Versioning and tagging on source code and the artifacts using github and nexus.

We need to maintain different versions on source code to avoid unwanted changes and bugs in code and also we required to maintain versioning on artifact to quick revert back to the old working version.

Why artifact versioning if we already have source code versioning.

1. For quick revert back
2. For avoid build process multiple times
3. For version upgrade

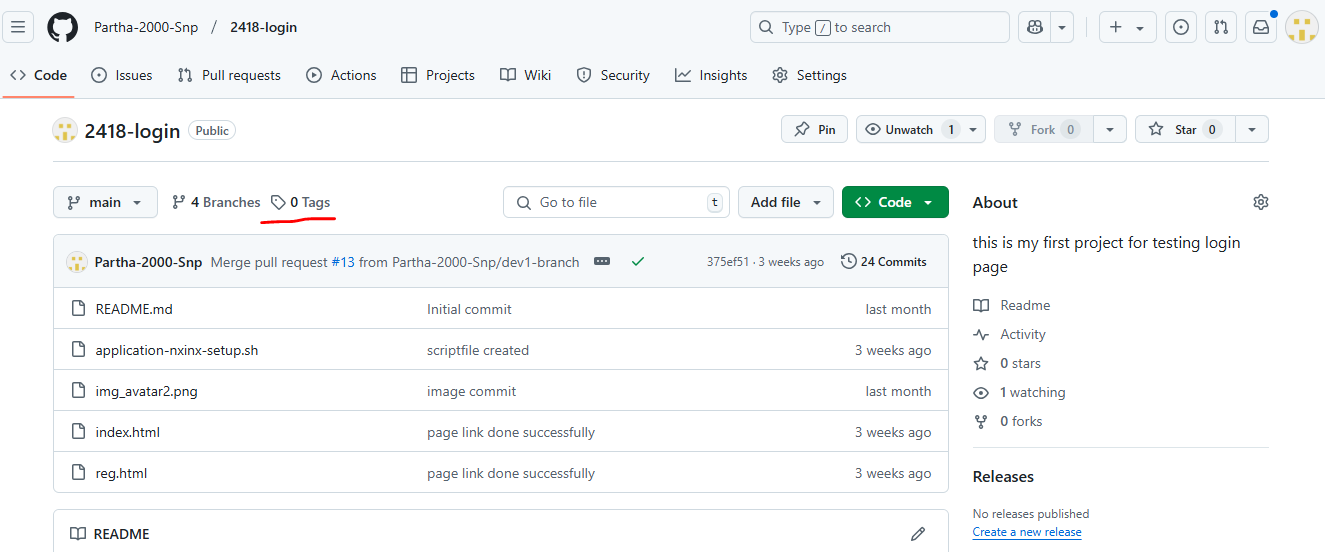
For code we can maintain version control from github.

This basically takes snapshot of targeted branch code and create a zip and tar format. Moving forward no more changes can be done on that only we can download that and use the code.

We will have a big advantage. As a developer we can just share the zip or tar file links to the devops team they can directly download it to their system and use it need to invite or clone process.

Step 1-

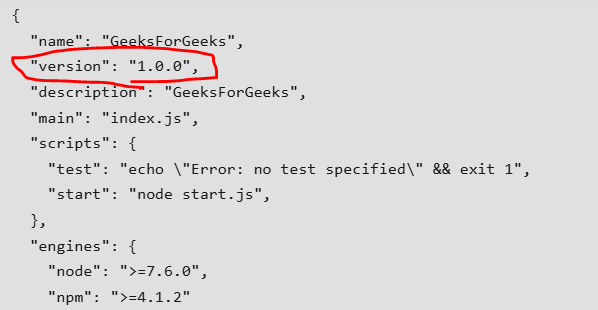
Go to git hub.



Here go for tags.

Note- for actual production application it will contains the metadata file for the project such as package.json. .pom files etc..

For those we can define the version on those metadata files for more visibility.

