Jenkins

ARCTECH INFO PRIVATE LTD

What is CI/CD

CONTINUOUS INTEGRATION

A development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.

Every commit made in the repository is then automatically built.

This allows the teams to detect the problems early.

Apart from this, there are several other functions like
- deploying the build application on the test server,
- providing the concerned teams with the build and test results,
- etc.

CONTINUOUS DELIVERY

Continuous delivery is an extension of continuous integration.

It automatically deploys all code changes to a testing and/or production environment after the build stage.

On top of automated testing, you have an automated release process

You can deploy your application any time by clicking a button.

To get the benefits of continuous delivery, you should deploy to test server as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

Benefits of CI

WHAT YOU NEED (COST)

Need to write automated tests for each new feature, improvement or bug fix.

Need a continuous integration server that can monitor the main repository and run the tests automatically for every new commits pushed.

Developers need to merge their changes as often as possible, at least once a day.

WHAT YOU GAIN

Less bugs get shipped to production as regressions are captured early by the automated tests.

Building the release is easy as all integration issues have been solved early.

Developers are alerted as soon as they break the build and can work on fixing it before they move to another task.

Testing costs are reduced drastically – your CI server can run hundreds of tests in the matter of seconds.

Benefits of CD

WHAT YOU NEED (COST)

Your testing culture needs to be at its best.

The quality of your test suite will determine the quality of your releases.

Your documentation process will need to keep up with the pace of deployments.

WHAT YOU GAIN

You can develop faster as there's no need to pause development for releases.

Deployments pipelines are triggered automatically for every change.

Releases are less risky and easier to fix in case of problem as you deploy small batches of changes.

Customers see a continuous stream of improvements, and quality increases every day, instead of every month, quarter or year.

Jenkins introduction

Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes.

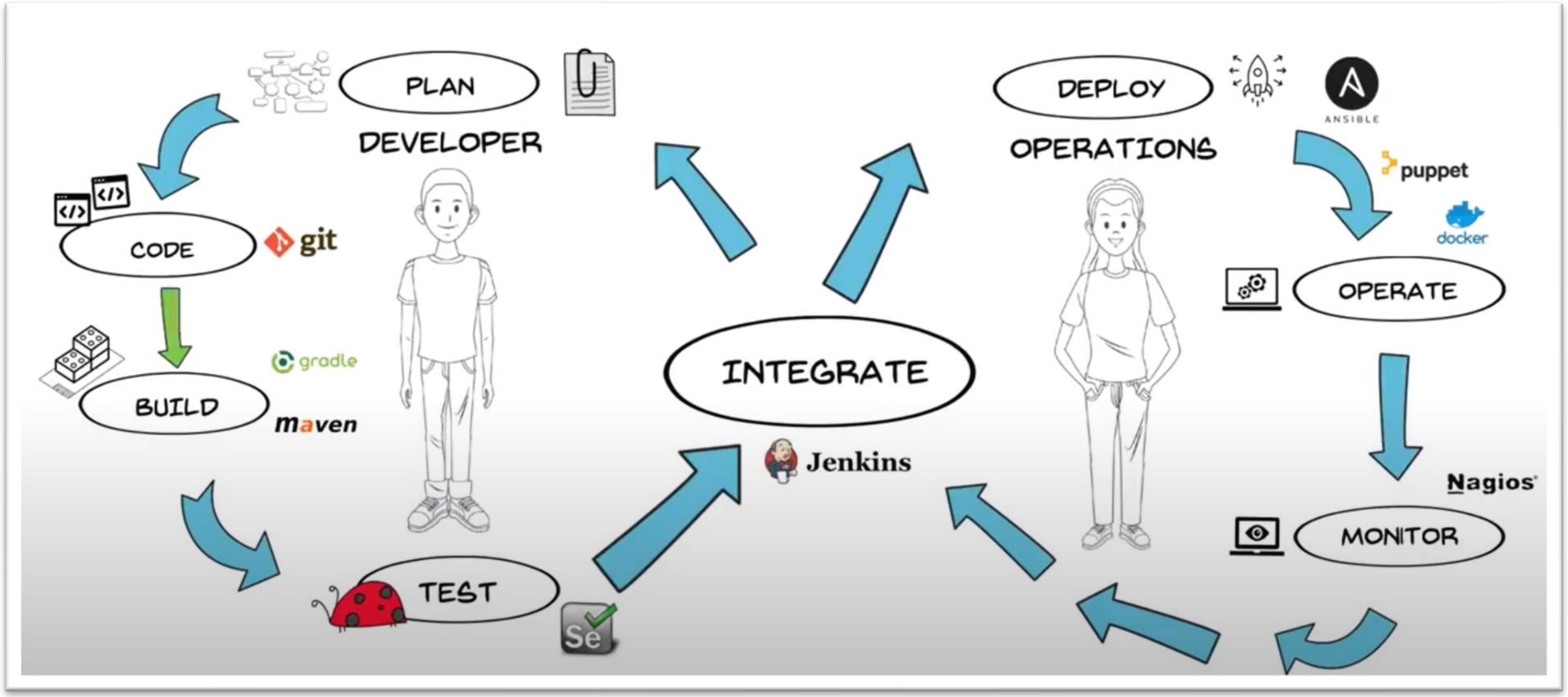
Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build.

With Jenkins, organizations can accelerate the software development process through automation.

Jenkins integrates development life-cycle processes of all kinds, including

- build,
- document,
- test,
- package,
- stage,
- deploy, and much more.

