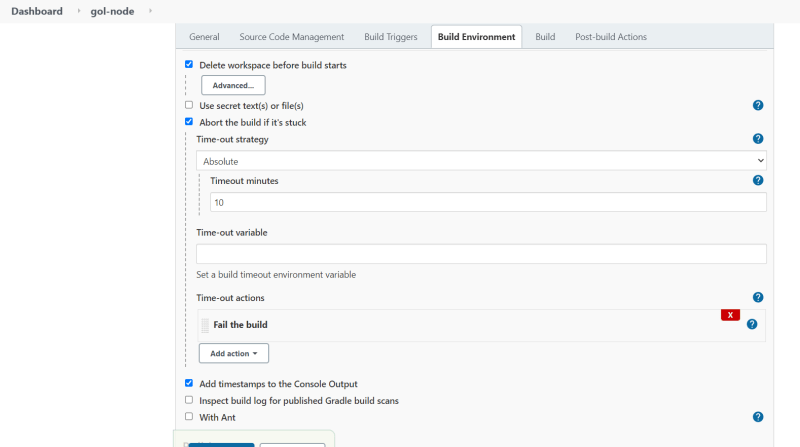
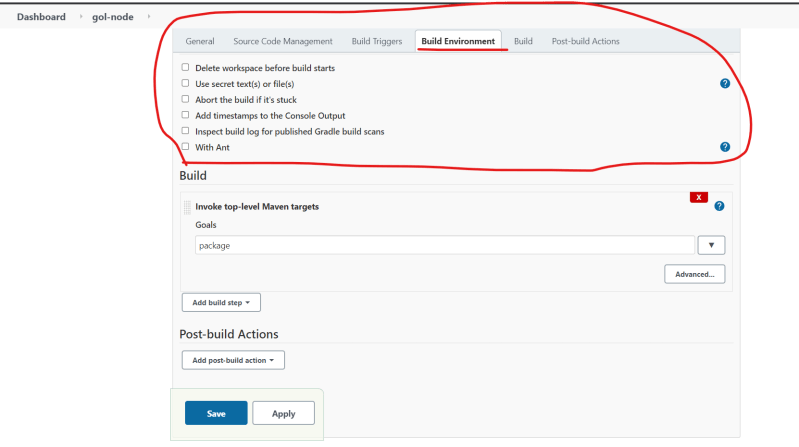
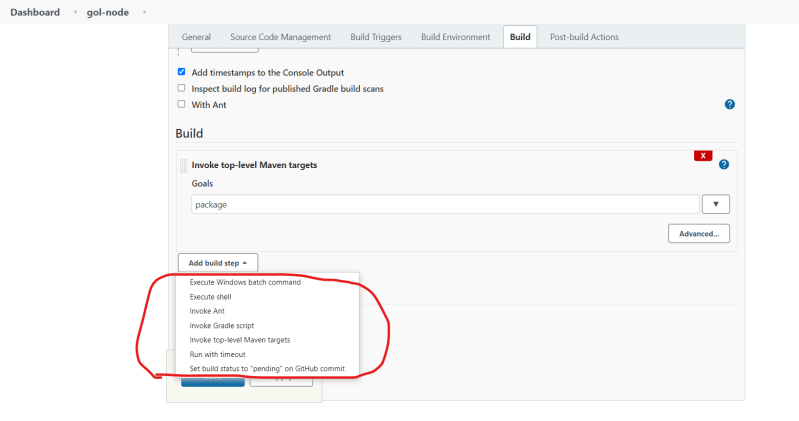
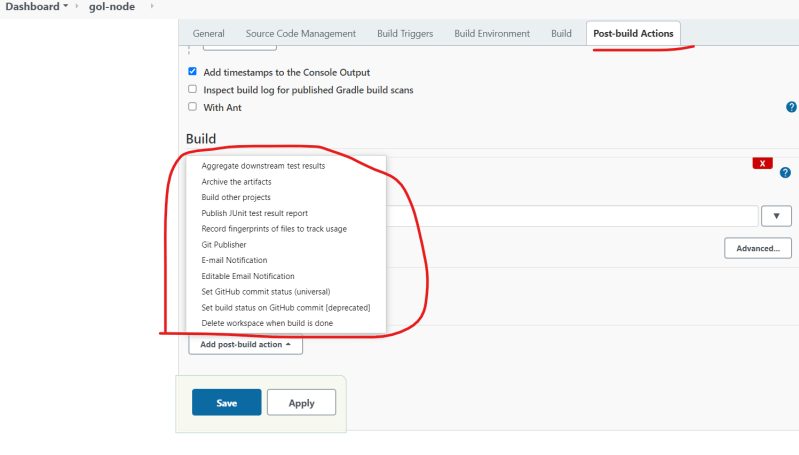
Build Environment 