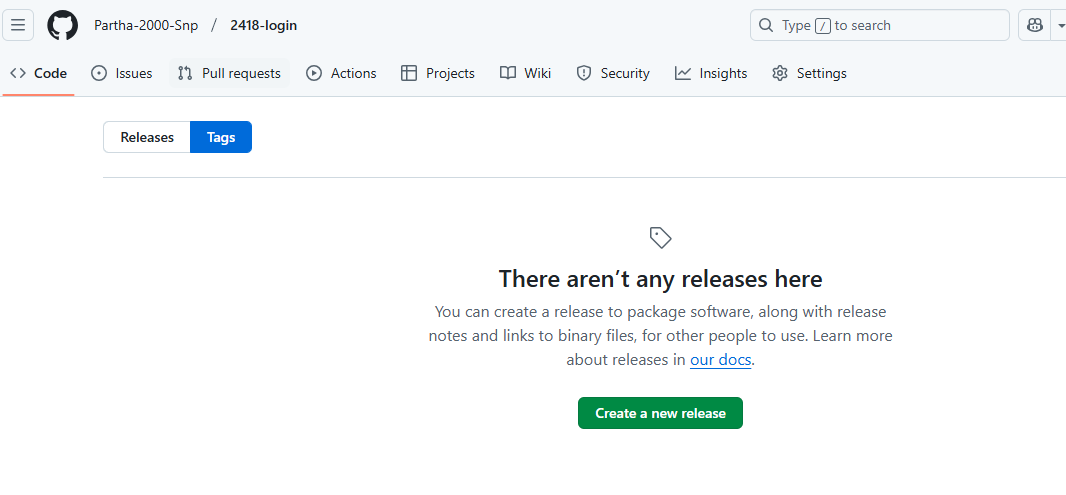


Step 2-

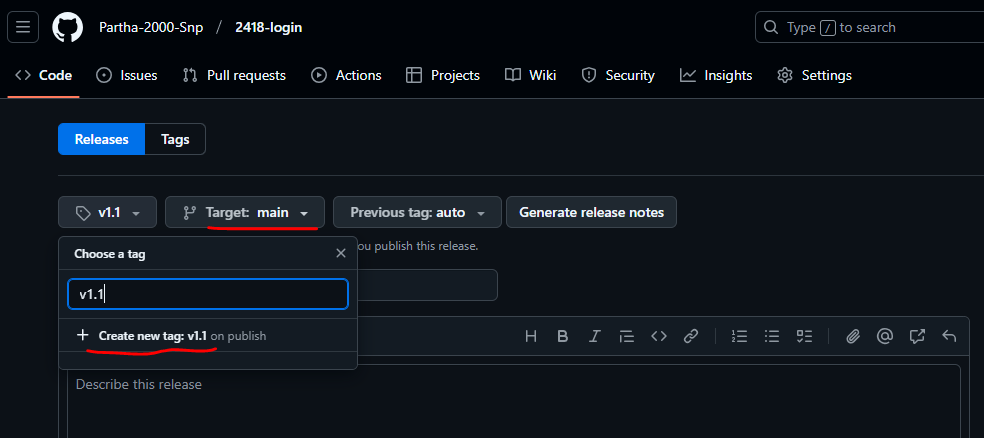
Go to targeted repository where do we want to assign tag.

Go for tag option.

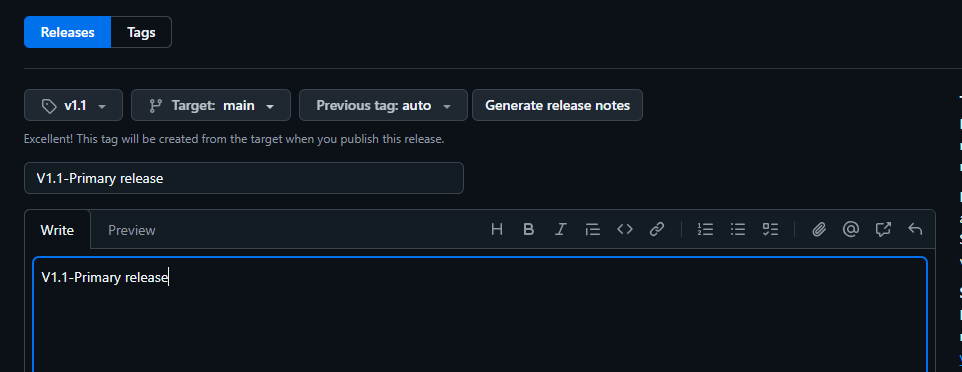
Go for create new tags.



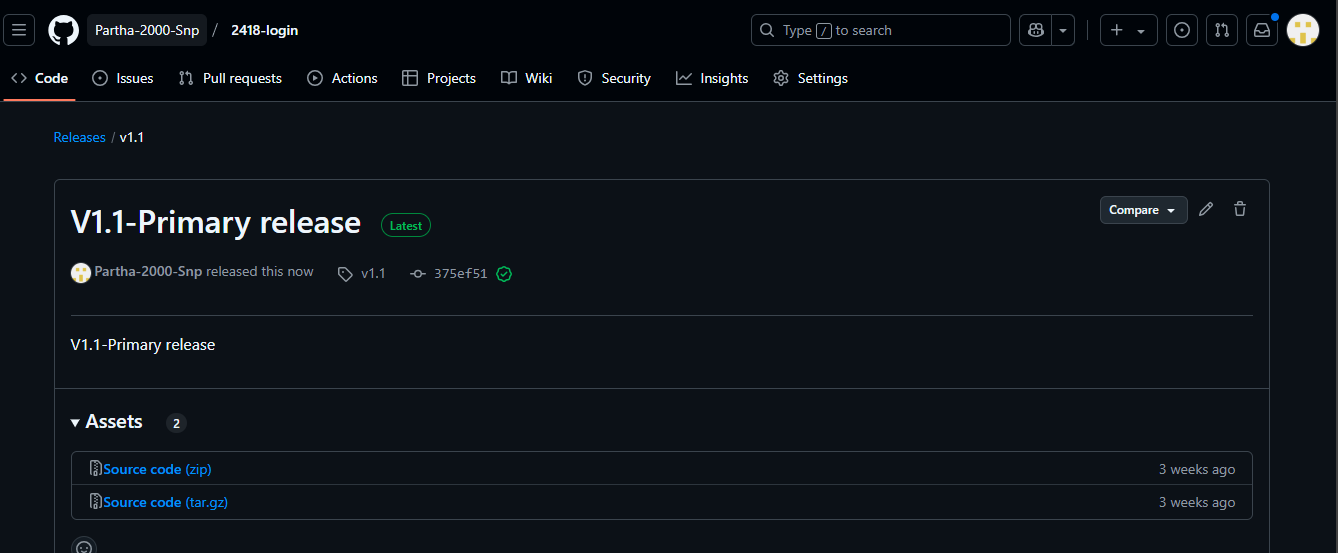
Type the tag name and choose to create and select branch.



