

TP2 devops

Théophile Avenel



1 Intégration continue fournie par Gitlab

Your work / Projects / New project / Create blank project



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL

Project slug

Visibility Level ⓘ

☐ Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal
The project can be accessed by any logged in user except external users.

☒ Public
The project can be accessed without any authentication.

Project Configuration

☒ 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.

☒ Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more.](#)

Figure 1: Création d'un blank project sur Gitlab

Nous commençons par créer un nouveau dépôt dans GitLab puis nous commençons par gérer la configuration pour permettre à l'autre membre du binôme du TP de collaborer.

Pour effectuer cette étape, il faut se rendre dans l'onglet manage puis cliquer sur members et ajouter un user

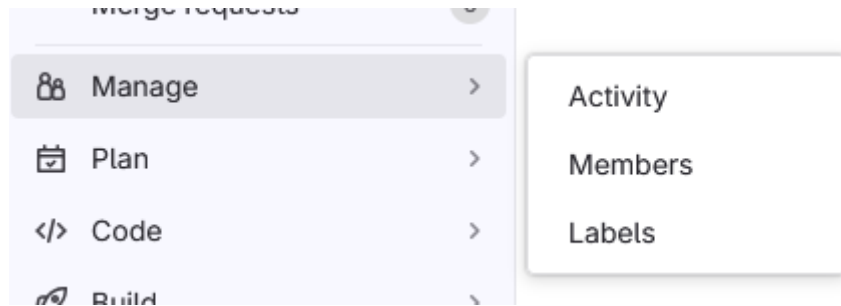


Figure 2: Clic sur l'onglet Manage de Gitlab pour manager les "membres" du repository

Il faut ensuite cliquer sur le bouton invite members

A screenshot of the 'Invite members' modal window in GitLab. The window has a title bar with a close button (X). The main content area includes: a heading 'Invite members'; a sub-header 'You're inviting members to the tpci project.'; a section titled 'Username, name or email address' with a text input field containing 'theophile.avenel'; a prompt 'Select members or type email addresses'; a section titled 'Select a role' with a dropdown menu currently showing 'Maintainer'; a link 'Read more about role permissions'; a section titled 'Access expiration date (optional)' with a date input field showing 'YYYY-MM-DD' and a calendar icon; a link 'Read more about access expiration'; and at the bottom right, two buttons: 'Cancel' and 'Invite'.

Figure 3: Fenêtre d'invitation des membres au repository gitlab

L'utilisateur apparait ensuite comme invité via le pending invitations.

Project members

You can invite a new member to tpci or invite another group.

Import from a project

Invite a group

Invite members

Members 1

Pending invitations 1

Account	Invited	Role	Expiration
 theophile.avenel@gmail.com	Nov 26, 2024 by Theophile AVENEL Awaiting user signup	Maintainer	Expiration date   

Figure 4: Onglet des invitations en attente

Pour faciliter la collaboration, j'ai créé une branche develop ce qui permet de garder la branche main comme branche de production avec livraison du code final.

Deux autres branches ont été créé pour que chaque members puissent avoir une branche de feature (feature-th1 et feature th-2).

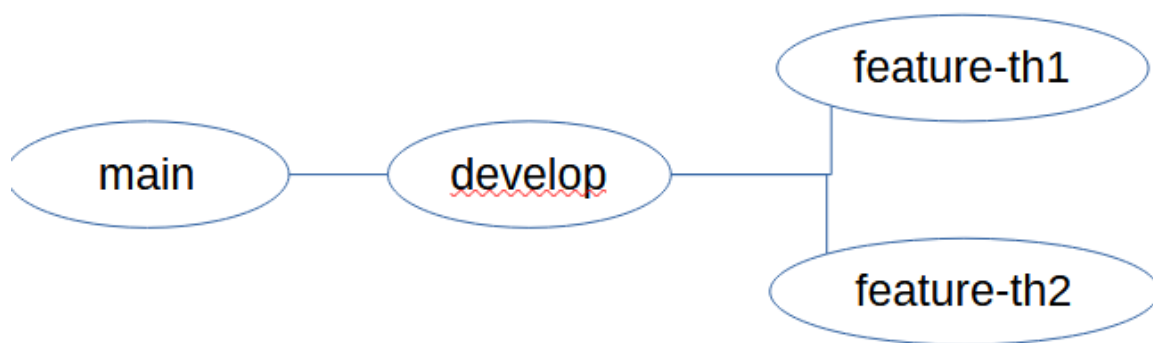


Figure 5: Répartition des branches sur le repository Gitlab

Nous allons commencer par rédiger un code main.c dans la branche feature-th1. Pendant ce temps l'autre membre ajoute main.cpp dans le code feature-th2

Chaque développeur a pu coder sur sa branche son fichier main lors d'une peer review ils ont décidé de merge leur travail sur la branche develop.

Voici un exemple de merge request d'une branche feature sur develop.

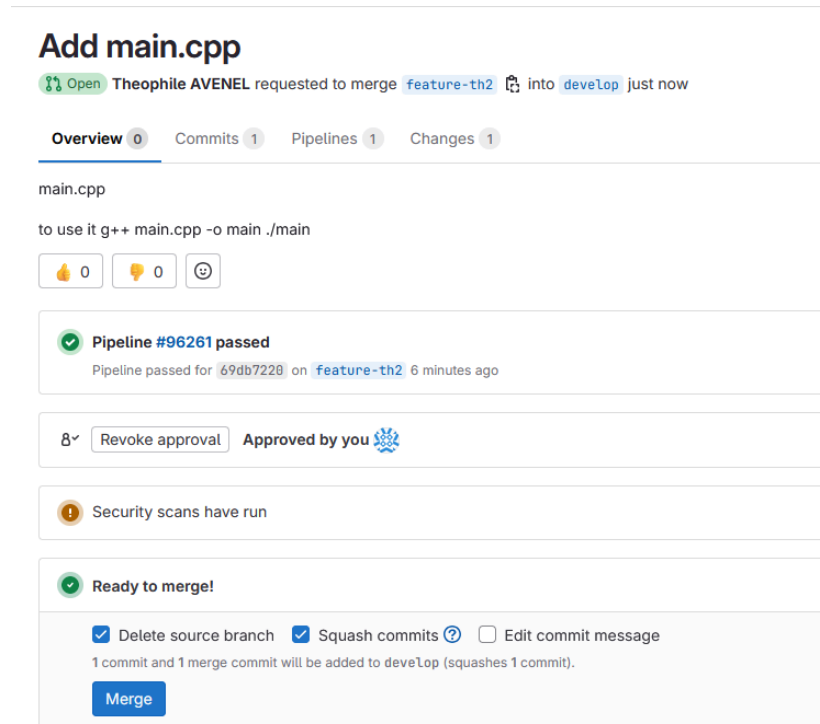


Figure 6: Merge request de la branche feature-th2 vers develop

Voici ci dessous un exemple de résultat de merge request

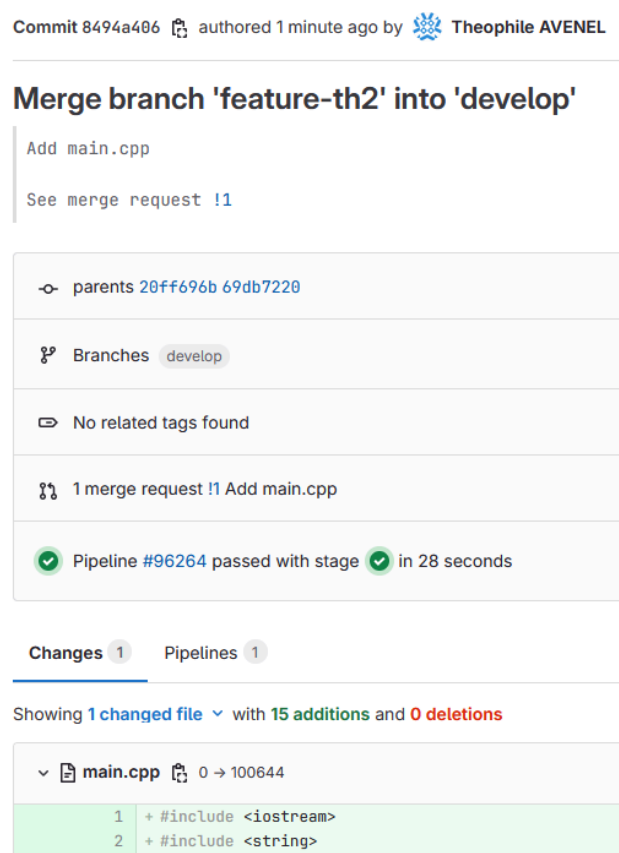


Figure 7: La merge de la branche feature-th2 vers develop a fonctionné avec succès