The DevOps Life Cycle



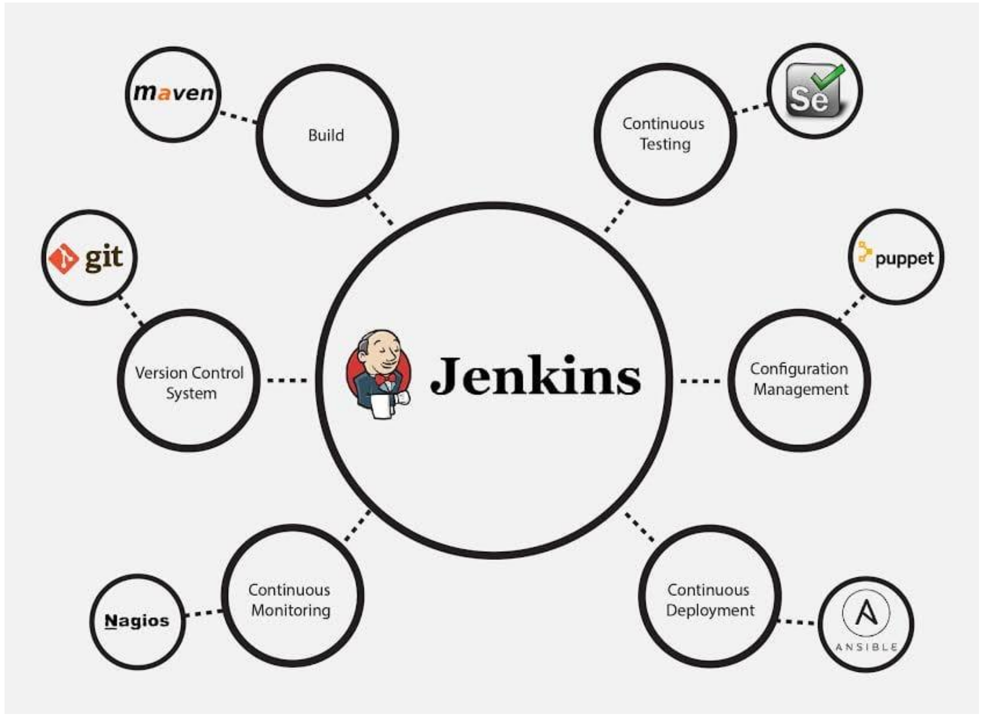
Jenkins CI with Plugins

Jenkins achieves Continuous Integration with the help of plugins.

Plugins allow the integration of Various DevOps stages.

If you want to integrate a particular tool, you need to install the plugins for that tool. For example:

- Git,
- Amazon EC2, etc.



Advantages of Jenkins

It is an open-source tool with great community support.

It is easy to install.

It has 1000+ plugins to ease your work.

If a plugin does not exist, you can code it and share it with the community.

It is free of cost.

It is built with Java and hence, it is portable to all the major platforms.

Life without CI

Let us imagine a scenario where the complete source code of the application was built and then deployed on test server for testing.

It sounds like a perfect way to develop software, but this process has many flaws.

Developers have to wait until the complete software is developed for the test results.

There is a high possibility that the test results might show multiple bugs.

It was tough for developers to locate those bugs because they have to check the entire source code of the application.

It slows the software delivery process.

Continuous feedback pertaining to things like coding or architectural issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.

The whole process was manual which increases the risk of frequent failure.

CI with Jenkins

Without CI

- the software delivery process became slow,
- the quality of software also went down.

This leads to customer dissatisfaction.

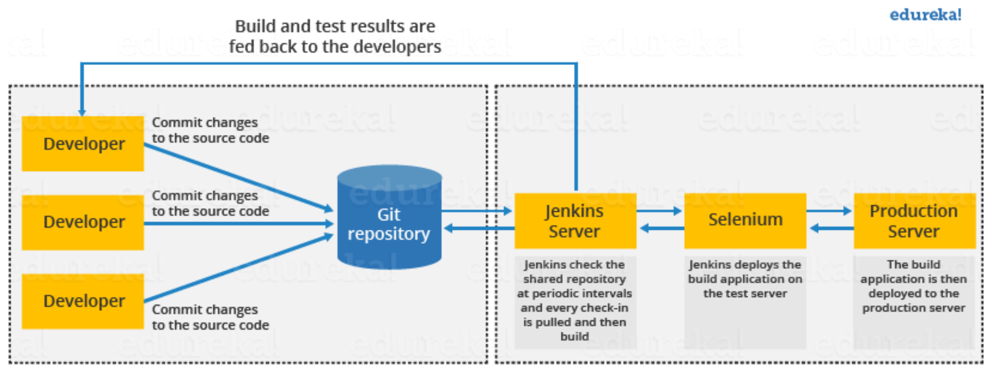
To overcome such chaos there was a dire need for developers to continuously trigger a build and test for every change made in the source code.

This is what CI is all about.

Jenkins is the most mature CI tool available

Continuous Integration with Jenkins helps overcome the above shortcomings.

CI with Jenkins


edureka!
Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.

Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.

If the build fails, then the concerned team will be notified.

If built is successful, then Jenkins deploys the built in the test server.

After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.

It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

Before and After Jenkins

Before Jenkins	After Jenkins
The entire source code was built and then tested. Locating and fixing bugs in the event of build and test failure was difficult and time-consuming, which in turn slows the software delivery process.	Every commit made in the source code is built and tested. So, instead of checking the entire source code developers only need to focus on a particular commit. This leads to frequent new software releases.
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run.
The whole process is manual	You only need to commit changes to the source code and Jenkins will automate the rest of the process for you.

Jenkins Build Pipeline

It is used to know which task Jenkins is currently executing.

