|  |  |
| --- | --- |
| **项目编号** | **20160617** |
| **文档编号** | **01** |
| **密 级** | **绝密** |

# 基于Jenkins持续集成

**成都优易数据有限公司**

评 审 日 期： 2016年06月17日

|  |  |  |  |
| --- | --- | --- | --- |
| 编制说明 | | | |
| 编制人 | 编制时间 | 版本号 | 概述 |
| 都柯 | 2016-06-17 | 1.0 | Jenkins的搭建和任务的使用。 |
| 都柯 | 2016-06-23 | 1.1 | 更新持续集成部分描述，增加“构建Jenkins任务”章节 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 安装Jinkens（Linux）

安装Jinkens之前，需要在linux系统中安装JDK,MAVEN和Tomcat，官方下载最新的Jinkens war包，部署在tomcat webapps目录下。

注意：由于Jinkens 在跑job的时候需要内存比较大，根据实际使用情况增加tomcat的内存。

修改tomcat内存大小（catalina.sh）

位置cygwin=false前

JAVA\_OPTS='-Xms256m -Xmx5120m -XX:MaxNewSize=1024m -XX:MaxPermSize=1024m'

修改jenkins 的config.xml文件

<hudson>

<disabledAdministrativeMonitors/>

<version>jenkins版本</version>

<numExecutors>3</numExecutors>

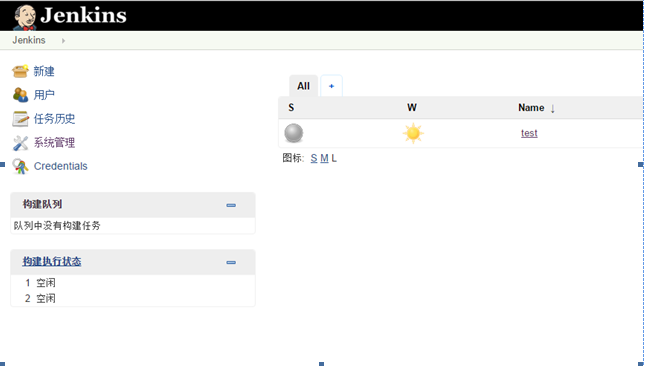
<mode>NORMAL</mode>

<useSecurity>true</useSecurity>

<authorizationStrategy class="hudson.security.AuthorizationStrategy$Unsecured"/>

<securityRealm class="hudson.security.SecurityRealm$None"/>

启动Tomcat后，访问<http://xxx.xx.x.x:8080/jenkins/>,进入Jinkens系统



## 系统配置

## 配置JAVA\_HOME和MAVEN\_HOME

进入“系统管理”-> “系统设置”



分别配置JDK和Maven。



## 插件安装

进入“系统管理”->“管理插件”