En se rendant dans l'onglet merge requests nous pouvons cliquer sur l'onglet merged pour voir les merge requests qui ont été effectuées

https://gitlab.univ-nantes.fr/E23C014B/tpci/-/merge_requests?scope=all&state=merged

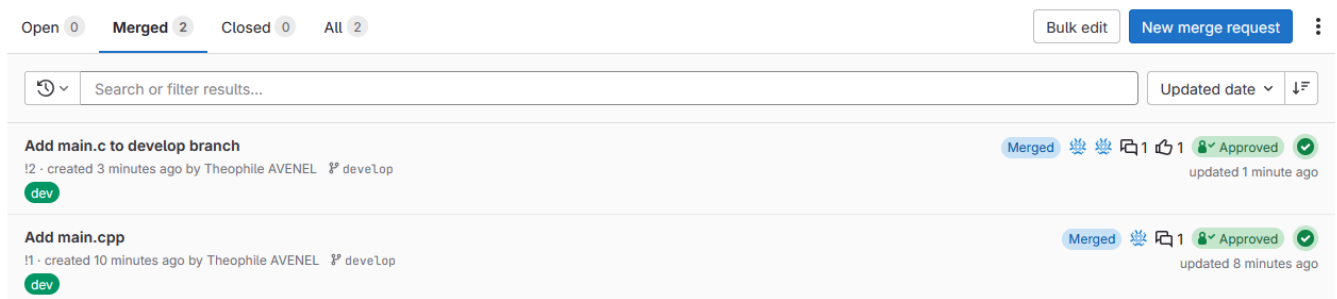


Figure 8: Onglet des merge requests

Voici ci dessous un exemple d'exécution des différents main

```
(base) (athena@DESKTOP-D0DH9M9) - [~/Document/gitwithctp]
$ gcc main.c -o main
./main
Veuillez saisir votre nom : user1
Bonjour, user1

(base) (athena@DESKTOP-D0DH9M9) - [~/Document/gitwithctp]
$ g++ main.cpp -o main
./main
Veuillez saisir votre nom : test
Bonjour, test!
```

Figure 9: Exemple d'exécution des différents main

Pour éviter les problèmes de “ça marche sur ma machine mais pas moi” l'équipe de dev a décidé de faire du pair programming sur la partie de devops.



Figure 10: Figure représentant le pair programming

L'équipe de dev a donc décidé de créer un pipeline de C et un pipeline de C++ pour tester et exécuter le code main.c et le code main.cpp. Ce pipeline CI/CD permettra d'automatiser le déploiement sur les machines de productions.

Voici ci dessous le résultat de ce premier pipeline.

Add artifacts to the test jobs

✓ Passed Theophile AVENEL created pipeline for commit 7636ebc9 4 minutes ago, finished 1 minute ago

For develop

latest 5 jobs 3 minutes 33 seconds, queued for 3 seconds

Pipeline Jobs 5 Tests 0



Figure 11: Résultat du premier pipeline sur Gitlab

Le pipeline de build et de test a fonctionné voici le lien ci-dessous

<https://gitlab.univ-nantes.fr/E23C014B/tpci/-/pipelines/96290>

Une fois que le workflow CI/CD de Gitlab a bien été exécuté, nous pouvons télécharger l'artefact pour essayer de l'exécuter sur notre machine.

Theophile AVENEL / tpci / Pipelines / #96299

Merge branch 'develop' into 'main'

Passed

Theophile AVENEL created pipeline for commit d8eb5786 16 minutes ago, finished 14 minutes ago

For main

latest 60 5 jobs 2 minutes 22 seconds, queued for 2 seconds

Pipeline

Jobs 5

Tests 0

Status	Job	Stage	Coverage
Passed 00:01:17 14 minutes ago	#228200: test_cpp main → d8eb5786	test	
Passed 00:01:13 14 minutes ago	#228199: test_c main → d8eb5786	test	
Passed 00:01:33 14 minutes ago	#228198: semgrep-sast main → d8eb5786	test	
Passed 00:00:47 15 minutes ago	#228197: build_cpp main → d8eb5786	build	
Passed 00:00:49 15 minutes ago	#228196: build_c main → d8eb5786	build	

Figure 12: Liste des jobs



Figure 13: Exécution de l'artefact du job

Comme le montre la figure 13, l'artefact généré par git pour le job est un fichier binaire

Pour vérifier que le pipeline réagit coorectement, j'ai inclu une erreur dans le code main.c pour vérifier que le code ne se compile pas et examiner le message d'erreur du build.

<https://gitlab.univ-nantes.fr/E23C014B/tpci-/jobs/228245>

build_c

Failed Started just now by Theophile AVENEL

```
1 Running with gitlab-runner 17.6.0 (374d34fd)
2 on Main GitLab Runner vEencKxM, system ID: r_11MI9CNhWE5A
3 Preparing the "docker" executor
4 Using Docker executor with image ruby:2.7 ...
5 Pulling docker image ruby:2.7 ...
6 Using docker image sha256:3cb86b8c626861ba5167462228a0835db6b2deae893f3e8fafcf256d9d3abdaa for ruby:2.7 with digest ruby@sha256:2347de892e419c7160fc21dec721d5952736909f8c3fbb7f84cb4a07aaf9ce7d ...
7 Preparing environment
8 Running on runner-veenckxm-project-27053-concurrent-0 via 9c70ca562974...
9 Getting source from Git repository
10 $ git config --global http.proxy $HTTP_PROXY; git config --global https.proxy $HTTPS_PROXY
11 Fetching changes with git depth set to 20...
12 Reinitialized existing Git repository in /builds/E23C014B/tpci/.git/
13 Checking out 4c264783 as detached HEAD (ref is develop)...
14 Removing gl-sast-report.json
15 Removing main_c
16 Removing main_cpp
17 Removing semgrep.sarif
18 Skipping Git submodules setup
19 Executing "step_script" stage of the job script
20 Using docker image sha256:3cb86b8c626861ba5167462228a0835db6b2deae893f3e8fafcf256d9d3abdaa for ruby:2.7 with digest ruby@sha256:2347de892e419c7160fc21dec721d5952736909f8c3fbb7f84cb4a07aaf9ce7d ...
21 $ gcc main.c -o main_c
22 main.c: In function 'main':
23 main.c:11:29: error: 'nom_incorrect' undeclared (first use in this function)
24     11 |     printf("Bonjour, %s\n", nom_incorrect); // Erreur ici : nom_incorrect n'est pas déclaré
25         |                               ~~~~~
26 main.c:11:29: note: each undeclared identifier is reported only once for each function it appears in
27 Cleaning up project directory and file based variables
28 ERROR: Job failed: exit code 1
```

Figure 14: Le pipeline échoue quand on onclu une erreur

L'erreur : **ERROR/ Job failed: exit code 1** montre que le pipeline réagit correctement en cas d'erreur.

Nous avons ensuite pu rajouter les fichiers calcul.c et calcul.cpp tout en modifiant main.c et main.cpp puis nous avons pu modifier le gitlab workflow en effectuant une emrge request final sur la branche main avec un nouveau README.md .

Le repository gitlab est en mode d'accès Public est est accessible depuis l'url :

<https://gitlab.univ-nantes.fr/E23C014B/tpci>

Vous pouvez donc trouver le code des fichiers sur le repository Gitlab.

2 Installation du serveur Jenkins

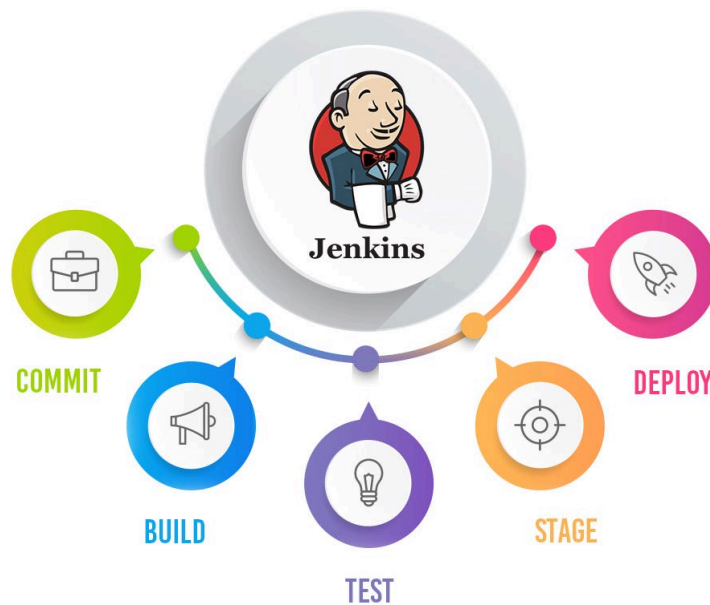


Figure 15: Illustration de jenkins

L'installation dans Kubernetes suppose que nos outils font partie d'un espace de nom dédié pour le développement qui est différent des autres services qui pourront être utilisé dans d'autres aspects. Pour commencer nous pouvons utiliser la commande `minikube start` pour starter minikube.

```
athena@ubuntumate2204:~$ minikube start
🐳 minikube v1.34.0 sur Ubuntu 22.04 (vbox/amd64)
```

Figure 16: Lancement de minikube

Remarque: des erreurs peuvent apparaitre si on ne fait pas de `minikube start`

```
athena@ubuntumate2204:~$ kubectl get serviceaccounts -n devops-tools
Unable to connect to the server: dial tcp 192.168.49.2:8443: connect: no route to host
athena@ubuntumate2204:~$ nano service-account.yaml
athena@ubuntumate2204:~$ minikube status
minikube
type: Control Plane
host: Stopped
kubelet: Stopped
apiserver: Stopped
kubeconfig: Stopped
```

Figure 17: Des erreurs se produisent si on n'effectue pas la commande `minikube start`

Nous commençons par créer un espace de nom nommé `devops-tools`.