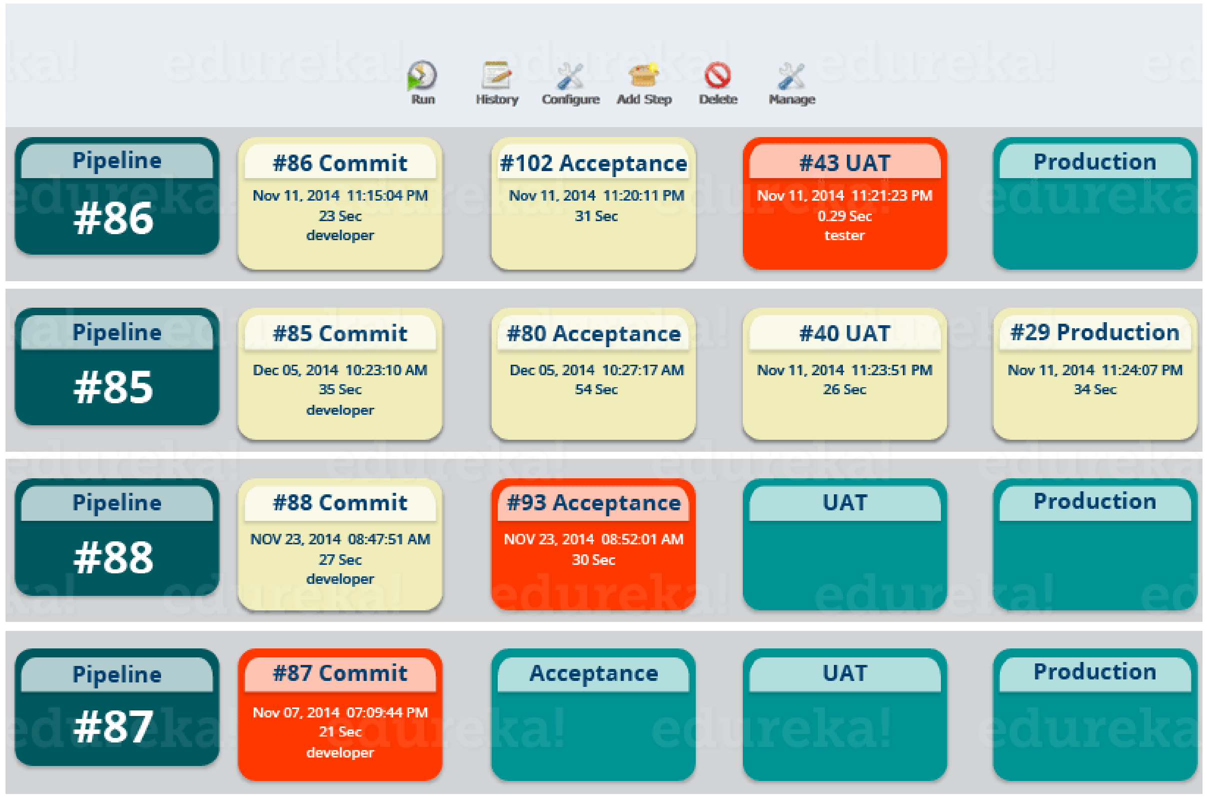
Often several different changes are made by several developers at once, so it is useful to know which change is getting tested or which change is sitting in the queue or which build is broken.

The Jenkins Pipeline gives you an overview of where tests are up to.

In build pipeline the build as a whole is broken down into sections, such as the unit test, acceptance test, packaging, reporting and deployment phases.

The pipeline phases can be executed in series or parallel, and if one phase is successful, it automatically moves on to the next phase.

Jenkins Build Pipeline



Pipeline #86	#86 Commit Nov 11, 2014 11:15:04 PM 23 Sec developer	#102 Acceptance Nov 11, 2014 11:20:11 PM 31 Sec	#43 UAT Nov 11, 2014 11:21:23 PM 0.29 Sec tester	Production
Pipeline #85	#85 Commit Dec 05, 2014 10:23:10 AM 35 Sec developer	#80 Acceptance Dec 05, 2014 10:27:17 AM 54 Sec	#40 UAT Nov 11, 2014 11:23:51 PM 26 Sec	#29 Production Nov 11, 2014 11:24:07 PM 34 Sec
Pipeline #88	#88 Commit NOV 23, 2014 08:47:51 AM 27 Sec developer	#93 Acceptance NOV 23, 2014 08:52:01 AM 30 Sec	UAT	Production
Pipeline #87	#87 Commit Nov 07, 2014 07:09:44 PM 21 Sec developer	Acceptance	UAT	Production

Jenkins build job types

Freestyle Project:

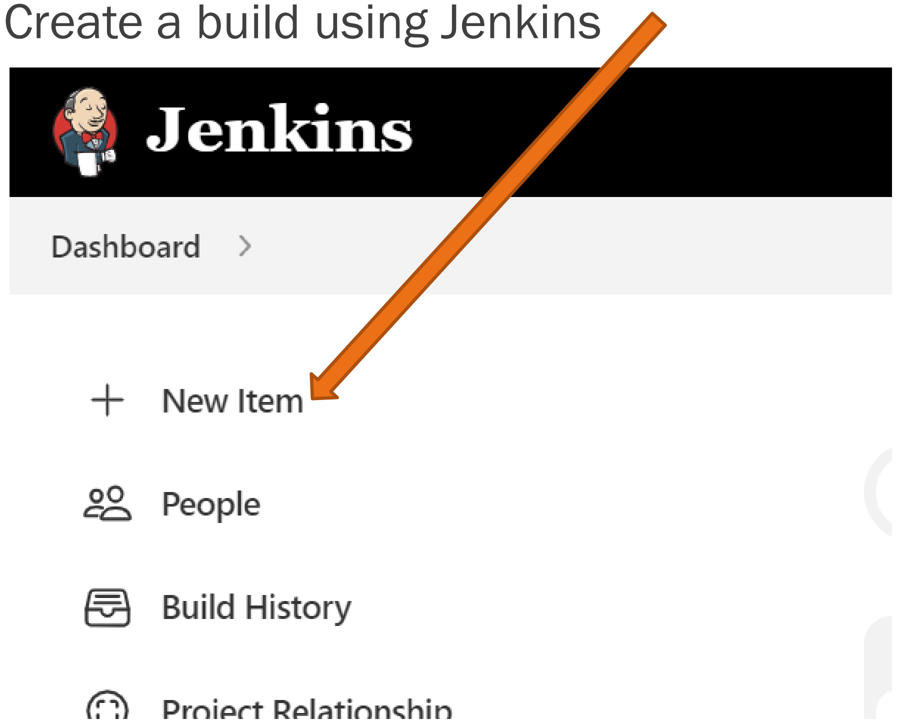
- Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility.
- The freestyle build job is the most flexible and configurable option,
- They can be used for any type of project.
- It is relatively straightforward to set up, and many of the options we configure here also appear in other build jobs.

Multiconfiguration Job:

- The “multiconfiguration project” (also referred to as a “matrix project”) allows you run the same build job on different environments.
- It is used for testing an application in different environments, with different databases, or even on different build machines.

