University Göttingen
GWDG
Martin Paleico

Exercise 1 / February 23, 2023
HPC System Administration / WiSe 2022/23
55 Minutes Total

Exercise Introduction

In this exercise, we will install Gitlab CE (Community Edition) on the course VMs. We will then configure the service, add users for the course participants, start a new repository, and use it to test Git and issue/task tracking.

The idea of this session is to test service setup in general, and in particular show a very good service that can be used for collaborative work and project management for small projects.

Contents

Task 1: Installing Gitlab in your VM (5 min)

Install Gitlab CE in your frontend VM cluster-manager (or a VM with at least 4 GB of RAM). The steps are rather straightforward: installing some extra libraries from the CentOS' repository, and then Gitlab CE from Gitlab's repository.

1. `$ sudo yum -y update`
2. `$ sudo yum -y install curl vim policycoreutils python3-policycoreutils`
3. Optional: For email services: `$ sudo yum -y install postfix`
4. Optional: For email services: `$ sudo systemctl enable postfix && sudo systemctl start postfix`
5. `$ curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash`
6. `$ sudo yum install gitlab-ce`

Task 2: Configuring (5 min)

The last step before being able to access the service is telling Gitlab where to serve its homepage from. In this case we can do this without a reverse proxy service such as Apache or nginx.

1. Open the configuration file `$ sudo vi /etc/gitlab/gitlab.rb` (instead of `vi` you may also use `nano`)
2. Look for the external_url option near the top of the file, and change it to your machine's **external**/floating IP (the one you set up on OpenStack, that you use to ssh into the machine) while keeping the http header (e.g.: external_url 'http://14X.XXX.XXX.XXX').
3. By default Gitlab will run on port 80, the one reserved for http. If you need to change this to a different port, chose one from those available in the VM, and set external_url to e.g. 'http://14X.XXX.XXX.XXX:8010'
4. Run the following command to let Gitlab acknowledge your change (this will take a while the first time):
   `$ sudo gitlab-ctl reconfigure`
5. In your browser on your computer, go to IP:Port and the Gitlab page should appear (if you leave Port undefined, your browser will try Port 80 by default, and 443 if you specify https)
6. If you get a 502 error on your browser, the port you selected was probably being used for another service, change it. Also try restarting the service with `$ sudo gitlab-ctl restart`

Hints

- Whatever port you use (the default 80 http port or a different one), make sure it has been made available for your VM. Go to the OpenStack interface and check Network - Security Groups - default - Manage Rules. Your port should be activated for TCP Ingress (Egress is available in our default Security Group configuration already for all ports and protocols).

Task 3: Admin and New Users (10 min)

For this section, pick one of the members in the study group to be the admin. You will work together in their Gitlab CE instance (except for the Basics of git task, which should be done in your own computer).

Retrieve the default root admin password from `$ sudo vi /etc/gitlab/initial_root_password`. This file gets deleted after 24 hrs, so copy the password as soon as you can, or copy the file to somewhere safe like your home folder. Admin password can be also reset in other ways if need be.

Use this to sign in to the Gitlab instance as an admin. It is also recommended changing the default admin password to something more manageable (for setting the new password, do be aware Gitlab is a bit picky on what passwords it will allow). You can do this from the browser interface by clicking on your user icon on the top right, then on preferences.

Register new users: By default new user registration is open, requires no email confirmation but does require admin approval. To change any of these, from the admin account, open the menu on the top left next to Gitlab's icon, then click on the "Admin" button. In the new page, go to Settings, General, Sign-up Restrictions.

Have all the VM users create a normal user account (you can enter a fake email, but make sure the password you enter is easy to type since we will need it for another task; also, in testing, there were some problems when resetting an account's password so better not to do that). Then, to approve the newly created users, go to the Admin panel as before, Overview, Users, Pending Approval.

Hints

- If you are doing this alone, you can use different computers, different browsers in the same computer, or a browser plus incognito windows to simulate multiple users without having to log in and out constantly.

