**Jenkins**

| Attribute | Value |
| --- | --- |
| Owner | Gaurav Jethawa (430707) |
| Contact Information | [GJ00437070@techmahindra.com](mailto:GJ00437070@techmahindra.com) |

## 

Revision History

| Author | Date | Version # | Revision Description |
| --- | --- | --- | --- |
| Gaurav Jethawa | 31/03/2016 | 1.0 | Initial Draft Version  Jenkins Continuous Integration |

**Table of Contents**

[Continuous Integration 3](#_Toc450916418)

[Jenkins 3](#_Toc450916419)

[Installation 3](#_Toc450916420)

[Integrations 5](#_Toc450916421)

[Jenkins Maven Integration 5](#_Toc450916422)

[Jenkins Gradle Integration 10](#_Toc450916423)

[Jenkins SonarQube Integration 12](#_Toc450916424)

[Jenkins Git Integration 12](#_Toc450916425)

[Jenkins SVN Integration 12](#_Toc450916426)

[Jenkins MS Build (.Net Project) Integration 14](#_Toc450916427)

[Jenkins Email Integration 16](#_Toc450916428)

[Troubleshooting 20](#_Toc450916429)

[References 20](#_Toc450916430)

# Continuous Integration

* "Continuous integration" refers to a process that builds and tests code on a frequent basis.
* The continuous integration servers constantly monitor source code repositories and as soon as new changes/commits are detected, they initiate a new build cycle. Each check-in gets verified by an automated build. Hence problems or errors are detected in early stages.
* The build cycle actually involves code compilation and, in addition, may involve various tests and code analysis
* Because you’re integrating so frequently, there is significantly less back-tracking to discover where things went wrong, so you can spend more time building features.
* Catch issues fast and nip them in the bud.

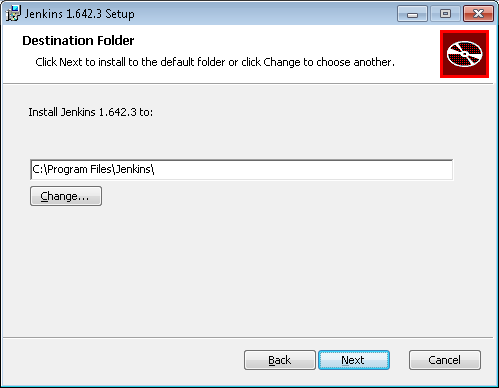
# Jenkins

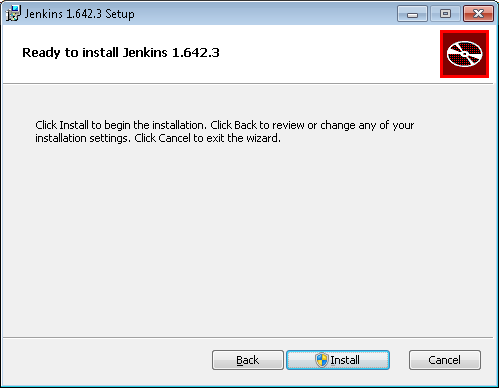
Jenkins is an open source, cross-platform, continuous integration and continuous delivery tool written in Java, that increases your productivity. Use Jenkins to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by providing powerful ways to define your build pipelines and integrating with a large number of testing and deployment technologies. It is a server-based system running in a servlet container such as Apache Tomcat. Jenkins will be installed on a server where the central build will take place. It supports SCM tools including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase and RTC, and can execute Apache Ant and Apache Maven based projects as well as arbitrary shell scripts and Windows batch commands

# Installation

Step1) Download the latest LTS release from <https://jenkins.io/index.html>

Step2) Follow the basic installer steps





Step3) Once setup is done navigate to the path where Jenkins is installed and run below command:

**java -jar jenkins.war --httpPort=9090**

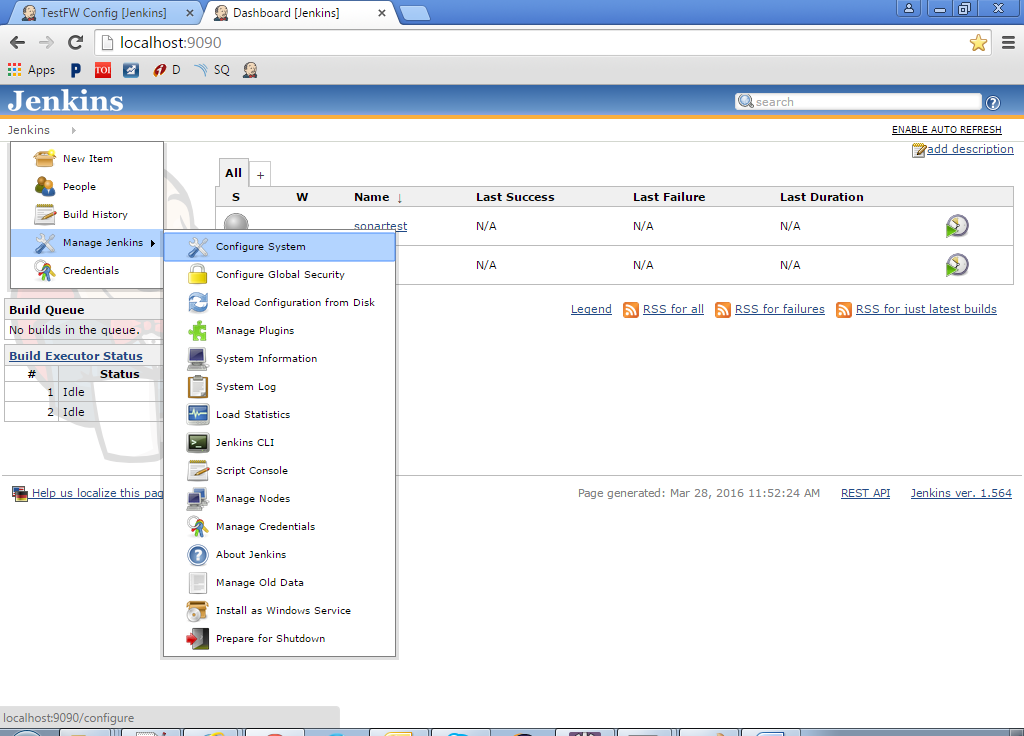
httpPort is optional and the default port will be 8080.