L'espace de nom (namespace) permet d'isoler les ressources Kubernetes. Pour créer le namespace `devops-tools`, j'utilise la commande `kubectl create namespace devops-tools`. Pour lister les namespaces j'utilise la commande `kubectl get namespaces`

```

athena@ubuntumate2204:~$ alias kubectl='minikube kubectl --'
athena@ubuntumate2204:~$ kubectl create namespace devops-tools
> kubectl.sha256: 64 B / 64 B [-----] 100.00% ? p/s 0s
> kubectl: 53.77 MiB / 53.77 MiB [-----] 100.00% 3.29 MiB p/s 17s
namespace/devops-tools created
athena@ubuntumate2204:~$ kubectl get namespaces
NAME                STATUS    AGE
default             Active   77m
devops-tools        Active   24s
kube-node-lease     Active   77m
kube-public         Active   77m
kube-system         Active   77m
athena@ubuntumate2204:~$ █

```

Figure 18: Création d'un alias pour appeler kubectl avec minikube

Nous allons ensuite créer un service administrateur à partir du fichier service-account.yaml.

Pour réaliser cette étape, j'ai pu configurer le fichier service-account.yaml en définissant un ServiceAccount nommé admin-service-account dans le namespace devops-tools. J'ai ensuite pu "appliquer" ce fichier avec la commande `kubectl apply -f service-account.yaml -n devops-tools`. Pour vérifier sa création j'ai pu utiliser `kubectl get serviceaccounts -n devops-tools`, où admin-service-account devrait apparaître dans la liste.

```

athena@ubuntumate2204:~$ kubectl apply -f service-account.yaml -n devops-tools
serviceaccount/jenkins-admin created
role.rbac.authorization.k8s.io/jenkins created
rolebinding.rbac.authorization.k8s.io/jenkins-role-binding created
athena@ubuntumate2204:~$ kubectl get serviceaccounts -n devops-tools
NAME                SECRETS    AGE
admin-service-account  0          162m
default             0          173m
jenkins-admin        0          15s
athena@ubuntumate2204:~$ █

```

Figure 19: Application du service-account.yaml au kubectl

J'ai ensuite pu créer "un secret" pour accéder au contrôleur en appliquant le fichier secret.yaml

Pour réaliser cette étape, j'ai pu configurer le fichier secret.yaml en y incluant l'username et le mot de passe encodées en base64, comme le nom d'utilisateur et le mot de passe. Avec la commande `kubectl apply -f secret.yaml -n devops-tools` j'ai pu "appliquer" le secret au namespace devops-tools. Pour vérifier sa création j'ai pu exécuter la commande `kubectl get secrets -n devops-tools`, où jenkins-secret devrait être listé.

```

athena@ubuntumate2204:~$ kubectl apply -f secret.yaml -n devops-tools
secret/sa-token-secret created
athena@ubuntumate2204:~$ kubectl get secrets -n devops-tools
NAME                TYPE              DATA  AGE
jenkins-secret      Opaque            2      159m
sa-token-secret     kubernetes.io/service-account-token  3      14s
athena@ubuntumate2204:~$ █

```

Figure 20: Application du secret.yaml au kubectl

Une fois que les étapes précédentes ont été réalisés, j'ai pu créer un déploiement à partir du fichier deployment.yaml

J'ai donc pu configurer le fichier deployment.yaml pour déployer Jenkins dans le namespace devops-tools, en définissant un seul réplica, l'image officielle de Jenkins, les ports nécessaires et le montage d'un volume pour les données persistantes. J'ai pu appliquer ce changement avec la commande `kubectl apply -f deployment.yaml -n devops-tools`. Pour vérifier le déploiement et l'état des pods j'ai pu exécuter `kubectl get deployments -n devops-tools` et `kubectl get pods -n devops-tools`, en vous assurant que le pod Jenkins est en état Running.

```

athena@ubuntumate2204:~$ kubectl apply -f deployment.yaml -n devops-tools
persistentvolumeclaim/jenkins-pv-claim created
deployment.apps/jenkins-deployment configured
service/jenkins-service configured
athena@ubuntumate2204:~$ kubectl get deployments -n devops-tools
NAME                READY  UP-TO-DATE  AVAILABLE  AGE
jenkins-deployment  1/1    1           1          157m
athena@ubuntumate2204:~$ kubectl get pods -n devops-tools
NAME                READY  STATUS    RESTARTS  AGE
jenkins-deployment-659f779669-62krq  1/1    Running   1 (91m ago)  158m
jenkins-deployment-769696f4b4-hh8s9  0/1    Running   0          30s
athena@ubuntumate2204:~$

```

Figure 21: Application du deployment.yaml au kubectl

Nous pouvons désormais passer à l'étape d'identification de l'adresse du service Jenkins et l'étape d'accès au dashboard.

Pour cela j'ai créé le fichier service.yaml

```

GNU nano 6.2 service.yaml *
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
  namespace: devops-tools
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30000 # Vous pouvez choisir un autre port disponible
  selector:
    app: jenkins

```

Figure 22: Le fichier service.yaml

Ce fichier `service.yaml` permet de définir un Service Kubernetes nommé `jenkins-service`, configuré pour exposer une application Jenkins. Le champ `apiVersion: v1` indique qu'il utilise la première version de l'API Kubernetes pour cet objet. Le `kind: Service` précise qu'il s'agit d'un service, utilisé pour rendre les pods accessibles. Sous `metadata`, le nom du service est spécifié comme `jenkins-service`, et il est associé au namespace `devops-tools`. Dans la section `spec`, le type de service est défini comme `NodePort`, permettant l'accès à Jenkins depuis l'extérieur du cluster via un port spécifique. La configuration des ports indique que le port interne du service (8080) est mappé au `nodePort: 30000`, qui est un port accessible sur les nœuds du cluster. Enfin, le `selector: app: jenkins` relie ce service aux pods ayant le label `app: jenkins`, garantissant que le trafic est acheminé vers les pods Jenkins correspondants.

J'ai ensuite pu appliquer le contenu du `service.yaml` au `kubectl` via la commande `kubectl apply -f service.yaml -n devops-tools`.

```

athena@ubuntumate2204:~$ kubectl apply -f service.yaml -n devops-tools
service/jenkins-service created

```

Figure 23: Application du service.yaml au kubectl

Une fois que j'ai pu appliquer toutes les configurations précédentes, j'ai pu me connecter au Jenkins via la commande `minikube service jenkins-service -n devops-tools --url`

```

athena@ubuntumate2204:~$ minikube service jenkins-service -n devops-tools --url
http://192.168.49.2:30000

```

Figure 24: Lancement de Jenkins et de l'attribution de l'url

Quand je me rends sur l'url <http://192.168.49.2:30000> j'arrive sur une page m'invitant à "Débloquer Jenkins"

Démarrage

Débloquer Jenkins

Pour être sûr que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs (où le trouver) ainsi que dans ce fichier sur le serveur :

```
/var/jenkins_home/secrets/initialAdminPassword
```

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

Figure 25: Fenêtre pour débloquer Jenkins

J'ai pu chercher le mot de passe en cherchant le pod associé à devops-tools avec la commande `kubectl get pods -n devops-tools`. Une fois le nom obtenu j'ai pu exécuter la commande `kubectl logs <nom_pod> -n devops-tools` et j'ai pu trouver le mot de passe.

```
athena@ubuntumate2204:~$ kubectl get pods -n devops-tools
NAME                                READY   STATUS    RESTARTS   AGE
jenkins-deployment-659f779669-62krq 1/1     Running   0          36m
athena@ubuntumate2204:~$ kubectl logs jenkins-deployment-659f779669-62krq -n devops-tools
```