Task 4: Start a Repository (5 min)

Have one of the users create a blank repository/project, and other users join it. Initialize the repository with a README.md. To join the repository, either create a public repository and share the link with the other users, or a private one and invite other members by going to the repository, Project Information, Members, and searching for their usernames. Then the repository will appear on their Projects page.

Task 5: Optional: Basics of Git (10 min)

This task is optional, in case you have not used Git before. You can manipulate the repository directly from Gitlab itself too. Otherwise, everything in this task is performed on your own command line and not the browser.

On your own computer or the VM, install git from your usual package repository (e.g.: yum install gitlab). Go to a new folder, and clone the repository started in Gitlab. For this, go to the project's page, click on the blue clone button, and copy the contents of the "Clone with HTTP" header. In the new folder, use the command `$ git clone copied-text`, and enter your username and password on the Gitlab instance when prompted.

Now we can test the following Git features:

- Create an empty file, add some text to it, then add it to your local Git's "staging" with `$ git add myfile.txt`. The file has not been committed to the repository yet, it is just pre-staged.
- Commit the file (and any other staged changes) to the local repository with `$ git commit -m "Created myfile.txt"`. The text after -m is a comment that will appear together with the file's commit entry.
- Make a change to the file, save it, and check its status with `$ git diff myfile.txt`. Stage and commit your changes once again.
- Push your changes to the remote repository (the Gitlab instance) with `$ git push`. Check on Gitlab that the changes have actually taken place.
- If the other users have already created files, you might not be able to push your changes until your local repository has been updated. For this, use `$ git pull`.
- If the other users created files with the same name as yours, you might run into a merge conflict. These are common when working collaboratively, but are beyond our scope for today.
- Use `$ git log` to look at a history of the changes in your repository.

Hints

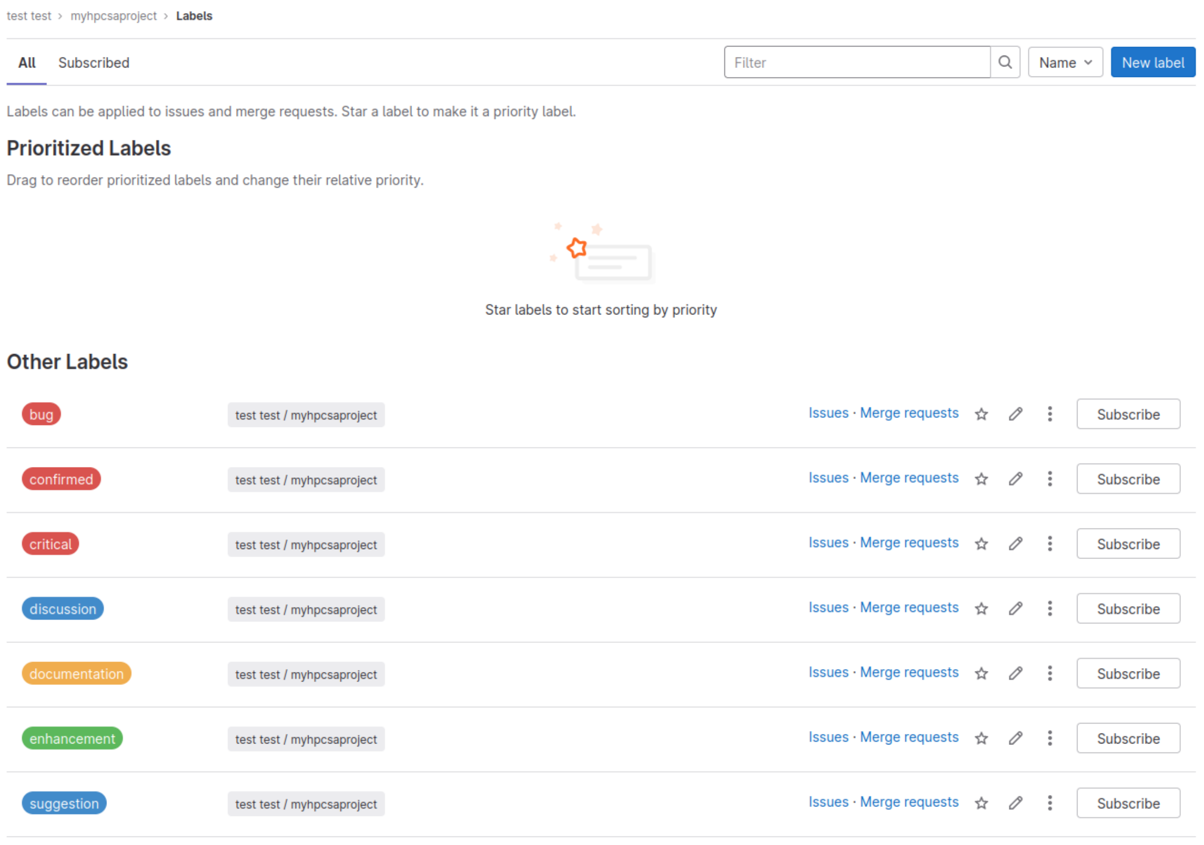
- Note that with Git you can also start a local repository without a remote copy (which can also be added after the fact). For this, look at the command `$ git init`.
- You can also clone the repository by setting up SSH keys on the Gitlab instance and using the "Clone with SSH" option, but this takes a bit more work (and is untested on this Gitlab CE setup). The advantage is there is no need to enter your user and password when cloning or pushing/pulling.

