《基于Jenkins的PHP自动构建》

1. Jenkins安装
2. 下载

wget <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>

官网：<https://jenkins.io>

1. 安装

sudo mkdir /usr/local/jenkins

sudo mkdir /usr/local/jenkins/home

sudo mkdir /usr/local/jenkins/logs

sudo mv ./jenkins.war /usr/local/jenkins/jenkins.war

su root

{

vi /etc/profile

export JENKINS\_HOME=/usr/local/jenkins/home

source /etc/profile

或

vi ~/.bash\_profile

export JENKINS\_HOME=/usr/local/jenkins/home

source ~/.bash\_profile

}

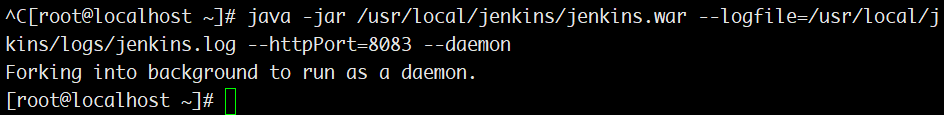
java --jar /usr/local/jenkins/jenkins.war \

--logfile=/usr/local/jenkins/logs/jenkins.log \

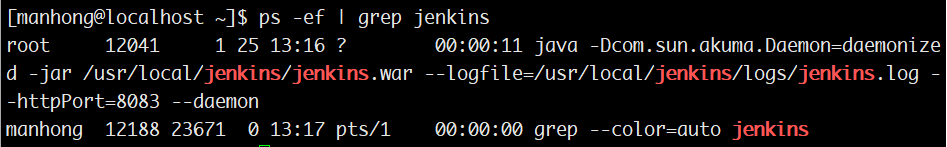
--httpPort=8083 --daemon

1. 首次启动

#命令行启动



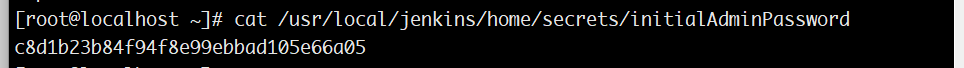
#检查守护进程



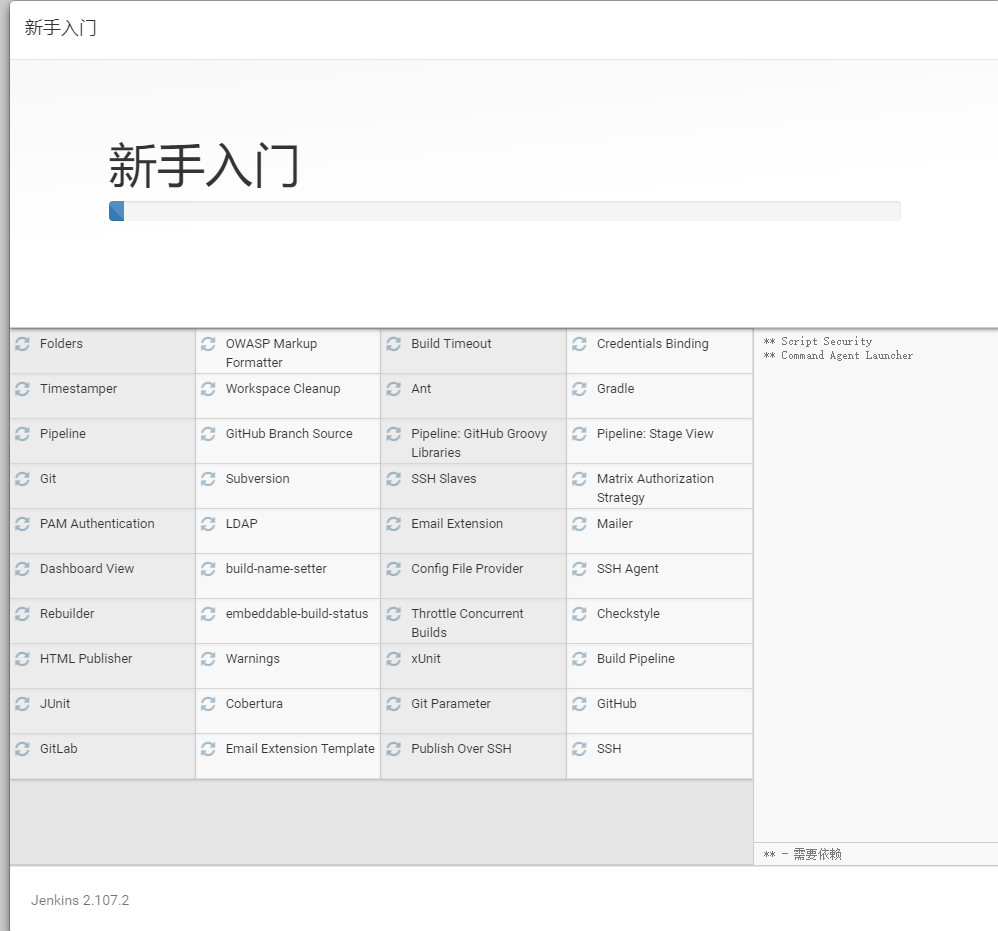
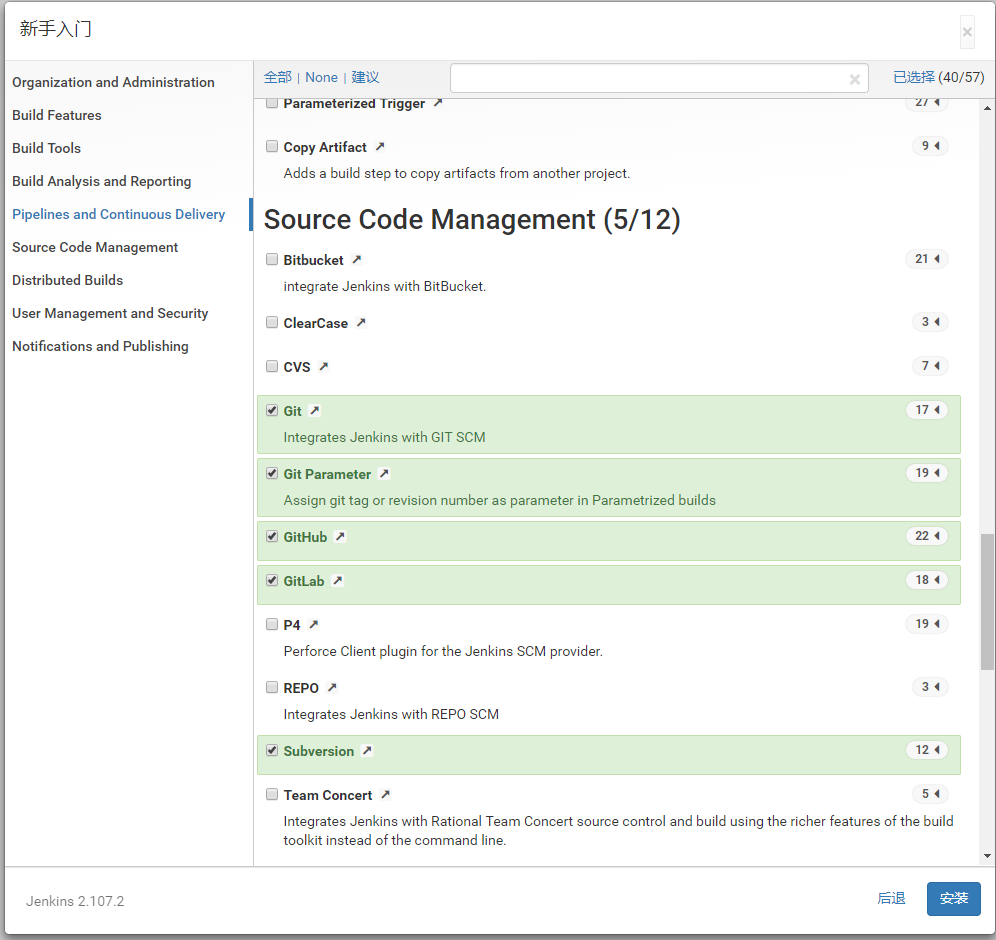
#浏览器访问