Write the version name and the description about the version.



Go ahead and publish the release. Zip and tar format.



If you wat to deploy this version of the application we will provide those zip or tar file link to devops engineer to proceed further. So they can download it using “wget link” and work.

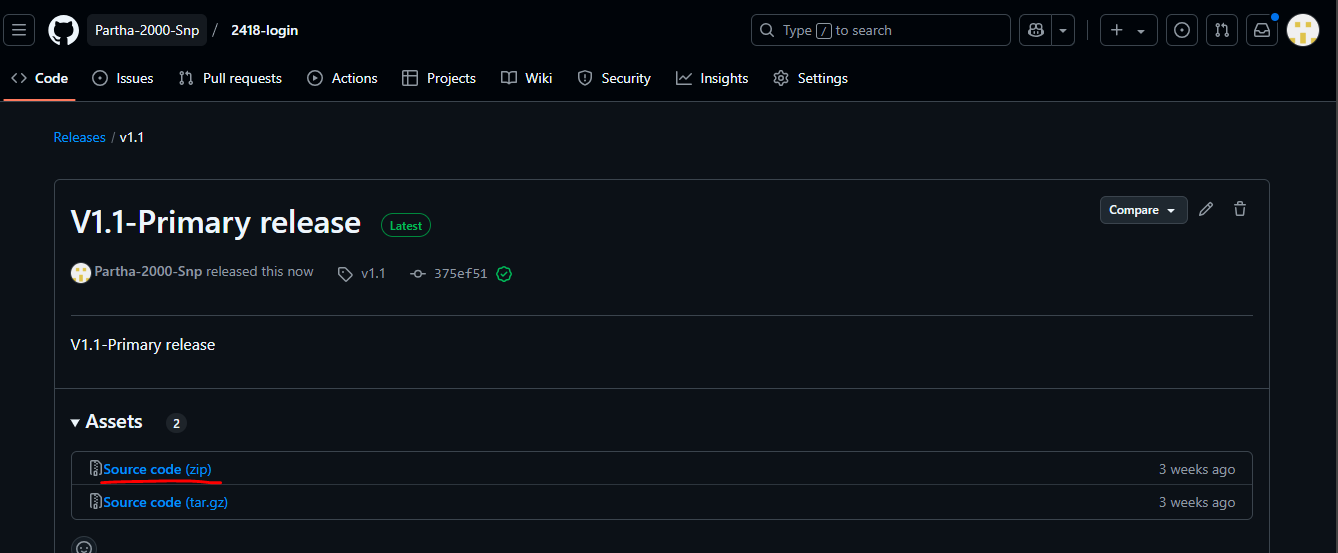
From here the devops team will get the code on build server and build the code to generate the artifact. Store the artifact to artifact library nexus, azure or any other third party.

From developer end again the changes done on code we will test it and assign another version to the code.

Let’s demonstrate.

We are providing the new version 1.1 zip link to developer

Right click on zip and copy the address.

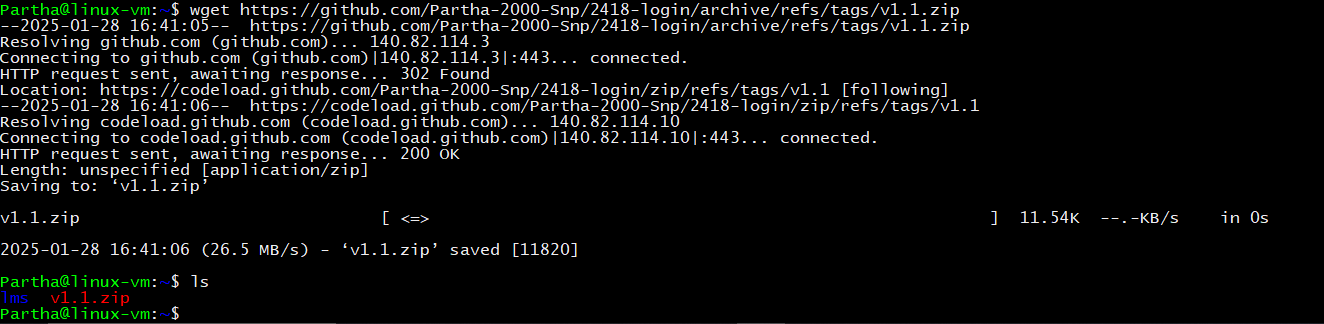


On devops side

>> wget link

Ex- wget <https://github.com/Partha-2000-Snp/2418-login/archive/refs/tags/v1.1.zip>

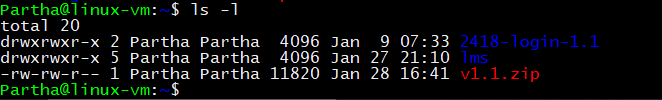
File will be downloaded



Let’s unzip it and deploy.