- Other basic git features to explore: creating, deleting and merging branches; .gitignore files, git rm, how to revert to an old commit or version of a file (git checkout), how to resolve merge conflicts.

Task 6: Issue Tracking (15 min)

We will look at some of Gitlab's issue and task tracking functionality:

Labels: Can be found under Project Information - Labels. Can be assigned a name, description, color. Also, a default set of labels is available. **Create the default set of labels or test your own.** Labels can be used to group issues by service, criticality, urgency, category, etc. These can then be issued to filter issues or collect them in a board, or subscribe to a particular label to get notifications whenever a new labelled issue is opened.



Labels can be applied to issues and merge requests. Star a label to make it a priority label.

Prioritized Labels

Drag to reorder prioritized labels and change their relative priority.

Star labels to start sorting by priority

Other Labels

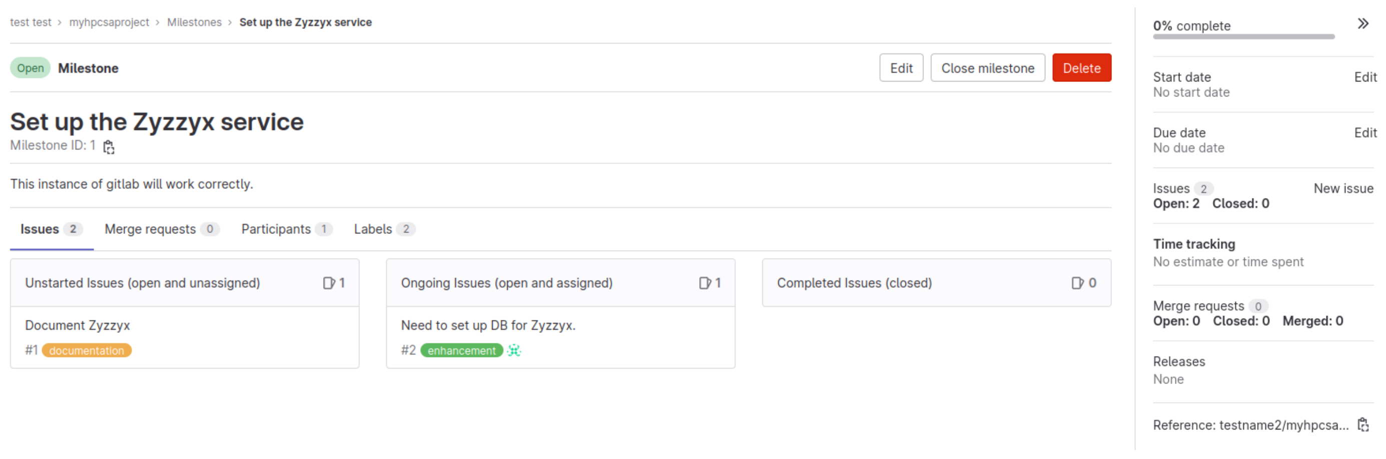
bug	test test / myhpcsaproject	Issues · Merge requests	Subscribe
confirmed	test test / myhpcsaproject	Issues · Merge requests	Subscribe
critical	test test / myhpcsaproject	Issues · Merge requests	Subscribe
discussion	test test / myhpcsaproject	Issues · Merge requests	Subscribe
documentation	test test / myhpcsaproject	Issues · Merge requests	Subscribe
enhancement	test test / myhpcsaproject	Issues · Merge requests	Subscribe
suggestion	test test / myhpcsaproject	Issues · Merge requests	Subscribe

