用来全局地唯一标识一此 build，例如 jenkins-JavaHelloWorld-23；
EXECUTOR_NUMBER， 例如 0；
NODE_NAME，slave 的名字，例如 MyServer01；
NODE_LABELS，slave 的 label，标识 slave 的用处，例如 JavaHelloWorld MyServer01；

JAVA_HOME， java 的 home 目录，例如 C:\Program Files (x86)\Java\jdk1.7.0_01；
WORKSPACE，job 的当前工作目录，例如 c:\jenkins\workspace\JavaHelloWorld；
HUDSON_URL = JENKINS_URL， jenkins 的 url，例如 http://localhost:8000/ ；
BUILD_URL，build 的 url 例如 http://localhost:8000/job/JavaHelloWorld/23/；
JOB_URL， job 的 url，例如 http://localhost:8000/job/JavaHelloWorld/；
SVN_REVISION，svn 的 revison， 例如 4；

Jenkins 的 Windows Slave 的配置

参考：
https://wiki.jenkins-ci.org/display/JENKINS/Step+by+step+guide+to+set+up+master+and+slave+machines

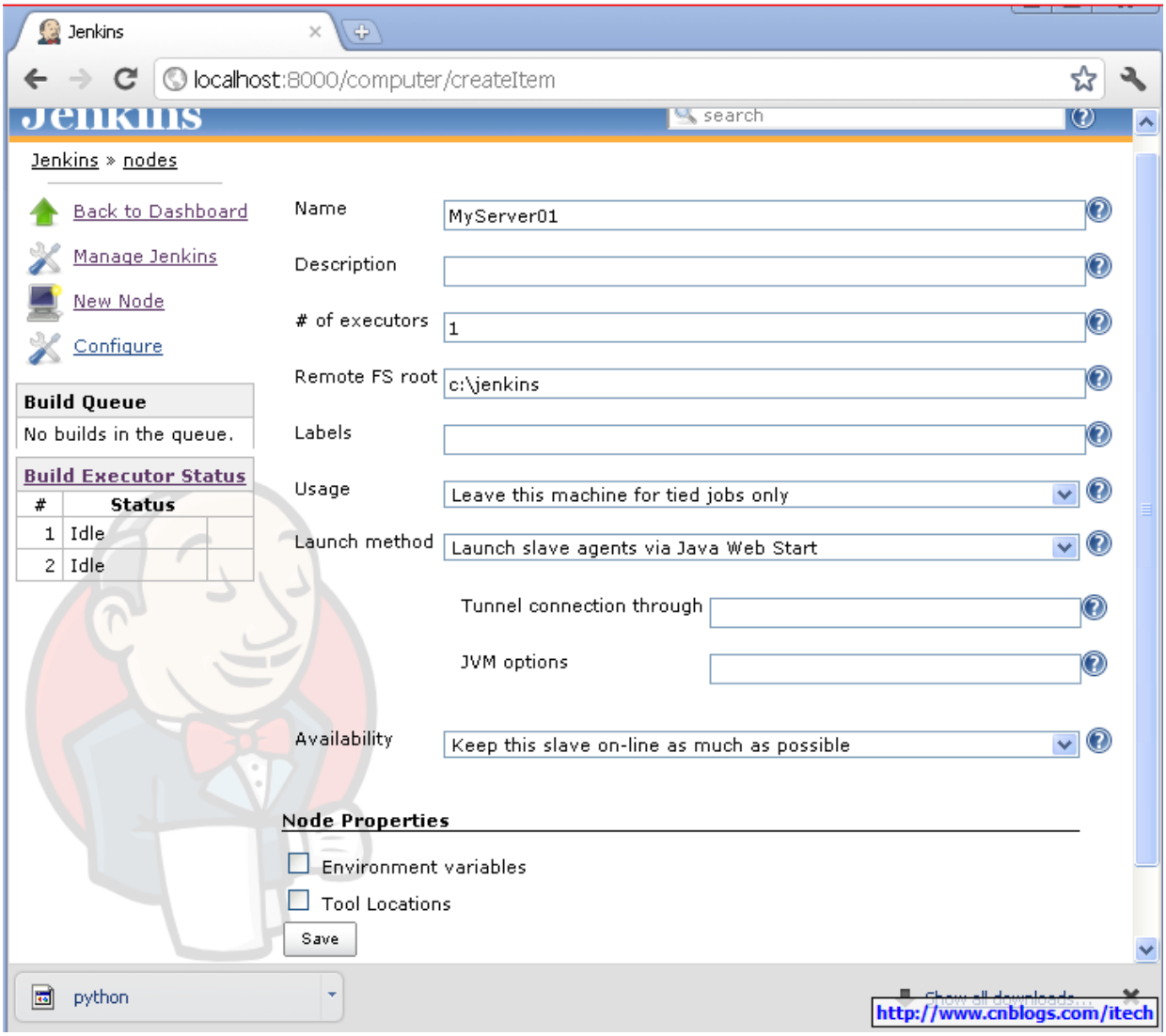
一 创建新的 Slave

注意 Jenkins 中 slave 称为 note。 所以下面文章中的 slave 和 node 指的是一回事。

1）在 Manage Jenkins-->Manage Nodes -->New Node 下：输入 Node Name，且选择 Dumb Slave 作为 Slave 的类型，然后 OK。

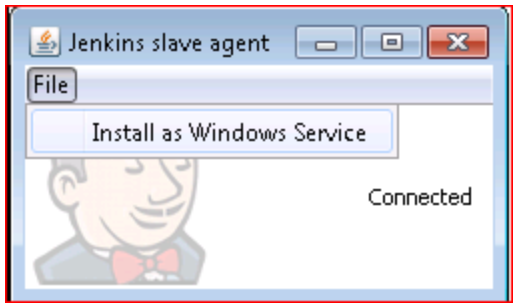
2）在 Slave 的配置页面，输入如下：

*executors 的数量，1 或多个；
*输入 Slave 上的跟目录，例如 c:\jenkins；
*Usage 选择：*Leave this machine for tied jobs only*；
**Lunch Method 选择：Launch slave agents via Java Web Start* ；
* Avaliablitiy 选择：*Keep this slave online as much as possible*；
* *然后保存；*



3）在 slave 所在的机器登录 jenkins master，且进入 Manage Jenkins-->Manage Nodes-->新建的 Note，点击 launch，然后安装 slave 为 service 如下：

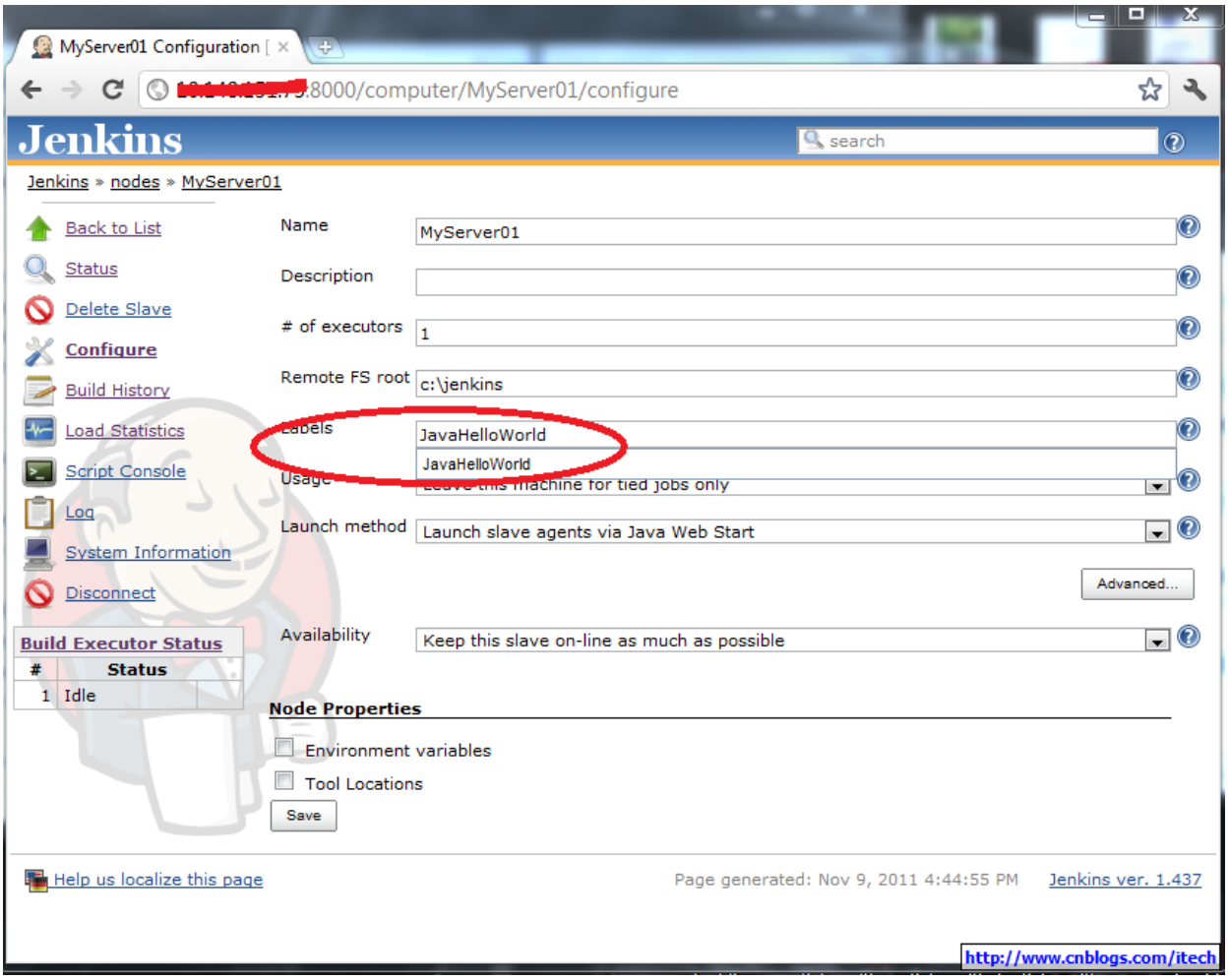
MyServer01 [Jenkins]
Welcome to Chrome
[redacted]8000/computer/MyServer01/
Jenkins
search
Jenkins » nodes » MyServer01
ENABLE AUTO REFRESH
Back to List
Status
Delete Slave
Configure
Build History
Load Statistics
Script Console
Log
System Information
Mark this node temporarily offline
Slave MyServer01
Connect slave to Jenkins one of these ways:
Launch
Launch agent from browser on slave
Run from slave command line:
javaws http://[redacted]:8000/computer/MyServer01/slave-agent.jnlp
Or if the slave is headless:
java -jar slave.jar -jnlpUrl http://[redacted]:8000/computer/MyServer01/slave-agent.jnlp
Build Executor Status
Status
Projects tied to MyServer01
None
Help us localize this page
Page generated: Nov 9, 2011 3:13:19 PM
Jenkins ver. 1.437
http://www.cnblogs.com/itech


Jenkins slave agent
File
Install as Windows Service
Connected

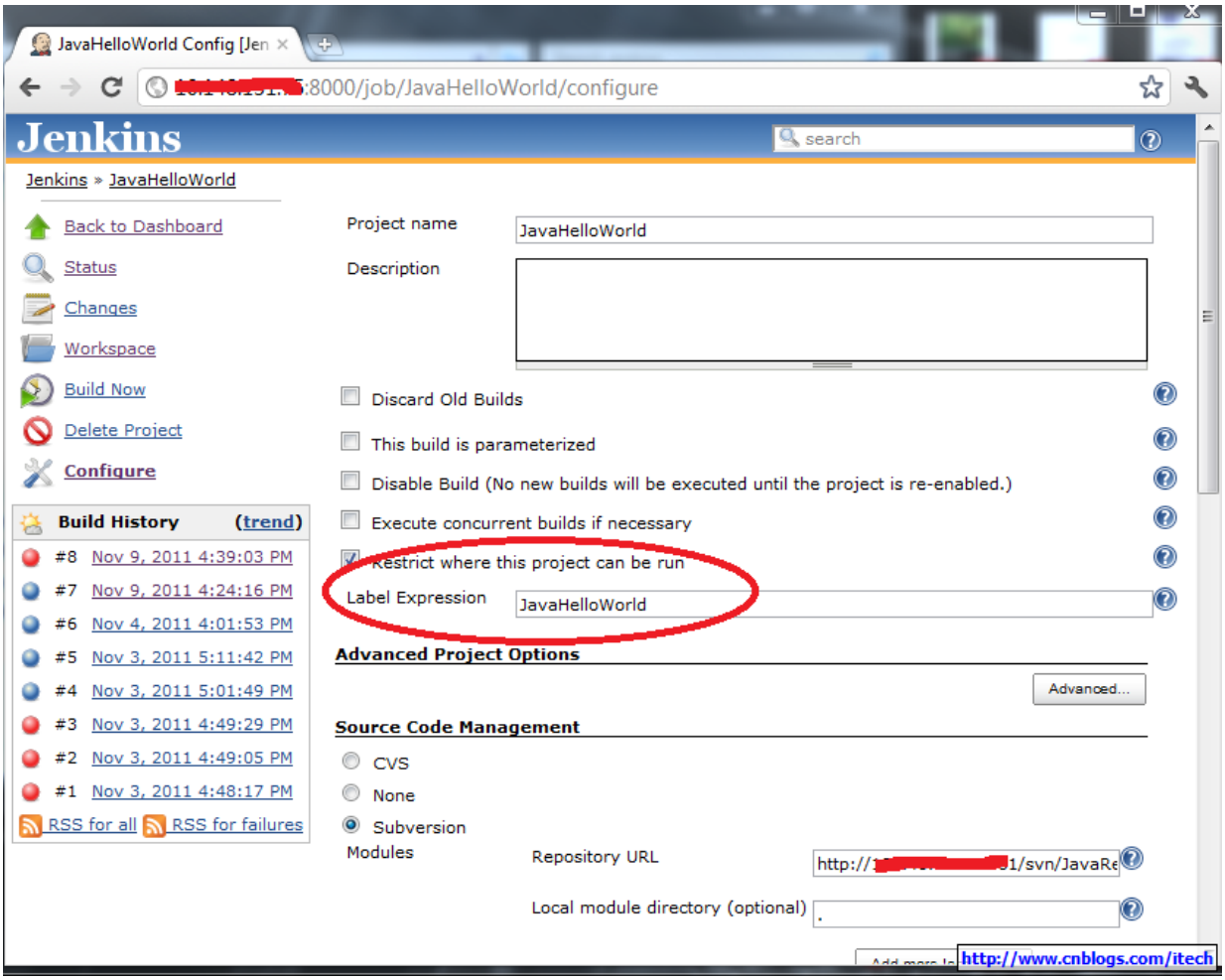
4）安装成功后显示如下：

二 在 slave 上运行 job

对上面的 slave 增加 label，从而表示此 slave 的用处，且同时对 uage 选择 leave this machine for tied jobs only：



对 Jenkins 构建 JavaHelloWorld 中的 job 修改如下：
选择 restrict where this project can be run 且输入 note（slave）的 label。
另外注意 SVN 的地址因该正确，jenkins 会提示输入 svn 的用户名和密码。



此时 job 将会在 slave 所在的机器运行，当然 build 所需要的环境要在 slave 上配置好哦，运行如下：

注 意： 对 slave 系统环境变量的修改，jenkins slave 不会立即生效，需要重启 jenkins slave service。 例如我在 slave 上装了 ant，设置到 path 中后仍然找不到，需要 restart jenkins slave service。

更多参考：

https://wiki.jenkins-ci.org/display/JENKINS/Distributed+builds
http://community.jboss.org/wiki/HudsonWindowsSlavesSetup

Jenkins 的 Linux 的 Slave 的配置

作为 slave 的 Linux 机器为 centos 系统。

1） Linux 的 Slave 机器设置
创建 jenkins 用户 sudo /usr/sbin/useradd -m jenkins -d /home/jenkins；
查看 jenkins 用户及组的信息 id jenkins ：
uid=506(jenkins) gid=506(jenkins) groups=506(jenkins) ；
使用 sudo /usr/bin/passwd jenkins 来设置用户 jenkins 的密码为 0；
切换到用户 jenkins 环境下 su - jenkins；
确保 java 安装正确：java --version；
确保 sshd 正确运行： /sbin/service --status-all | grep ssh；
安装 ant，在 root 下运行 yum install ant;