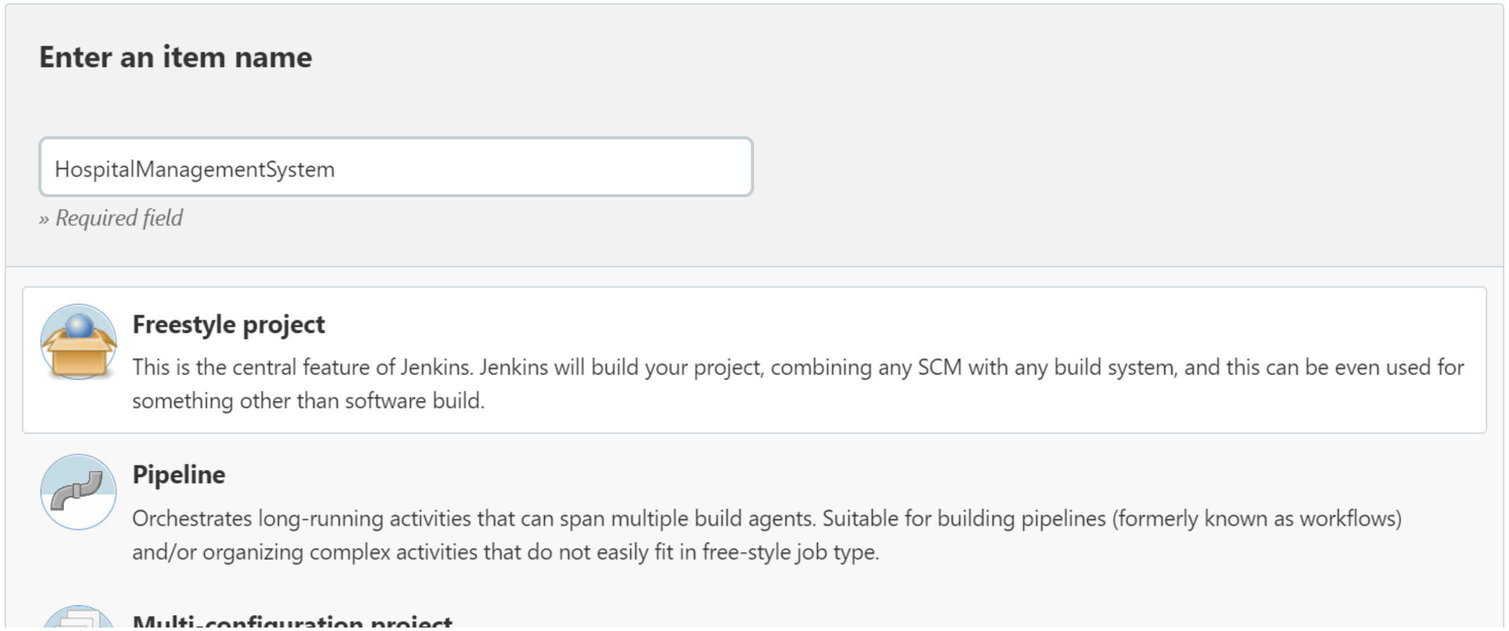
Jenkins Tutorial – Step 1

Create a build using Jenkins



Jenkins tutorial – Step 2

Enter a name and select Freestyle project.



Enter an item name

HospitalManagementSystem

» Required field

Freestyle project

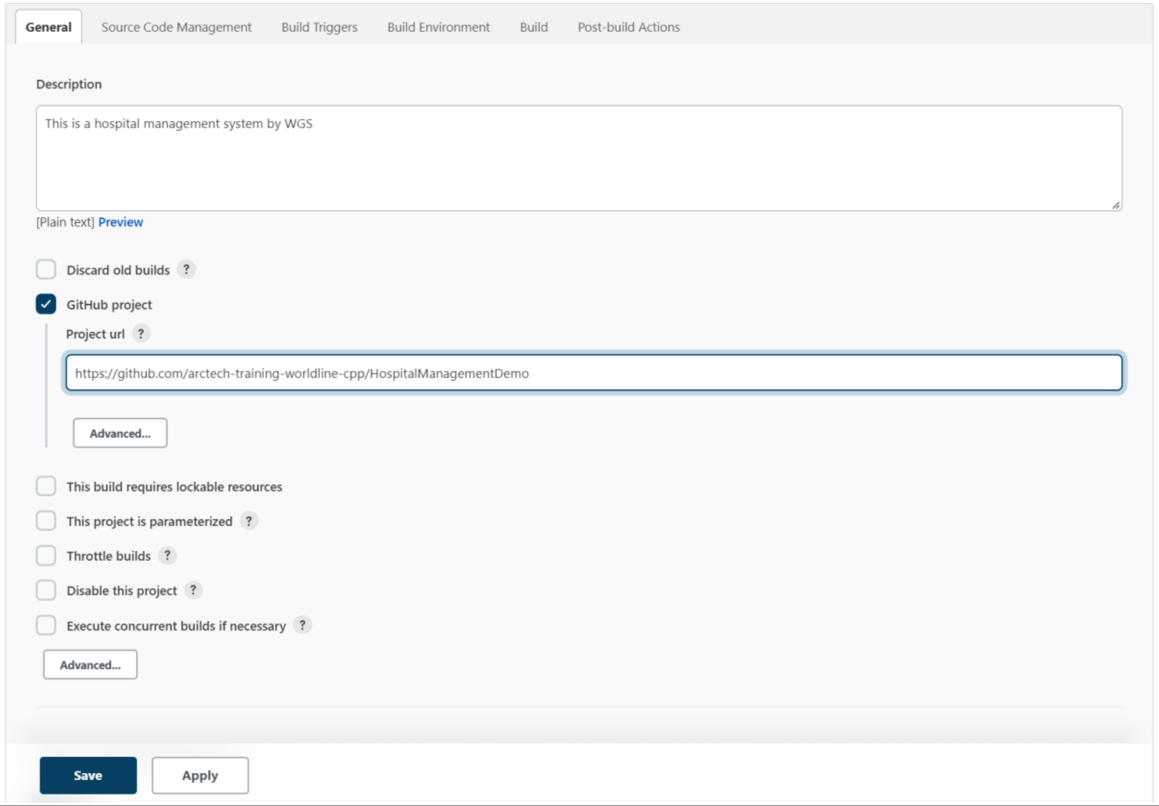
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Jenkins tutorial – Step 3

