

Hamidou_Mary_gitlab

Rappel du contexte de la consigne

Introduction ;

Aujourd'hui, le cloud prend une part de plus en plus importante dans les infrastructures et le développement. De plus l'utilisation des services managés et de l'Infrastructure As Code a changé les paradigmes de nos métiers. Dans ce cadre, votre évaluation va consister en un projet à réaliser par deux, de préférence entre {un,une} Ops et {un,une} Dev. Le but sera d'éprouver les capacités de collaboration offertes par Git et Gitlab. Vous devrez donc monter deux projets Git, l'un pour l'infra et l'autre pour le dev, totalement automatisés.

Pour cela vous allez aborder les deux thématiques suivantes :

- L'Infrastructure As Code avec Terraform sur le cloud AWS
- Le développement dit "serverless" sur une Lambda AWS de code Python

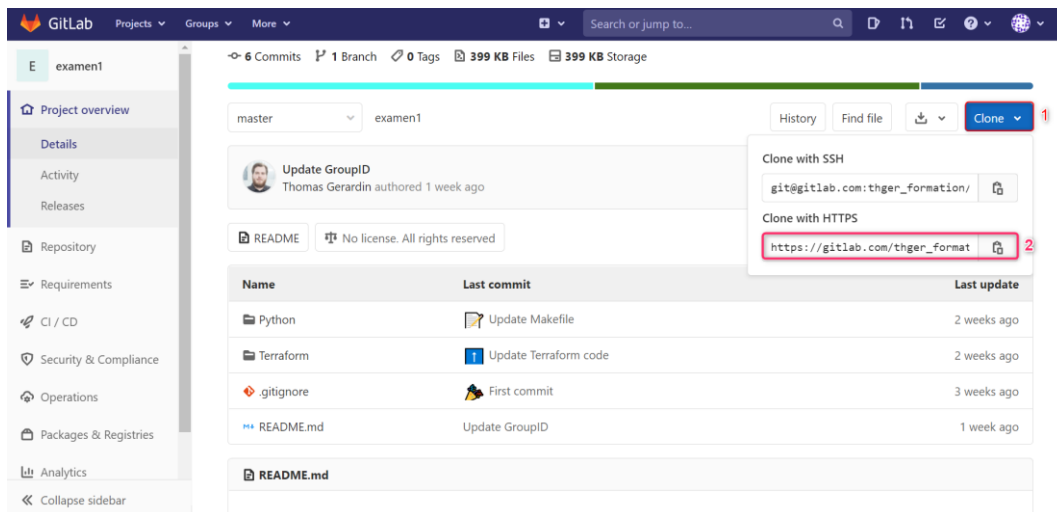
Consignes :

Pour les solos :

Sur une instance Gitlab CE autohébergée ou sur Gitlab.com, cela peut-être celle créée pendant les TP, effectuer les actions suivantes :

- 2 projets Git et 2 comptes associés pour chacune des personnes. L'un pour l'infrastructure l'autre pour le code. Le code vous est fourni dans les 2 répertoires Terraform pour la partie OPS et Python pour la partie Dev.
- 2 chaînes CD complète pour chacun des projets (décrite en détail dans le Makefile)
 - Pour l'infrastructure, la chaîne sera composée des étapes suivantes :
 - test qui exécutera une validation locale des fichiers Terraform
 - apply qui lui permettra de déployer l'infrastructure et stockera les fichiers terraform.tfstate dans un artefact (15 jours de conservation)
 - destroy qui va permettre de supprimer l'infrastructure
 - Pour le côté Dev, la chaîne sera composée des étapes :
 - test qui lancera des tests unitaires
 - release qui va packager le code dans un livrable (.zip ici, pour être déployé sur une Lambda)
 - deploy qui lui doit-être uniquement exécuté sur la branche master, et qui va comme son nom l'indique déployer le code

Aller sur le site Gitlab https://gitlab.com/thger_formation/examen1 pour pouvoir récupérer le projet dont nous aurons besoin. En cliquant sur **Clone**, une fenêtre apparaît avec deux adresses url pour à copier en fonction de notre méthode de communication avec le gitlab.



N'ayant pas fait la connexion en SSH, je vais copier l'url **HTTPS**

Clone with SSH

git@gitlab.com:thger_formation/

Clone with HTTPS

https://gitlab.com/thger_format

Sur la machine virtuelle, après s'être authentifié en super utilisateur et se positionner sur le répertoire de travail, il faut taper **git clone** et coller l'url que l'on a copié précédemment.

Dans notre cas, **git clone** https://gitlab.com/thger_formation/examen1.git

```
root@debian:/home/user# git clone https://gitlab.com/thger_formation/examen1.git
Clonage dans 'examen1'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 35 (delta 11), reused 0 (delta 0), pack-reused 0
```

En tapant la commande **ls**, nous devons voir le répertoire **examen1** apparaître.

```
root@debian:/home/user# ls
democd Dockerfile examen1 mapremierechainecd
```












On souhaite voir si répertoire examen1 contient tous les fichiers et dossier du projet cloné en faisant un **cd Terraform/**

```
root@debian:/home/user# cd examen1/
root@debian:/home/user/examen1# ls
Python README.md Terraform
```









Puis un **ls** pour lister les fichiers du répertoire.

```
root@debian:/home/user/examen1# cd Terraform/  
root@debian:/home/user/examen1/Terraform# ls  
lambda_function.zip  main.tf  Makefile  variables.tf
```

On compare les fichiers clonés et les fichiers du projet d'origine pour nous assurer que le clone à bien fonctionné.

<div> Update GroupID Thomas Gerardin authored 1 week ago</div> <div>cb71a5b4 </div>		
<div> README</div> <div> No license. All rights reserved</div>		
Name	Last commit	Last update
 Python	 Update Makefile	2 weeks ago
 Terraform	 Update Terraform code	2 weeks ago
 .gitignore	 First commit	3 weeks ago
 README.md	Update GroupID	1 week ago

Ce qui nous intéresse plus précisément, ce sont les fichiers se trouvant dans le dossier Terraform.

Name	Last commit	Last update
..		
 Makefile	 Update doc	2 weeks ago
 lambda_function.zip	 First commit	3 weeks ago
 main.tf	 Update Terraform code	2 weeks ago
 variables.tf	 Update Terraform code	2 weeks ago

Pour toutes les étapes qui vont suivre, je fais le faire sur mon serveur Gitlab par choix personnel. Il y a plus de fonctionnalité sur le serveur installé et je m'y retrouve plus facilement.

Pour pouvoir se connecter en ssh, il faut générer une nouvelle clé en tapant la commande **ssh-keygen -t ed25519** et laisser les configurations par défaut.

```

root@debian:/home/user/examen1/Terraform# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
/root/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519.
Your public key has been saved in /root/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:brn9AwzykN5FQFwCvHP8vYBodUS0nSTeLKip8XSDN2k root@debian
The key's randomart image is:
+---[ED25519 256]---+
|      .++XB..      |
|      . o.== .      |
|      = +..o        |
|      O B E         |
|      . /S@ +        |
|      =.*.= .        |
|      . + o .        |
|      . o o          |
|      . ....        |
+-----[SHA256]-----+

```

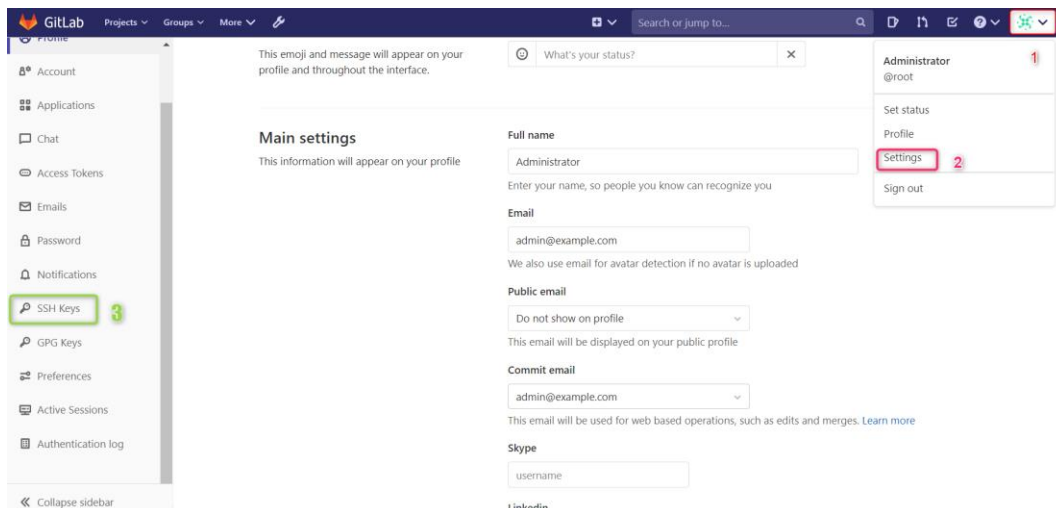
Les clés générées sont stockées dans le répertoire `/root/.ssh` et pouvoir voir la clé publique on tape `cat /root/.ssh/id_ed25519` et on **copie** intégralement cette clé.

```

root@debian:/home/user/examen1/Terraform# cat /root/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7rYzYv5ZzNROy1PIaGs5d5AwjErZz1I0H+oUHSeTwR root@debian

```

Sur le navigateur Gitlab, il faut se rendre dans **paramètres d'administration**, cliquer sur **Settings** et cliquer sur l'onglet **SSH Keys**.