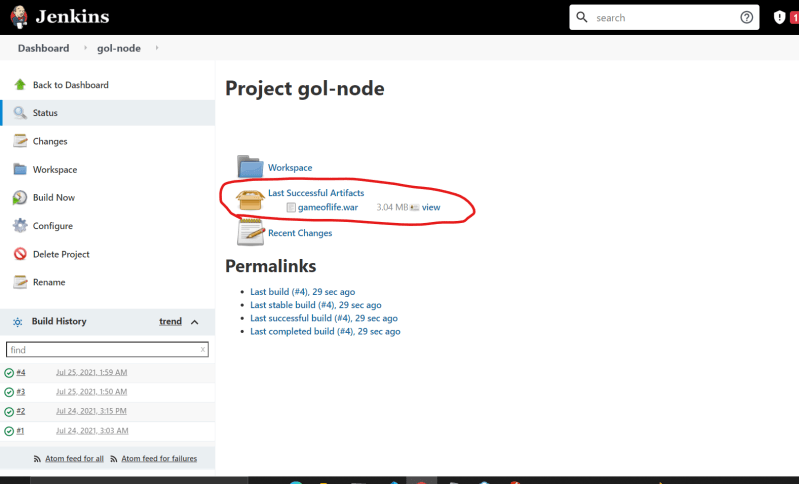
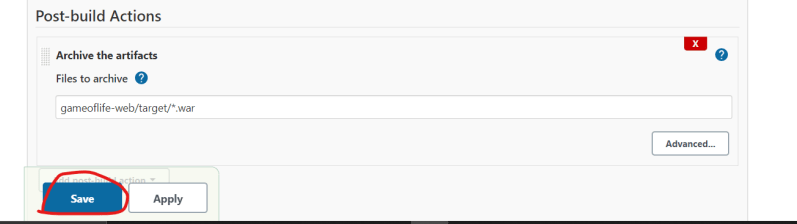
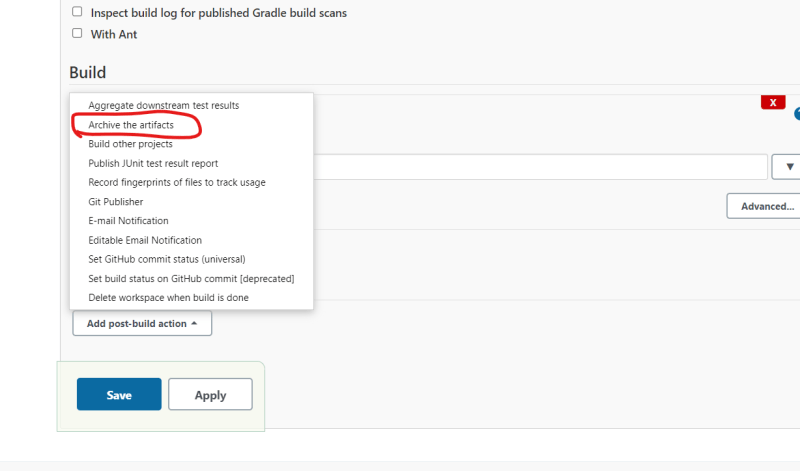
Build 

