

# Comment installer Git sur Ubuntu 16.04

## Introduction

Un outil indispensable dans le développement de logiciels modernes est une sorte de système de contrôle de version. Les systèmes de contrôle de version vous permettent de garder une trace de votre logiciel au niveau de la source. Vous pouvez suivre les modifications, revenir aux étapes précédentes et créer des branches pour créer des versions alternatives de fichiers et de répertoires.

L'un des systèmes de contrôle de version les plus populaires est `git`, un système de contrôle de version distribué. De nombreux projets conservent leurs fichiers dans un référentiel `git`, et des sites comme GitHub et Bitbucket ont rendu le partage et la contribution au code simples et précieux.

Dans ce guide, nous montrerons comment installer et configurer `git` sur un système Ubuntu 16.04. Nous verrons comment installer le logiciel de deux manières différentes, chacune présentant des avantages.

## Comment installer Git avec Apt

**Remarque :** Pour réaliser ce TP il faut avoir un compte `git hub`

De loin, le moyen le plus simple d'être `git` installé et prêt à l'emploi consiste à utiliser les référentiels par défaut d'Ubuntu. C'est la méthode la plus rapide, mais la version peut être plus ancienne que la version la plus récente. Si vous avez besoin de la dernière version, envisagez de suivre les étapes pour compiler à `git` partir des sources. Vous pouvez utiliser les `apt` outils de gestion des packages pour mettre à jour votre index de package local. Ensuite, vous pouvez télécharger et installer le programme :

```
root@berenger:~# apt update
Atteint :1 http://ppa.launchpad.net/ansible/ansible-2.10/ubuntu xenial InRelease
Atteint :2 http://sn.archive.ubuntu.com/ubuntu xenial InRelease
```

```
root@berenger:~# apt-get install git
```

Cela va télécharger et installer `git` sur votre système. Vous devrez toujours suivre les étapes de configuration que nous couvrons dans la section "Configuration", alors n'hésitez pas à passer à [cette section](#) maintenant.

Pour voir la version de `git` installé on tape la commande :

```
root@berenger:~/git# git --version
git version 2.7.4
root@berenger:~/git#
```

## Comment installer Git à partir de la source

Une méthode d'installation plus flexible consiste à compiler le logiciel à partir de la source. Cela prend plus de temps et ne sera pas maintenu via votre gestionnaire de packages, mais cela vous permettra de télécharger la dernière version et vous donnera un certain contrôle sur les options que vous incluez si vous souhaitez personnaliser. Avant de commencer, vous devez installer le logiciel qui `git` en dépend. Tout cela est disponible dans les référentiels par défaut, nous pouvons donc mettre à jour notre index de packages local, puis installer les packages :

```

root@berenger:~# apt-get install build-essential libssl-dev libcurl4-gnutls-dev libexpat1-dev gettext unzip
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
build-essential est déjà la version la plus récente (12.1ubuntu2).
gettext est déjà la version la plus récente (0.19.7-2ubuntu3.1).
libexpat1-dev est déjà la version la plus récente (2.1.0-7ubuntu0.16.04.5).
libexpat1-dev passé en « installé manuellement ».
libssl-dev est déjà la version la plus récente (1.0.2g-1ubuntu4.20).
unzip est déjà la version la plus récente (6.0-20ubuntu1.1).
Paquets suggérés :
  libcurl4-doc libcurl3-dbg libgnutls-dev libidn11-dev librtmp-dev
Les NOUVEAUX paquets suivants seront installés :
  libcurl4-gnutls-dev
0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 268 ko dans les archives.
Après cette opération, 1280 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n]
Réception de :1 http://sn.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libcurl4-gnutls-dev amd64 7.47.0-1ubunt
u2.19 [268 kB]
268 ko réceptionnés en 1s (194 ko/s)
Sélection du paquet libcurl4-gnutls-dev:amd64 précédemment désélectionné.
(Lecture de la base de données... 269624 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de ../libcurl4-gnutls-dev_7.47.0-1ubuntu2.19_amd64.deb ...

```

Après avoir installé les dépendances nécessaires. Vous pouvez le saisir wget et le suivre en collant l'adresse que vous avez copiée. L'URL que vous avez copiée peut être différente de la mienne :

```

root@berenger:~# wget https://github.com/git/git/archive/v2.8.1.zip -O git.zip
--2022-05-04 06:59:17-- https://github.com/git/git/archive/v2.8.1.zip
Résolution de github.com (github.com)... 140.82.121.4
Connexion à github.com (github.com)[140.82.121.4]:443... connecté.
requête HTTP transmise, en attente de la réponse... 302 Found
Emplacement : https://codeload.github.com/git/git/zip/refs/tags/v2.8.1 [suivant]
--2022-05-04 06:59:17-- https://codeload.github.com/git/git/zip/refs/tags/v2.8.1
Résolution de codeload.github.com (codeload.github.com)... 140.82.121.9
Connexion à codeload.github.com (codeload.github.com)[140.82.121.9]:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : non indiqué [application/zip]
Enregistre : «git.zip»

git.zip [====>] 6,71M
1,71MB/s ds 4,2s

2022-05-04 06:59:22 (1,61 MB/s) - «git.zip» enregistré [7032279]

```

Décompressez le fichier que vous avez téléchargé et déplacez-vous dans le répertoire résultant en tapant :

```

root@berenger:~# unzip git.zip
Archive: git.zip
d95553a6b8c5153f541adfc3346004e8249b0e6
  creating: git-2.8.1/
  inflating: git-2.8.1/.gitattributes
  inflating: git-2.8.1/.gitignore
  inflating: git-2.8.1/.mailmap
  inflating: git-2.8.1/.travis.yml
  inflating: git-2.8.1/COPYING
  creating: git-2.8.1/Documentation/
 extracting: git-2.8.1/Documentation/.gitattributes
  inflating: git-2.8.1/Documentation/.gitignore
  inflating: git-2.8.1/Documentation/CodingGuidelines
  inflating: git-2.8.1/Documentation/Makefile
  creating: git-2.8.1/Documentation/RelNotes/
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.1.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.2.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.3.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.4.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.5.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.6.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.7.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.0.txt
  inflating: git-2.8.1/Documentation/RelNotes/1.5.1.1.txt

```

Puis on se déplace dans le dossier

```

root@berenger:~# cd git-*

```

Maintenant, vous pouvez créer le package et l'installer en tapant ces deux commandes :

```

root@berenger:~/git-2.8.1# make prefix=/usr/local all
GIT_VERSION = 2.8.1
* new build flags
CC credential-store.o
* new link flags
CC abspath.o
CC advice.o
CC alias.o
CC alloc.o
CC archive.o
CC archive-tar.o
CC archive-zip.o
CC argv-array.o
* new prefix flags
CC attr.o
CC base85.o
CC bisect.o
CC blob.o
CC branch.o
CC bulk-checkin.o
CC bundle.o
CC cache-tree.o
CC color.o
CC column.o

```

```

root@berenger:~/git-2.8.1# sudo make prefix=/usr/local install
SUBDIR perl
/usr/bin/perl Makefile.PL PREFIX='/usr/local' INSTALL_BASE='' --localedir='/usr/local/share/locale'
Generating a Unix-style perl.mak
Writing perl.mak for Git
Writing MYMETA.vml and MYMETA.json
GEN git-add--interactive
GEN git-difftool
GEN git-archimport
GEN git-cvsexportcommit
GEN git-cvimport
GEN git-cvserver
GEN git-relink
GEN git-send-email
GEN git-svn
SUBDIR git-gui
SUBDIR gitk-git
SUBDIR perl
"/usr/bin/perl" -pe "s<Q++LOCALEDIR++\E></usr/local/share/locale>" <Git/SVN/Utils.pm >blib/lib/Git/SVN/Utils.pm
"/usr/bin/perl" -pe "s<Q++LOCALEDIR++\E></usr/local/share/locale>" <Git/IndexInfo.pm >blib/lib/Git/IndexInfo.pm
"/usr/bin/perl" -pe "s<Q++LOCALEDIR++\E></usr/local/share/locale>" <Git/SVN/Memoize/YAML.pm >blib/lib/Git/SVN/Memoize/YAML.pm

```

Maintenant que vous avez `git` installé, si vous souhaitez mettre à niveau vers une version ultérieure, vous pouvez simplement cloner le référentiel (en veillant à changer d'abord votre répertoire personnel), puis compiler et installer. Pour trouver l'URL à utiliser pour l'opération de clonage, accédez à la branche ou à la balise souhaitée sur la [page GitHub du projet](#), puis copiez l'URL de clonage sur le côté droit :

Accédez à votre répertoire personnel et utilisez `git clone` sur l'URL que vous venez de copier :

```

root@berenger:~/git-2.8.1# cd ~
root@berenger:~# git clone https://github.com/git/git.git
Clonage dans 'git'...
remote: Enumerating objects: 327671, done.
remote: Total 327671 (delta 0), reused 0 (delta 0), pack-reused 327671
Réception d'objets: 100% (327671/327671), 189.62 MiB | 1.37 MiB/s, fait.
Résolution des deltas: 100% (244869/244869), fait.
Vérification de la connectivité... fait.

```

Cela créera un nouveau répertoire dans votre répertoire actuel où vous pourrez reconstruire le package et réinstaller la version la plus récente, comme vous l'avez fait ci-dessus. Cela remplacera votre ancienne version par la nouvelle version :

```

root@berenger:~# cd git

```

```

root@berenger:~/git# git clone https://github.com/git/git.git
Clonage dans 'git'...
remote: Enumerating objects: 327671, done.
^Cception d'objets: 7% (22937/327671), 9.05 MiB | 1.56 MiB/s
root@berenger:~/git# make prefix=/usr/local all
GIT_VERSION = 2.36.0.44.g0f82833
* new build flags
CC fuzz-commit-graph.o
CC fuzz-pack-headers.o
CC fuzz-pack-idx.o
CC daemon.o
* new link flags
CC common-main.o
CC abspath.o
CC add-interactive.o
CC add-patch.o
CC advice.o
CC alias.o
CC alloc.o
CC apply.o
CC archive-tar.o

```

## Comment configurer Git

Maintenant que vous avez `git` installé, vous devez faire quelques choses pour que les messages de validation qui seront générés pour vous contiennent vos informations correctes. La façon la plus simple de le faire est d'utiliser la `git config` commande. Plus précisément, nous devons fournir notre nom et notre adresse e-mail, car `git` ces informations sont intégrées dans chaque engagement que nous effectuons. Nous pouvons aller de l'avant et ajouter ces informations en tapant:

```

root@berenger:~/git# git config --global user.name "Berenger BENAM"
root@berenger:~/git# git config --global user.email "berengerbenam@gmail.com"

```

Nous pouvons voir tous les éléments de configuration qui ont été définis en tapant :

```

root@berenger:~/git# git config --global user.name "Berenger BENAM"
root@berenger:~/git# git config --global user.email "berengerbenam@gmail.com"
root@berenger:~/git# git config --list
user.name=Berenger BENAM
user.email=berengerbenam@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/git/git.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master

```

Comme vous pouvez le voir, cela a un format légèrement différent. Les informations sont stockées dans votre `git` fichier de configuration, que vous pouvez éventuellement modifier à la main avec votre éditeur de texte comme ceci :

nano `~/gitconfig`

```

GNU nano 2.5.3          Fichier : /home/berenger/.gitconfig
[user]
  name = Berenger BENAM
  email = berengerbenam@gmail.com

```

Donc on c'est possible de changer même l'adresse pour la connexion Git Hub.

On crée le dossier **Projet\_git**

```
root@berenger:~# mkdir Projet_git
root@berenger:~# █
```

On se déplace dans le dossier : **Projet\_git**

```
root@berenger:~# cd Projet_git/
root@berenger:~/Projet_git# pwd
/home/berenger/Projet_git
root@berenger:~/Projet_git#
```

- Comment Configurer l'outil github

```
root@berenger:~/Projet_git# git config --global user.name "Berenger Benam"
root@berenger:~/Projet_git# █
```

On veut configurer l'outil git de manière globale en précisant le nom de l'utilisateur.

```
root@berenger:~/Projet_git# git config --global user.mail berengerbenam@gmail.com
root@berenger:~/Projet_git#
```

Sur cette capture on précise aussi l'email de l'utilisateur.

-pour lister la configuration globale on tape cette commande :

```
root@berenger:~/Projet_git# git config --global --list
user.name=Berenger Benam
user.email=berengerbenam@gmail.com
user.mail=berengerbenam@gmail.com
root@berenger:~/Projet_git# █
```

Oui la configuration marche bien et on peut voir les informations sur le terminal.

-si on veut avoir de l'aide d'utilisation de git on tape la commande : **git** ou bien **gitt --help**

```

root@berenger:~/Projet_git# git
usage: git [--version] [--help] [-C <path>] [-c name=value]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          <command> [<args>]

Ci-dessous les commandes Git habituelles dans diverses situations :

démarrer une zone de travail (voir aussi : git help tutorial)
  clone      Cloner un dépôt dans un nouveau répertoire
  init       Créer un dépôt Git vide ou réinitialiser un existant

travailler sur la modification actuelle (voir aussi : git help revisions)
  add        Ajouter le contenu de fichiers dans l'index
  mv         Déplacer ou renommer un fichier, un répertoire, ou un lien symbolique
  reset      Réinitialiser la HEAD courante à l'état spécifié
  rm         Supprimer des fichiers de la copie de travail et de l'index

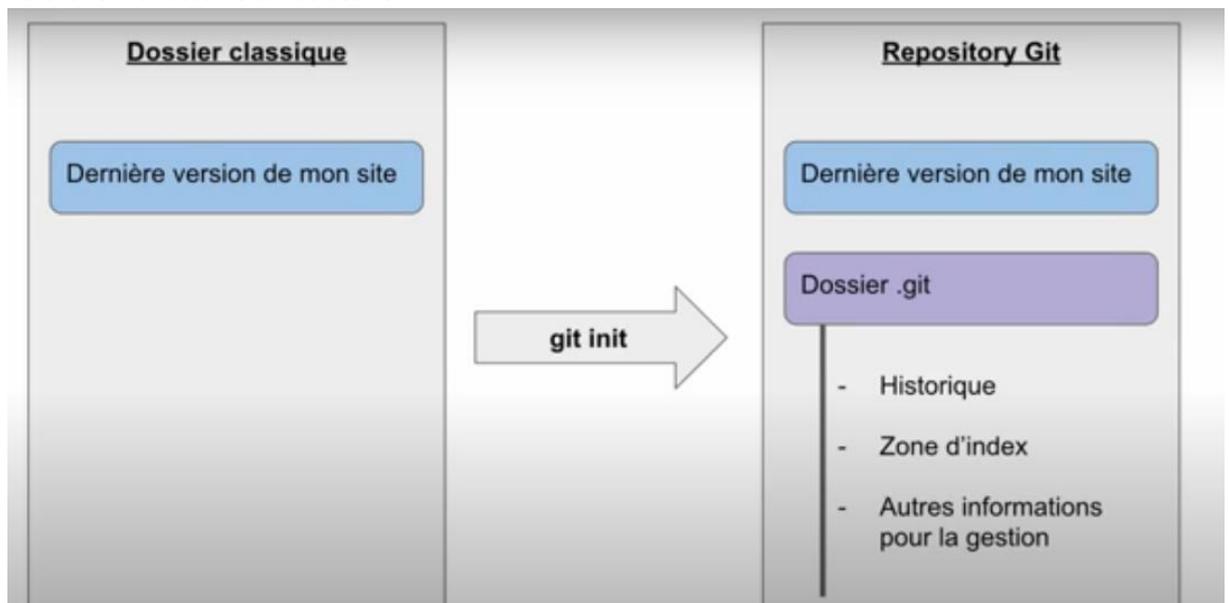
examiner l'historique et l'état (voir aussi : git help revisions)
  bisect     Trouver par recherche binaire la modification qui a introduit un bogue
  grep       Afficher les lignes correspondant à un motif
  log        Afficher l'historique des validations

```

- Dans cette partie on va voir comment réaliser un dépôt git et des différentes actions via la commande **commit**.

**Qu'est-ce qu'un dépôt git** : c'est simplement c'est utiliser dossier dans lequel vous avez déjà votre projet puis de le transformer via une commande **git init**(va ajouter un dossier caché nommé **.git** et ce dossier contient tous les informations git)

Voici une architecture sur ce dossier



Dossier **.git** contient les informations tels que : Historique, zone d'index et les autres informations pour la gestion de git.

On va recréer un nouveau dossier pour contenir notre dépôt git : **mon\_dossier\_web**

```

root@berenger:~/Projet_git# mkdir mon_dossier_web
root@berenger:~/Projet_git# █

```

Puis on se déplace dans ce dossier :

```
root@berenger:~/Projet_git# cd mon_dossier_web/
root@berenger:~/Projet_git/mon_dossier_web#
```

On va le transformer a dossier git par la commande :

La commande **git init** veut dire initialiser un dépôt dans le dossier courant

```
root@berenger:~/Projet_git/mon_dossier_web# git init
```

Si on fait juste **ls** pour lister le contenu du dossier

```
root@berenger:~/Projet_git/mon_dossier_web# ls
root@berenger:~/Projet_git/mon_dossier_web# ls -la
total 12
drwxr-xr-x 3 root root 4096 me  4 09:13 .
drwxr-xr-x 3 root root 4096 me  4 09:10 ..
drwxr-xr-x 7 root root 4096 me  4 09:17 .git
root@berenger:~/Projet_git/mon_dossier_web#
```

Il crée un dossier cachet git et pour le voir il faut utiliser la commande : **ls -la** pour afficher tous les dossiers cachets.