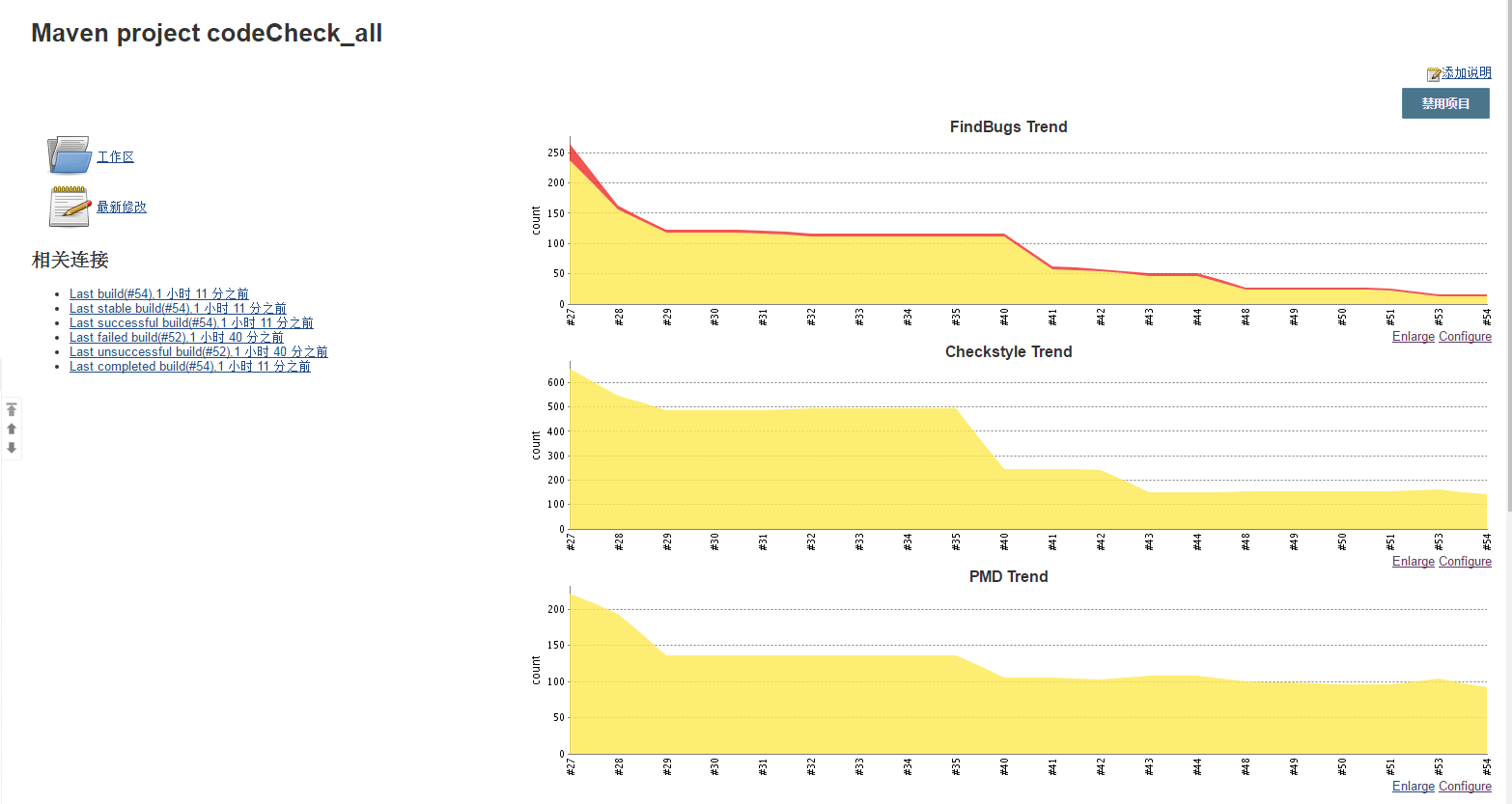


安装checkstyle, pmd,findbug, Static Analysis Collector, [JUnit Plugin](https://wiki.jenkins-ci.org/display/JENKINS/JUnit+Plugin), emma, Publish Over SSH,Zentimestamp plugin，

SVN Publisher plugin(SVN),Locale plugin(汉化),Deploy to container Plugin插件

说明：前面4个插件用于静态检查，[JUnit Plugin](https://wiki.jenkins-ci.org/display/JENKINS/JUnit+Plugin)h和emma分别用于JAVA单元测试报告和代码覆盖率，Publish Over SSH用于远程部署，Zentimestamp plugin用于获取构建job的时间。

