

# Jenkins Continuous Integration Admin Guide

This article describes how to install and set up Jenkins for continuous integration (e.g. build automation).

## Expected Outcome

Whenever engineers push out code changes to GitHub or GitLab, Jenkins server will pull the latest copy of our source codes (e.g. FOSS mission operations software, Julia for internal engineering), build the software, run test automation, and publish to the target testing or production machine within minutes without manual intervention.

## Overview

### When to Use

Whenever there is any code change, you want to re-build the software codes, and push them to your target system environment automatically.

### Use Case Scenario

Open source developers have checked in their new code changes to <https://github.com/audacyDevOps/missionOpsWidgets> (Mission Operations widgets). Jenkins server will automatically detect the code changes, start the build job to pull the codes from GitHub (#5 MissionOpsApp Build from the build pipeline screenshot below), retain a copy of the relevant artifacts or binary files (#7 MissionOpsApp Prepare), execute any automated tests (#4 MissionOpsApp Test). If these 3 steps are executed successfully, Jenkins will push the final build to the target server host (e.g. production Web server).

The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are visible, along with a search bar and user information (ray | log out). Below the header, a sidebar on the left lists navigation options: New Item, People, Build History, Edit View, Delete View, Project Relationship, Check File Fingerprint, and Manage Jenkins. The main content area is titled "Build Pipeline" and shows a sequence of three build stages connected by arrows. The first stage is "#5 MissionOpsApp Build" (green box), the second is "#7 MissionOpsApp Prepare" (green box), and the third is "#4 MissionOpsApp Test" (red box). Each stage includes a status icon, a timestamp, and a duration. Above the stages, there are icons for Run, History, Configure, Add Step, Delete, and Manage. The interface also includes a "DISABLE AUTO REFRESH" link in the top right corner.

## Benefits

- Continuous integration (e.g. test automation)
- Customized workflow by projects

## Jenkins Capabilities

In DevOps context, the entire workflow to pull code changes, run test automation and push final binaries to the target system environment is called "Build Pipeline". In other words, build pipeline is a visual representation of the build and deployment workflow process. Build pipeline is part of the Development Operations capabilities to achieve continuous integration and continuous delivery.

Jenkins server plays the role of workflow orchestration, by defining each workflow step, and customizing scripts and processing logic. With recent new technology updates in DevOps space, Jenkins server is used to:

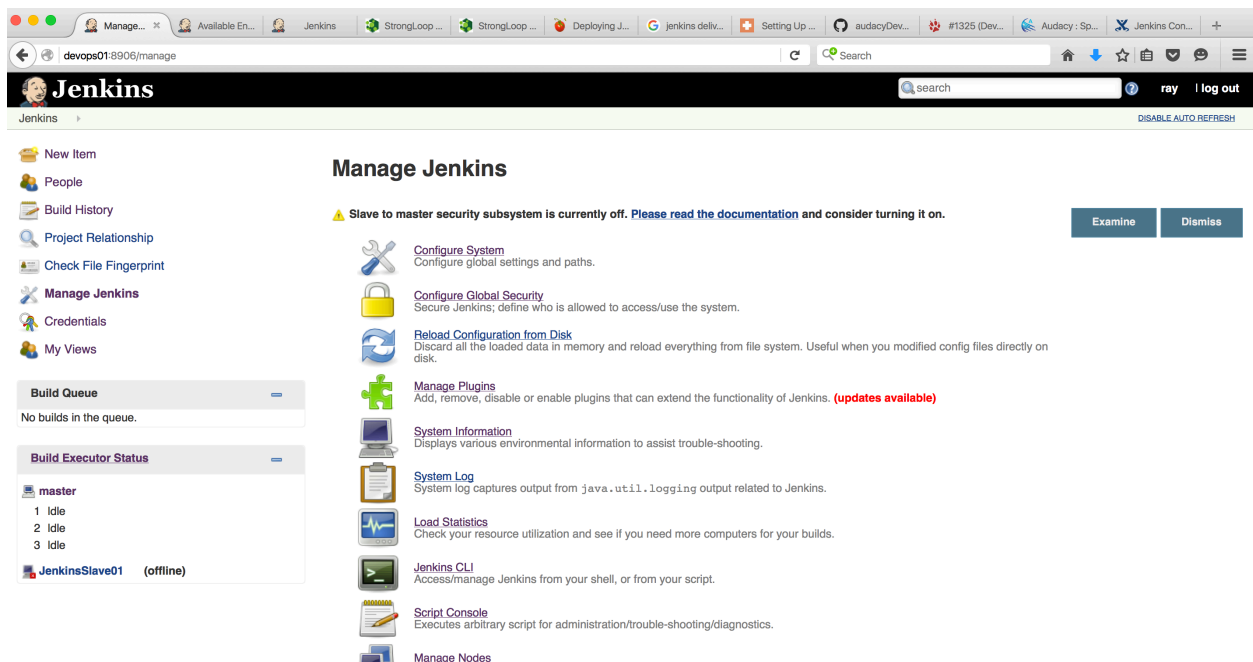
- Integrate with GitHub or GitLab (e.g. execute a build if there is a code change in GitHub)
- Integrate with NodeJS Web server
- Integrate with configuration management tools such as Ansible
- Customize security test automation tools such as w3af