Specify the job details in the Job Configuration Screen – General Tab



Description

This is a hospital management system by WGS

[Plain text] Preview

- [ ] Discard old builds
- [✓] GitHub project

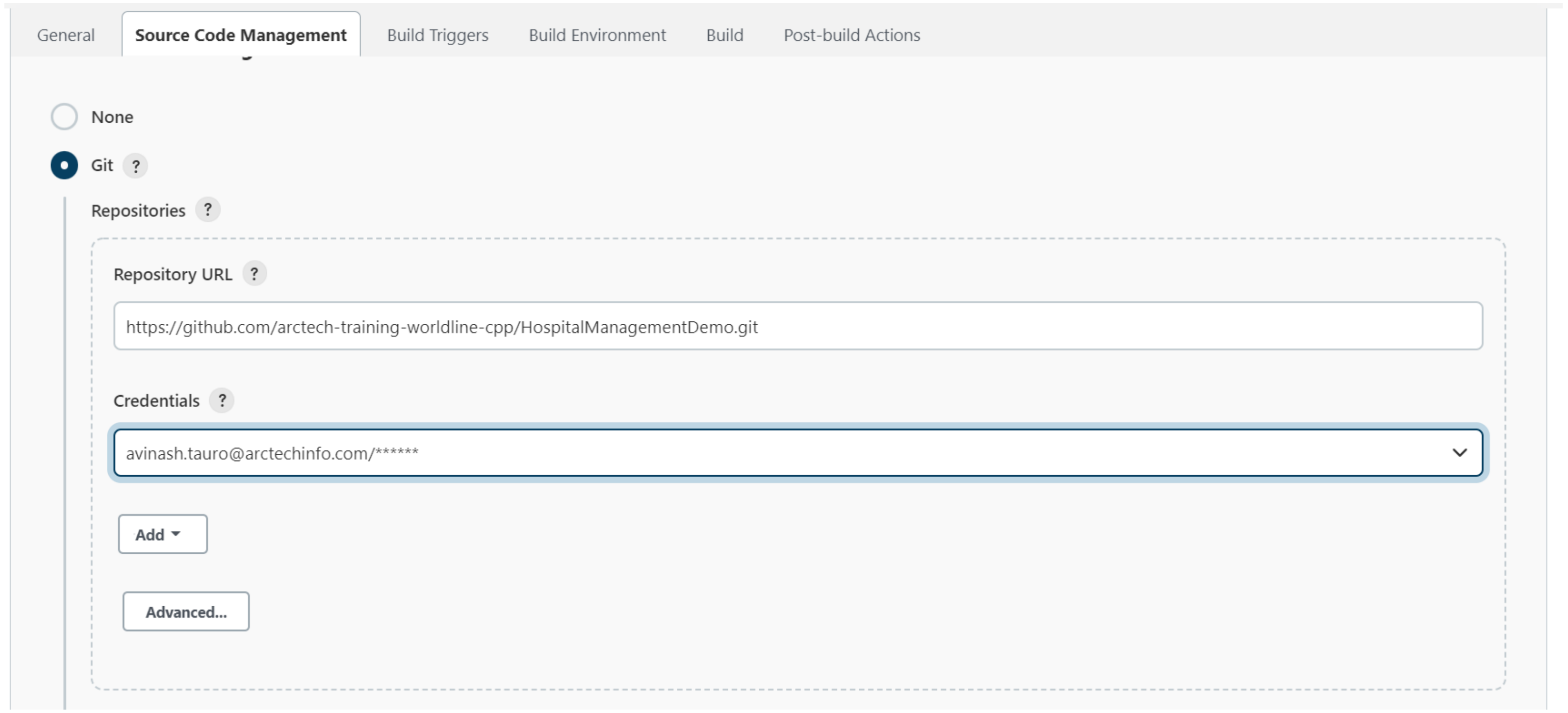
Project url

https://github.com/arctech-training-worldline-cpp/HospitalManagementDemo

- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

Jenkins tutorial – Step 4

In Source Code Management Tab, specify the Github repo path and user credentials



Git [✓]

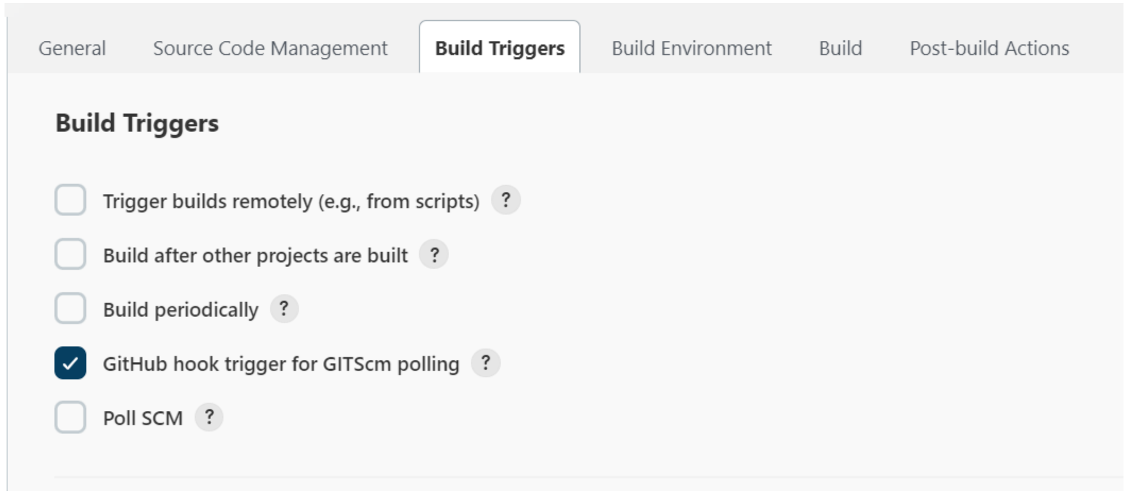
Repository URL: https://github.com/arctech-training-worldline-cpp/HospitalManagementDemo.git

Credentials: avinash.tauro@arctechinfo.com/******

Jenkins tutorial – Step 5

In the Build Trigger tab, specify the trigger which causes a Build action

If you select GitHub hook, Jenkins will periodically poll GitHub to find any new code is pushed by developer. If found, a build action will be automatically triggered