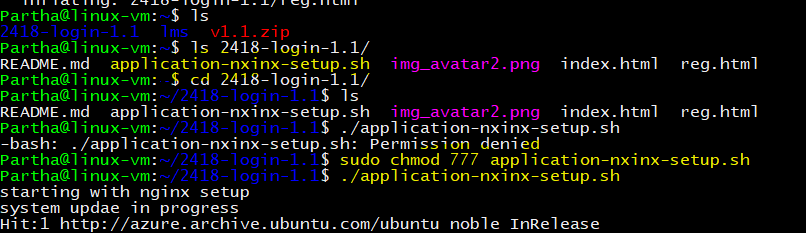
>> unzip zipped-file-name

Ex- unzip v1.1.zip

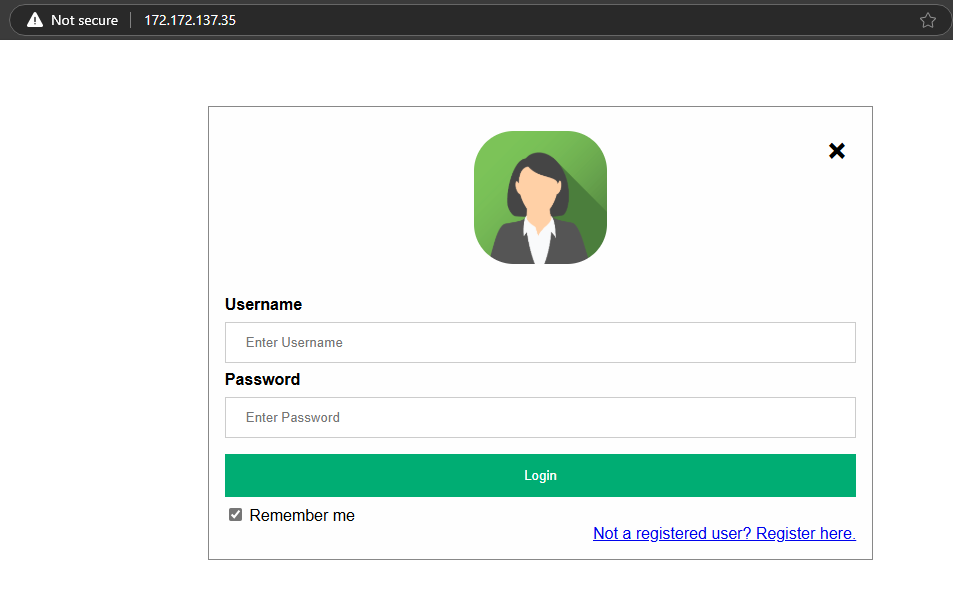


Let’s deploy this to nginx and test.

Go to unzipped folder and check for files.

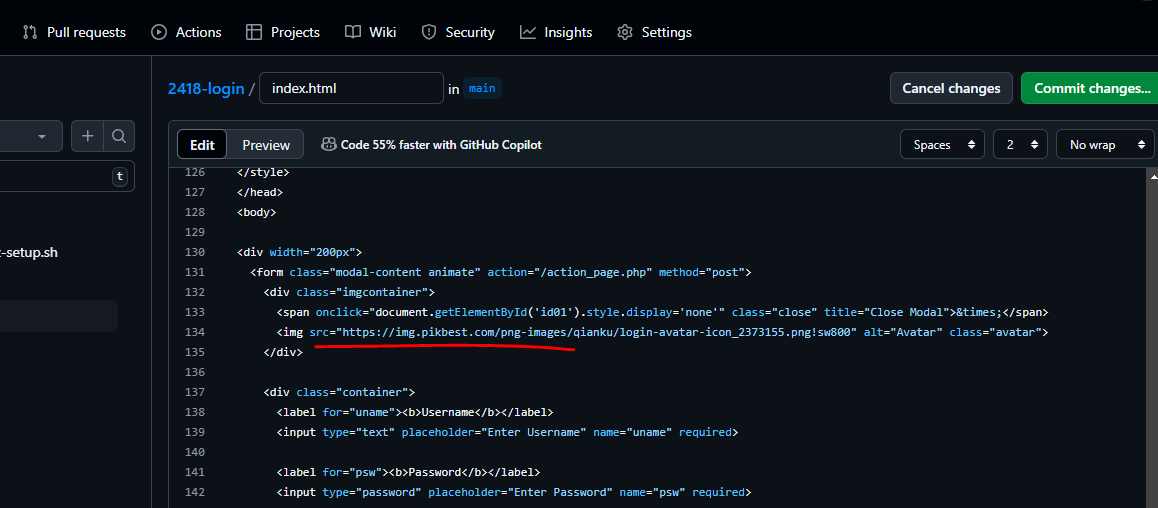


In my case I already had script file that can do all setup for application on nginx so I just run that and application setup was done.