#解锁jenkins



#自定义Jenkins插件



#创建管理员账号



#就绪



1. 创建Jenkins任务

#创建新任务



#给任务起名



#保存任务





1. PHP自动化构建工具
2. PHPCS

#安装

wget <https://squizlabs.github.io/PHP_CodeSniffer/phpcs.phar>

chmod +x phpcs.phar

sudo cp phpcs.phar /usr/local/bin/phpcs

1. PHPMD

wget -c <http://static.phpmd.org/php/latest/phpmd.phar>

chmod +x phpmd.phar

sudo cp phpmd.phar /usr/local/bin/phpmd

1. PHCPD

wget <https://phar.phpunit.de/phpcpd.phar>

chmod +x phpcpd.phar

sudo cp phpcpd.phar /usr/local/bin/phpcpd

1. PDEPEND

wget <http://static.pdepend.org/php/latest/pdepend.phar>

chmod +x pdepend.phar

sudo cp pdepend.phar /usr/local/bin/pdepend

1. PHPLOC

wget https://phar.phpunit.de/phploc.phar

chmod +x phploc.phar

sudo cp phploc.phar /usr/local/bin/phploc

1. PHPDOX

wget http://phpdox.de/releases/phpdox.phar

chmod +x phpdox.phar

sudo cp phpdox.phar /usr/local/bin/phpdox

1. PHPUNIT

#PHP7

wget https://phar.phpunit.de/phpunit-7.phar

chmod +x phpunit-7.phar

sudo cp phpunit-7.phar /usr/local/bin/phpunit

#PHP5.6

wget https://phar.phpunit.de/phpunit-5.phar

chmod +x phpunit-5.phar

sudo cp phpunit-5.phar /usr/local/bin/phpunit5

1. PHING

wget <https://www.phing.info/get/phing-2.16.1.phar>

chmod +x phing-2.16.1.phar

sudo cp phing-2.16.1.phar /usr/local/bin/phing

1. 配置PHING
2. 创建部署配置目录及配置文件

sudo mkdir -p /var/deploy/NEW\_TEST/phing

sudo mkdir -p /var/deploy/NEW\_TEST/phpdox

sudo mkdir -p /var/deploy/NEW\_TEST/phpmd

sudo mkdir -p ${JENKINS\_HOME}/workspace/NEW\_TEST/\_\_logs

sudo touch /var/deploy/NEW\_TEST/phing/build.xml

sudo touch /var/deploy/NEW\_TEST/phpdox/gen.xml

#sudo touch /var/deploy/NEW\_TEST/phpmd/rule.xml

PS：NEW\_TEST是Jenkins构建任务名称

1. build.xml公共配置

<property environment="env"/>

<property name="deploy"

value="/var/deploy/${env.JOB\_NAME}" />

<property name="workspace" value="${env.JENKINS\_HOME}/workspace/${env.JOB\_NAME}" />

<property name="bin" value="/usr/local/bin" />

1. 配置PHPCS

#定义可执行文件所在路径

<property name="phpcs" value="${bin}/phpcs" />

#定义执行目标

<target name="phpcs-ci" unless="phpcs-ci.done">

<echo msg="PHPCS:Executing..." level="info" />

<exec executable="${phpcs}" output="/dev/null" taskname="phpcs">

<arg value="--report=checkstyle" />

<arg value="--report-file=${workspace}/\_\_logs/checkstyle.xml" />

<arg value="--standard=PSR2" />

<arg value="--extensions=php" />

<arg value="--ignore=autoload.php" />

<arg path="${workspace}/app/models" />

<arg path="${workspace}/app/controllers" />

</exec>

<property name="phpcs-ci.done" value="true"/>

</target>

1. 配置PHPMD

#定义可执行文件所在路径

<property name="phpmd" value="${bin}/phpmd" />

#定义执行目标

<target name="phpmd-ci" unless="phpmd-ci.done">

<echo msg="PHPMD:Executing..." level="info" />

<exec executable="${phpmd}" taskname="phpmd">

<arg path="${workspace}/app/models,${workspace}/app/controllers" />

<arg value="xml" />

<!--arg path="${deploy}/phpmd/rule.xml" /-->

<arg value="cleancode,codesize,controversial,design,naming,unusedcode" />

<arg value="--reportfile" />

<arg path="${workspace}/\_\_logs/pmd.xml" />

</exec>

<property name="phpmd-ci.done" value="true"/>

</target>

1. 配置PHPCPD

#定义可执行文件所在路径

<property name="phpcpd" value="${bin}/phpcpd" />

#定义执行目标

<target name="phpcpd-ci" unless="phpcpd-ci.done">

<echo msg="PHPCPD:Executing..." level="info" />

<exec executable="${phpcpd}" taskname="phpcpd">

<arg value="--log-pmd" />

<arg path="${workspace}/\_\_logs/pmd-cpd.xml" />

<arg path="${workspace}/app/models" />

<arg path="${workspace}/app/controllers" />

</exec>

<property name="phpcpd-ci.done" value="true"/>

</target>

1. 配置PDEPEND

#定义可执行文件所在路径

<property name="pdepend" value="${bin}/pdepend" />

#定义执行目标

<target name="pdepend" unless="pdepend.done">

<echo msg="PDEPEND:Executing..." level="info" />

<exec executable="${pdepend}" taskname="pdepend">

<arg value="--jdepend-xml=${workspace}/\_\_logs/jdepend.xml" />

<arg value="--jdepend-chart=${workspace}/\_\_logs/dependencies.svg" />

<arg value="--overview-pyramid=${workspace}/\_\_logs/overview-pyramid.svg" />

<arg path="${workspace}/app/models,${workspace}/app/controllers" />

</exec>

<property name="pdepend.done" value="true"/>

</target>

1. 配置PHPLOC

#定义可执行文件所在路径

<property name="phploc" value="${bin}/phploc" />

#定义执行目标

<target name="phploc-ci" unless="phploci-ci.done">

<echo msg="PHPLOC:Executing..." level="info" />

<exec executable="${phploc}" taskname="phploc">

<arg value="--count-tests" />

<arg value="--log-csv" />

<arg path="${workspace}/\_\_logs/phploc.csv" />

<arg value="--log-xml" />

<arg path="${workspace}/\_\_logs/phploc.xml" />

<arg path="${workspace}/app/models" />

<arg path="${workspace}/app/controllers" />

</exec>

<property name="phploc-ci.done" value="true"/>

</target>

1. 配置PHPDOX

#定义可执行文件所在路径

<property name="phpdox" value="${bin}/phpdox" />

#定义执行目标

<target name="phpdox" unless="phpdox.done">

<echo msg="PHPDOX:Executing..." level="info" />

<exec executable="${phpdox}" taskname="phpdox" output="${workspace}/\_\_logs/phpdox\_c.log">

