Exercise – A Gitlab server of our very own

Objective

This exercise looks at setting up your own GitLab server and allowing other people access to it. We will set up the server on aws. The GitLab server can be used in the Case Study if you want to investigate how to create a full pipeline.

Part 1: Setup

We will need to create a new server in aws to run Gitlab for us. Create a new EC2 instance with the following properties:

- AMI – Amazon Linux AMI 2018.x
- Size – t2.medium
- Storage – 20 GiB
- Tag – name: GitLab
- Security group: Open port 80/tcp for access from anywhere
- Assign an Elastic IP (See the “QG Assigning Elastic IPs to EC2 instances)”

Connect to your server using putty or SSH. If you need help with any of these steps, please see the quick guides for creating a new instance, generating the key and connecting via putty or the command line.

Part 2: Installing Gitlab

To install GitLab we need to do the following:

1. Run the first three lines of the script. This will install the services required for GitLab, start postfix (a mail server) tell postfix to automatically start.

```
$ sudo yum install curl openssh-server postfix cronie -y
$ sudo service postfix start
$ sudo chkconfig postfix on
```

2. Next, we want to grab a copy of the installation script with curl and pipe it through bash to run. Once the script has run we install gitlab-ce through yum.

```
$ curl
https://packages.gitlab.com/install/repositories/gitlab/gitlab-
ce/script.rpm.sh | sudo bash

$ sudo yum install gitlab-ce -y
```

3. Configure GitLab

```
$ sudo gitlab-ctl reconfigure
```

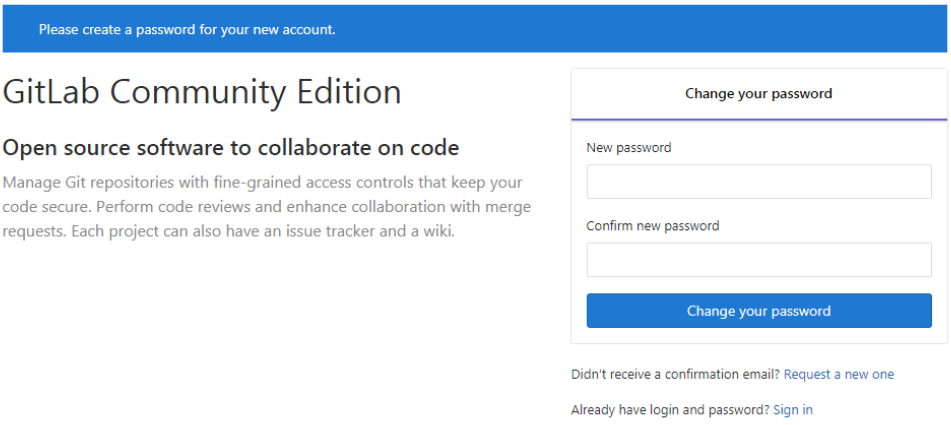
Gitlab uses chef-solo to configure and set itself up.

If the reconfigure command fails, run it a second time. There is a bug in the gitlab script that assumes that you will have the bridge module enabled in your OS kernel. The second reconfigure fixes this problem.

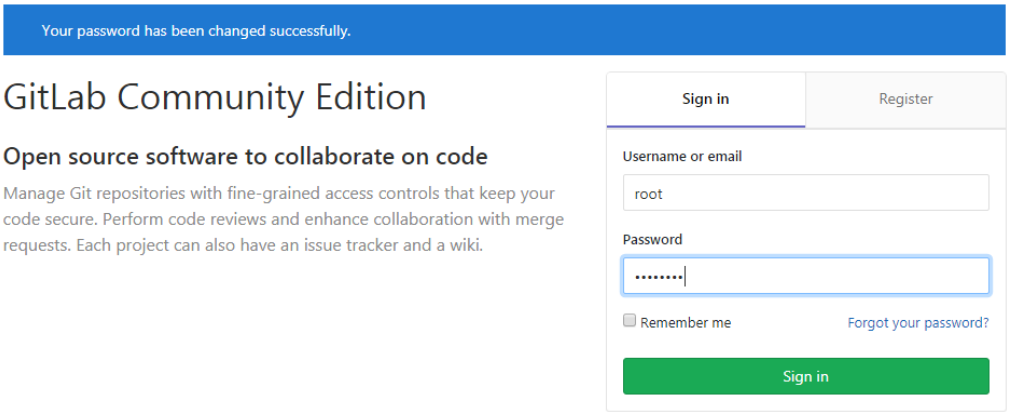
When everything has finished have a look at your new GitLab server in a browser with **http://[your_gitlab_ip]**