Let’s do some other changes and go for next version. V-1.2

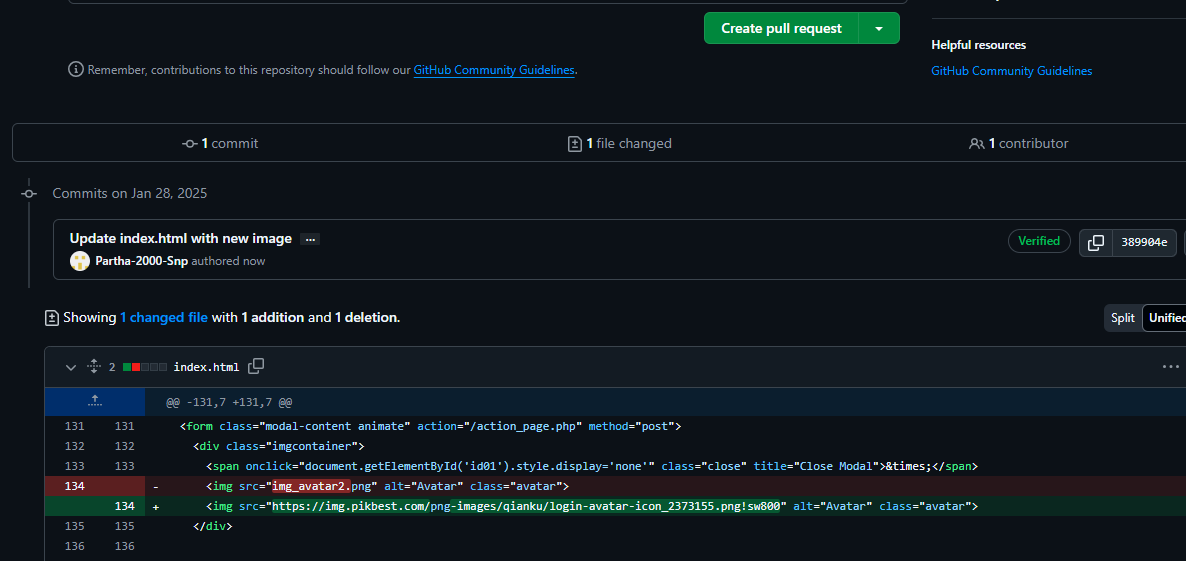
Updated the login page image.



Let’s commit it and take new version.

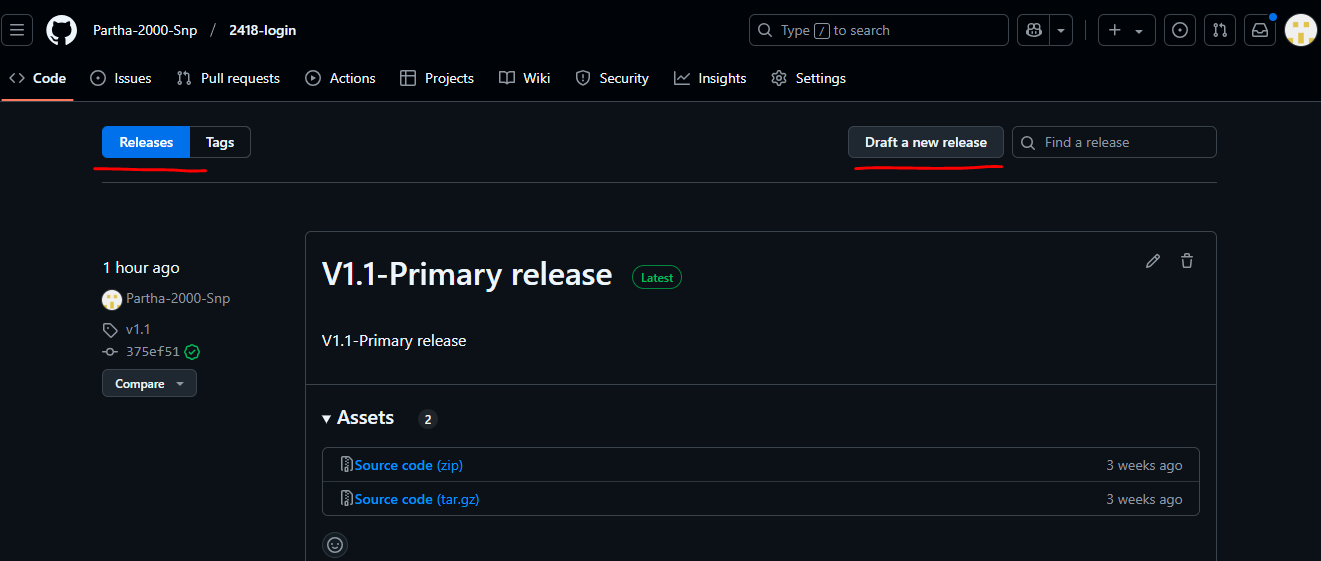
Committing to a new branch name as devops-branch.

Not going to have pull request and marge to main branch.

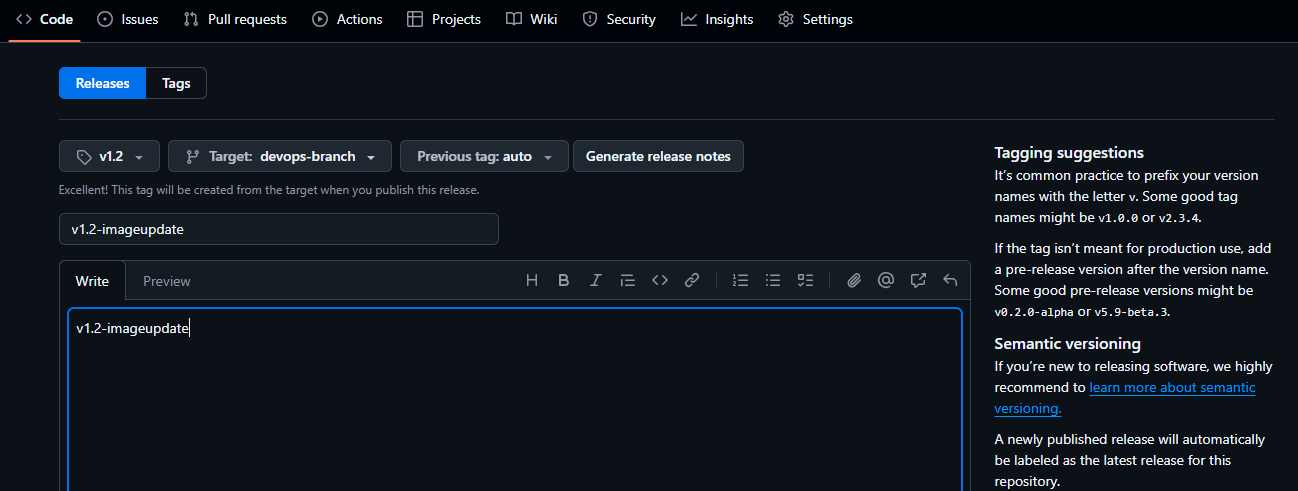


As we have flexibility to have tag on any branch I will assign tag on this changed branch directly.