On **colle** la clé que l'on a copié précédemment dans l'emplacement Key, on met une **date de validité** de la clé puis on clique sur **Add Key**.

User Settings > SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key
To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key
Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

1 `ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7rYzYv5ZzNR0y1PIaGs5d5AwjErZz1I0H+oUHSeTwR root@debian`

Title
root@debian

Expires at
30/10/2020 2

Give your individual key a title. This will be publicly visible.

3 **Add key**

Your SSH keys (1)

Le récapitulatif de l'ajout de la clé SSH ce qui nous permettra de se connecter et faire des manipulations sans s'authentifier à chaque fois.

User Settings > SSH Keys > root@debian

SSH Key	ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP7rYzYv5ZzNR0y1PIaGs5d5AwjErZz1I0H+oUHSeTwR root@
Title: root@debian	
Created on: Oct 23, 2020 4:01pm	
Expires: Oct 30, 2020 12:00am	
Last used on: Never	
Fingerprints	
MD5: e6:f1:a6:1e:48:83:7c:bc:cf:2f:56:64:5b:dc:91:f3	
SHA256: brn9AwzykN5FQFwCvHP8vYBodUS0nSTE1KiP8XSDN2k	

Remove

Dans notre interface web Gitlab, on va devoir créer un nouveau projet. Pour cela, il faut cliquer sur :

- 1- **Projects**
- 2- **New project**

GitLab Projects Groups More Search or jump to...

Projects 1

2 **New project**

Your projects 3 Starred projects Explore projects Filter by name... Last updated

All Personal

M	Administrator / MaPremiereChaineCD Maintainer	★ 0 ♻ 0 📄 0	Updated 3 weeks ago
D	Administrator / demoCD Maintainer	★ 0 ♻ 0 📄 0	Updated 3 weeks ago
M	GitLab Instance / Monitoring Owner This project is automatically generated and will be used to help monitor this GitLab instance. More information	★ 0 ♻ 0 📄 0	Updated 3 weeks ago

Une fenêtre s'ouvre. On ajoute le nom du projet « **Terraform** » puis on clique sur **Create project**

Blank project Create from template Import project

Project name
Terraform 1

Project URL Project slug
http://192.168.130.245/ root terraform

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)
Description format

Visibility Level [?](#)
☐ Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.
☐ Internal
The project can be accessed by any logged in user.
☒ Public
The project can be accessed without any authentication.

☐ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project 2 Cancel

Le projet est créé mais il est vide. Gitlab donne des instructions en fonction du type de projet que l'on veut créer avec les commandes données.

Administrator > Terraform > Details

T

Terraform

Project ID: 4

Star 0

The repository for this project is empty

You can get started by cloning the repository or start adding files to it with one of the following options.

Clone

New file

Add README

Add LICENSE

Add CHANGELOG

Add CONTRIBUTING

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "Administrator"
git config --global user.email "admin@example.com"
```

Create a new repository

```
git clone git@192.168.130.245:root/terraform.git
cd terraform
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Ce qui nous intéresse le plus dans ces instructions c'est de créer un projet avec un répertoire existant. Il nous faudra suivre les étapes ci-dessous.

Push an existing folder

```
cd existing_folder
git init
git remote add origin git@192.168.130.245:root/terraform.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Pour faire la configuration globale, il faut taper la commande **git config --global user.name « Administrator »**

```
root@debian:/home/user/examen1/Terraform# git config --global user.name "Administrator"
```

Pour pouvoir créer un projet à partir d'un répertoire existant, il faut entrer respectivement les commandes suivantes :

- 1- **git init**
- 2- **git remote add origin git@192.168.130.145:root/terraform.git**
- 3- **git add .**
- 4- **git commit -m «Initial commit »**
- 5- **git push -u origin master**

```
root@debian:/home/user/examen1/Terraform# git init
Dépôt Git vide initialisé dans /home/user/examen1/Terraform/.git/
root@debian:/home/user/examen1/Terraform# git remote add origin git@192.168.130.245:root/terraform.git
root@debian:/home/user/examen1/Terraform# git add .
root@debian:/home/user/examen1/Terraform# git commit -m "Initial commit"
[master (commit racine) facd8fb] Initial commit
 4 files changed, 74 insertions(+)
 create mode 100644 Makefile
 create mode 100644 lambda_function.zip
 create mode 100644 main.tf
 create mode 100644 variables.tf
root@debian:/home/user/examen1/Terraform# git push -u origin master
remote:
remote: INFO: Your SSH key is expiring soon. Please generate a new key.
remote:
Énumération des objets: 6, fait.
Décompte des objets: 100% (6/6), fait.
Compression des objets: 100% (6/6), fait.
Écriture des objets: 100% (6/6), 1.47 KiB | 188.00 KiB/s, fait.
Total 6 (delta 0), réutilisés 0 (delta 0)
To 192.168.130.245:root/terraform.git
 * [new branch]      master -> master
La branche 'master' est paramétrée pour suivre la branche distante 'master' depuis 'origin'.
root@debian:/home/user/examen1/Terraform#
```

Sur le Navigateur Web, on voit que notre projet contient tous les documents importés.

The screenshot shows the Terraform web interface. On the left is a sidebar with navigation links: Project overview, Details, Activity, Releases, Repository, Issues, Merge Requests, CI / CD, Operations, Packages & Registries, Analytics, and Wiki. The main content area shows the 'Details' page for a project named 'Terraform' (Project ID: 4). It displays statistics: 1 Commit, 1 Branch, 0 Tags, 164 KB Files, and 164 KB Storage. Below this, there's a section for the 'Initial commit' by 'Administrator' 32 seconds ago. A table lists the files in the commit:

Name	Last commit	Last update
Makefile	Initial commit	33 seconds ago
lambda_function.zip	Initial commit	33 seconds ago
main.tf	Initial commit	33 seconds ago
variables.tf	Initial commit	33 seconds ago

Pour chaque projet Gitlab, il faut qu'il y un fichier README.md donc on en crée un à l'aide de la commande **vim README.md**