Et voit bien le dossier **.git**

## Nous avons créé notre 1<sup>er</sup> dépôt git

On va ajouter les différentes fichiers dans le dépôt git

```
root@berenger:~/Projet_git/mon_dossier_web# touch hello.html
```

On crée le fichier hello.html

Voici son contenu juste un message de bienvenue

```
root@berenger:~/Projet_git/mon_dossier_web# cat hello.html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Formation GIT</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <!-- TITRE -->
    <div>
      Bienvenue dans la page GIT !
    </div>
  </body>
</html>
root@berenger:~/Projet_git/mon_dossier_web#
```

Idem pour css

```
root@berenger:~/Projet_git/mon_dossier_web# touch style.css
```

```
root@berenger:~/Projet_git/mon_dossier_web# cat style.css
#top-banner {
  color : rgb(255,255,255);
  font-size: 30px;

  background-color: rgba(30,30,30,0.7);
  overflow: auto;
  text-align: center;
}

#top-banner-img{
  float:right;
}

#menu {
  width: 260px;
  height: 100%;
  border-right:1px solid #ccc;
  font-family: "Lato";
  font-weight: normal;
```

```
  font-family: "Lato";
  font-weight: normal;
  font-size: 18px;
  min-height: 500px;
  background-color: rgba(30,30,30,0.7);
  box-shadow: 3px 0px 5px rgba(0,0,0,.3);
  position:absolute;
  overflow:auto;
}

#menu .menu-button {
  padding:0.01em 16px 8px;
  line-height: 45px;
  z-index: 5;
}

#menu .menu-button > a {
  background-color:rgba(97,97,97,1);
  color:#fff;
  box-shadow: 1px 1px 4px rgba(0,0,0,.2);
  cursor: pointer;
  padding:4px 2px 4px 16px;
  display:block;
```

```
}

#top-banner, #menu {
  border-color:#616161;
}

#toggle-menu-button:hover {
  color:#757575;
}

#toggle-menu-button-label {
  font-family: "Lato";
  font-size: 14px;
}

#content {
  margin-left: 300px;
  margin-top: 40px;
}
```

```
root@berenger:~/Projet_git/mon_dossier_web# █
```

## TEST :



On voit la page Bienvenue nickel !

La commande : **git status** va nous donner l'état actuel de notre environnement de travail.

```
root@berenger:~/Projet_git/mon_dossier_web# git status Sur la branche master
Validation initiale
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

    hello.html
    style.css

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
root@berenger:~/Projet_git/mon_dossier_web#
```

On va indexer les deux fichiers par la commandes **git add**

```
root@berenger:~/Projet_git/mon_dossier_web# git add hello.html style.css
root@berenger:~/Projet_git/mon_dossier_web#
```

Pour vérifier que la commande a été bien appliquer on refaire la commande **git status**

```
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Validation initiale
Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)

    nouveau fichier : hello.html
    nouveau fichier : style.css

root@berenger:~/Projet_git/mon_dossier_web#
```

On remarque que les deux fichiers sont prêts en enregistrés dans le dépôt.

Si on veut désindexer nos deux fichiers dans le dépôt il suffit de taper la commande : **git reset nom\_ficheir**

### Exemple :

```
root@berenger:~/Projet_git/mon_dossier_web# git reset hello.html style.css
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Validation initiale
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

    hello.html
    style.css

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
root@berenger:~/Projet_git/mon_dossier_web#
```

On remarque que les deux fichiers sont désindexés dans le dépôt.

On remet notre configuration pour continuer le TP.

```
root@berenger:~/Projet_git/mon_dossier_web# git add hello.html style.css
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master

Validation initiale

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)

    nouveau fichier : hello.html
    nouveau fichier : style.css

root@berenger:~/Projet_git/mon_dossier_web#
```

-comment réaliser le 1<sup>er</sup> commit

```
root@berenger:~/Projet_git/mon_dossier_web# git commit -m"mon premier commit"
[master (commit racine) ab28a78] mon premier commit
2 files changed, 102 insertions(+)
 create mode 100644 hello.html
 create mode 100644 style.css
root@berenger:~/Projet_git/mon_dossier_web#
```

On réalise le 1<sup>er</sup> commit et l'option `-m` permet d'attacher nos deux fichiers de base et le commit a comme message mon premier commit

```
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
rien à valider, la copie de travail est propre
root@berenger:~/Projet_git/mon_dossier_web#
```

Si on fait un status le git nous a dit l'espace de travail est vide.

On ajoute une bannière dans le code html

```
root@berenger:~/Projet_git/mon_dossier_web# cat hello.html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Formation GIT</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <!-- TITRE -->
    <div id="top-banner">
      Bienvenue dans la page GIT !
    </div>
  </body>
</html>
root@berenger:~/Projet_git/mon_dossier_web#
```

Voici on ajoute un id j'ai expliqué déjà dans le fichier `style.css`

TEST :



C'est un peu plus joli.

Si on fait git status

```
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Modifications qui ne seront pas validées :
 (utilisez "git add <fichier>.." pour mettre à jour ce qui sera validé)
 (utilisez "git checkout -- <fichier>.." pour annuler les modifications dans la copie de travail)

    modifié :      hello.html

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
root@berenger:~/Projet_git/mon_dossier_web#
```

On voit que le fichier **hello.html** a été modifié.

- **Comment visualiser les modifications qui ont été apportés à ce fichier pour cela on va utiliser la commande : git diff**

```
root@berenger:~/Projet_git/mon_dossier_web# git diff
diff --git a/hello.html b/hello.html
index e87bcd4..84efc00 100644
--- a/hello.html
+++ b/hello.html
@@ -13,7 +13,7 @@
 <body>

     <!-- TITRE -->
-   <div>
+   <div id="top-banner">
       Bienvenue dans la page GIT !
-   </div>
+   </div>

root@berenger:~/Projet_git/mon_dossier_web#
```

On voit la modification qu'on a fait dans le fichier hello.html et précisément la ligne **div**.

Si on refait git add

```
root@berenger:~/Projet_git/mon_dossier_web# git add hello.html
root@berenger:~/Projet_git/mon_dossier_web# git diff
root@berenger:~/Projet_git/mon_dossier_web#
```

On constate que avec **git add hello.html** il applique la confirmation de l'ajout et la 2eme commande **git diff** on voit aucune modification grâce à l'ajout de **git add** il ne détecte pas les modifications qui ont été ajouté dans la zone indexage.

```

root@berenger:~/Projet_git/mon_dossier_web# git diff --cached
diff --git a/hello.html b/hello.html
index e87bcd4..84efc00 100644
--- a/hello.html
+++ b/hello.html
@@ -13,7 +13,7 @@
     <body>

         <!-- TITRE -->
-        <div>
+        <div id="top-banner">
             Bienvenue dans la page GIT !
         </div>

root@berenger:~/Projet_git/mon_dossier_web# █

```

Si on fait cette commande on peut voir tous les informations et la ligne de modification.

```

root@berenger:~/Projet_git/mon_dossier_web# git commit -m"ajout du top banner"
[master 86ba669] ajout du top banner
1 file changed, 1 insertion(+), 1 deletion(-)
root@berenger:~/Projet_git/mon_dossier_web#

```

On remarque git a ajouté les modifications dans notre projet.

-on va éditer le fichier **hello.html** et apporter une modification

```

root@berenger:~/Projet_git/mon_dossier_web# cat hello.html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Formation GIT</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <link rel="stylesheet" href="style.css">
  </head>

  <body>

    <!-- TITRE -->
    <div id="top-banner">
      Bienvenue dans la page GIT !
    </div>
    <div>
      <p>Bonjour. je m'appelle Berenger c'est ma 1ere fois d'utiliser git et je kiffe trop</p>
    </div>
  </body>
</html>
root@berenger:~/Projet_git/mon_dossier_web#

```

J'ai ajouté une div et avec la balise p pour faire un petit paragraphe qui contient le message je m'appelle Berenger...

Si on fait un **git status**

```

root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Modifications qui ne seront pas validées :
 (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
 (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

   modifié :   hello.html

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
root@berenger:~/Projet_git/mon_dossier_web#

```

git a détecté une modification dans le fichier **hello.html**.

Si on fait `git diff`

```
root@berenger:~/Projet_git/mon_dossier_web# git diff
diff --git a/hello.html b/hello.html
index 84efc00..400b20f 100644
--- a/hello.html
+++ b/hello.html
@@ -16,6 +16,8 @@
     <div id="top-banner">
         Bienvenue dans la page GIT !
     </div>
-
+     <div>
+         <p>Bonjour. je m'appelle Berenger c'est ma 1ere fois d'utiliser git et je kiffe trop</p>
+     </div>
 </body>
</html>
root@berenger:~/Projet_git/mon_dossier_web#
```

Il me dit j'ai ajouté mes deux balises div et p et il montre le contenu du message.

On va indexer la modification html par la commande `git add nom_fichier_indexer`

```
root@berenger:~/Projet_git/mon_dossier_web# git add hello.html
root@berenger:~/Projet_git/mon_dossier_web#
```

Toute marche bien

```
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Modifications qui seront validées :
 (utilisez "git reset HEAD <fichier>..." pour désindexer)

    modifié :      hello.html

root@berenger:~/Projet_git/mon_dossier_web#
```

On voit que mon fichier hello.html est bien indexer.

```
root@berenger:~/Projet_git/mon_dossier_web# git commit -m"description"
[master 2edac08] description
1 file changed, 3 insertions(+), 1 deletion(-)
root@berenger:~/Projet_git/mon_dossier_web#
```

On a bien enregistré le fichier indexé.

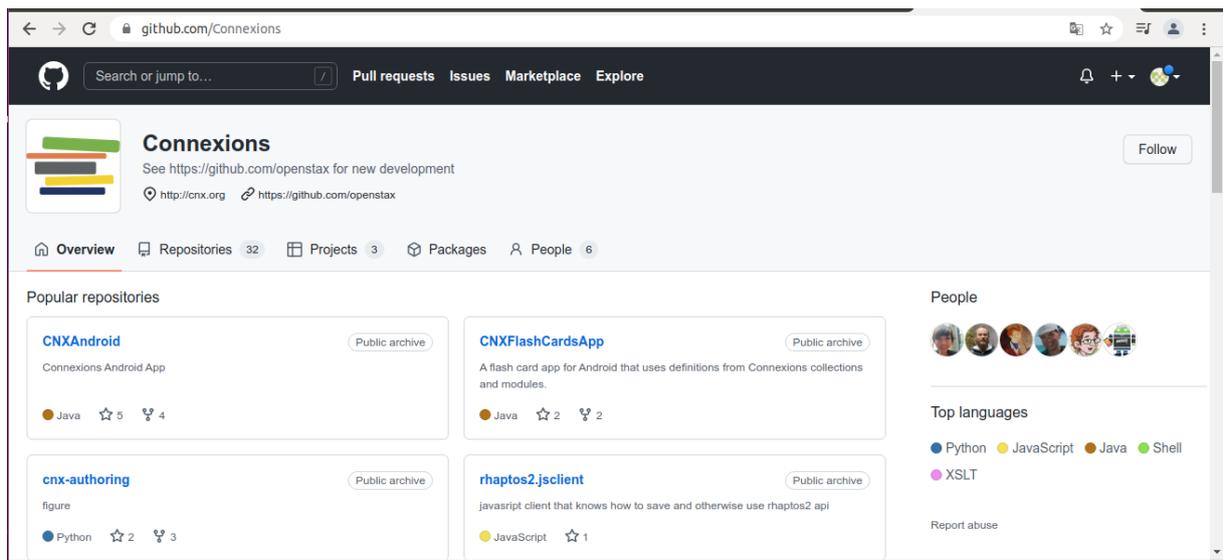
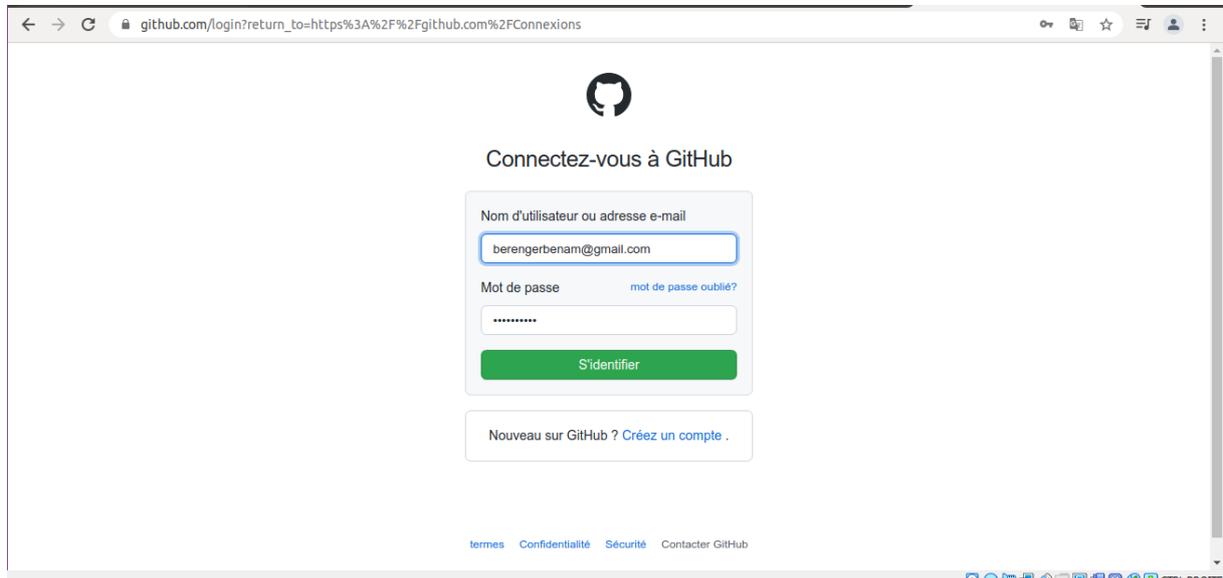
**TEST : pour voir la modification**



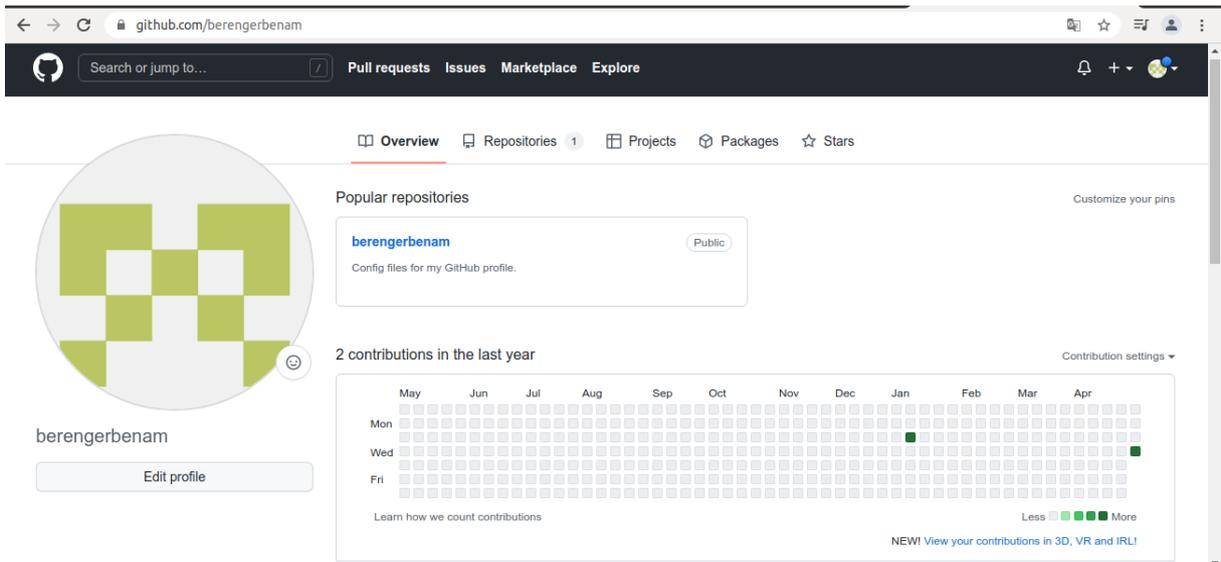
On voit bien notre page a été modifiée.

- Dans cette partie on va voir comment envoyer son code via git hub

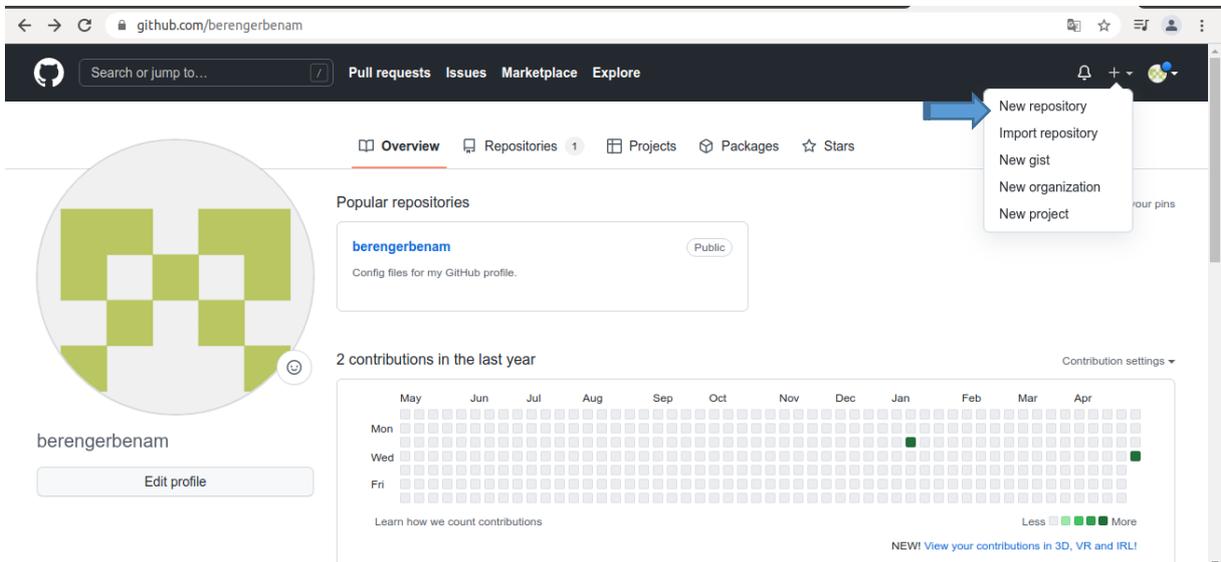
Pour réaliser on se connecte a son compte github.