Figure 26: Commande pour obtenir le mot de passe administrateur de Jenkins

```
*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password
generated.
Please use the following password to proceed to installation:

b88290157c45449ca6f8cf49b2230ba9

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

Figure 27: Obtention du mot de passe administrateur de Jenkins

J'ai donc pu rentrer le mot de passe "b88290157c45449ca6f8cf49b2230ba9" pour le mot de passe administrateur et je peux désormais passer à l'étape de configuration.

3 Configurer

J'ai pu passer à la partie configuration du Jenkins et j'ai pu choisir l'installation des plugins suggérés automatiquement.

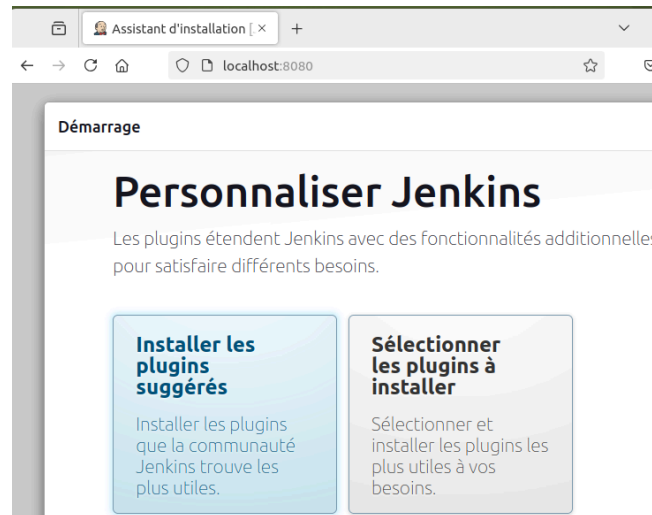


Figure 28: Sélection de l'onglet Installer les plugins suggérés

Cependant de nombreuses erreurs sont apparus en raison de problème de connexion internet

```
figuration#download: Downloading ldap
2024-11-26 13:52:44.304+0000 [id=83] SEVERE h.model.UpdateCenter$DownloadJob
run: Failed to install ldap
java.io.IOException: Failed to load: LDAP Plugin (ldap 770.vb_455e934581a_)
- Failed to load: Jenkins Mailer Plugin (mailer 489.vd4b_25144138f)
  at hudson.PluginWrapper.resolvePluginDependencies(PluginWrapper.java:992)
    at hudson.PluginManager.dynamicLoad(PluginManager.java:982)
Caused: java.io.IOException: Failed to install ldap plugin
  at hudson.PluginManager.dynamicLoad(PluginManager.java:996)
  at hudson.model.UpdateCenter$InstallationJob._run(UpdateCenter.java:2379)
Caused: java.io.IOException: Failed to dynamically deploy this plugin
  at hudson.model.UpdateCenter$InstallationJob._run(UpdateCenter.java:2383)
    at hudson.model.UpdateCenter$DownloadJob.run(UpdateCenter.java:2012)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Unknown
Source)
    at java.base/java.util.concurrent.FutureTask.run(Unknown Source)
    at hudson.remoting.AtMostOneThreadExecutor$Worker.run(AtMostOneThreadExe
utor.java:121)
    at java.base/java.lang.Thread.run(Unknown Source)
2024-11-26 13:52:44.304+0000 [id=83] INFO h.model.UpdateCenter$DownloadJob
run: Starting the installation of ldap out on behalf of admin
```

Figure 29: Erreur de connexion internet dans les logs

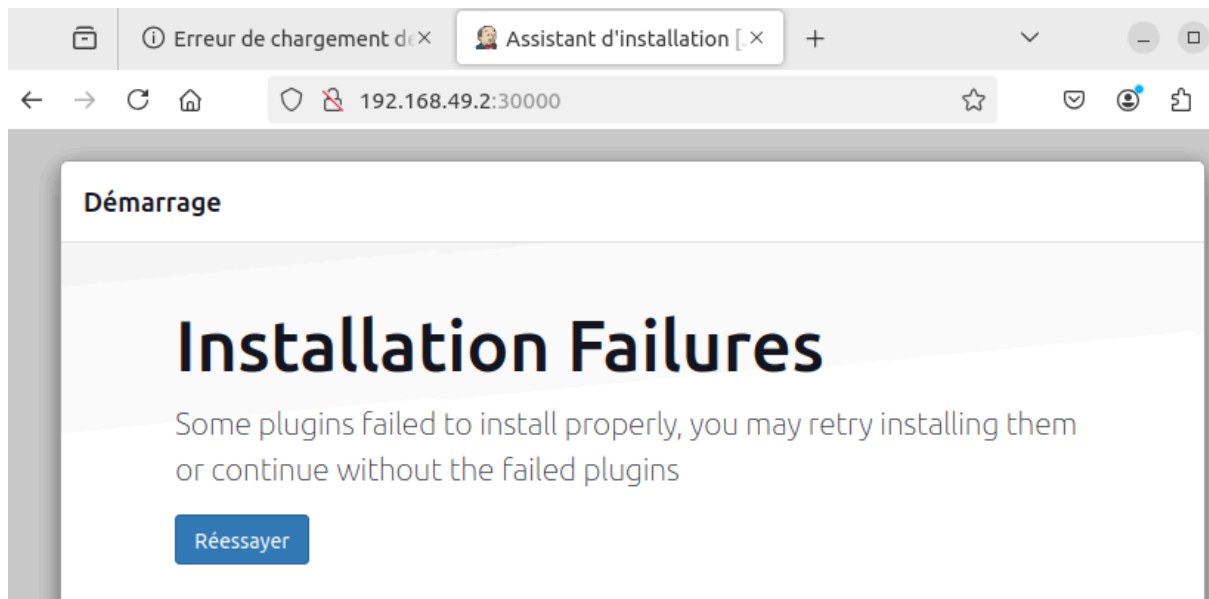


Figure 30: Erreur de connexion internet lors de l'installation des plugins

Après de nombreux clics sur le bouton Réessayer les plugins se sont installés et j'ai pu passer à la création du premier utilisateur Administrateur

A screenshot of the Jenkins installation assistant's 'Créer le 1er utilisateur Administrateur' (Create the first administrator user) screen. The page is titled 'Démarrage' (Getting Started). The main heading is 'Créer le 1er utilisateur Administrateur'. Below the heading, there are five input fields: 'Nom d'utilisateur' (Username) with the value 'admin', 'Mot de passe' (Password) masked with dots, 'Confirmation du mot de passe' (Confirm password) masked with dots, 'Nom complet' (Full name) with the value 'admin', and 'Adresse courriel' (Email address) with the value 'admin@admin.com'. At the bottom right, there is a blue button labeled 'Sauver et continuer' (Save and continue). A small inset window shows the same screen, and a status bar at the bottom indicates 'Assistant d'installation [Jenkins] — Mozilla Firefox'.

Figure 31: Création du premier utilisateur Administrateur

Une fois cette étape de création de compte administrateur réalisé je suis arrivé sur le tableau de bord de Jenkins.

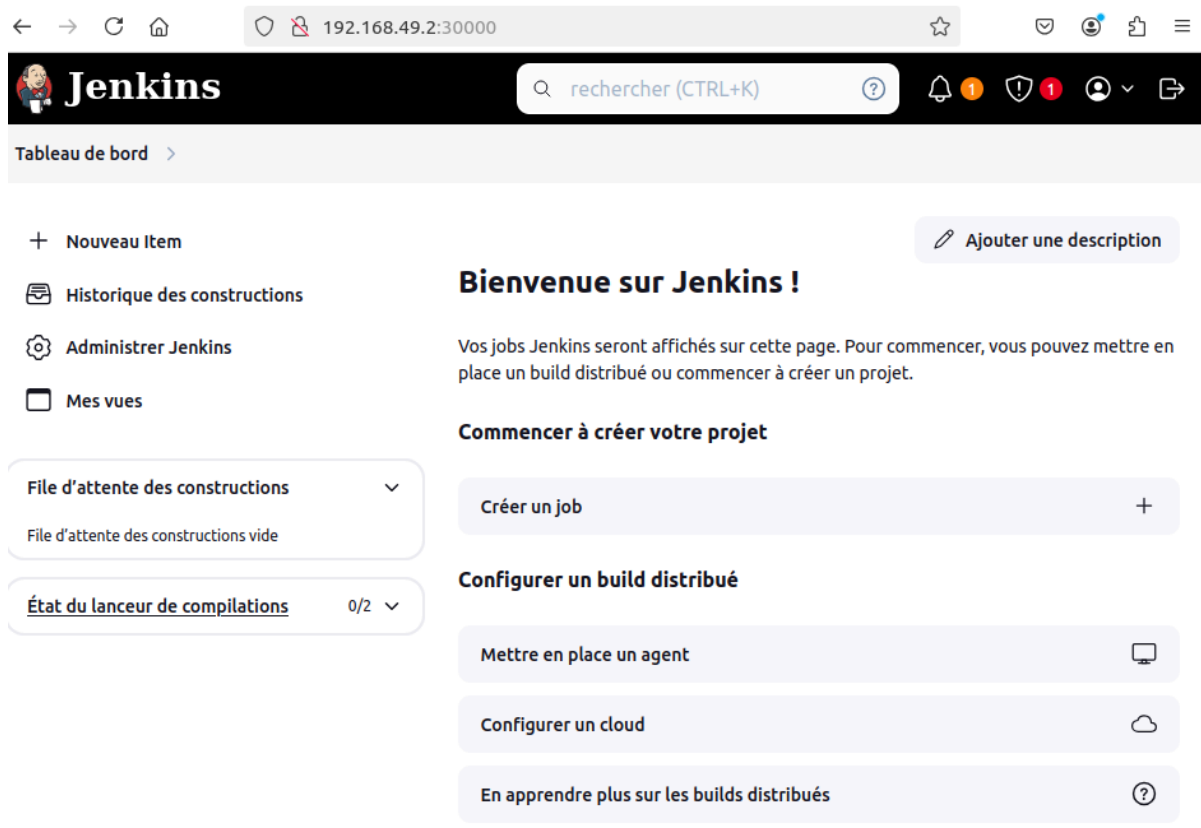


Figure 32: Page d'accueil de Jenkins

3.1 Cloud

J'ai ensuite pu installer le plugin cloud pour gérer les builds dans un cluster Kubernetes.

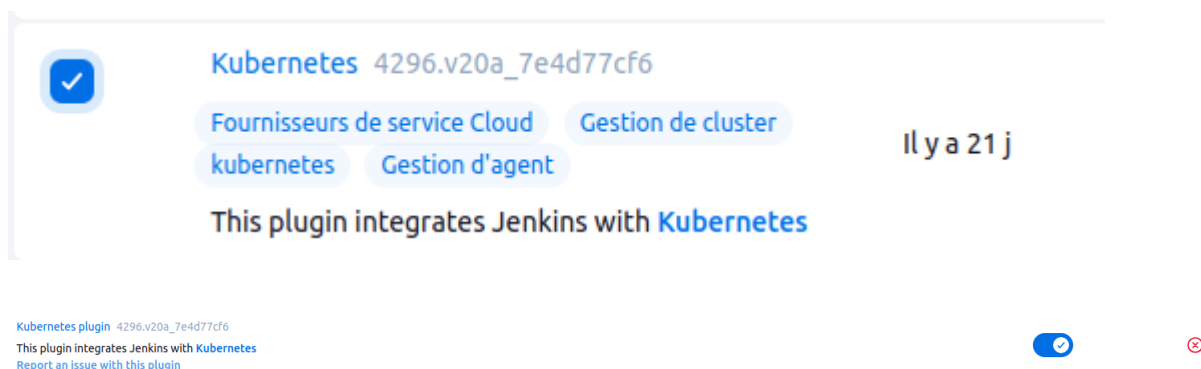


Figure 33: Installation du plugin cloud Kubernetes

J'ai ensuite pu configurer le cloud en ajoutant l'adresse IP du nœuds Jenkins dans le champ jenkins URL. J'ai pu appeler ce nouveau cloud could-kubernetes et j'ai pu lui donner le type Kubernetes.

New cloud

Cloud name

cloud-kubernetes

Type



Kubernetes

Create

Figure 34: Création du cloud “cloud-kubernetes” de type Kubernetes

Une fois que le cloud a été créé, j’ai pu m’attaquer à la partie configuration de ce cloud.

Configuration du cloud cloud-kubernetes

Name ?

cloud-kubernetes

Kubernetes URL ?

http://192.168.49.2:30000/

Figure 35: Configuration du cloud “cloud-kubernetes” de type Kubernetes

Pour ajouter le certificat j’ai exécuter la commande minikube ssh “sudo cat /var/lib/minikube/certs/ca.crt” > ca.crt et afficher son contenu