<arg value="--file" />

<arg path="${deploy}/phpdox/gen.xml" />

</exec>

<property name="phpdox.done" value="true"/>

</target>

#配置gen.xml

<?xml version="1.0" encoding="utf-8" ?>

<phpdox xmlns="http://xml.phpdox.net/config">

<project name="NEW\_TEST" source="${JENKINS\_HOME}/NEW\_TEST/app" workdir="${JENKINS\_HOME}/NEW\_TEST/\_\_logs/phpdox">

<collector backend="parser" />

<generator output="${JENKINS\_HOME}/NEW\_TEST /\_\_logs/phpdox">

<build engine="html" output="html"/>

</generator>

</project>

</phpdox>

1. 配置PHPUNIT

#定义可执行文件所在路径

<property name="phpunit" value="${bin}/phpunit" />

#定义执行目标

<target name="phpunit" unless="phpunit.done">

<echo msg="PHPUNIT:Executing..." level="info" />

<exec executable="${phpunit}" returnProperty="result.phpunit" taskname="phpunit" output="${workspace}/\_\_logs/phpunit.log">

<arg value="--configuration"/>

<arg path="${workspace}/phpunit.xml" />

</exec>

<property name="phpunit.done" value="true" />

</target>

#PS: ${workspace}/phpunit.xml指代的单元测试配置文件应该放在被构建项目的版本库中

#验证单元测试是否成功，如果失败则报错并停止构建

<target name="-check-failure">

<fail message="PHPUnit did not finish successfully">

<condition>

<not>

<equals arg1="${result.phpunit}" arg2="0"/>

</not>

</condition>

</fail>

</target>

#兼容CI[PHP5.6]

ln –s /usr/local/php56/bin/php /usr/local/php/bin/php5

cat>phpunit5.sh<<EOF

> cmd="/usr/local/php/bin/php5 /home/manhong/.config/composer/vendor/bin/phpunit5 ${1} ${2}"

> eval $cmd

> exit $?

> EOF

chmod +x phpunit5.sh

sudo cp phpunit5.sh /usr/local/bin/phpunit5.sh

#定义兼容PHP5.6的build.xml

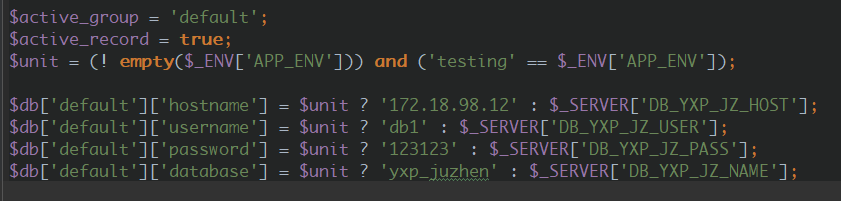
<property name="phpunit" value="${bin}/phpunit5.sh" />

PS：如果要进行代码覆盖率统计则需要安装PHP的XDEBUG扩展，Laravel自带配置文件phpunit.xml和默认测试用例，但是需要配置代码覆盖过滤器和报告文件；而CI连启动脚本都没有。不管是CI还是Laravel都需要根据phpunit.xml中的APP\_ENV环境变量配置单元测试时使用的配置