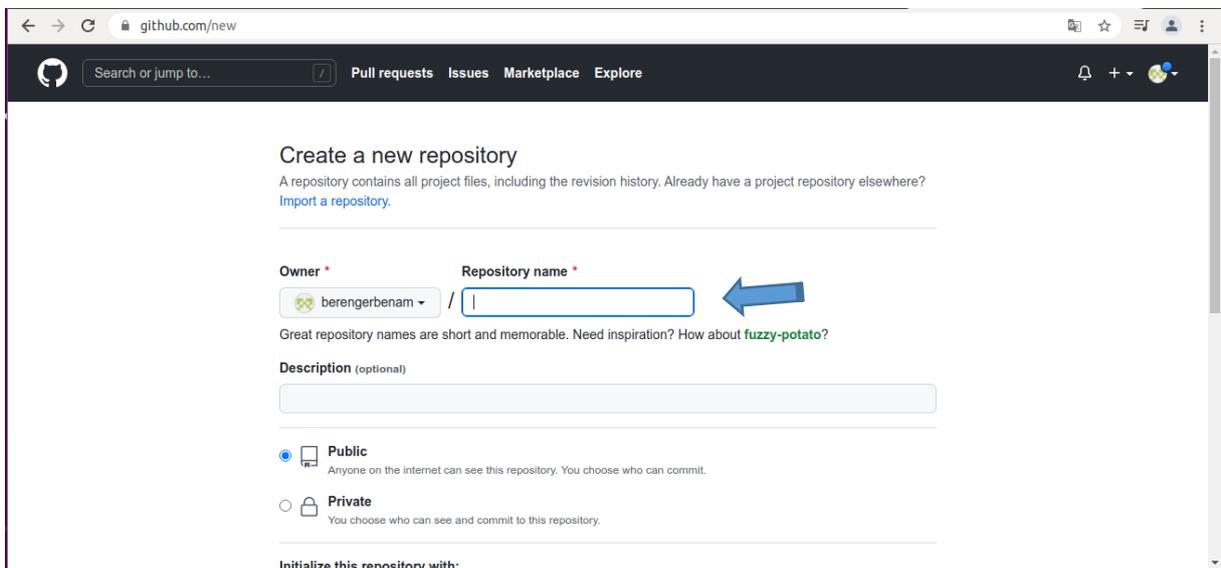
Si on clique sur profil



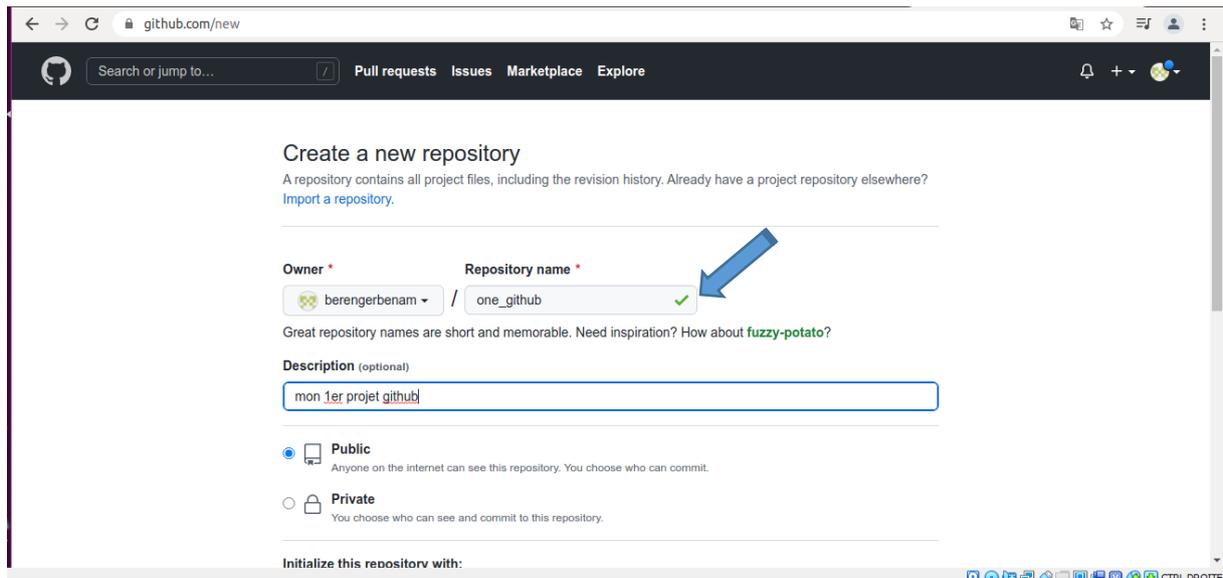
Pour le moment je n'ai pas ajouté des projets dans mon github.



On clique sur + new repository



On donne un nom à notre projet.



github.com/new

Search or jump to... Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*  / Repository name \*

Great repository names are short and memorable. Need inspiration? How about [fuzzy-potato?](#)

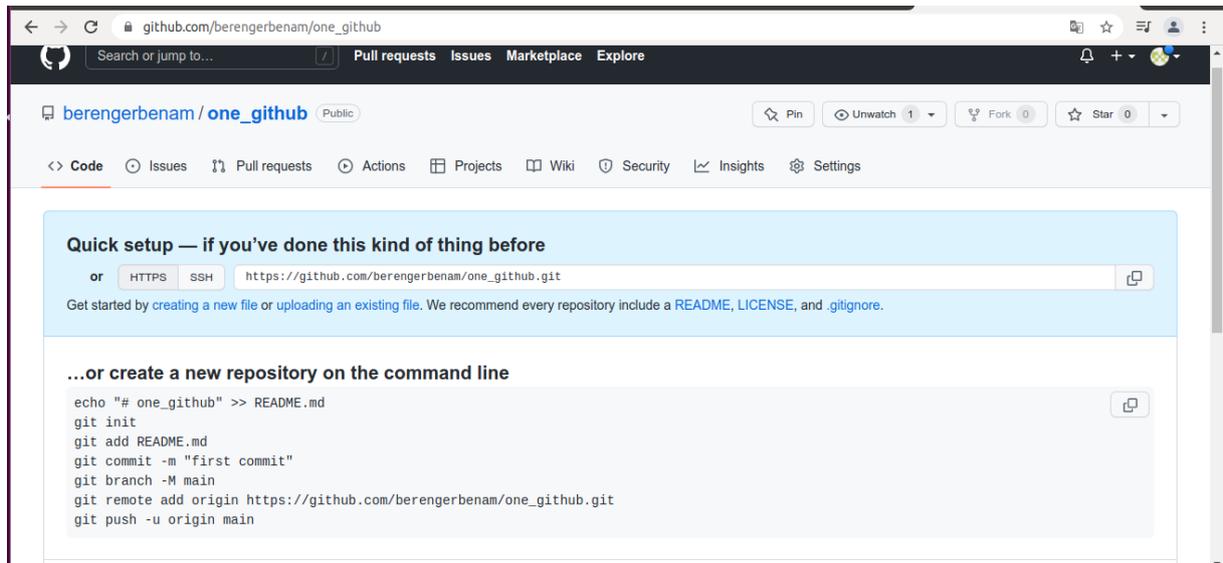
Description (optional)

**Public**  
Anyone on the internet can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

On lui donne le nom **one\_github** puis cliquez sur **create repository**



github.com/berengerbenam/one\_github

berengerbenam / one\_github Public

Pin Unwatch 1 Fork 0 Star 0

<< Code Issues Pull requests Actions Projects Wiki Security Insights Settings

#### Quick setup — if you've done this kind of thing before

or

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

#### ...or create a new repository on the command line

```
echo "# one_github" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/berengerbenam/one_github.git
git push -u origin main
```

On envoie tous les contenus du dossier **mon\_dossier\_web**

```
root@berenger:~/Projet_git/mon_dossier_web# git add .
root@berenger:~/Projet_git/mon_dossier_web#
```

On fait commit

```
root@berenger:~/Projet_git/mon_dossier_web# git commit -m 'init'
```

On copie cette ligne

or HTTPS SSH [https://github.com/berengerbenam/one\\_github.git](https://github.com/berengerbenam/one_github.git)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# one_github" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/berengerbenam/one_github.git
git push -u origin main
```

```
root@berenger:~/Projet_git/mon_dossier_web# git remote add origin https://github.com/berengerbenam/one_github.git
root@berenger:~/Projet_git/mon_dossier_web#
```

On colle sur le terminal.

### ...or create a new repository on the command line

```
echo "# one_github" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/berengerbenam/one_github.git
git push -u origin main
```

On copie cette ligne et colle dans le terminal mais une chose il faut changer **main** par **master** qui signifie **master** c'est la branche par défaut.

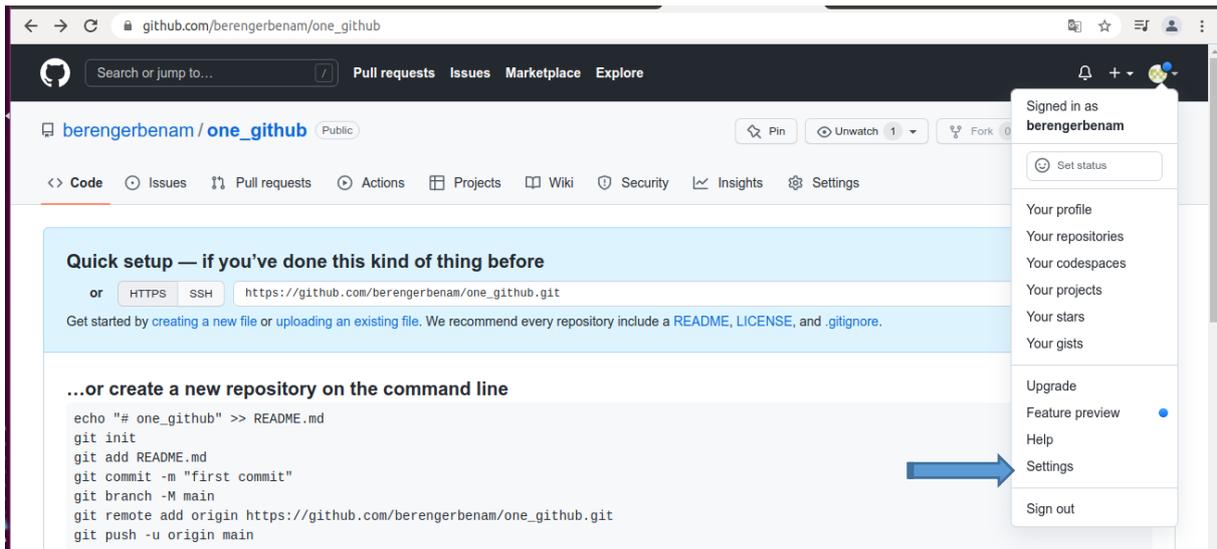
Si c'est la 1ere fois il demande login de l'utilisateur

```
root@berenger:~/Projet_git/mon_dossier_web# git push -u origin master
Username for 'https://github.com':
```

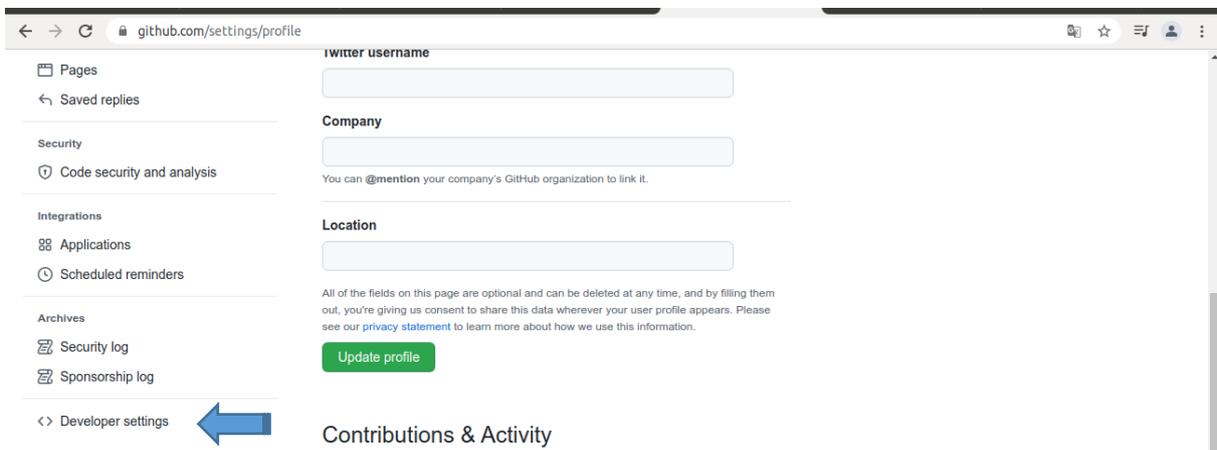
On renseigne le nom de l'utilisateur