```
root@debian:/home/user/examen1/Terraform# vim README.md
```

Le fichier se crée et s'ouvre puis on ajoute du texte.

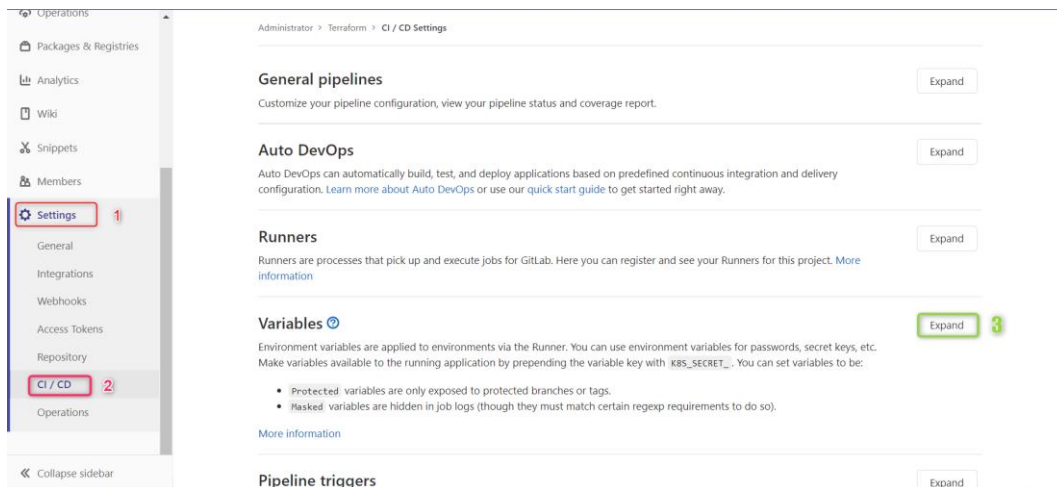
```
Création de document README.md
~
~
```

Une fois le fichier modifié et enregistré, il faudra a nouveau faire les commandes afin de permettre de le rendre visible en ligne.

CREATION DE VARIABLES PROTEGEES

Pour créer une variable protégée, il faut :

- 1- Cliquer sur l'onglet **Settings**
- 2- Lorsque le menu se déroule, cliquer sur **CI / CD**
- 3- Recherche la rubrique « Variable » puis cliquer sur **Expand**



La rubrique se déroule et on clique sur **Add Variable**

Variables ?

Collapse

Environment variables are applied to environments via the Runner. You can use environment variables for passwords, secret keys, etc. Make variables available to the running application by prepending the variable key with `K8S_SECRET_`. You can set variables to be:

- **Protected** variables are only exposed to protected branches or tags.
- **Masked** variables are hidden in job logs (though they must match certain regexp requirements to do so).

[More information](#)

Environment variables are configured by your administrator to be **protected** by default

Type	↑ Key	Value	Protected	Masked	Environments
There are no variables yet.					

Add Variable

On saisit le **nom** de la clé, la **valeur** de la clé, on coche la case « **Protect variable** » (si ce n'est pas fait) puis on clique sur « **Add Variable** »

Add variable

×

Key

AWS_ACCESS_KEY_ID 1

Value

AKIA2G2XJY5P5L4HR070 2

Type

Variable

Environment scope

All (default)

Flags

☒ Protect variable ?

Export variable to pipelines running on protected branches and tags only.

☐ Mask variable ?

Variable will be masked in job logs. Requires values to meet regular expression requirements. [More information](#)

💡 Deploying to AWS is easy with GitLab

[Use a template to deploy to ECS](#), or use a docker image to [run AWS commands in GitLab CI/CD](#).



[Learn more about deploying to AWS](#)

3

Cancel

Add variable

Répéter les étapes pour les variables :

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY_ID**
- **GROUP_ID**

Variables ?

Collapse

Environment variables are applied to environments via the Runner. You can use environment variables for passwords, secret keys, etc. Make variables available to the running application by prepending the variable key with `K8S_SECRET_`. You can set variables to be:

- **Protected** variables are only exposed to protected branches or tags.
- **Masked** variables are hidden in job logs (though they must match certain regexp requirements to do so).

[More information](#)

Environment variables are configured by your administrator to be **protected** by default

Type	↑ Key	Value	Protected	Masked	Environments	
Variable	AWS_ACCESS_KEY_ID	*****	✓	✗	All (default)	
Variable	AWS_SECRET_ACCESS_KEY	*****	✓	✗	All (default)	
Variable	GROUP_ID	*****	✓	✗	All (default)	

[Reveal values](#) [Add Variable](#)

CREATION DES RUNNERS

Nous allons à présent créer des Runners que nous pourrons exécuter dans nos jobs.

Dans notre exercice, nous avons besoin de trois runner :

- **test**
- **deploy**
- **destroy**

Il faut faire les étapes suivantes :

- 1- Cliquer sur l'outil **Admin Area**
- 2- Faire dérouler l'**Overview** en cliquant dessus
- 3- Cliquer sur **Runners**
- 4- Suivre les étapes dans l'encadré « **Set up a shared Runner manually** »

GitLab Projects Groups More

Admin Area

Overview 2

Dashboard

Projects

Users

Groups

Jobs

Runners 1

GitLab Servers

Admin Area > Runners

A "Runner" is a process which runs a job. You can set up as many Runners as you need. Runners can be placed on separate users, servers, even on your local machine.

Each Runner can be in one of the following states and/or belong to one of the following types:

- **shared** - Runner runs jobs from all unassigned projects in its group
- **group** - Runner runs jobs from all unassigned projects in its group
- **specific** - Runner runs jobs from assigned projects
- **locked** - Runner cannot be assigned to other projects
- **paused** - Runner will not receive any new jobs

Set up a shared Runner manually

1. Install GitLab Runner
2. Specify the following URL during the Runner setup: `http://192.168.130.245/`
3. Use the following registration token during setup: `ai3KSA0Qy43y5h9_yHh`
4. Start the Runner!

Reset runners registration token

Recent searches: Search or filter results... Created date: Runners currently online: 2

Type/State	Runner token	Description	Version	IP Address	Projects	Jobs	Tags	Last contact
shared locked	x75xC6iA	container	13.5.0-beta.8...	192.168.130...	n/a	4	container locked test	7 minutes ago
shared locked	YD95dJzy	test-runner	13.5.0-beta.8...	192.168.130...	n/a	15	test-runner	7 minutes ago

Pour pouvoir créer les runners, il faut installer gitlab-runner.

Dans mon cas il est déjà installé. Se rendre sur sa machine virtuelle. Pour vérifier l'installation on fait entre la commande **gitlab-runner status**

```
root@debian:/home/user/examen1/Terraform# gitlab-runner status
Runtime platform arch=amd64 os=linux pid=94914 revision=laa4b2fc version=13.5.0-beta.80.glaa4b2fc
gitlab-runner: Service is running!
```

Ensuite nous pouvons suivre les étapes du « Set up a shared Runner manually » en parallèle que l'on exécute les commandes.

Set up a shared Runner manually

1. Install GitLab Runner

2. Specify the following URL during the Runner setup: <http://192.168.130.245/>



3. Use the following registration token during setup: [m1jK5A9Qy41yShh9_yMN](#)



[Reset runners registration token](#)

4. Start the Runner!

Entrer la commande suivante : **gitlab-runner register**

Puis suivre instruction :

- 1- Copier l'adresse <http://192.168.130.245> se trouvant dans le « set up a shared Runner manually » et le coller
- 2- Le token est demandé. Faire Copier le **token** et le coller
- 3- Dans la description, entrer container-test
- 4- On ajoute auquel on veut l'associer : **test, docker-test**
- 5- On sélectionne un exécuteur : **docker**
- 6- On met l'image docker : **hashicorp/terraform:light** (présent dans le docker hub)

```
root@debian:/home/user/examen1/Terraform# gitlab-runner register
Runtime platform arch=amd64 os=linux pid=96175 revision=laa4b2fc version=13.5.0-beta.80.glaa4b2fc
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.130.245/
Please enter the gitlab-ci token for this runner:
m1jK5A9Qy41yShh9\_yMN
Please enter the gitlab-ci description for this runner:
(debian): Container-test
Please enter the gitlab-ci tags for this runner (comma separated):
test,docker-test
Registering runner... succeeded runner=m1jK5A9Q
Please enter the executor: virtualbox, docker-ssh, parallels, shell, ssh, docker+machine, docker-ssh+machine, custom, docker, kubernetes:
docker
Please enter the default Docker image (e.g. ruby:2.6):
hashicorp/terraform:light
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

En rafraîchissant le navigateur, on voit le message du succès de la mise à jour du Runner et qu'une nouvelle ligne a été ajoutée.

The screenshot shows the GitLab Runners interface. At the top, a blue notification bar states "Runner was successfully updated." with a red '1' next to it. Below this, there's a section titled "Set up a shared Runner manually" with a list of steps: 1. Install GitLab Runner, 2. Specify the following URL during the Runner setup: `http://192.168.130.245/`, 3. Use the following registration token during setup: `m1jK5A9Qy41yShh9_y7W`, and 4. Start the Runner!. A button "Reset runners registration token" is also present.

Below the setup instructions, there's a table of runners. The table has columns: Type/State, Runner token, Description, Version, IP Address, Projects, Jobs, Tags, and Last contact. A red '2' points to the first row of the table, which is highlighted with a pink border. The first row shows a runner with Type/State "shared locked", Runner token "GYGreASH", Description "container-test", Version "13.5.0~beta.8...", IP Address "192.168.130.245", Projects "n/a", Jobs "0", Tags "docker-test test", and Last contact "just now".

