GitLab is the software, is a web-based Git repository manager with wiki and issue tracking features, using an open source license.

- The code is written in Ruby.
- Developed for self-hosting.
- Issues and milestone tracking implemented.
- Support for attachment and code snippets.
- GitLab is an open-core product whereas GitHub and BitBucket are closed-source products.

Why GitLab? Features? Use? Comparison?

Features	GitLab	GitHub	BitBucket
Released	September 2011	April 2008	September 2008
Pricing	Unlimited public and private repositories / unlimited public and private collaborators	Free for public repositories / Paid plans for private repositories	Free for public repositories / Paid plans for private repositories
Code inspection	✓	✕	✕
Modification	✓	✕	✕
Attachments	✓ You can attach any file to any issue	✕	✕
Bug & issue tracking	✓	✓, but not that efficient	✓, but not that efficient
OS independent	✓	✕	✕
Protected branch (only masters can push to)	✓	✕ (only with paid plans)	✕
Build system	✓	✕ (only with 3rd party service)	✕ (only with 3rd party service)
Self-hosting	✓	✕ (only with enterprise plan)	✕
Back up	✓	✕	✕
Authentication Level	✓ Set permissions according to people's role, rather than either read or write access to a repository.	✕	✕
Detailed pricing	Free: gitlab.com / Free: GitLab Community Edition / $39 / Year: GitLab Enterprise	Free: public projects / $7/month: Personal plan / $25/month: organization plan / $2.500/year: Enterprise	Free: 5 users / $10/month: Basic Plan / $200/month : Premium plan

GitLab offers a continuous integration service. If you add a .gitlab-ci.yml file to the root directory of your repository, and configure your GitLab project to use a Runner, then each merge request or push triggers your CI pipeline.

Getting Started

There are two ways of getting started with GitLab Pages: either you fork an existing project, or you create a new one for yourself.
The key to having everything up and running as expected is the GitLab CI configuration file, called .gitlab-ci.yml. This file configures how your website will be built by a Runner. It is written in YAML, which has its own syntax which needs to be placed at your root directory.
The most important fact is that with GitLab CI:

- You take control over your builds.
- They won't be in an invisible black box where you don't know what is going on!
- You can actually see any build running live by navigating to your Pipelines > Builds > Build ID.
- You can also add any command to your script.

Before you push any .gitlab-ci.yml to your project, you can validate its syntax with the tool called CI Lint.

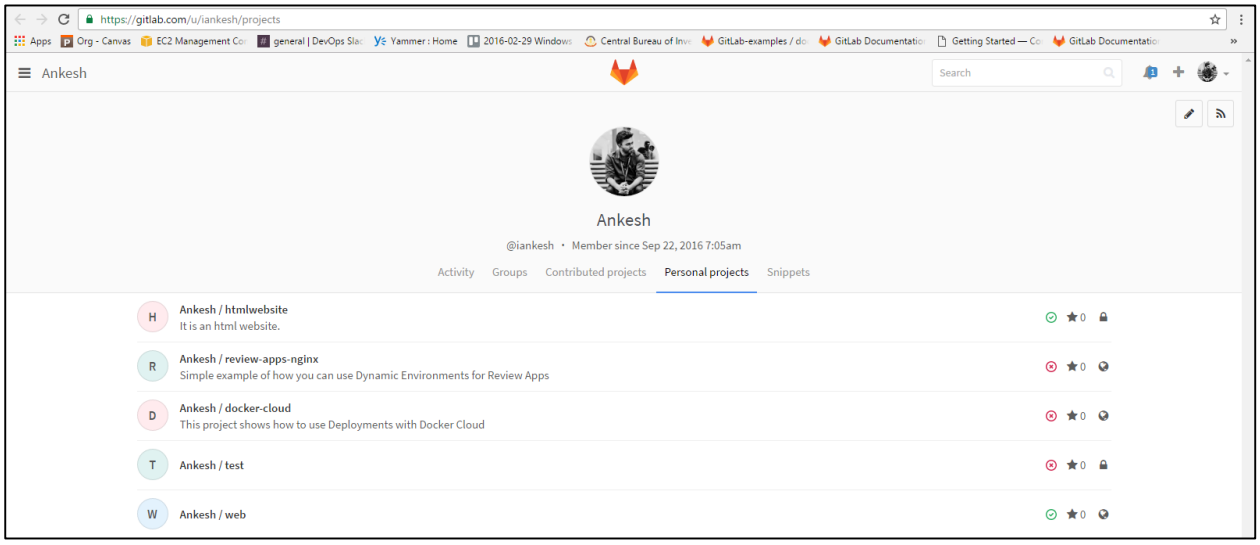
Creating new "HTML WEBSITE" PROJECT:

We will create a project which hosts a website. We can create the changes in the website in our local repository. And these changes are automatically be deployed to the server.
Here is an overview of the steps we'll take, assuming you already have your GitLab.com account:

1. Create a new project.
2. Clone your repository in your local server.
3. Add the configuration file (.gitlab-ci.yml).
4. Upload your website content.
5. Push all the files to your remote repository.
6. Add your custom domain (optional).
7. Build it with GitLab.
8. Check the website deployed in their server.