```
root@berenger:~/Projet_git/mon_dossier_web# git push -u origin master
Username for 'https://github.com': Berenger Benam
Password for 'https://Berenger Benam@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/berengerbenam/one_github.git/'
root@berenger:~/Projet_git/mon_dossier_web#
```

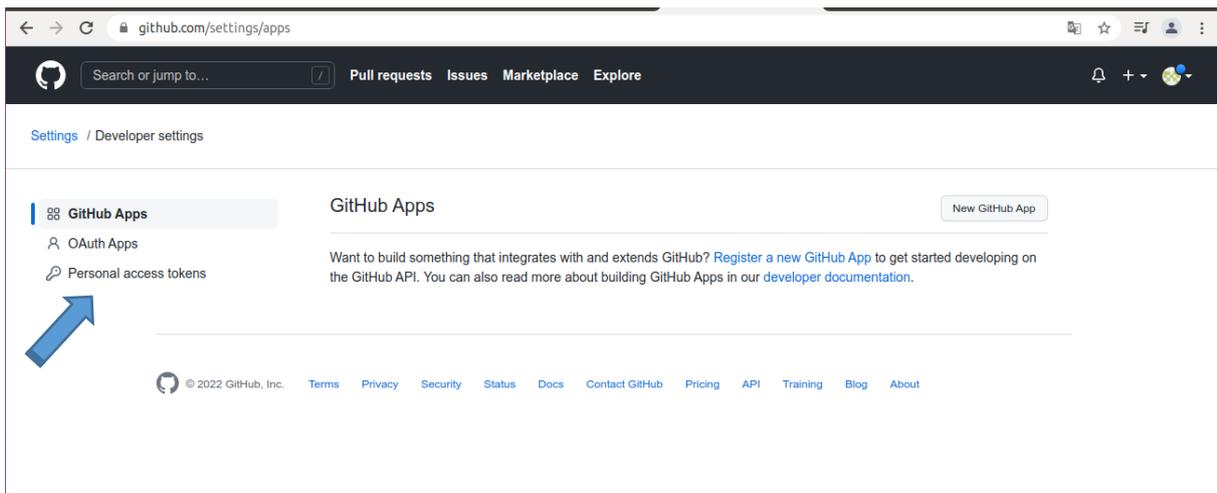
Dans mon cas j'ai rencontré des erreurs connexions c'est le problème jeton. Pour corriger cette erreur on va dans le paramètre.



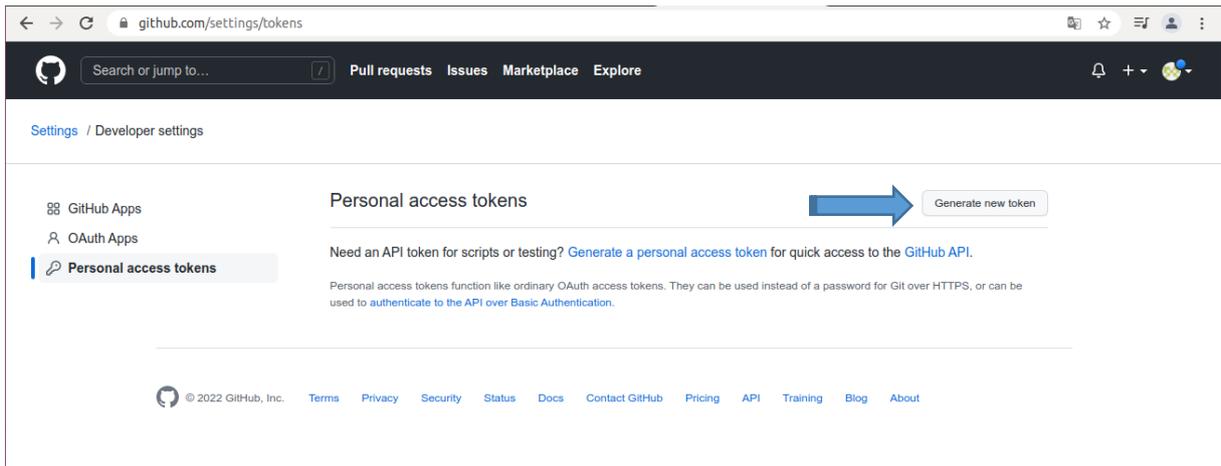
Cliquez sur setting



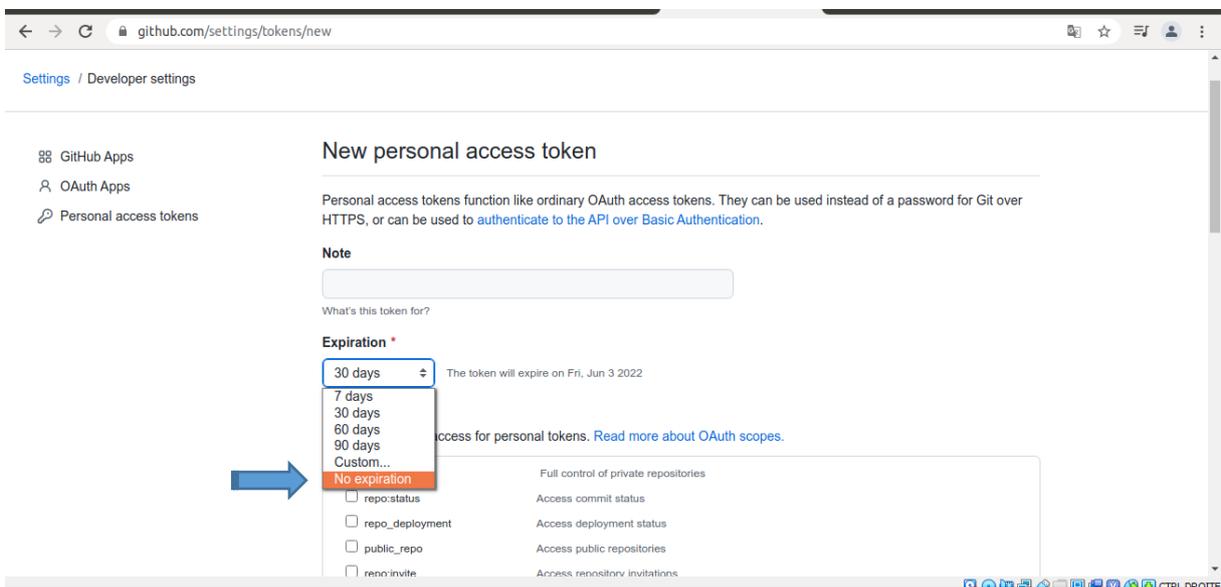
Cliquez sur Developer setting



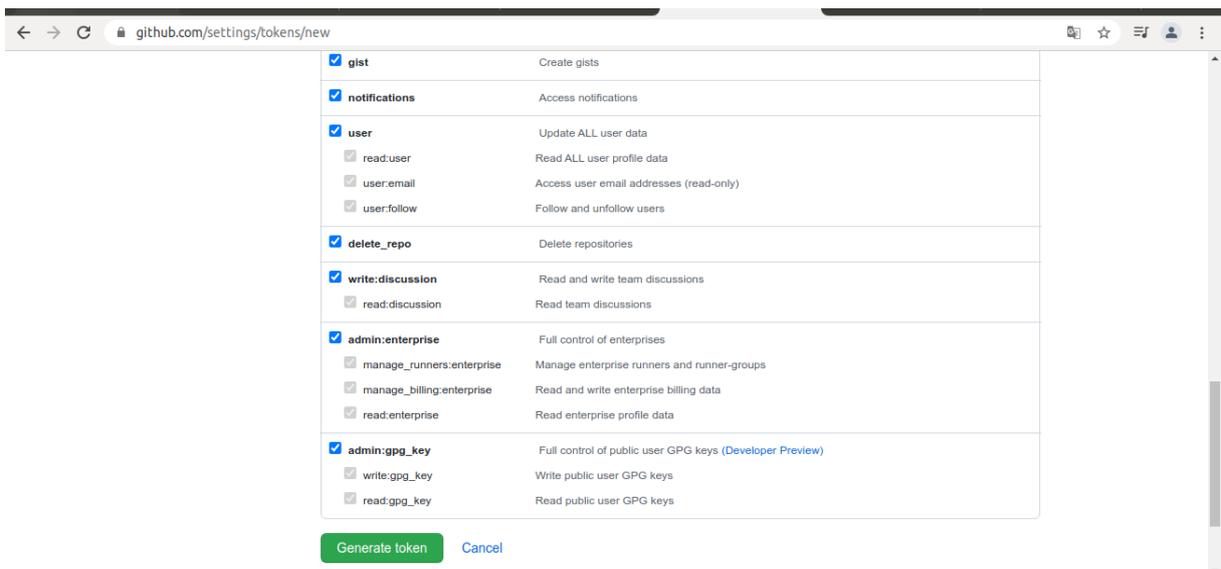
Cliquez sur Personal access tokens



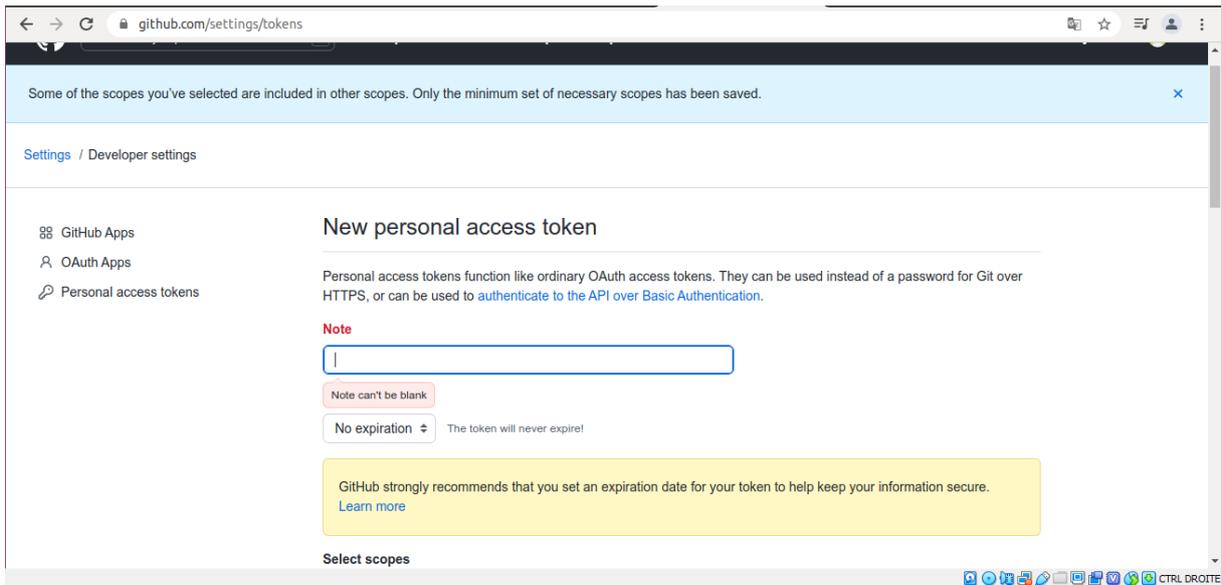
Cliquez sur Generate new token



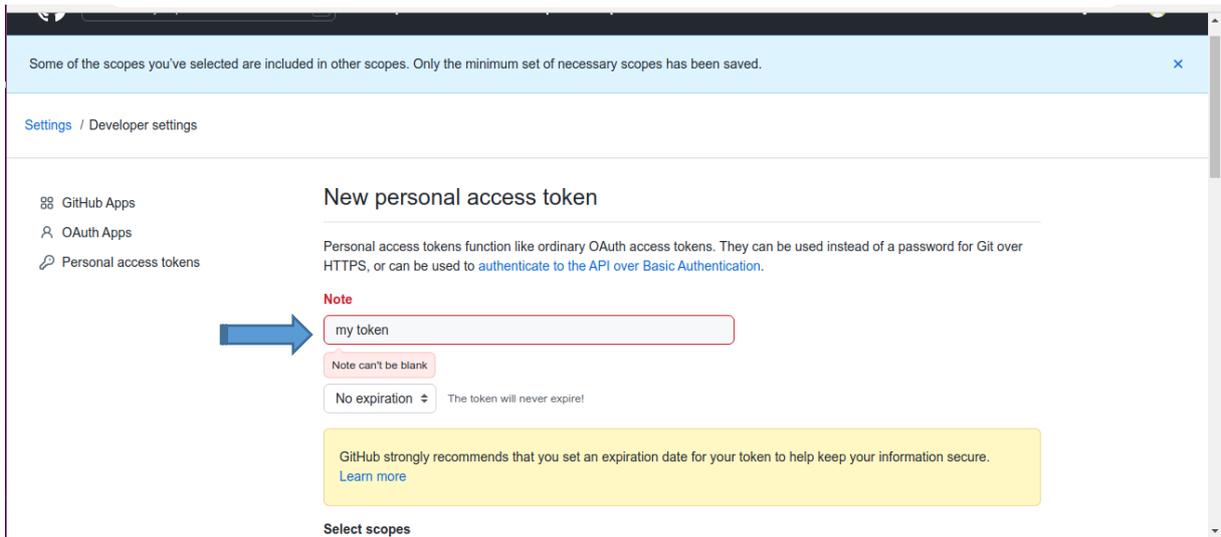
Cliquez sur No expiration



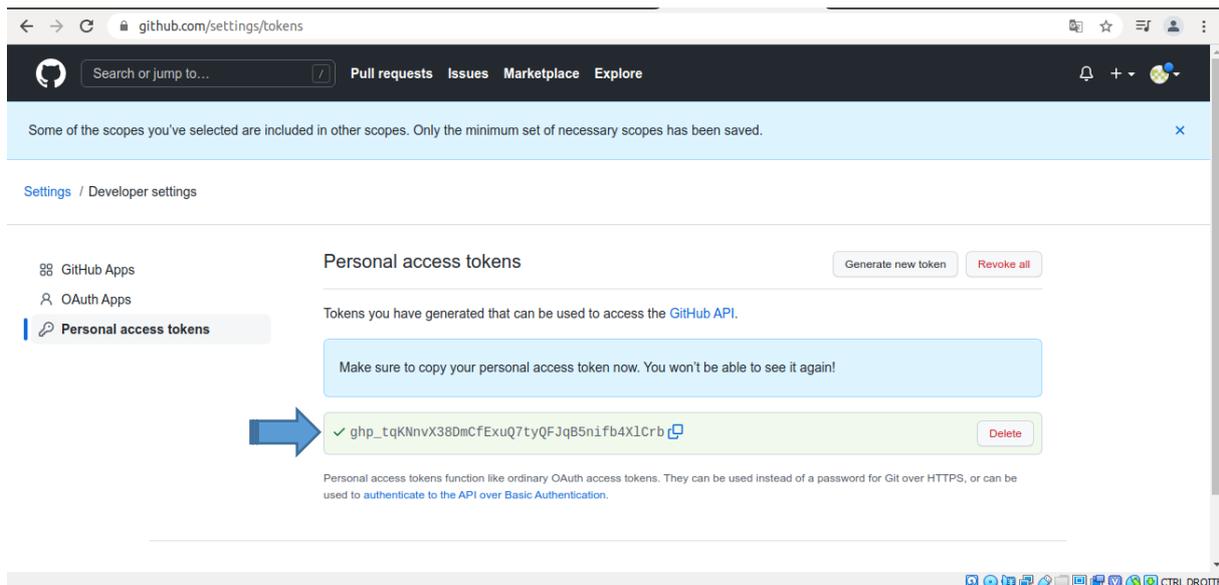
Cochez tous les cases ensuite cliquez sur gerate token



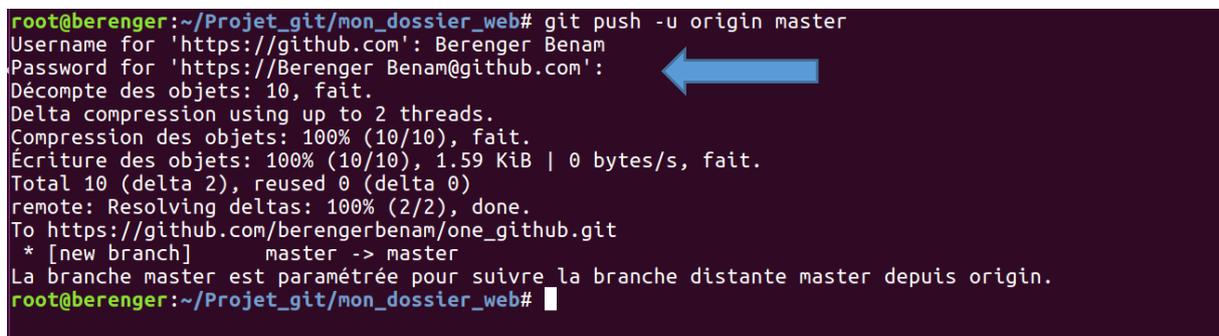
Donnez juste un nom



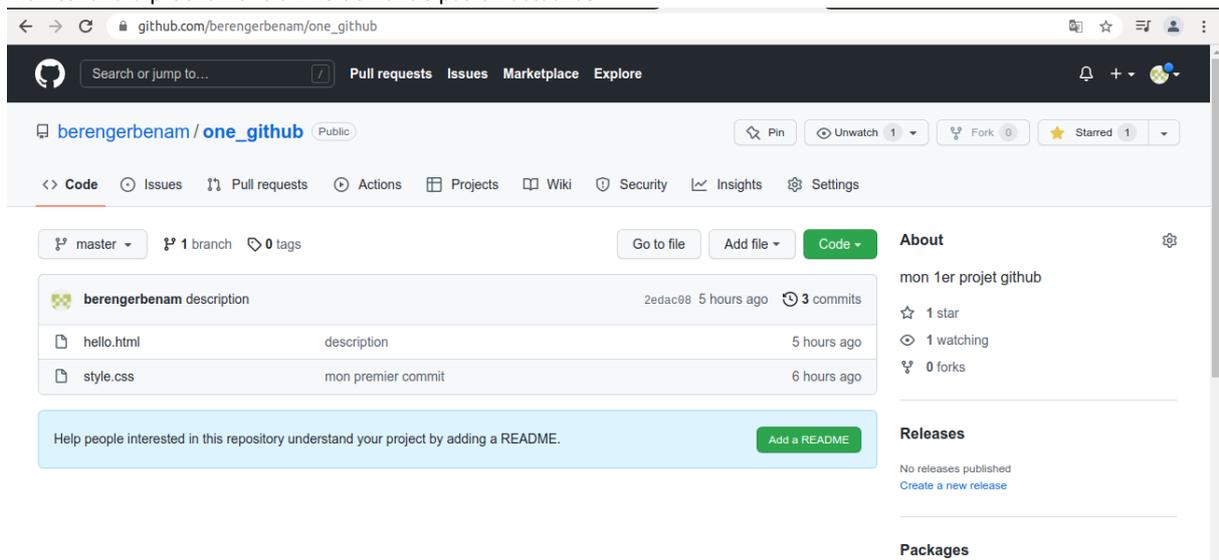
Cliquez sur generate token



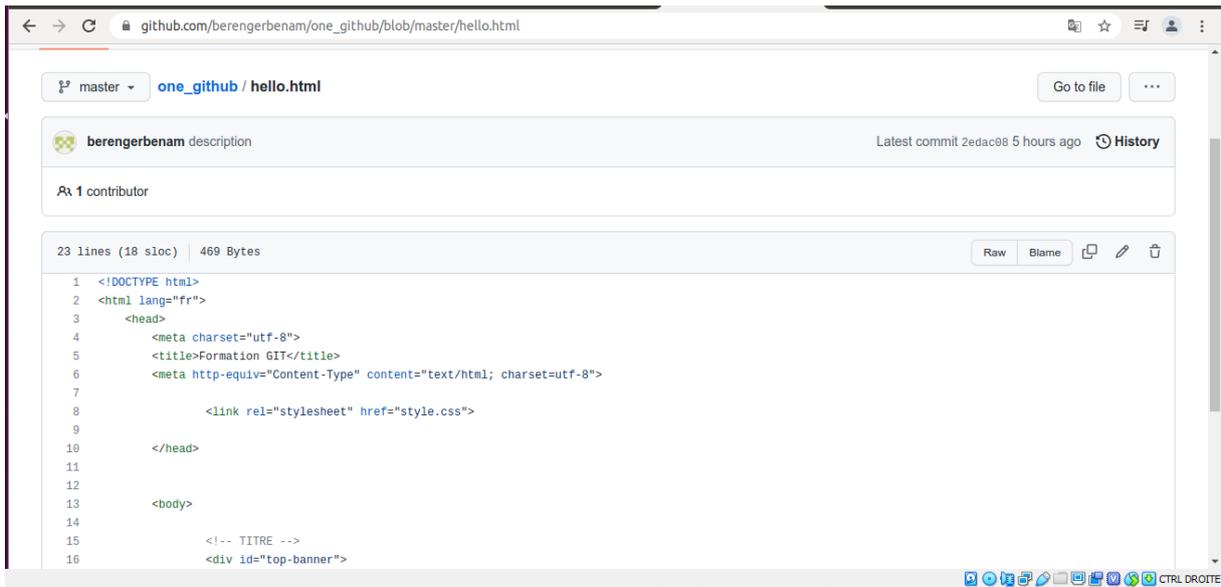
Copier le jeton et relancer la commande puis coller a la place de mot de passe



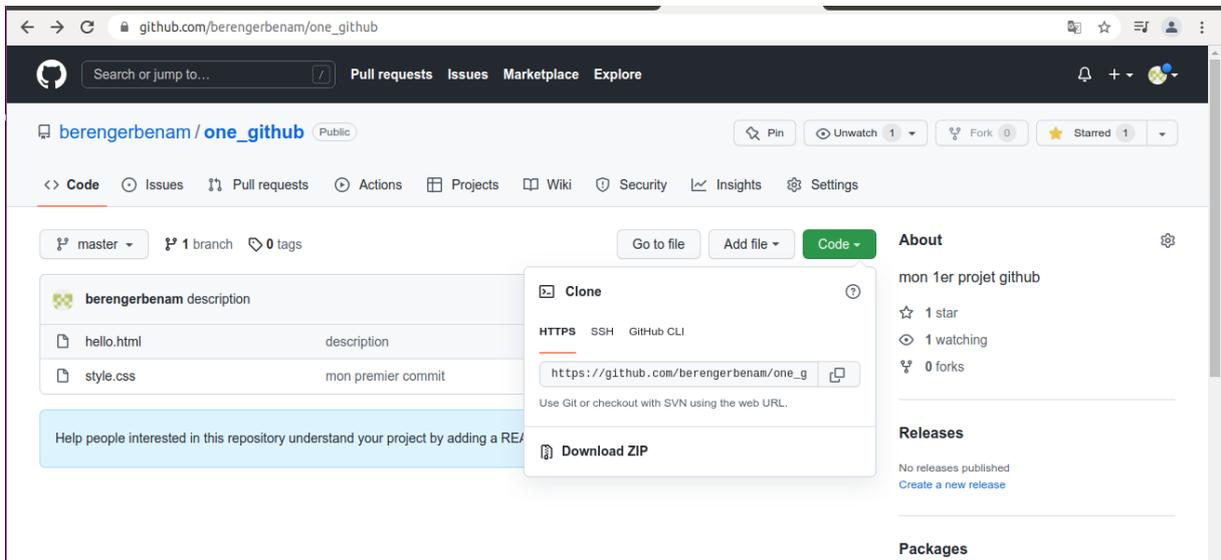
Donc c'est important de lire les erreurs c'est juste un problème de jeton il faut générer un token pour la 1ere fois maintenant la prochaine fois il ne demande pas.on actualise



On actualise la page on trouve nos deux fichiers **hello.html** et **style.css**.



Super maintenant on peut télécharger même le code sur mon compte github.



Super !

On va essayer de faire un changement dans le code **hello.html** et voir le comportement sur le compte github.