CREATION D'UNE NOUVELLE BRANCHE DE TRAVAIL

Dans les bonnes pratiques, il est vivement déconseillé de travailler directement sur la branch master.

```
root@debian:/home/user/examen1/Terraform# git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.
rien à valider, la copie de travail est propre

stage:
- test

Terra_test:
  image: hashicorp/terraform:light
  stage: test
  tags: docker
  script:
    - terraform init .
    - terraform validate .
```

On va créer une nouvelle branche appelé **Ops** à l'aide de la commande **git checkout -b Ops**. Cette commande va créer la branche Ops et faire le basculement sur celle-ci.

```
root@debian:/home/user/examen1/Terraform# git checkout -b Ops
Basculement sur la nouvelle branche 'Ops'
```

Pour savoir sur quelle branche on se trouve, il suffit d'entrer **git branch** et la branche sur laquelle on se trouve sera mis en évidence.

```
root@debian:/home/user/examen1/Terraform# git branch
* Ops
master
```

CREATION DU FICHIER .GITLAB-CI.YML

Une fois placé sur la branch Ops, on va pouvoir créer notre document avec la commande **vim .gitlab-ci.yml**

```
root@debian:/home/user/examen1/Terraform# vim .gitlab-ci.yml
```

Le document va s'ouvrir et on l'initialise le premier job que l'on va appeler **Terra_test**

```
stages:
  - test
  - deploy
  - destroy

Terra_test:
  image:
    name: hashicorp/terraform:light
    entrypoint: [""]
  stage: test
  tags:
    - test
  script:
    - terraform init .
    - terraform validate .
```

Faire un **git add .gitlab-ci.yml** pour ajouter le fichier dans le staging

```
root@debian:/home/user/examen1/Terraform# git add .gitlab-ci.yml
```

EN faisant un **git status**, on voit un nouveau fichier crée et en attente.

```
root@debian:/home/user/examen1/Terraform# git status
Sur la branche Ops
Modifications qui seront validées :
  (utilisez "git reset HEAD <fichier>..." pour désindexer)

    nouveau fichier : .gitlab-ci.yml
```

Entrer la commande **git commit -m « initialisation du fichier .gitlab-ci.yml et ajout du stage test »**

```
root@debian:/home/user/examen1/Terraform# git commit -m "Initialisation du fichier .gitlab-ci.yml et ajout du stage test"
[Ops 753d94e] Initialisation du fichier .gitlab-ci.yml et ajout du stage test
1 file changed, 13 insertions(+)
create mode 100644 .gitlab-ci.yml
```

Faire un **git push -u origin Ops** pour qu'il soit visible depuis le repository gitlab distant

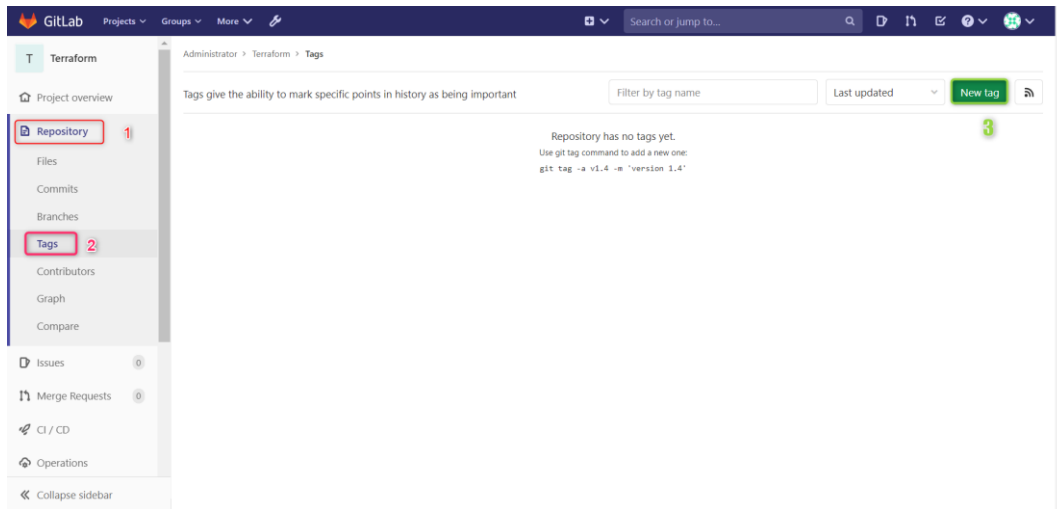
```
root@debian:/home/user/examen1/Terraform# git push -u origin Ops
remote:
remote: INFO: Your SSH key is expiring soon. Please generate a new key.
remote:
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 415 bytes | 59.00 KiB/s, fait.
Total 3 (delta 1), réutilisés 0 (delta 0)
remote:
remote: To create a merge request for Ops, visit:
remote:   http://192.168.130.245/root/terraform/-/merge_requests/new?merge_request%5Bsource_branch%5D=Ops
remote:
To 192.168.130.245:root/terraform.git
 * [new branch]      Ops -> Ops
La branche 'Ops' est paramétrée pour suivre la branche distante 'Ops' depuis 'origin'.
```

CREATION D'UN TAG

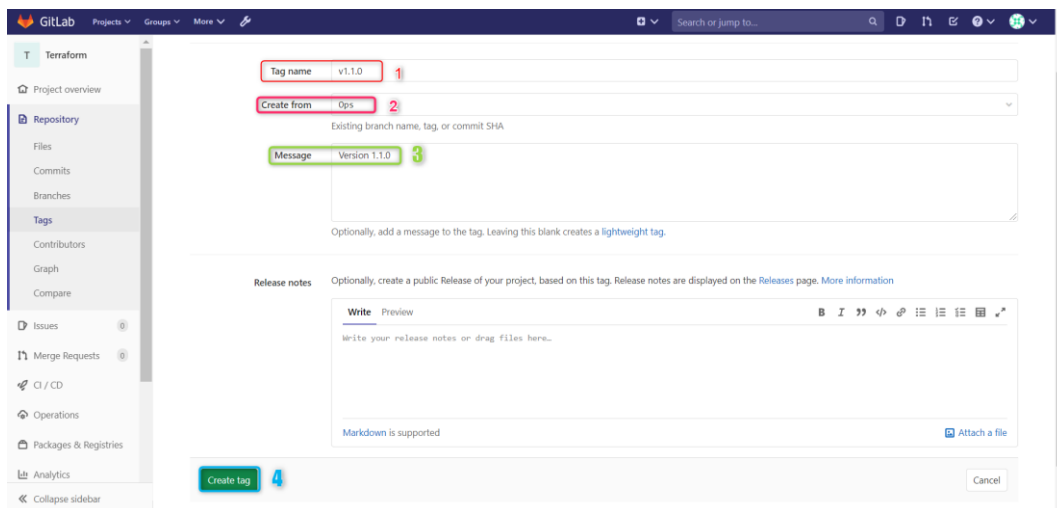
Il y a deux moyens d'ajouter un tag à un fichier commit :

- En ligne de commande avec la commande **git tag -a v1.0 -m « version 1.0 »**
- Sur le navigateur web de **Gitlab**

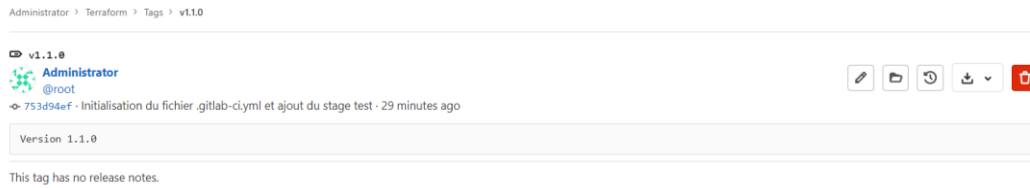
Pour le premier je vais le faire depuis le navigateur, en cliquant sur l'onglet **Repository, Tags** puis **New tag**