```

athena@ubuntumate2204:~$ minikube ssh "sudo cat /var/lib/minikube/certs/ca.crt"
> ca.crt
athena@ubuntumate2204:~$ more ca.crt
-----BEGIN CERTIFICATE-----
MIIDBjCCAE6gAwIBAgIBATANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEWptaW5p
a3ViZUNBMB4XDTE0MTEyNTYwMVVoXDTM0MTEyNDEwNTYwMVowFTETMBEGA1UE
AxMKbWluaWt1YmVDTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANpy
5CZR14gKyji9NPskVYXJKuRXDwnLE10frSGDxTxNNWPopL8Xo6SauT0zVn3dADry
z0yRPK67cYoig+bI/xT/ZbjLCDHwKFgmTxBp/SMpHwnlK9rDiOXmaZEz6KqTnPwk
Oi0T0PYvDiR7jB0QemexkYNNK0mmYfa8ge056LeUkkRG9BXL3GeU87n5LkcQablGf
ciFqxbtg+LzSwsvCiFmKAqlbzmiKNDDGomxn7lWfCIz4R+cbv3Izqm4FmT27nyuU
bw6tktrS4k4v79QSaQzJ4EaeicCgz1A1CvjryUTtX0a9RNq5bGNKH0HCTWPg+kBf
06bANTgz2LZhBIaVgNECAwEAAANhMF8wDgYDVVR0PAQH/BAQDAgKkMB0GA1UdJQQW
MBQGCCsGAQUFBwMCBggrBgEFBQcDATAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQW
BBS6pFjwezCzx/WL08ps0X0p85DynTANBgkqhkiG9w0BAQsFAA0CAQEAR35owaMu
gzXIj3JrtLaCjuosjo6Ii+4TkWDUxmb/brmSE6bTHTdR0+PUs0ADprEgNHSXcxmj
NXMjLAYA8h8NhDn0BoTKwWOUInpq+renpJSbCYWj/64ugKi2hNl5ikCOMb5kIMbm
eQGJ5M88HBkX9RvHKzBh7Dff4LCfBkONiRNgaIUzTzVpLQ25whjY3r1T6Lg5fjqV
zUek9jwJtKE40NBVQYD7DcclqK+9bxteeTExQkE9Mq+0S1vR05oXrsldeD146HSU
7wwNuOwf91F3+/hpEJLMrJ64nSXXtTfWV3nKWx2ohMZbxVS5vwK04SqaK4skRMDm
0VEbch07F36kEQ==
-----END CERTIFICATE-----

```

Figure 36: Certificat attaché à minikube

J'ai pu ensuite cliquer sur le bouton Test Connection pour vérifier que la connexion fonctionnait correctement mais j'ai eu une erreur..

HTTP ERROR 403 No valid crumb was included in the request

Test Connection

URI: /manage/descriptorByName/org.csanchez.jenkins.plugins.kubernetes.KubernetesCloud/testConnection

STATUS: 403

MESSAGE: No valid crumb was included in the request

SERVLET: Stapler

Powered by Jetty:// 12.0.13

Figure 37: Erreur 403

J'ai donc pu effectuer un refresh pour corriger l'erreur

New cloud

Name ?

kubernetes-cloud

Kubernetes URL ?

☐ Use Jenkins Proxy ?

Kubernetes server certificate key ?