```
<body>
  <!-- TITRE -->
  <div id="top-banner">
    Bienvenue dans la page GIT !
  </div>
  <div>
    <p>Bonjour. je m'appelle Berenger c'est ma 1ere fois d'utiliser git et je kiffe trop</p>
  </div><br><br>
  <h2>Ajouter les informations d'un étudiant</h2><br>
  <h1>Ajout Github</h1>
  <label>Prénom</label>
  <p><input type="text" name="prn"></p>
  <label>Nom</label>
  <p><input type="text" name="nom"></p>
  <label>telephone</label>
  <p><input type="text" name="telephone"></p>
  <label>adresse</label>
  <p><input type="text" name="adresse"></p>
  <label>email</label>
  <p><input type="text" name="email"></p>
  <label>service</label>
  <p><input type="text" name="service"></p>
  <input type="submit" value="valider">
  </body>
</html>
root@berenger:~/Projet_git/mon_dossier_web#
```

Voici l'ajout que j'ai fait dans le fichier **hello.html**.

Si on fait **git status**

```
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

   modifié :      hello.html

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
root@berenger:~/Projet_git/mon_dossier_web#
```

On voit que Ya changement.au niveau du fichier **hello.html**.

Il faut ajouter le fichier hello.html via la commande **git add hello.html**

```
root@berenger:~/Projet_git/mon_dossier_web# git add hello.html
root@berenger:~/Projet_git/mon_dossier_web# git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.
Modifications qui seront validées :
  (utilisez "git reset HEAD <fichier>..." pour désindexer)

   modifié :      hello.html

root@berenger:~/Projet_git/mon_dossier_web#
```

Maintenant il est ajouté au dépôt.

```
root@berenger:~/Projet_git/mon_dossier_web# git commit -m 'changement du code hello.html'
[master 4c4e0dc] changement du code hello.html
 1 file changed, 23 insertions(+), 1 deletion(-)
root@berenger:~/Projet_git/mon_dossier_web#
```

On voit bien que le changement est appliqué avec une nouvelle description.

-on va renvoyer les codes via le compte github en utilisant la commande **git push**

```
root@berenger:~/Projet_git/mon_dossier_web# git push
warning: push.default n'est pas défini ; sa valeur implicite a changé dans Git 2.0
de 'matching' vers 'simple'. Pour supprimer ce message et maintenir
le comportement actuel après la modification de la valeur de défaut, utilisez :

git config --global push.default matching

Pour supprimer ce message et adopter le nouveau comportement maintenant, utilisez :

git config --global push.default simple

Quand push.default vaudra 'matching', git poussera les branches locales
sur les branches distantes qui existent déjà avec le même nom.

Depuis Git 2.0, Git utilise par défaut le comportement plus conservatif 'simple'
qui ne pousse la branche courante que vers la branche distante correspondante
que 'git pull' utilise pour mettre à jour la branche courante.

Voir 'git help config' et chercher 'push.default' pour plus d'information.
(Le mode 'simple' a été introduit dans Git 1.7.11. Utilisez le mode similaire
'current' au lieu de 'simple' si vous utilisez de temps en temps d'anciennes versions de Git)

Username for 'https://github.com': Berenger Benam
Password for 'https://Berenger Benam@github.com':
Décompte des objets: 3, fait.
Delta compression using up to 2 threads.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 753 bytes | 0 bytes/s, fait.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/berengerbenam/one_github.git
 2edac08..4c4e0dc master -> master
root@berenger:~/Projet_git/mon_dossier_web#
```

On voit ça marche bien maintenant on actualise la page github.

```
root@berenger:~/Projet_git/mon_dossier_web# git push
warning: push.default n'est pas défini ; sa valeur implicite a changé dans Git 2.0
de 'matching' vers 'simple'. Pour supprimer ce message et maintenir
le comportement actuel après la modification de la valeur de défaut, utilisez :

git config --global push.default matching

Pour supprimer ce message et adopter le nouveau comportement maintenant, utilisez :

git config --global push.default simple

Quand push.default vaudra 'matching', git poussera les branches locales
sur les branches distantes qui existent déjà avec le même nom.

Depuis Git 2.0, Git utilise par défaut le comportement plus conservatif 'simple'
qui ne pousse la branche courante que vers la branche distante correspondante
que 'git pull' utilise pour mettre à jour la branche courante.

Voir 'git help config' et chercher 'push.default' pour plus d'information.
(Le mode 'simple' a été introduit dans Git 1.7.11. Utilisez le mode similaire
'current' au lieu de 'simple' si vous utilisez de temps en temps d'anciennes versions de Git)

Username for 'https://github.com': Berenger Benam
Password for 'https://Berenger Benam@github.com':
Décompte des objets: 3, fait.
Delta compression using up to 2 threads.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 753 bytes | 0 bytes/s, fait.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/berengerbenam/one_github.git
 2edac08..4c4e0dc master -> master
root@berenger:~/Projet_git/mon_dossier_web#
```

TEST : sur navigateur

file:///home/berenger/Projet\_git/mon\_dossier\_web/hello.html

### Ajouter les informations d'un étudiant

#### Ajout Github

Prénom

Nom

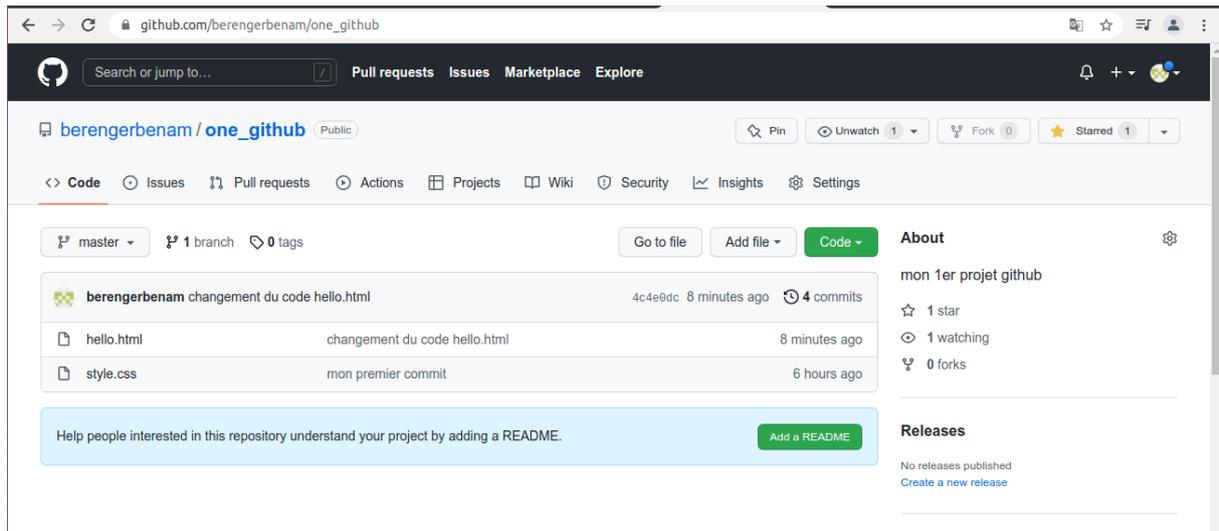
telephone

adresse

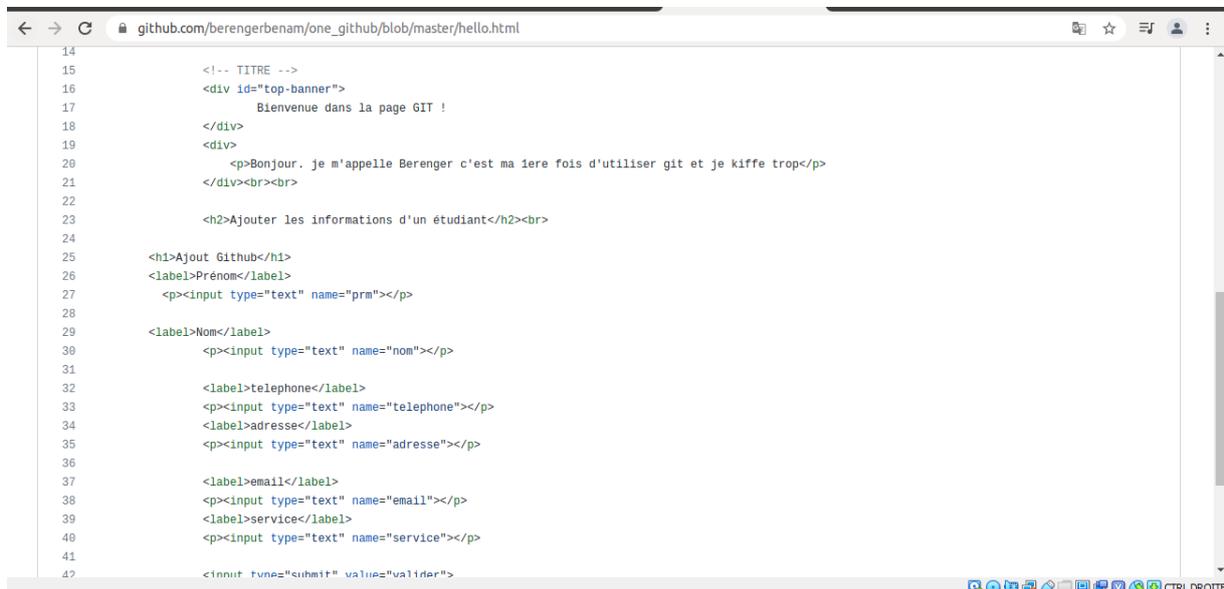
email

service

Idem pour le compte github



On voit le changement



Super !

## ➤ Comment Héberger un site gratuit sur GitHub

On crée le dossier du projet voir la capture :

```
root@berenger:~# mkdir Site_GitHub_Berenger
root@berenger:~# cd Site_GitHub_Berenger/
root@berenger:~/Site_GitHub_Berenger#
```

On crée le fichier index.html

```
root@berenger:~/Site_GitHub_Berenger# touch index.html
```

```

GNU nano 2.5.3                               Fichier : index.html                               Modifié
<!DOCTYPE html>
<html>
  <head>
    <title>RTN/EC2LT</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    </header>
    <marquee behavior="alternate">
      <h3>
        RESEAUX ET TECHNIQUES NUMERIQUES
      </h3>
    </marquee>
    <section>
      <aside>
        
      </aside>
      <h2>
        <nav>
          <a href=".." />RESEAUX</a>
          <a href=".." />DEVELOPPEMENT</a>
          <a href=".." />TELECOMS</a>
          <a href="http://www.ec2lt.sn">EC2LT</a>
        </nav>
      </h2>
    </section>
  </body>
</html>

```

```

    <div id="deuxiemeTrait"></div>
    <div id="copyrightEtIcons">
      <div id="copyright">
        <span>rtn@rtn.sn</span>
        <span>(+221) 77 446 71 63 | | (+221) 33 868 18 85 </span>
      </div>
      <div id="icons">
        <a href="http://www.twitter.fr"><i class="fa fa-twitter"></i></a>
        <a href="http://www.facebook.fr"><i class="fa fa-facebook"></i></a>
        <a href="http://www.instagram.com"><i class="fa fa-instagram"></i></a>
      </div>
    </div>
  </div>
</body>
</html>

```

Voici le contenu du fichier **index.html**

On crée le deuxième fichier **contactez-nous.html**

```

root@berenger:~/Site_GitHub_Berenger# touch contactez-nous.html

```

Le code :

```

GNU nano 2.5.3                               Fichier : contactez-nous.html
<html>
  <head>
    <title>Contactez-nous</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h2 id="contact">Contactez-nous</h2>
    <form>
      <input placeholder="Nom">
      <input placeholder="Prenom">
      <input placeholder="Profil">
      <input placeholder="E-mail">
      <input placeholder="Téléphone">
      <input placeholder="Ville">
      <textarea placeholder="Motif de la demande"></textarea>
      <button>Valider</button>
    </form>
  </body>
</html>

```

On crée le fichier **style.css**

```

root@berenger:~/Site_GitHub_Berenger# touch style.css

```

Code :

```
GNU nano 2.5.3 Fichier : style.css
body { background: #00ffff;
}
h2 {
font-family: 'Dancing Script', cursive;
text-align: center;
color: antiquewhite;
font-size: 3em;
text-shadow: 1px 3px 2px black;
}
h3 {
font-family: 'Dancing Script', cursive;
text-align: center;
color: antiquewhite;
font-size: 3em;
text-shadow: 1px 3px 2px black;
}
footer {
background-color: #524A3A;
color: #FFFAE1;
padding: 20px 0 10px 0;
text-align: center;
}
form {
margin: 0 auto;
max-width: 900px;
}
input, textarea, button {
border: none;
padding: 15px 10px;
margin: 1px 0;
font-size: 1.2em;
font-family: 'Advent Pro', sans-serif;
border-radius: 8px;
}
textarea {
height: 120px;
}
button {
font-size: 1.5em;
background-color: #FFFAE1;
}
button:hover {
background-color: #e55039;
color: #FFFAE1;
transition: 0.3s ease-in-out;
}
#deuxiemeTrait {
height: 1px;
width: 75%;
background-color: #FFFAE1;
margin: 60px auto;
}
#copyrightEtIcons {
display: flex;
margin-bottom: 20px;
padding: 0 10%;
}
#copyright {
width: 50%;
transition: 0.3s ease-in-out;
}
#deuxiemeTrait {
height: 1px;
width: 75%;
background-color: #FFFAE1;
margin: 60px auto;
}
#copyrightEtIcons {
display: flex;
margin-bottom: 20px;
padding: 0 10%;
}
#copyright {
width: 50%;
text-align: left;
color: #FFFAE1;
}
#icons {
width: 50%;
text-align: right;
}
#icons a {
display: inline-block;
padding: 0 15px;
font-size: 1.3em;
color: #FFFAE1;
}
#icons a:hover {
transform: scale(1.5);
transition: 0.1s;
}
```

```
padding: 15px 10px;
margin: 1px 0;
font-size: 1.2em;
font-family: 'Advent Pro', sans-serif;
border-radius: 8px;
}
textarea {
height: 120px;
}
button {
font-size: 1.5em;
background-color: #FFFAE1;
}
button:hover {
background-color: #e55039;
color: #FFFAE1;
transition: 0.3s ease-in-out;
}
#deuxiemeTrait {
height: 1px;
width: 75%;
background-color: #FFFAE1;
margin: 60px auto;
}
#copyrightEtIcons {
display: flex;
margin-bottom: 20px;
padding: 0 10%;
}
#copyright {
width: 50%;
```

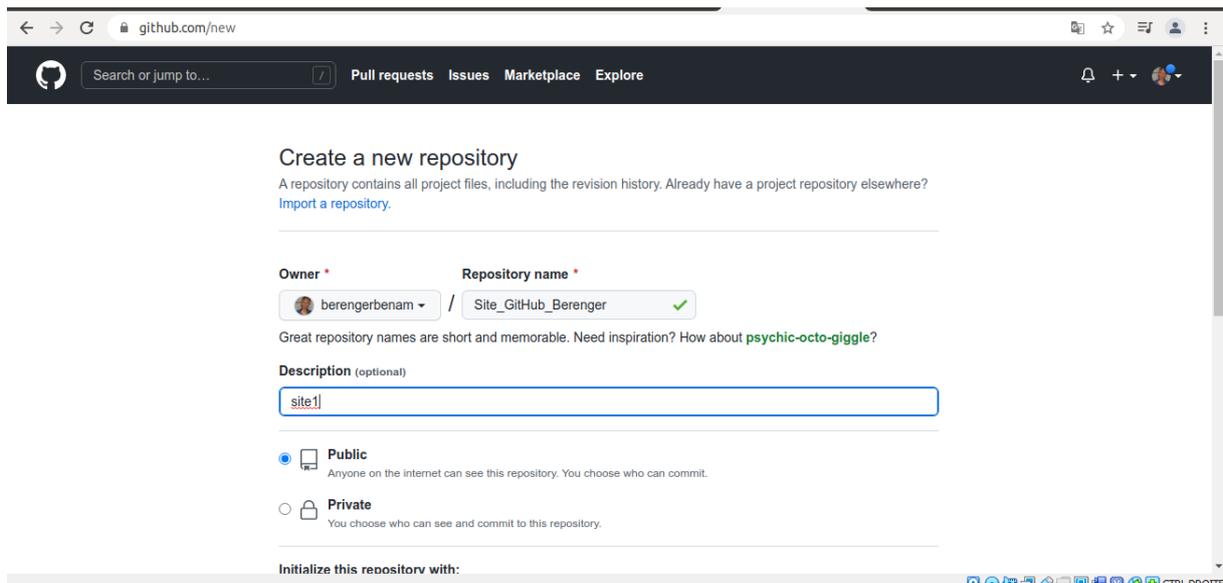
```
transition: 0.3s ease-in-out;
}
#deuxiemeTrait {
height: 1px;
width: 75%;
background-color: #FFFAE1;
margin: 60px auto;
}
#copyrightEtIcons {
display: flex;
margin-bottom: 20px;
padding: 0 10%;
}
#copyright {
width: 50%;
text-align: left;
color: #FFFAE1;
}
#icons {
width: 50%;
text-align: right;
}
#icons a {
display: inline-block;
padding: 0 15px;
font-size: 1.3em;
color: #FFFAE1;
}
#icons a:hover {
transform: scale(1.5);
transition: 0.1s;
}
```

Et voici le contenu du dossier Site\_GitHub\_Berenger