The current scope of Jenkins setup is focused on the integration with GitHub only.

## Software Installation

- Pre-requisites:
  - Create a Linux machine (aka Docker Host) from RackSpace (or Amazon AWS in future).
  - Create a shared storage volume /mnt/data01 and attach to the Linux host. Refer to RackSpace help for details of how to create shared storage volume.
  - Select CentOS 7 which is Redhat Enterprise compatible because it is widely used for enterprise and commercial software deployment, particularly for high availability support.
  - Install Docker RPM package. Refer to <https://docs.docker.com/engine/installation/linux/centos/> for details.
  - Assumptions: use "sudo" (root privilege) to execute docker commands.
  - Example: the Linux host has IP address 192.168.1.10. I add this alias devops01 in my /etc/hosts for ease of SSH access. We can create subdomain name later.
- Create a docker instance
  - Issue the command from docker host: "sudo docker run --name jenkins01 -d -p 8906:8080 -p 50000:50000 -v /mnt/data01/[jenkins:/var/jenkins\\_home](#) jenkins"

- This docker command will create a docker container with the name "jenkins01". You can access the Jenkins server via port 8906 of the docker host (Linux machine set up earlier, e.g. 192.168.1.10). Internally, docker will do a port forwarding from port 8906 of the docker host (parent Linux machine) to the port 8080 of the docker container (Jenkins instance). It will store the Jenkins configuration and working data under /mnt/data01/jenkins of the Linux machine (docker host), but the docker container will map this shared storage volume to its local folder /var/jenkins\_home.
- From a Web browser, enter the URL <http://192.168.1.10:8906>. Refer to the following screenshot. There is no user login on startup, and you need to start configuration to enable security. Refer to the section How to Configure Jenkins.



- Install plugins
  - From Jenkins home page, select Manage Jenkins | Manage Plugins.
  - Select the following plugins.
    - Checkstyle
    - Github
    - Github authentication
    - Copy Artifact
    - NodeJS
    - Embeddable build status
    - TAP
    - Build pipeline
    - Delivery pipeline
  - You can select Restart server (by checking install with restart option) later.

Updates Available Installed Advanced						
Enabled	Name ↓	Version	Previously installed version	Pinned	Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Ant Plugin</a> Uses the <a href="#">OWASP Java HTML Sanitizer</a> to allow safe-seeming HTML markup to be entered in project descriptions and the like.	<a href="#">1.2</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Build Pipeline Plugin</a> This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.	<a href="#">1.4.9</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Checkstyle Plug-in</a> This plug-in collects the <a href="#">Checkstyle</a> analysis results of the project modules and visualizes the found warnings. If you like this open source plug-in please consider supporting my work by buying my Android game <a href="#">Inca Trails</a> .	<a href="#">3.44</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Clover plugin</a> This plugin integrates <a href="#">Clover code coverage reports</a> to Jenkins.	<a href="#">4.5.0</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Copy Artifact Plugin</a> Adds a build step to copy artifacts from another project.	<a href="#">1.37</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Credentials Plugin</a> This plugin allows you to store credentials in Jenkins.	<a href="#">1.24</a>	Downgrade to 1.18	Unpin	Uninstall	
<input checked="" type="checkbox"/>	<a href="#">CVS Plug-in</a> Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient.	<a href="#">2.11</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Delivery Pipeline Plugin</a> This plugin visualize Delivery Pipelines (Jobs with upstream/downstream dependencies)	<a href="#">0.9.8</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">embeddable-build-status</a> This plugin adds the embeddable build status badge to Jenkins so that you can easily hyperlink/show your build status from elsewhere.	<a href="#">1.9</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">External Monitor Job Type Plugin</a> Adds the ability to monitor the result of externally executed jobs.	<a href="#">1.4</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Git client plugin</a> Shared library plugin for other Git related Jenkins plugins.	<a href="#">1.19.4</a>			Uninstall	
<input checked="" type="checkbox"/>	<a href="#">Git plugin</a>	<a href="#">2.4.2</a>			Uninstall	