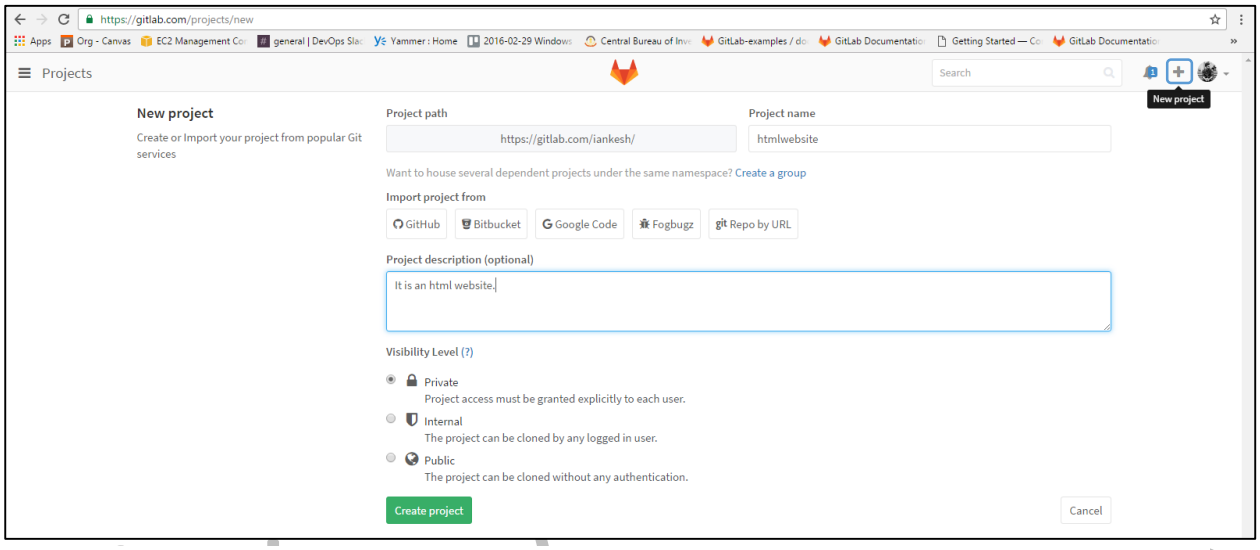
Start:

We will start by creating a GitLab account:



Step 1: Create a new project

1. On your dashboard you will see a big green button called **+** New Project. Click on it.
2. Set the first things up:
    - Project Name - your project's name
    - Privacy - choose if you want your project to be visible and accessible just for you (private), just for GitLab.com users (internal) or free to anyone to view, clone, fork and download it (public)



Step 2: Clone your repository in your local server

1. Git global setup

```
root@ip-172-31-16-128:~# git config --global user.name "Ankesh"
root@ip-172-31-16-128:~# git config --global user.email "iankesh@hotmail.com"
root@ip-172-31-16-128:~#
```

2. Create a new repository

```
root@ip-172-31-16-128:~# git clone git@gitlab.com:iankesh/htmlwebsite.git
Cloning into 'htmlwebsite'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
root@ip-172-31-16-128:~# cd htmlwebsite
root@ip-172-31-16-128:~/htmlwebsite#
```

3. Add README.md

```
root@ip-172-31-16-128:~/htmlwebsite# vi README.md
This is a demo project.
I am going to show you the CI & CD using Gitlab & AUTOMATIC TAG DEPLOYEMENT.
GitLab will do these FOUR things alone.
Here is the url in which the website is hosted:->
https://iankesh.gitlab.io/web/web.html
-Ankesh (M1036336)
```

4. Add .gitignore

   A .gitignore is very useful to avoid uploading to your remote repository any file or folder within your project. If you want to know more about it, check the .gitignore official docs.

```
root@ip-172-31-16-128:~/htmlwebsite# vi .gitignore
_site
```

Step 3: Add the configuration file: .gitlab-ci.yml

1. In order to build your plain HTML site with GitLab Pages, your .gitlab-ci.yml file doesn't need much:

```
root@ip-172-31-16-128:~/htmlwebsite# vi .gitlab-ci.yml
pages:
  stage: build
  script:
  - mkdir .public
  - cp -r * .public
  - mv .public public
  artifacts:
    paths:
    - public
  only:
  - master
  tags:
    - shell
```

2. What this code is doing is creating a job called pages telling the Runner to deploy the website artifacts to a public path, whenever a commit is pushed only to the master branch.
3. All pages are created after the build completes successfully and the artifacts for the pages job are uploaded to GitLab.