```
root@berenger:~/Site_GitHub_Berenger# ls
contactez-nous.html index.html logo.png style.css
root@berenger:~/Site_GitHub_Berenger#
```

On a une image logo.png

## On crée un repository



github.com/new

Search or jump to... Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*  / Repository name \*

Great repository names are short and memorable. Need inspiration? How about [psychic-octo-giggle?](#)

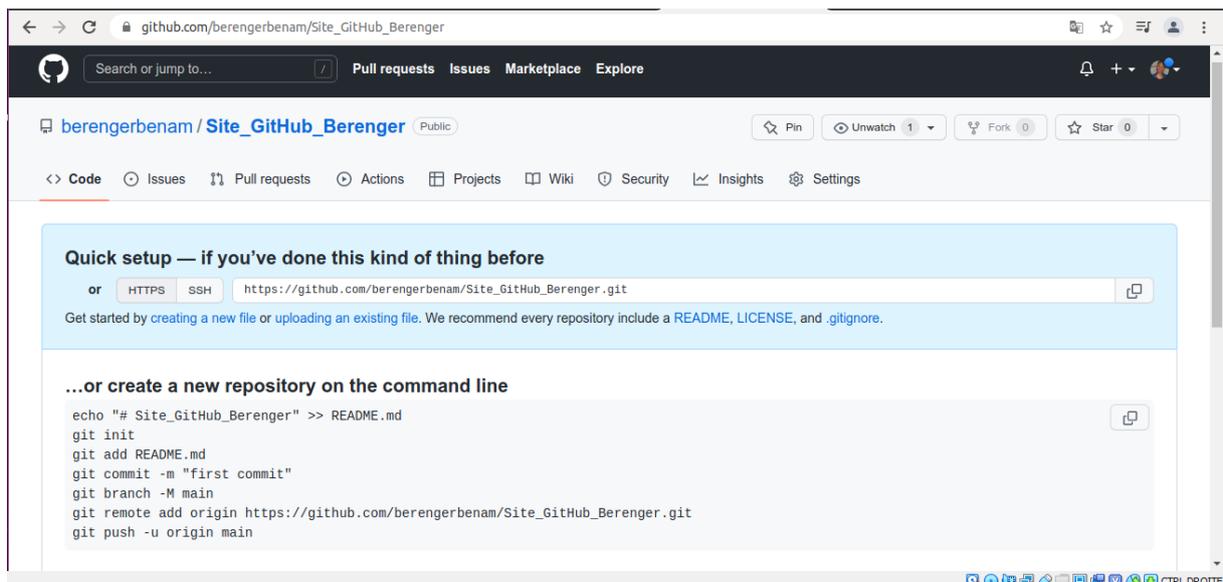
Description (optional)

Public  
Anyone on the internet can see this repository. You choose who can commit.

Private  
You choose who can see and commit to this repository.

Initialize this repository with:

Le nom c'est **Site\_GitHub\_Berenger** puis cliquez sur **create repository**



github.com/berengerbenam/Site\_GitHub\_Berenger

berengerbenam / Site\_GitHub\_Berenger Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

#### Quick setup — if you've done this kind of thing before

or  HTTPS  SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

#### ...or create a new repository on the command line

```
echo "# Site_GitHub_Berenger" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/berengerbenam/Site_GitHub_Berenger.git
git push -u origin main
```

Maintenant on tombe sur cette page ses commandes nous permet d'envoyer notre dépôt locale vers notre dépôt distant qu'on vient de créer.

On fait **git init** pour initialiser le dépôt

```
root@berenger:~/Site_GitHub_Berenger# git init
Dépôt Git vide initialisé dans /home/berenger/Site_GitHub_Berenger/.git/
root@berenger:~/Site_GitHub_Berenger#
```

On fait **git add .** il va ajouter tous mes fichiers

```
root@berenger:~/Site_GitHub_Berenger# git add .
root@berenger:~/Site_GitHub_Berenger#
```

```

root@berenger:~/Site_GitHub_Berenger# git commit -m 'ajout de mes fichiers'
[master (commit racine) dfca7ac] ajout de mes fichiers
 4 files changed, 153 insertions(+)
 create mode 100644 contactez-nous.html
 create mode 100644 index.html
 create mode 100644 logo.png
 create mode 100644 style.css
root@berenger:~/Site_GitHub_Berenger#

```

On fait commit pour sauvegarder tous les fichiers.

```

root@berenger:~/Site_GitHub_Berenger# git branch -M main
root@berenger:~/Site_GitHub_Berenger#

```

Cette commande permet de renommer la branche master a main

Si on fait **git branch**

```

root@berenger:~/Site_GitHub_Berenger# git branch
* main
root@berenger:~/Site_GitHub_Berenger#

```

Il a renommé la branche master à main.

```

root@berenger:~/Site_GitHub_Berenger# git remote add origin https://github.com/berengerbenam/Site_GitHub_Berenger.git
root@berenger:~/Site_GitHub_Berenger#

```

Cette ligne permet de lier le dépôt local à dépôt distant.