On renseigne les champs comme indiqué dans la capture ci-dessous



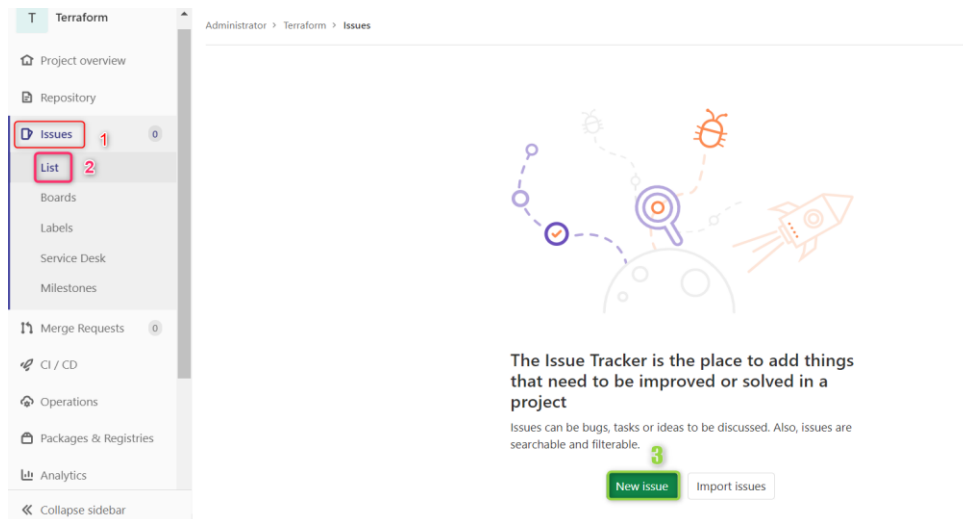
Le tag est créé et il s'allie au dernier fichier commit et push



CREATION D'UN TICKET

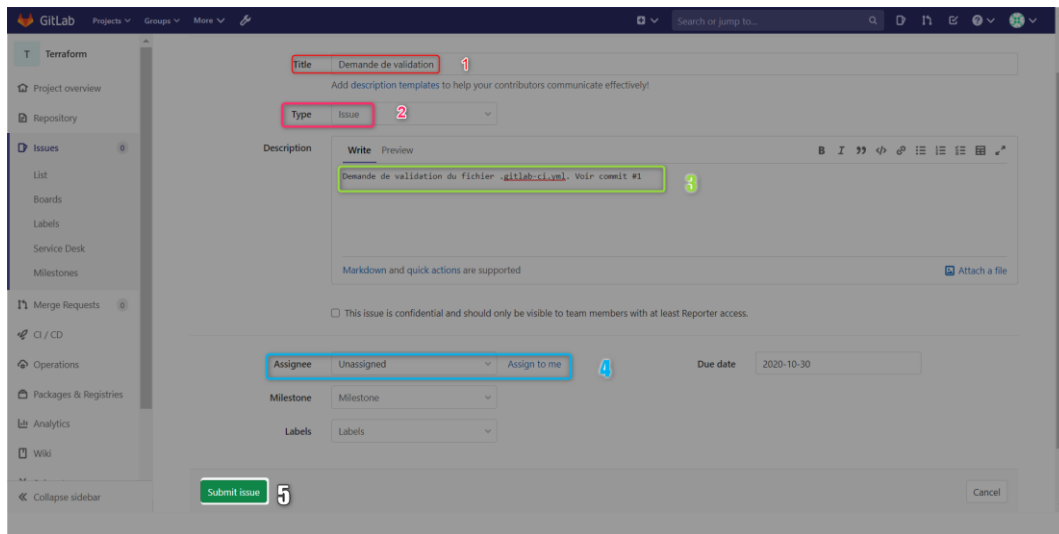
Pour créer un ticket, il faut :

- 1- Cliquer sur l'onglet **Issues**
- 2- Une fois déroulé, cliquer sur **List**
- 3- Puis cliquer sur **New Issue**

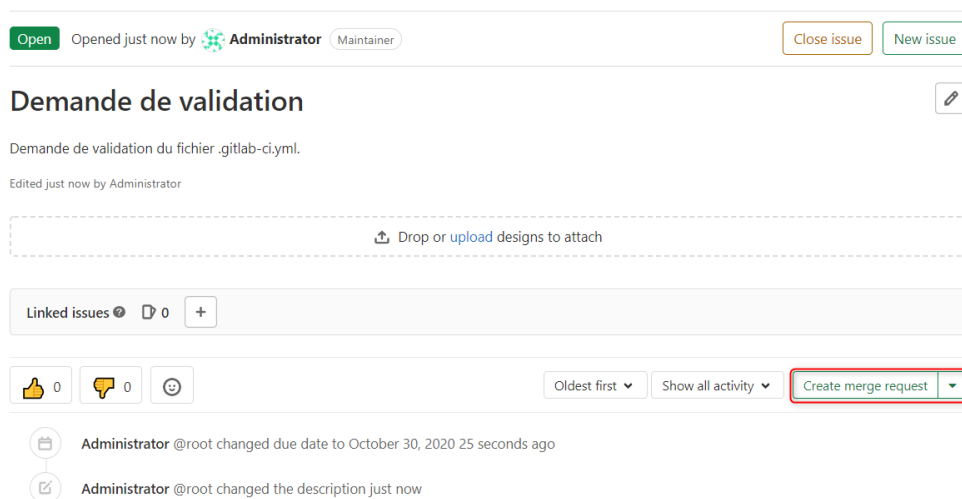


La fenêtre de création de ticket :

- 1- On ajoute le **titre** du ticket
- 2- On précise le **type** du ticket (Issue ou Incident)
- 3- On entre la **description** du ticket
- 4- On peut **assigner** le ticket à une personne ou un groupe pour validation
- 5- On clique sur **Submit Issue**



Le ticket est créé.

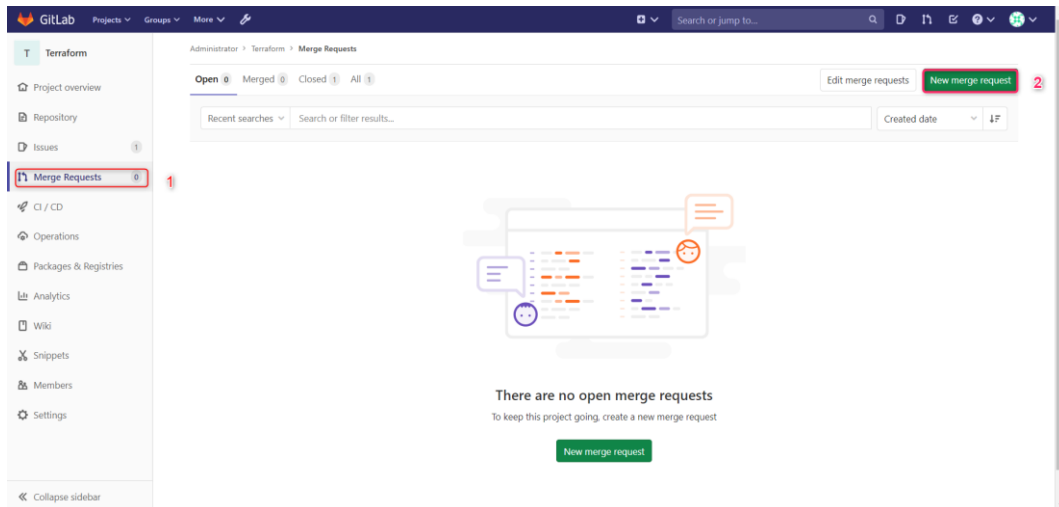


CREATION D'UNE MERGE REQUEST

La Merge Request permet de fusionner le fichier se trouvant dans la branche Ops avec le fichier de la branche Master.