```
-----BEGIN CERTIFICATE-----
MIIDBJCCAe6gAwIBAgIBATANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwptaW5p
a3ViZUNBMB4XDTI0MTEyNTUwNTYwMVVoXDTM0MTEyNDEwNTYwMVowFTETMBEGA1UE
AxMKbWluaWt1YmVDTCCASiWdQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANpy
5CZR14gKyji9NPskVYXJKuRXDwnLE10frSGDxTxNNWPopL8Xo6SauT0zVn3dADry
70uDPK67cVoig+hI/xT/7hilCDHwKEgmTxBo/SMoHWnlK9rDiQYma7E76KqToPwK
-----END CERTIFICATE-----
```

☐ Disable https certificate check ?

Figure 38: Ajout du certificat de minikube au cloud "kubernetes-cloud"

Le test de connexion c'est révélé concluant.

Connected to Kubernetes v1.31.0

Test Connection

Figure 39: Le Test Connection a fonctionné

Comme le montre la figure 39 ci-dessus, la connexion fonctionné car nous obtenons le résultat "Connected to Kubernetes v1.31.0"

3.2 Agents de builds

Les tâches des builds sont des tâches lourdes d'habitude, il est donc préférable de consacrer des pods dédiés pour les exécuter.

Le noeud correspondant au contrôleur ne doit pas épuiser ces ressources pour effectuer les tâches libres comme les builds. Dans la section correspondante à la gestion de Jenkins, modifier les paramètres pour que le contrôleur n'exécute pas les tâches des builds.

The screenshot shows the Jenkins web interface at the URL `192.168.49.2:30000/computer/(built-in)/configure`. The page title is "Jenkins". The breadcrumb navigation is "Tableau de bord > Nœuds > contrôleur > Configurer". On the left sidebar, there are links for "Statut", "Configurer" (active), "Historique des constructions", "Statistiques d'utilisation", and "Console de script". Below the sidebar, there is a section "État du lanceur de compilations" with a dropdown arrow and the text "No executors, agents or clouds are configured." The main content area has a "Nombre d'exécuteurs" field set to "0", an "Étiquettes" field, and an "Utilisation" dropdown set to "Réserver ce nœud uniquement pour les jobs qui lui sont attachés". Below these is a section "Propriétés du Nœud" with three checkboxes: "Disable deferred wipeout on this node", "Disk Space Monitoring Thresholds", and "Variables d'environnement". At the bottom is a blue "Sauvegarder" button.

Figure 40: Modification de la configuration du contrôleur

Ensuite, on va s'intéresser à configurer un agent pour les builds. J'ai donc pu créer un nouvel agent dans l'espace de noms Jenkins

The screenshot shows the Jenkins web interface at the URL `192.168.49.2:30000/computer/new`. The page title is "Jenkins". The breadcrumb navigation is "Tableau de bord > Nœuds > New node". The main content area is titled "New node". It has a "Nom du nœud" field containing "jenkins-agent". Below it is a "Type" section with a radio button selected for "Permanent Agent". The text below the radio button reads: "Ajoute un agent complet permanent à Jenkins. Il est appelé 'permanent' car Jenkins ne fournit pas d'intégration de plus haut niveau avec ces agents, comme des provisionnements dynamiques. Sélectionnez ce type si aucun autre type d'agent ne s'applique - par exemple lorsque vous ajoutez un ordinateur physique, une machine virtuelle gérée hors de Jenkins, etc." At the bottom is a blue "Create" button.

Figure 41: Ajout du nouveau "node" de type Permanent Agent

Il faut cocher la checkbox permanent agent pour pouvoir cliquer sur le bouton Create. Pour pouvoir le faire pour effectuer les builds j'ai pu associer un label kubeagent à l'agent.



Jenkins

rechercher (CTRL+K) ?

Tableau de bord > Administrer Jenkins > Clouds > kubernetes-cloud > New pod template

Status

Pod Templates

Configure

Delete Cloud

New pod template settings

Name ?

jenkins-pod

Namespace ?

devops-tools

Labels ?

kubeagent

Usage ?

Réserver ce nœud uniquement pour les jobs qui lui sont attachés

Pod template to inherit from ?

Name of the container that will run the Jenkins agent ?

jenkins-container

Figure 42: Création du nouveau pod template de nom jenkins-pod

J'ai ensuite pu configurer le template avec le nom jnlp avec l'image jenkins/inbound-agent:latest et le working directory /home/jenkins/agent

☐ Inject Jenkins agent in agent container ?

Containers ?

List of container in the agent pod

Container Template

Name ?

inp

Docker image ?

jenkins/inbound-agent:latest

☐ Always pull image ?

Working directory ?

/home/jenkins/agent

Command to run ?

sleep

Arguments to pass to the command ?

9999999

☐ Allocate pseudo-TTY ?

Environment Variables ?

List of environment variables to set in agent pod

Create

Figure 43: Création et configuration du contianer template

Le pod templates est désormais visible dans la liste des pods tempaltes



Figure 44: Le pod jenkins-pod apaprait dans les pods templates

4 Programmer des tâches

4.1 Tâche de test

Nous allons désormais créer une tâche pour tester si les agents de builds peuvent être lancés correctement en affichant simplement le message "succès" comme étape de build unique. Pour cela lors de la création du nouvel item, j'ai cliqué sur Pipeline

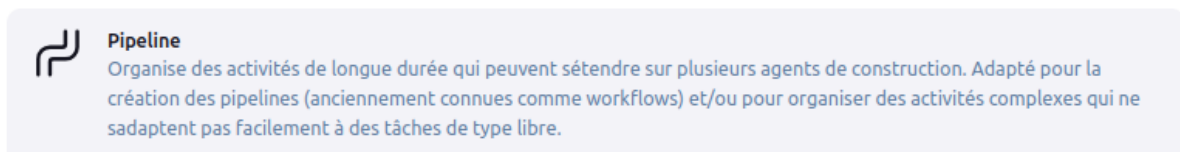


Figure 45: Sélection d'un nouvel item de type Pipeline

J'ai ensuite pu rentrer le Pipeline Script

Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {  
2   agent any  
3   stages {  
4     stage('Test Agent') {  
5       steps {  
6         echo 'Succès'  
7       }  
8     }  
9   }  
10 }  
11
```

try sample Pipeline...

☒ Use Groovy Sandbox ?

Figure 46: Création du Pipeline Script

Cela a pu générer un jenkins pipeline comme ci dessous

jenkins-pipeline

 Ajouter une description

Liens permanents

- [Dernier build \(#1\), il y a 9 mn 29 s](#)

Figure 47: Création du Jenkins Pipeline

Le pipeline attend que l'agent se réveille

État du lanceur de compilations

0/1 ▼



jenkins-agent

(déconnecté)