Step4) Access Jenkins dashboard by hitting: <http://localhost:9090>

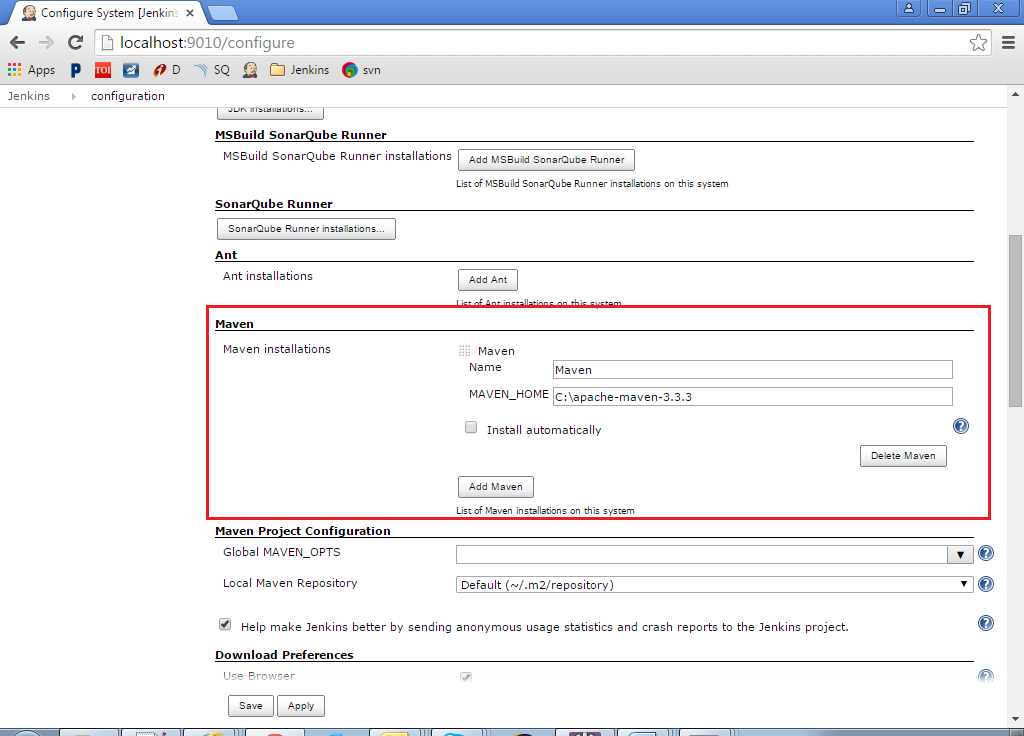
# Integrations

## Jenkins Maven Integration

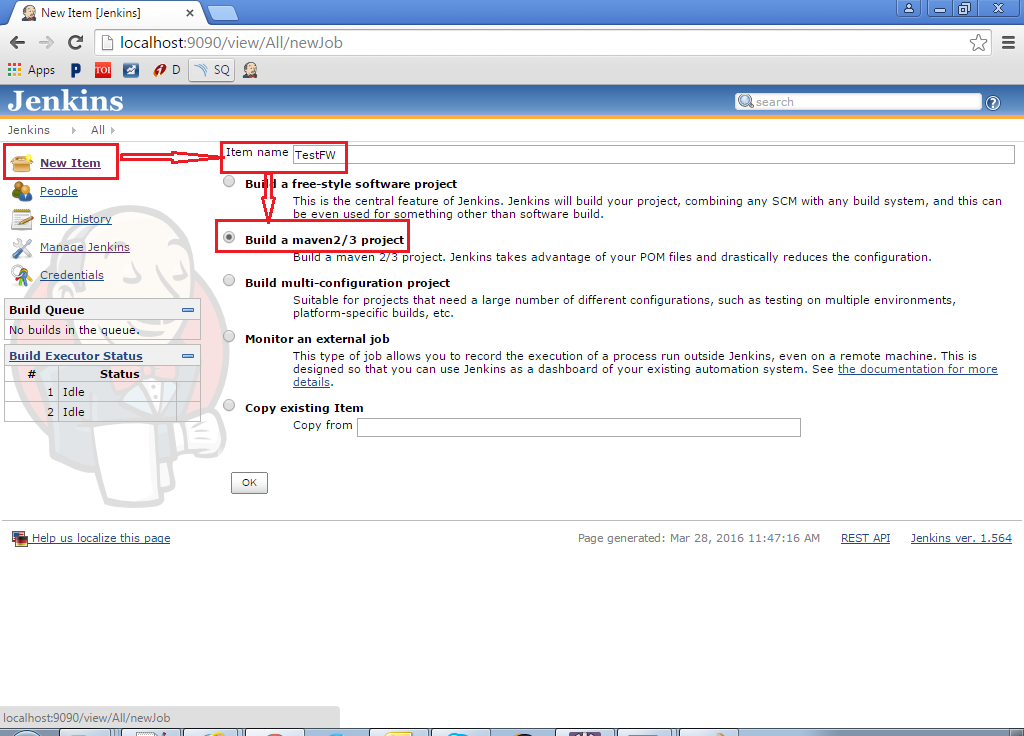
Step1) Go to Jenkins Dashboard 🡪Manage Jenkins 🡪 Configure System