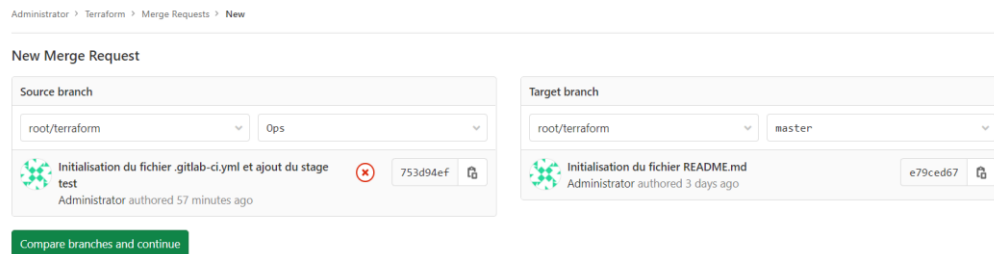
Pour en créer une faire les étapes suivantes :

- 1- Cliquer sur l'onglet **Merge Requests**
- 2- Cliquer sur **New merge request**

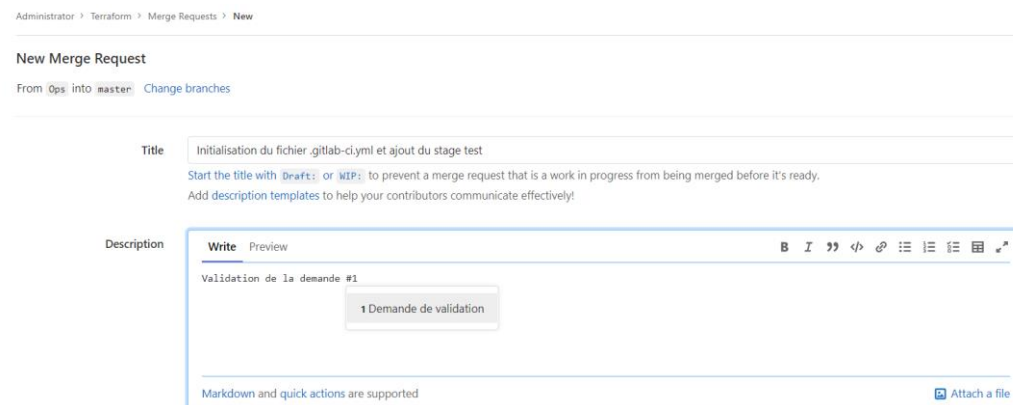


Sélectionner la branche **Ops** pour la source et la branche **master** pour la cible.

/ ! \ Attention erreur sur la capture d'écran fichiers source et destination sont différents !!



Une fenêtre s'ouvre et dans la description on fait appel au ticket créé en écrivant **#1** puis on clique sur **Submit merge request**



Assignee: Administrator

Milestone: Milestone

Labels: Labels

Merge options: ☒ Delete source branch when merge request is accepted.
☐ Squash commits when merge request is accepted. ?

Submit merge request Cancel

Commits 1 Pipelines 1 Changes 1

27 Oct, 2020 1 commit

Initialisation du fichier .gitlab-ci.yml et ajout du stage test
Administrator authored 1 hour ago

Il faut approuver la MR en cliquant sur **approve** puis cliquer sur **Merge**

Administrator > Terraform > Merge Requests > 12

Open Opened just now by Administrator Maintainer Edit Close merge request

Initialisation du fichier .gitlab-ci.yml et ajout du stage test

Overview 0 Commits 1 Pipelines 2 Changes 1

Validation de la demande #1

Request to merge Ops into master Open in Web IDE Check out branch

Detached merge request pipeline #14 failed for 753d94ef

1 Revoke approval Merge request approved. Approved by

2 Merge Delete source branch

> 1 commit and 1 merge commit will be added to master. Modify merge commit

Mentions #1


You can merge this merge request manually using the command line


Confirmation de la Merge request



Initialisation du fichier .gitlab-ci.yml et ajout du stage test




Overview 0 Commits 1 Pipelines 2 Changes 1


Validation de la demande #1




 Request to merge Ops into master

 Detached merge request pipeline #14 failed for 753d94ef

 Merge request approved. Approved by 

 Merged by  Administrator 10 seconds ago [Revert](#) [Cherry-pick](#)
The changes were merged into master with 639dbfb5 
Mentions #1

 Pipeline #15 failed for 639dbfb5 on master

 0  0 

[Oldest first](#) [Show all activity](#)

Une fois la MR faite, on clos le ticket

Terraform

Project overview

Repository

Issues 1

List

Boards

Labels

Service Desk

Milestones

Merge Requests 0

CI / CD

Operations

Packages & Registries

Analytics

Wiki

Collapse sidebar

Administrator Terraform Issues

Open 1 Closed 0 All 1

Recent searches Search or filter results...

Created date 1F

Demande de validation

#1 - opened 27 minutes ago by Administrator Oct 30, 2020

1 1 updated 3 minutes ago

The screenshot shows a GitLab issue page for 'Demande de validation' (Validation Request) in the 'Terraform' project. The issue is marked as 'Open' and was created 31 minutes ago by 'Administrator'. The description is 'Demande de validation du fichier .gitlab-ci.yml.' and it was edited 25 minutes ago by 'Administrator'. There is a section for 'Drop or upload designs to attach' and a 'Linked issues' section. Below that, 'Related merge requests' are listed: 'Draft: Resolve "Demande de validation" 11' and 'Initialisation du fichier .gitlab-ci.yml et ajout du stage test 12'. A note states: 'When these merge requests are accepted, this issue will be closed automatically.' There are two comments from 'Administrator' regarding the due date and description changes. The right sidebar shows various settings like 'Assignees', 'Milestone', 'Time tracking', 'Due date' (Oct 30, 2020), 'Labels', 'Confidentiality', 'Lock issue', and '1 participant'. At the bottom, there are filters for 'Open', 'Closed', and 'All', a search bar, and a list of recent searches. The issue is currently 'CLOSED' and was updated just now.

Lorsque l'on retourne sur Repository > Files, on s'aperçoit que le fichiers .gitlab-ci.yml est bien à jour.

The screenshot shows the 'Repository' page for the 'Terraform' project, specifically the 'Files' view. It displays a list of files with columns for 'Name', 'Last commit', and 'Last update'. The file '.gitlab-ci.yml' is highlighted with a red box, showing its last commit as 'Initialisation du fichier .gitlab-ci.yml et ajout du stage test' from 1 hour ago. Other files listed include 'Makefile', 'README.md', 'lambda_function.zip', 'main.tf', and 'variables.tf'. Below the file list, there is a section for 'README.md' with the text 'Création de document README.md'. The top of the page shows the 'master' branch selected and a 'Clone' button.

Il faut toujours travailler sur une branche différente de la master.

Pour toute la suite on considère que l'on travaille sur la branche Ops et que l'on fait une merge request pour chaque modification ou ajout de fichier.

MIS A JOUR DU FICHIER .GITLAB-CI.YML

Il y a une erreur sur notre fichier .yml du coup on met à jour le fichier

```
stages:
  - test

Terra_test:
  image:
    name: hashicorp/terraform:light
    entrypoint: [""]

  stage: test

  tags:
    - test

  script:
    - terraform init .
    - terraform validate .
```

On procède à nouveau aux étapes pour envoyer sur le repository distant

```
root@debian:/home/user/examen1/Terraform# vim .gitlab-ci.yml
root@debian:/home/user/examen1/Terraform# git add .gitlab-ci.yml
root@debian:/home/user/examen1/Terraform# git commit -m "Mise à jour du fichier .gitlab-ci.yml"
[Ops 5fa39d1] Mise à jour du fichier .gitlab-ci.yml
1 file changed, 3 insertions(+), 2 deletions(-)
root@debian:/home/user/examen1/Terraform# git tag -a vl.1.1 -m "Version 1.1.1"
root@debian:/home/user/examen1/Terraform# git push
remote:
remote: INFO: Your SSH key is expiring soon. Please generate a new key.
remote:
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 340 bytes | 48.00 KiB/s, fait.
Total 3 (delta 2), réutilisés 0 (delta 0)
remote:
remote: To create a merge request for Ops, visit:
remote:   http://192.168.130.245/root/terraform/-/merge_requests/new?merge_request%5Bsource_branch%5D=Ops
remote:
To 192.168.130.245:root/terraform.git
 * [new branch]      Ops -> Ops
```

```
root@debian:/home/user/examen1/Terraform# vim .gitlab-ci.yml
root@debian:/home/user/examen1/Terraform# git add .gitlab-ci.yml
root@debian:/home/user/examen1/Terraform# git tag -a vl.1.3 -m "Version 1.1.3"
root@debian:/home/user/examen1/Terraform# git commit -m "Mise à jour du tag test"
[Ops 8a6e9ac] Mise à jour du tag test
1 file changed, 1 insertion(+), 1 deletion(-)
```

