Full CI/CD Pipeline

* All pieces of automation that need to interact with the source code will use SCM:
  + Continuous Integration (CI) will get the code from SCM
  + SCM will notify the CI server when code needs to be b uilt
* On a CentOS environment, you can generate an ssh key pair like this:
* ssh-keygen –t rsa –b 4096
* After generating the key pair, copy the contents of ~/.ssh/id\_rsa.pub.
* git clone <repository url>
* git init
* git status
* git add <file>
* If not, you may need to specify which remote repository and which remote branch you want push to use:
* git push –u <remote name, usually origin> <branch name>
* For example, some teams maintain a “production” branch, and merging changes into this branch initiates automated processes involved with deploying to production.
* git checkout <branch>
* You can create a new branch and check it out immediately with the –b flag: git checkout –b <new branch>
* git tag
* Tags in git are simply pointers to a particular commit. They can be used to provide a name that can be used to reference that commit in the future or tag version of software it is associated for

Pull Requests: teams work using multiple branches to manage all of their changes. At some point, these branches need to be merged together (preferably as often as possible)

|  |  |
| --- | --- |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
|  |
| git push -u origin [branch name] | Push changes to remote repository (and remember the branch) |
|  |  |

* Build Automation:
* Build automation is the automation of tasks needed in order to process and prepare source code for deployment to production. It is an important component of continuous integration.
* Compiling
* Dependency Management
* Executing Automated Tests
* Packaging the App for D ependency
* Use Gradle for Build Automation
  + Wrapper feature: which is a script that doenaloads specified Gradle version, before it is required
  + Wrapper removes the dependency of having it installed before in order to run the build
  + Automated build easy with only pre-requisite of Java

If you already have a normal Gradle install on your system, you can use it to easily install the wrapper:

* cd /your/project/root/directory
* gradle wrapper
* Run Gradle commands against your project like this:
  + cd /your/project/root/directory
  + ./gradlew build
* Gradle build is defined in build.gradle in the root directory of the project
* Gradle init
* Gradle builds consist of a set of tasks that you call from the command line:
* ./gradlew someTask someOtherTask

Defining Tasks:

* Build.gradle
  + Task myTask(
  + Println ‘Hello, World’
  + }
* Add Task Dependencies