Step 4: Upload your website content

1. Add website.html

```
root@ip-172-31-16-128:~/htmlwebsite# vi website.html
<div class="city">
  <h2>Ankesh K</h2>
  <h3>Ankesh is in DevOps-COE</h3>
  <p>MID - M1036336</p>
  <p>Ph - 8197434654</p>
  <p>Desk - MTC-5F-120</p>
</div>
<div class="city">
  <h2>Sanjay CSK</h2>
  <h3>Sanjay is in DevOps-COE</h3>
  <p>MID - M1036322</p>
  <p>Ph - 7598660673</p>
  <p>Desk - MTC-5F-122</p>
</div>
<div class="city">
  <h2>Sameer M</h2>
  <h3>Sameer is in DevOps-COE</h3>
  <p>MID - M1036298</p>
  <p>Ph - 8884986885</p>
  <p>Desk - MTC-5F-121</p>
</div>
<div class="city">
  <h2>Subhash V</h2>
  <h3>Subhash is in DevOps-COE</h3>
  <p>MID - M1036316</p>
  <p>Ph - 734990408 </p>
  <p>Desk - MTC-5F-119</p>
</div>
</body>
</html>
```

Step 5: Push all the files to your remote repository

1. Add all the new files.
2. Commit the changes.
3. Push it to master branch of repository.

```
root@ip-172-31-16-128:~/htmlwebsite# git add README.md
root@ip-172-31-16-128:~/htmlwebsite# git add .gitignore
root@ip-172-31-16-128:~/htmlwebsite# git add website.html
root@ip-172-31-16-128:~/htmlwebsite# git add .gitlab-ci.yml
root@ip-172-31-16-128:~/htmlwebsite# git commit -m "first commit"
[master (root-commit) d94c2fd] first commit
 4 files changed, 68 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .gitlab-ci.yml
 create mode 100644 README.md
 create mode 100644 website.html
root@ip-172-31-16-128:~/htmlwebsite# git push origin master
Counting objects: 6, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.01 KiB | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To git@gitlab.com:iankesh/htmlwebsite.git
 * [new branch]      master -> master
```

Step 6: Add your custom domain (optional)

If you want, you are free to add your own domain(s) name to your website hosted by GitLab.com.

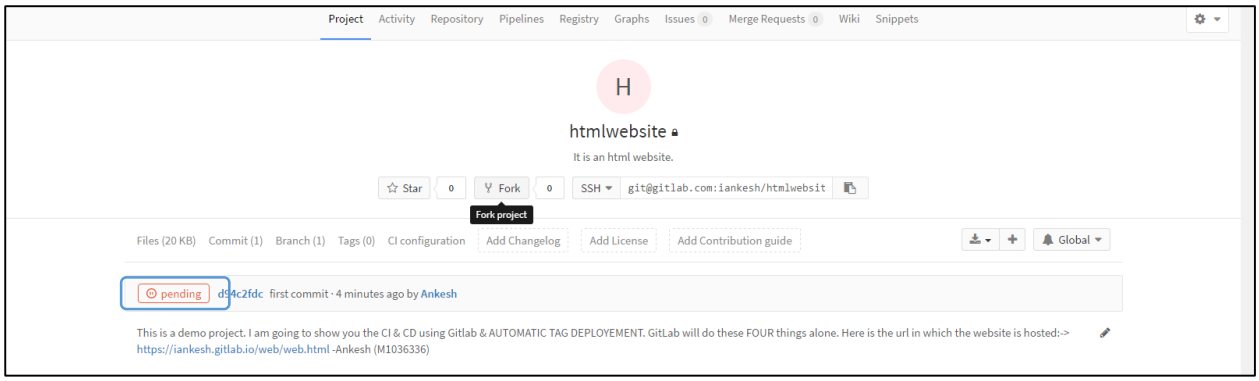
Steps to set up a custom domain:

1. From your project's dashboard, go to Settings () > Pages > New Domain
2. Add your domain to the first field: mydomain.com
3. If you have an SSL/TLS digital certificate and its key, add them to their respective fields. If you don't, just leave the fields blank.
4. Click on Create New Domain.
5. Finally, access your domain control panel and create a new DNS A record pointing to the IP of GitLab Pages server:

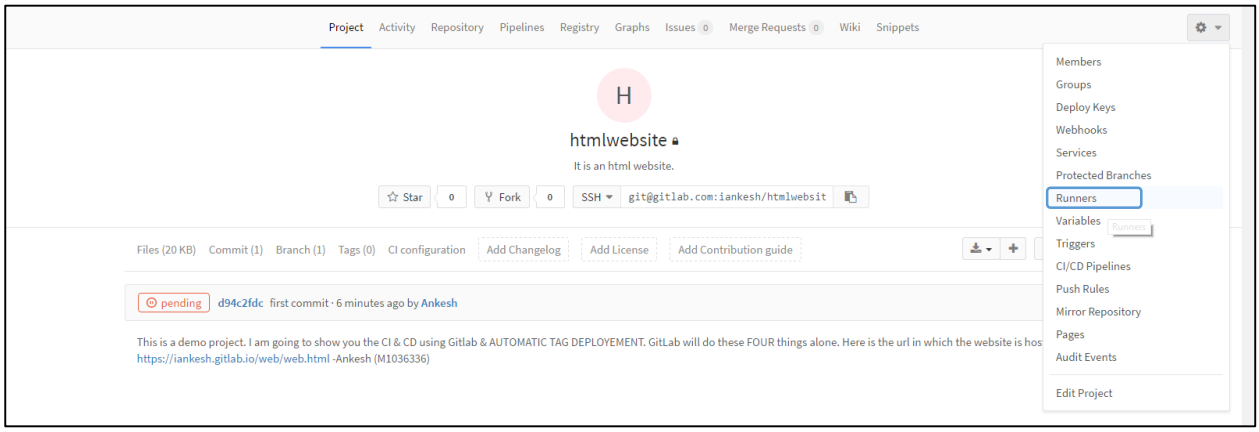
```
mydomain.com  A  52.89.67.41
```

Step 7: Build it with GitLab

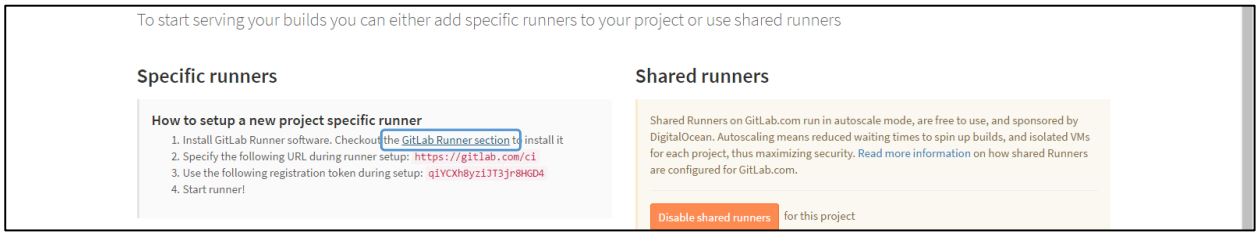
1. As soon as we push the files to remote repository, we can see that the build is pending.



2. So, we need to install Runners to get build. Settings > Runners



3. We have to install Runners for this project to run. We will follow the link provided:



a) If you want to use Docker runner, install it before using the multi runner:

```
root@ip-172-31-16-128:~/htmlwebsite# curl -sSL https://get.docker.com/ | sh
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can safely ignore this message.

You may press Ctrl+C now to abort this script.
+ sleep 20
```

...

```
docker-engine is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
+ sh -c docker version
Client:
 Version:      1.12.1
 API version:  1.24
 Go version:   go1.6.3
 Git commit:   23cf638
 Built:        Thu Aug 18 05:22:43 2016
 OS/Arch:      linux/amd64

Server:
 Version:      1.12.1
 API version:  1.24
 Go version:   go1.6.3
 Git commit:   23cf638
 Built:        Thu Aug 18 05:22:43 2016
 OS/Arch:      linux/amd64

If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!
```

b) Add GitLab' s official repository via apt-get or yum:

```
root@ip-172-31-16-128:~/htmlwebsite# curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-ci-mu
lti-runner/script.deb.sh | sudo bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5243    0  5243    0     0  41633      0 --:--:-- --:--:-- --:--:-- 41944
Detected operating system as Ubuntu/trusty.
Checking for curl...
Detected curl...
Running apt-get update... done.
Installing apt-transport-https... done.
Installing /etc/apt/sources.list.d/runner_gitlab-ci-multi-runner.list...done.
Importing packagecloud gpg key... done.
Running apt-get update... done.

The repository is setup! You can now install packages.
```

c) Install gitlab-ci-multi-runner:

```
root@ip-172-31-16-128:~/htmlwebsite# sudo apt-get install gitlab-ci-multi-runner
Reading package lists... Done
Building dependency tree
Reading state information... Done
gitlab-ci-multi-runner is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
```

d) Register the runner:

```
root@ip-172-31-16-128:~/htmlwebsite# sudo gitlab-ci-multi-runner register
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
https://gitlab.com/ci
Please enter the gitlab-ci token for this runner:
qiYCXh8yziJT3jr8HGD4
Please enter the gitlab-ci description for this runner:
[ip-172-31-16-128]: website-runner
Please enter the gitlab-ci tags for this runner (comma separated):
shell
Registering runner... succeeded                     runner=qiYCXh8y
Please enter the executor: docker+machine, kubernetes, docker, docker-ssh, ssh, docker-ssh+machine, parallels, sh
ell, virtualbox:
docker
Please enter the default Docker image (eg. ruby:2.1):
ruby:2.1
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatic
ally reloaded!
```