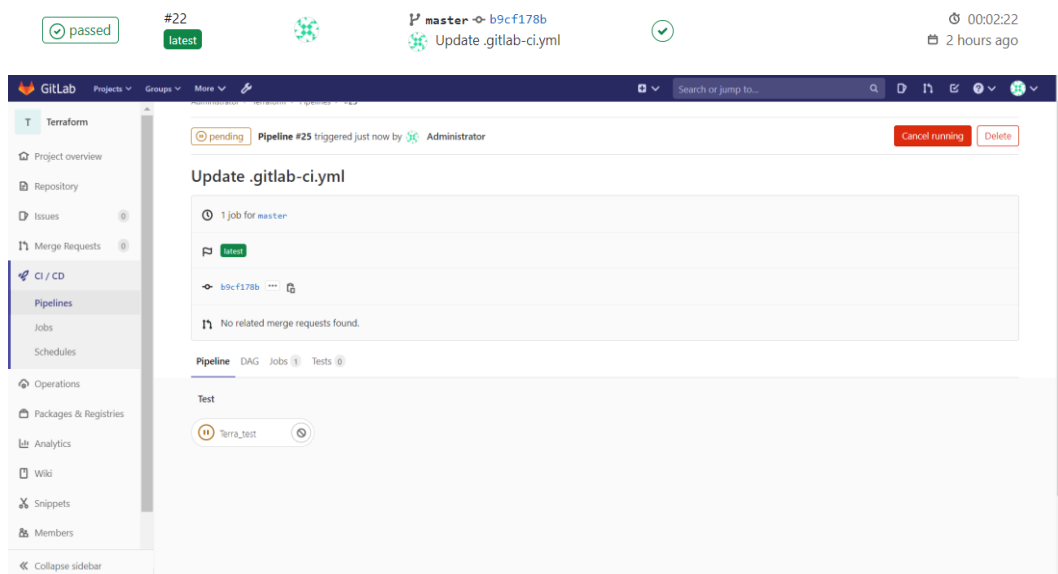
```
root@debian:/home/user/examen1/Terraform# git push -u origin Ops
remote:
remote: INFO: Your SSH key is expiring soon. Please generate a new key.
remote:
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 299 bytes | 42.00 KiB/s, fait.
Total 3 (delta 2), réutilisés 0 (delta 0)
remote:
remote: To create a merge request for Ops, visit:
remote:   http://192.168.130.245/root/terraform/-/merge_requests/new?merge_request%5Bsource_branch%5D=Ops
remote:
To 192.168.130.245:root/terraform.git
 5a048f4..8a6e9ac Ops -> Ops
La branche 'Ops' est paramétrée pour suivre la branche distante 'Ops' depuis 'origin'.
```

```
root@debian:/home/user/examen1/Terraform# git branch
* Ops
  master
```

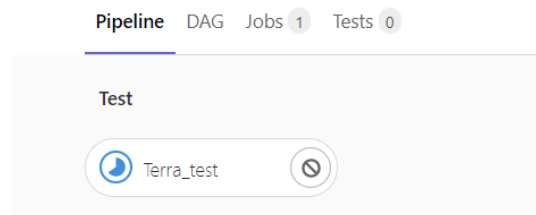
En ligne de commande, on merge la branche Ops sur la branche master avec la commande **git merge Ops**

```
root@debian:/home/user/examen1/Terraform# git merge Ops
Mise à jour e79ced6..8a6e9ac
Fast-forward
 .gitlab-ci.yml | 16 ++++++
 1 file changed, 16 insertions(+)
 create mode 100644 .gitlab-ci.yml
root@debian:/home/user/examen1/Terraform# git merge master
Déjà à jour.
```

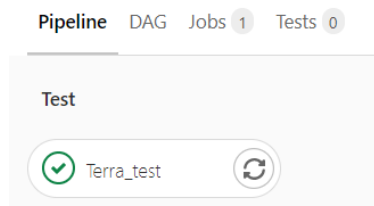
En allant sur le navigateur, on peut suivre l'état de notre Pipeline en cliquant sur l'onglet **CI/CD** puis **Pipelines**



Le job Terra_test en état « **Running** »



Si le job passe sans erreur il passe en état « **Passed** »



On peut également suivre toutes les étapes en cliquant sur [Jobs](#)

```
1 Running with gitlab-runner 13.5.0~beta.80.g1aa4b2fc (1aa4b2fc)
2   on container-test GYGreASH
3   ✓ Preparing the "docker" executor
4     Using Docker executor with image hashicorp/terraform:light ...
5     Pulling docker image hashicorp/terraform:light ...
6     Using docker image sha256:7ea6097f7359410e769ab1596746cae33ee461c91d27e2806ba87b48b829655f for hashicorp/terraform:light with diges
t hashicorp/terraform@sha256:943b1d56d978e713315e1058e1e1297eafdf166c95bb7bbe2712699b0a664f48 ...
7   ✓ Preparing environment 00:01
8     Running on runner-gygreash-project-4-concurrent-0 via debian...
9   ✓ Getting source from Git repository 00:01
10     Fetching changes with git depth set to 50...
11     Reinitialized existing Git repository in /builds/root/terraform/.git/
12     Checking out b9cf178b as master...
13     Skipping Git submodules setup
14   ✓ Executing "step_script" stage of the job script 02:18
15     $ terraform init .
16     Initializing the backend...
17     Initializing provider plugins...
18     - Finding latest version of hashicorp/aws...
19     - Finding latest version of hashicorp/archive...
20     - Installing hashicorp/aws v3.12.0...
21     - Installed hashicorp/aws v3.12.0 (signed by HashiCorp)
22     - Installing hashicorp/archive v2.0.0...
23     - Installed hashicorp/archive v2.0.0 (signed by HashiCorp)
24     The following providers do not have any version constraints in configuration,
25     so the latest version was installed.
26     To prevent automatic upgrades to new major versions that may contain breaking
27     changes, we recommend adding version constraints in a required_providers block
28     in your configuration, with the constraint strings suggested below.
```

On voit ici que le job Terra_test est en **succès**

```
29 * hashicorp/archive: version = "~> 2.0.0"
30 * hashicorp/aws: version = "~> 3.12.0"
31 Terraform has been successfully initialized!
32 You may now begin working with Terraform. Try running "terraform plan" to see
33 any changes that are required for your infrastructure. All Terraform commands
34 should now work.
35 If you ever set or change modules or backend configuration for Terraform,
36 rerun this command to reinitialize your working directory. If you forget, other
37 commands will detect it and remind you to do so if necessary.
38 $ terraform validate .
39 Success! The configuration is valid.
40 Job succeeded
```

Nous allons à présent créer deux autres Runners qui seront :

- Deploy (pour déployer Terraform)
- Destroy (pour le détruire)

On recommence les étapes pour la [création d'un Runner](#)

```
root@debian:/home/user/examen1/Terraform# gitlab-runner register
Runtime platform arch=amd64 os=linux pid=389 revision=1aa4b2fc version=13.5.0~beta.80.g1aa4b2fc
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.130.245/
Please enter the gitlab-ci token for this runner:
mljK5A9Qy4lyShh9_yMN
Please enter the gitlab-ci description for this runner:
[debian]: Deployment
Please enter the gitlab-ci tags for this runner (comma separated):
deploy, apply
Registering runner... succeeded runner=mljK5A9Q
Please enter the executor: parallels, shell, ssh, docker+machine, docker-ssh+machine, custom, docker, docker-ssh, virtualbox, kubernetes:
docker
Please enter the default Docker image (e.g. ruby:2.6):
hashicorp/terraform:light
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

Liste des Runners créés

A 'Runner' is a process which runs a job. You can set up as many Runners as you need. Runners can be placed on separate users, servers, even on your local machine.

Each Runner can be in one of the following states and/or belong to one of the following types:

- shared** - Runner runs jobs from all unassigned projects
- group** - Runner runs jobs from all unassigned projects in its group
- specific** - Runner runs jobs from assigned projects
- locked** - Runner cannot be assigned to other projects
- paused** - Runner will not receive any new jobs

Set up a shared Runner manually

1. Install [GitLab Runner](#)
2. Specify the following URL during the Runner setup: <http://192.168.130.245/>
3. Use the following registration token during setup: [mljK5A9Qy4lyShh9_y8W](#)
4. Start the Runner!

Reset runners registration token

Runners currently online: 5

Type/State	Runner token	Description	Version	IP Address	Projects	Jobs	Tags	Last contact
shared locked	RzyXEShf	Destroy_infra	13.5.0-beta.80...	192.168.130.245	n/a	0	destroy	just now
shared locked	zikjDWhh	Deploy_infra	13.5.0-beta.80...	192.168.130.245	n/a	0	apply deploy	just now
shared locked	GYGreASH	Test_infra	13.5.0-beta.80...	192.168.130.245	n/a	6	clicker-test test	just now

Etapes de création du runner **destroy**

```
root@debian:/home/user/examen1/Terraform# gitlab-runner register
Runtime platform
  arch=amd64 os=linux pid=2094 revision=laa4b2fc version=13.5.0-beta.80.glaa4b2fc
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.130.245/
Please enter the gitlab-ci token for this runner:
mljK5A9Qy4lyShh9_y8W
Please enter the gitlab-ci description for this runner:
[debian]: Destroy
Please enter the gitlab-ci tags for this runner (comma separated):
destroy
Registering runner... succeeded runner=mljK5A9Q
Please enter the executor: docker-ssh, virtualbox, docker+machine, docker-ssh+machine, kubernetes, custom, docker, parallels, shell, ssh:
docker
Please enter the default Docker image (e.g. ruby:2.6):
hashicorp/terraform:light
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
```

Le fichier `.gitlab-ci.yml` avec les 3 jobs créés.