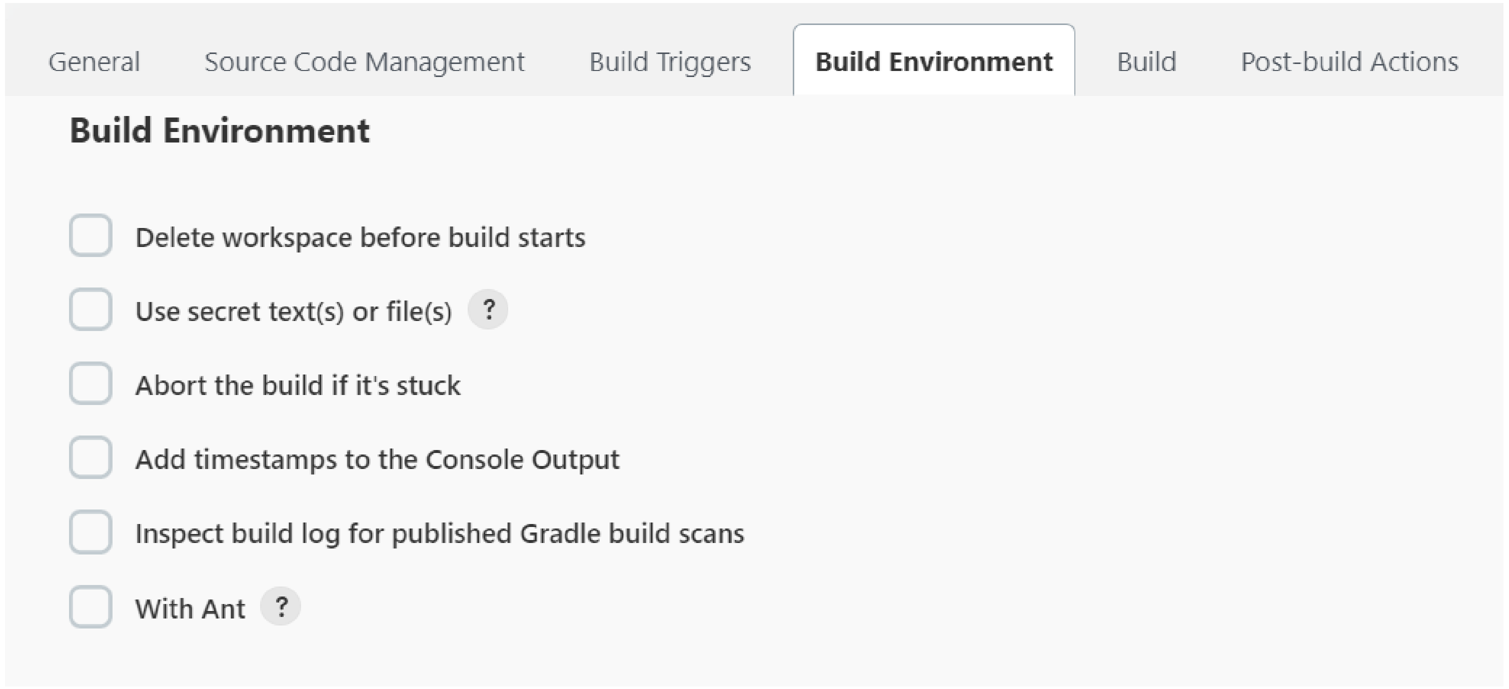


Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts) ?
- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?

Jenkins tutorial – Step 6

In the Build Environment Tab, select options as required for your project



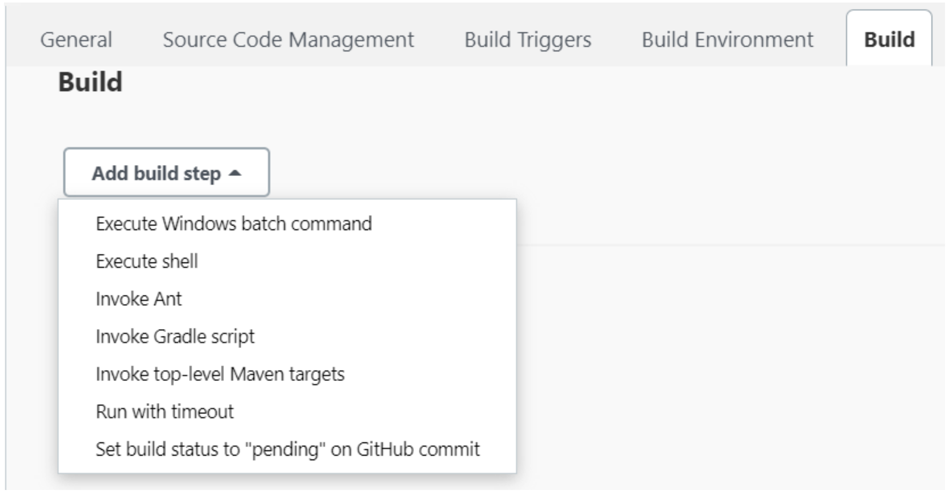
Build Environment

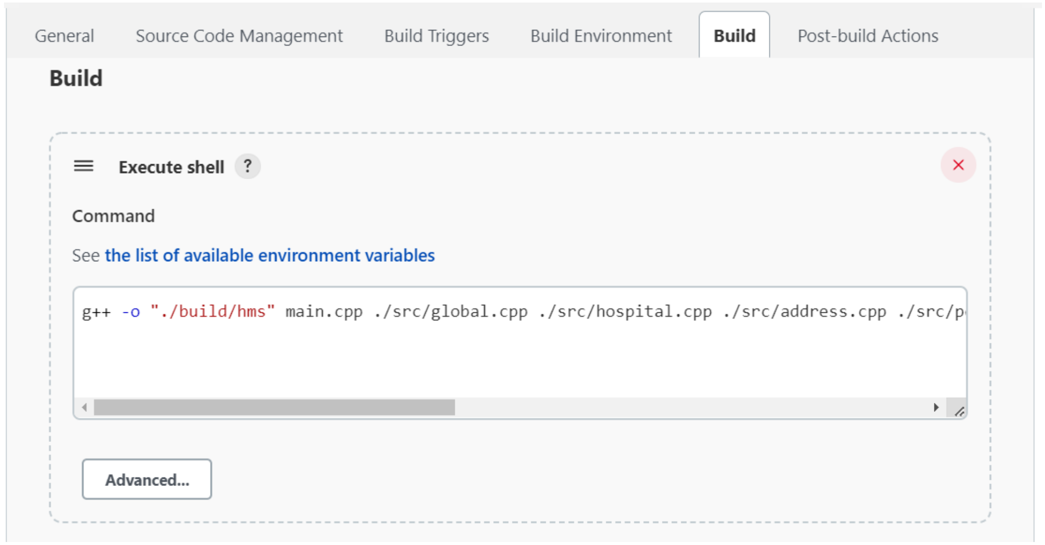
- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s) ?
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published Gradle build scans
- [ ] With Ant ?