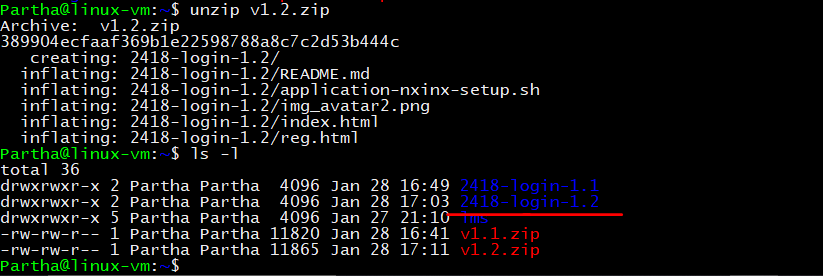
Go for tag and release then draft a new release.



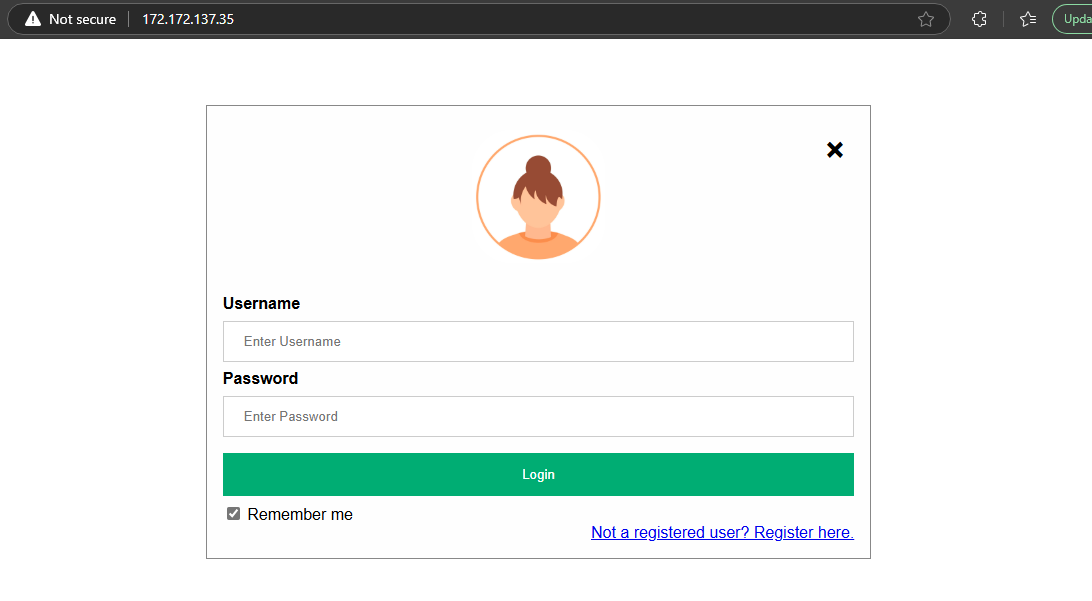
Here, we will select branch and name for the version.



Get this version 1.2 to server and test.



Run version 1.2



If you did not like it can revert back to previous version code easily as that code is not getting corrupted because by tagging we are taking snapshot of it and storing it safely.

In real time code will not be deployed directly as code because framework code need to build first to convert to binary artifact to run it properly.

Here in same way we will take LMS application assign tag for it on code take different versions to build server build it and run it.

Here we do need to store the build artifacts as version wise for easy use release update and quick revert back we will use nexus to do those all.

**Why Build is Needed**

1. **HTML/CSS/JavaScript**:
   * These applications are static.
   * No preprocessing is required because browsers can directly interpret HTML, CSS, and JavaScript files.
2. **Node.js Applications**:
   * Involves server-side code written in JavaScript (or TypeScript).
   * May need dependencies installed using npm install before running the application.
   * Example build steps:
     + Bundling assets (e.g., Webpack).
     + Transpiling ES6+ code or TypeScript to JavaScript.
     + Optimizing assets for production.
3. **Angular Applications**:
   * Involves TypeScript, SCSS, and other advanced tools.
   * Needs a build step to:
     + Compile TypeScript into JavaScript.
     + Minify JavaScript and CSS.
     + Bundle the application into files the browser can use.

**Basic Build Steps**

**1. Node.js Applications**

* **Install Dependencies**:

bash

CopyEdit

npm install

* **Build for Production** (if applicable):

bash

CopyEdit

npm run build

* **Start the Application**:

bash

CopyEdit

npm start

**2. Angular Applications**

* **Install Dependencies**:

bash

CopyEdit

npm install

* **Build the Application**:

bash

CopyEdit

ng build --prod

* This creates an optimized dist/ folder with all static files for deployment.