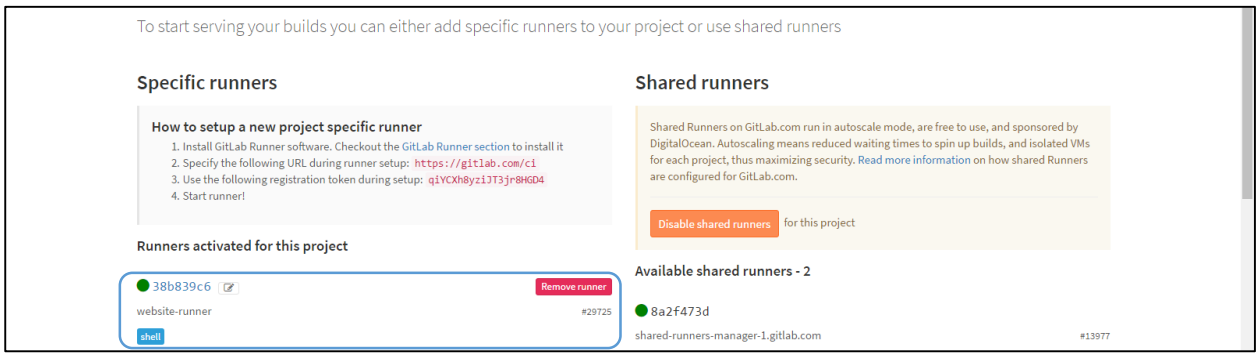
e) Simply execute to install latest version:

```
root@ip-172-31-16-128:~/htmlwebsite# sudo apt-get update
Ign http://us-west-2.ec2.archive.ubuntu.com trusty InRelease
Hit http://us-west-2.ec2.archive.ubuntu.com trusty-updates InRelease
Hit http://us-west-2.ec2.archive.ubuntu.com trusty-backports InRelease
Hit http://us-west-2.ec2.archive.ubuntu.com trusty Release.gpg
Hit http://us-west-2.ec2.archive.ubuntu.com trusty Release
Hit http://us-west-2.ec2.archive.ubuntu.com trusty-updates/main Sources
Hit http://us-west-2.ec2.archive.ubuntu.com trusty-updates/restricted Sources
Hit http://us-west-2.ec2.archive.ubuntu.com trusty-updates/universe Sources
```

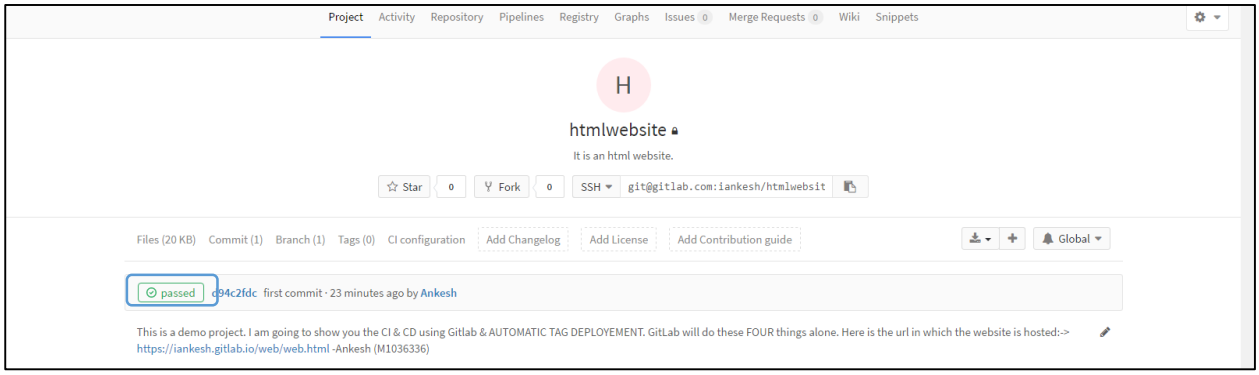
...

```
root@ip-172-31-16-128:~/htmlwebsite# sudo apt-get install gitlab-ci-multi-runner
Reading package lists... Done
Building dependency tree
Reading state information... Done
gitlab-ci-multi-runner is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
```