Jenkins tutorial – Step 7

In the Build Tab, specify the build command. I selected Execute shell, to specify the unix shell command for compiling the project.

In the Execute shell section, enter the g++ unix shell command to compile your project

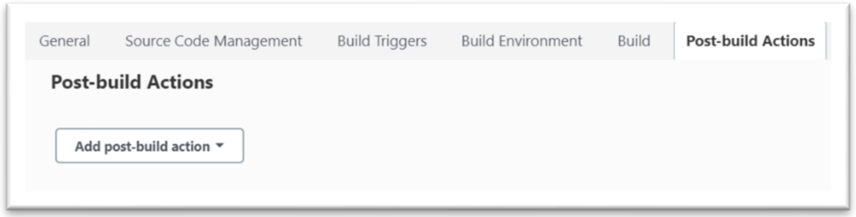


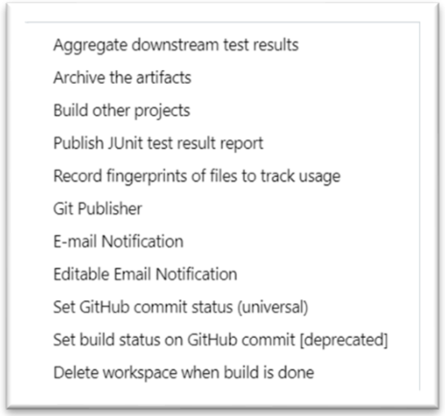


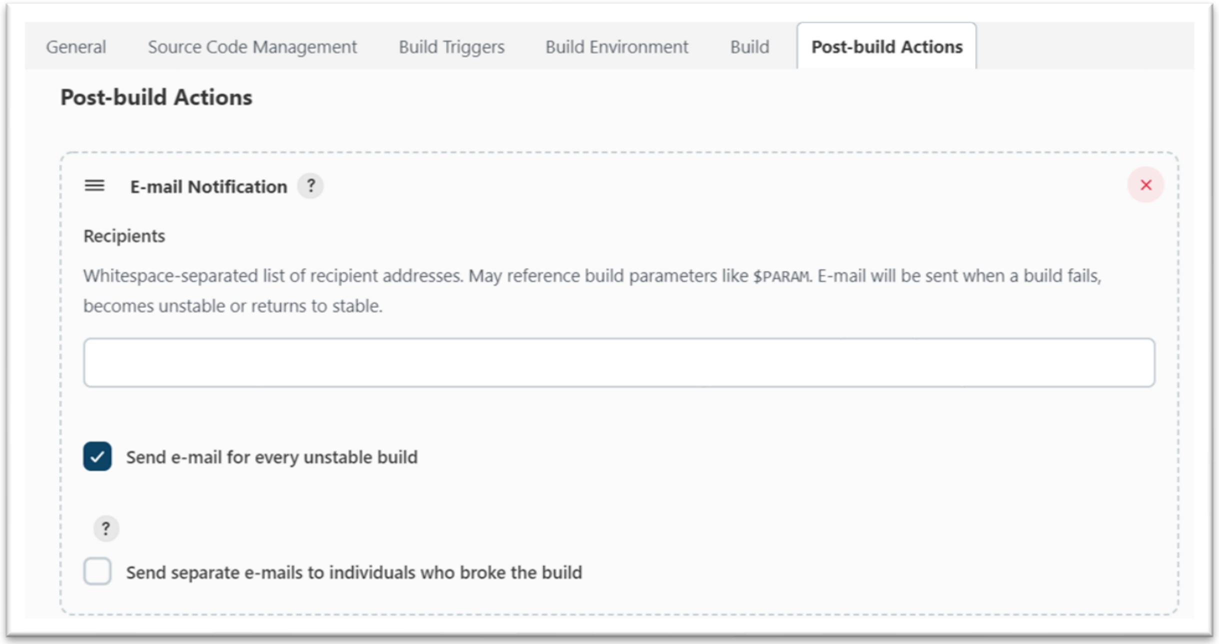
```
g++ -o "./build/hms" main.cpp ./src/global.cpp ./src/hospital.cpp ./src/address.cpp ./src/p
```

Jenkins tutorial – Step 8

In the Post-build Actions Tab, click on the Add post-build action button to select an action




Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Record fingerprints of files to track usage
Git Publisher
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done



