Setting up Continuous-Delivery Pipeline using Jenkins' Delivery-Pipeline-View Plugin

Jenkins is a widely used CI tool which helps DevOps teams in automating the multiple tasks. The use of Jenkins has widely increased over a period of time due to a rich set of functionalities which it provides in the form of plugins. Jenkins has plugins for automating almost everything at the infrastructure level. In this blog, I am going to explain the use of "Delivery Pipeline View" plugin of Jenkins by which we can setup complete continuous delivery pipeline.

Continuous-Delivery is the process of automating the code delivery tasks from development to production environment consisting of version control management, deployment on each environment followed by automated testing and taking decisions based on the results of the previous step. If the test cases are passed, the code will be automatically moved to the subsequent environment without any manual intervention till its deployed on production environment followed by post-deployment tasks.

In Jenkins, each job has a post-build step in which we can specify the downstream projects on the basis of the result of the current step. I've made a following flow of the build pipeline for demonstration purpose:

1. Dev-Deployment
2. Dev-Testing
3. UAT-Deployment
4. UAT-Testing
5. Prod-Deployment
6. Prod-Testing

Note: Each step is an independent Jenkins job in itself. If at a certain point, any of my steps fails, the pipeline will get stopped. Each step will be started only if the previous step has run successfully.

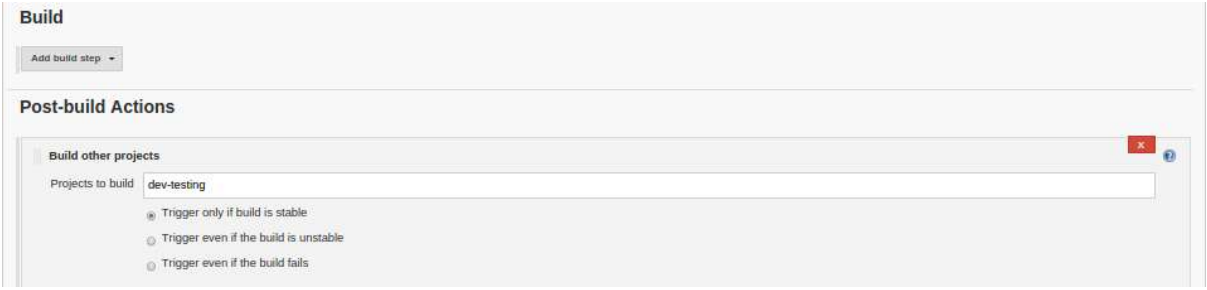
Deployment Scenario

In deployment case, I will be deploying an index page on Nginx from the git branch of the same environment of my git repository. For, e.g., for dev.example.com, I will be deploying the index page from the dev branch of my git repository. It contains the text that "This is dev site."

Testing Scenario

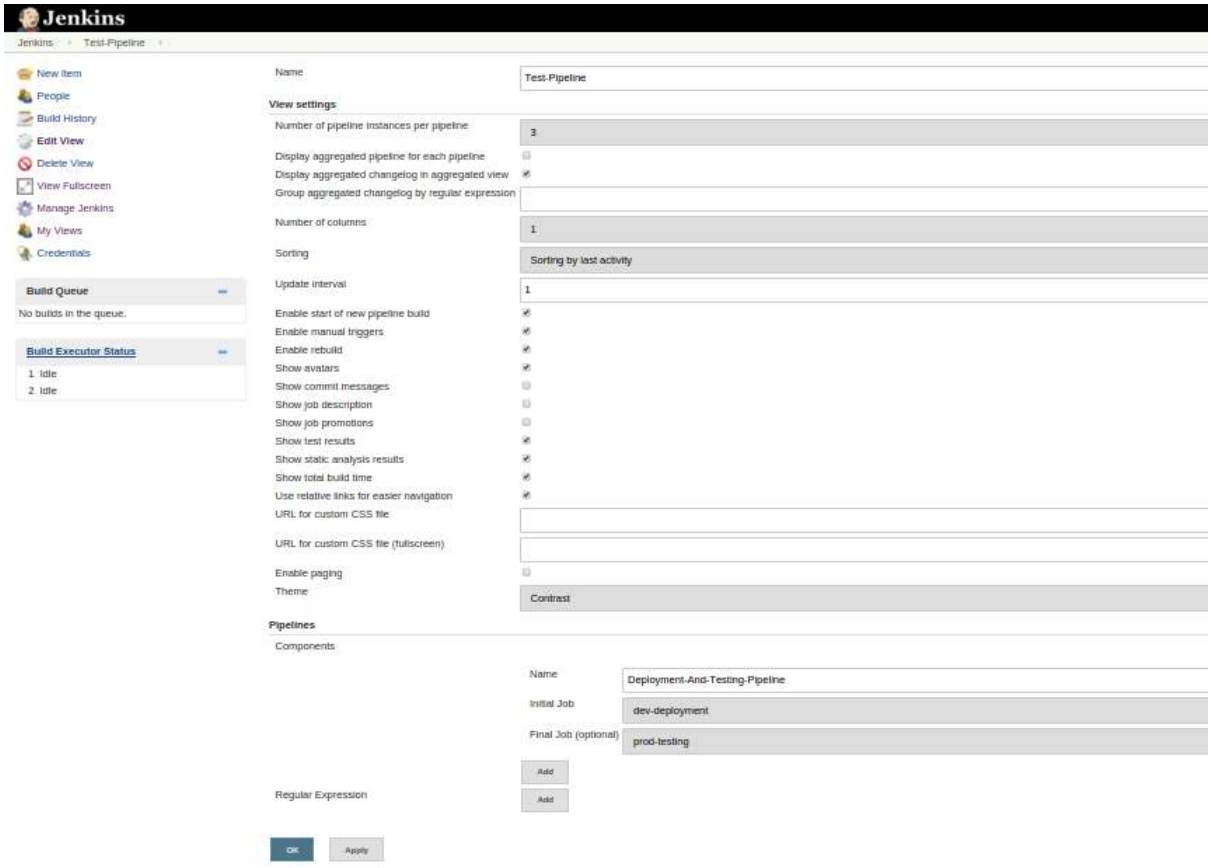
In my testing scenario, I am just checking whether the deployed index page contains the environment name. For e.g., "This is dev site" contains the word "dev" in it. Means the test case is passed. The same process is repeated for all the environments.You can define the test cases as per the requirements.