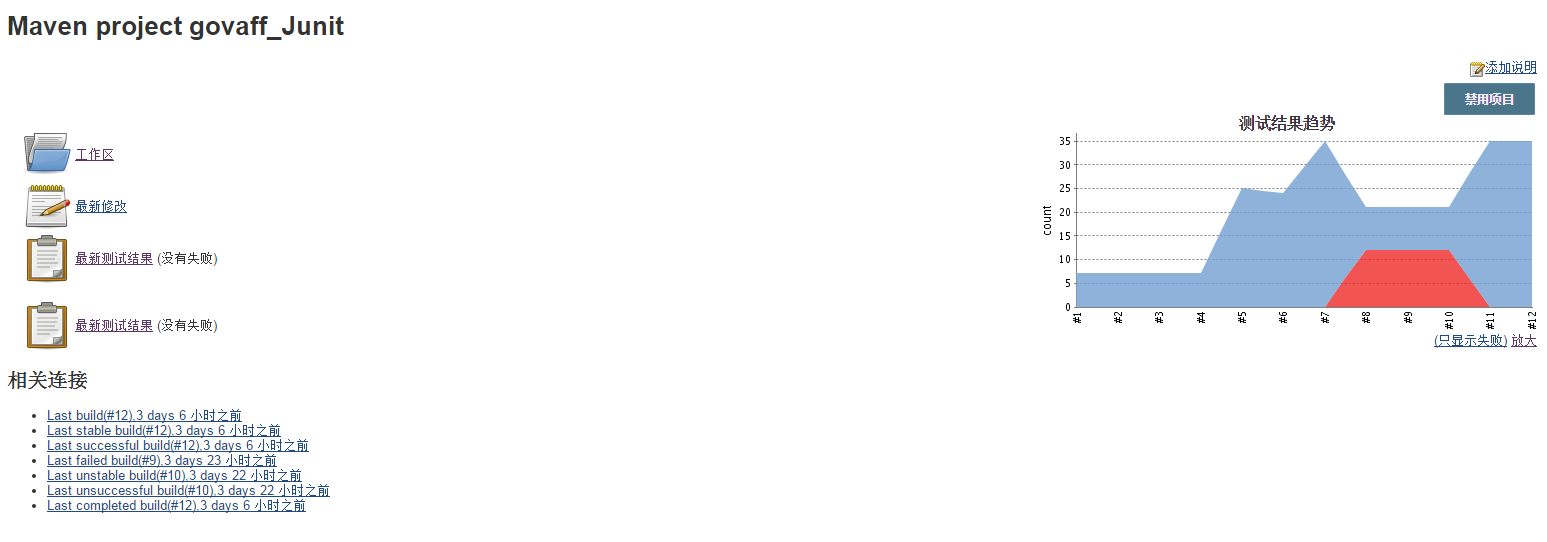
## 静态检查任务



任务codeCheck\_all是通过Jenkins检查JAVA代码质量，其中包括（checkstyle, findbugs和pmd）, 跑此任务需要在pom.xml中添加相关配置信息

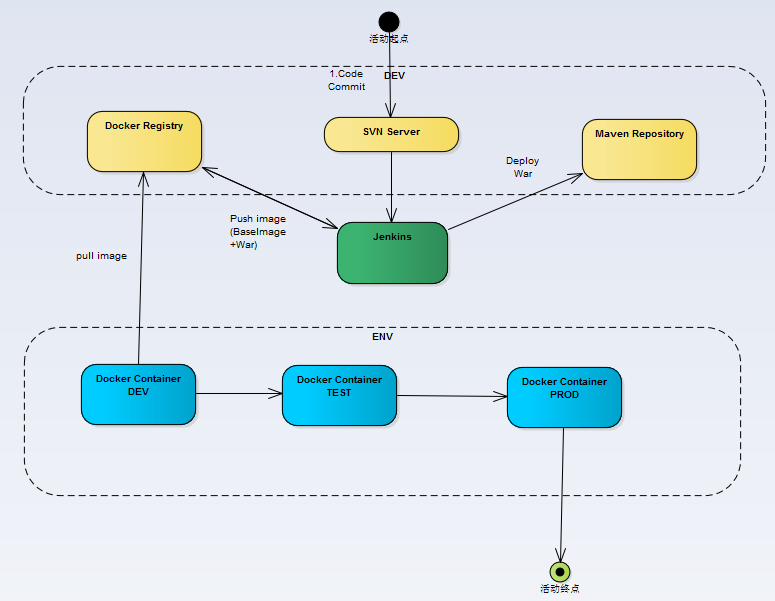


## 单元测试



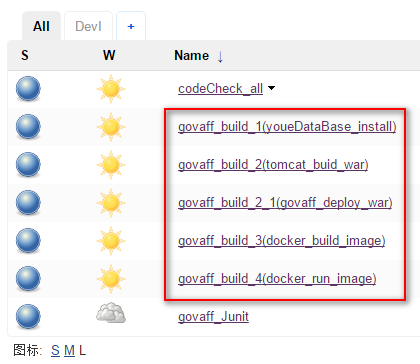
任务govaff\_Junit用于运行govaff代码的单元测试，并把测试结果显示在任务中。计划后续加入代码覆盖插件到此任务。