**3. React Applications (Example)**

* **Install Dependencies**:

bash

CopyEdit

npm install

* **Build the Application**:

bash

CopyEdit

npm run build

* This generates a build/ folder with all deployable files.

**Deploying After Build**

* After building the application, you deploy the generated folder (e.g., dist/ for Angular or build/ for React) to a web server like:
  + **Apache/Nginx** (for static files).
  + **Node.js runtime** (for server-side apps).
  + **Cloud platforms** (e.g., Azure, AWS, Netlify, Vercel).

If you want, I can guide you through a specific app setup or deployment process!

Iet’s go ahead and build different versions of LNS.

A screenshot of a computer

Description automatically generated

Version 1.1. let’s take it to build server and build it and run. We have backend fixed. So, it must run with out any issue.

A close-up of a computer

Description automatically generated

Lets. Go for another version of test with yellow BG.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Same way version 3.1 for uat.

Here if we want we can also compare 2 different versions.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with numbers and letters

Description automatically generated

A screenshot of a computer

Description automatically generated

Now we have all different versions tag applied on code and also 3 different versions of build artifact.

We need to upload those artifacts to a repository to deploy on different environments like prod, test and uat.

Nexus- is a software use to store and use artifacts.

can we use nexus software without installing it on a server like github, because in that case if the server is crashed then nexus will not be accessible for getting artifacts?

* + No, you cannot use Nexus software without installing it on a server; it is designed to run on a dedicated server and requires installation to function as a repository manager for storing and distributing artifacts, meaning if the server crashes, you won't be able to access Nexus to retrieve artifacts.

Solution- Looks for different artifacts management repository such as azure artifact repository, ACR etc..

To use nexus we must install nexus on a server.

Manual setup Process - <https://www.fosstechnix.com/how-to-install-nexus-repository-on-ubuntu/>

This manual setup process is complex and required more time effort.

To avoid that we can take help of docker to install nexus.

It’s better Idea to have a different server for building the applications.

* Nexus Requires “t2.medium instance” i.e 4 GB RAM - 2 CPU’s on AWS, with 8 GB as Storage.
* Nexus works on “port 8081”, make sure to add port 8081 as part of your Security Group.

Step 1-

Verify the server has docker installed. To install sw using docker container we must have docker installed on the server.

A screenshot of a computer

Description automatically generated

If not already installed install doker.

>> curl -fsSL https://get.docker.com -o get-docker.sh && sh get-docker.sh

Once the docker is ready to use. Visit docker site and choose docker image to use.

A screenshot of a computer

Description automatically generated

Sonatype I am choosing because it’s the most popular one.

Create container by using docker image and run.

We can follow the docker page to get all details including run cmd etc..

Install docker.

>> sudo docker container run -dt --name nexus --restart=always -p 8081:8081 sonatype/nexus3

If any package issue update the server.

>> sudo apt update -y

A computer screen with white text

Description automatically generated



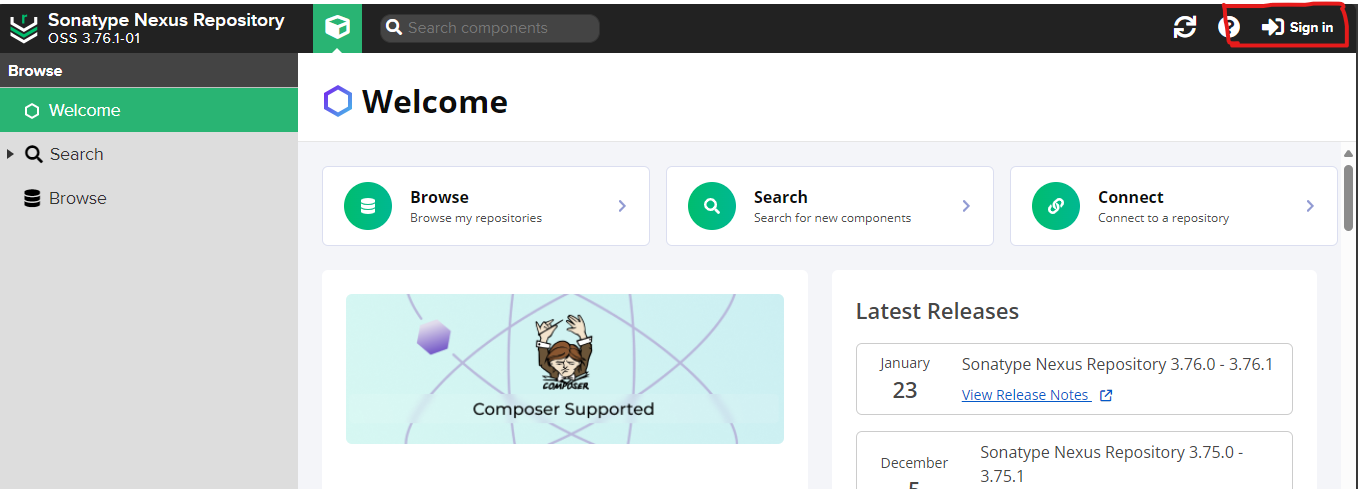
Enable nexus port “8081” to get the application on browser.

A screenshot of a computer screen

Description automatically generated

Step 2-

Go for sign in to nexus to use nexus.



Nexus the default username is “admin”

Th get default password run below cmd.

>> sudo docker container exec nexus cat /nexus-data/admin.password



Make sure you copied password correctly without taking username as shown in above screenshot.