```
stages:
  - test
  - deploy

Terra_test:
  stage: test

  image:
    name: hashicorp/terraform:light
    entrypoint: [""]

  tags:
    - test

  script:
    - terraform init .
    - terraform validate .

Terra_deploy:
  stage: deploy

  image:
    name: hashicorp/terraform:light
    entrypoint: [""]

  tags:
    - apply

  script:
    - terraform init -var="group_id=s1" .
    - terraform apply -var="group_id=s1" -auto-approve .
```


Le fichier .gitlab-ci.yml avec l'ajout de l'artifacts

```
stages:
  - test
  - deploy
  - destroy

Terra_test:
  image:
    name: hashicorp/terraform:light
    entrypoint: [""]
  stage: test
  tags:
    - test
  script:
    - terraform init .
    - terraform validate .

Terra_deploy:
  image:
    name: hashicorp/terraform:light
    entrypoint: [""]
  stage: deploy
  tags:
    - apply
  script:
    - terraform init -var="group_id=s1" .
    - terraform apply -var="group_id=s1" -auto-approve .

  artifacts:
    name:
    paths:
      - terraform.tfstate
    expire_in: 1 month

Terra_destroy:
  stage: destroy
  image:
    name: hashicorp/terraform:light
    entrypoint: [""]
  tags:
    - destroy
  script:
    - terraform init .
    - terraform destroy -var="group_id=s1" -auto-approve .
```

Le job Terra_deploy rencontre une erreur pour un fichier se trouve dans le dossier **Python**

The screenshot shows the GitLab CI/CD interface for a project named 'Terraform'. The pipeline consists of three jobs: Terra_test, Terra_deploy, and Terra_destroy. Terra_test passed successfully. Terra_deploy failed with an error. The error message is displayed in the console output of the Terra_deploy job. The right sidebar shows the job status as 'Failed' and provides details about the job's duration, timeout, and artifacts.

Error Message:

```
42 Error: error configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: InvalidSignature: /20201027/us-east-1/ats/aws_request not a valid keyvalue pair (missing equal-sign) in Authorization header: 'AWS-HMAC-908156 Credential=us-east-1:20201027T00:00:00Z/20201027T00:00:00Z/us-east-1/ats/aws_request, SignedHeaders=content-length,content-type,host;x-amz-date, Signature=af6166ae629517641c940c0166eb0a8279d5acf767e731e9cc16b04dc83f'.
43 status code: 400, request id: 857c1804-237c-438f-9f56-b7715ca51455
44 ERROR: Job failed: exit code 1
```

Le fichier qu'il cherche est **index.py**. Du coup on copie le fichier index dans le répertoire **Terraform**

```
root@debian:/home/user/examen1/Terraform# cd ..
root@debian:/home/user/examen1# ls
Python README.md Terraform
root@debian:/home/user/examen1# cd Python/
root@debian:/home/user/examen1/Python# ls
index.py lambda_function.zip Makefile
root@debian:/home/user/examen1/Python# cp index.py ../Terraform/
root@debian:/home/user/examen1/Python# cd ..
root@debian:/home/user/examen1# cd Terraform/
root@debian:/home/user/examen1/Terraform# ls
index.py lambda_function.zip main.tf Makefile README.md variables.tf
```

Dans le fichier **main.tf**, il faut qu'on modifie également le chemin du fichier **index.py**

```
root@debian:/home/user/examen1/Terraform# vim main.tf

provider "aws" {
    region = var.aws_region
}

data "archive_file" "lambda_zip" {
    type           = "zip"
    source_file    = "../index.py"
    output_path    = "lambda_function.zip"
}
```

Dans ce même fichier **main.tf**, il faut que l'on ajoute des providers aws contenant :

- Access_key
- Secret_key

```
provider "aws" {
    region = var.aws_region
    access_key = $AWS_ACCESS_KEY_ID
    secret_key = $AWS_SECRET_KEY_ID
}

data "archive_file" "lambda_zip" {
    type           = "zip"
    source_file    = "../index.py"
    output_path    = "lambda_function.zip"
}

resource "aws_lambda_function" "this" {
    function_name = "gitlab-examen-${var.group_id}"
    s3_bucket     = aws_s3_bucket.this.bucket
    s3_key        = "lambda_function.zip"
    role          = data.aws_iam_role.this.arn
    handler       = "index.handler"
    source_code_hash = data.archive_file.lambda_zip.output_base64sha256
    runtime       = var.runtime

    tags = {
        Name = var.group_id
    }
}
```

Dans le fichier **variable.tf**, il faut que l'on ajoute les variables initialisées dans le **main.tf**

```
variable "aws_region" {
  default = "eu-west-3"
}

variable "runtime" {
  type    = string
  default = "python3.8"
}

variable "group_id" {
  type    = string
}

variable "aws_access_key" {
  default = $AWS_ACCESS_KEY_ID
}

variable "aws_secret_key" {
  default = $AWS_SECRET_KEY_ID
}
```

Une fois les documents mis à jour, on push tous les documents en respectant pour chaque documents les procédures :

- 1- Git add « nom du fichier »
- 2- Git commit -m « modification fait sur le document »
- 3- Git tag -a v1.X.X -m « Version 1.X.X »
- 4- Git push -u origin Ops
- 5- git checkout master
- 6- git merge Ops

Ci-dessous nous avons toujours une erreur. Je soupçonne le fait que le token soit expiré.

The screenshot displays a GitHub Actions workflow run for a Terraform deployment. The left sidebar shows the repository structure with 'CI / CD' selected. The main panel shows the workflow steps, with the 'Terraform apply' step failing. The error message is: 'Error: error configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: InvalidClientTokenId: The security token included in the request is invalid.' The right sidebar shows the job artifacts and the commit hash 34ea9d31.

```
10 Initializing provider plugins...
11 - Finding latest version of hashicorp/aws...
12 - Finding latest version of hashicorp/archive...
13 - Installing hashicorp/aws v3.13.0...
14 - Installing hashicorp/archive v2.0.0...
15 - Installing hashicorp/archive v2.0.0 (signed by HashiCorp)
16 The following providers do not have any version constraints in configuration,
17 so the latest version was installed.
18 To prevent automatic upgrades to new major versions that may contain breaking
19 changes, we recommend adding version constraints in a required_providers block
20 in your configuration, with the constraint strings suggested below.
21 * hashicorp/archive: version = "~> 2.0.0"
22 * hashicorp/aws: version = "~> 3.13.0"
23 Terraform has been successfully initialized!
24 You may now begin working with Terraform. Try running "terraform plan" to see
25 any changes that are required for your infrastructure. All Terraform commands
26 should now work.
27
28 If you ever set or change modules or backend configuration for Terraform,
29 rerun this command to reinitialize your working directory. If you forget, other
30 commands will detect it and remind you to do so if necessary.
31 $ terraform apply -vars="group_id=1" -auto-approve .
32 data.archive_file.lambda.zip: Refreshing state...
33 Error: error configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: InvalidClientTokenId: The security token included in the request is invalid.
34 status code: 403, request id: cbe9b23a-1f20-4cb6-9489-9e3fc3dfe3be
35 ERROR: Job failed: exit code 1
```