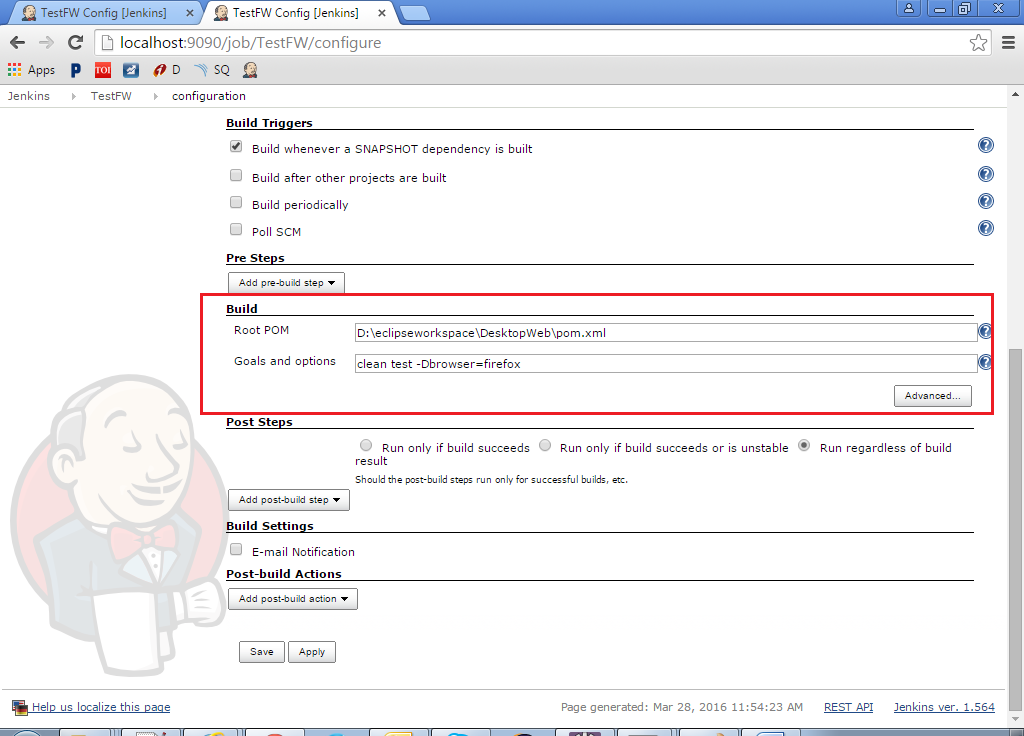
Step2) Update Maven home



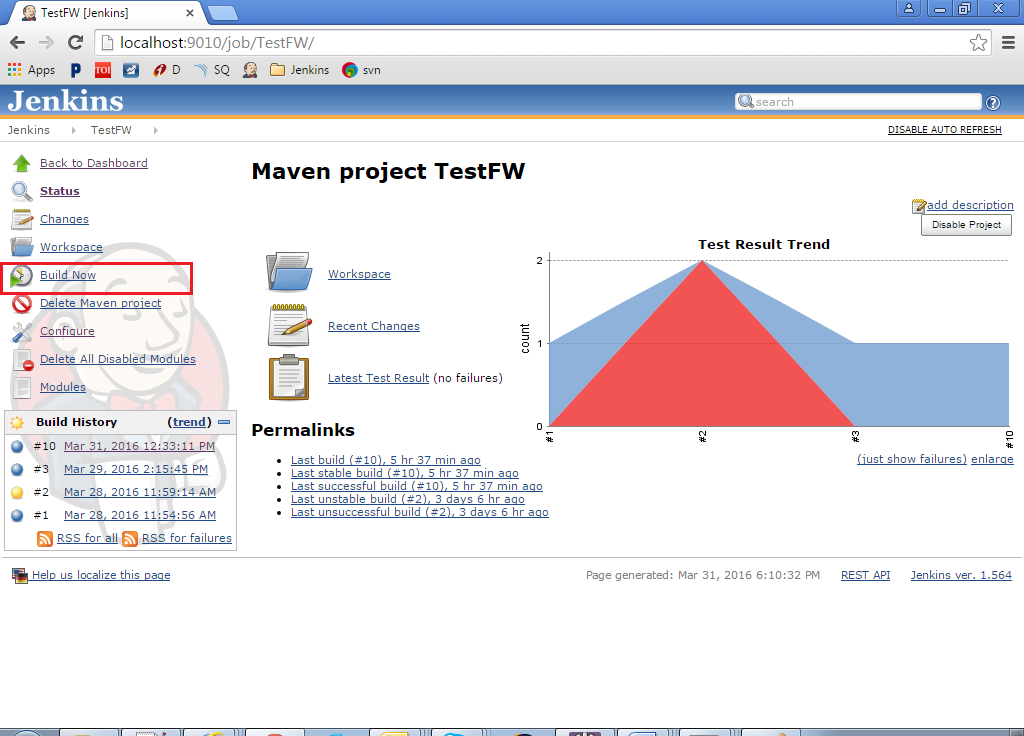
Step3) Create new item 🡪 Maven project



Step4) Specify maven pom.xml file path and goals



Step5) Click Build now



Step6) Click on the scheduled build and check console output.



## Jenkins Gradle Integration

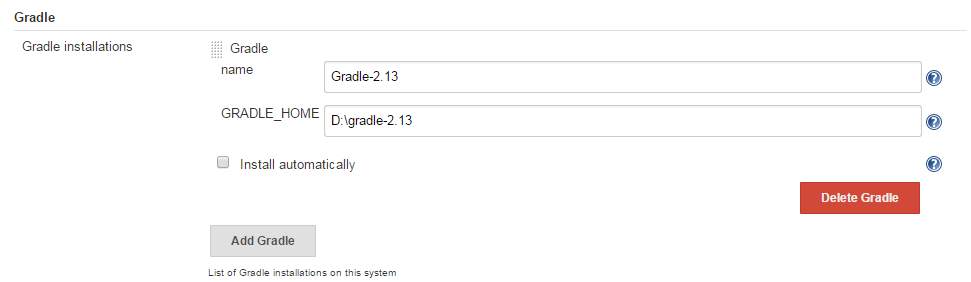
Step1) Download the gradle.hpi file from

<https://wiki.jenkins-ci.org/display/JENKINS/Gradle+Plugin>

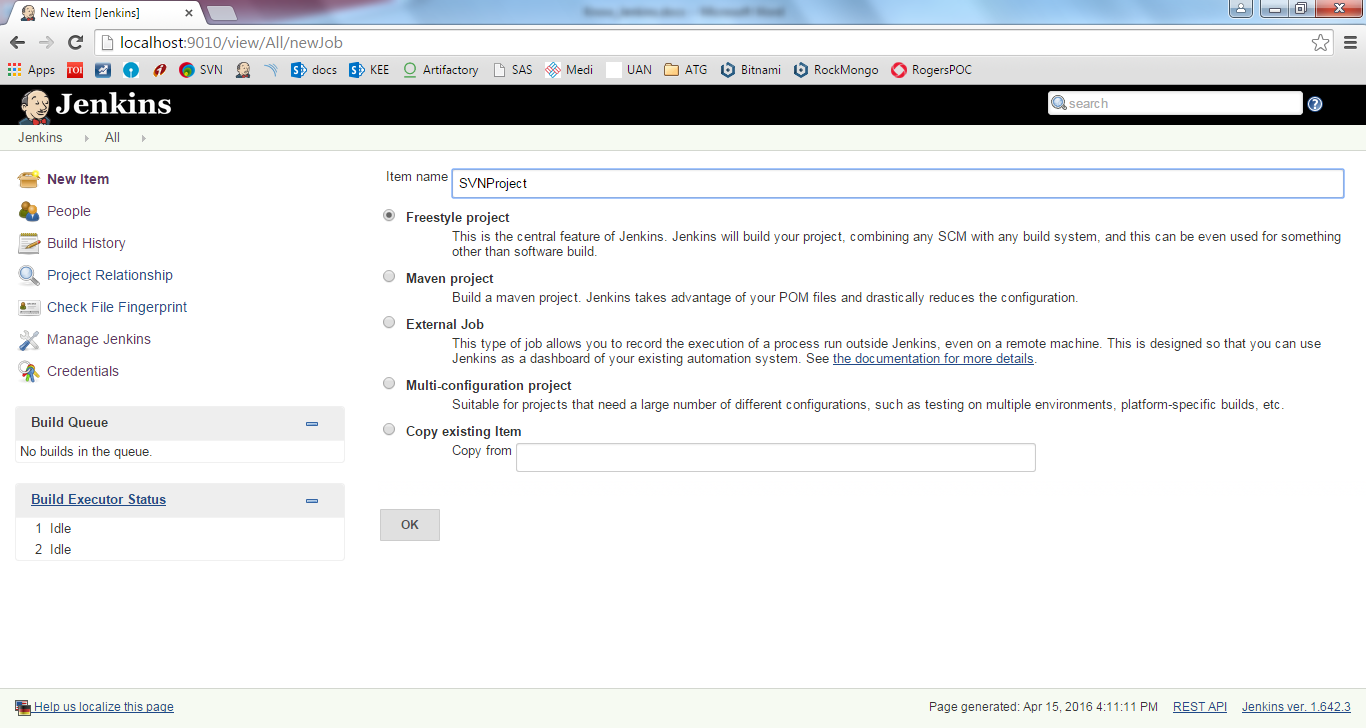
Step2) Go to Jenkins 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Advanced tab.

