

[illegible]

| | |
|-------------|---|
| Name | Nginx Software |
| URL | https://www.attackdefense.com/challengedetails?cid=2041 |
| Type | DevOps Basics: Continuous Integration |

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

The continuous Integration development process dictates that developers push the code into the master development repository frequently and each (or a small number of) code pushes can trigger the automated build, tests, and deploy the latest build on the test server.

[Jenkins](#) is an open-source automation server that is used widely for continuous integration.

A Jenkins instance and a Gitlab instance are provided to the user. The source code of Nginx is stored on the Gitlab instance.

Objective: Create a Jenkins job to configure and build the Nginx binary!

Instructions:

- The GitLab server is reachable with the name 'gitlab'
- Gitlab credentials:

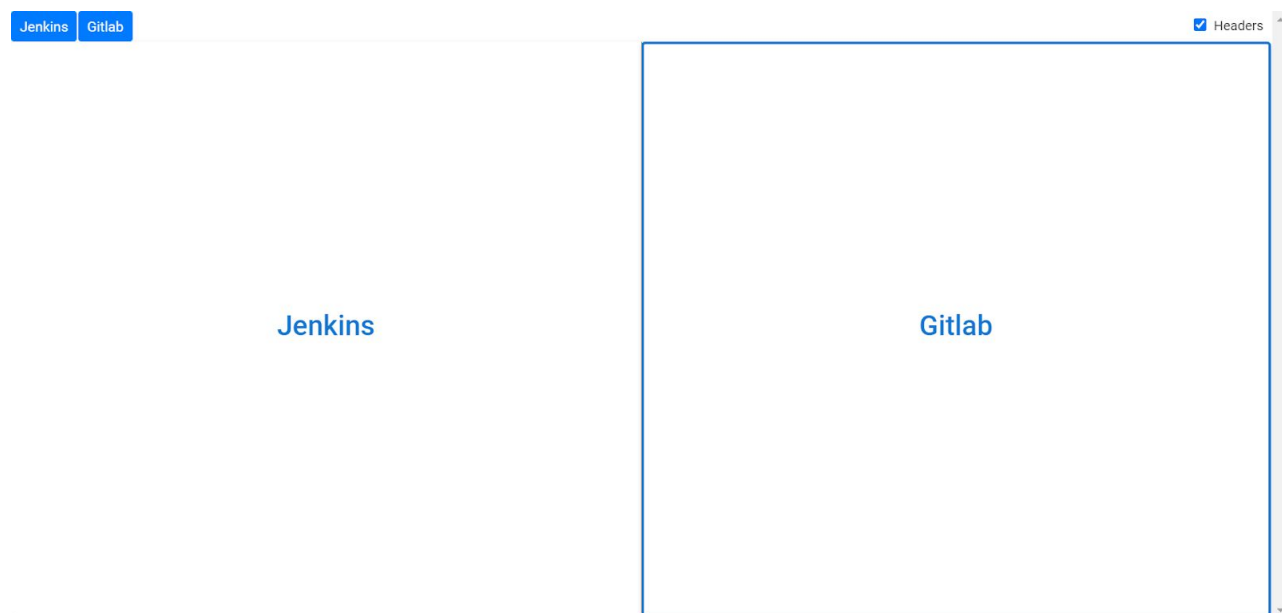
| Username | Password |
|-----------------|-----------------|
| root | welcome123 |

- The Jenkins server is reachable with the name 'jenkins'
- Jenkins credentials:

| Username | Password |
|----------|------------|
| admin | welcome123 |

Lab Setup

On starting the lab, the following interface will be accessible to the user.



On choosing (clicking the text in the center) left left panel, **Jenkins web UI** will open in a new tab



Welcome to Jenkins!

Sign in

☐ Keep me signed in

On selecting the right panel, a web UI of **Gitlab** will open in a new tab.



GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

| Sign in | Register |
|--------------------------------------|---------------------------------------|
| Username or email | |
| <input type="text"/> | |
| Password | |
| <input type="password"/> | |
| <input type="checkbox"/> Remember me | Forgot your password? |
| Sign in | |

The GitLab instance takes time to function properly. Hence, for some time the following wait page might be visible. This page reloads automatically.