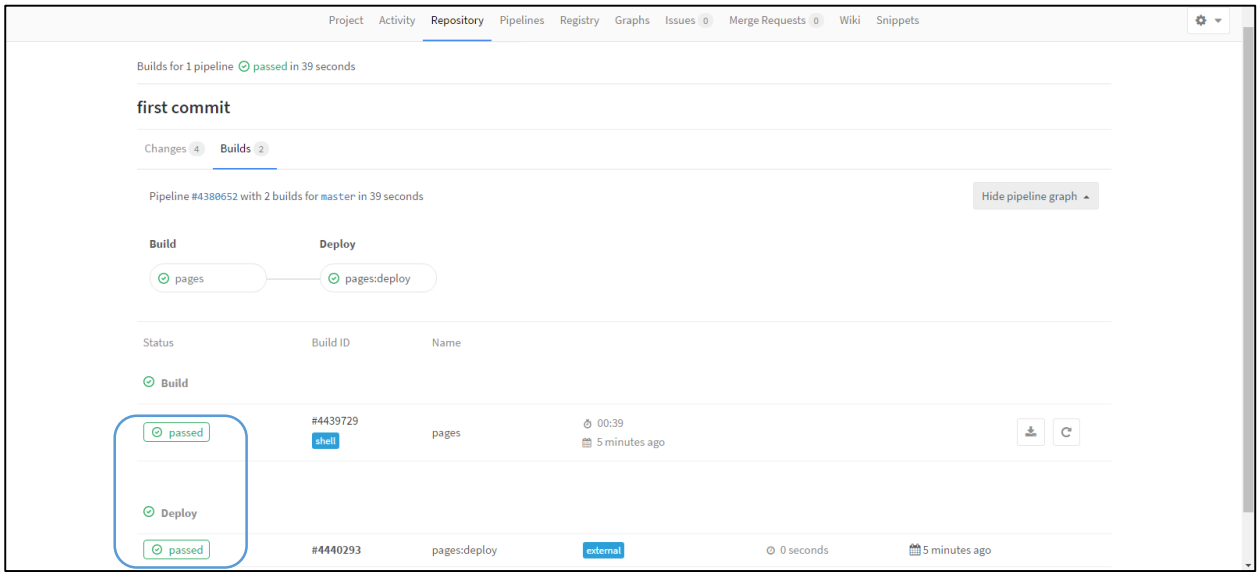
4. Now we can see that there is a runner for our project.



5. Now we can see that the build is automatically passed.



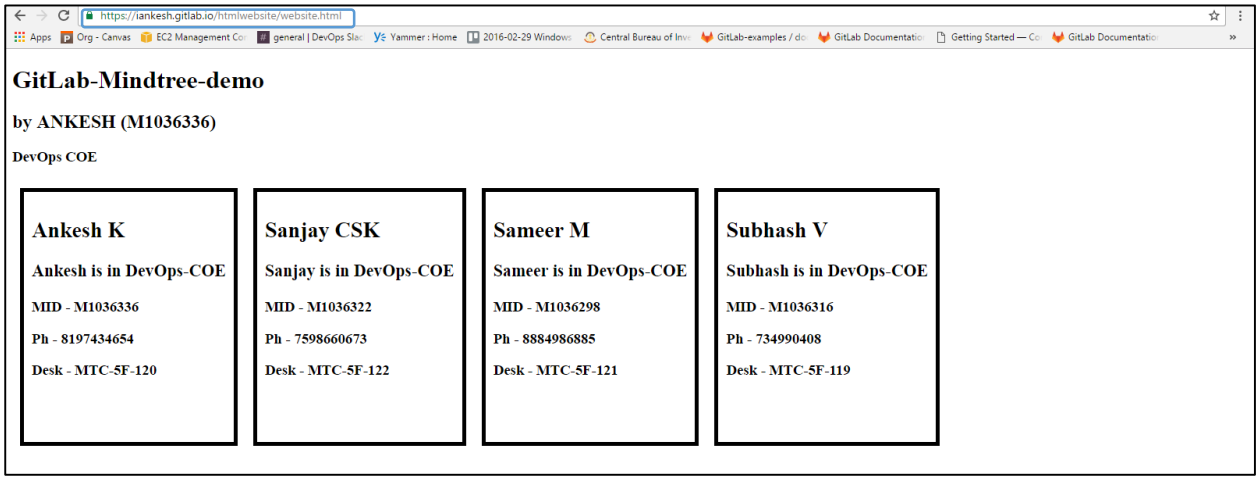
6. We can see the different stages of build which we have configured in YML file.