2) 在 Slave 的 linux 机器上创建 public/private key pair：

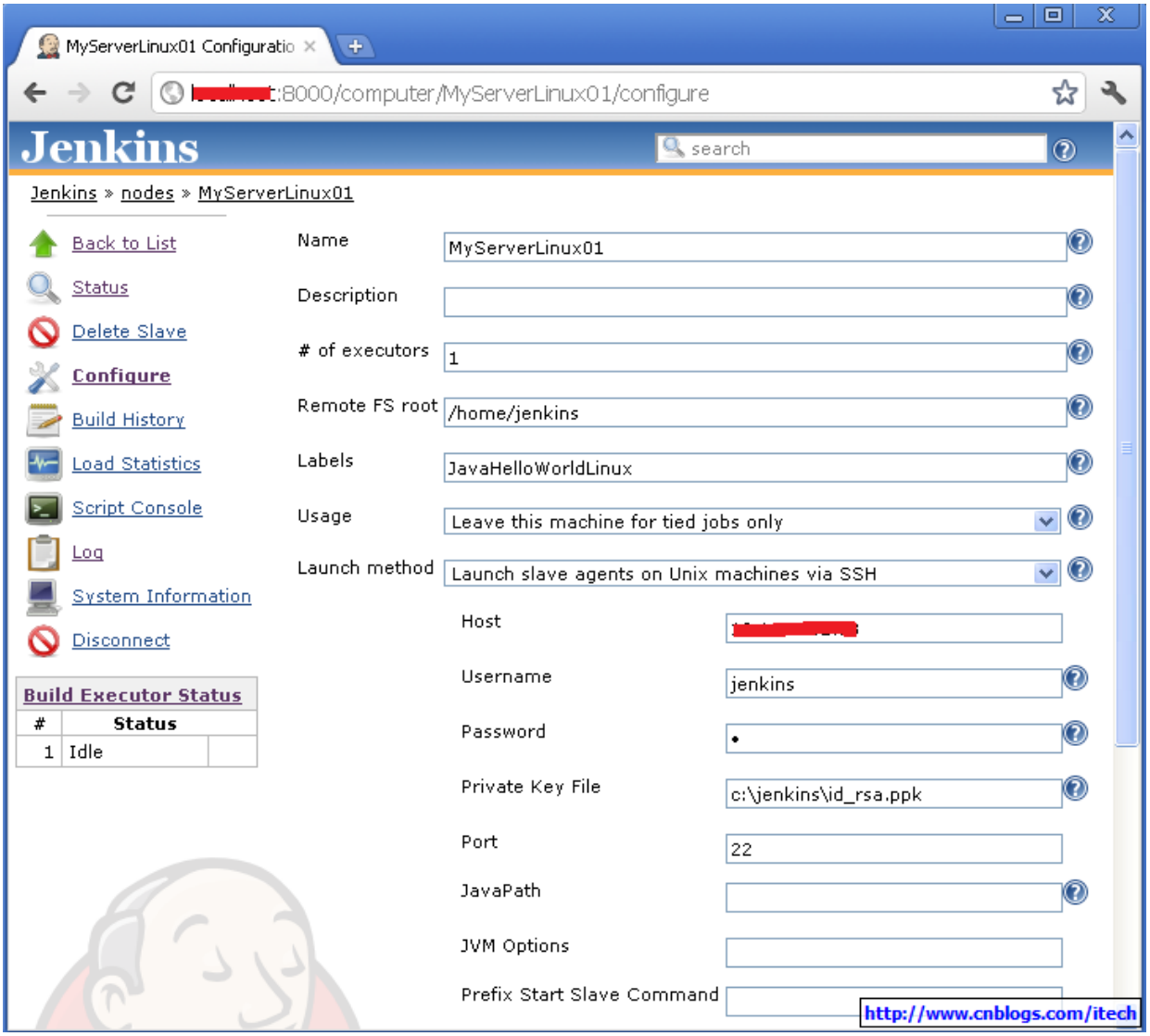
确保当前用户为 jenkins；

执行 ssh-keygen 来创建 public/private key pair，直接 enter，表示 key 将存储在 /home/jenkins/.ssh/id_rsa 下，再直接 enter，表示不设置密码，再次 enter 确认密码为空；

创建 authorized_keys：
cd .ssh
cat id_rsa.pub > authorized_keys
chmod 700 authorized_keys
;

将 id_rsa(相当于 privatekey）拷贝到 jenkins master 机器上，例如 c:\jenkins\id_rsa 下。

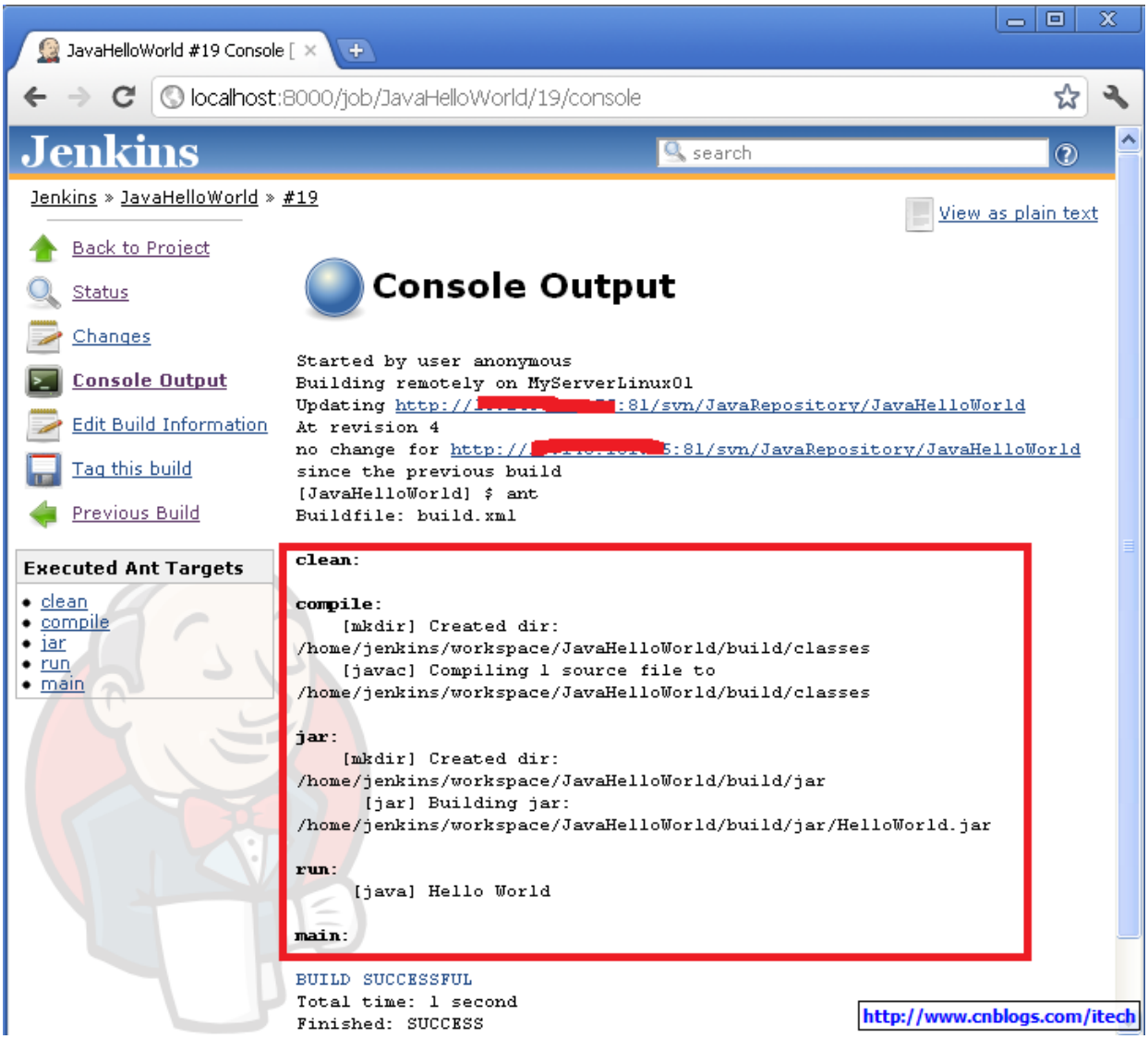
3）创建 Slave（note）,配置如下：



确保 jenkins 中 ssh slave plugin 正确安装，一般默认安装。

然后 lunch slave，使得 master 和 slave 通过 ssh 成功连接。其实 launch 的时候 jenkins 自动地从 `http://yourserver:port/jnlpJars/slave.jar` 拷贝 `slave.jar` 到 `slave`，然后运行通过命令 java -jar slave.jar 来运行 slave。

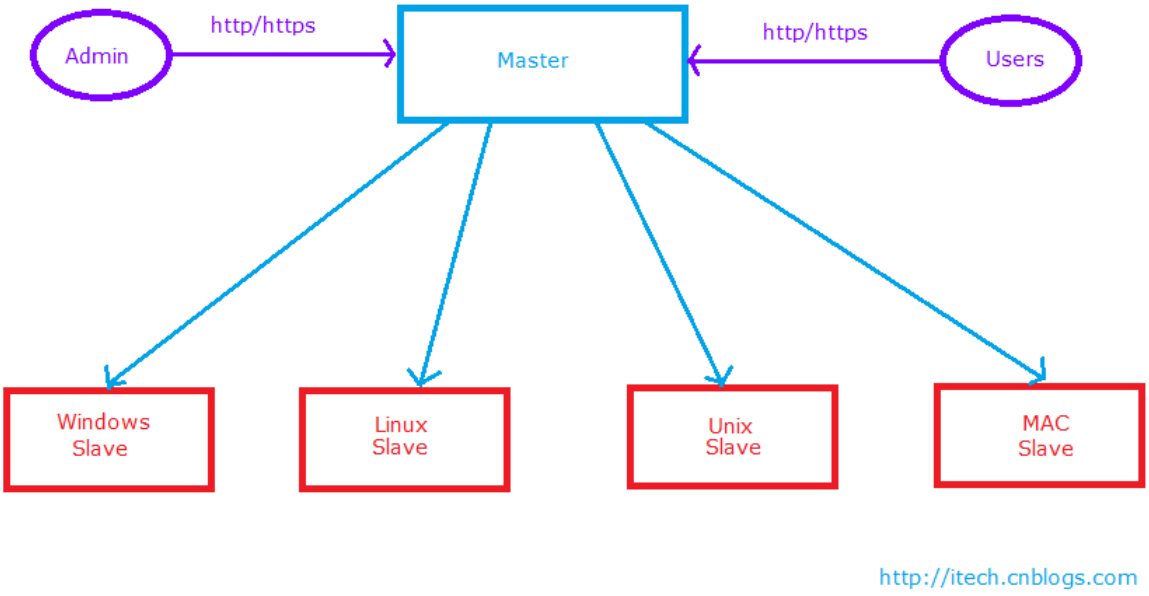
4）在新建的 Linux 的 Slave 上运行上节中的 JavaHelloWorld（Jenkins 构建 JavaHelloWorld），且需要修改 JavaHelloWorld job 的 Label 为 JavaHelloWorldLinux 来使用此 slave，运行如下：


JavaHelloWorld #19 Console [
localhost:8000/job/JavaHelloWorld/19/console
Jenkins
search
Jenkins » JavaHelloWorld » #19
View as plain text
Back to Project
Status
Changes
Console Output
Edit Build Information
Tag this build
Previous Build
Executed Ant Targets
clean
compile
jar
run
main
Console Output
Started by user anonymous
Building remotely on MyServerLinux01
Updating http://[redacted]:81/svn/JavaRepository/JavaHelloWorld
At revision 4
no change for http://[redacted]:81/svn/JavaRepository/JavaHelloWorld
since the previous build
[JavaHelloWorld] $ ant
Buildfile: build.xml
clean:
compile:
[mkdir] Created dir:
/home/jenkins/workspace/JavaHelloWorld/build/classes
[javac] Compiling 1 source file to
/home/jenkins/workspace/JavaHelloWorld/build/classes
jar:
[mkdir] Created dir:
/home/jenkins/workspace/JavaHelloWorld/build/jar
[jar] Building jar:
/home/jenkins/workspace/JavaHelloWorld/build/jar/HelloWorld.jar
run:
[java] Hello World
main:
BUILD SUCCESSFUL
Total time: 1 second
Finished: SUCCESS
http://www.cnblogs.com/itech

Jenkins Master/Slave 架构

一 Jenkins Master/Slave 架构

Master/Slave 相当于 Server 和 agent 的概念。Master 提供 web 接口让用户来管理 job 和 slave，job 可以运行在 master 本机或者被分配到 slave 上运行。一个 master 可以关联多个 slave 用来为不同的 job 或相同的 job 的不同配置来服务。



当 job 被分配到 slave 上运行的时候，此时 master 和 slave 其实是建立的双向字节流的连接，其中连接方法主要有如下几种：

1）master 通过 ssh 来启动 slave

Jenkins 内置有 ssh 客户端实现，可以用来与远程的 sshd 通信，从而启动 slave agent。这是对*unix 系统的 slave 最方便的方法，因为*unix 系统一般默认安装有 sshd。在创建 ssh 连接的 slave 的时候，你需要提 供 slave 的 host 名字，用户名和 ssh 证书。创建 public/private keys，然后将 public key 拷贝到 slave 的~/.ssh/authorized_keys 中，将 private key 保存到 master 上某 ppk 文件中。jenkins 将会自动地完成其他的配置工作，例如 copy slave agent 的 binary，启动和停止 slave。但是你的 job 运行所依赖其他的项目需要你自己设置。

2）master 通过 WMI+DCOM 来启动 windows slave

对 于 Windows 的 Slave，Jenkins 可以使用 Windows2000 及以后内置的远程管理功能（WMI+DCOM），你只需要提供对 slave 有管理员访问权限的用户名和密码，jenkins 将远程地创建 windows service 然后远程地启动和停止他们。

对于 windows 的系统，这是最方便的方法，但是此方法不允许运行有显示交互的 GUI 程序。

注意：不想其他类型的链接方式，此种方式 slave（note）的名字非常重要，将被用来当做 slave 的地址访问 slave。

3）实现自己的脚本来启动 slave

如果上面成套的方法不够灵活，你可以实现自己的脚本来启动 slave。你需要将启动脚本放到 master，然后告诉 jenkins master 在需要的时候调用此脚本来启动 slave。

典型地，你的脚本使用远程程序执行机制，例如 SSH，RSH，或类似的方法（在 windows，可以通过 cygwin 或 psexec 来完成），

在脚本的最后需要执行类似 java -jar slave.jar 来启动 slave。slave.jar 可以从 http://yourjenkinsserver:port/jnlpjars/slave.jar 下载，也可以在脚本的开始先下载此 slave.jar 从而保证 slave.jar 正确的版本。 但是如果使用 ssh slave plugin 的话，此 plugin 将自动地更新 slave.jar。

4）通过 Java web start 来启动 slave

jave web start（jnlp）是另一种启动 slave 的方法。用这种方法你需要登录到 slave，打开浏览器，打开 slave 的配置页面来连接。还可以安装为 windows service 来使得 slave 在后台运行。

如果你需要运行的程序需要 UI 的交互，使用下面的方法：在 slave 系统上创建 jenkins 用户，设置自动登录，在系统的 startup items 增加 slave JNLP 文件的快捷方式，使得 slave 在系统登录的时候自动启动。

5）直接启动 slave

此方式类似于 java web start，可以方便地在*unix 系统上将 slave 运行为 daemon。需要配置 slave 为 JNLP 类型连接，然后在 slave 机器上执行

```
java -jar slave.jar -jnlpUrl http://yourserver:port/computer/slave-
name/slave-agent.jnlp
```

二 Slave 配置的好的建议

* 每个 slave 都有用户 jenkins，所有的机器使用相同的 UID 和 GID，使得 slave 的管理更加简单；
* 每个机器上 jenkins 用户的 home 目录都相同/home/jenkins，拥有相同的目录结构使得维护简单；
* 所有的 slave 运行 sshd，windows 运行 cygwin sshd；
* 所有的 slave 安装 ntp client，用来与相同的 ntp server 同步；
* 使用脚本 sh 来自动地配置 slave 的环境，例如创建 jenkins 用户，安装 sshd，安装 java，ant，maven 等；
* 使用脚本来启动 slave，保证 slave 总是运行在相同的参数下：

```
#!/bin/bash JAVA_HOME=/opt/SUN/jdk1.6.0_04 PATH=$PATH:$JAVA_HOME/bin export
PATH java -jar /var/jenkins/bin/slave.jar
```