PENTESTER ACADEMY

WebApp will appear once deployed!

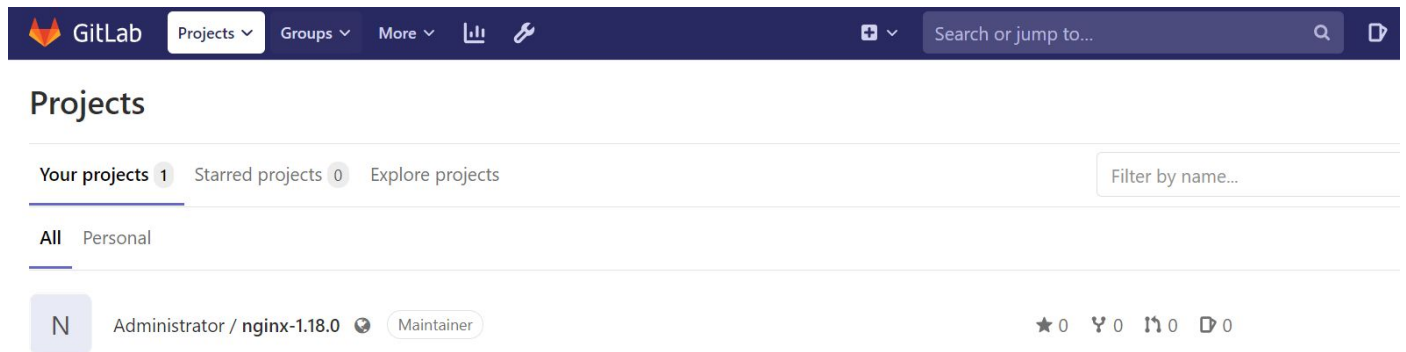
This page refreshes automatically.

Solution

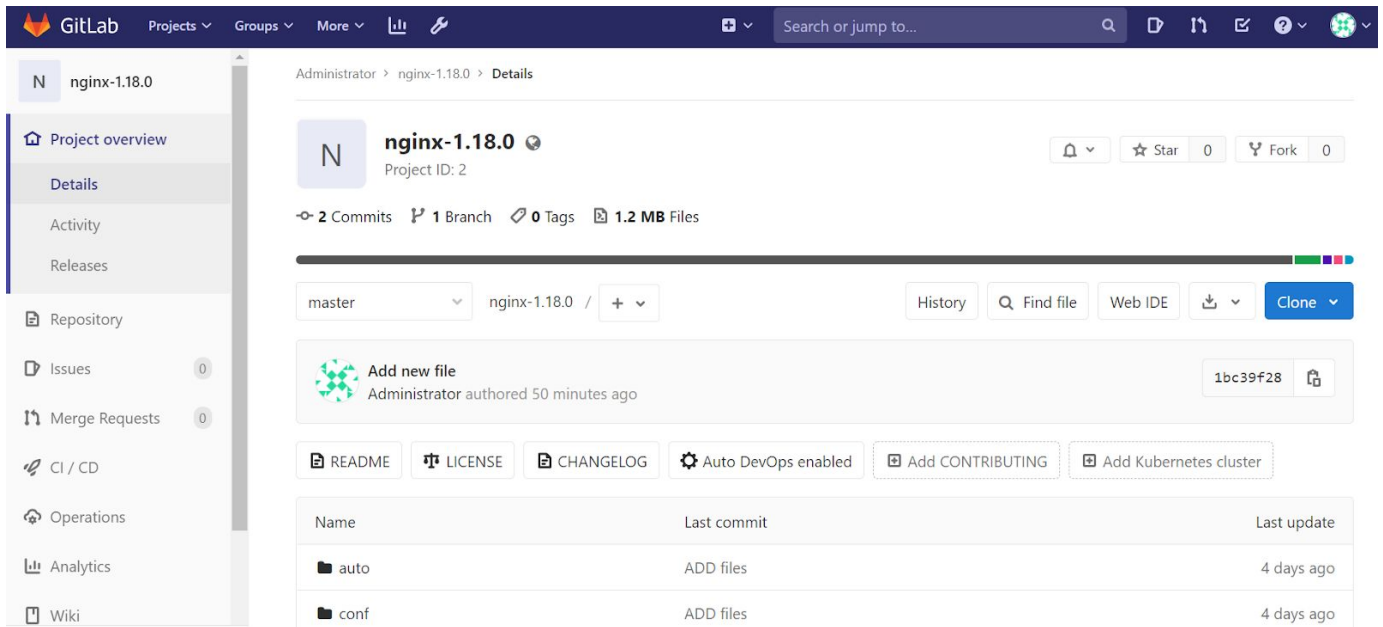
Step 1: Login into GitLab instance using the provided credentials:

Username: root

Password: welcome123



There is a repository present in the Administrator's account. Click on the repository link to open the repository page.

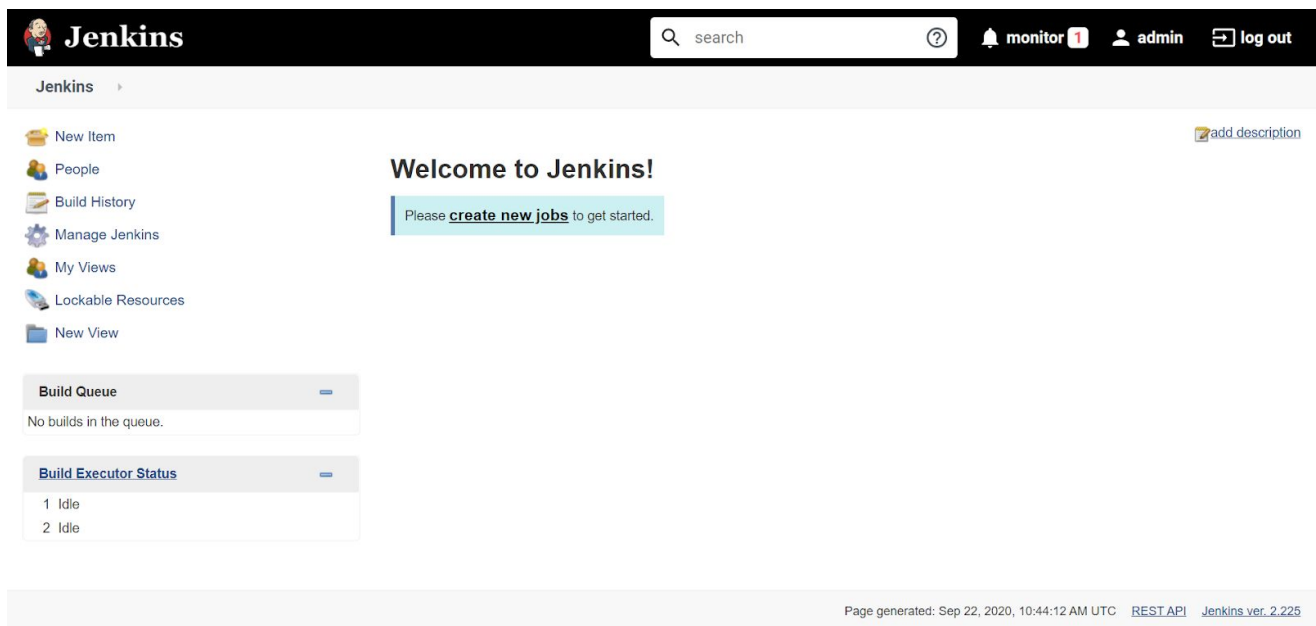


The image shows the GitLab web interface for a project named 'nginx-1.18.0'. The left sidebar contains navigation links: Project overview, Details (selected), Activity, Releases, Repository, Issues (0), Merge Requests (0), CI / CD, Operations, Analytics, and Wiki. The main content area shows the project details, including the project ID (2), 2 commits, 1 branch, 0 tags, and 1.2 MB of files. A 'master' branch is selected. Below this, there's a section for 'Add new file' with a commit hash '1bc39f28'. A table lists files: 'auto' and 'conf', both with 'ADD files' as the last commit and '4 days ago' as the last update. At the bottom, there are buttons for README, LICENSE, CHANGELOG, and options to enable Auto DevOps, add a CONTRIBUTING file, or add a Kubernetes cluster.