Also, in each job, I've specified what are the upstream and downstream projects to build for the specific job. For eg. dev-testing has a downstream project for uat-deployment and upstream project for dev-deployment. Each downstream project will be run if its upstream project has run successfully. The below snapshot is the configuration of dev-deployment job:

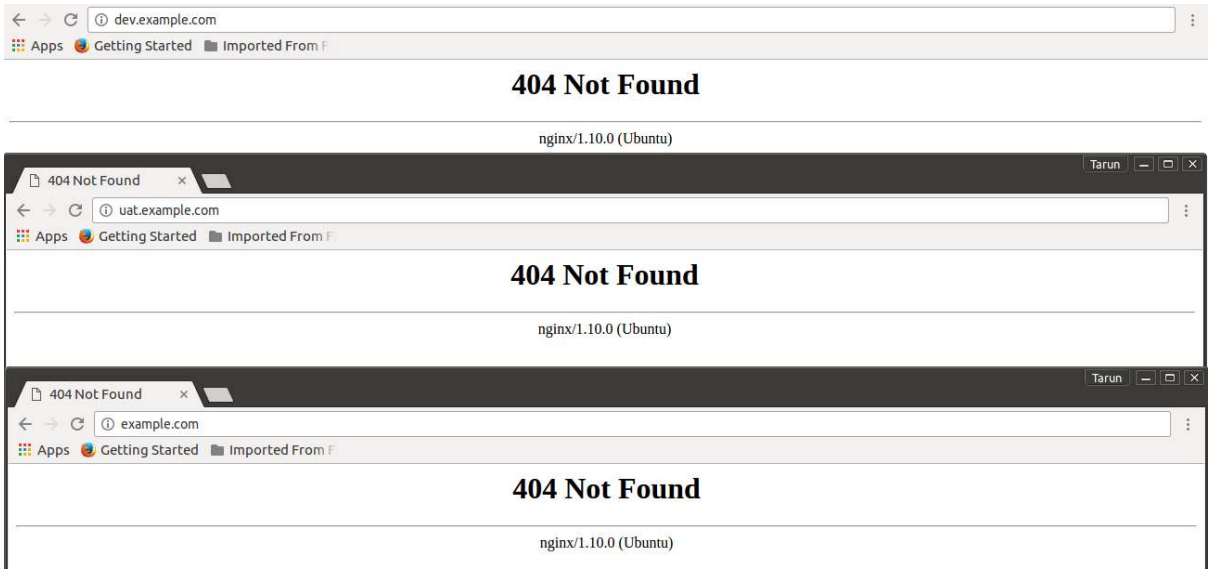


For the view of the delivery pipeline, the required plugin is **Delivery Pipeline Plugin.** Please make sure the plugin is installed:

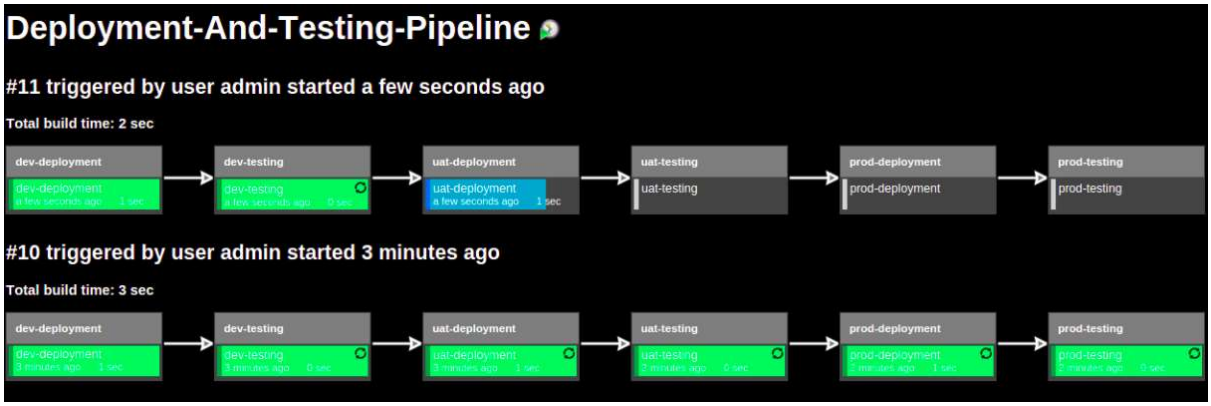
Below is my pipeline configuration:
In the pipeline configuration, we can define the endpoint jobs of the pipeline like in my case the initial job is dev-deployment and final job is prod-testing:



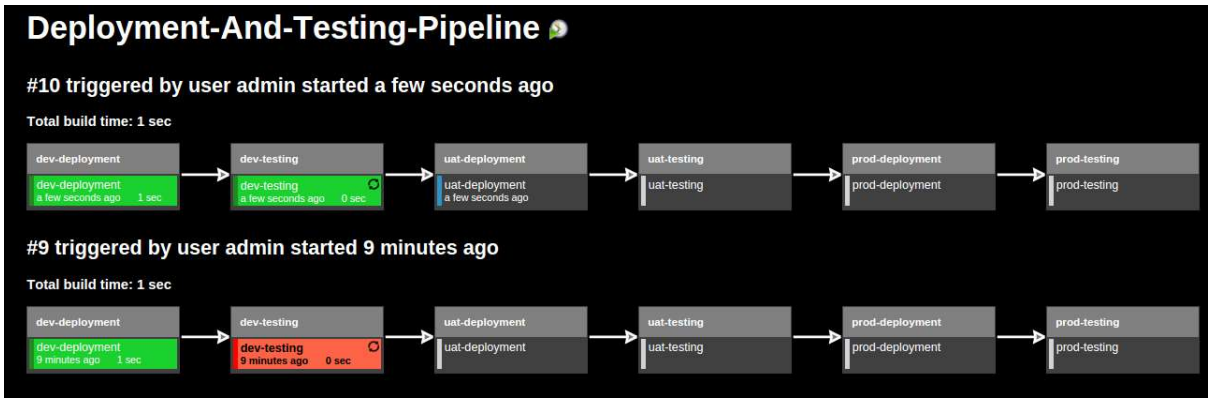
Before Executing the Pipeline: The status of my 3 sites (dev.example.com, uat.example.com, example.com) is as follows:



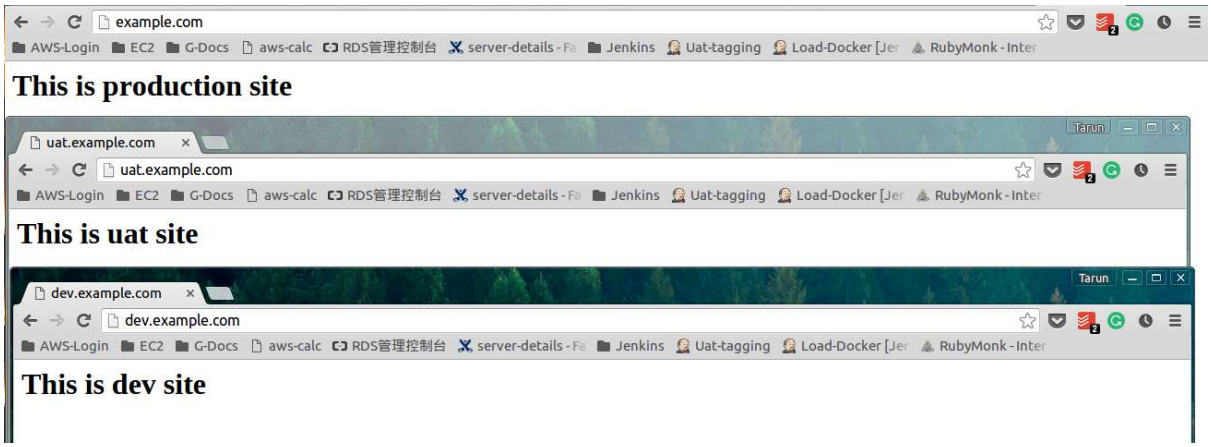
Executing the Pipeline: Click on the build icon to start building the pipeline process:



Note: If something gets break during the pipeline, the build pipeline gets aborted. For eg. in Image7, my build 9 is aborted at dev-testing due to failure of the test cases:



After executing build-pipeline successfully:



The code has been successfully deployed on all the environments after proper testing.
So, Jenkins is a fantastic tool to automate all the DevOps process