From Upload Plugin section, upload the hpi file.

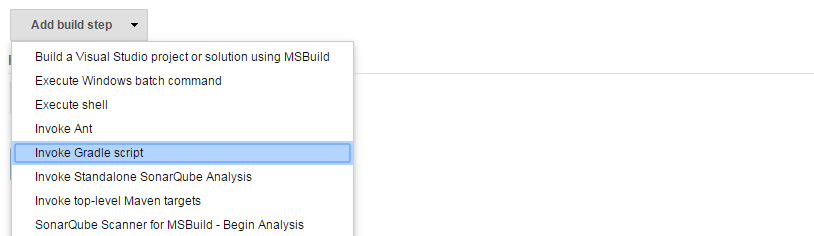
Step3) Go to Manage Jenkins 🡪 Configure System and provide gradle Name and home path



Step4) Create a New Item - Freestyle project



Step5) In Build Section – click Add build step 🡪 Invoke Gradle script



Step6) Select Gradle version, provide gradle Tasks (test, build, etc.) and provide Root build script path (path where build.gradle file is located. Mostly the root of project.) Build File is required if the build file name is different from build.gradle

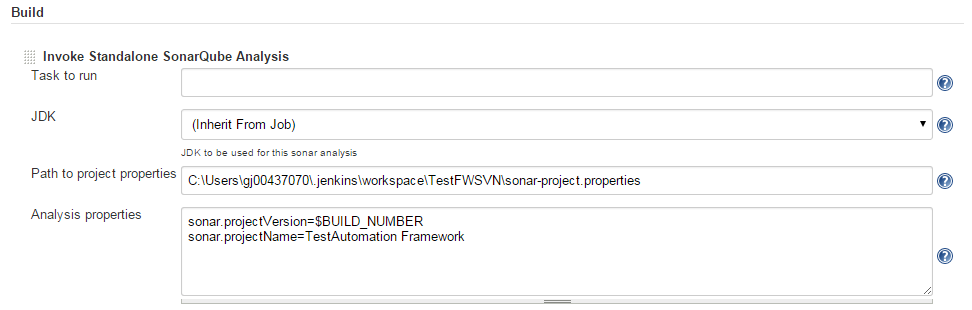


Step7) Click Build Now

## Jenkins SonarQube Integration

Refer Know\_SonarQube document for Jenkins integration

For Passing Jenkins build number to SonarQube version:

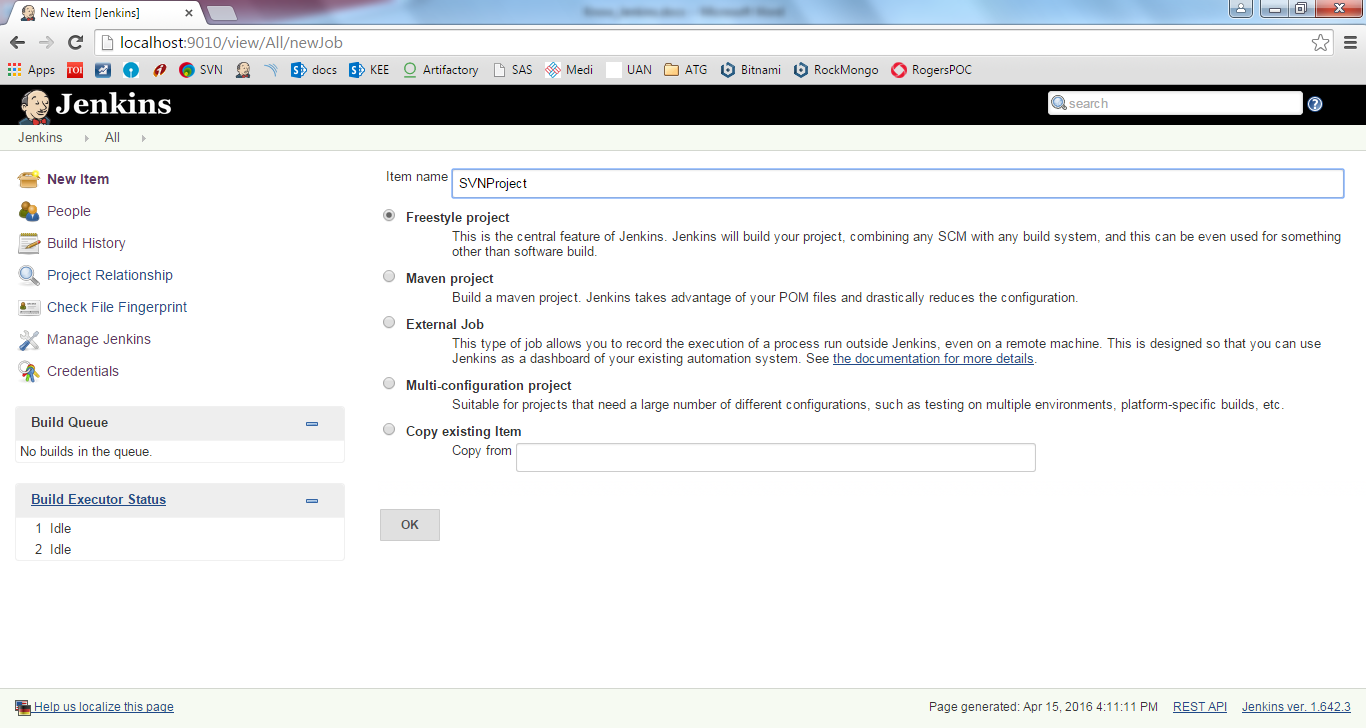


## Jenkins Git Integration

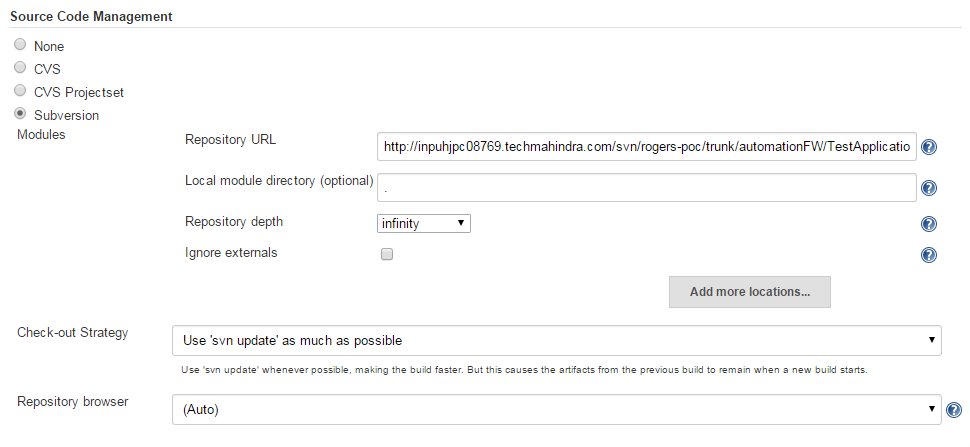
## Jenkins SVN Integration

Create a new job in following manner:

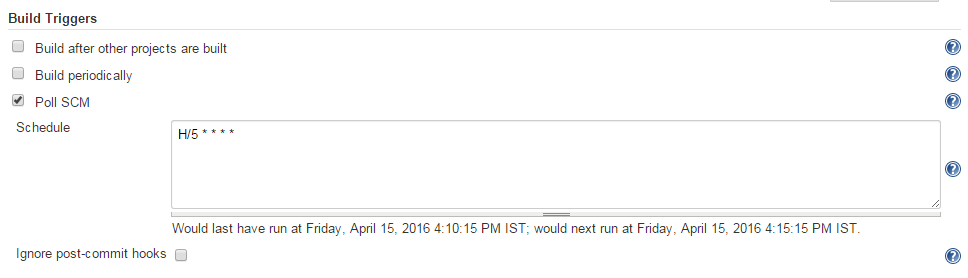
Step1) Create a Freestyle project



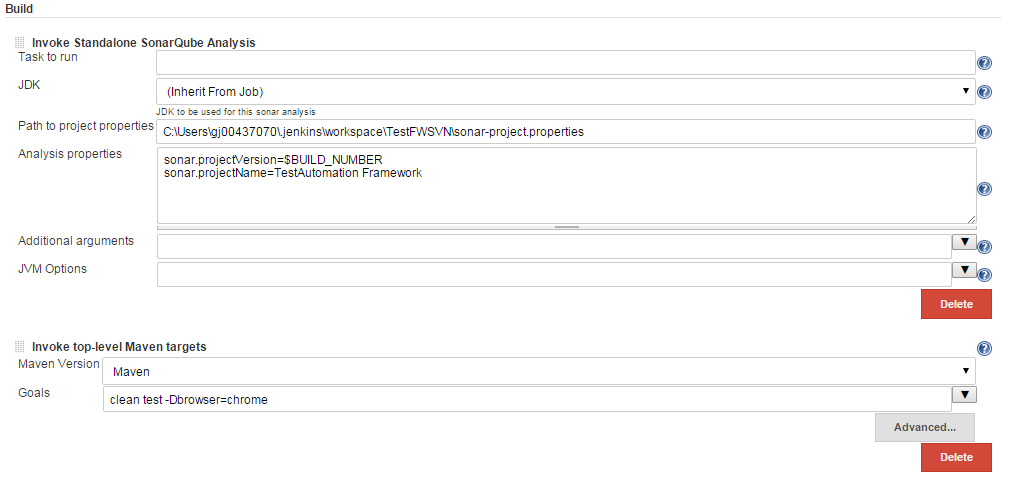
Step2) Go to Source Code Management section and click subversion. Provide the svn repository url



Step3) Go to Build Triggers section and check Poll SCM and give Schedule e.g. H/5 \* \* \* \*: This will check for SVN update every 5 minutes and will trigger the build if update is found.



Step4) In Build section, provide Maven targets (if it is a Maven project), or do SonarQube analysis, etc.



## Jenkins MS Build (.Net Project) Integration

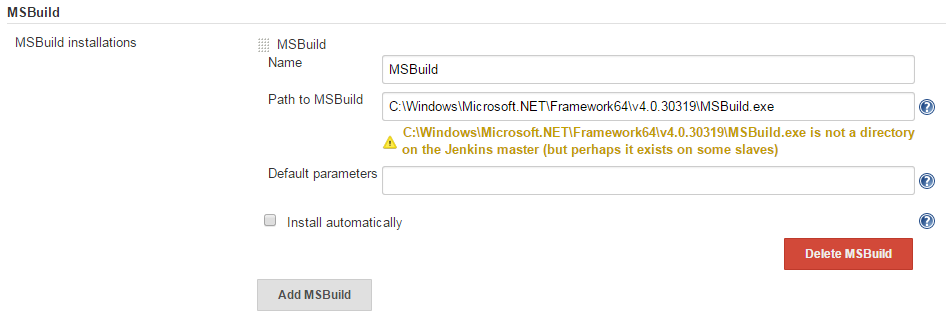
Step1) Download the MSBuildPlugin.hpi file from