Jenkins 最佳实践

Jenkins 最佳实践，其实大部分对于其他的 CI 工具同样的适用：

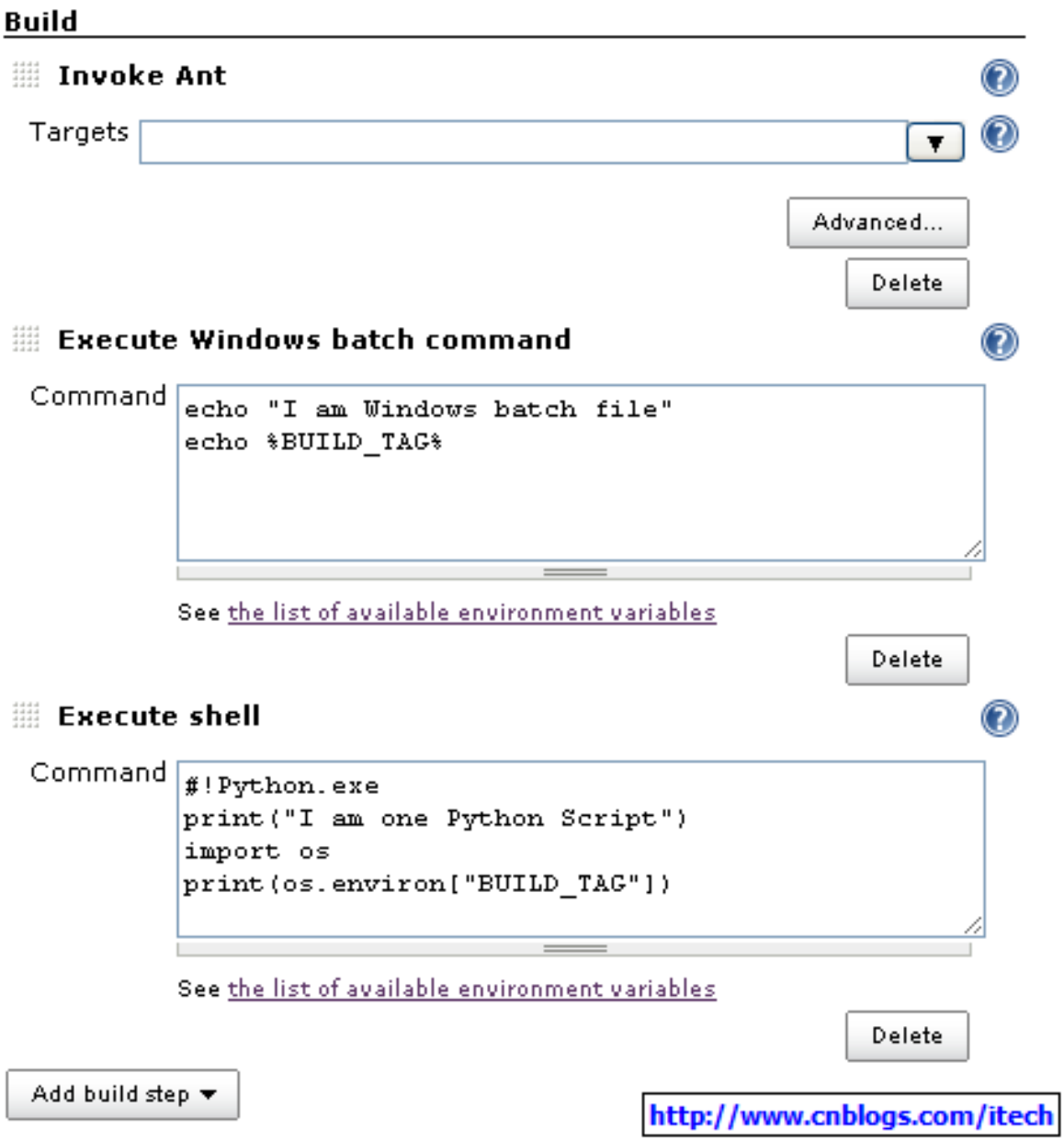
* Jenkins 的安全。对 Jenkins 的用户使用授权和访问控制。默认地 Jenkins 不执行任何的安全检查，这意味着任何人都可以访问 Jenkins 来配置 Jenkins，修改 job，和执行 build。这对于在企业内部使用也许可以接受，但是存在很高的安全风险，例如其他人错误滴删除了 job，错误 地配置你的 job 在每分钟运行，启动太多的 builds 等。所以一般使用 plugin 来对 Jenkins 增加授权和访问控制。
* 有规律地对 Jenkins 的 home 目录的备份。
* 使用 file fingerprinting 来管理依赖关系。当在 Jenkins 上你的 job 依赖其他的 job 时，可以使用 file fingerprinting 来帮助定位依赖的版本信息。
* 最可靠的 build 是 clean builds，clean builds 意思是与 build 相关的所有的 3rd party，build 脚本，发布说明等都需要在 Source code control。
* 与 issue tracking 系统紧密的集成，例如 JIRA 或 bugzilla，从来减少对 change log 的修改。
* 与 repository 浏览工具紧密的集成，例如 FishEye 如果你使用 Subversion 作为 source code 管理工具。
* 总是配置 job 产生趋势报告和自动化测试，当你运行一个 Java build。趋势报告帮助项目经理和开发人员快速地了解当前项目的进度和状态。
* 确保 Jenkins 的 home 目录拥有足够的空间。
* 在删除不使用的 job 前请先存档。
* 为不同的 branch 建立不同的 job，build 来尽早地发现错误。
* 为并行的项目 builds 分配不同的端口，来避免多个 jobs 同时启动时所遇到的冲突。
* 为不同的项目的开发人员建立 email aliais，使得项目所有相关的人员都第一时间了解项目的状态。
* 增加额外的步骤来尽早地发现失败。例如 log 检查，微测试等。
* 对于经常的维护性的工作可以使用 job 来自动地完成，例如对磁盘的清除工作。
* 在 build 成功后对远代码 Tag，label 或 baseline。
* 配置 Jenkins bootstrapper 来在 build 前更新工作目录。

翻译自：https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Best+Practices

Jenkins 中执行 batch 和 Python

Jenkins 的 job->build 支持 Ant，maven，windows batch 和 Shell， 但是我们知道 python，perl，ruby 等脚本其实也是 shell 脚本，所以这里的 Shell 可以扩展为 python，perl，ruby 等。

例如： 下面执行 windows batch 和 python



执行后的输入如下：

```
Total time: 1 second
[JavaHelloWorld] $ cmd /c call
C:\WINDOWS\TEMP\hudson3362264396373364573.bat

c:\jenkins\workspace\JavaHelloWorld>echo "I am Windows batch file"
"I am Windows batch file"

c:\jenkins\workspace\JavaHelloWorld>echo jenkins-JavaHelloWorld-21
jenkins-JavaHelloWorld-21

c:\jenkins\workspace\JavaHelloWorld>exit 0
[JavaHelloWorld] $ Python.exe
C:\WINDOWS\TEMP\hudson6871295144360615461.sh
I am one Python Script
jenkins-JavaHelloWorld-21
Finished: SUCCESS
```

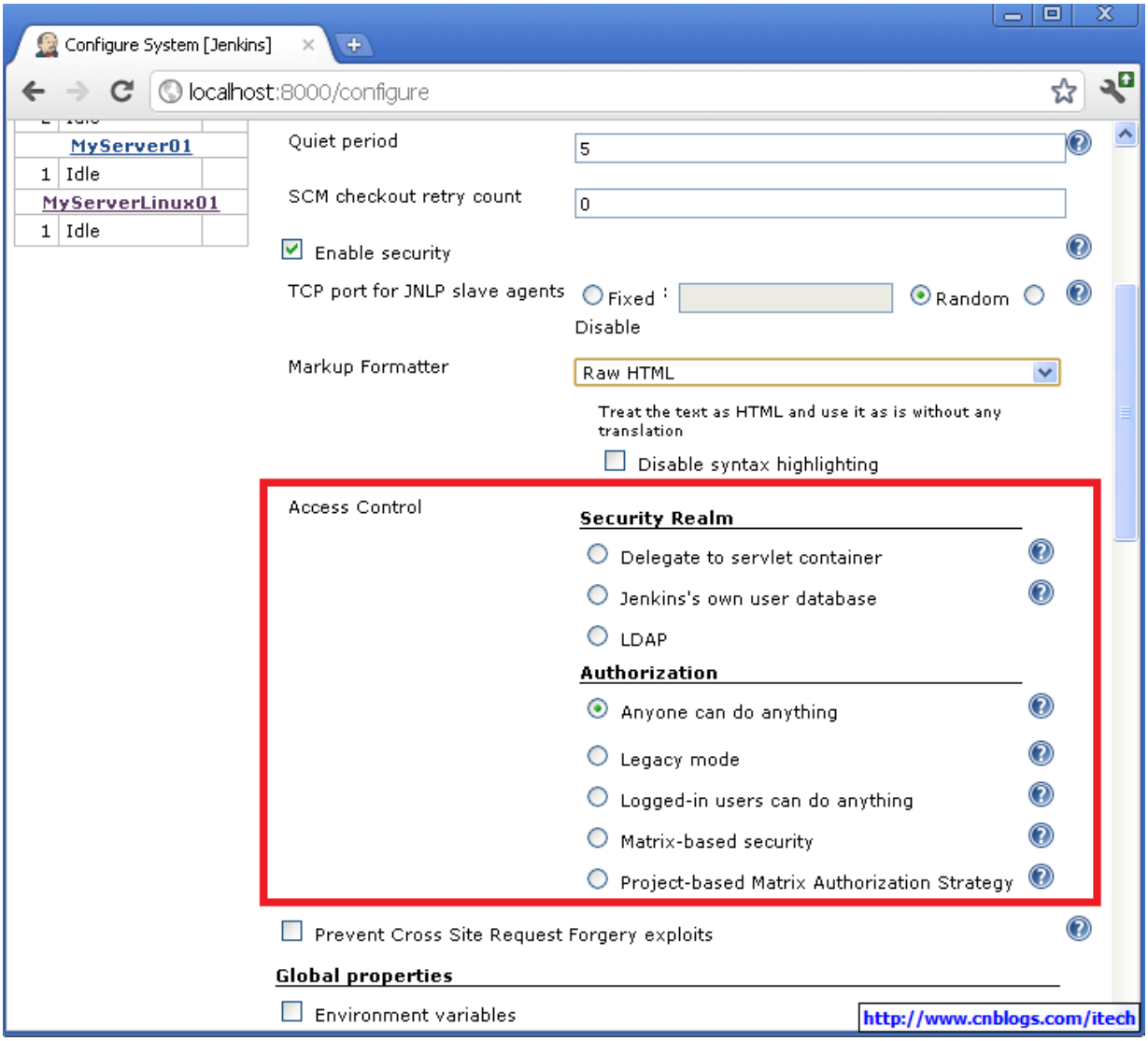
http://www.cnblogs.com/itech

可以看到 windows batch 和 shell 脚本被保存到 slave 上的临时目录下，然后再执行。

Jenkins 的授权和访问控制

一 Jenkins 的授权和访问控制
默认地 Jenkins 不包含任何的安全检查，任何人可以修改 Jenkins 设置，job 和启动 build 等。显然地在大规模的公司需要多个部门一起协调工作的时候，没有任何安全检查会带来很多的问题。 我们可以通过以下 2 方面来增强 Jenkins 的安全：
1） Security Realm，用来决定用户名和密码，且指定用户属于哪个组；
2） Authorization Strategy，用来决定用户对那些资源有访问权限；

在 Manage Jenkins -> Configure System -> Enable Security 下可以看到可以使用多种方式来增强 Jenkins 的授权和访问控制。



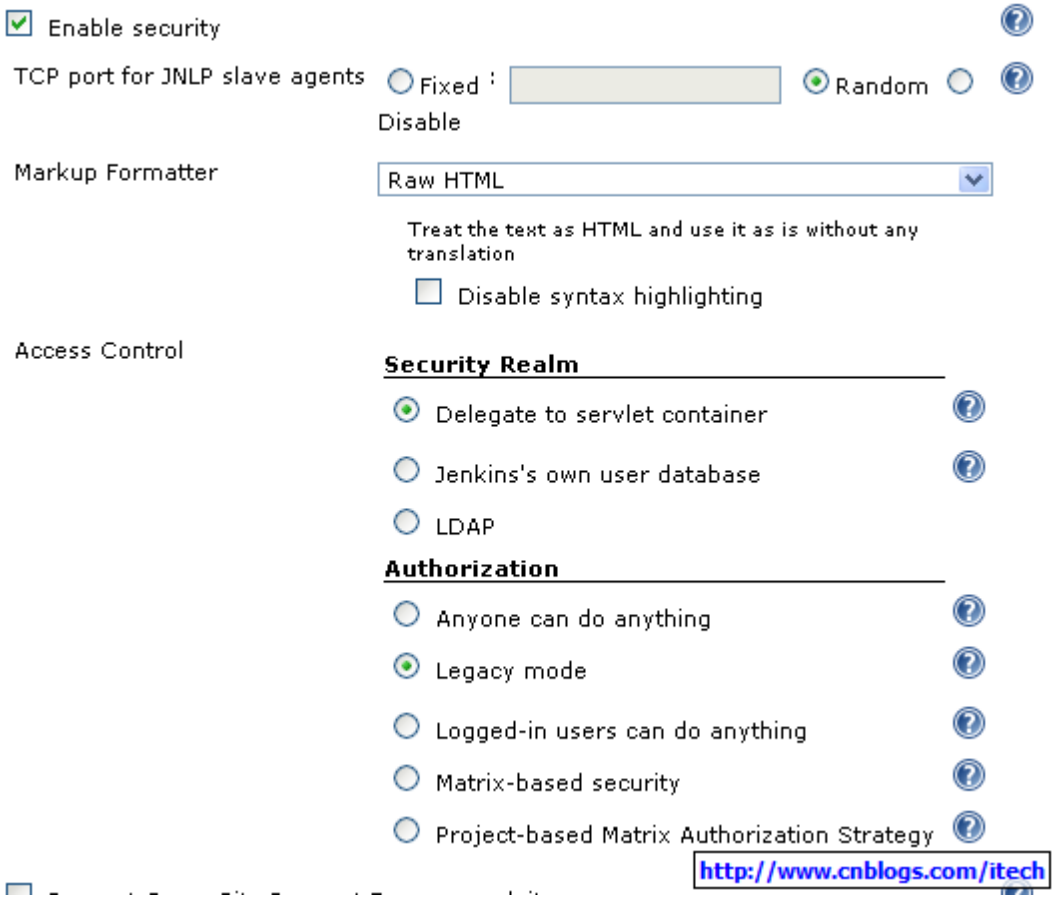
二 创建管理员账号+匿名只读

简单地设置一个管理员账号，用来管理 jenkins 设置，修改 job 和执行 build 等。其他的匿名访问的用户将只有只读的权限，不能修改 Jenkins 的设置，不能修改 job，且不能运行 build，但是可以访问 build 的结构，查看 build 的 log 等。

1）需要对 Jenkins 增加如下的启动参数来创建管理员账号：
java -jar jenkins.war --argumentsRealm.passwd.user=password --argumentsRealm.roles.user=admin

例如设置管理员用户为 jenkins 且密码为 swordfish，如下：
java -jar jenkins.war --argumentsRealm.passwd.jenkins=swordfish --argumentsRealm.roles.jenkins=admin

2） 在启动后需要在 Manage Jenkins -> Configure System 来选择 enable security，然后选择对 Security Realm 选择 Delegate to servlet container，对 Authorization 选择 Legacy Mode。



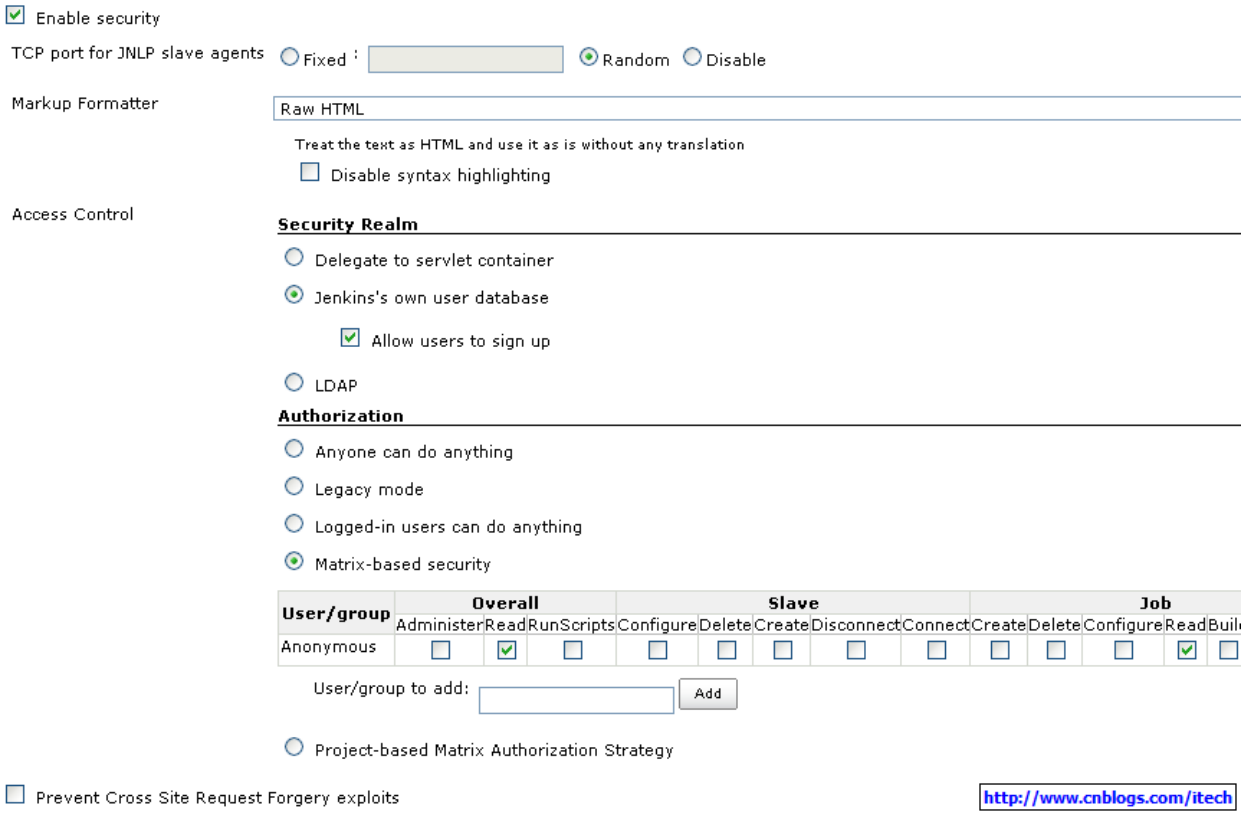
3）然后可以在右上角点击 login 或者 http://yourhost/jenkins/loginEntry 来登录，登录后此时你用户管理员权限，可以执行任何的操作。执行完操作后可以选择 logout。

4） 对 developers 增加 rebuild 的权限。使用管理员登录对需要 developer rebuild 的 job，选择 Trigger builds remotely，且设置 Authentication Token，例如设置为 devbuild，然后 developers 可以访问 http://jenkinsHost/job/project /build?token=token 来启动 build。
其中 Project 为你需要启动 build 的 job。
其中 token 为你设置的 Authentication Token。
如果你有 webserver，你可以创建如下的 webpage 来让 developers 来启动 build：