Step 2: Login into Jenkins instance using the provided credentials:

Username: admin

Password: welcome123

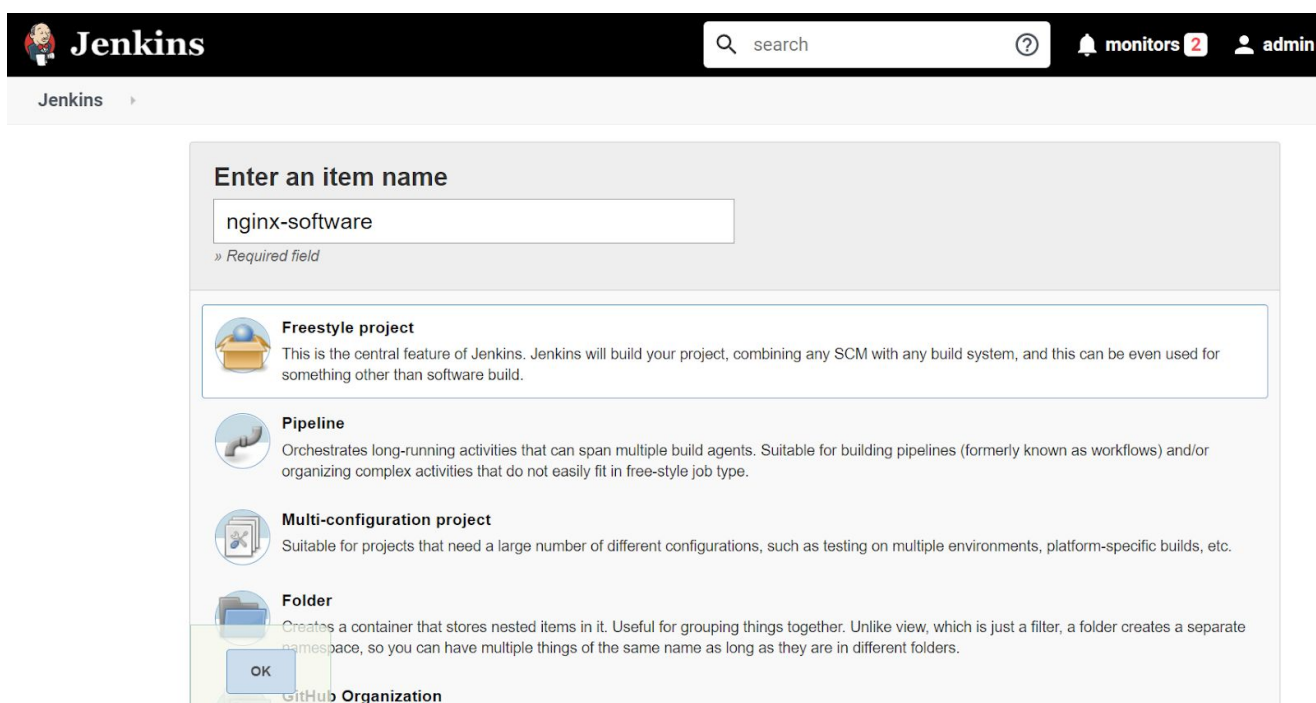


The image shows the Jenkins dashboard. The top navigation bar includes the Jenkins logo, a search bar, a monitor icon with a red notification, a user icon labeled 'admin', and a 'log out' button. The main content area features a 'Welcome to Jenkins!' message with a button to 'create new jobs'. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. Below this, there's a 'Build Queue' section showing 'No builds in the queue.' and a 'Build Executor Status' section showing two idle executors. The footer indicates the page was generated on Sep 22, 2020, at 10:44:12 AM UTC, with links to the REST API and Jenkins version 2.225.