Figure 1: Labels view in Gitlab

Milestones: Milestones are a way of collecting a number of issues together to keep track of them, or to list issues to achieve an ultimate goal. As a test, **start a new milestone** simulating the setup of a new service (for example, "Setup of the Zyzzyx service").

Issues: Issues can be bugs that need to be fixed, or tasks that need to be completed to achieve a certain milestone or implement a given feature. Issues are very versatile, and can even be used to start merge requests. **Create a new issue** by going to Issues - List. You can assign only one "assignee", but you can mention multiple people in the description by typing the at symbol (@) and their username. The mentioned users will receive notifications when they are mentioned, when the issue is updated, when the issue's deadline approaches, etc. (and they will also receive emails if the Gitlab instance is set up as such).

A number of special commands are available to be used in an issue's description. You can get an autocomplete list by typing / . Of note, you can mention other users (as explained above), reference other issues in the repository with #, reference files in the repository with [README](README.md), reference specific commits or merge requests, assign expected efforts and actual work hours, etc.


test test › myhpcsaproject › Milestones › **Set up the Zyzzyx service**

Open **Milestone** Edit Close milestone Delete

Set up the Zyzzyx service

Milestone ID: 1

This instance of gitlab will work correctly.

Issues 2 Merge requests 0 Participants 1 Labels 2

Unstarted Issues (open and unassigned) 1	Ongoing Issues (open and assigned) 1	Completed Issues (closed) 0
Document Zyzzyx #1 documentation	Need to set up DB for Zyzzyx. #2 enhancement	