Step 8: Check the website deployed in their server

We can check the website deployed in the url:

https://iankesh.gitlab.io/htmlwebsite/website.html



CI & CD using GitLab:

Step 1:

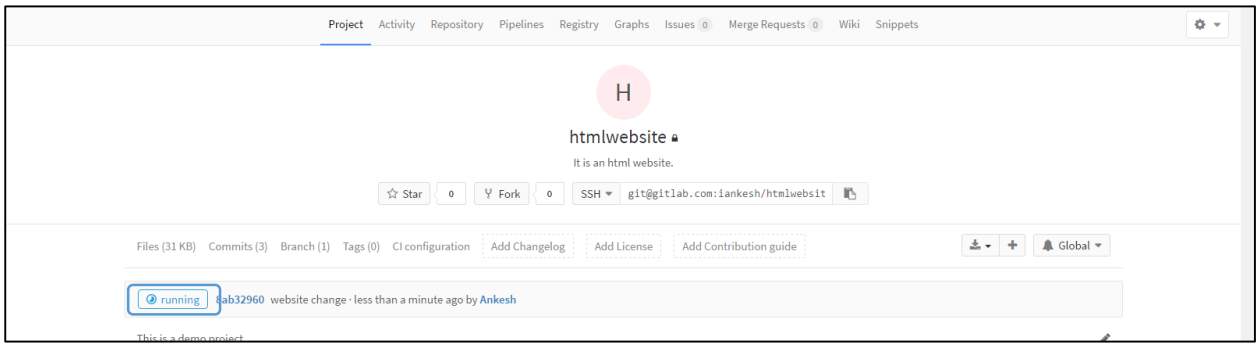
- We will make some changes in the website from our local repository.
- We will add a line “showing CI & CD” in our website code and push it to the remote repository.

```
root@ip-172-31-16-128:~/htmlwebsite# vi website.html
<!DOCTYPE html>
<html lang="en-us">
<head>
<style>
.city {
   float: left;
   margin: 10px;
   padding: 10px;
   max-width: 300px;
   height: 300px;
   border: 5px solid black;
}
</style>
</head>
<body>
<h1>GitLab-Mindtree-demo</h1>
<h2>by  ANKESH (M1036336)</h2>
<h3>DevOps COE<h3>
<h4>showing CI & CD</h4>
<div class="city">
  <h2>Ankesh K</h2>
  <h3>Ankesh is in DevOps-COE</h3>
  <p>MID - M1036336</p>
```

```
root@ip-172-31-16-128:~/htmlwebsite# git commit -am "website change"
[master 8ab3296] website change
 1 file changed, 1 insertion(+)
root@ip-172-31-16-128:~/htmlwebsite# git push origin master
Counting objects: 6, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 303 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To git@gitlab.com:iankesh/htmlwebsite.git
   8663550..8ab3296  master -> master
```

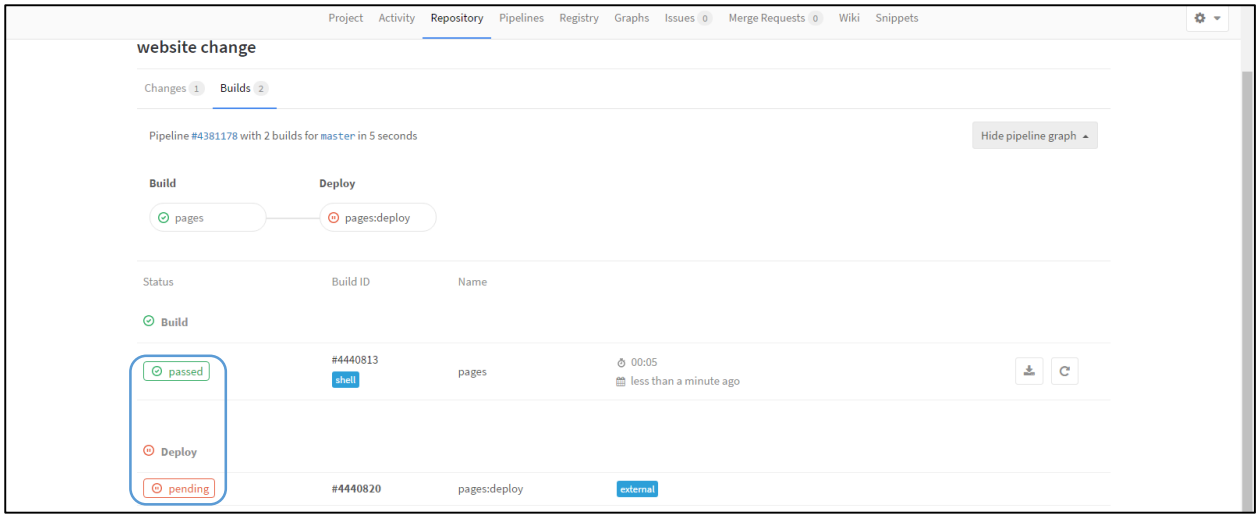
Step 2:

- As soon as we pushed the new code in GitLab remote, the build is automatically proceeded.
- And while running it shows the status as running.

