Password - fd3364619a68427482a7d8509e68bb24 ; also present in C:\Program Files\Jenkins\jenkins.err

Username – admin

Password – admin

Full Name – George John Chavady

Email – [georgejohn92@gmail.com](mailto:georgejohn92@gmail.com)

CI/CD pipeline – Automatically building the code in several stages and at each stage, testing and promoting it to the next stage is called a pipeline.

Plugin management – Manage Jenkins 🡪 Manage Plugins 🡪 Installed Plugins

You can make your environments even more trustworthy in the following ways:

* Follow a code review process where at least one other team member must approve a pull request
* Configure build and unit test enforcement on all pull requests, so it is impossible to merge code that would “fail” (whatever that means for your team/application)
* Establish branch protection in your Git repository so users cannot accidentally (or intentionally) push code directly to environment-related branches in the team repository, thus circumventing the review process
* Set up a deployment hook, so that a Jenkins build job is automatically triggered when code is committed (or merged in) to the corresponding branch. This may make sense for the develop branch!
* Be cautious about who has access to configure Jenkins jobs; I recommend two developers only. One person is too few due to the Bus Factor, and more than two unnecessarily increases the likelihood of a job being changed without the appropriate communication or consensus.
* Display the version of the code in the application somewhere, such as the footer or in the “about” menu. (Or, put it in an Easter Egg if you don’t want it visible to users.) The way you obtain the version, specifically, will depend greatly on the language of your app and the platform you use to run it.

**Jenkins Pipeline**

Build tasks span multiple build slaves.

* This project is parameterized – add parameters that can be used
* SCM – git repo and password
* Post-build tasks can be set

**Github Webhook**

Webhooks are required to be configured in the github repository. But, it cannot be configured to a local url. So we use ***socketxp.com*** to provide a public URL.

Repository 🡪 Settings 🡪 Webhooks 🡪 Add webhook

Payload URL 🡪 https://test-user-a29dfe42e3.socketxp.com/github-webhook/

**Connecting to socketxp**

* Traverse to the socketxp.exe folder
* socketxp login "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MDAyOTg0MzYsImlkIjoiZ2Vvcmdlam9obmNoYXZhZHlAZ2l0aHViLmNvbSJ9.MyB8TrwhD23cWq3JIU53A7L1v9Y3Su50RFhaSNSIF-o"
* token present in the socketxp.com 🡪 tokens

socketxp connect http://localhost:8080  
  
Connected.  
Public URL -> https://test-user-a29dfe42e3.socketxp.com

**Configure SonarQube – Code Analysis**

Install SonarQube ; runs at <http://localhost:9000>

* Open Jenkins 🡪 Manage Jenkins 🡪 add Sonar Scanner plugin and others
* Configure System 🡪 SonarQube Servers 🡪 add localhost URL
* Global Tool Config 🡪 Add Sonar Scanner: path C:\Program Files\sonarqube-8.4.1.35646\sonarqube-8.4.1.35646\lib\scanner (did not work – so install automatically)
* Build 🡪 error: .java files; add binary