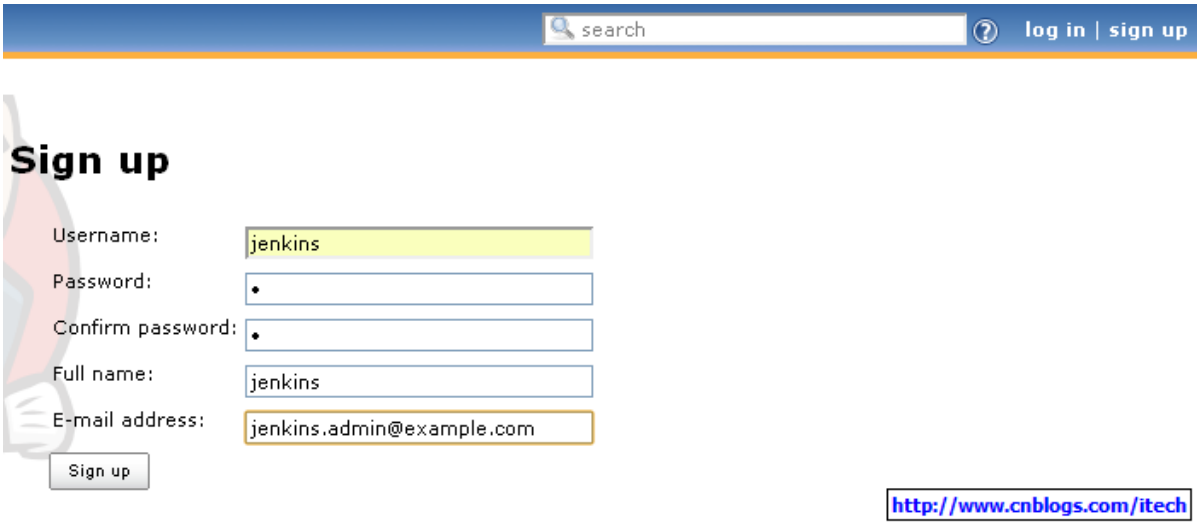
```
<h1>Jenkins Force Build Page</h1>
<ul>
  <li>
  <a href="http://jenkins:8080/job/FOO/build?token=build">Force build of Project FOO on Jenkins</a>
  </li>
</ul>
```

三 使用 Jenkins 的数据库管理用户且设置用户的访问权限

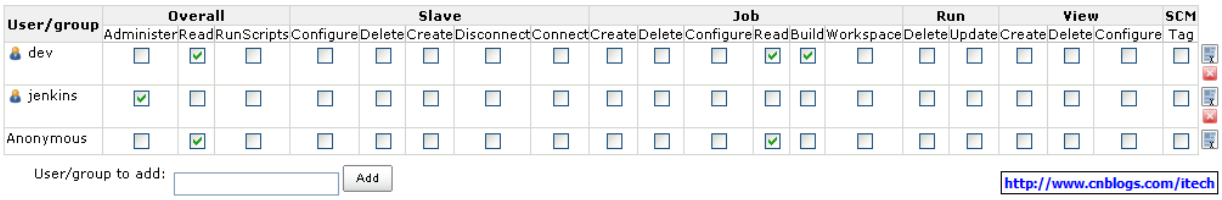
1） 在 Manage Jenkins -> Configure System -> Enable Security 下为 Security Realm 选择 Jenkins's own user database，且确保 Allow users to sign up 选中，为 Authentication 选择 Matrix-based security。如下：



2）接着在主页上 signup 创建第一个管理员账号 jenkins 如下：



3）除了第一个账号以后 signup 的账号将为只读账号需要管理员分配权限。例如你可以 signup 来创建 dev 账号，然后分配权限使得 dev 可以启动 job 的 build。如下：



注意：

在%JENKINS_HOME%\config.xml 中可以查看和修改所有用户的权限设置，但是修改后需要重新启动 Jenkins。

参考：

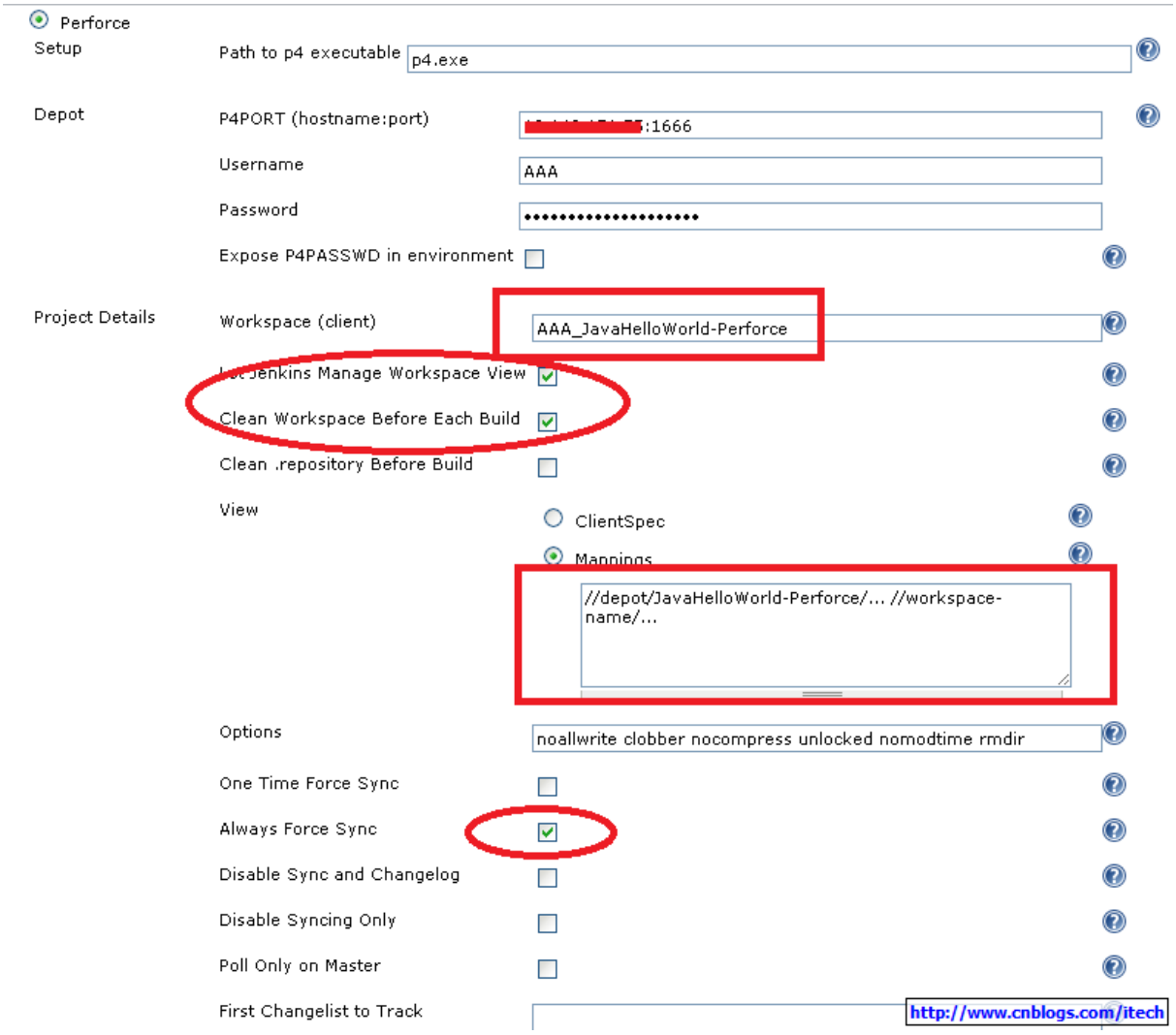
https://wiki.jenkins-ci.org/display/JENKINS/Quick+and+Simple+Security
https://wiki.jenkins-ci.org/display/JENKINS/Standard+Security+Setup

Jenkins 插件之 Perforce 访问

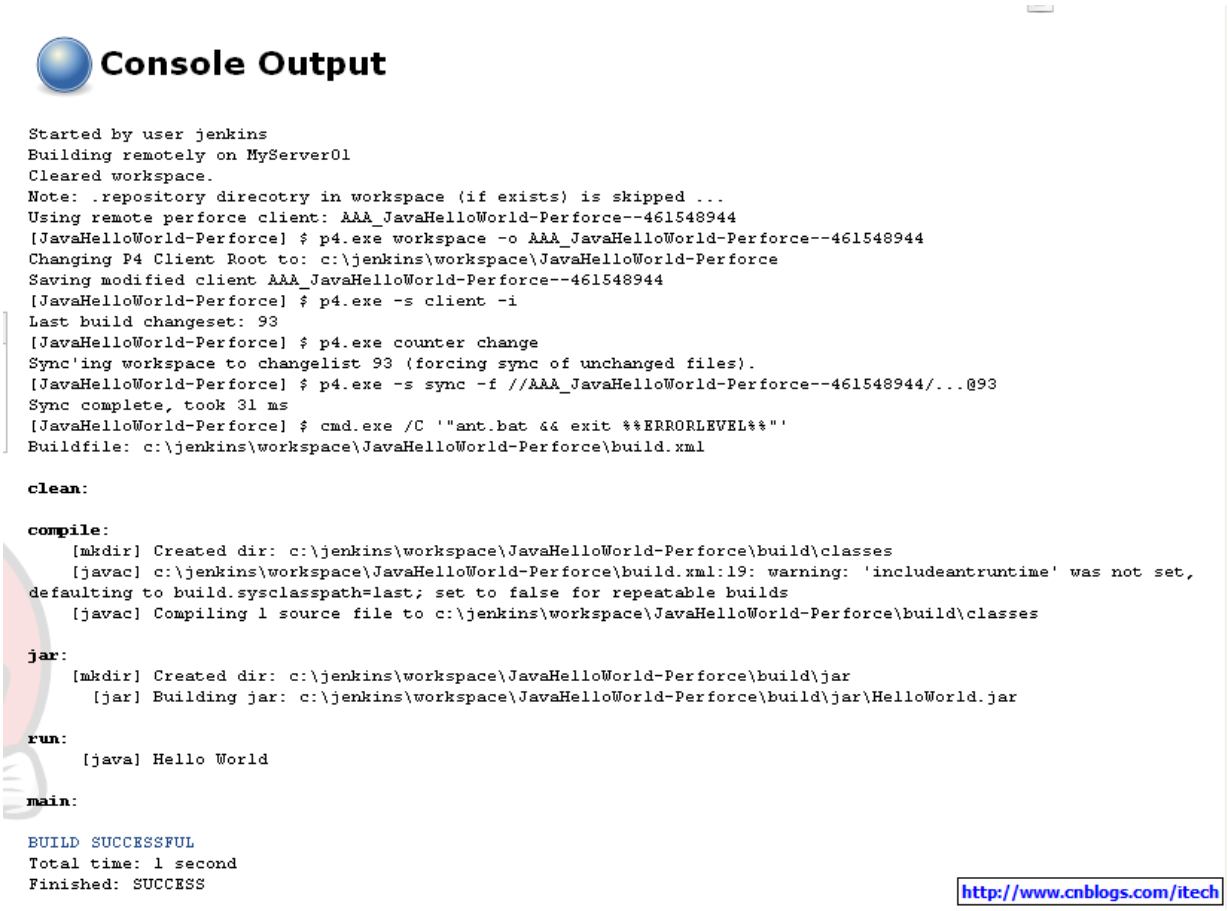
Perforce Plugin，在 Jenkins 的管理页面的插件管理下面安装 Perforce 插件，然后重启 Jenkins。

一 使用 perforce 插件来 build

对 job 的设置如下图：



job 执行后的 log 如下：

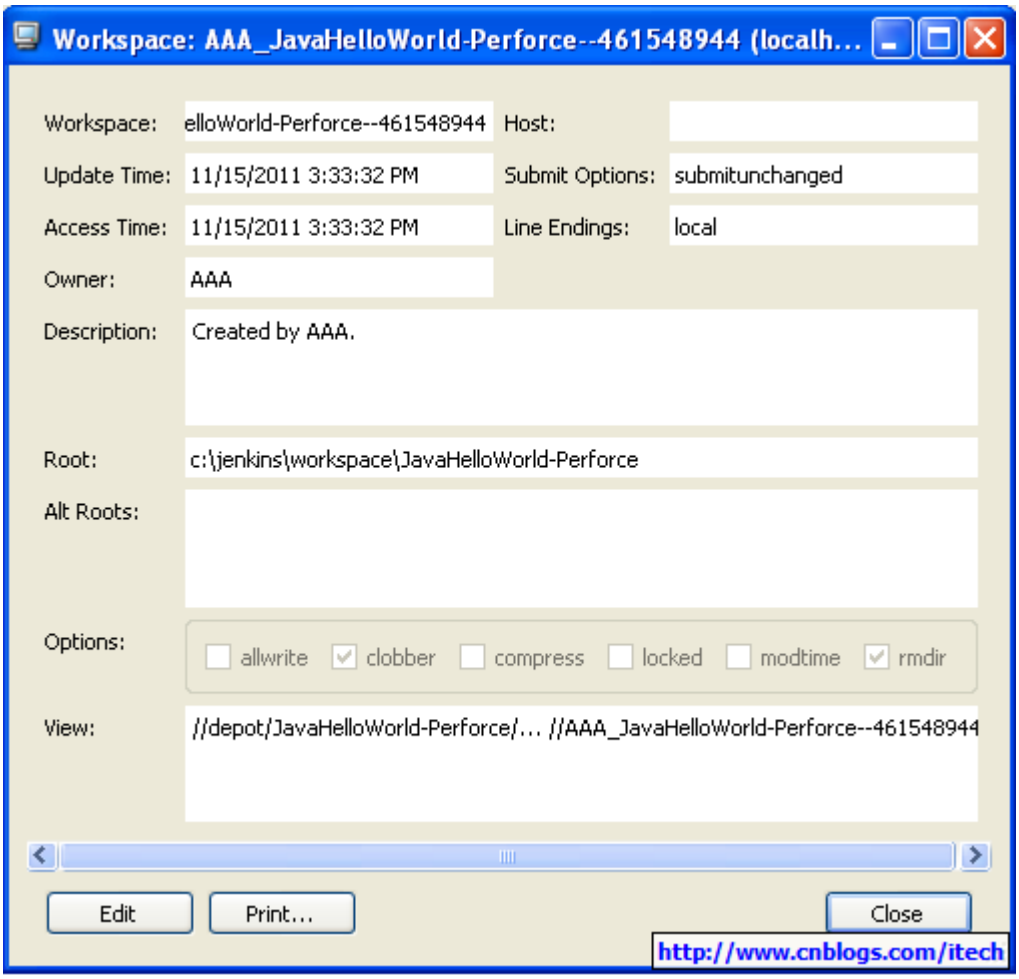


Console Output

```
Started by user jenkins
Building remotely on MyServer01
Cleared workspace.
Note: .repository direcotry in workspace (if exists) is skipped ...
Using remote perforce client: AAA_JavaHelloWorld-Perforce--461548944
[JavaHelloWorld-Perforce] $ p4.exe workspace -o AAA_JavaHelloWorld-Perforce--461548944
Changing P4 Client Root to: c:\jenkins\workspace\JavaHelloWorld-Perforce
Saving modified client AAA_JavaHelloWorld-Perforce--461548944
[JavaHelloWorld-Perforce] $ p4.exe -s client -i
Last build changeset: 93
[JavaHelloWorld-Perforce] $ p4.exe counter change
Sync'ing workspace to changelist 93 (forcing sync of unchanged files).
[JavaHelloWorld-Perforce] $ p4.exe -s sync -f //AAA_JavaHelloWorld-Perforce--461548944/...@93
Sync complete, took 31 ms
[JavaHelloWorld-Perforce] $ cmd.exe /C '"ant.bat && exit %%ERRORLEVEL%%"'
Buildfile: c:\jenkins\workspace\JavaHelloWorld-Perforce\build.xml

clean:

compile:
    [mkdir] Created dir: c:\jenkins\workspace\JavaHelloWorld-Perforce\build\classes
    [javac] c:\jenkins\workspace\JavaHelloWorld-Perforce\build.xml:19: warning: 'includeantruntime' was not set,
defaulting to build.sysclasspath=last; set to false for repeatable builds
    [javac] Compiling 1 source file to c:\jenkins\workspace\JavaHelloWorld-Perforce\build\classes

jar:
    [mkdir] Created dir: c:\jenkins\workspace\JavaHelloWorld-Perforce\build\jar
      [jar] Building jar: c:\jenkins\workspace\JavaHelloWorld-Perforce\build\jar\HelloWorld.jar

run:
     [java] Hello World

main:

BUILD SUCCESSFUL
Total time: 1 second
Finished: SUCCESS
```

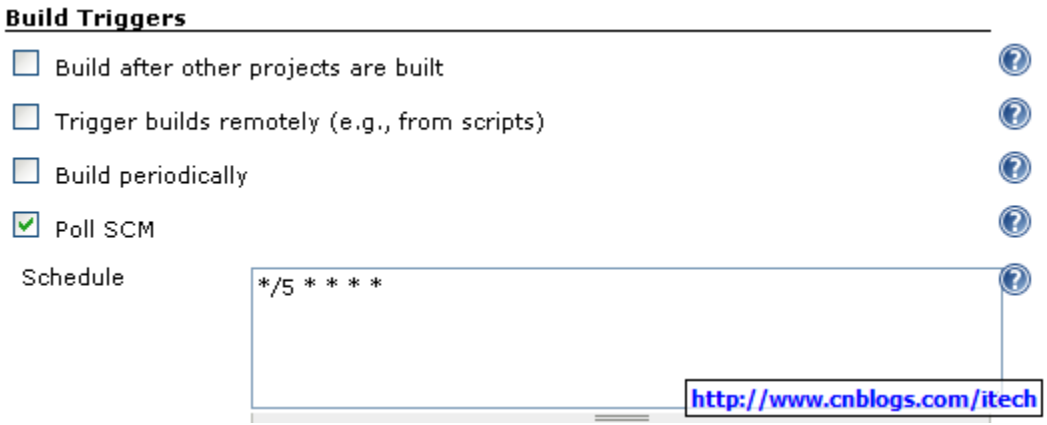
http://www.cnblogs.com/itech

可 以看到 Jenkins 在执行的过程中创建了新的 clientspec，新的 clientspec 是拷贝自上面参数 workspace 设置的 clientspec，且修改了新的 clientroot 目录，其中的 view 是来自上面参数 view->mapping 中的设置。
如下：