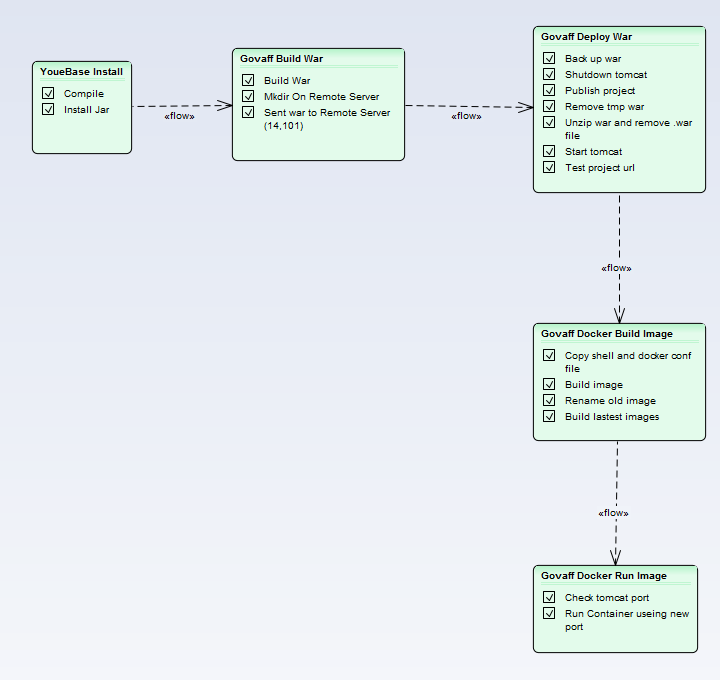
## 持续集成任务



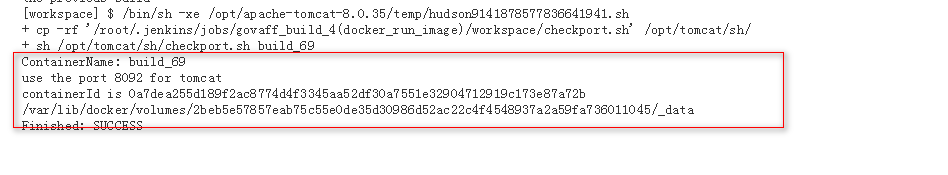
我们通过Docker和Jenkins实现了持续集成（CI）。定时触发Jenkins Job进行自动构建，如果构建成功便会从Docker Registry里面拿到[Base Tomcat Image](#_Docker_镜像层)和构建产生的war包，生成Docker Image并push到镜像仓库，更新maven仓库中的文件。基于这个Docker Image，通过Jenkins job可以运行docker 命令pull Docker Image到开发环境中，最后可以通过PaaS的API或界面，进行开发测试环境的实例更新，并最终进行生产环境的实例更新，从而实现持续集成和持续交付。

为了达到以上目标，我们在Jenkins中创建了5个job来完成持续集成（目前只针对开发环境）





* **govaff\_build\_1(youeDataBase\_install)**：该任务主要用于编译和打包youeBase中的代码库，并把打出的jar包更新的Maven仓库。
* **govaff\_build\_2(tomcat\_buid\_war)：**该任务用于govaff源码的编译和打包，同时打出war包到Maven 仓库，并copy相关文件（docker配置文件，war文件）到remote Server(开发环境172.16.0.14和Docker 开发环境172.16.0.102)
* **govaff\_build\_2\_1(govaff\_deploy\_war)：**部署war文件到Tomcat（目前是172.16.0.14）并重启tomcat。
* **govaff\_build\_3(docker\_build\_image)：**从Docker registry中获取Base Tomcat image，加上第二部中的war包，build出最新的image到主机上172.16.0.102
* **govaff\_build\_4(docker\_run\_image)：**在Docker从上一步创建的镜像中启动一个docker的tomcat容器，并在Jenkins控制台输出该容器的一些信息。

****

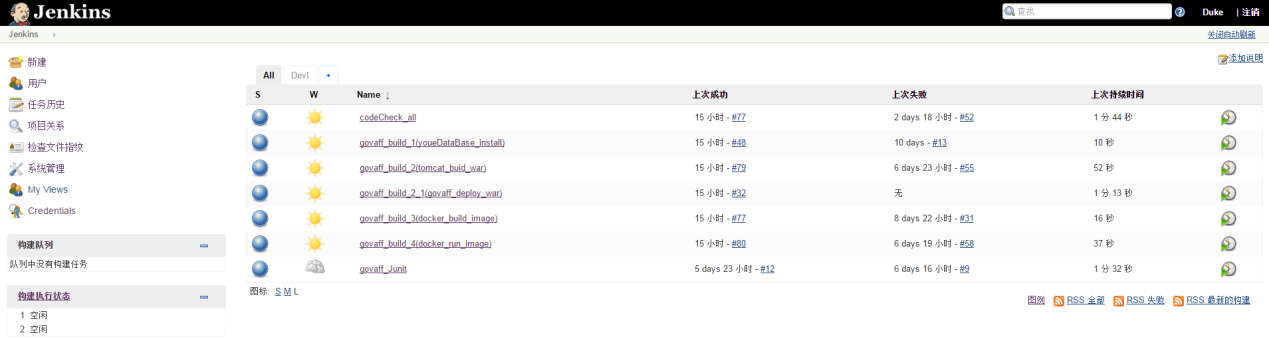
有了这些信息，开发人员可以到主机上手动启停删除这些容器，并可以到相关目录上去替换一些配置文件和class文件