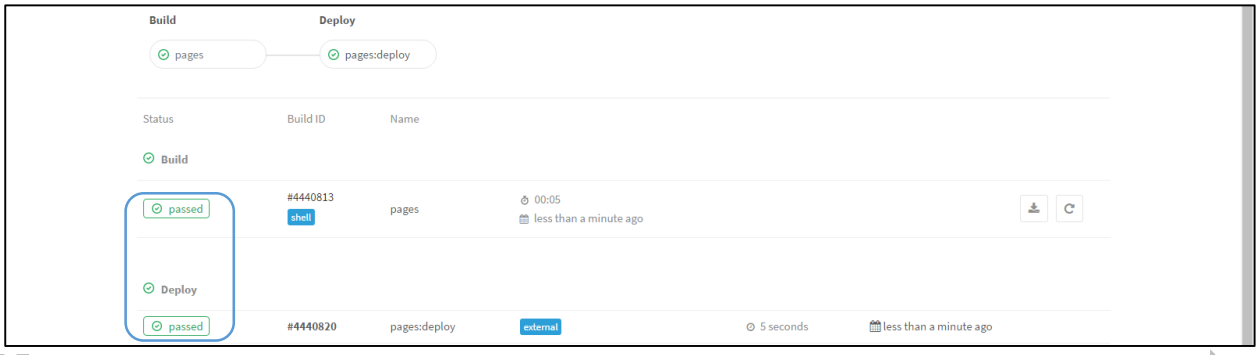


Step 3:

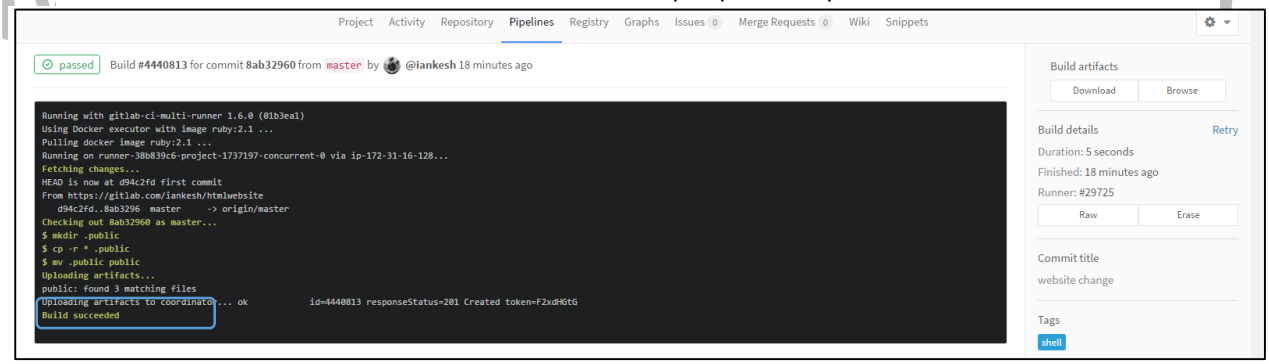
- We can see the build process by clicking on Repository section.
- We can see that GitLab will build in stages, it will keep the stage in pending if the last stage is in build process.
- If the first stage build is failed, it will not build the next stage. It will show error.



- When it builds both of the stages, it will pass to the deployment process.

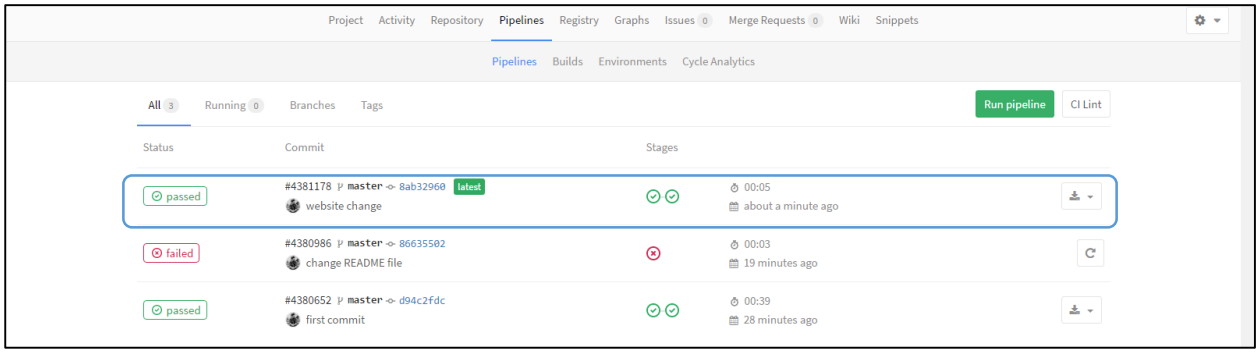


- We can see the build status here in console output part in Pipeline Section:



Step 4:

We can check out the builds in Pipeline section and we can download the artifacts manually by just clicking on the build number.



Step 5:

Check the website again. You can see the changes deployed.