```

root@berenger:~/Site_GitHub_Berenger# git push -u origin main
Username for 'https://github.com': Berenger Benam
Password for 'https://Berenger.Benam@github.com':
Décompse des objets: 6, fait.
Delta compression using up to 2 threads.
Compression des objets: 100% (6/6), fait.
Écriture des objets: 100% (6/6), 64.30 KiB | 0 bytes/s, fait.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/berengerbenam/Site_GitHub_Berenger.git
 * [new branch]      main -> main
La branche main est paramétrée pour suivre la branche distante main depuis origin.
root@berenger:~/Site_GitHub_Berenger#

```

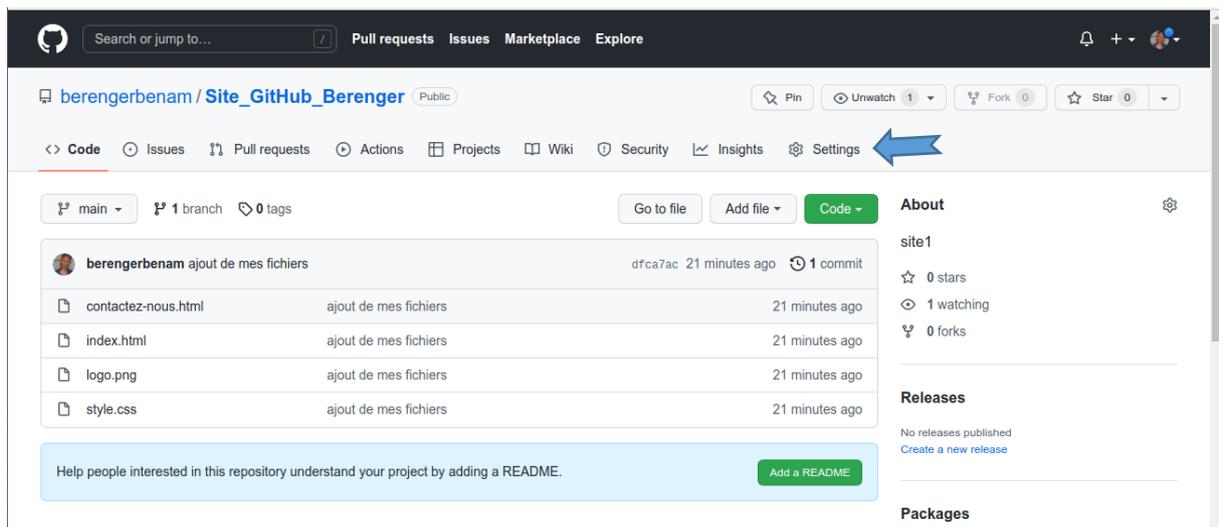
On envoie le dépôt local Vers le dépôt distant.

On actualise

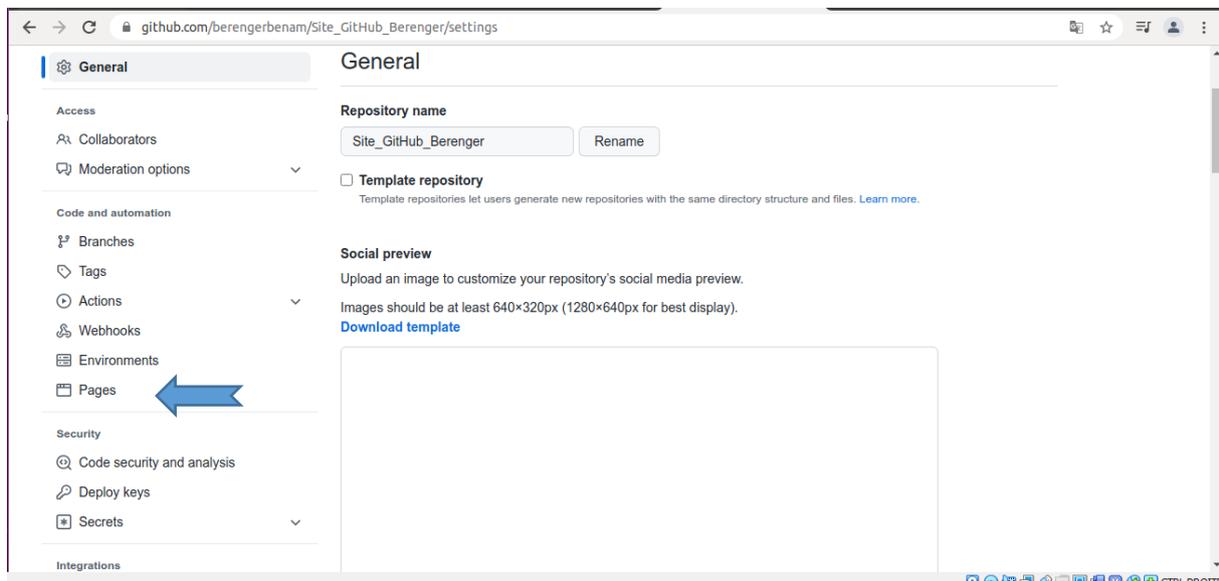
The screenshot shows the GitHub web interface for the repository 'berengerbenam / Site\_GitHub\_Berenger'. The main content area displays a commit by 'berengerbenam' with the message 'ajout de mes fichiers' (dfca7ac, 15 minutes ago). The commit includes four files: 'contactez-nous.html', 'index.html', 'logo.png', and 'style.css', all added 15 minutes ago. The repository has 1 branch (main) and 0 tags. The right sidebar shows repository statistics: 0 stars, 1 watching, and 0 forks. There are also sections for 'Releases' (no releases published) and 'Packages'.

On voit bien tous les codes. On arrive à envoyer le dépôt local vers le dépôt distant

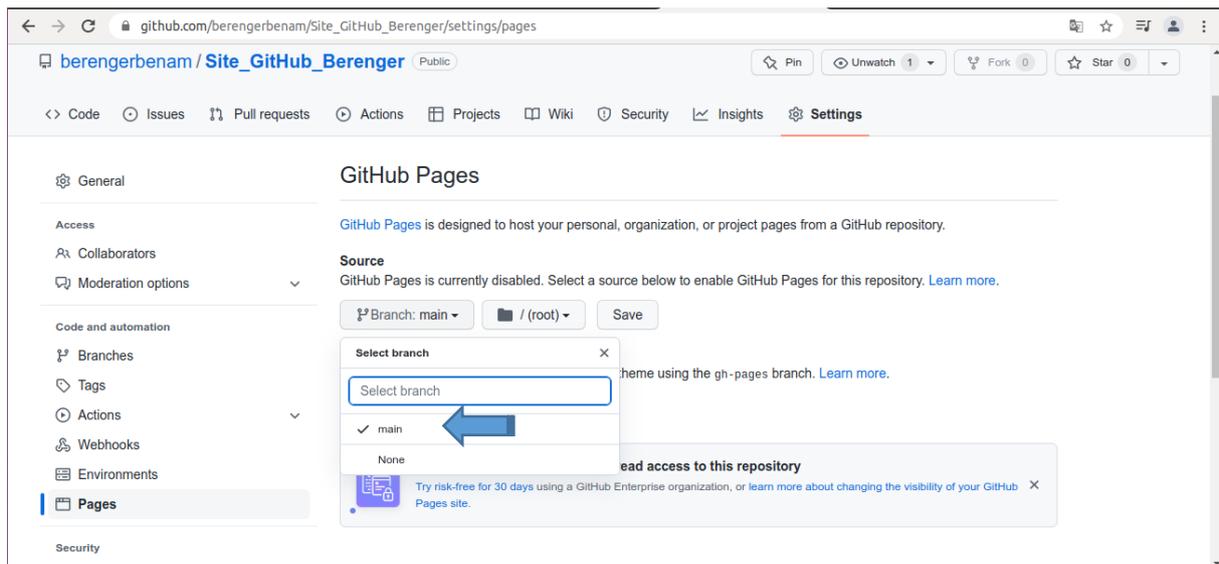
Il reste quelque chose à faire on a modifié la branche tout à l'heure.



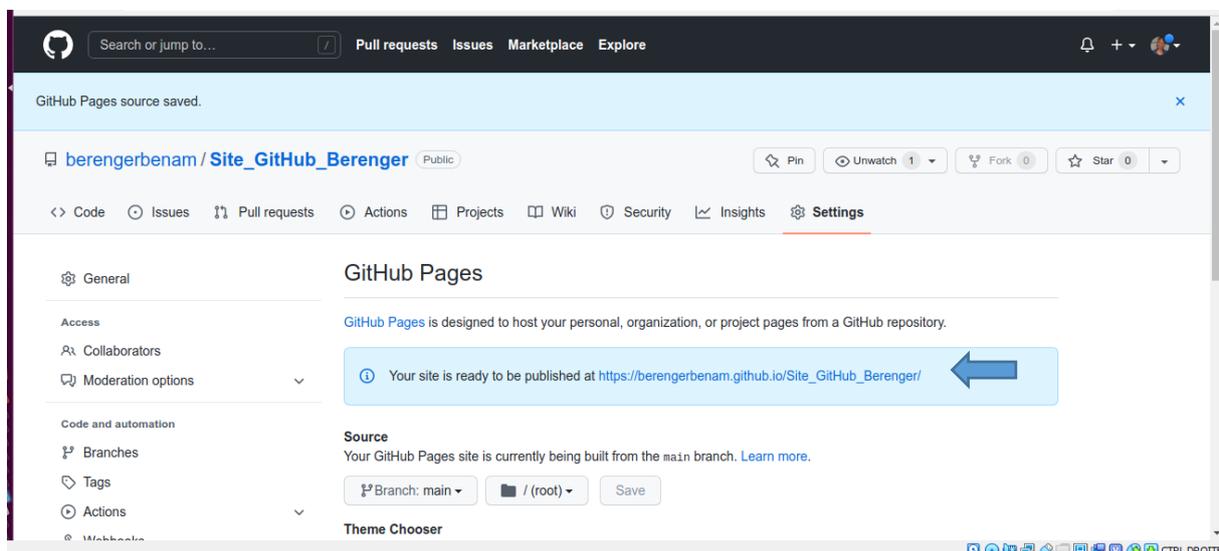
On clique sur setting



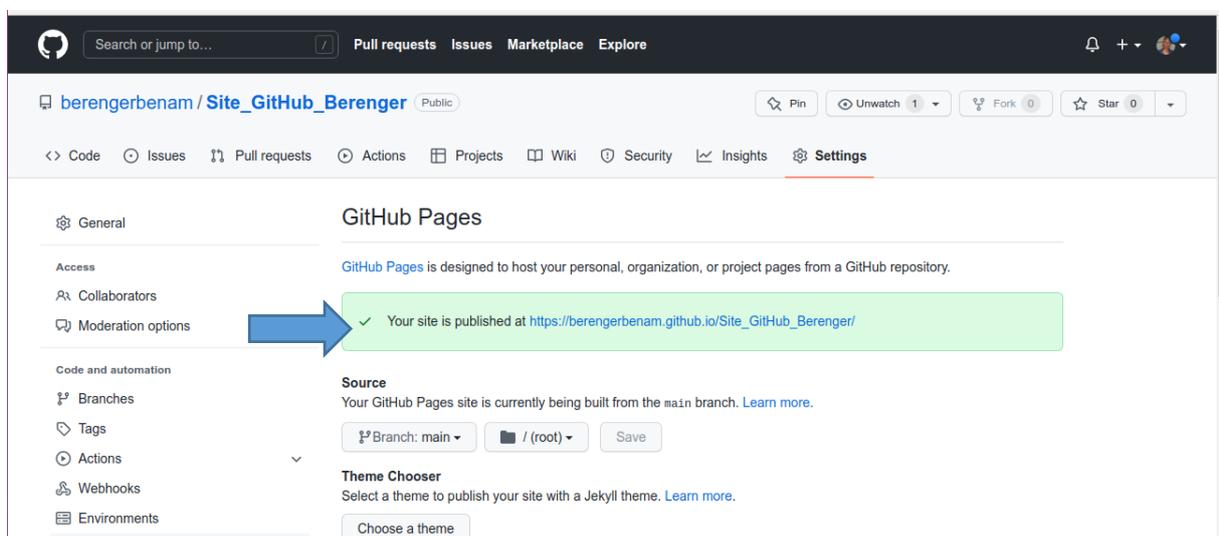
On clique sur Pages



On choisit la branche main puis cliquer sur Save.



Il faut patienter pendant quelques minutes pour que la page charge si toute marche la page charge en vert.

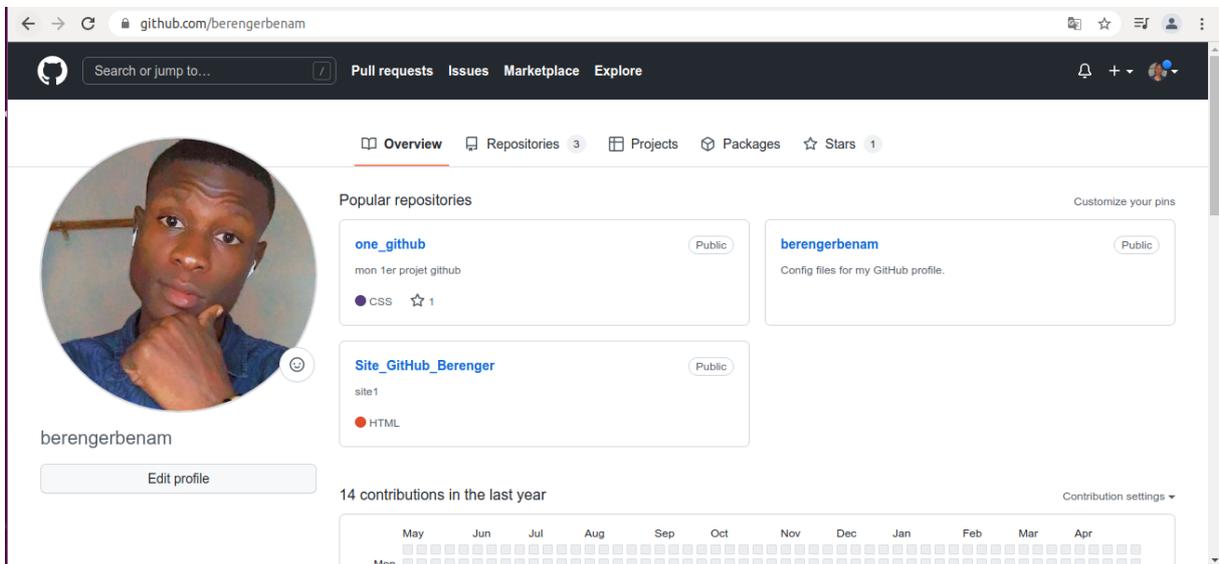


Cette flèche indique que la page a été chargée correctement et on clique sur l'URL.

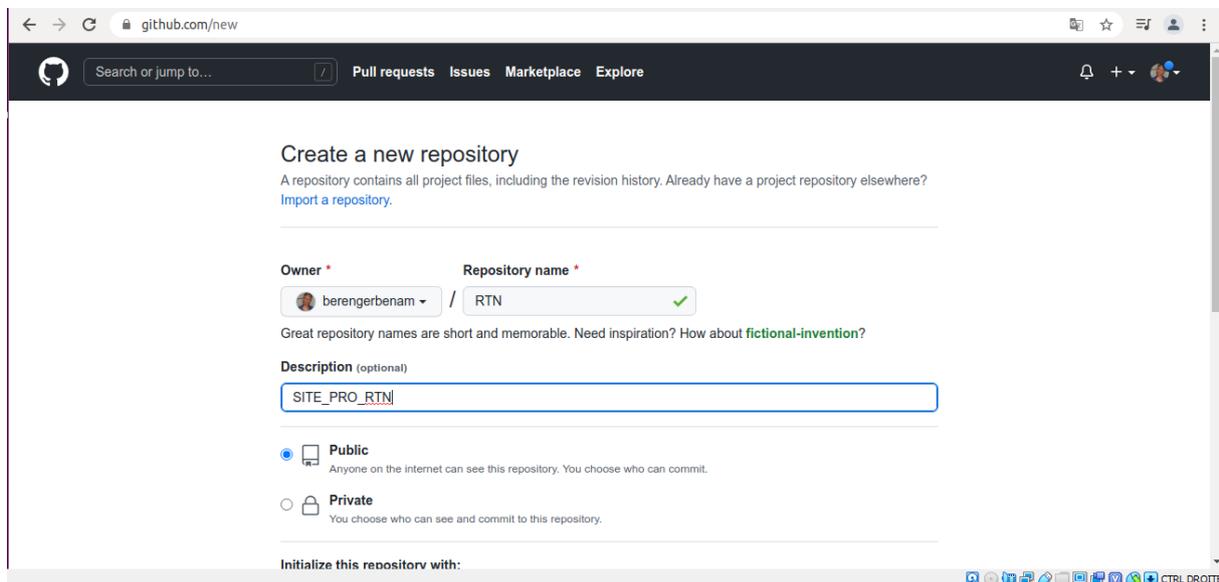
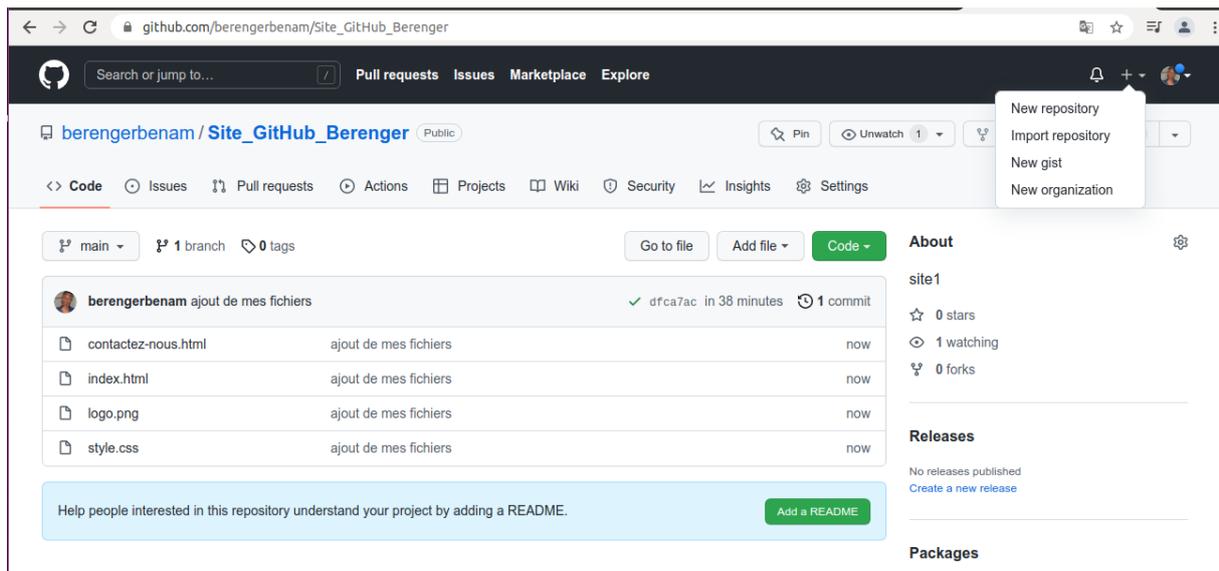


Si on clique sur ce lien on tombe sur le site RTN. Voici le lien

[https://berengerbenam.github.io/Site\\_GitHub\\_Berenger/](https://berengerbenam.github.io/Site_GitHub_Berenger/)



On créer le site RTN



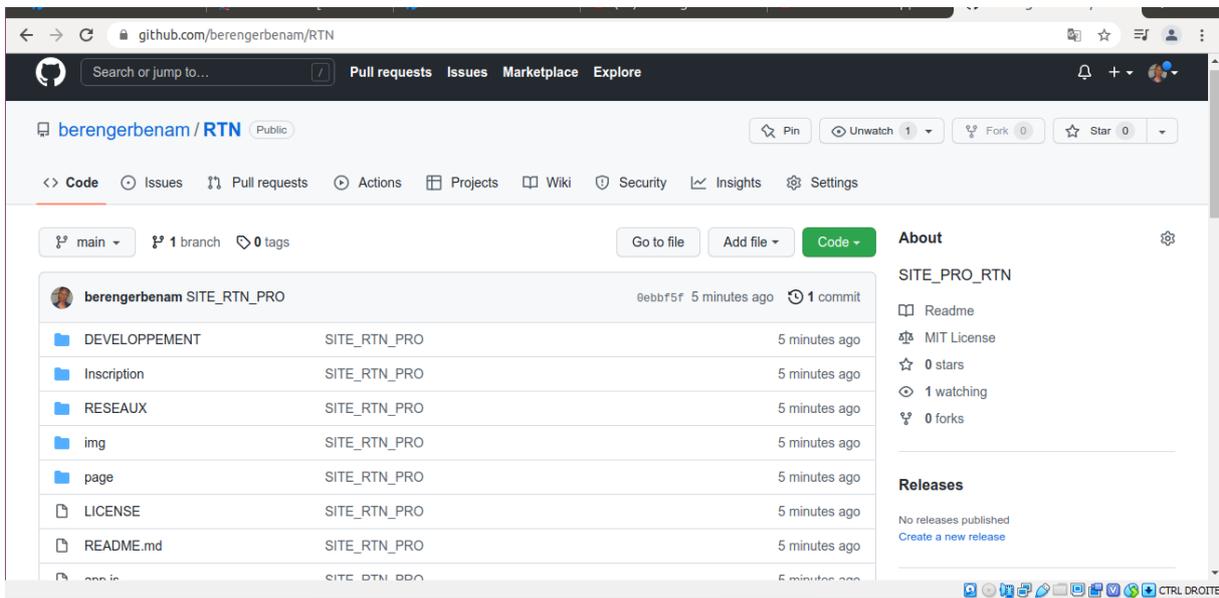
Cliquez sur create repository

Création du dossier **SITE\_RTN**



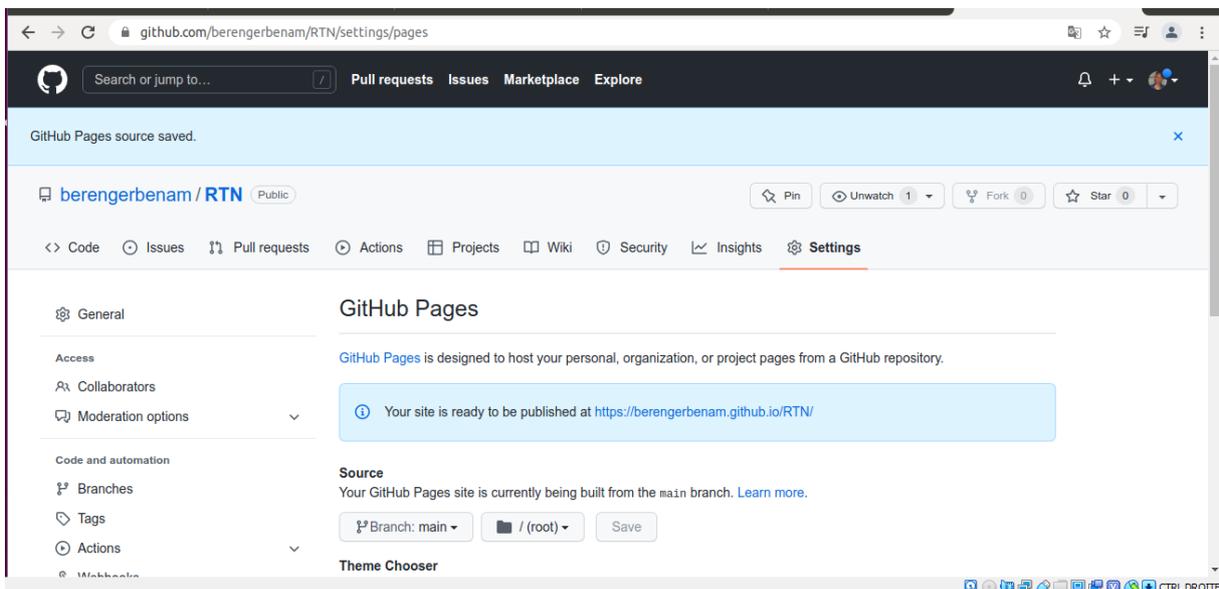
```
root@berenger:~/SITE_RTN# git branch -M main
root@berenger:~/SITE_RTN# git branch
* main
root@berenger:~/SITE_RTN# git remote add origin https://github.com/berengerbenam/RTN.git
root@berenger:~/SITE_RTN# git push -u origin main
Username for 'https://github.com': Berenger Benam
Password for 'https://Berenger Benam@github.com':
Décompte des objets: 294, fait.
Delta compression using up to 2 threads.
Compression des objets: 100% (290/290), fait.
Écriture des objets: 100% (294/294), 36.95 MiB | 478.00 KiB/s, fait.
Total 294 (delta 11), reused 0 (delta 0)
remote: Resolving deltas: 100% (11/11), done.
To https://github.com/berengerbenam/RTN.git
 * [new branch]      main -> main
La branche main est paramétrée pour suivre la branche distante main depuis origin.
root@berenger:~/SITE_RTN#
```

On actualise la page

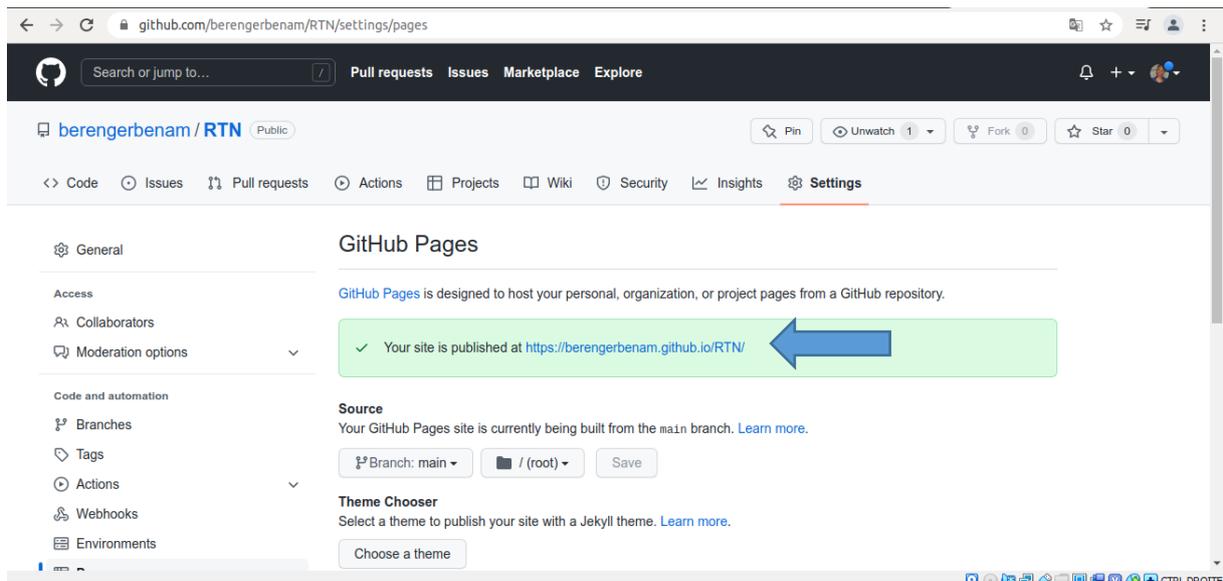


Super !

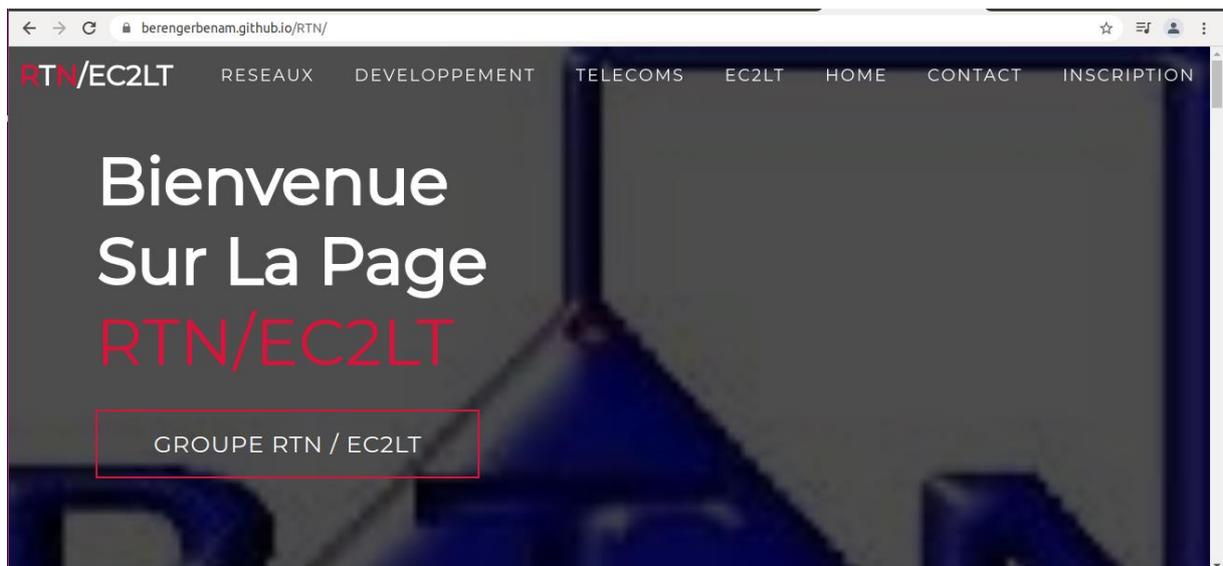
On choisit la branche main

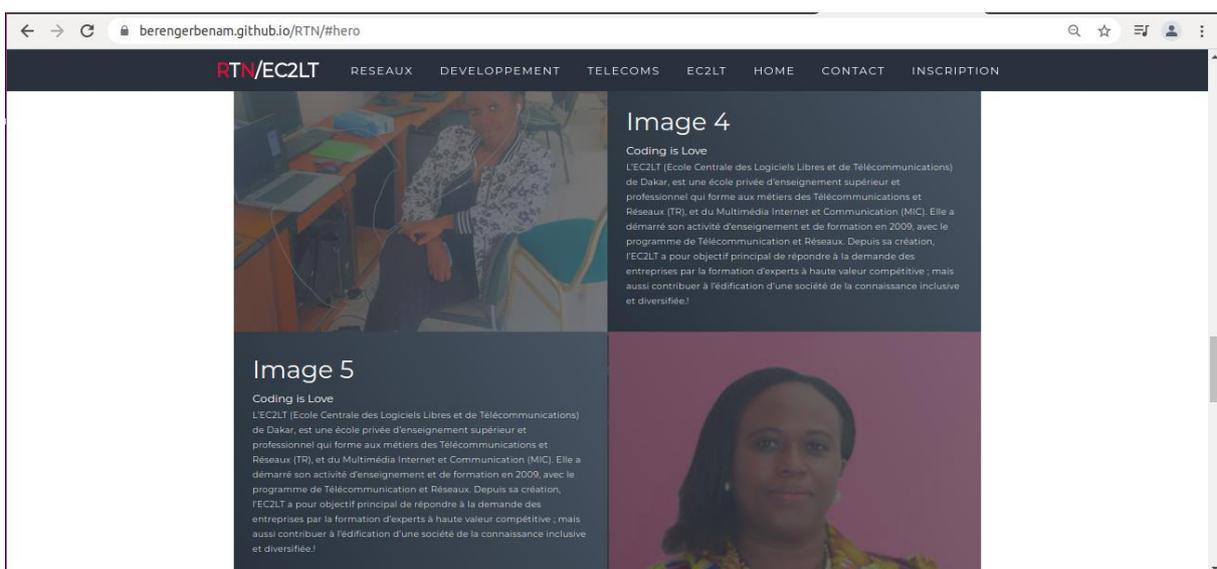
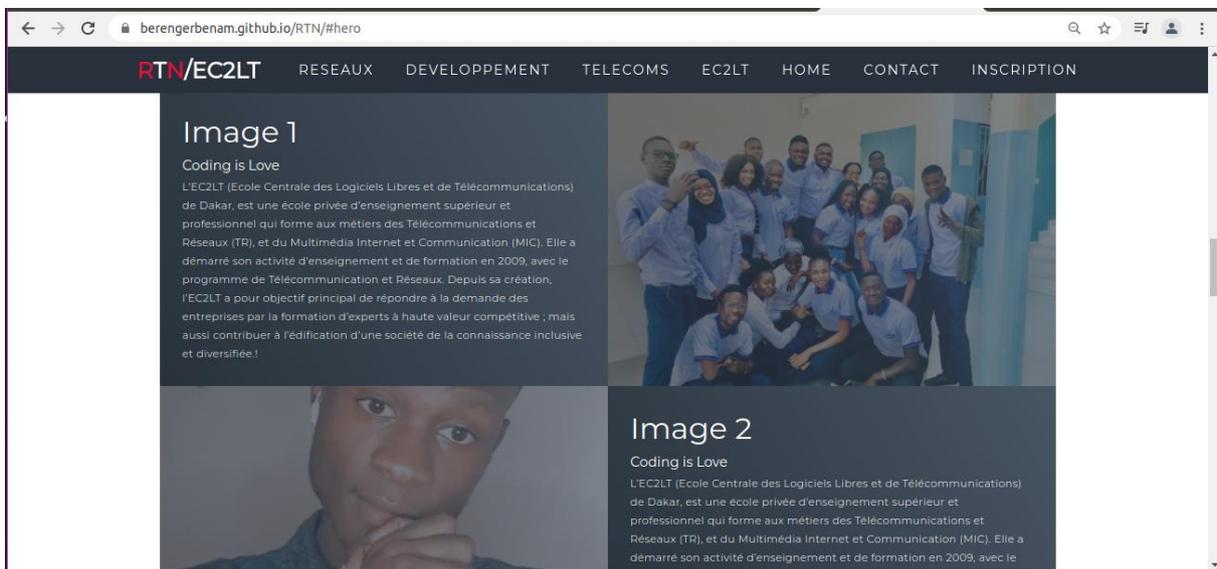
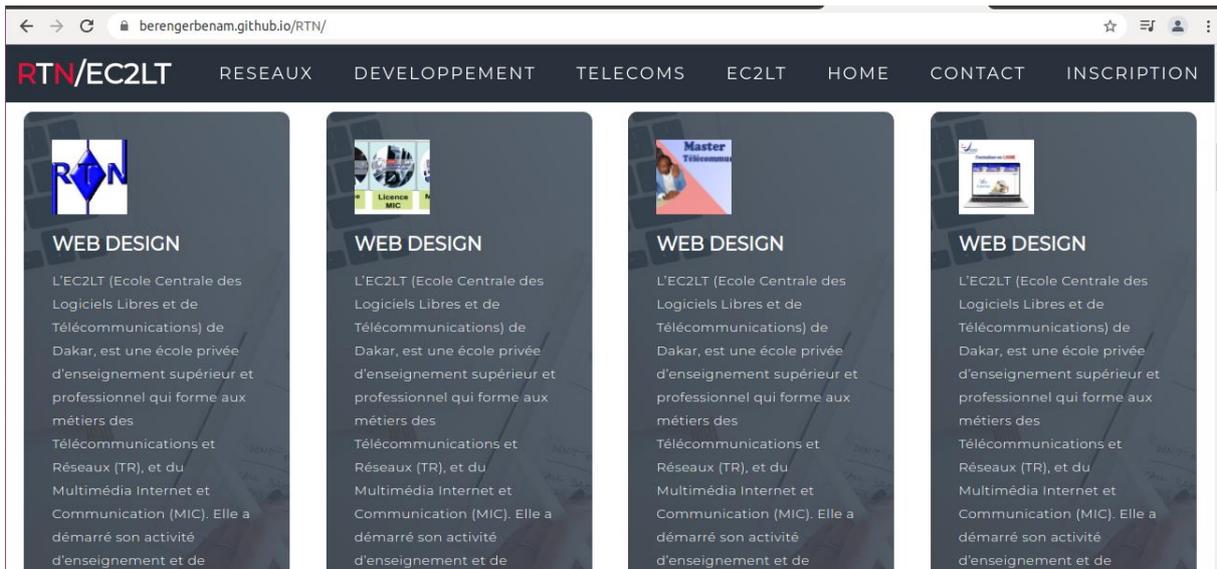


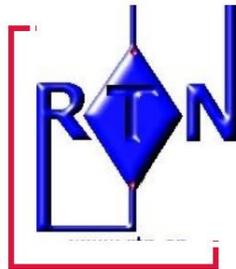
Il faut patienter que la page charge.



La page a été chargée on clique sur l'url.







# ABOUT

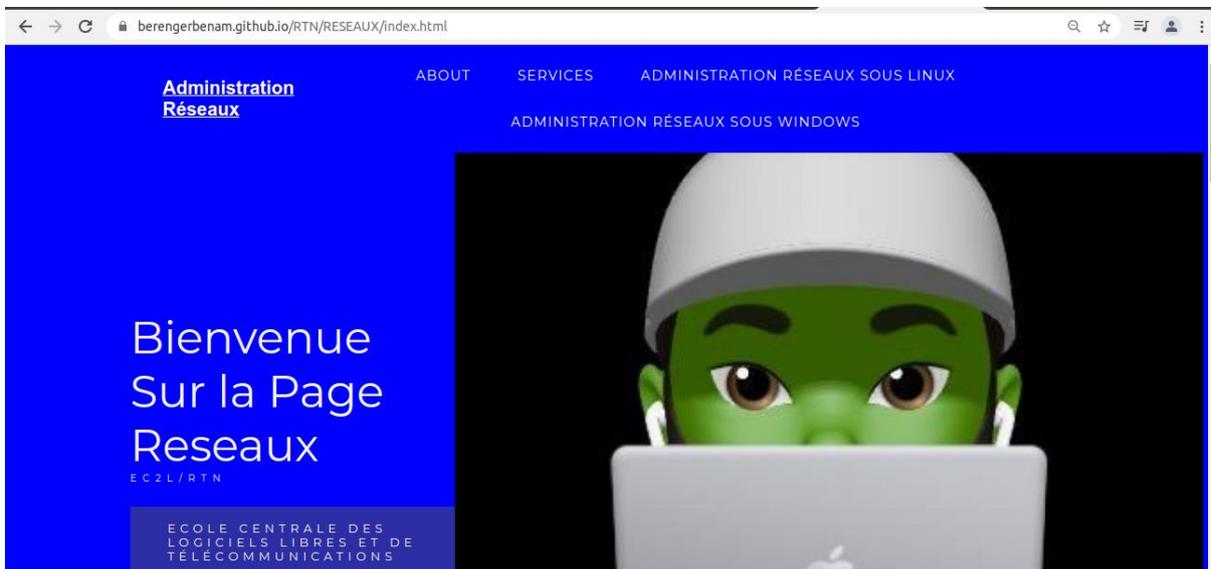
**Front End Developer**

03/06/2021 C'est Officiel !!! Le groupe RTN relance ses activités à travers des formations accélérées à des prix promotionnels dans les modules suivants : - Administration système Linux - Services réseau de base en environnement Linux et Windows - Administration des réseaux locaux et équipements actifs - Protocoles sécurisés - Services d'annuaire et d'authentification - Service d'authentification RADIUS, etc... Vous êtes étudiant ou professionnel et vous souhaitez vous perfectionner ou découvrir le monde de l'administration système et réseau ? Réservez dès maintenant votre place en appelant le 33 824 10 60. L'Ecole Centrale des Logiciels Libres et de Télécommunications Site web : <https://www.ec2lt.sn/> L'une pour ne pas dire la première école dans le domaine du logiciel libre en Afrique Francophone ; L'EC2LT (Ecole Centrale des Logiciels Libres et de Télécommunications) de Dakar, est une école privée d'enseignement supérieur et professionnel qui forme aux métiers des Télécommunications et Réseaux (TR), et du Multimédia Internet et Communication (MIC.)