There is no job in this Jenkins server.

Step 3: Click on the “create new jobs” link. A job creation wizard will run. Enter the name of the job on the first page. You are free to select any name you like. We are using “nginx-software”.

Then select the “Freestyle project” option and click the “OK” button.



Jenkins search ? monitors 2 admin

Jenkins »

Enter an item name

nginx-software

» 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.

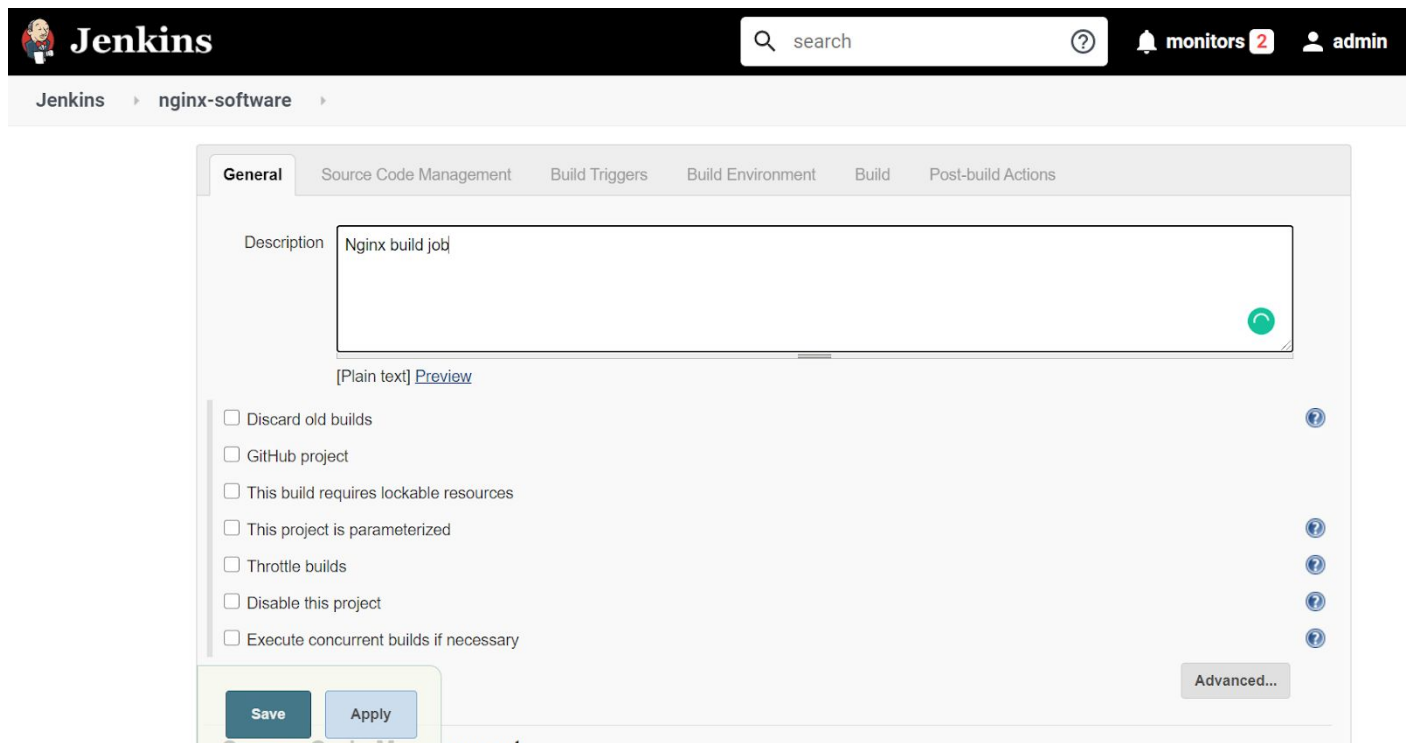
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