Figure 48: Etat déconnecté de l'agent jenkins-agent

L'agent jenkins-agent est actuellement dans un état déconnecté, il faut donc exécuter quelques commandes sur VM qui exécute le jenkins

```
athena@ubuntu2204:~$ curl -sO http://192.168.49.2:30000/jnlpJars/agent.jar
athena@ubuntu2204:~$ echo 159aa1ef08510bbccaefc9afd53a2da257d00f87fc0c69cf3601debee47369ae > secret-file

athena@ubuntu2204:~$ java -jar agent.jar -url http://192.168.49.2:30000/ -secret @secret-file -name "jenkins-agent" -webSocket -workDir "/home/athena/jenkins-agent-workdir"
nov. 26, 2024 7:47:52 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/athena/jenkins-agent-workdir/remoting as a remoting work directory
nov. 26, 2024 7:47:52 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/athena/jenkins-agent-workdir/remoting
nov. 26, 2024 7:47:52 PM hudson.remoting.Launcher createEngine
INFO: Setting up agent: jenkins-agent
nov. 26, 2024 7:47:52 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3261.v9c670a_4748a_9
nov. 26, 2024 7:47:52 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/athena/jenkins-agent-workdir/remoting as a remoting work directory
nov. 26, 2024 7:47:52 PM hudson.remoting.Launcher$CuiListener status
INFO: WebSocket connection open
nov. 26, 2024 7:47:52 PM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

Figure 49: Exécution des commandes pour réveiller l'agent jenkins-agent

Une fois que ces commandes ont été exécuté l'agent est désormais actif et peut répondre aux tâches de build en attente d'être traité.

Il faut vérifier que l'agent dispose d'un répertoire de travail comme sur la figure ci-dessous.

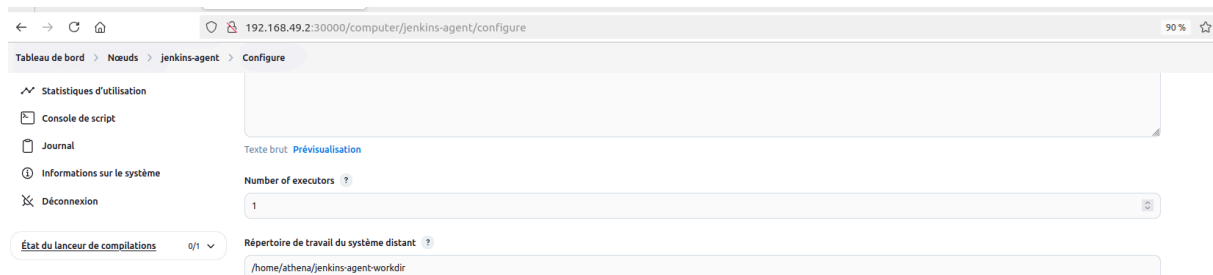


Figure 50: Ajout du répertoire de travail du système distant pour l'agent jenkins-agent

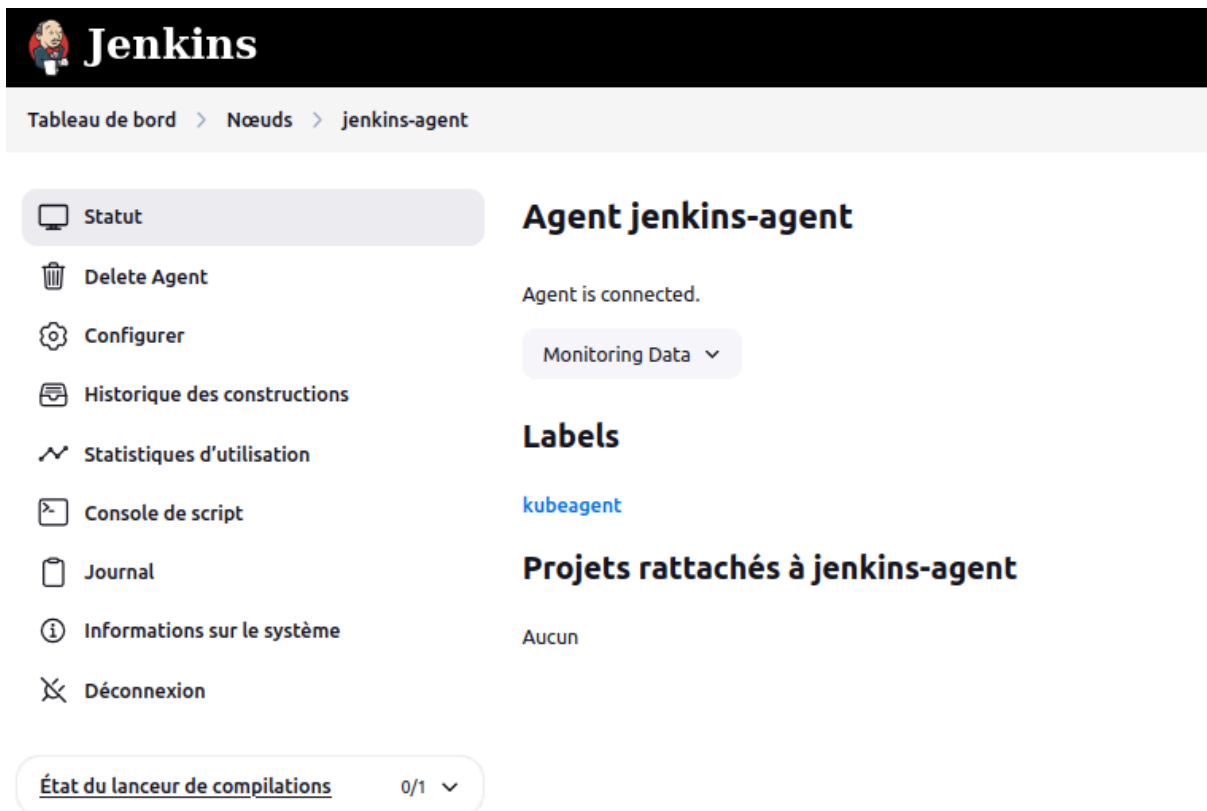


Tableau de bord > Nœuds > jenkins-agent

Agent jenkins-agent

Agent is connected.

Monitoring Data ▾

Labels

kubeagent

Projets rattachés à jenkins-agent

Aucun

État du lanceur de compilations 0/1 ▾

Figure 51: Affichage de l'agent jenkins-agent qui est désormais en état connecté



✓ #1 (26 nov. 2024, 18:32:26) [Ajouter une description](#) [Conserver cette construction sans limite de temps](#)

Lancé par l'utilisateur [test](#) Démarrée il y a 15 mn. A duré 15 mn

This run spent:

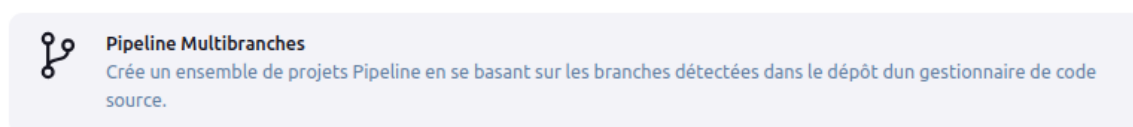
- 15 mn waiting;
- 15 mn build duration;
- 15 mn total from scheduled to completion.

</> [Aucun changement](#)

Figure 52: Lancement du pipeline Jenkins qui peut désormais être exécuter par un agent

4.2 Tâche à partir d'un Git

J'ai ensuite pu créer une tâche de build qui doit être exécuté lors d'un commit sur mon repository gitlab. Comme j'ai la branche develop et la branche main, j'ai choisi de créer un item Pipeline Multibranches.



Pipeline Multibranches

Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

Figure 53: Création d'un nouvel item de type Pipeline Multibranches

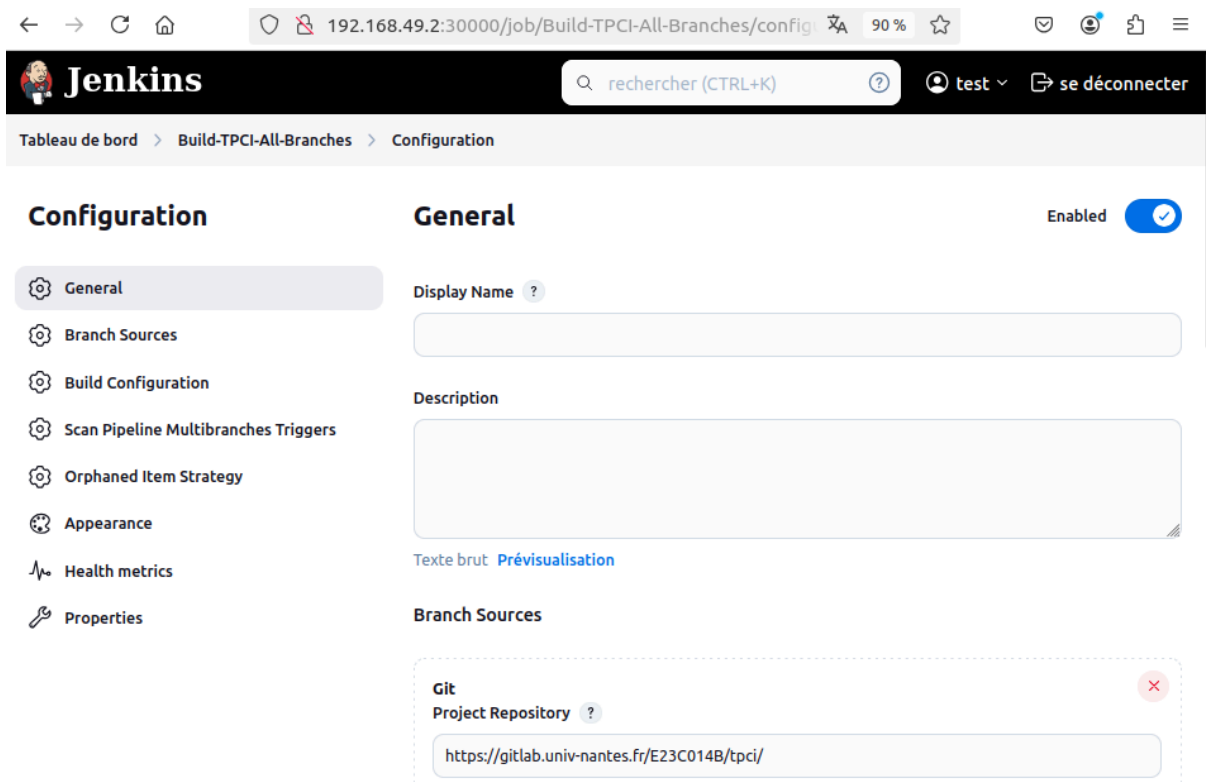


Figure 54: Création d'un pipeline multibranche tapant dans le repository gitlab du projet

J'ai ensuite pu réaliser différents commits dans mes branches

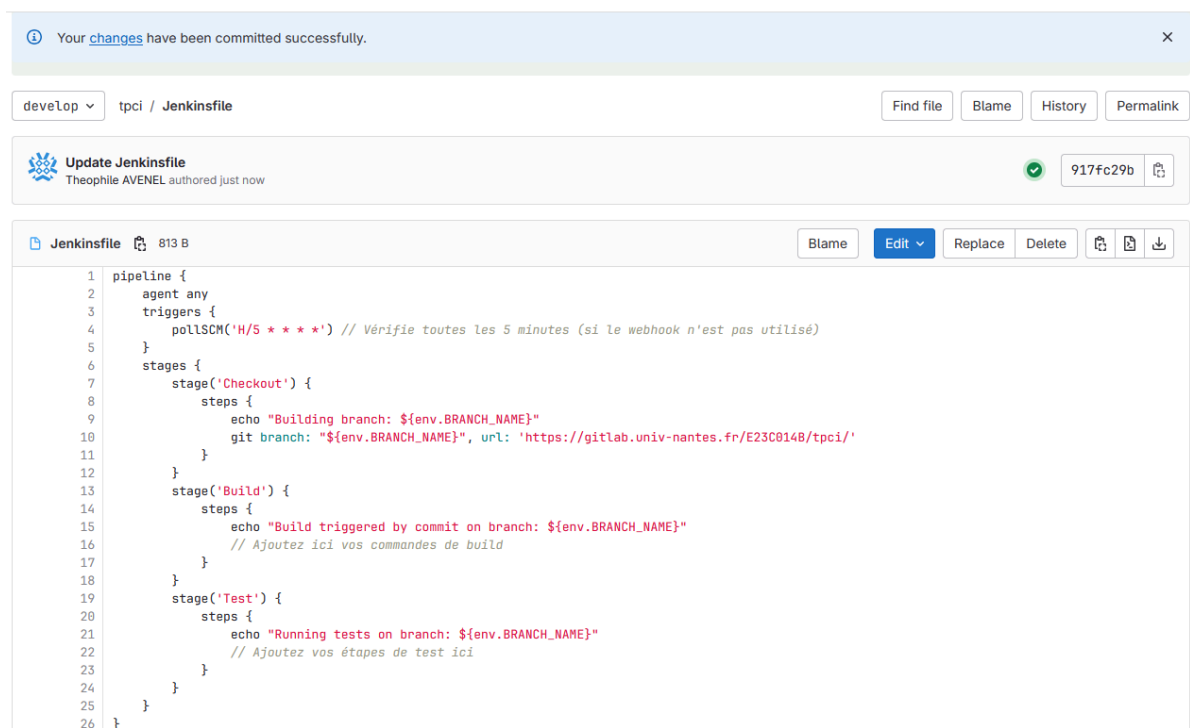


Figure 55: Création du Jenkinsfile mis dans le projet gitlab

Ce Jenkinsfile configure une pipeline CI/CD avec trois étapes principales : Checkout, Build, et Test. Il utilise un agent générique (agent any) pour s'exécuter sur n'importe quel nœud disponible. La pipeline est déclenchée automatiquement toutes les 5 minutes via pollSCM,

sauf si un webhook est configuré. Dans l'étape Checkout, le code source est extrait de la branche spécifiée (env .BRANCH_NAME) depuis un dépôt GitLab. L'étape Build inclut les commandes nécessaires pour compiler ou préparer l'application. Enfin, l'étape Test exécute les tests pour valider le code. Les messages echo fournissent des logs pour chaque étape. Voici ci-dessous le résultat de l'exécution de ce Jenkinsfile