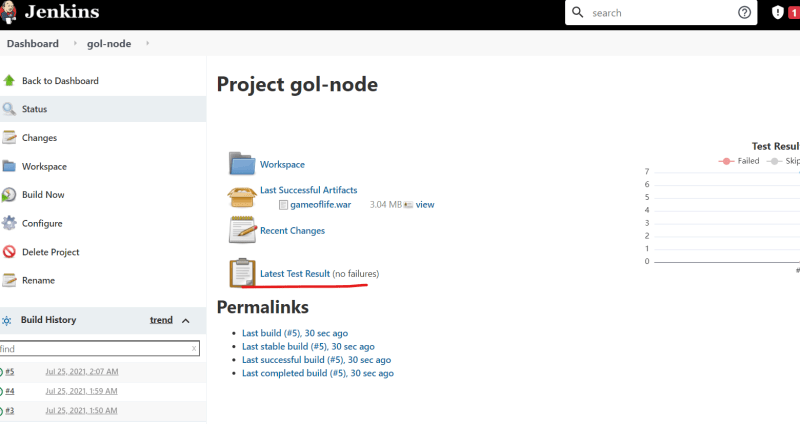
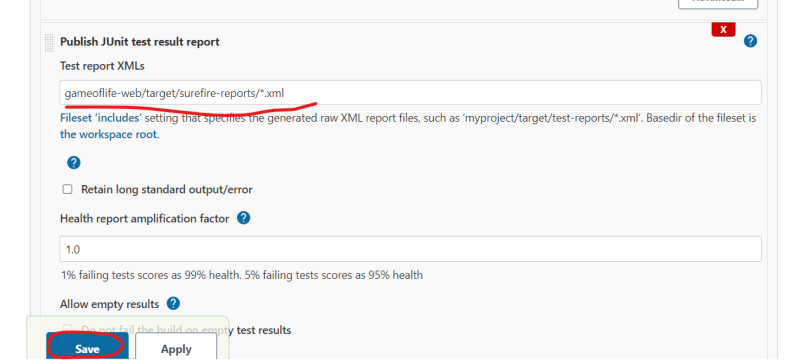
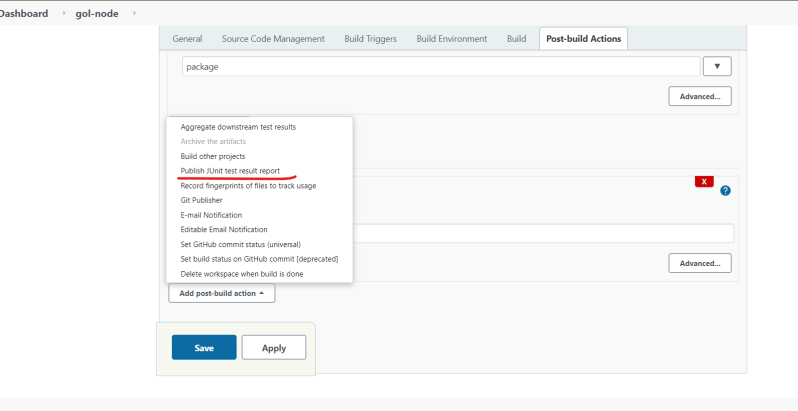
After the build is finished, the steps which we want to perform are called as post build actions.

* + Build might be failed
  + Build might be success
  + Build might be aborted

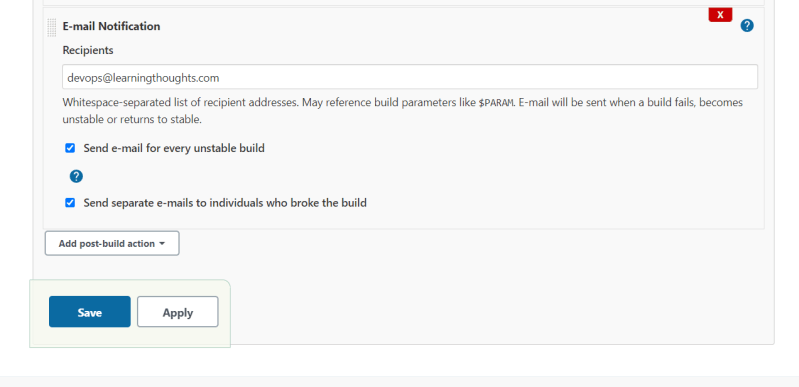


Archiving the artifacts:

Using this we can archive the build artifacts that can be dowloaded for the jenkins ui directly 

* Publishing the Junit test results: 

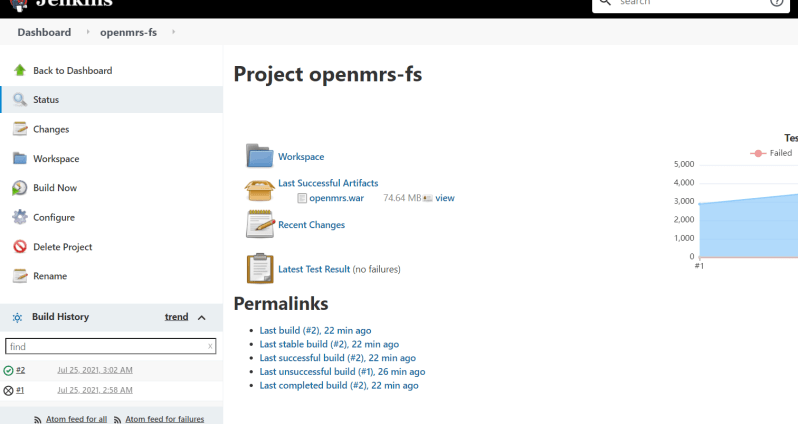
**Configuring Email Notifications in Jenkins**

* For the lab setup we will be using mail trap which is simulated smtp server(Link: <https://mailtrap.io/> )
* Navigate to Manage Jenkins => Configure System 

**Exercise – 1**

* Create a Jenkins job to build a java project openmrs (Link: <https://github.com/openmrs/openmrs-core> )
* You need to archive the openmrs war file and publish junit test results
* Also send email notification when the build is unstable.
* For configuring the test reports from any folder use the following expression for configuring the junit test results

\*\*/TEST-\*.xml



**Problems with Free Style Projects**

* The build steps are configured in external jenkins jobs
* Changes in the build steps are not version controlled.

**Jenkins 2**

* Jenkins in its newer versions started supporting pipelines-as-code feature.
* We write the build steps or the whole pipeline in a text file generally Jenkinsfile
* This Jenkins file will be part of the code
* Jenkins 2 supports two kinds of pipelines
  + Scripted Pipeline
  + Declarative Pipeline
* With Jenkins 2 the new Job Types are added
  + Pipeline
  + Folder
  + Organization
  + Multibranch Pipeline

**Syntax: Scripted vs Declarative Piplelines**

* Scripted referes to the initial way that pipelines-as-code have been done in Jenkins
* Scripted syntax relies heavily on the Groovy Language and Groovy constructs for things like error checkings and dealing with exceptions
* Declarative syntax is the newer option. This is Jenkins DSL

# Scripted Pipeline

node('GOL') {

stage('SCM') {

// clone the code

git 'https://github.com/asquarezone/game-of-life.git'

}

stage('build') {

// build the code

sh 'mvn package'

}

}

# Declarative Pipeline

pipeline {

agent { label 'GOL' }

stages {

stage('SCM') {

steps {

git 'https://github.com/asquarezone/game-of-life.git'

}

}

stage('COMPILE'){

steps {

sh 'mvn package'

}

}

}

}

* Advantages of Scripted Pipeline
  + Generally fewer section and less specification needed
  + Capability to use more procedural
  + More like creating a program
  + More flexible to do custom operations if needed
  + Ability to model more complex workflows and pipelines
* Disadavantages of Scripted Pipeline
  + More programming required
  + Syntax checking limited to Groovy Language and environment
  + Further away from traditional Jenkins model
* Advantages of Declarative Pipeline
  + More Structure – close to traditional sections of Jenkins web forms(free style project)
  + More capability to declare what is need, so more readable
  + Can be generated from Blue Ocean Graphical Interface
  + Better syntax check and error identification
* Disadvantages of Declarative Pipeline
  + Less support for iterative logic
  + Still evolving
  + More rigid structure(harder to handle customizations)
  + Not suite for complex pipelines and workflows

**Foundations**

* Jenkins Master
* Node
* Agent
* Executor 