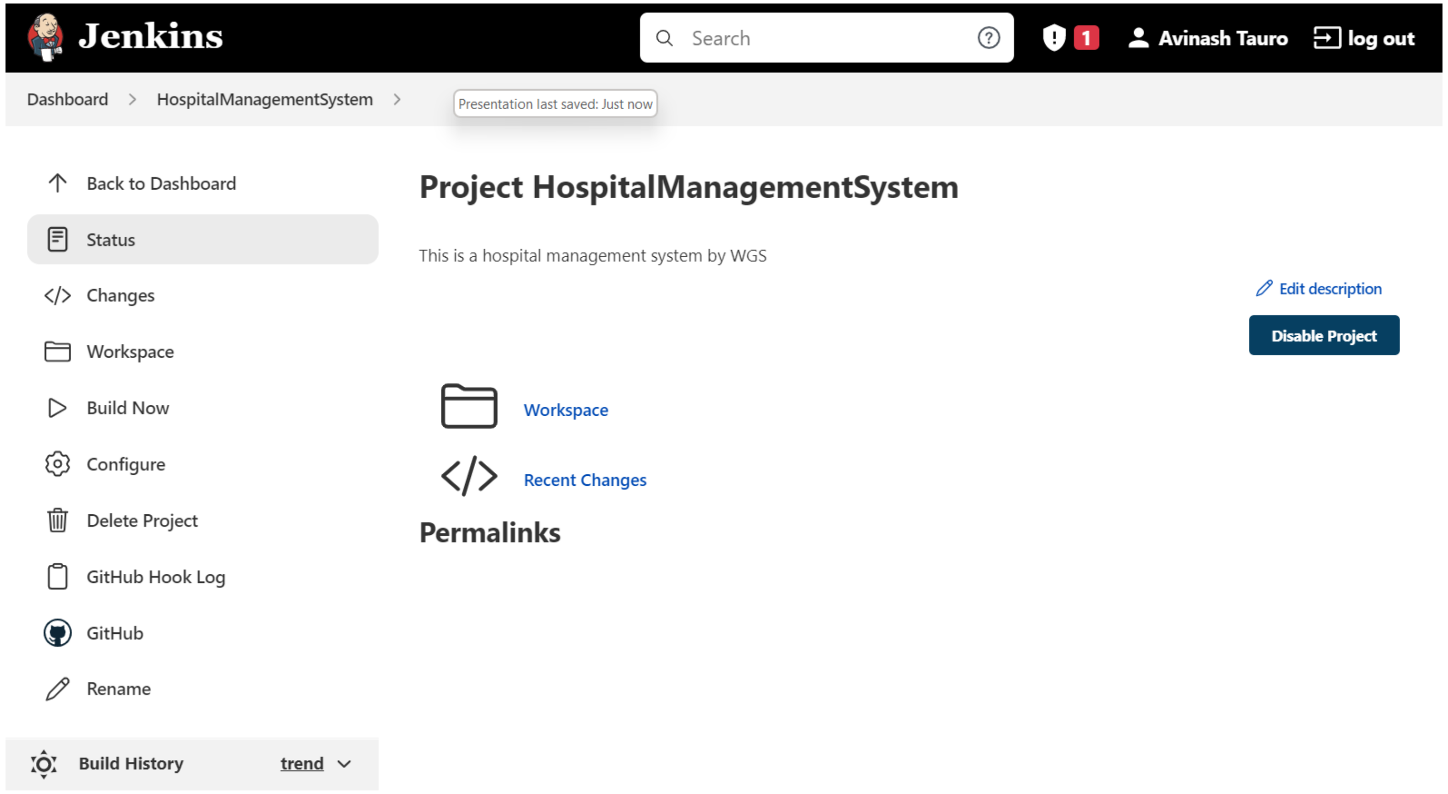
Recipients

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

[✓] Send e-mail for every unstable build

[ ] Send separate e-mails to individuals who broke the build

Project Dashboard



Jenkins

Search

1 Avinash Tauro log out

Dashboard > HospitalManagementSystem >

Presentation last saved: Just now

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

GitHub Hook Log

GitHub

Rename

Build History trend

Project HospitalManagementSystem

This is a hospital management system by WGS

Edit description

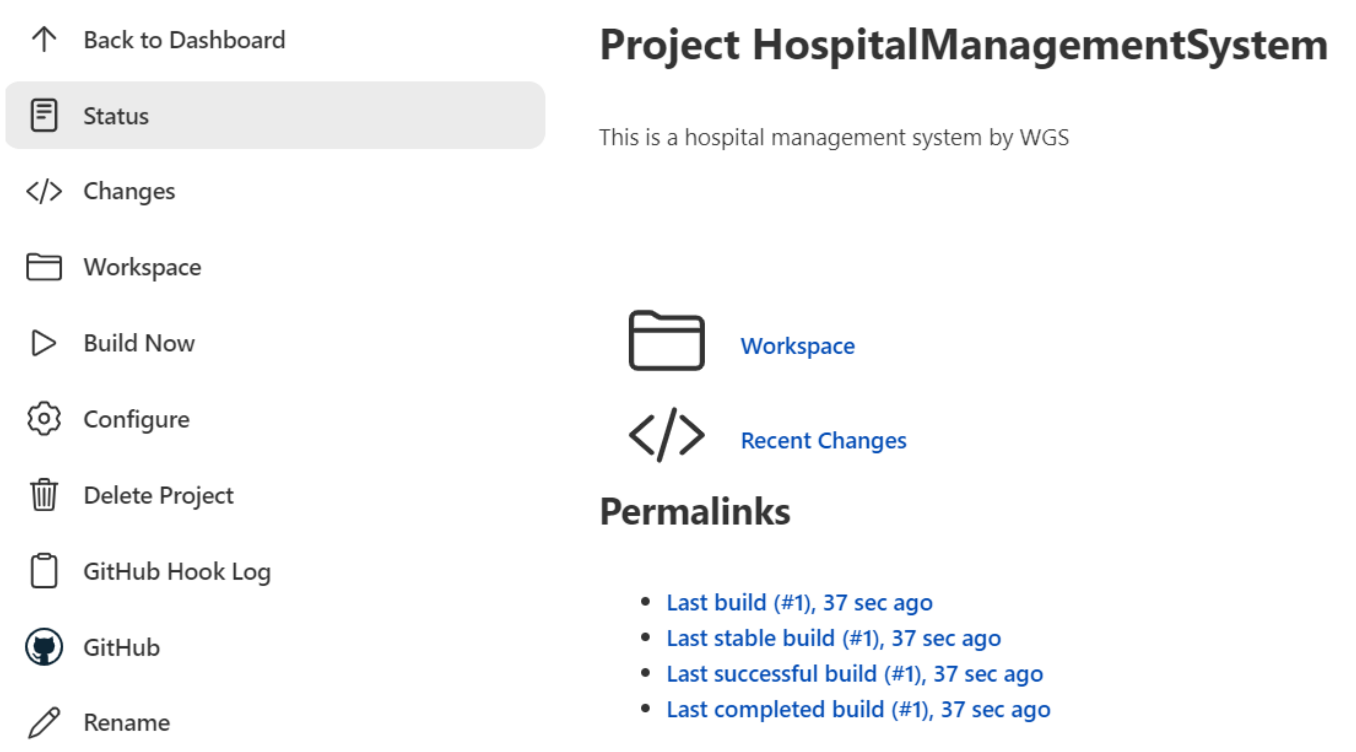
Disable Project

Workspace

Recent Changes

Permalinks

Build Result



Project HospitalManagementSystem

This is a hospital management system by WGS

Workspace

Recent Changes

Permalinks

- Last build (#1), 37 sec ago
- Last stable build (#1), 37 sec ago
- Last successful build (#1), 37 sec ago
- Last completed build (#1), 37 sec ago