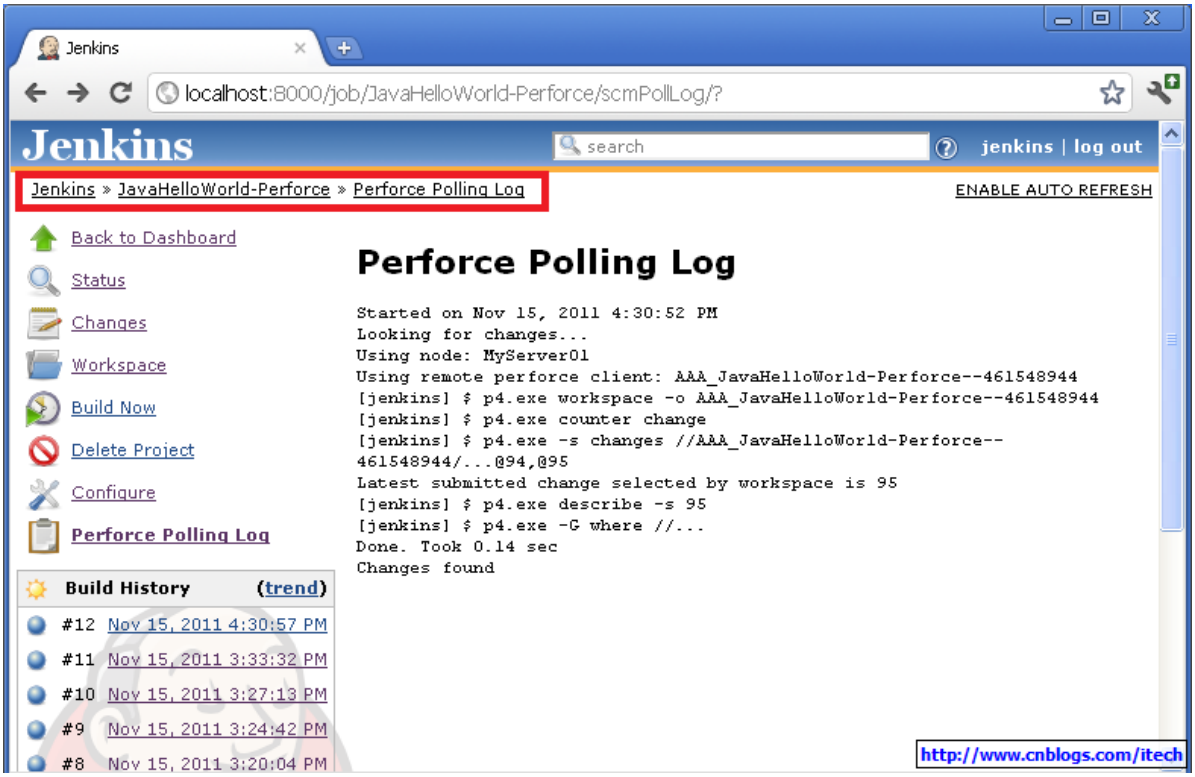


二 使用 perforce 插件的 poll 功能来触发 build

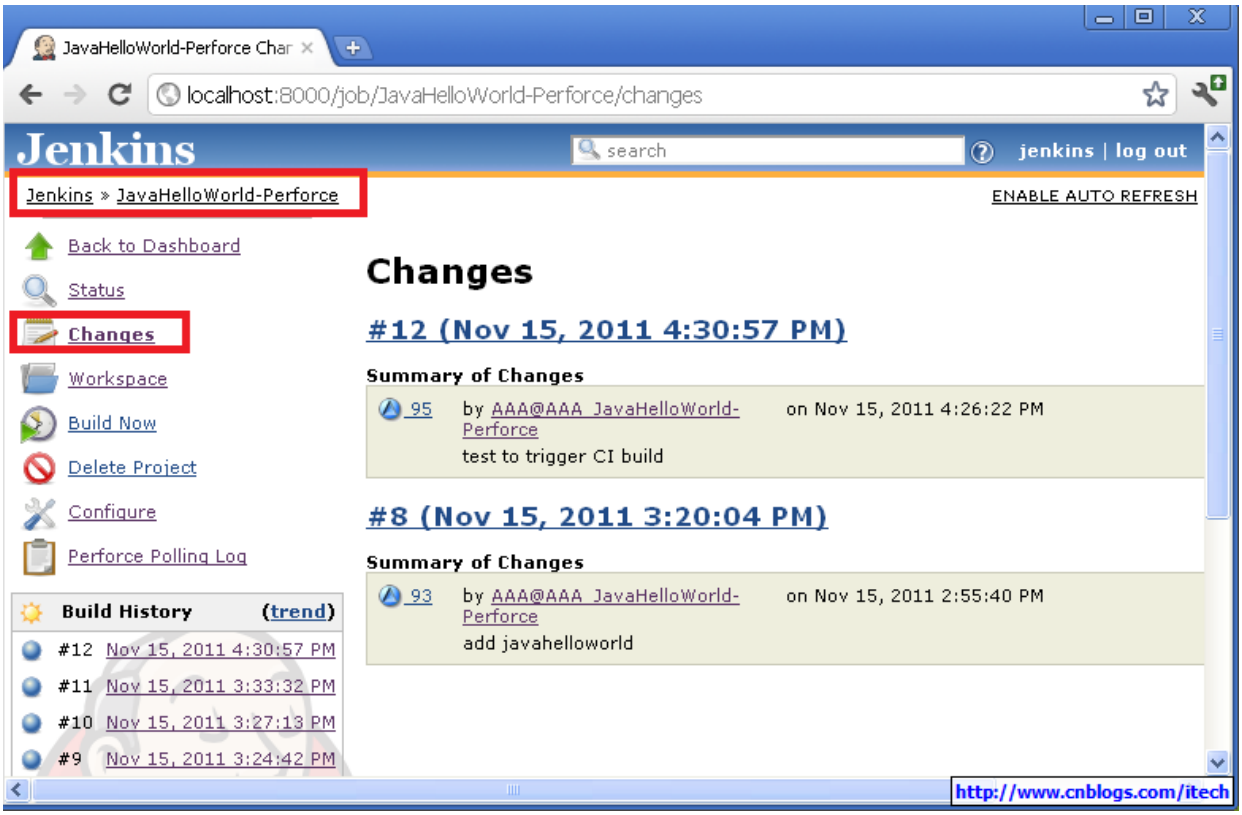
配置如下：



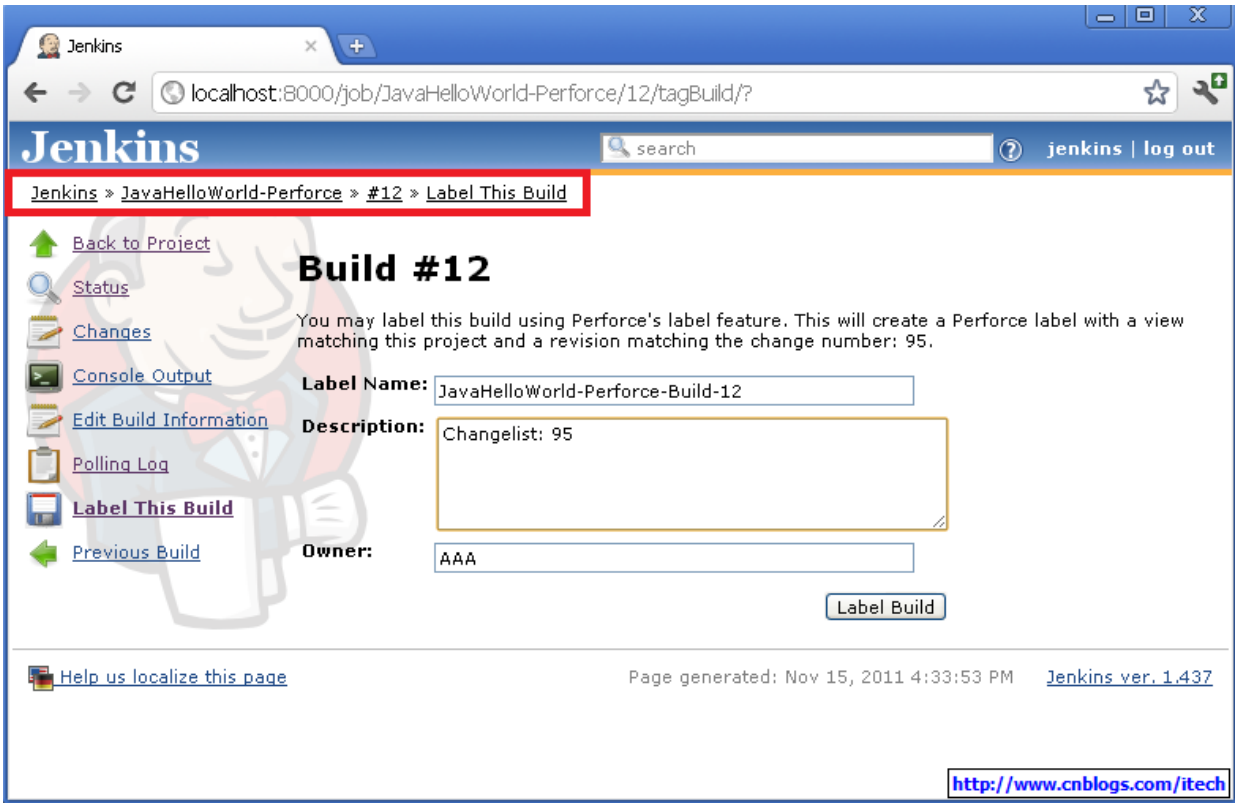
查看如下：



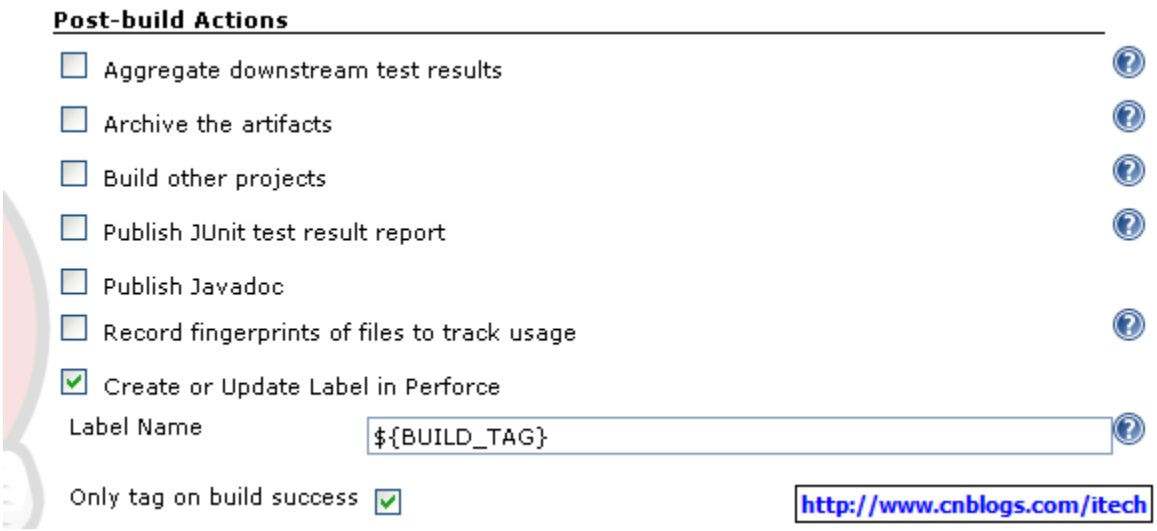
三 使用 perforce 插件在 Jenkins 中查看最新的修改



四 使用 perforce 的 label 功能来对成功的 build 进行 label



五 使用 perforce 插件的自动 label 功能



更多的插件：

https://wiki.jenkins-ci.org/display/JENKINS/Plugins

Jenkins 插件之 trigger

一 Jenkins 内置的 trigger 插件

1） build after other projects are built
可以设置多个依赖的 jobs，当任意一个依赖的 jobs 成功后启动此 build。 多个依赖的 jobs 间使用,隔开。

2） Trigger builds remotely (e.g., from scripts)
在 Authentication Token 中指定 TOKEN_NAME，然后可以通过连接 JENKINS_URL/job/JOBNAME/build?token=TOKEN_NAME 来启动 build。

3） build periodically
在 schedule 中设置，语法类似于 cron 中语法。

4） Poll SCM
在 schedule 中设置时间间隔来抓取 SCM server，如果有新的修改，则启动 build。 所以这里的作用相当于 continous build。

二 其他的 trigger 插件

需要手动安装插件。

Maven Dependency Update trigger： 当有检测到有 Maven dependency 跟新的时候启动 build，类似于 continuous build。

BuildResultTrigger Plugin： 根据其他的 job 的成功或失败来启动此 build。

Files Found Trigger：检测指定的目录，如果发现指定模式的文件则启动 build。

更多：

https://wiki.jenkins-ci.org/display/JENKINS/Plugins

Jenkins 插件之构建与 MSBuild

一 Jenkins 内置的 buildtools

Jenkins 已经内置了 Ant|Maven|Windows batch|Shell(Perl,Python)的支持。

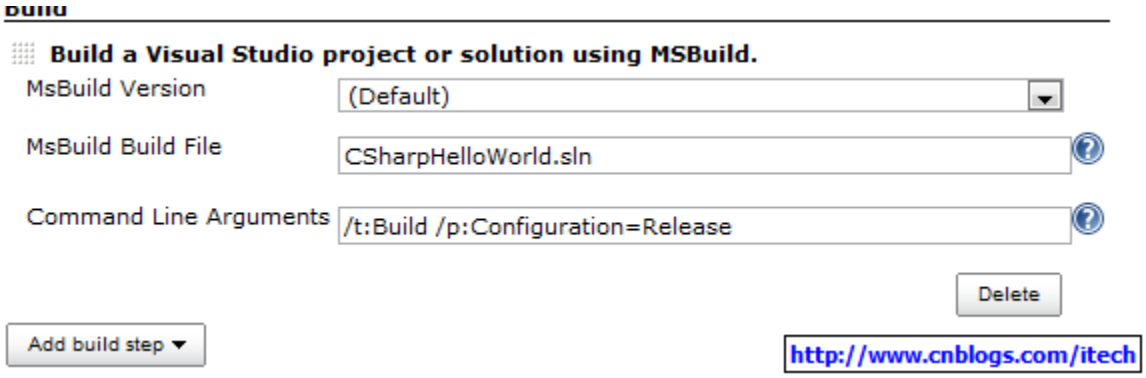
二 其他的 buildtools
cmakebuilder Plugin： 支持 cmake 的构建；
Copy Artifact Plugin： 拷贝依赖的组件；
Job Exporter Plugin： 将当前的运行参数导出到属性文件， 可以供以后的步骤调用；
MSBuild Plugin： 使用 MSBuild 来构建.NET 工程；
NAnt Plugin： 用来支持 NAnt;
Python Plugin : 用来支持 python；
qmakebuilder Plugin： 用来支持 qmake；
Rake plugin： 用来支持 rake 构建；
SCons Plugin： 用来支持 Scons 构建；
Xcode Plugin： 用来支持 MAC，iphone 等的构建；

三 使用 MSBuild 来构建 CsharpHelloWorld

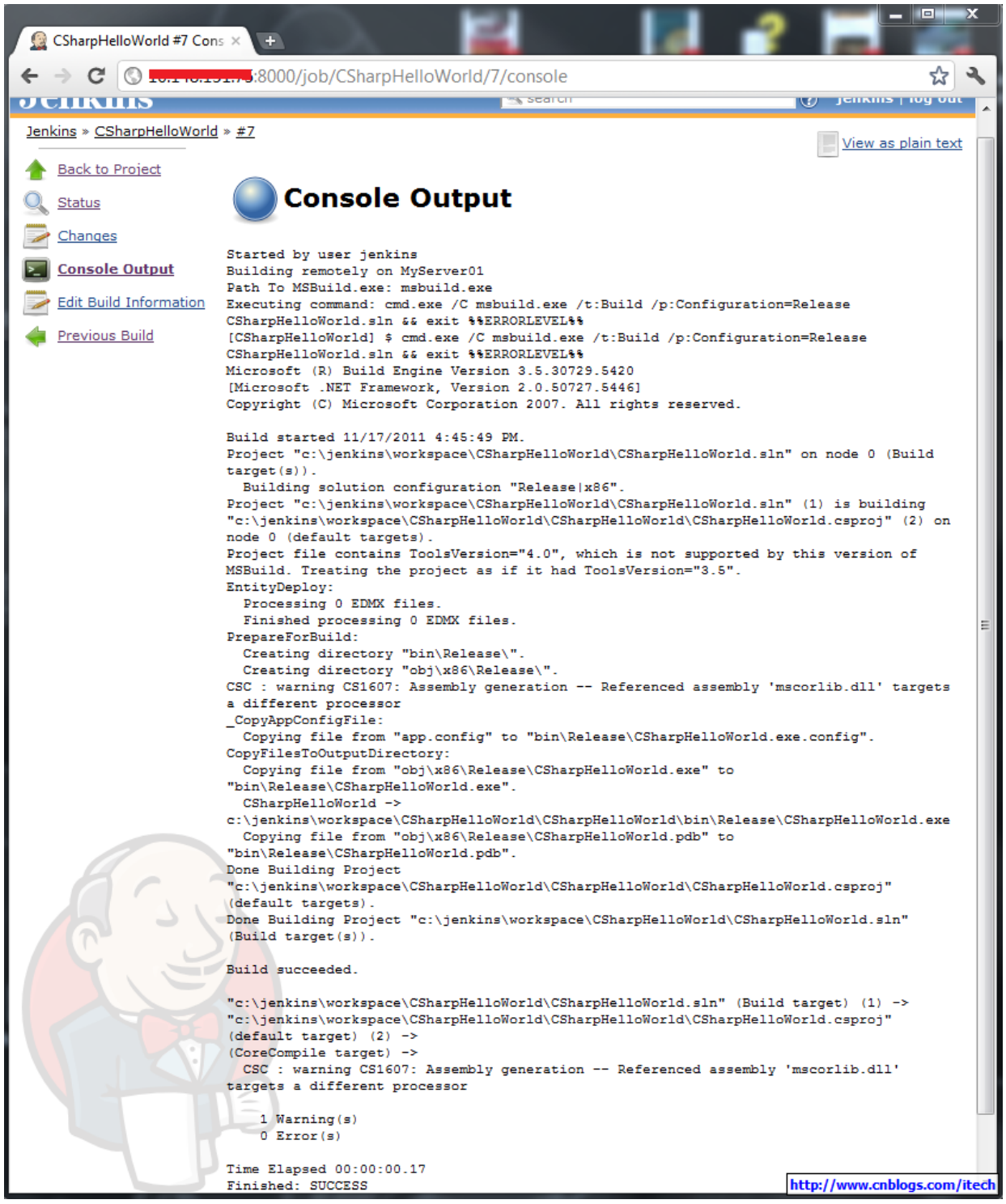
1） CSharp 的 console project 代码如下：

```
using System;

namespace CSharpHelloWorld
{
  class Program
  {
    static void Main(string[] args)
    {
      System.Console.WriteLine("Hello World!");
    }
  }
}
```