0% complete

Start date No start date Edit

Due date No due date Edit

Issues 2 New issue
Open: 2 Closed: 0

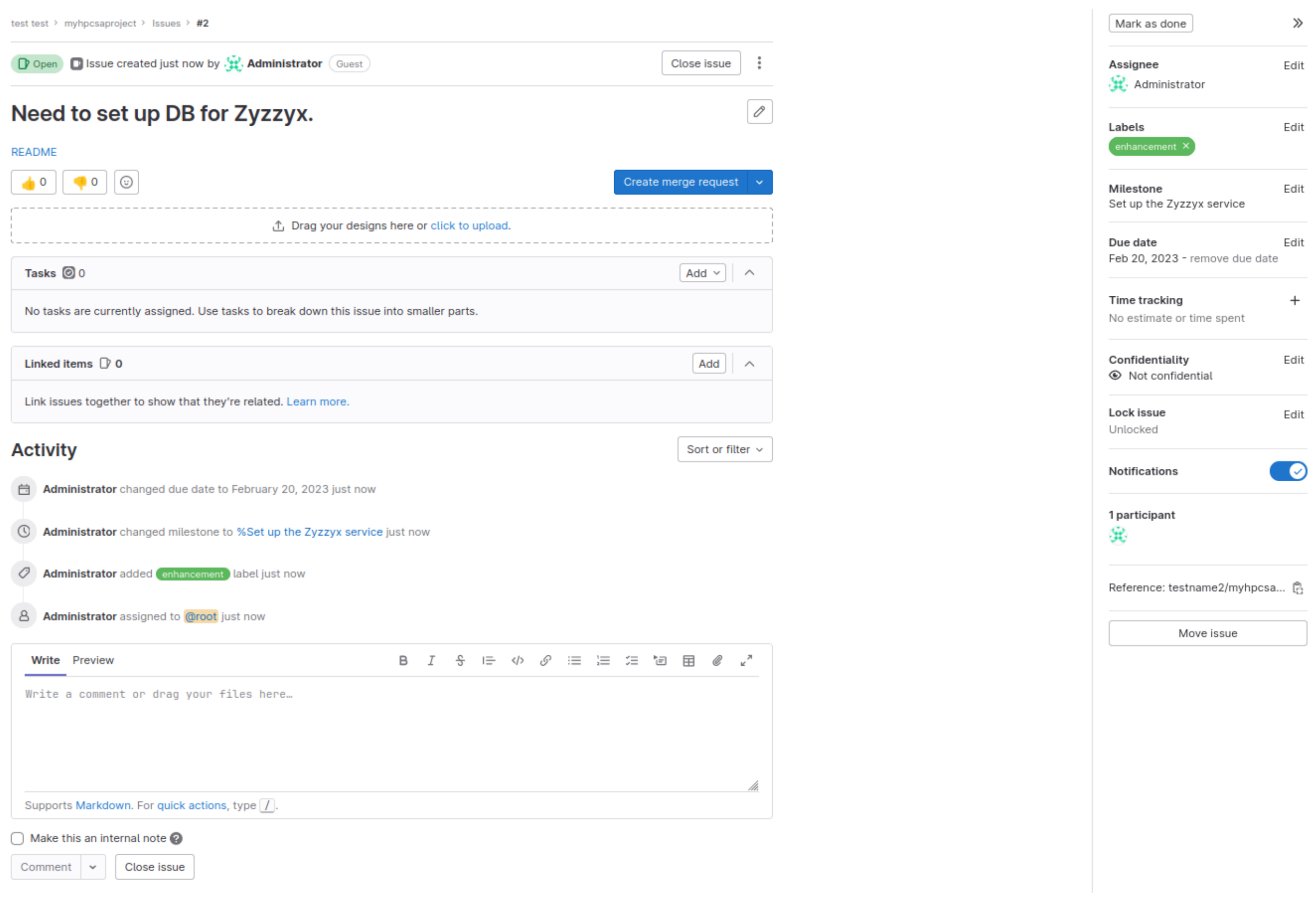
Time tracking
No estimate or time spent

Merge requests 0
Open: 0 Closed: 0 Merged: 0

Releases
None

Reference: testname2/myhpcsa...

Figure 2: Milestones GUI in Gitlab


test test › myhpcsaproject › Issues › **#2**

Open Issue created just now by **Administrator** Guest Close issue

Need to set up DB for Zyzzyx.

README

👍 0 👎 0 Create merge request

Drag your designs here or click to upload.

Tasks 0 Add

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0 Add

Link issues together to show that they're related. Learn more.

Activity Sort or filter

Administrator changed due date to February 20, 2023 just now

Administrator changed milestone to %Set up the Zyzzyx service just now

Administrator added enhancement label just now

Administrator assigned to @root just now

Write Preview

Write a comment or drag your files here…

Supports Markdown. For quick actions, type /.

Make this an internal note

Comment Close issue

Mark as done

Assignee Edit
Administrator

Labels Edit
enhancement

Milestone Edit
Set up the Zyzzyx service

Due date Edit
Feb 20, 2023 - remove due date

Time tracking
No estimate or time spent

Confidentiality Edit
Not confidential

Lock issue Edit
Unlocked

Notifications

1 participant

Reference: testname2/myhpcsa...