*附录：用户访问地址：<http://172.16.0.101:xxxx/govaff-web/>*

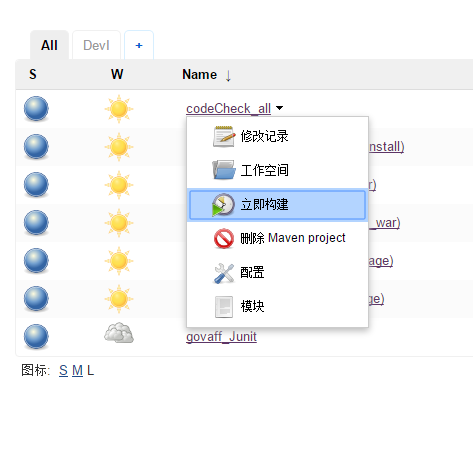
*Docker 命令重启tomcat容器命令：docker restart <containerId>*

说明：这5个任务会依次build，且向后的任务依赖前一个任务。

## 构建Jenkins任务



如上图所示，有了任务后我们就可以构建这些任务了



如上图所示，点击任务右边的箭头，再点击“立即构建”。



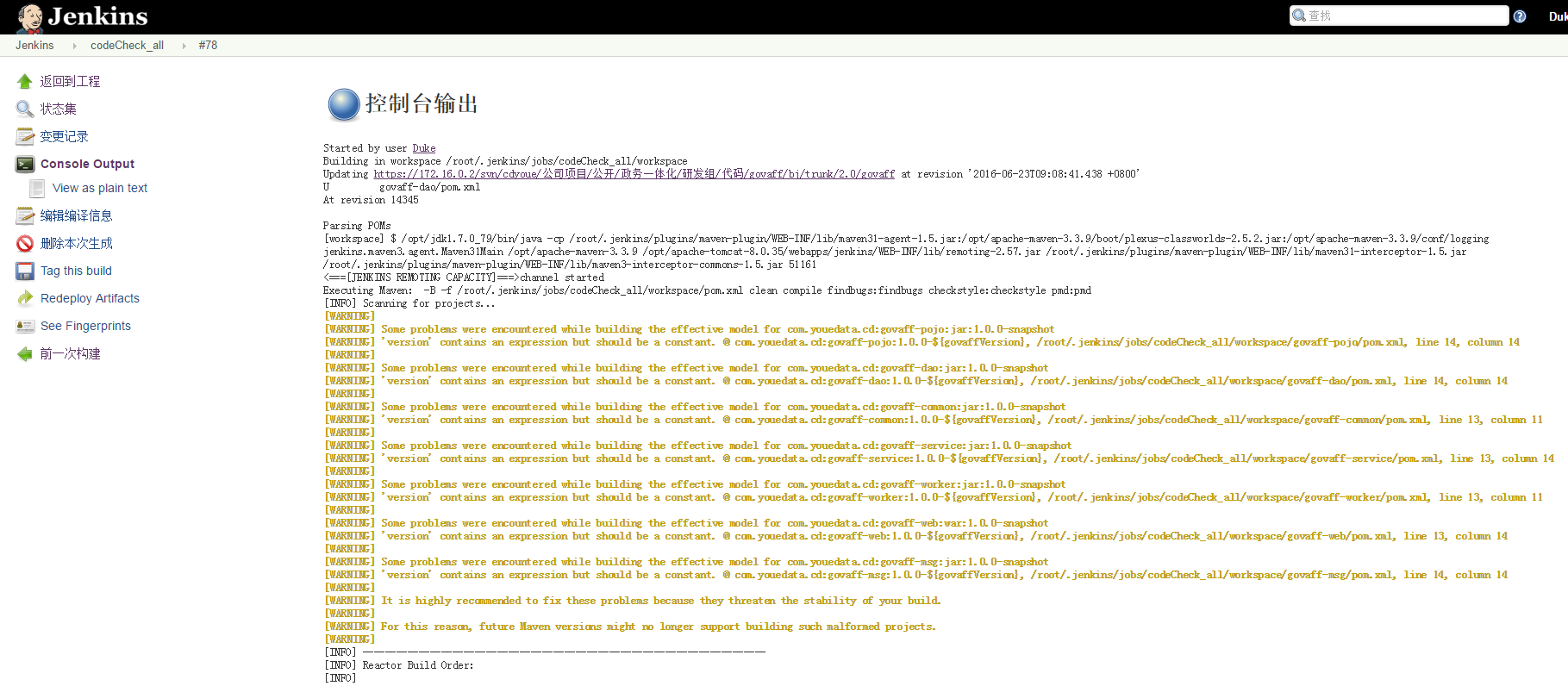
如上图所示，可在首页的左下方看到构建队列及状态



如上图所示，进入任务后可看到改任务的构建历史，点击刚刚构建的任务（#78）



如上图所示，可看到该任务的详情，点击Console Output还可以看到任务构建的日志。



## 附录

## Docker 镜像层