2） 创建 Jenkins job CSharpHelloWorld，设置如下： 需要确保 slave 机器上 msbuild 的路径在系统 path 环境变量中，例如 C:\Windows\Microsoft.NET\Framework\v4.0.30319


Build
Build a Visual Studio project or solution using MSBuild.
MsBuild Version
(Default)
MsBuild Build File
CSharpHelloWorld.sln
Command Line Arguments
/t:Build /p:Configuration=Release
Delete
Add build step
http://www.cnblogs.com/itech

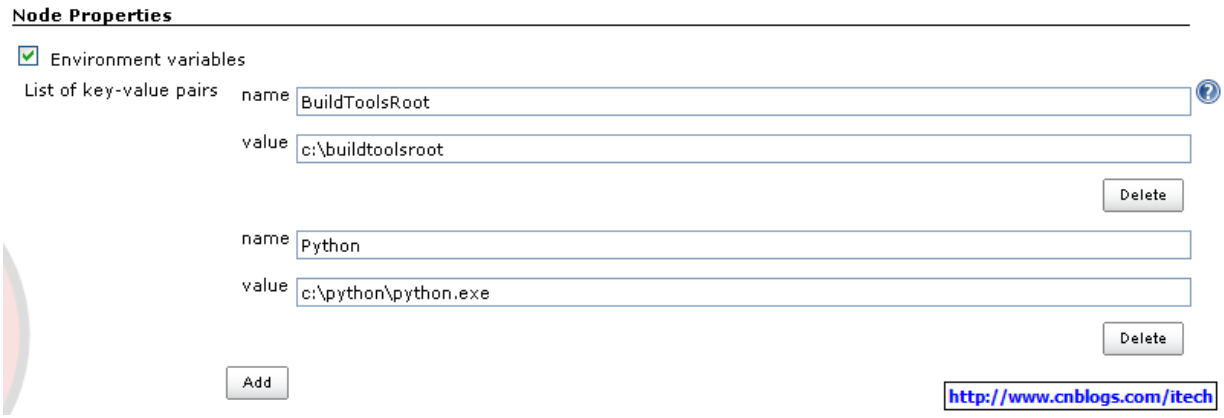
3） build 结果如下：


CSharpHelloWorld #7 Cons
[redacted]:8000/job/CSharpHelloWorld/7/console
Jenkins » CSharpHelloWorld » #7
View as plain text
Back to Project
Status
Changes
Console Output
Edit Build Information
Previous Build
Console Output
Started by user jenkins
Building remotely on MyServer01
Path To MSBuild.exe: msbuild.exe
Executing command: cmd.exe /C msbuild.exe /t:Build /p:Configuration=Release CSharpHelloWorld.sln && exit %%ERRORLEVEL%%
[CSharpHelloWorld] $ cmd.exe /C msbuild.exe /t:Build /p:Configuration=Release CSharpHelloWorld.sln && exit %%ERRORLEVEL%%
Microsoft (R) Build Engine Version 3.5.30729.5420
[Microsoft .NET Framework, Version 2.0.50727.5446]
Copyright (C) Microsoft Corporation 2007. All rights reserved.
Build started 11/17/2011 4:45:49 PM.
Project "c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld.sln" on node 0 (Build target(s)).
Building solution configuration "Release|x86".
Project "c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld.sln" (1) is building "c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld\CSharpHelloWorld.csproj" (2) on node 0 (default targets).
Project file contains ToolsVersion="4.0", which is not supported by this version of MSBuild. Treating the project as if it had ToolsVersion="3.5".
EntityDeploy:
Processing 0 EDMX files.
Finished processing 0 EDMX files.
PrepareForBuild:
Creating directory "bin\Release\".
Creating directory "obj\x86\Release\".
CSC : warning CS1607: Assembly generation -- Referenced assembly 'mscorlib.dll' targets a different processor
_CopyAppConfigFile:
Copying file from "app.config" to "bin\Release\CSharpHelloWorld.exe.config".
CopyFilesToOutputDirectory:
Copying file from "obj\x86\Release\CSharpHelloWorld.exe" to "bin\Release\CSharpHelloWorld.exe".
CSharpHelloWorld -> c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld\bin\Release\CSharpHelloWorld.exe
Copying file from "obj\x86\Release\CSharpHelloWorld.pdb" to "bin\Release\CSharpHelloWorld.pdb".
Done Building Project "c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld\CSharpHelloWorld.csproj" (default targets).
Done Building Project "c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld.sln" (Build target(s)).
Build succeeded.
"c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld.sln" (Build target) (1) ->
"c:\jenkins\workspace\CSharpHelloWorld\CSharpHelloWorld\CSharpHelloWorld.csproj" (default target) (2) ->
(CoreCompile target) ->
CSC : warning CS1607: Assembly generation -- Referenced assembly 'mscorlib.dll' targets a different processor
1 Warning(s)
0 Error(s)
Time Elapsed 00:00:00.17
Finished: SUCCESS
http://www.cnblogs.com/itech

Jenkins 插件之环境变量插件 EnvInject

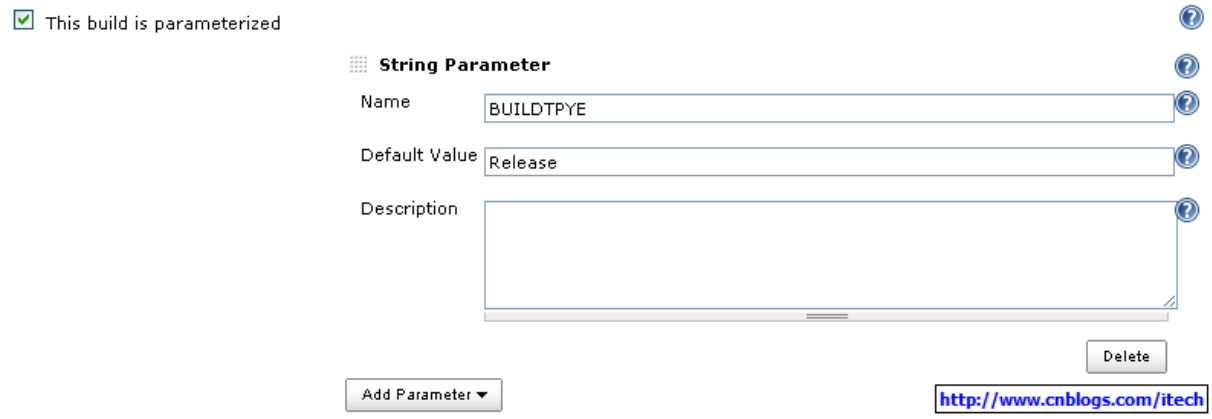
一 Master/Slave 的 Node Properties

用来定义 slave 特定的变量，例如很多的命令所在的路径。



二 job 中的 build parameter

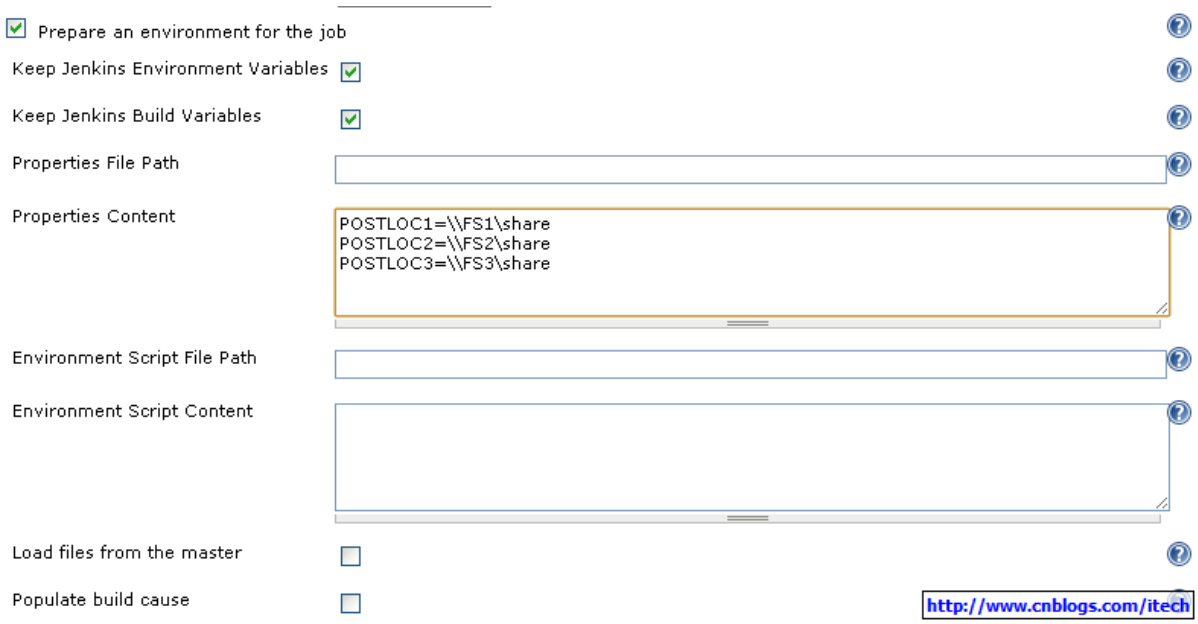
设置后在 build 启动的时候提示修改也可以使用默认值。例如启动改 build 的时候决定是 build release 还是 debug。



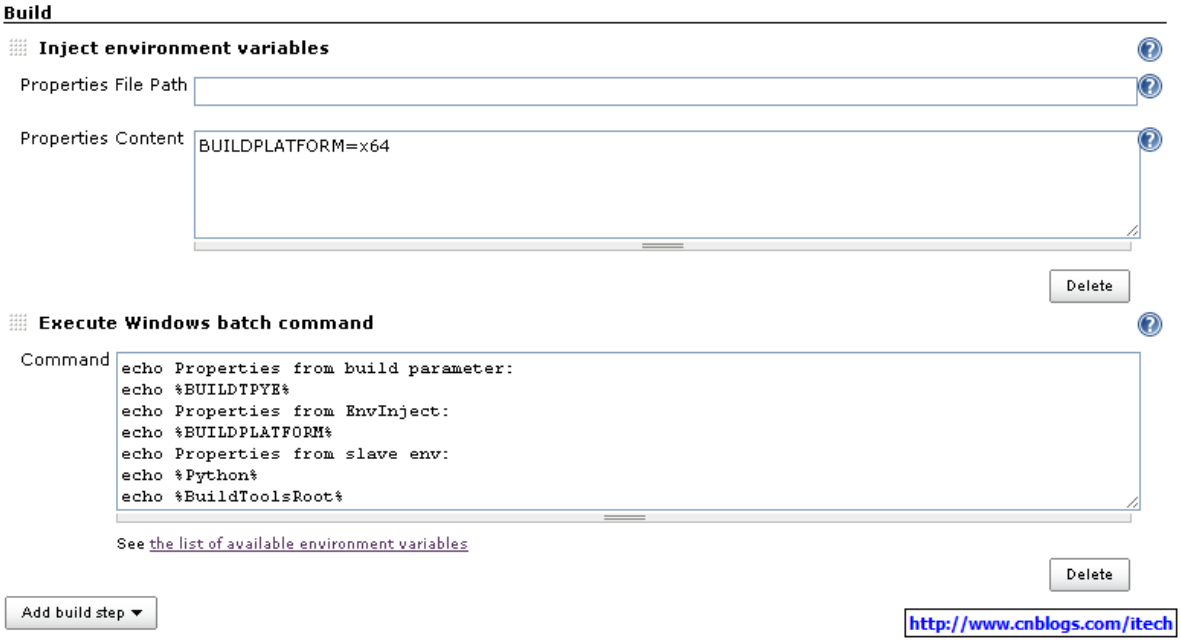
启动 build 时提示如下：

三 EnvInject 插件

需要手动安装此插件，用来对 job 定义环境变量，还可以定义的 ob 的 step 来在 build 的过程中修改环境变量，例如为 job 定义公共的 post location：



在 job 的 step 中修改变量，例如修改 buildplatform 的值：


Build
Inject environment variables
Properties File Path
Properties Content
BUILDPLATFORM=x64
Delete
Execute Windows batch command
Command
echo Properties from build parameter:
echo %BUILDTPYE%
echo Properties from EnvInject:
echo %BUILDPLATFORM%
echo Properties from slave env:
echo %Python%
echo %BuildToolsRoot%
See the list of available environment variables
Delete
Add build step
http://www.cnblogs.com/itech

四 运行结果如下：

Console Output

```
Started by user jenkins
[EnvInject] - Injecting as environment variables the properties content
 'POSTLOC1=\\FS1\share
POSTLOC2=\\FS2\share
POSTLOC3=\\FS3\share'

[EnvInject] - Unset unresolved 'USERNAME' variable.
Building remotely on MyServer01
[EnvInject] - Injecting as environment variables the properties content
 'BUILDPLATFORM=x64'

[EnvInject] - Unset unresolved 'USERNAME' variable.
[ENVTest] $ cmd /c call C:\WINDOWS\TEMP\hudson2094025249050073417.bat

c:\jenkins\workspace\ENVTest>echo Properties from build parameter:
Properties from build parameter:

c:\jenkins\workspace\ENVTest>echo Release
Release

c:\jenkins\workspace\ENVTest>echo Properties from EnvInject:
Properties from EnvInject:

c:\jenkins\workspace\ENVTest>echo x64
x64

c:\jenkins\workspace\ENVTest>echo Properties from slave env:
Properties from slave env:

c:\jenkins\workspace\ENVTest>echo c:\python\python.exe
c:\python\python.exe

c:\jenkins\workspace\ENVTest>echo c:\buildtoolsroot
c:\buildtoolsroot

c:\jenkins\workspace\ENVTest>exit 0
Finished: SUCCESS
```

http://www.cnblogs.com/itech

参考：

EnvInject Plugin； 也可以考虑使用 Tool Environment Plugin 。

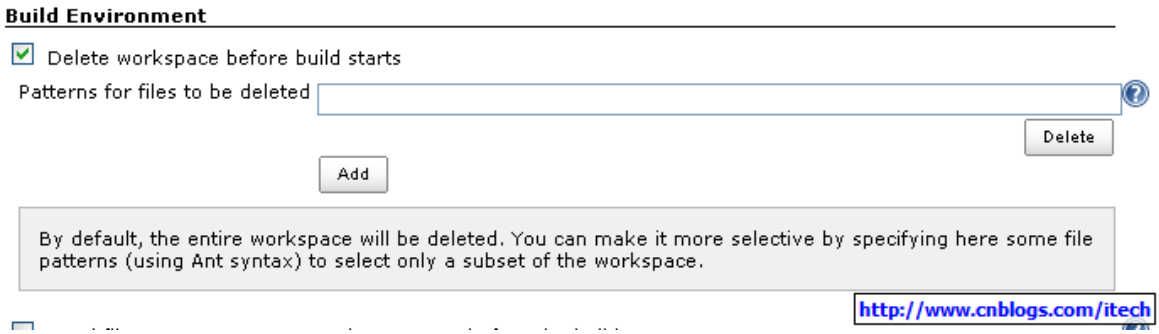
Jenkins 插件之 Workspace cleanup + Copy to slave