# How to Configure Jenkins Server

- After DevOps engineer launches the docker instance of Jenkins server, enter the URL <http://192.168.1.10:8906> from a Web browser.
- Create your admin user id
  - From top of the page, select Sign-up.
  - Create your first admin user id, e.g. audacy.
  - Select a strong password, e.g. at least 8 alpha-numeric characters, mixed with special characters.
  - Optional: You can add a backup user id, or a few more user ids. You can assign them role-based access later without the need to share 1 root password.
- Enable security
  - From main menu (Manage Jenkins), select Configure Global Security, check Enable Security. Select Access Control | Security Realm | Jenkins' own user database.
  - Uncheck "Allow users to sign up" (note: make sure you have all admin user ids created in last step)
  - In future, we can integrate with LDAP (Crowd LDAP server).
  - For Authorization, select Project-based Matrix Authorization Strategy



# Configure Global Security

☒ Enable security

TCP port for JNLP slave agents ☒ Fixed :  ☐ Random ☐ Disable

Disable remember me ☐

Access Control

## Security Realm

☐ Delegate to servlet container

☐ Github Authentication Plugin

☒ Jenkins' own user database

☐ Allow users to sign up

☐ LDAP

☐ Unix user/group database

## Authorization

☐ Anyone can do anything

☐ Github Committer Authorization Strategy

☐ Legacy mode

☐ Logged-in users can do anything

☐ Matrix-based security

☒ Project-based Matrix Authorization Strategy

- Set up role-based access control
  - From Manage Jenkins | Configure Global Security | Enable Security | Access Control | Authorization | Project-based Matrix Authorization Strategy, add your user id.
  - You can grant appropriate access rights. At a minimum, you need to check Overall | Administer for the target user id.
  - You can check "Job | ViewStatus" for the user Anonymous. This allows build status to be displayed on the Web page without relying on specific user permission after login.

## Authorization

- ☐ Anyone can do anything
- ☐ Github Committer Authorization Strategy
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☐ Matrix-based security
- ☒ Project-based Matrix Authorization Strategy

User/group	Overall	Credentials	Slave	Job	Run	View	SCM
	ConfigureUpdateCenter Administer	Read UploadPlugins RunScripts Create Delete ManageDomains Update View Configure Connect Create Delete Disconnect Build	Build Configure Connect Create Delete Disconnect Build	Build Configure Connect Create Delete Discover Read ViewStatus Workspace Delete Update Configure Create Delete Read Tag			
 audacyAdmin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
 ray	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add:

Add

Refer to <https://strongloop.com/strongblog/roll-your-own-node-js-ci-server-with-jenkins-part-1/> for details.

# How to Set Up Trigger in GitHub

In order for Jenkins to kick off any build job right after developers check in codes to GitHub, we need to configure GitHub Webhook. This is a followup task after Jenkins has configured GitHub hook/integration.