Ex- 06ef0acc-1a43-497e-9ee2-4549aeb70a1f

Sign in and reset admin password.

A screenshot of a computer

Description automatically generated

Once you prompt the new password it will ask to have enable anonymous access or to disable it.

It’s exactly same as how we can keep our repository public or private.

We can keep it private for security reasons. So to upload or download artifact it will require user account and password.

A screenshot of a computer

Description automatically generated

Step 3-

Create repository.

Once you are done with setup you will be navigated to admin page. Go for administrator option and create repository.

A screenshot of a computer

Description automatically generated

Creation time can choose different. Raw can be used to upload any types. If you are not sure choose this.

“raw hosted”

Create repository.

A screenshot of a computer

Description automatically generated

Once created repository we will be able to browse it.

A screenshot of a computer

Description automatically generated

To upload artifacts, we can go to repository browse for the artifact file and upload it.

A screenshot of a browser

Description automatically generated

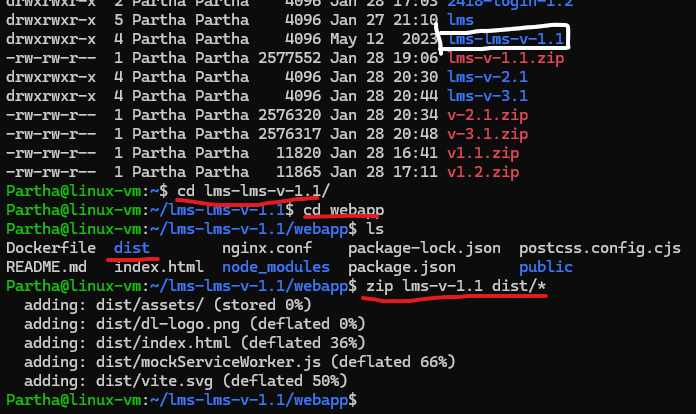
To upload from any other server (build server) or any other linux server we can do that by cmd.

>> curl -v -u username:password --upload-file <file> <nexus-repo-link>

To upload file it’s best idea to zip the artifact file and upload. To deploy on server we can download unzip and use. So, the file will not be corrupted.

Zip the artifact files

>> zip lms-v-1.1 dist/\*



Note- linux files does not depends on file extension. Such as .zip, .txt, .png etc..

But it’s good for identification and management so assign with extension.

>> zip lms-v-1.1.zip dist/\*

A black background with white text and blue and red letters

Description automatically generated

Upload zip to nexus repository.

>> curl -v -u username:password --upload-file <file> <nexus-repo-link>

Note- nexus repo link you will get from nexus repository.

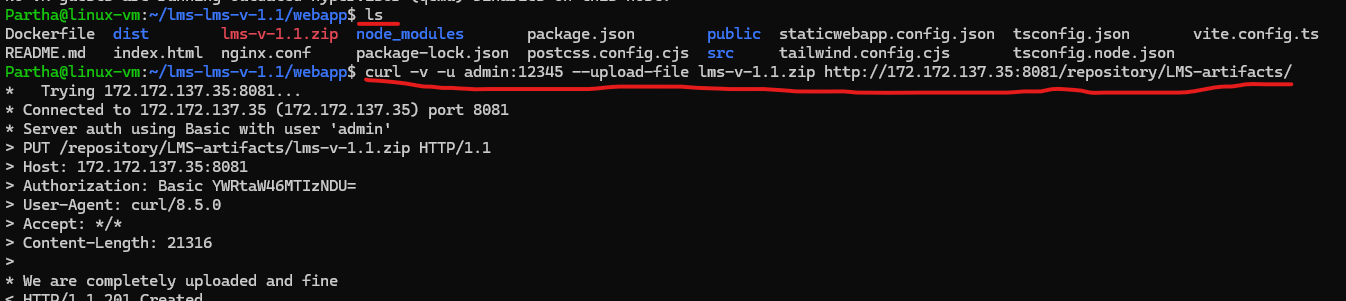
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Ex- curl -v -u admin:12345 --upload-file lms-v-1.1.zip <http://172.172.137.35:8081/repository/LMS-artifacts/>



A screenshot of a computer

Description automatically generated

Step 4-

Pull to server to deploy.

Right click, copy link.

A screenshot of a computer

Description automatically generated

Get on server-

>> wget nexus artifact zip link

A computer screen with white text

Description automatically generated

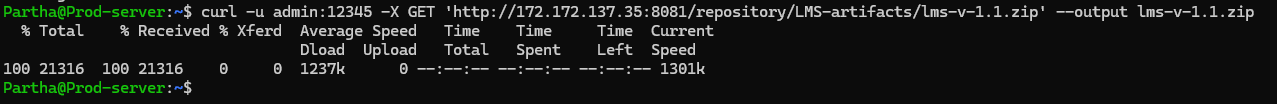
Error- Un authorized. This is due to we created private repo so authentication required during pulling artifact to the server.

>> curl -u username:password -X GET 'Nexus-repo-artifact-link' --output expected-file-name

Ex-

curl -u username:password -X GET 'http://20.172.187.108:8081/repository/lms/lms-1.1.zip' --output lms-1.1.zip

“--output lms-1.1.zip”- it defines which name the artifact will be. We can either use same name or different as atifact name.



Now unzip and deploy on server

A screen shot of a computer program

Description automatically generated

A computer with a pineapple in the background

Description automatically generated

We can store different versions on nexus repo and also the backend build files.

Same process will repeat once any new release happened.