#phpunit.xml实例，以及CI启动脚本实例和默认测试用例



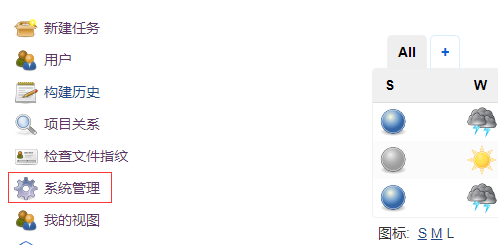
#CI配置文件修改例子



1. 完整build.xml实例

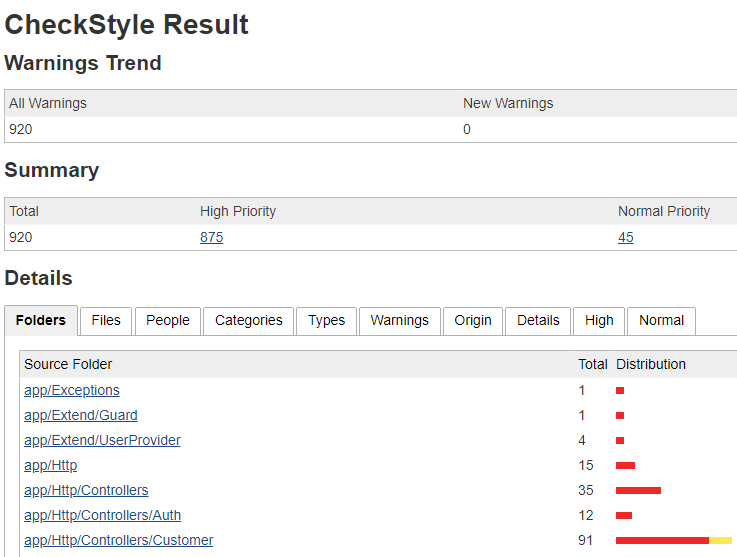


1. Jenkins插件安装及配置
2. Jenkins插件安装

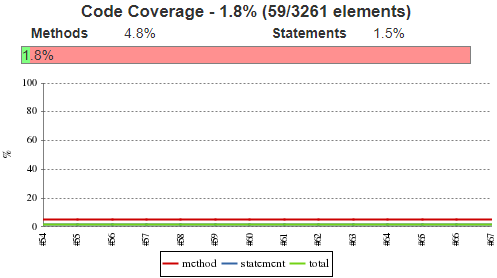


1. PHP自动化构建相关插件

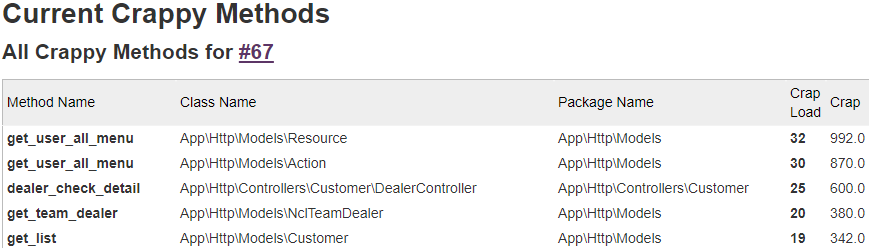
#[Checkstyle](http://wiki.jenkins-ci.org/display/JENKINS/Checkstyle+Plugin) 用于展示PHPCS产生的报告



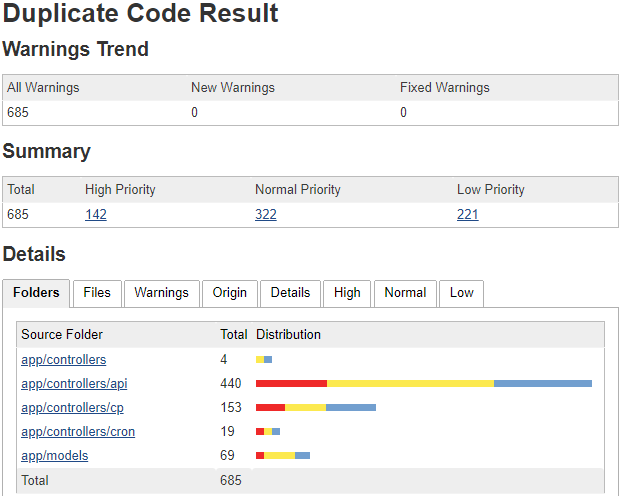
#[Clover PHP](http://wiki.jenkins-ci.org/display/JENKINS/Clover+PHP+Plugin) 展示PHPUNIT生成的代码覆盖率报告



#[Crap4J](http://wiki.jenkins-ci.org/display/JENKINS/Crap4J+Plugin) 展示由PHPUNIT生成的垃圾代码报告



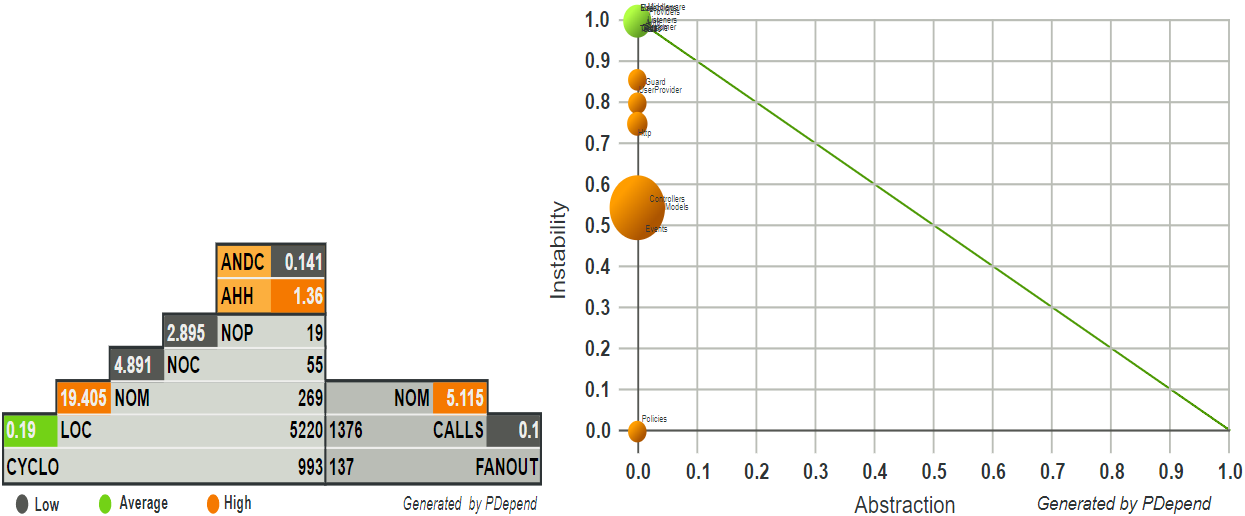
#[DRY](http://wiki.jenkins-ci.org/display/JENKINS/DRY+Plugin) 展示由PHPCPD生成代码复制粘贴报告