<https://wiki.jenkins-ci.org/display/JENKINS/MSBuild+Plugin>

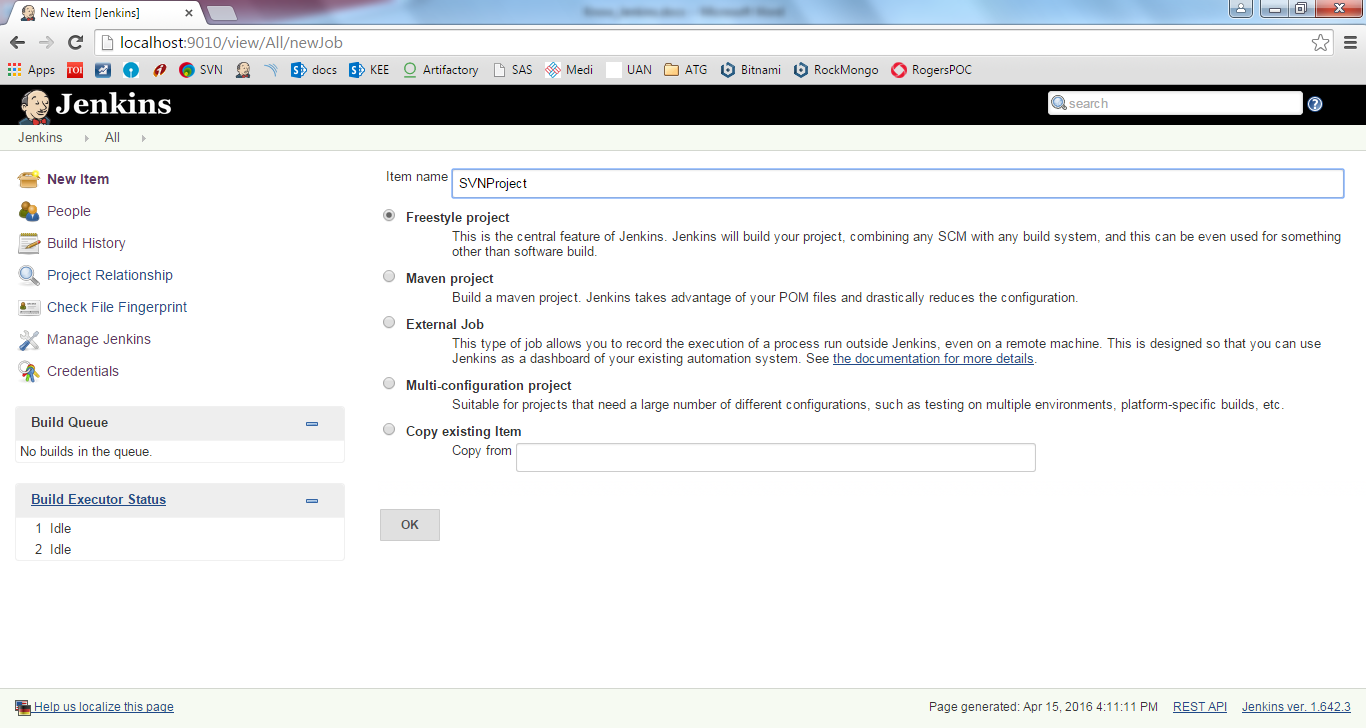
Step2) Go to Jenkins 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Advanced tab.

From Upload Plugin section, upload the hpi file.

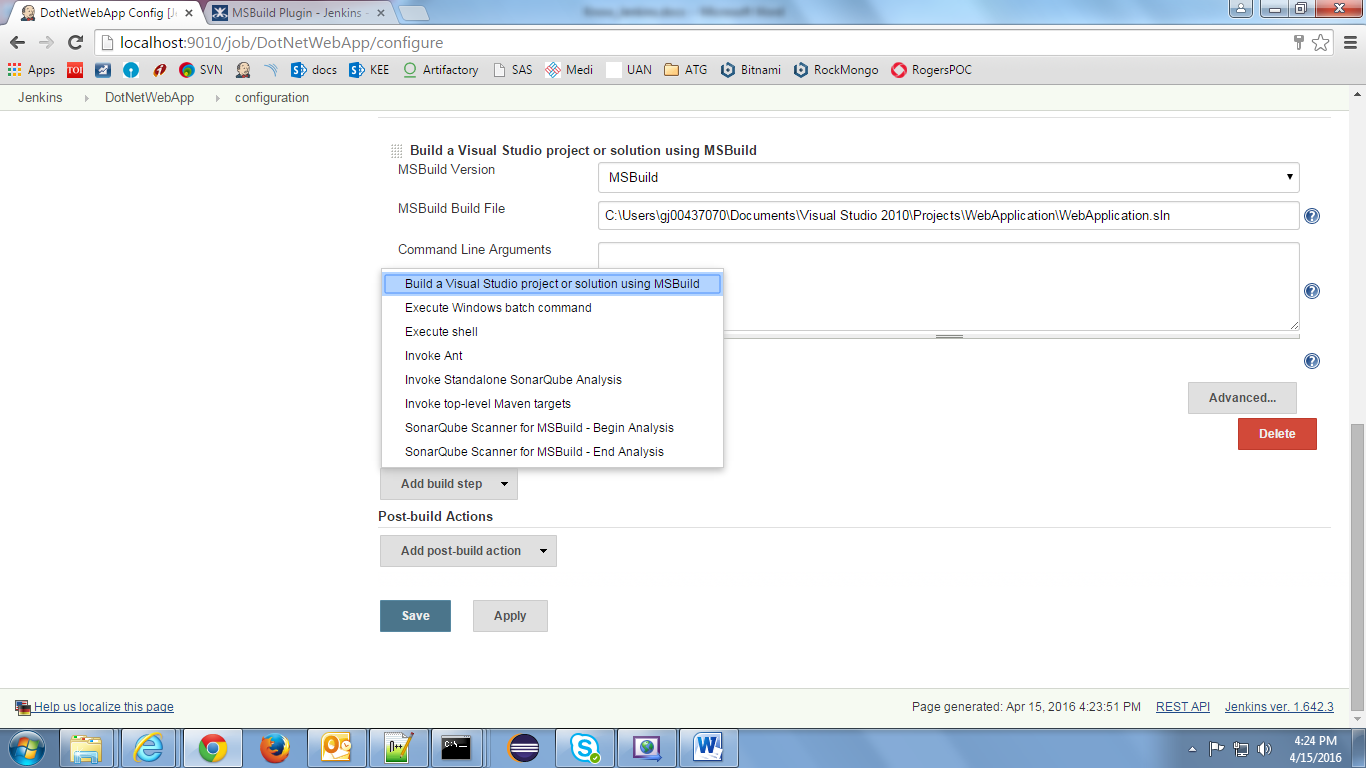
Step3) Go to Manage Jenkins 🡪 Configure System and provide MSBuild Name and path (MSBuild.exe will be located @ C:\Windows\Microsoft.NET\Framework\v4.X)



