**Problem Statement**

You have been asked to:

Create a pipeline in Jenkins. Once push is made to “develop” branch in git, trigger job “test”. This will copy git files to test node. If test job is successful, then prod job should be triggered. Prod jobs should copy files to prod node.

**Solution Approach**

**Step 1: Create ec2 instances**

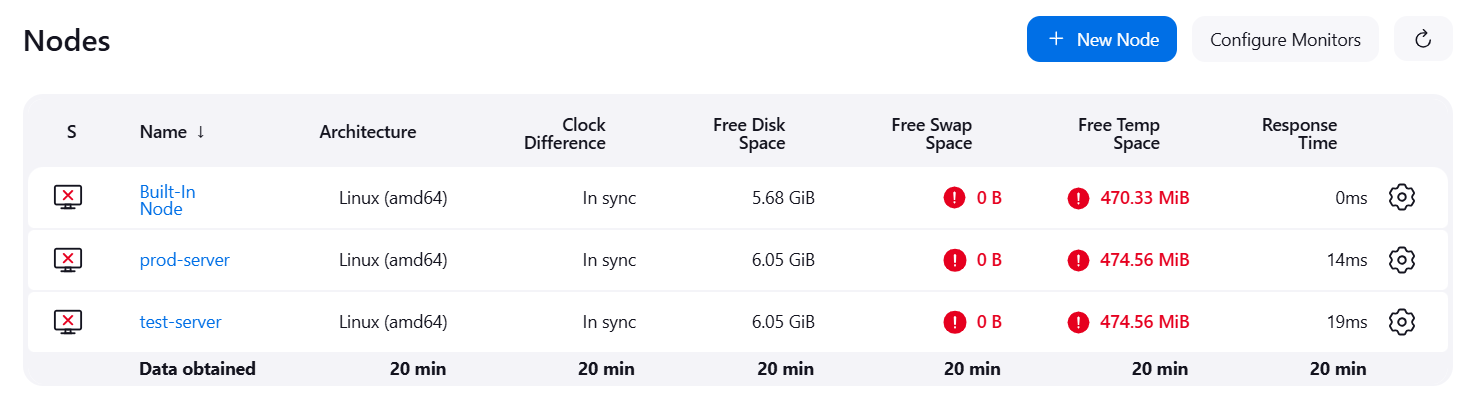
* We will create 3 ec2 instance: Master, Slave 1 (test server) and Slave 2 (prod server)
  + Master instance is where we download and host Jenkins
  + Slave 1 is where we will copy the file when a push is made to test branch
  + Slave 2 is where we will copy the file when a push is made to prod branch

**Step 2: Set Up a Git Repository**

* We used the Git Repository for Zendrix Softwares that was set up as part of the git assignment.
* The git repository was set up with test and main branches that would be used to trigger builds
* Link to repository: <https://github.com/vyombhatt/zendriix-softwares-workflow>
* Along with this, we will be using the same webhooks that were set up as part of the precious assignment

**Step 3: Set up the 2 Nodes on Jenkins**

1. Setting up the *test-server* node:
   * Name: *test-server*
   * Number of executors: *1*
   * Remote root directory: */home/ec2-user/jenkins/*
   * Labels: *test*
   * Usage: *Use this node as much as possible*
   * Launch Method: *Launch agents via SSH*
     + Host: *172.31.11.227 (Jenkins slave 1 private IP)*
     + Credentials: *ec2-user*
     + Host key verification strategy: *Non verifying*
   * Availability: *Keep this agent online as much as possible*
2. Setting up the *prod-server* node:
   * Name: *prod-server*
   * Number of executors: *1*
   * Remote root directory: */home/ec2-user/jenkins/*
   * Labels: *test*
   * Usage: *Use this node as much as possible*
   * Launch Method: *Launch agents via SSH*
     1. Host: *172.31.5.154 (Jenkins slave 2 private IP)*
     2. Credentials: *ec2-user*
     3. Host key verification strategy: *Non verifying*
   * Availability: *Keep this agent online as much as possible*