Workspace Cleanup Plugin

Copy To Slave Plugin

一 workspace cleanup 插件

用来在 build 开始前或 build 完成后清理 workspace。

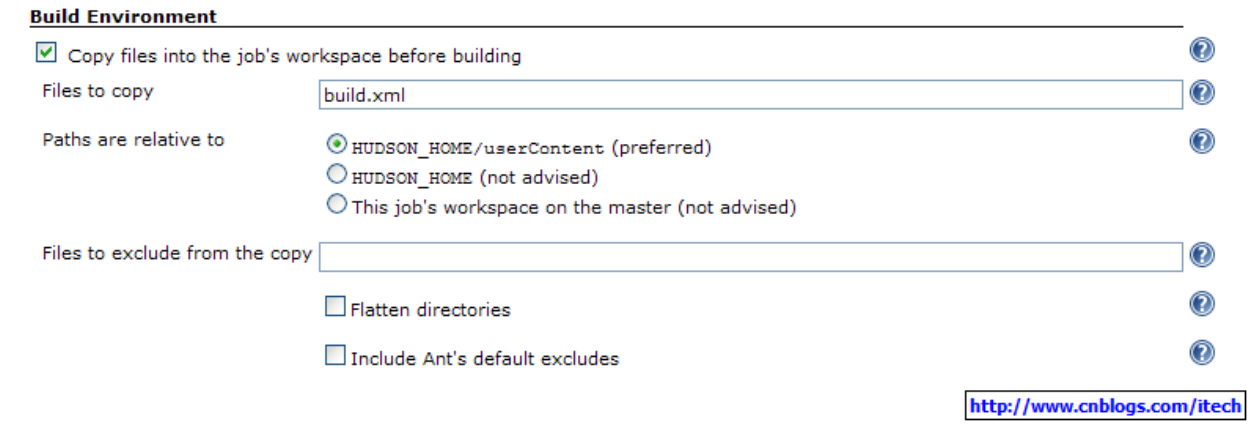


还可以 Post-build actions 中设置 delete workspace when build done。

二 Copy to slave 插件

用来将文件自动地从 master 上拷贝到 slave，或从 slave 拷贝回 master。

使用情况：
1）在 build 前自动地将 build 需要的文件从 master 上拷贝到 build 的 workspace，例如拷贝没有在 SCM 中的 build 脚本；
2）在 build 后从 slave 拷贝文件到 master，这些文件将被下次 build 或其他的 build 使用；


Build Environment
Copy files into the job's workspace before building
Files to copy
build.xml
Paths are relative to
HUDSON_HOME/userContent (preferred)
HUDSON_HOME (not advised)
This job's workspace on the master (not advised)
Files to exclude from the copy
Flatten directories
Include Ant's default excludes
http://www.cnblogs.com/itech

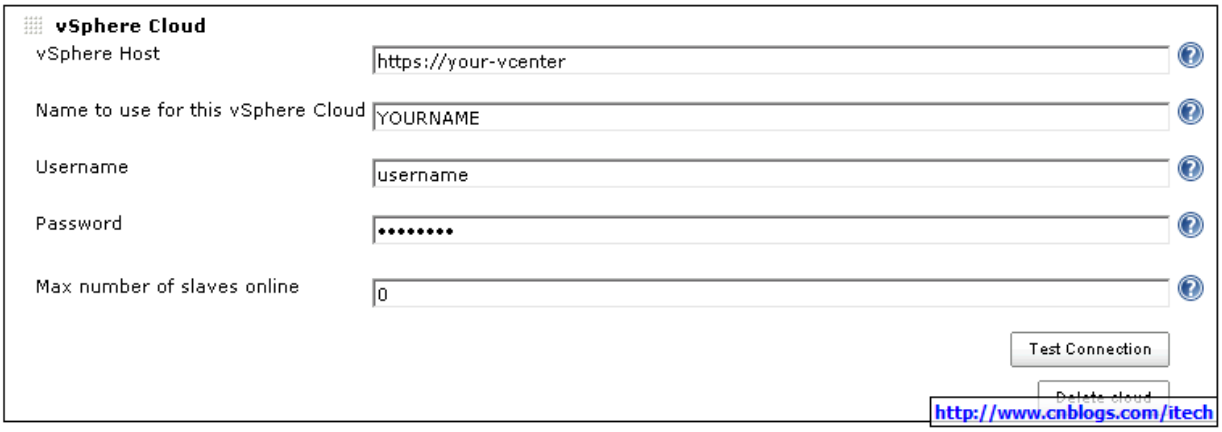
Copy files back to the master
Files to copy
TEST-*.xml
Files to exclude from the copy
http://www.cnblogs.com/itech

Jenkins 插件之 VShpere Cloud

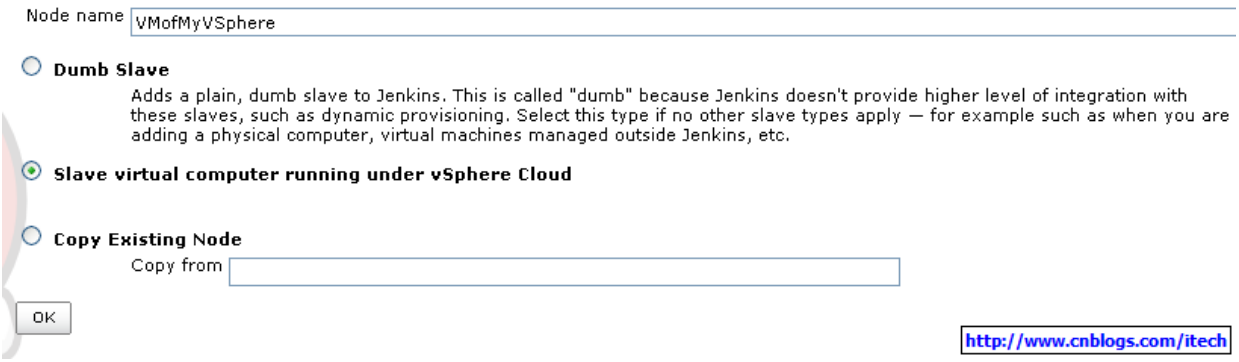
如果我们使用 VShpere 来管理所有的 build 机器，则使用 VSphere Cloud 插件使得虚拟机的管理更加简单，且能够更好地利用 VSphere 的资源。

VShphere Cloud 插件使得 Jenkins 可以控制 VMWare VShpere 中的虚拟机。可以配置 Jenkins 的 slave 为虚拟机，且可以指定 snapshot 的名字。Jenkins 将自动地恢复到设置的 snapshot，然后启动虚拟机作为 slave 来开始 build。在 build 结束后 Jenkins 将自动地关闭 slave，且恢复到指定的 snapshot。

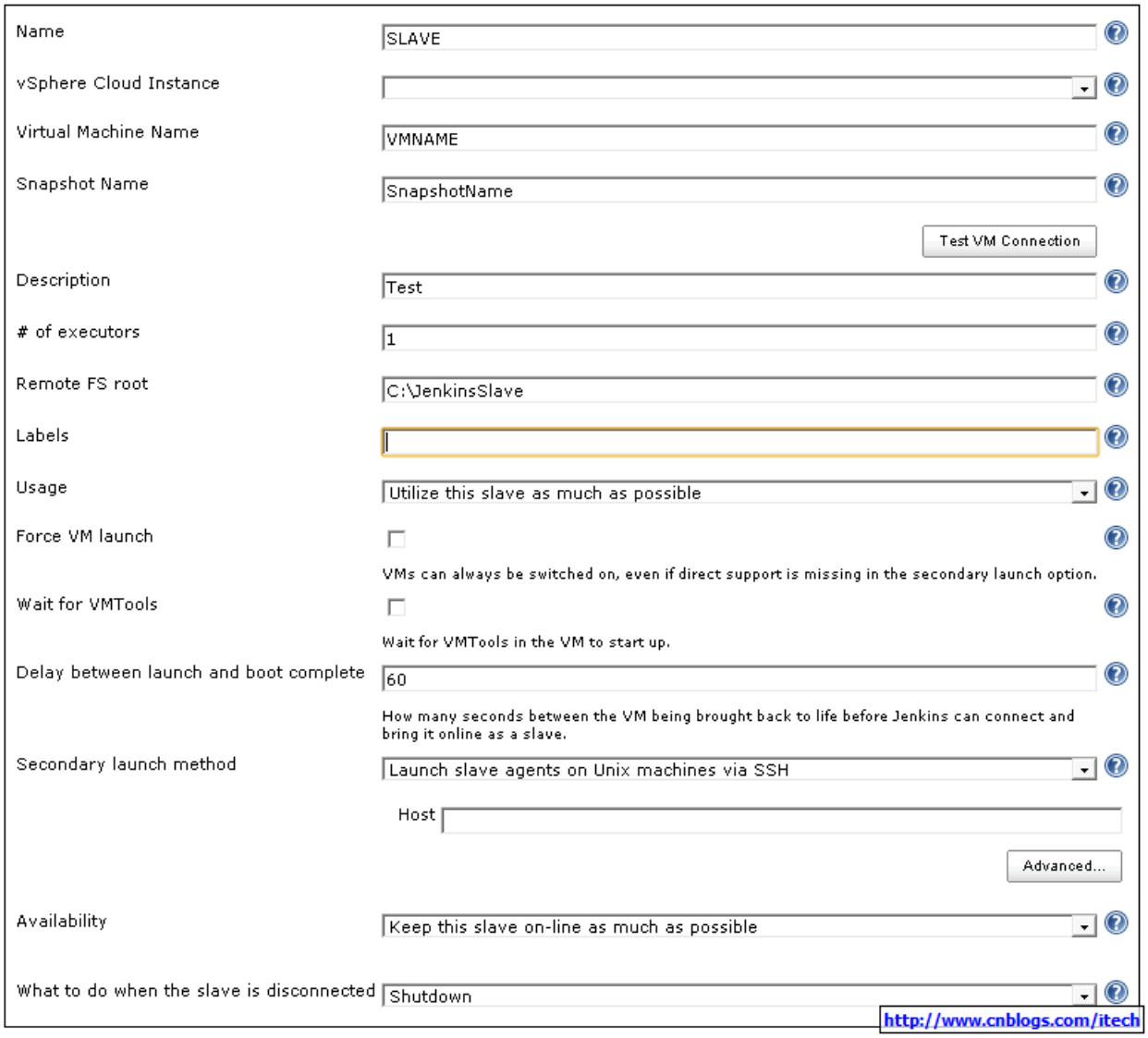
1）首先需要配置 VShpere server，在 Jenkins 的 Configure System 中，如下：
用户需要启动关闭和恢复虚拟机的权限。



2） 创建新的 slave 来使用 VSphere 中的 VM，



+

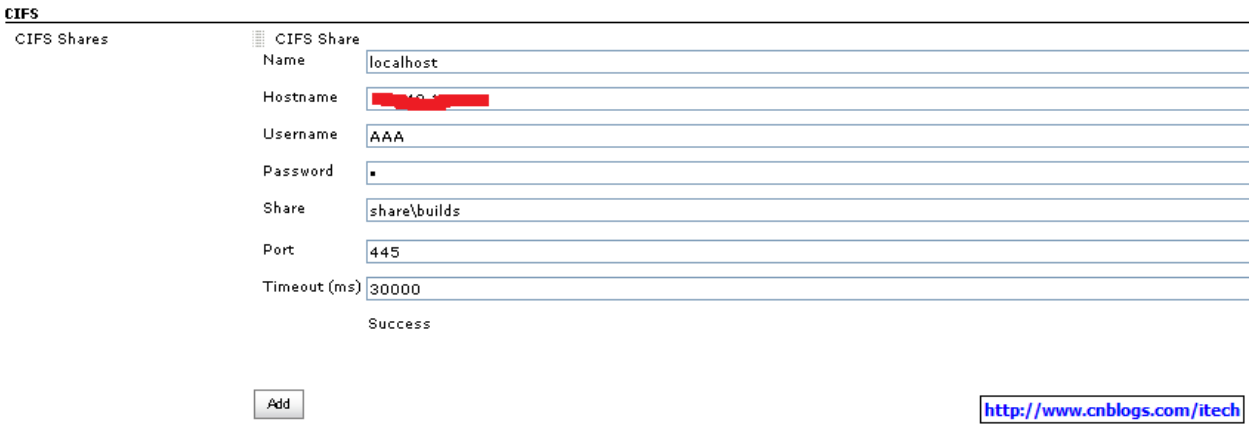

Name
SLAVE
vSphere Cloud Instance
Virtual Machine Name
VMNAME
Snapshot Name
SnapshotName
Test VM Connection
Description
Test
# of executors
1
Remote FS root
C:\JenkinsSlave
Labels
Usage
Utilize this slave as much as possible
Force VM launch
VMs can always be switched on, even if direct support is missing in the secondary launch option.
Wait for VMTools
Wait for VMTools in the VM to start up.
Delay between launch and boot complete
60
How many seconds between the VM being brought back to life before Jenkins can connect and bring it online as a slave.
Secondary launch method
Launch slave agents on Unix machines via SSH
Host
Advanced...
Availability
Keep this slave on-line as much as possible
What to do when the slave is disconnected
Shutdown
http://www.cnblogs.com/itech

Jenkins 插件之 Publish Over SSH/CIFS/FTP

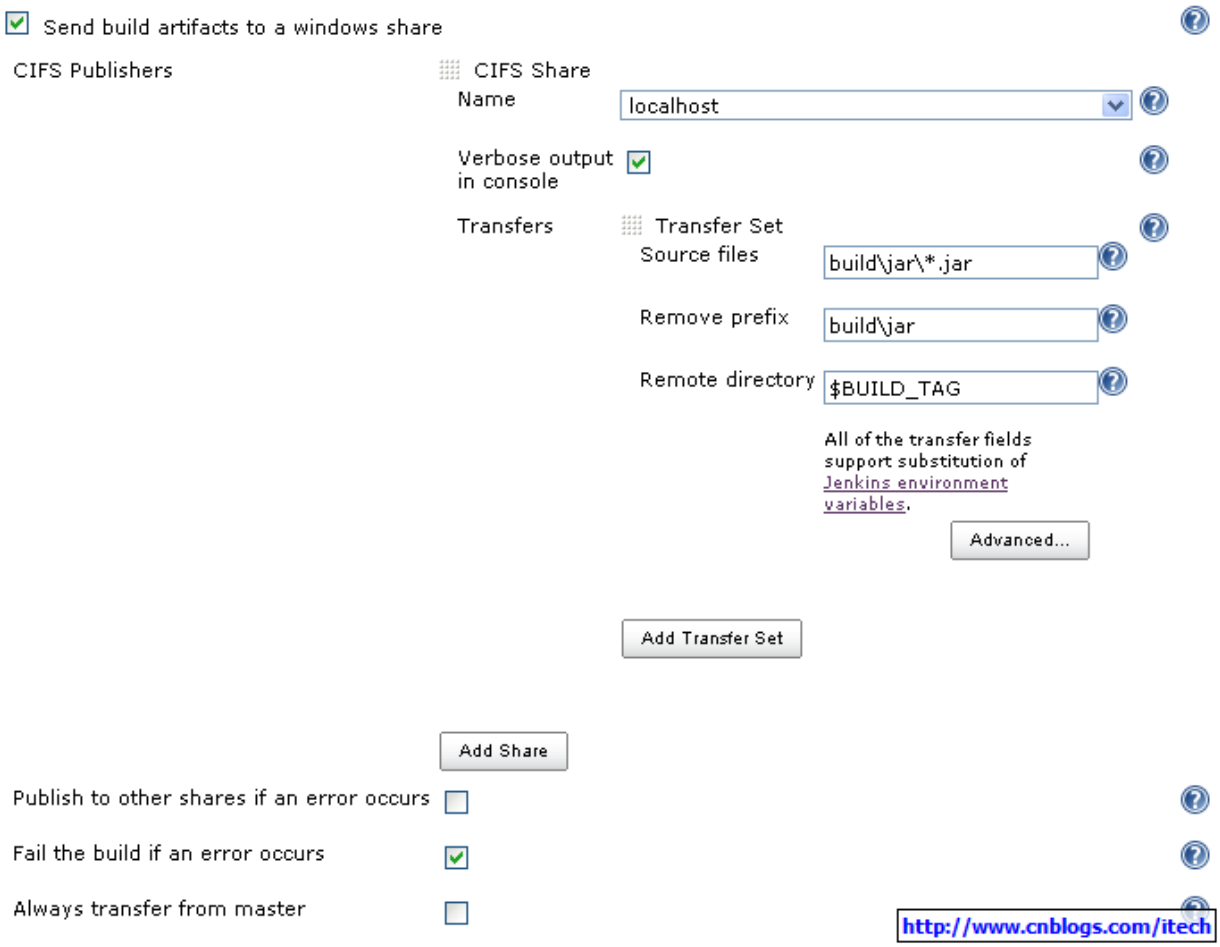
Publish 系列插件用来将 build 的结果发布到 Windows，Linux，FTP 共享。