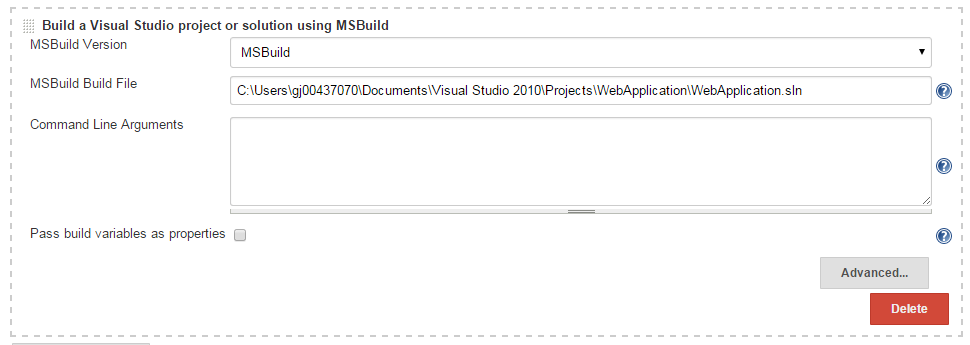
Step4) Create a New Item - Freestyle project



Step5) In Build Section – click Add build step 🡪 Build a visual studio project or solution using MSBuild



Step6) Provide MSBuild version and .sln file path



Step7) Click Build Now

## Jenkins Email Integration

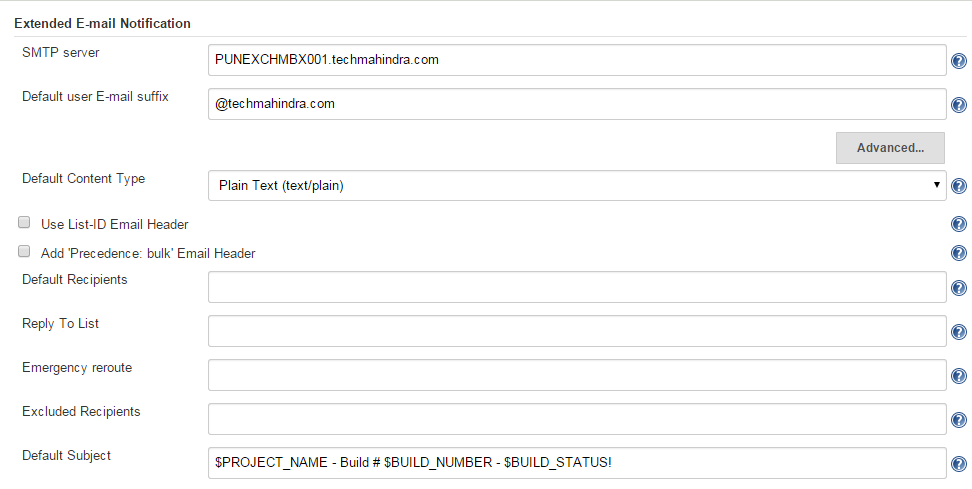
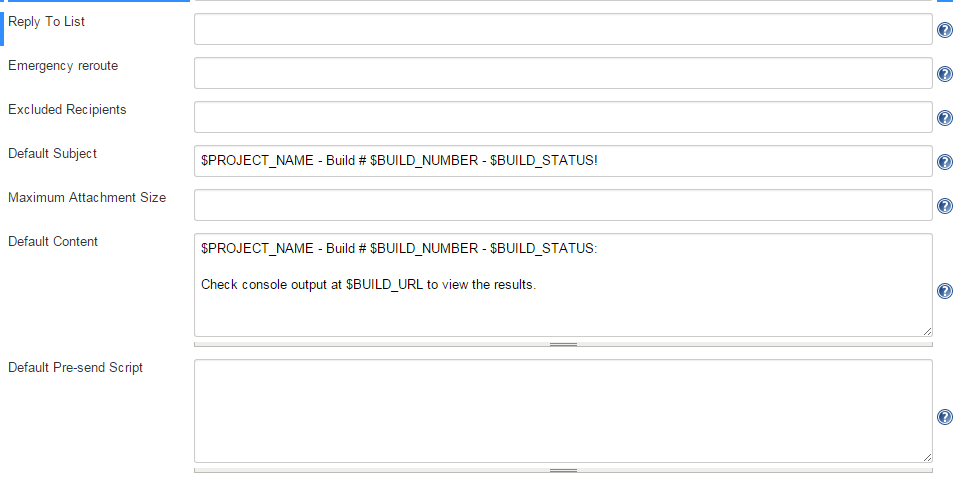
Step1) Download the email-ext.hpi file from

<https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+plugin>

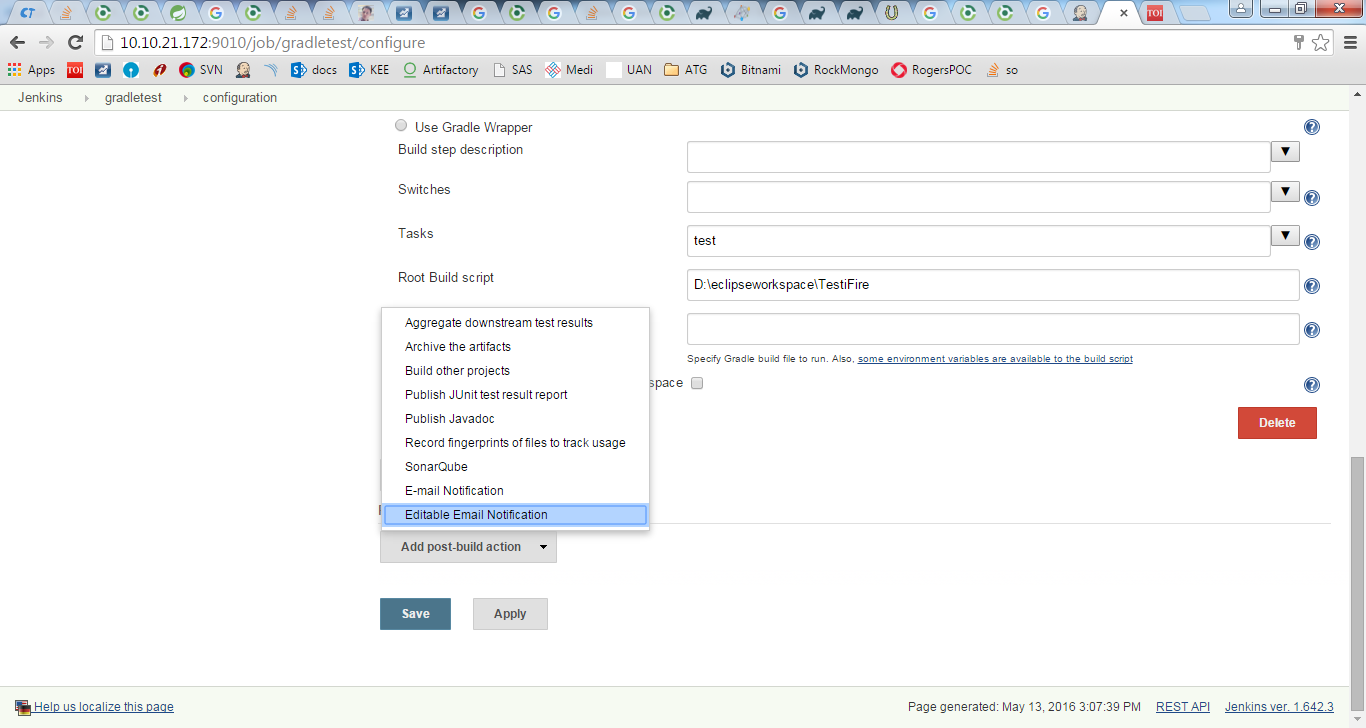
Step2) Go to Jenkins 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Advanced tab.

