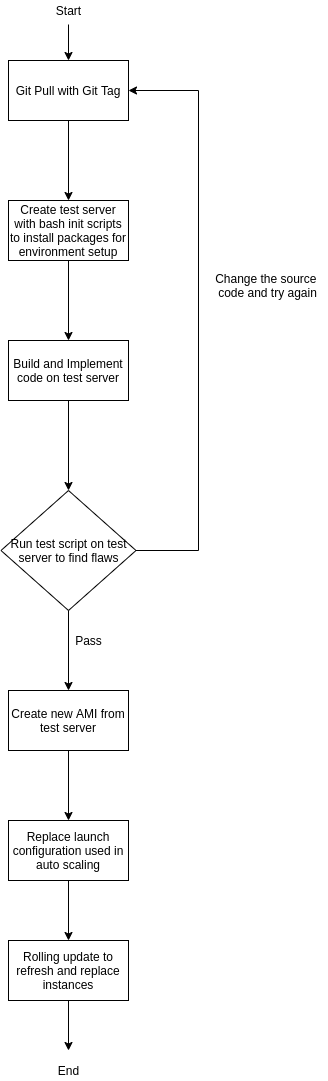
**Choice of tool**: Jenkins

FlowChart:



**Assumptions**:

* Assume that there is a jenkins master and a jenkins slave, jenkins master is used to control the slave to deploy code
* Assume that jenkins slave is properly configured with limited ssh access from jenkins master and have just enough rights to do deployment process. Its rights should be granted from a custom IAM role
* Jenkins slave is set in the public tier with elastic IP
* Different jobs are setup for different environments (i.e. 3 different jobs are required for DEV,UAT,PRD environments)
* Patching process will follow a similar process but without git pulling but instead patching after server is created and proceed with automated checking (3 more jobs are to be added if such functionality is required)

**Justification:**

* A newly created test server might incur small extra costs but a new server must be created and implemented with code for 2 purposes, to test code functionality and to create AMI after tests.
  + Testing in such manner is more rigorous and through rather than automated code scanning which might not reveal flaws if more functionality that require new packages are to be added.

**Steps explaination**:

1) Git pull code from repository, git tag must be provided to specify the version of code to be used

2) in the automated pipeline, use aws cli to create a new server with the smallest size (i.e. t2.nano) and run bash scripts to build its base environment and build the application

3)Automated test for the code can be run on the new server created. If the test fails, review and correct the code before continuing, delete test server.

4)If the test succeeds, create a new AMI to replace the old launch configuration in the auto scaling group, delete test server.

5)Do instance refresh to terminate old instances and create new instances in auto scaling group

**Steps explaination (in context of using Terraform code):**

1) Code can be pulled from git tag and saved on same folder as terraform file

2)Use terraform to generate a test environment and run automated test scripts to test code

3)If test fails, review and correct code

4)If test succeed, use terraform to deploy the new environment, destroy resource if necessary.