- Login to GitHub
- Pre-requisite: you create a personal access token (aka OAuth access token) for your Jenkins for access. Refer to previous section on Jenkins configuration.
- Select your GitHub repo (project), select Settings tab | Webhooks & services (do not confuse with the overall repository settings, which is an icon on top right corner of the GitHub home page)

audacityDevOps / sampleMissionOpsApp

Unwatch 1

★ Star 0

🍴 Fork 0

<> Code
0 Issues
Pull requests 0
Wiki
Pulse
Graphs
Settings

Options

Collaborators

Branches

Webhooks & services

Deploy keys

Webhooks
Add webhook

Webhooks allow external services to be notified when certain events happen within your repository. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ http://devops.audacity.space:8906/github-webhook/ (push)
✎ ✕

Services
Add service

- Click Add webhook.
  - Enter the name of your Jenkins server, with the suffix `"/github-webhook/"`. In our case, <http://devops.audacity.space:8906/github-webhook/>
  - Copy the personal access token to the field "Secret". This is the security token that you generated from earlier step when setting up GitHub and Jenkins GitHub plugin.

Options
Collaborators
Branches
Webhooks & services
Deploy keys

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL \*

http://devops.audacity.space:8906/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

\*\*\*\*\* — Edit

Which events would you like to trigger this webhook?

☒ Just the push event.
 ☐ Send me everything.
 ☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Update webhook

Delete webhook

# How to Set up Build Pipeline

This section explains how to configure Jenkins to set up a build pipeline to automatically pull new codes changes from GitHub, and push to a shared folder that is used by your target test Web server. It is based on a single docker host scenario. We need separate documentation for the multiple docker hosts scenario, where not all your target production servers can share the same shared folder.

### **Sample Scenario #1**

- Developers make code changes in GitHub (e.g. Open Source FOSS mission operations project)
- You want Jenkins to automatically pull the code changes, and to copy the codes to the target Web servers