From Upload Plugin section, upload the hpi file.

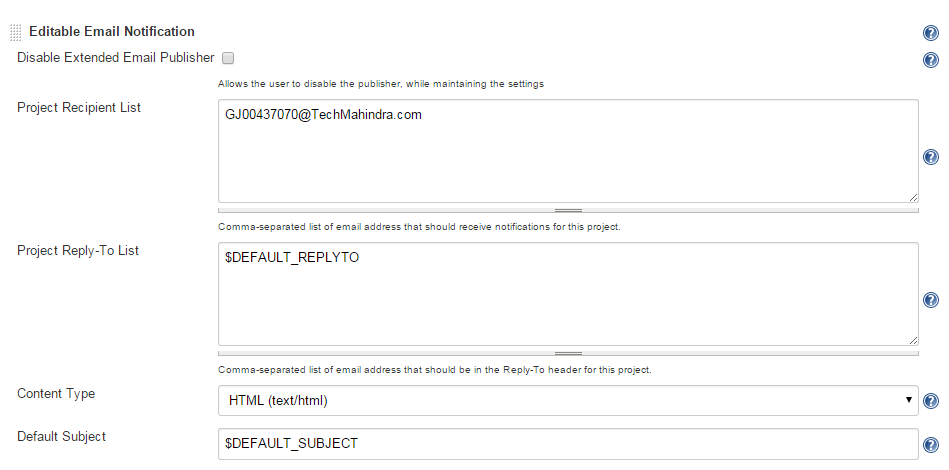
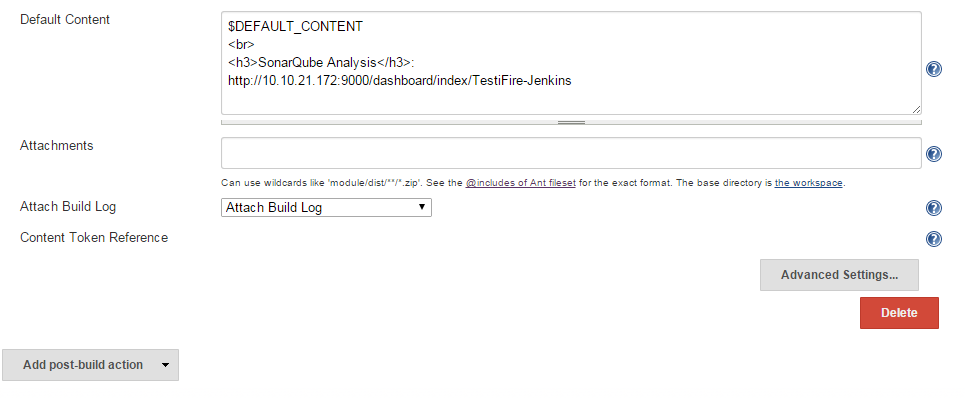
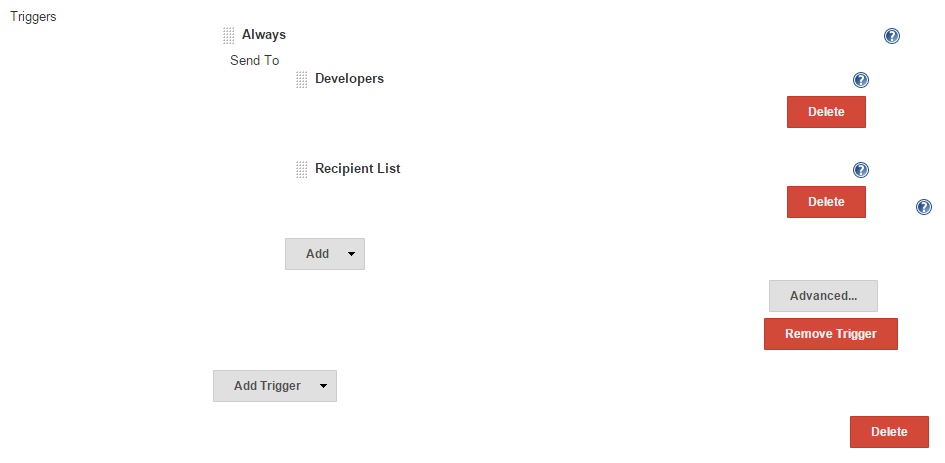
Step3) Go to Manage Jenkins 🡪 Configure System and provide SMTP server details and rest of optional details like recipients list, default subject, mail content, etc

Step4) Select any Jenkins job and go to Post-Build Actions – click Add post-build action 🡪 Editable email notification



Step5) Provide Project Recipient List, reply list, subject, content (if not provided, will be picked from system level i.e. Configure System). Click Advance and select ‘Always’ trigger so that mails will get sent for all build states. (By default email will be triggered only if build has failed or is unstable). Click save and trigger the build.

# Troubleshooting

# References

<http://www.tutorialspoint.com/jenkins/jenkins_overview.htm>