4. Gitlab will prompt you to change your password immediately. Change this to **Pa$$w0rd**

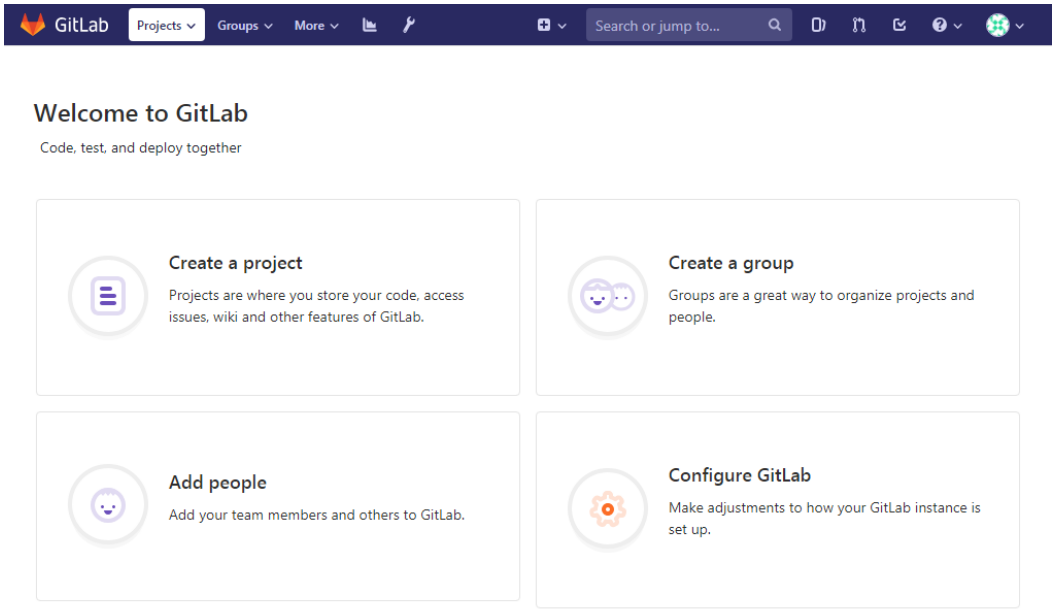
Your username is **root** by default



Login with username of **root** and a password of **Pa$$w0rd**

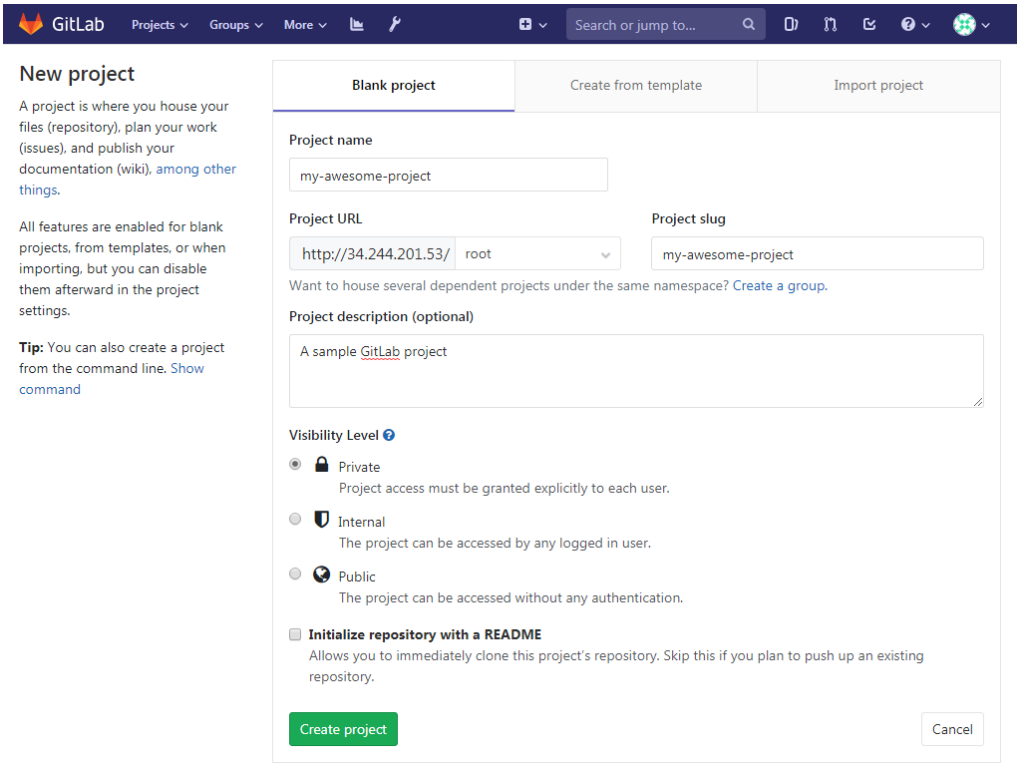


Once you have done this you will be rewarded with the main dashboard:

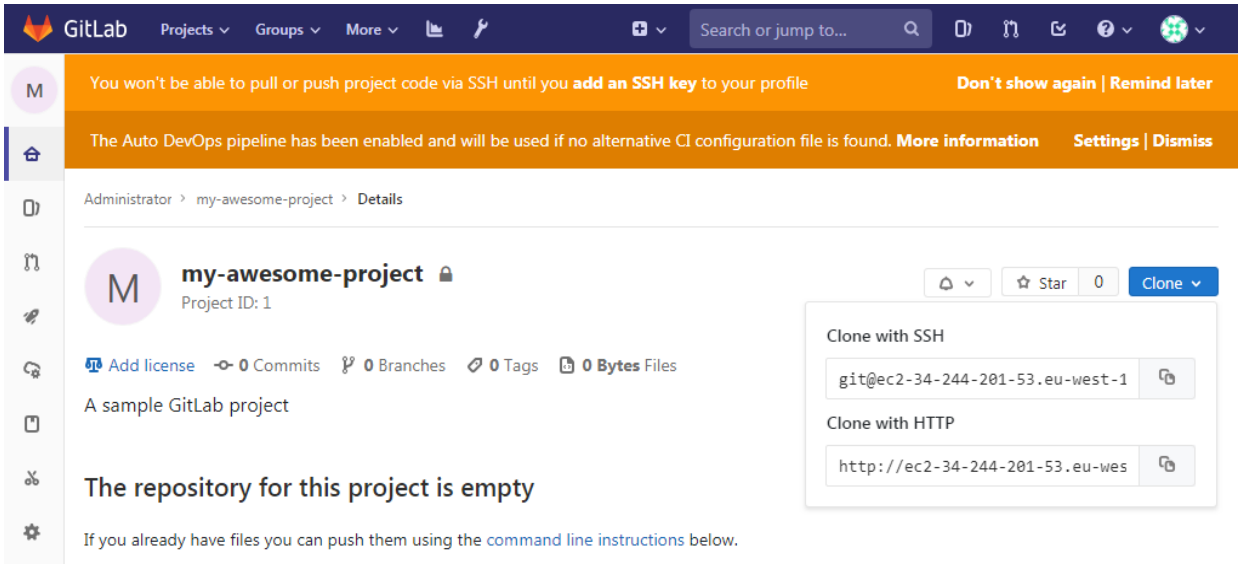


Part 3: Create a project

5. Now we have a server, we want to create a new project. Click on the Create a Project panel on the home page.
6. Give your project a name and a description. Any name will do. Then click the create project button.



7. Click the Clone button and then copy the **Clone with HTTP** url repository link.



8. We should be able to clone this repository to any machine. Try cloning your repository to another one of your aws instances.

```
$ sudo yum install git -y
$ git clone [http repo address]
```

9. Create a file in your new GitLab repository, commit and push it. Your username and password will be **root** and **Pa$$w0rd**. See if your files appear on your server.

Part 3: Automate the installation process

The main question that we should be asking whenever we install a new system is:

> "How can I automate that?"

AWS allows you to pass it a script that is run on start up. For linux this will be a bash script. It should have the extension of **.sh** and start with the line

```
#!/bin/bash
```

Write a script which goes through each of the steps required to install and setup a GitLab server. Create a new machine via aws and pass it your script and see if you can get GitLab to be installed without any help.

Part 4: Document

Write down the steps you took over this exercise in your documentation git repository. Add your script from part 3. Remember to write down any problems you had and how you solved them, so that you can refer to this later.

If you have time…

- Have a look at creating a backup for your GitLab server
- Create a new EC2 instance and restore your setup from this backup
- Look at how to automate the backup with cron table (crontab)