****

**Step 4: Set up the 2 Jobs**

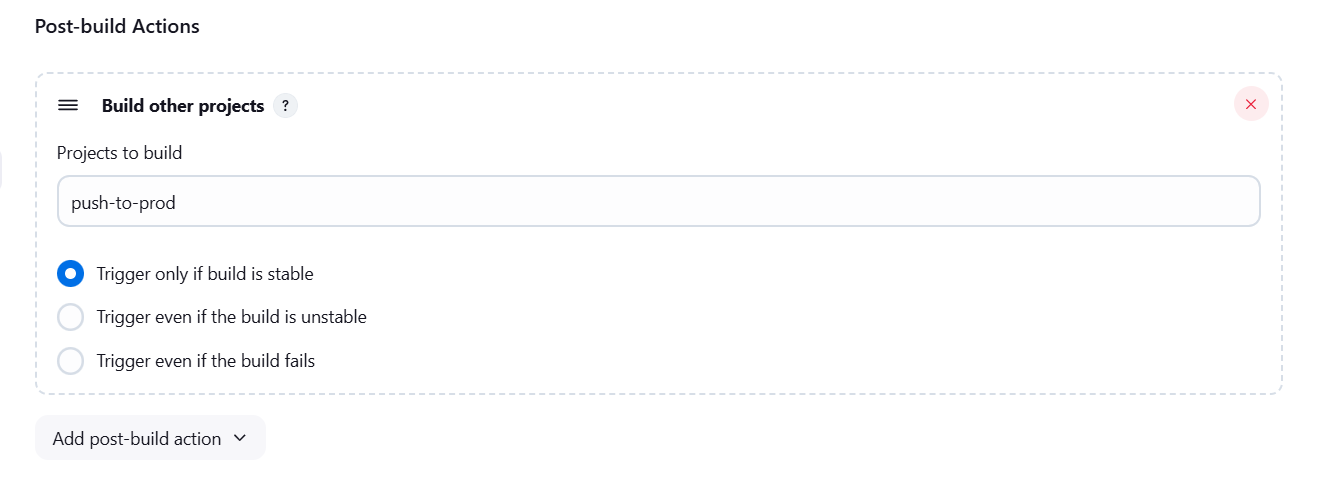
1. Setting up the *push-to-test* job:
   * Create new item as freestyle project: *push-to-test*
   * Description: *trigger when a push is made to test branch*
   * GitHub project: [*https://github.com/vyombhatt/zendriix-softwares-workflow.git/*](https://github.com/vyombhatt/zendriix-softwares-workflow.git/)
   * Restrict where the project is run: *test-server*
   * Source code management
     1. Repository URL: [*https://github.com/vyombhatt/zendriix-softwares-workflow.git*](https://github.com/vyombhatt/zendriix-softwares-workflow.git)
     2. Credentials: *git credentials provided*
     3. Branch Specifier: *\*/test*
   * Build Triggers: *GitHub hook trigger for GitScm polling*
2. Setting up the *push-to-prod* job:
   * Create new item as freestyle project: *push-to-prod*
   * Description: *trigger when a push is made to main branch*
   * GitHub project: [*https://github.com/vyombhatt/zendriix-softwares-workflow.git/*](https://github.com/vyombhatt/zendriix-softwares-workflow.git/)
   * Restrict where the project is run: *prod-server*
   * Source code management
     1. Repository URL: [*https://github.com/vyombhatt/zendriix-softwares-workflow.git*](https://github.com/vyombhatt/zendriix-softwares-workflow.git)
     2. Credentials: *git credentials provided*
     3. Branch Specifier: *\*/main*
   * Build Triggers: *GitHub hook trigger for GitScm polling*



**Step 5: Creating a Pipeline that connects the 2 jobs**

After setting up both the jobs, we modify the configuration for *push-to-test* job.