```
Started by user test
[Tue Nov 26 19:00:52 UTC 2024] Starting branch indexing...
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git ls-remote --symref -- https://gitlab.univ-nantes.fr/E23C014B/tpci/ #
timeout=10
Creating git repository in /var/jenkins_home/caches/
git-9d3dc4aa44280a46bfc84359c3a57f0b
> git init /var/jenkins_home/caches/git-9d3dc4aa44280a46bfc84359c3a57f0b #
timeout=10
Setting origin to https://gitlab.univ-nantes.fr/E23C014B/tpci/
> git config remote.origin.url https://gitlab.univ-nantes.fr/E23C014B/tpci/
# timeout=10
Fetching & pruning origin...
Listing remote references...
> git config --get remote.origin.url # timeout=10
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git ls-remote -h -- https://gitlab.univ-nantes.fr/E23C014B/tpci/ #
timeout=10
Fetching upstream changes from origin
> git config --get remote.origin.url # timeout=10
> git fetch --tags --force --progress --prune -- origin +refs/heads/*:refs/
remotes/origin/* # timeout=10
Checking branches...
  Checking branch develop
    'Jenkinsfile' found
    Met criteria
Scheduled build for branch: develop
  Checking branch main
    'Jenkinsfile' not found
    Does not meet criteria
Processed 2 branches
[Tue Nov 26 19:01:01 UTC 2024] Finished branch indexing. Indexing took 8.9
sec
Finished: SUCCESS
```

Figure 56: résultat d'exécution du Jenkinsfile

Le jenkins a pu tester un changement au lancement dans la base de code



Figure 57: Pipeline de test Jenkins effectué à la suite d'un changement dans la base de code du repository Gitlab

Nous avons ensuite pu automatiser la compilation à l'aide de l'outil cmake et effectuer le build par Jenkins à partir de cmake.

J'ai donc commencé par modifier le code du JenkinsFile

https://gitlab.univ-nantes.fr/E23C014B/tpci/-/blob/main/Jenkinsfile?ref_type=heads

Ce Jenkinsfile automatise une pipeline CI/CD pour un projet C/C++ utilisant CMake. La pipeline s'exécute sur n'importe quel agent disponible et vérifie les changements toutes les 5 minutes grâce à pollSCM. L'étape Checkout clone la branche spécifiée depuis le dépôt Git. L'étape Build utilise CMake pour configurer le répertoire de build et génère les fichiers makefiles, puis compile le projet avec make. Les fichiers générés sont archivés si la compilation réussit. Les étapes Test C et Test C++ exécutent respectivement les fichiers binaires main_c et main_cpp en simulant une entrée utilisateur (echo "Alice"). Enfin, les blocs post enregistrent le statut global de la pipeline, affichant des messages en cas de succès ou d'échec. Ce workflow garantit une intégration continue efficace pour un projet multi-langage.

J'ai aussi pu créer un fichier CMakeLists.txt

https://gitlab.univ-nantes.fr/E23C014B/tpci/-/blob/main/CMakeLists.txt?ref_type=heads

Ce fichier CMakeLists.txt configure le projet CMake. Il définit la version minimale de CMake (3.10) et nomme le projet MultiLangProject. Deux exécutables sont configurés : main_c, qui compile les fichiers C (main.c et calcul.c), et main_cpp, qui compile les fichiers C++ (main.cpp et calcul.cpp). Ce fichier permet une gestion simple et centralisée de la compilation pour les codes C et C++. Cependant lors de l'exécution je suis tombé sur une erreur

Tableau de bord > Build-TPCI-All-Branches > develop > #3 > Pipeline Console

Build #3 (Failed) [Rebuild] [Overview] [More]

En échec 1 mn 9 s ago in 41 s

- Checkout SCM
- Checkout
- Build**
- Test C
- Test C++
- Post Actions

Stage 'Build'

- Started 52 s ago
- Queued 0 ms
- Took 11 s
- Failure
- [View as plain text](#)

Build triggered by commit on branch: develop (0.42 s)

Print Message

Préparer le répertoire de build mkdir -p build cd build # Générer les fichiers Make avec CMake cmake .. # Compl... (9.3 s)

Shell Script

```
0 + mkdir -p build
1 + cd build
2 + cmake ..
3 /home/athena/jenkins-agent-workdir/workspace/Build-TPCI-All-Branches_develop@tmp/durable-52f9ea85/script.sh.copy: 7: cmake
not found
```

Build failed. Check the logs for errors. (0.56 s)

Print Message

Jenkins 2.479.1

Figure 58: Erreur à l'étape du cmake sur le pipeline Jenkins

L'erreur cmake not found signifie que cmake doit être installé sur la VM qu'a lancé l'agent. J'ai donc utilisé la commande `sudo apt update` suivi de la commande `sudo apt install cmake -y` pour installer cmake puis j'ai vérifié l'installation de cmake avec la commande `cmake --version`.

```
athena@ubuntumate2204:~$ sudo apt update
sudo apt install cmake -y
```

Figure 59: Installation de cmake

```
athena@ubuntumate2204:~$ cmake --version
cmake version 3.22.1

CMake suite maintained and supported by Kitware (kitware.com/cmake).
athena@ubuntumate2204:~$
```

Figure 60: Vérification de l'installation de cmake

Après cette vérification, j'ai pu rebuild er le pipeline est passé dans l'état success

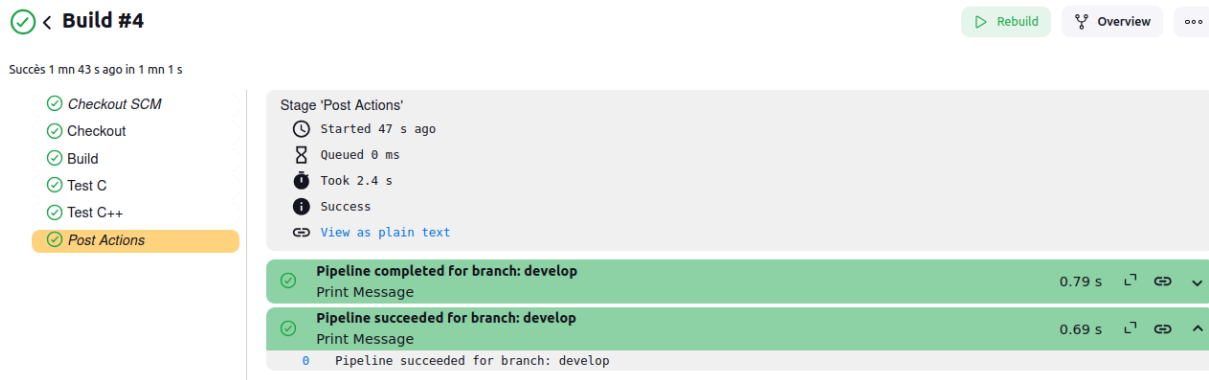


Figure 61: Le pipeline de test Jenkins s'est correctement exécuté

Pour voir une vidéo finale de l'exécution des pipelines de gitlab lors d'un commit dans develop, du pipeline CI/CD , d'une merge request de la branche develop de la branche main et les différentes exécutions de builds sur Jenkins :

<https://www.youtube.com/watch?v=gqHrv-xkqx0>