Jenkins and Deployment Pipelines

In the previous post, I described deployment pipelines and mentioned how you need a controller to drive software releases through each stage of the pipeline. This component is typically a Jenkins CI/CD server, although Jenkins’ built-in capabilities to capture pipelines of deployment activity are rather limited. There is an open source [Build Pipeline](https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin) plugin that gets the job done, for the most part. Other commercial tools in this area include: Atlassian Bamboo, ThoughtWorks Go, and XebiaLabs XLRelease.

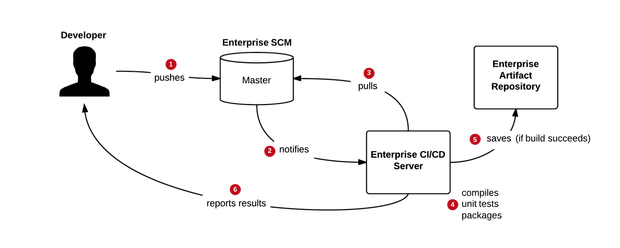
This is a screenshot from the Jenkins Build Pipeline plugin that shows various pipelines in progress.

[](https://trevormquinn.files.wordpress.com/2014/12/bp.png)

It helpfully color codes the status of each stage of each pipeline, and, combined with Jenkins itself, allows you to do role-based access to control how and when jobs are launched and restarted. Most importantly, it allows you to capture a flow of build jobs, from stage to stage. Stages can be configured to automatically begin based on successful completion of the stage preceding it or they can require manual push button approval to proceed.

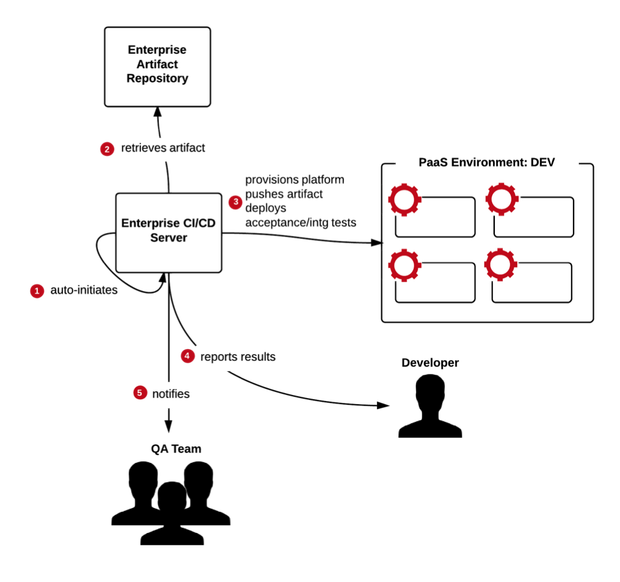
The Build Pipeline tool enables us to capture each of the deployment stages shown below:

**Commit Stage**

[](https://trevormquinn.files.wordpress.com/2014/12/commit.png)

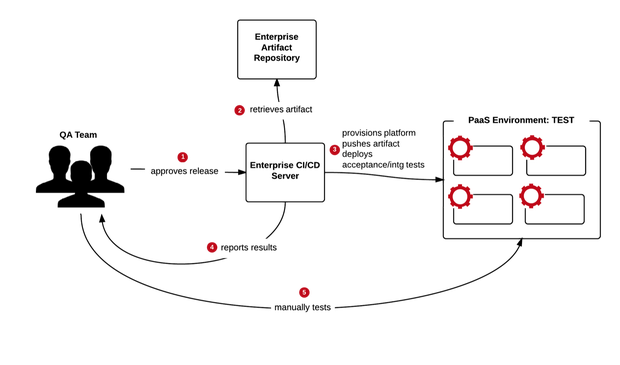
The Commit Stage is a continuous integration step, used to validate a particular code commit from a compilation, unit test, and code quality standpoint. It does not involve any deployment to an environment, but does include export of a compiled binary (in the case of a Java project) to an enterprise artifact repository such as Nexus.

**Acceptance Stage**

[](https://trevormquinn.files.wordpress.com/2014/12/acceptance.png)

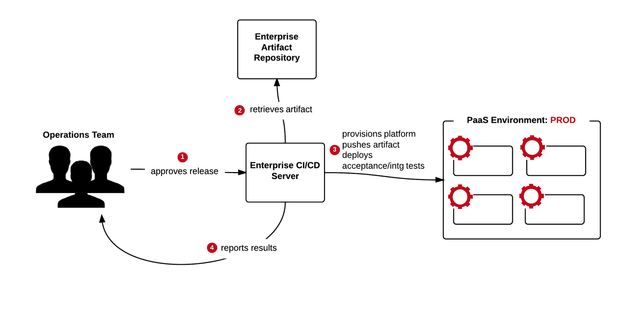
The Acceptance stage is where automated functional acceptance and integration tests are performed. To do this, Jenkins will deploy the code to an environment. PaaS makes this trivial, as I’ll shown in a later post.

**UAT Stage**

[](https://trevormquinn.files.wordpress.com/2014/12/uat.png)

In this deployment pipeline example, a separate UAT stage is available for manual user testing, important for confirming application usability. Advancement to the UAT Stage must be approved by the QA team, who log in to Jenkins to perform a push button deployment.

**Production Stage**

[](https://trevormquinn.files.wordpress.com/2014/12/production.png)

Release to production is the final stage, controlled by the Operations team. Like the two stages before it, it also involves CI/CD server deployment to the PaaS environment.

So that captures the deployment pipeline’s integration with the Jenkins CI/CD server. The next post will profile *how* the CI/CD server leverages PaaS to make both environment creation, configuration, and application deployment an easy process.