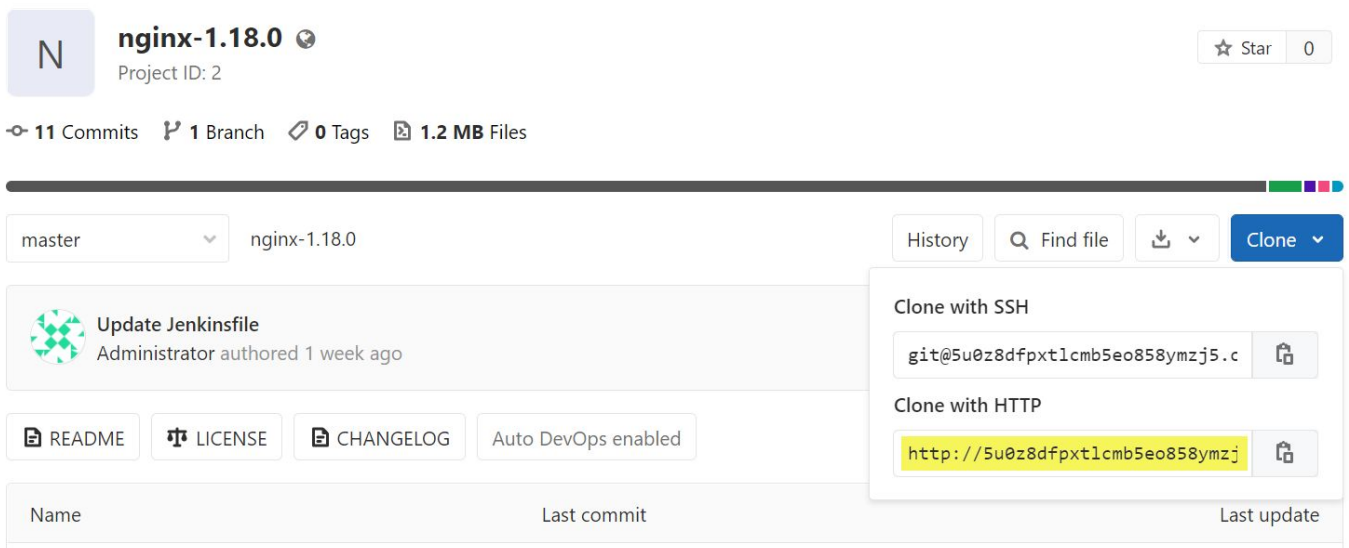
GitHub Organization

On the next page, enter some description of the job.



The image shows the Jenkins web interface. At the top is a black header with the Jenkins logo, a search bar, and a user profile labeled 'admin'. Below the header is a breadcrumb trail: 'Jenkins > nginx-software'. The main content area is titled 'General' and contains a 'Description' text box with the text 'Nginx build job'. Below the text box are several checkboxes: 'Discard old builds', 'GitHub project', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', and 'Execute concurrent builds if necessary'. At the bottom of the 'General' tab are 'Save' and 'Apply' buttons. To the right of the 'General' tab are other tabs: 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. An 'Advanced...' button is located at the bottom right of the 'General' tab.

Step 4: Open the Nginx project repository page on GitLab and copy the “Clone with HTTP” path.



The image shows the GitLab repository page for 'nginx-1.18.0'. The repository is owned by 'Administrator' and has a 'Project ID: 2'. It shows 11 commits, 1 branch, 0 tags, and 1.2 MB of files. The 'master' branch is selected, and the file 'nginx-1.18.0' is shown. A dropdown menu is open, showing the 'Clone with HTTP' path: 'http://5u0z8dfpxt1cmb5eo858ymzj5.c'. The 'Clone with SSH' path is also visible: 'git@5u0z8dfpxt1cmb5eo858ymzj5.c'. The 'Clone' button is highlighted in blue.

Step 5: Select “Git” in the Source Code Management section and paste this path in the “Repository URL” field.