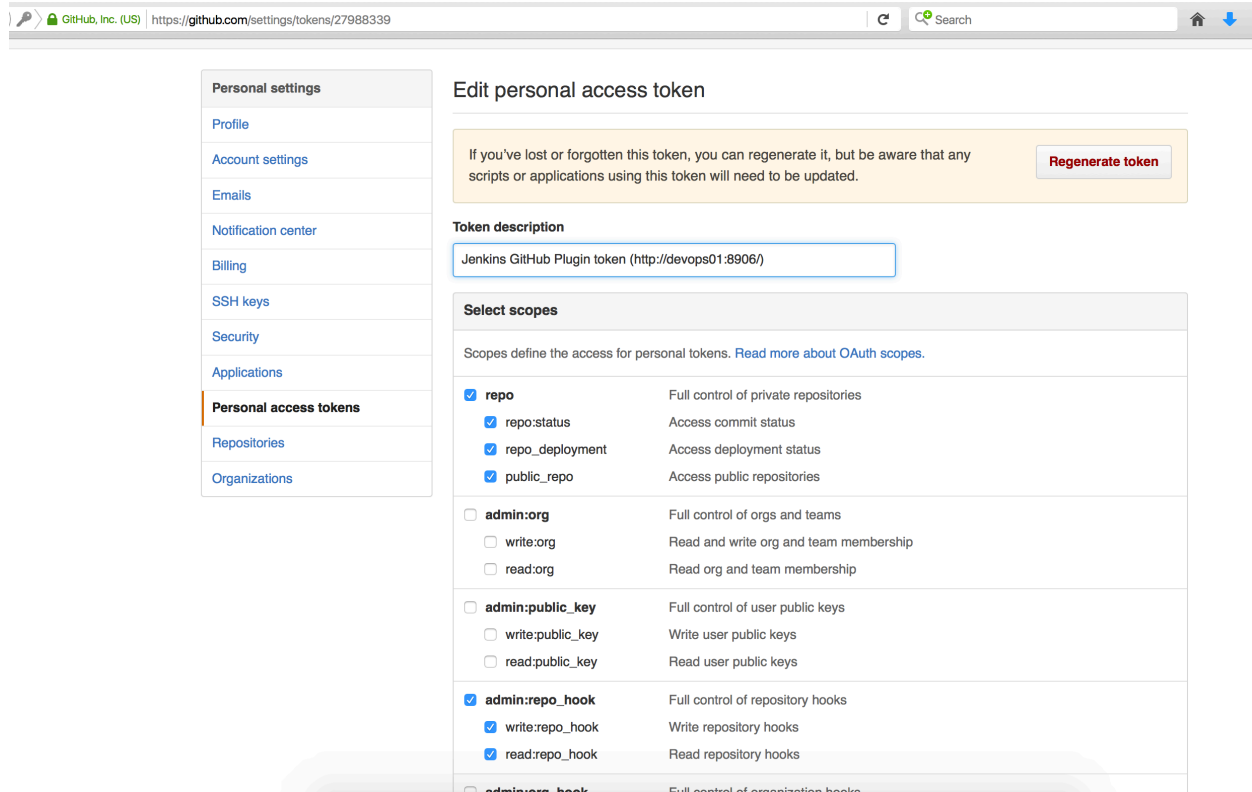
### **Sample Scenario #2**

- Audacy engineers complete orbit simulation codes using Julia in GitLab private project.
- You want Jenkins to automatically pull the Julia codes, test it with automated test scripts, and push the codes to the target Julia server.

### **Build Pipeline Instructions**

- Pre-requisites - set up GitHub
  - Create GitHub repository (or GitLab private project)
  - Create personal access token in GitHub (from top right menu, select Settings icon. Then pick Personal settings | Personal access tokens) for Jenkins to use. When you create personal access token, check the scope for (1) repo, (2) admin repo hook. Refer to the screenshot example below.





## Steps Purpose

## Instructions

- |  |  |   |
|--|--|---|
| <p>Design how many steps you need for your GitHub project</p> <p>1</p> | <p>N/A</p> <p>Create login to GitHub</p>   | <p>e.g. Git pull -&gt; Build -&gt; Copy</p> |
| <p>2</p> <p>personal access token</p>                                  | <p>From Personal settings create a personal access token.</p> <p>Select Scopes for (1) repo, (2) admin:repo hook</p> | <p>Refer to the GitHub screensh</p>         |

☒ Delivery Pipeline configuration

Stage Name

Task Name

Customize Task Description Template

From Jenkins home page, click New Item.


Edit description to describe this step for pulling code changes from GitHub to build the application.

Check Delivery Pipeline configuration, and enter the stage name and task name as "Build".

3 Create  
#1 Build  
step

Under Source Code Management section, select Git. Enter the repository URL for your GitHub repo. For credentials, select Add to add your personal access token (you need to specify "Secret Text" from the Add Credentials pop-up box).

By default, your GitHub will pull from your master unless you replace the "Branch to build" with a different branch name.


**Add Credentials**

Kind

Scope

Secret

Description

**Source Code Management**

☐ None  
☐ CVS  
☐ CVS Projectset  
☒ Git Repositories

Repository URL

Credentials

Name

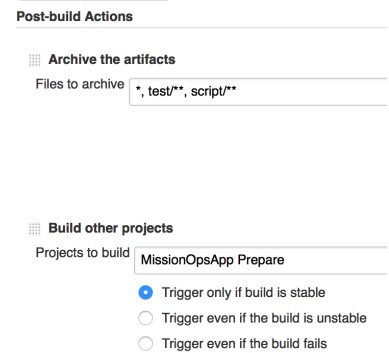
Refspec

Branches to build

Branch Specifier (blank = '\*/')

Under the section Post-build actions, add 2 steps:

1. Archive the artifacts - this step will retain a copy of your previous build step (which gets a copy of GitHub codes in the workspace temporarily)
2. Build other projects - this will start the next task (#2 Prepare)



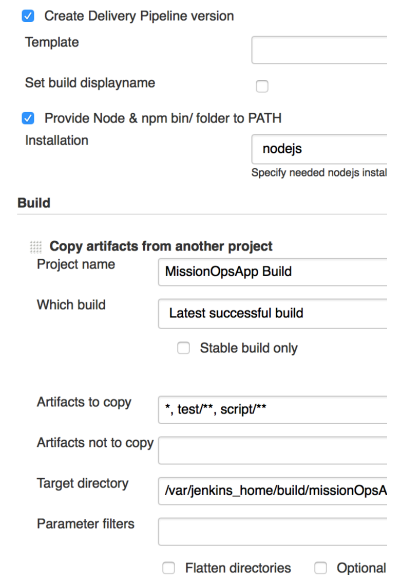
The screenshot shows the 'Post-build Actions' section in Jenkins. It contains two steps: 'Archive the artifacts' and 'Build other projects'. The 'Archive the artifacts' step has 'Files to archive' set to '\*.test/\*\*, script/\*\*'. The 'Build other projects' step has 'Projects to build' set to 'MissionOpsApp Prepare'. It also has three radio button options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'.