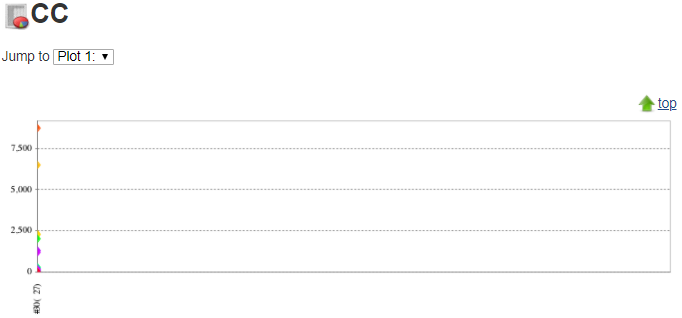
#[HTML Publisher](http://wiki.jenkins-ci.org/display/JENKINS/HTML+Publisher+Plugin) 用于展示PHPDOX生成的代码文档



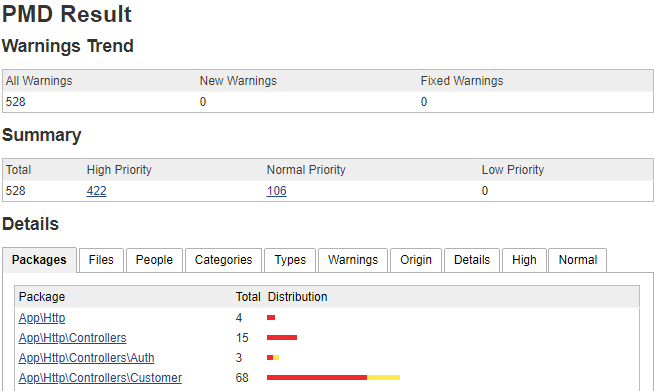
#[JDepend](http://wiki.jenkins-ci.org/display/JENKINS/JDepend+Plugin) 用于展示用PDEPEND生成的代码概览



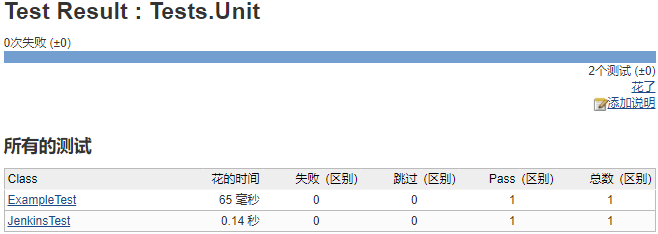
#[Plot](http://wiki.jenkins-ci.org/display/JENKINS/Plot+Plugin) 用于展示PHPLOC生成的代码圈复杂度报告



#[PMD](http://wiki.jenkins-ci.org/display/JENKINS/PMD+Plugin) 用于展示PHPMD生成代码臭味报告



#[xUnit](http://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin) 用于展示PHPUNIT生成的单元测试结果报告

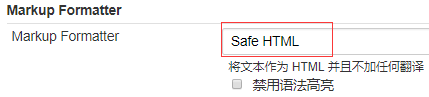
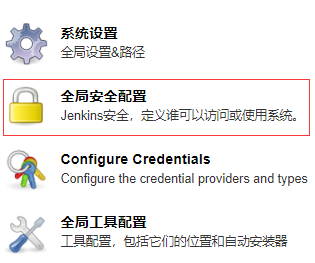


#[Git](http://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin) 负责从版本库拉取代码到工作空间，SVN默认已安装

#[Phing plugin](https://wiki.jenkins-ci.org/display/JENKINS/Phing+Plugin) PHP自动构建工具

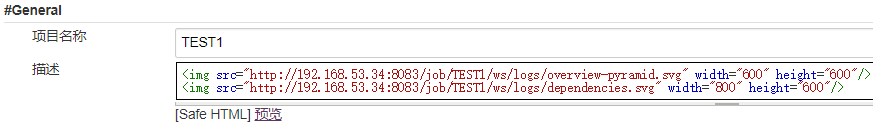
1. Jenkins构建任务配置
2. 任务基本信息配置

#配置超文本任务描述



#配置项目名称和描述



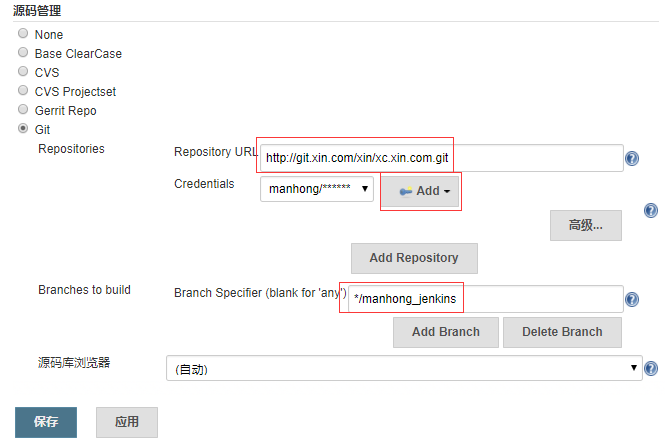


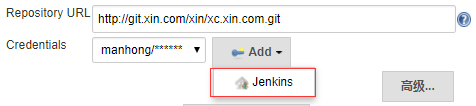
<img src="http://192.168.53.34:8083/job/TEST1/ws/logs/overview-pyramid.svg" width="600" height="600"/>