The copied URL will look something like this:
`http://<random_server_URL>/root/nginx-1.18.0.git`

This URL is to make sure that the GitLab files can be accessed through web UI. However, to clone it to the Jenkins server, change it to:

`http://gitlab/root/nginx-1.18.0.git`

The screenshot shows the Jenkins configuration page for a job, specifically the 'Source Code Management' tab. The 'General' tab is also visible. Under 'Source Code Management', the 'Git' radio button is selected. The 'Repositories' section contains a 'Repository URL' field with the value 'http://gitlab/root/nginx-1.18.0.git' and a 'Credentials' dropdown menu set to '- none -'. There are 'Advanced...' and 'Add Repository' buttons. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value '*/master' and an 'Add Branch' button. The 'Repository browser' dropdown is set to '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button. A 'Save' button is highlighted with a green box.

Step 6: In the build section, select “Execute shell”. Run configure and make to build the Nginx binary. Also, add a directory listing command of the output directory.

```
./configure
make
ls -l objs
```

Build

Execute shell

Command

```
./configure  
make  
ls -l objs
```

See [the list of available environment variables](#)

Advanced...

Add build step ▾

Click on the save button and the job will be saved then the page will redirect to the Job page.

Step 7: Click “Build now” from the left-hand menu to fire the job. An in-progress will appear under the “Build History”.

The screenshot shows the Jenkins web interface. At the top is the Jenkins logo and a search bar. Below the header, the breadcrumb path is 'Jenkins > nginx-software >'. On the left sidebar, there is a menu with icons and labels: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'Rename'. The main content area is titled 'Project nginx-software' with the subtitle 'Nginx build job'. Below this, there are two links: 'Workspace' (with a folder icon) and 'Recent Changes' (with a notepad icon). Further down is a section titled 'Permalinks'. At the bottom left, there is a 'Build History' section with a 'trend' link. It contains a search box with the text 'find' and a list of builds. The first build is labeled '#1' and shows a status bar with a blue and white pattern, indicating it is in progress. The timestamp for this build is 'Sep 22, 2020, 3:41 PM'.

Click on the number #1 to visit the build page.

Jenkins > nginx-software >

 [Back to Dashboard](#)

 [Status](#)

 [Changes](#)

 [Workspace](#)

 [Build Now](#)

 [Delete Project](#)

 [Configure](#)

 [Rename](#)

Project nginx-software

Nginx build job



[Workspace](#)




[Recent Changes](#)

Permalinks


Step 8: Click on “Console Output” to view the build logs.


Jenkins > nginx-software > #1

 [Back to Project](#)

 [Status](#)

 [Changes](#)

 [Console Output](#)

 [View as plain text](#)

 [Edit Build Information](#)

 [Git Build Data](#)

 [No Tags](#)



Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/nginx-software
No credentials specified
Cloning the remote Git repository
Cloning repository http://gitlab/root/nginx-1.18.0.git
> git init /var/lib/jenkins/workspace/nginx-software # timeout=10
Fetching upstream changes from http://gitlab/root/nginx-1.18.0.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- http://gitlab/root/nginx-1.18.0.git
timeout=10
```

```
+ ls -l objs
total 5376
-rw-r--r-- 1 jenkins jenkins 40144 Sep 22 16:00 Makefile
-rw-r--r-- 1 jenkins jenkins 17494 Sep 22 16:00 autoconf.err
-rwxr-xr-x 1 jenkins jenkins 5357584 Sep 22 16:01 nginx
-rw-r--r-- 1 jenkins jenkins 5375 Sep 22 16:01 nginx.8
-rw-r--r-- 1 jenkins jenkins 7004 Sep 22 16:00 ngx_auto_config.h
-rw-r--r-- 1 jenkins jenkins 657 Sep 22 16:00 ngx_auto_headers.h
-rw-r--r-- 1 jenkins jenkins 5856 Sep 22 16:00 ngx_modules.c
-rw-r--r-- 1 jenkins jenkins 45608 Sep 22 16:01 ngx_modules.o
drwxr-xr-x 9 jenkins jenkins 4096 Sep 22 15:58 src
Finished: SUCCESS
```

The job completed successfully and the Nginx binary is available for deployment.

Learning

Building and packaging the Nginx web server using Jenkins.