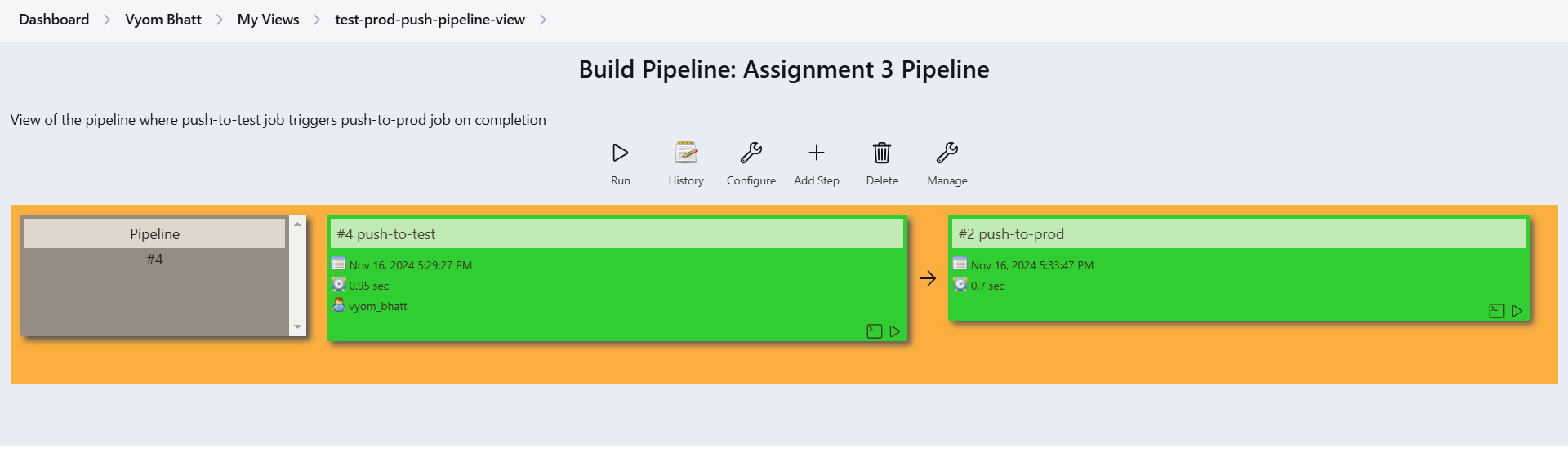
* Post Build Items:
  + Projects to build: *push-to-prod*
  + *Trigger only if build is stable*



**Step 6: Setting up the Pipeline View**

* Install the **Build Pipeline Plugin**
* Create a new view for our pipeline under *My Views*
  + Name: *test-prod-push-pipeline-view*
  + Description: *View of the pipeline where push-to-test job triggers push-to-prod job on completion*
  + Build Pipeline View Title: *Assignment 3 Pipeline*
  + Pipeline Flow:
    - Layout: *Based on upstream/downstream relationship*
    - Upstream/downstream config:
      * Select initial job: *push-to-test*

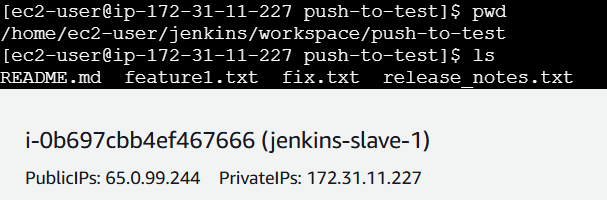
*[Rest of the settings are kept as default]*



**Step 7: Testing the Pipeline**

Commits were made on the test branch of the GitHub repository.

* The push command from the test branch triggered the *push-to-test* job that fetched all the files from the test branch and saved them in the *test-server*



* On completion of the pust-to-test job, the *push-to-prod* job was triggered that fetched all the files from the main branch and saved them in the *prod-server*