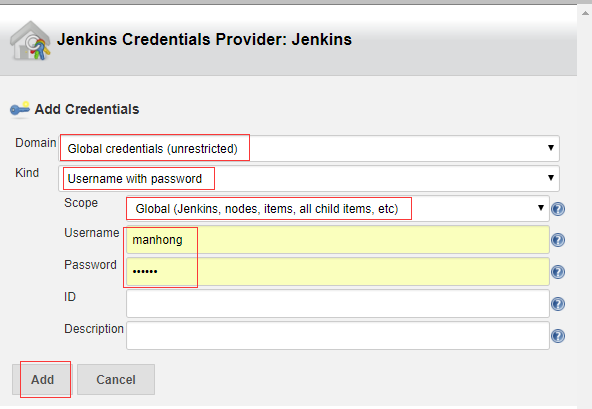
<img src="http://192.168.53.34:8083/job/TEST1/ws/logs/dependencies.svg" width="800" height="600"/>

PS：描述中若无这两个IMG标签，则任务主页中不会显示PDEPEND生成的图表

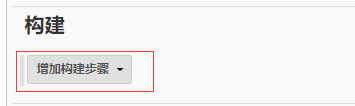
#配置版本库

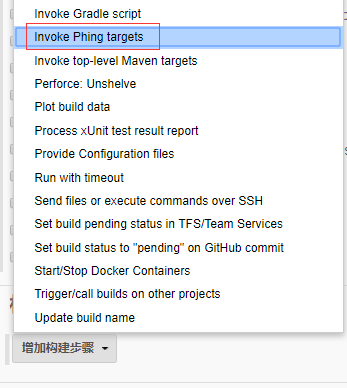


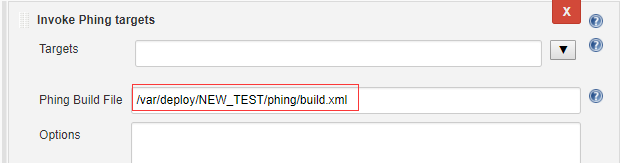




1. Phing插件配置

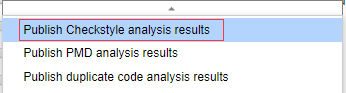


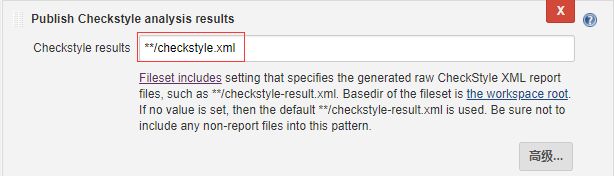




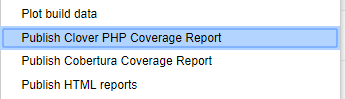
1. Checkstyle插件配置

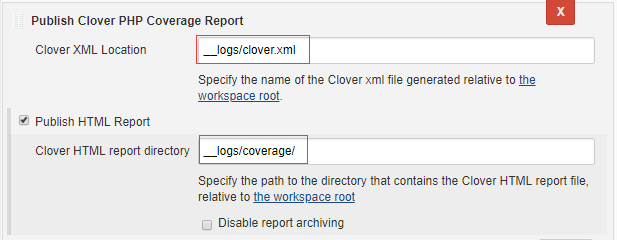




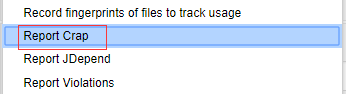


1. Clover PHP插件配置



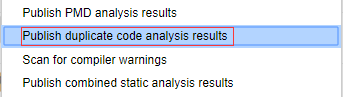


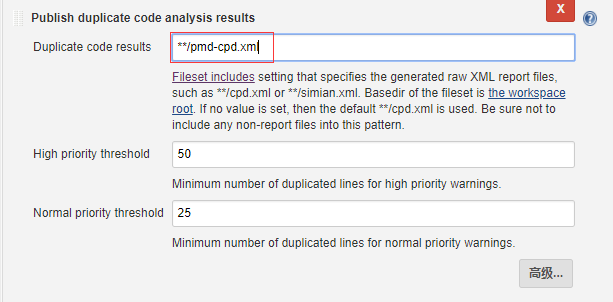