From Jenkins home page, click New Item.

Edit description to describe this step for pulling code changes from GitHub to copy the GitHub codes to the target shared folder.

Create build steps:

- 4 Create #2 Prepare
- Check "Create Delivery Pipeline version" (the status will show up in Delivery pipeline view).
  - Under Build section, select Copy artifacts from another project. Choose the project name from step #1 (e.g. MissionOpsApp Build). Specify which files you want to copy, and the target folder.



The screenshot shows the 'Build' section in Jenkins. It has a checkbox 'Create Delivery Pipeline version' which is checked. Below it is a 'Template' field. There is a 'Set build displayname' checkbox which is unchecked. Another checkbox 'Provide Node & npm bin/ folder to PATH' is checked. Below it is an 'Installation' field with 'nodejs' selected. There is a link 'Specify needed nodejs instal'. Below the 'Build' section is the 'Copy artifacts from another project' section. It has a 'Project name' field with 'MissionOpsApp Build' selected. There is a 'Which build' field with 'Latest successful build' selected and a checkbox 'Stable build only' which is unchecked. There is an 'Artifacts to copy' field with '\*.test/\*\*, script/\*\*' entered. There is an 'Artifacts not to copy' field which is empty. There is a 'Target directory' field with '/var/jenkins\_home/build/missionOpsA' entered. There is a 'Parameter filters' field which is empty. At the bottom are two checkboxes: 'Flatten directories' and 'Optional', both of which are unchecked.

In this example, the folder /var/jenkins\_home/build/missionOpsApp is a shared folder, which is or will be used by production Web server.

From Jenkins home page, click New Item.

- 5 Create #3 Test
- Edit description to describe this step for executing test automation scripts.
- Create build steps:

- Execute shell to run test scripts, e.g. ./test.sh.

6

## Create Build Pipeline View

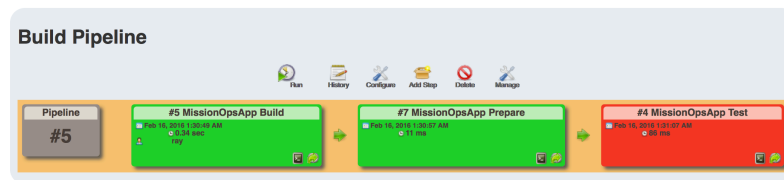
From Jenkins home page, click the "+" icon next to the tab All.

Create a build pipeline view by specifying the name, and check "Build Pipeline View".


Add description for your build pipeline.

Under the item "Layout" | Select Initial Job, select your tasks or jobs. you can add more jobs later.

Your build pipeline should look something like:





## Open Items




# Jenkins

Jenkins ▶

 New Item

 People

 Build History

View name

☐ Build Pipeline View

Shows the jobs in a build pipeline view.

☐ Delivery Pipeline View

Shows one or more delivery pipeline ins

☐ List View

Shows items in a simple list format. You

☐ My View

This view automatically displays all the j

OK

Name	
Description	
Filter build queue	<input type="checkbox"/>
Filter build executors	<input type="checkbox"/>
Build Pipeline View Title	
Layout	<div><div>B</div><div>S</div></div>
No Of Displayed Builds	<div><div>1</div></div>
Restrict triggers to most recent successful builds	<input type="checkbox"/>
Always allow manual trigger on pipeline steps	<input type="checkbox"/>
Show pipeline project headers	<input type="checkbox"/>
Show pipeline parameters in project headers	<input type="checkbox"/>
Show pipeline parameters in revision box	<input type="checkbox"/>
Refresh frequency (in seconds)	<div><div>3</div></div>

The following issues are either "followup tasks" or "open questions" to be addressed. They may not be any immediate issues.

- When Jenkins build software apps after pulling from GitHub, it will write to a shared folder. For demo, we set up the shared folder /mnt/data01 to be world writeable. This is NOT appropriate for production use. Followup actions are:
  - Create docker containers with appropriate R/W access rights for the shared folder
  - We may re-configure the docker command to be non-root, and make the user docker to have write/read access to the shared folder only
- Next step is to set up nginx as reverse proxy, and haproxy as load balancer.