Publish Over CIFS Plugin
Publish Over FTP Plugin
Publish Over SSH Plugin

一 publish 到 windows share
在 system configure 中配置 windows share 信息



在 job 中使用 publish over CIFS 插件：



运行结果：

```
BUILD SUCCESSFUL
Total time: 1 second
Archiving artifacts
CIFS: Connecting from host [[redacted]]
CIFS: Connecting with configuration [localhost] ...
CIFS: Removing WINS from name resolution
CIFS: Setting response timeout [30,000]
CIFS: Setting socket timeout [35,000]
CIFS: mkdir [smb://******@[redacted]/share/builds/jenkins-JavaHelloWorld-33/]
CIFS: copy [smb://******@[redacted]/share/builds/jenkins-JavaHelloWorld-33/HelloWorld.jar]
CIFS: Transferred 1 file(s)
CIFS: Disconnecting configuration [localhost] ...
```

http://www.cnblogs.com/itech

Jenkins 插件之 Deploy

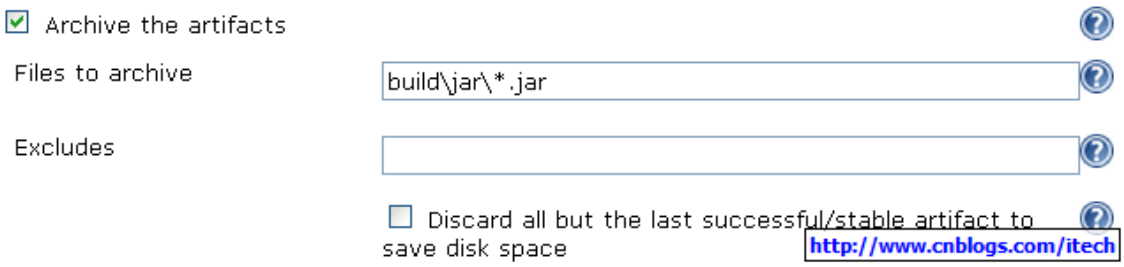
deploy 插件： Deploy Plugin

deploy 插件支持将 War/Jar 部署到远程的应用服务器上，例如 Tomcat,JBoss,Glassfish。

正在寻找或开发.NET web 应用的自动发布插件。

如何回滚或重新部署先前的 build：

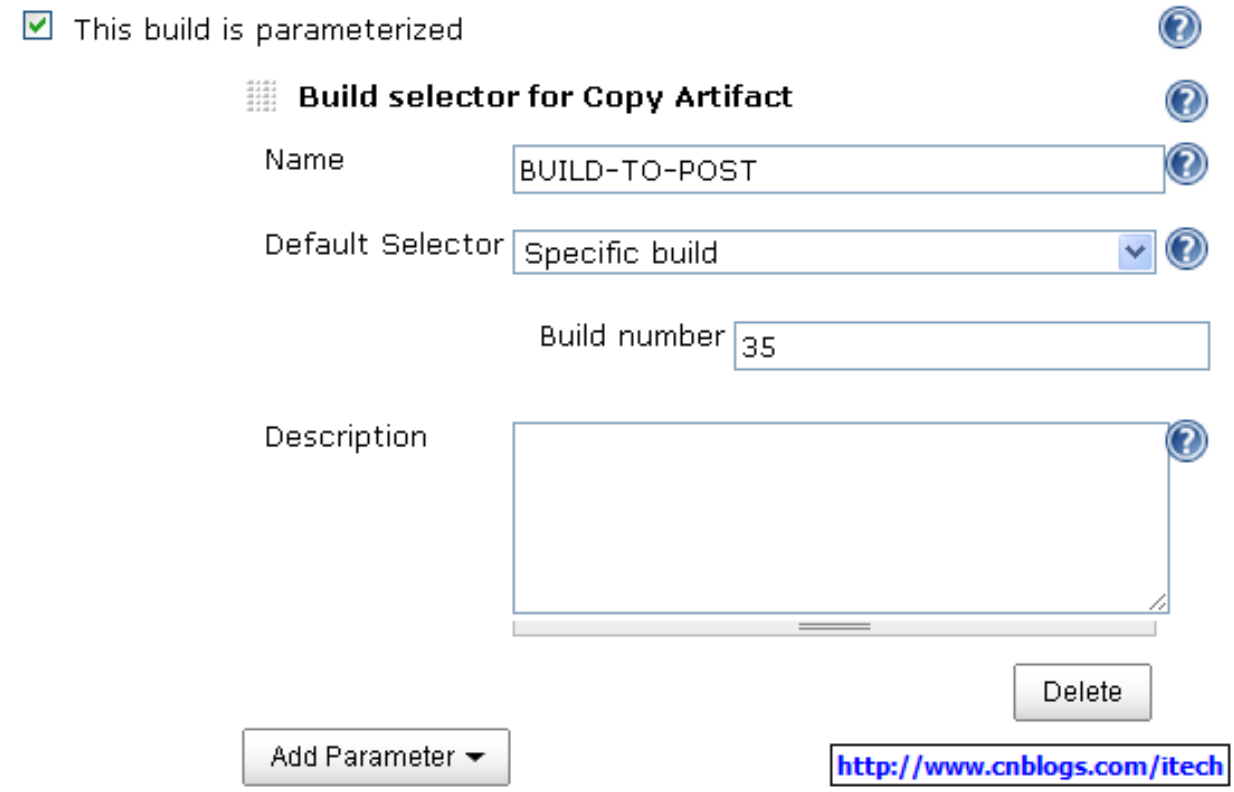
0） 需要被 deploy 的 job 的结果要存档，例如 JavaHelloWorld 的设置如下：



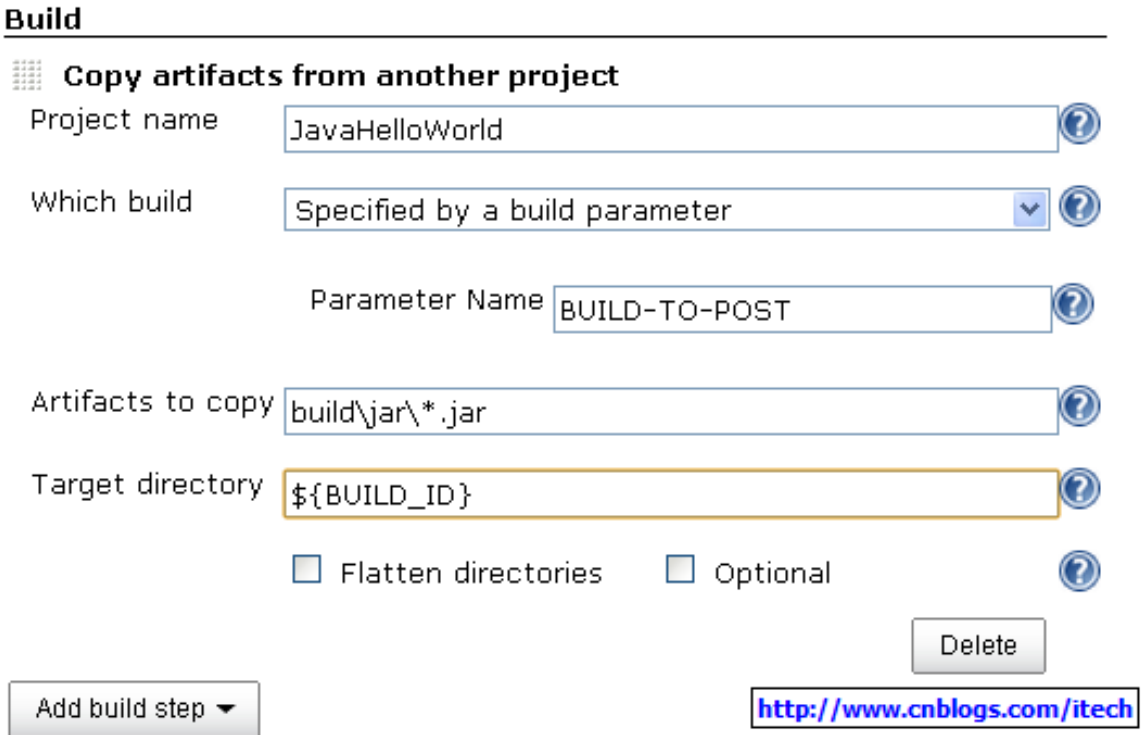
1） 安装 Copy Artifact Plugin；

2） 创建一个 job，在需要的时候手动启动，new job -> build a free-style software project，例如创建 DeployJavaHelloWorld 来 deploy JavaHelloWorld 的结果；

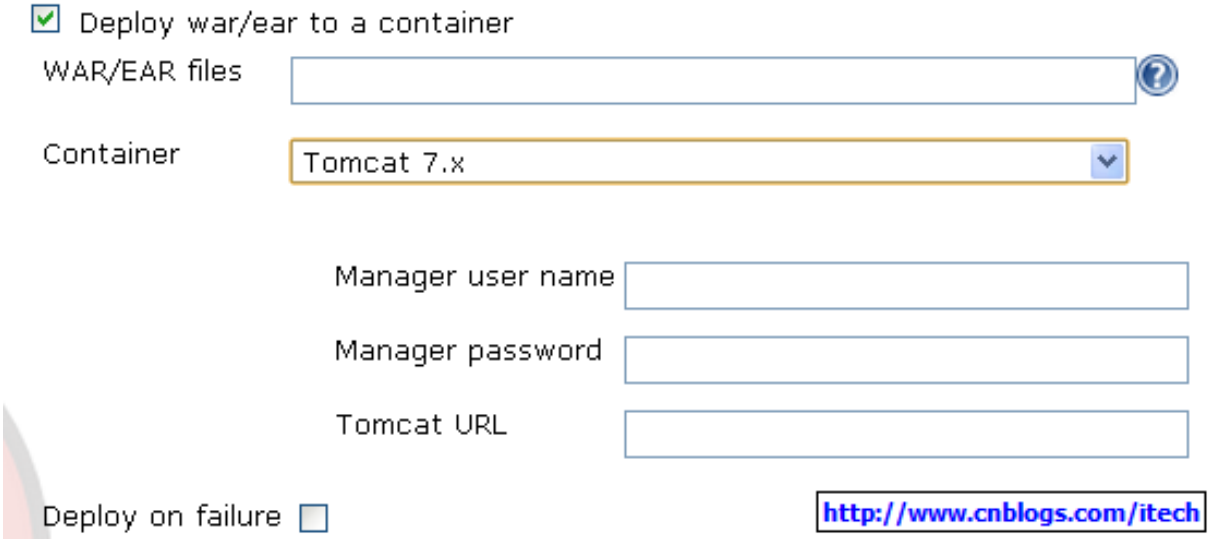
3） 配置 job，且 build 参数的类型为"Build selector for Copy Artifact", 且 copy artifact build 步骤使用"Specified by build parameter"来选择 build；


This build is parameterized
Build selector for Copy Artifact
Name
BUILD-TO-POST
Default Selector
Specific build
Build number
35
Description
Delete
Add Parameter
http://www.cnblogs.com/itech

+


Build
Copy artifacts from another project
Project name
JavaHelloWorld
Which build
Specified by a build parameter
Parameter Name
BUILD-TO-POST
Artifacts to copy
build\jar*.jar
Target directory
${BUILD_ID}
Flatten directories
Optional
Delete
Add build step
http://www.cnblogs.com/itech

4） 增加 post-build action 来 deploy 从其他的 job 拷贝来的 artifact；

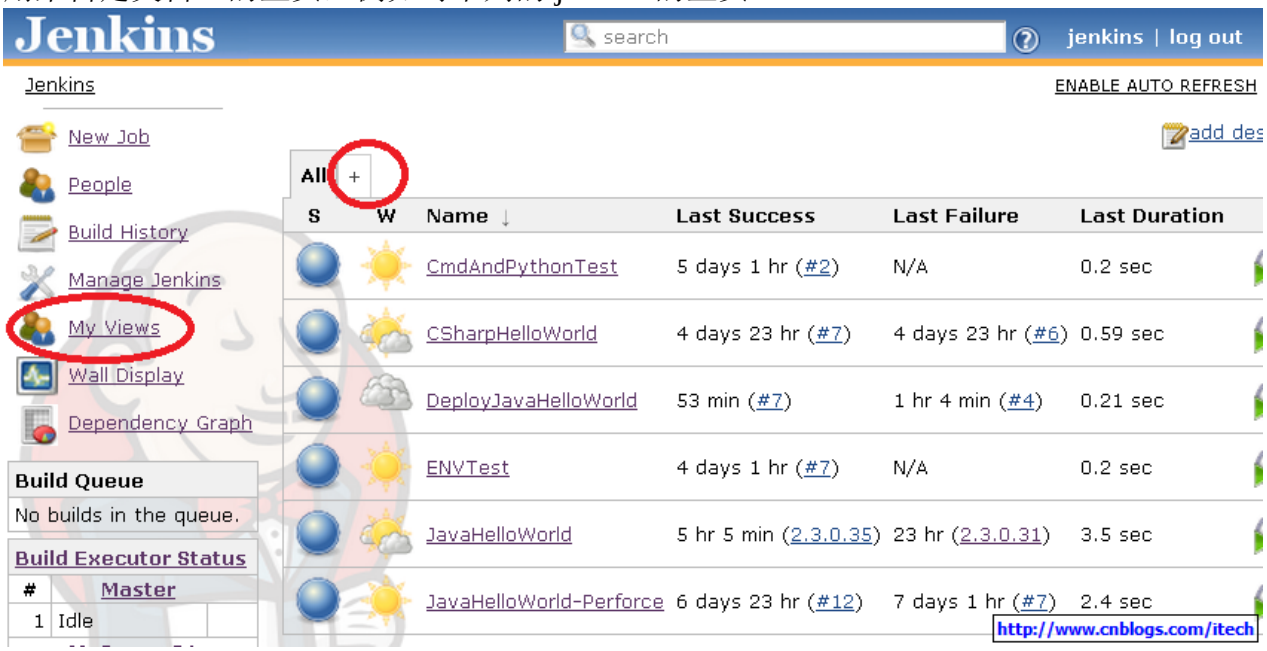


现在你可以启动 job 然后输入 build number 来选择 redploy 那个 build。

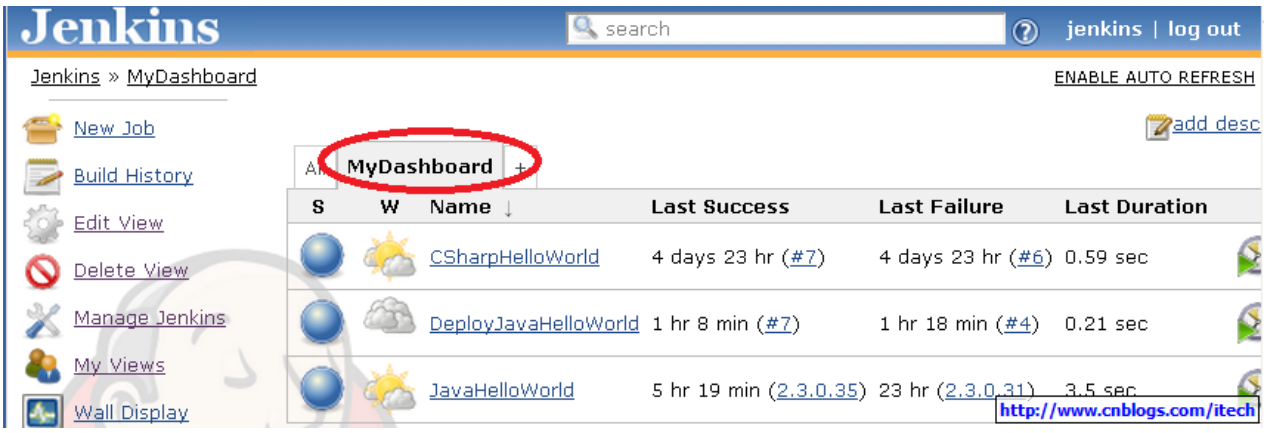
Jenkins 插件之 Dashboard 和 wall display

一 dashboard 插件 Dashboard View

用来自定义自己的主页，例如对下列的 jenkins 的主页



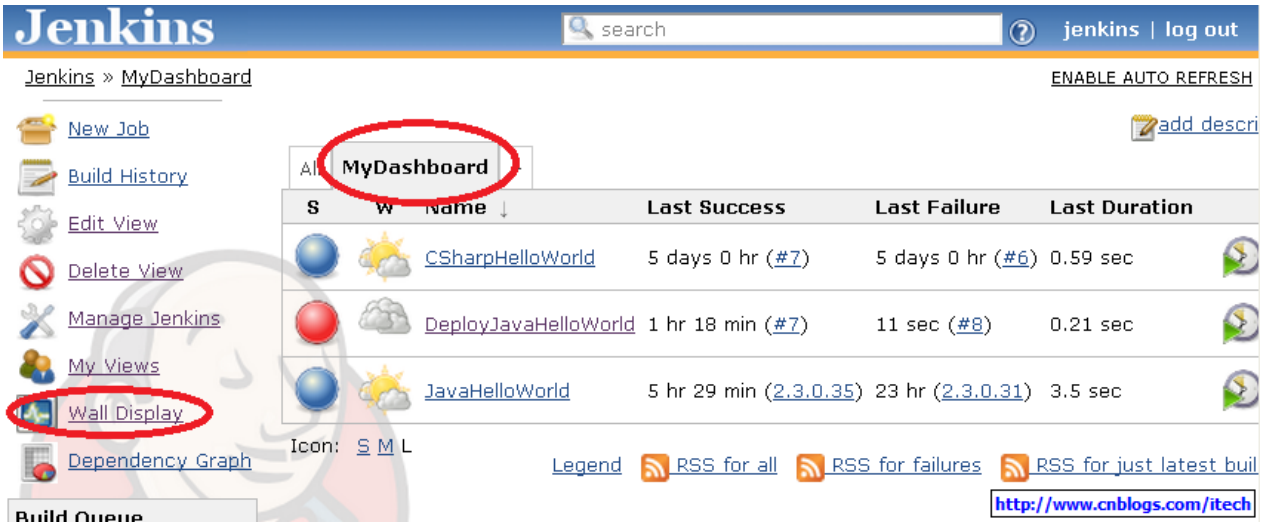
自定义 dashboard 来只显示自己感兴趣的 job：



二 Wall display

用来将 jobs 的状态更加直观地显示在大屏幕上。

例如将 view MyDashboard 显示在大屏幕上如下：



+在 MyDashboard 状态下点击 wall display 进入

Jenkins 插件之有用

Jenkins 整体的插件：

Rebuild Plugin

Slave Setup Plugin
Backup Plugin
Dependency Graph View Plugin

其他的有用的：

Cppcheck Plugin

Static Code Analysis Plug-ins

Doxygen Plugin

NUnit Plugin

xUnit Plugin

JIRA Plugin

Bugzilla Plugin

Trac Plugin

等等