Move issue

Figure 3: Example of an Issue in Gitlab

Generate at least one issue per user in the repository and assign it to them, you can also make use of the special commands to for example reference the files that have been created before. Example issues: "Document the Zyzzyx Service", "Add Database to the Zyzzyx service", etc. Remember to add the issues to the created Milestone, and to add relevant tags. Once the issue is created, look around in the issue's page at all the available features and test some of them (for example, add a comment, link another issue, add subtasks, etc.). Once you have a couple of issues open, check back on the Milestone's page to see how they have been collected.

Boards: Boards are a way of seeing all your issues at a glance. **Check out the board** for your project and see if the issues you have created have appeared there. Test creating a new board and new lists. Also test quickly creating a new issue from the boards interface.

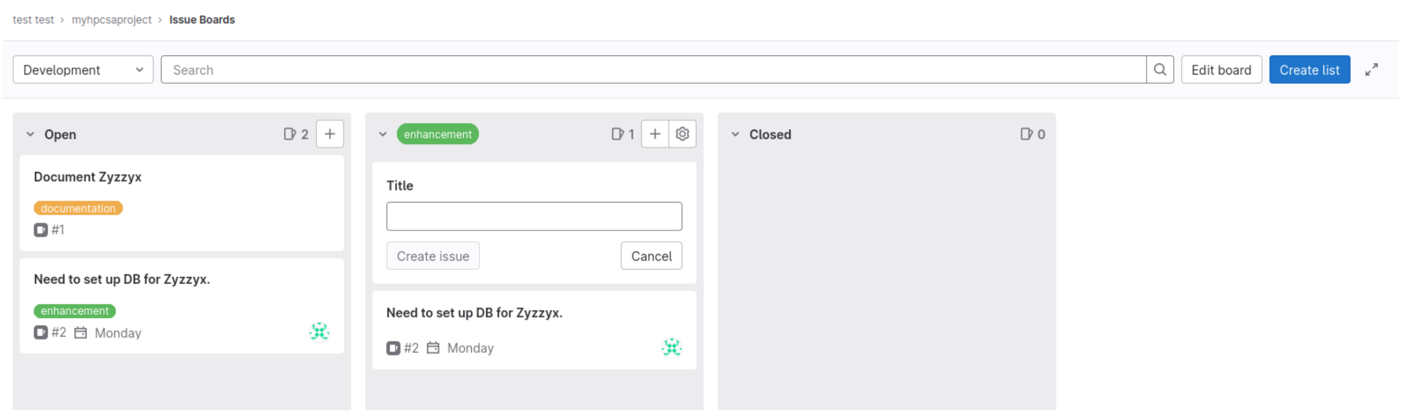


Figure 4: Basic view of Boards in Gitlab

Hints

- Your project also has an associated wiki that can be used for documentation.
- As an advanced feature, look up information on Gitlab's Service Desk option, which allows external users without an account in your repository to open issues (a sort of mini ticket system)
- Other similar tools to check out: OpenProjects for task tracking, Trello (and many other similar services) for issue boards, Confluence for wiki plus some task tracking, GitHub for an alternative to Gitlab (but no self-hosted instance as far as I can tell.
- There are many, many other features in Gitlab, it can become an all-in-one solution (for better and for worse).
- If you are interested in further techniques for collaborative development and organization, look into topics such as agile development and kanban (boards can be used for this last one).

Task 7: Optional: Cleaning Up (5 min)

Gitlab CE requires quite a bit of disk and memory resources, For that reason, it is recommended to uninstall it when you are done with the exercises. This is also important if you used the default port for Gitlab, and want to install other services that need it afterward (in that case you can also just change the config file to use a different port and rerun the configuration).

To uninstall:

1. Stop the Gitlab process: `$ sudo gitlab-ctl stop`
2. Uninstall from yum: `$ sudo yum uninstall gitlab-ce`, and the same for the other packages installed if you want to
3. You can also use `$ yum history` and `$ yum history undo ID` to revert changes
4. Optionally, to save on RAM you can just stop the Gitlab CE process from automatically running with `$ sudo systemctl disable gitlab-runsvdir.service`, and still use it by starting and stopping it manually with `$ sudo gitlab-ctl start/stop`