1. Crap4J插件配置



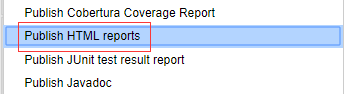


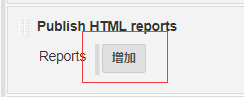
1. DRY插件配置

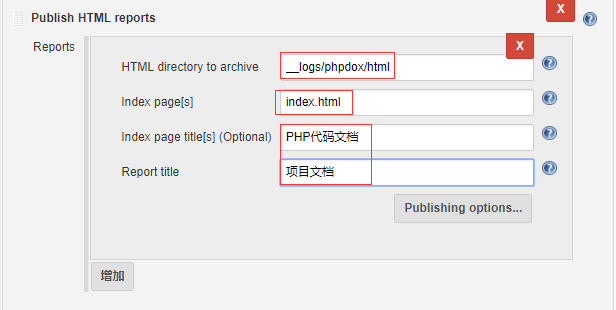




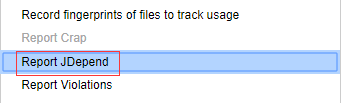
1. HTML Publisher插件配置

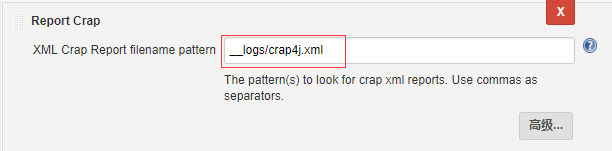




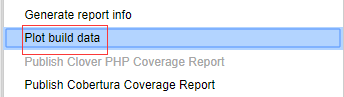


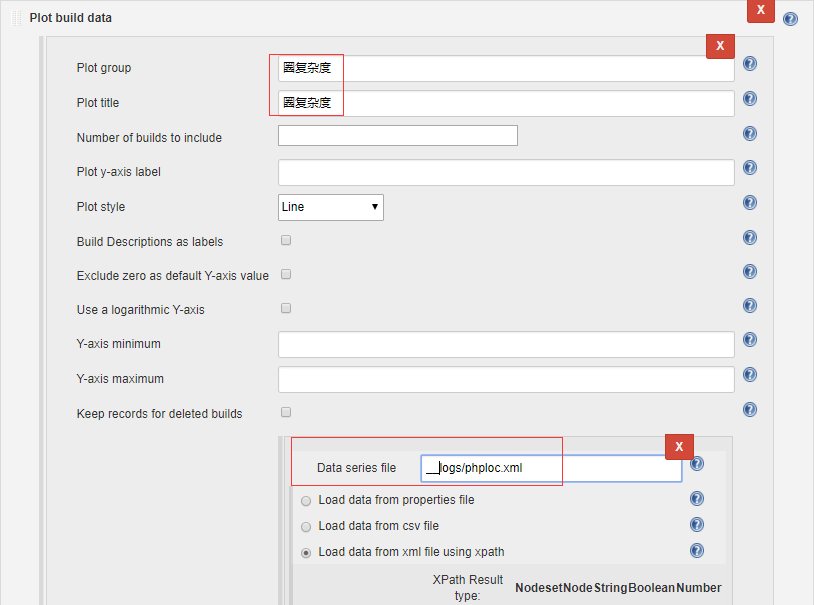
1. JDepend插件配置



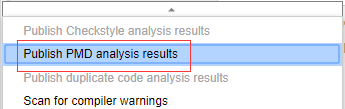


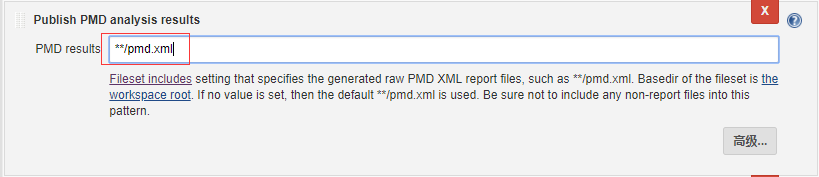
1. Plot插件配置



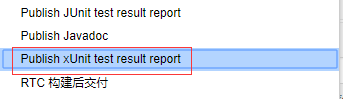


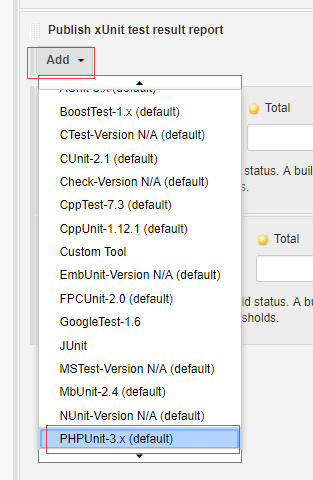
1. PMD插件配置

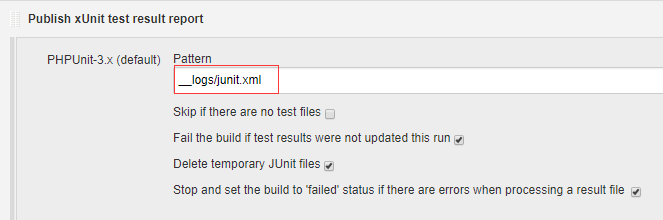




1. xUnit插件配置







1. Jenkins任务构建结果