RTN/EC2LT

Voici la page du site RTN.

On clique sur la page RESEAUX



berengerbenam.github.io/RTN/RESEAUX/index.html

Administration Réseaux

ABOUT SERVICES ADMINISTRATION RÉSEAUX SOUS LINUX ADMINISTRATION RÉSEAUX SOUS WINDOWS

## Bienvenue Sur la Page Réseaux

EC2LT/RTN

ECOLE CENTRALE DES LOGICIELS LIBRES ET DE TÉLÉCOMMUNICATIONS



Si on clique sur ABOUT



berengerbenam.github.io/RTN/RESEAUX/index.html#about

Administration Réseaux

ABOUT SERVICES ADMINISTRATION RÉSEAUX SOUS LINUX ADMINISTRATION RÉSEAUX SOUS WINDOWS

## SYNTHÈSE

En matière de réseaux informatiques, on peut aussi distinguer les réseaux privés des réseaux publics. Cette différence se situe au niveau de la confidentialité des données. Dans le cas d'un réseau privé, toutes les données qui transitent sur le réseau ainsi que toutes les ressources utilisées sont strictement réservées à la sphère privée, comme au sein d'une maison ou d'une entreprise. Les machines n'ont donc pas forcément besoin d'Internet pour communiquer entre elles. On peut ici aussi distinguer le réseau domestique du réseau d'entreprise. Un LAN domestique ne permet pas d'organiser les ordinateurs présents sur le réseau en domaine d'entreprise, mais en groupe de travail seulement. Cela est pourtant possible sur un réseau d'entreprise. Les réseaux privés incluent également les VPN ou Virtual Private Network, qui permettent de relier différents réseaux locaux distants en utilisant un tunnel protégé qui traverse Internet. Ces réseaux doivent appartenir à une même personne morale. On dénombre trois types de VPN : les VPN d'accès, les VPN d'intranet et les VPN d'extranet. Le réseau Ethernet est aujourd'hui le réseau le plus utilisé. Il s'agit d'un protocole permettant de connecter tous les ordinateurs et machines présents sur un réseau local grâce au WiFi (Ethernet sans fil) ou par l'Ethernet filaire. Une fois que des utilisateurs extérieurs peuvent accéder à un réseau, on fera alors référence à un réseau public...



CTRL DROIT



On clique sur la page DEVELOPPEMENT





berengerbenam.github.io/RTN/DEVELOPPEMENT/FRONT-END/js/html

## Les bases de JavaScript :

JavaScript est un langage de programmation qui ajoute de l'interactivité à votre site web (par exemple : jeux, réponses quand on clique sur un bouton ou des données entrées dans des formulaires, composition dynamique, animations). Cet article vous aide à débuter dans ce langage passionnant et vous donne une idée de ses possibilités.

## Qu'est le JavaScript, réellement ?

des Interfaces de Programmation d'Applications pour navigateurs (API) — API intégrées aux navigateurs web permettant de créer dynamiquement du HTML, de définir des styles de CSS, de collecter et manipuler un flux vidéo depuis la webcam de l'utilisateur ou de créer des graphiques 3D et des échantillonnages audio.

des API tierces-parties permettant aux développeurs d'incorporer dans leurs sites des fonctionnalités issues d'autres fournisseurs de contenu, comme Twitter ou Facebook.

des modèles et bibliothèques tierces-parties applicables à votre HTML permettant de mettre en œuvre rapidement des sites et des applications.

JavaScript (« JS » en abrégé) est un langage de programmation dynamique complet qui, appliqué à un document HTML, peut fournir une interactivité dynamique sur les sites Web. Il a été inventé par Brendan Eich, co-fondateur du projet Mozilla, de la Mozilla Foundation et de la Mozilla Corporation.

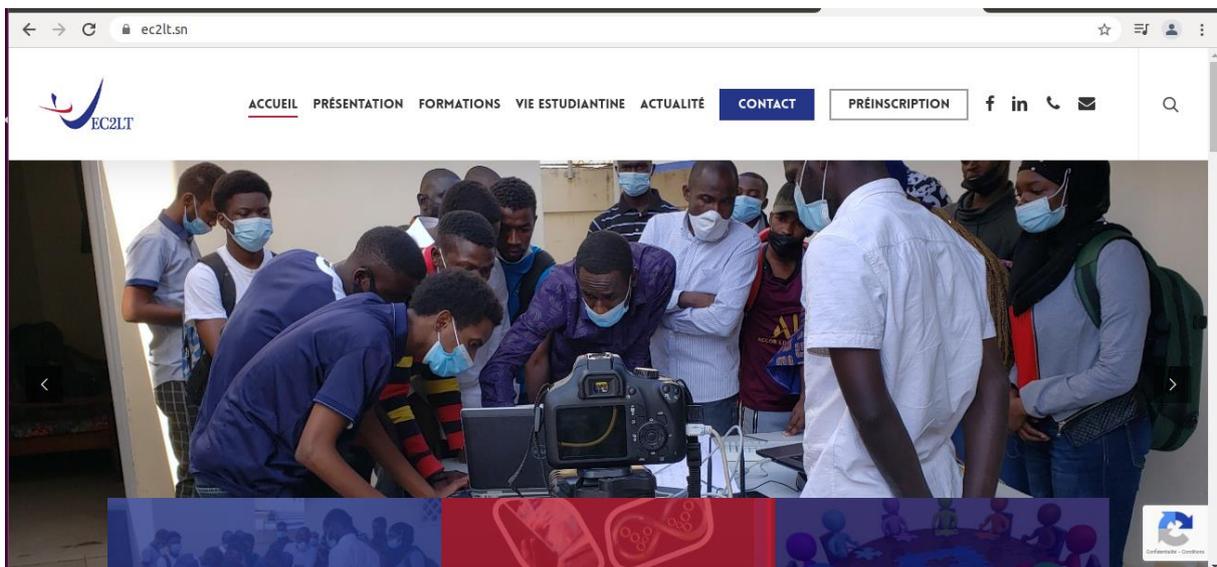
JavaScript est d'une incroyable flexibilité. Vous pouvez commencer petit, avec des carrousels, des galeries d'images, des variations de mises en page et des réponses aux clics de boutons. Avec plus d'expérience, vous serez en mesure de créer des jeux, des graphiques 2D et 3D animés, des applications complètes fondées sur des bases de données et bien plus encore !

JavaScript est plutôt compact tout en étant très souple. Les développeurs ont écrit de nombreux outils sur le cœur du langage JavaScript, créant des fonctionnalités supplémentaires très simplement. Parmi ces outils, il y a :

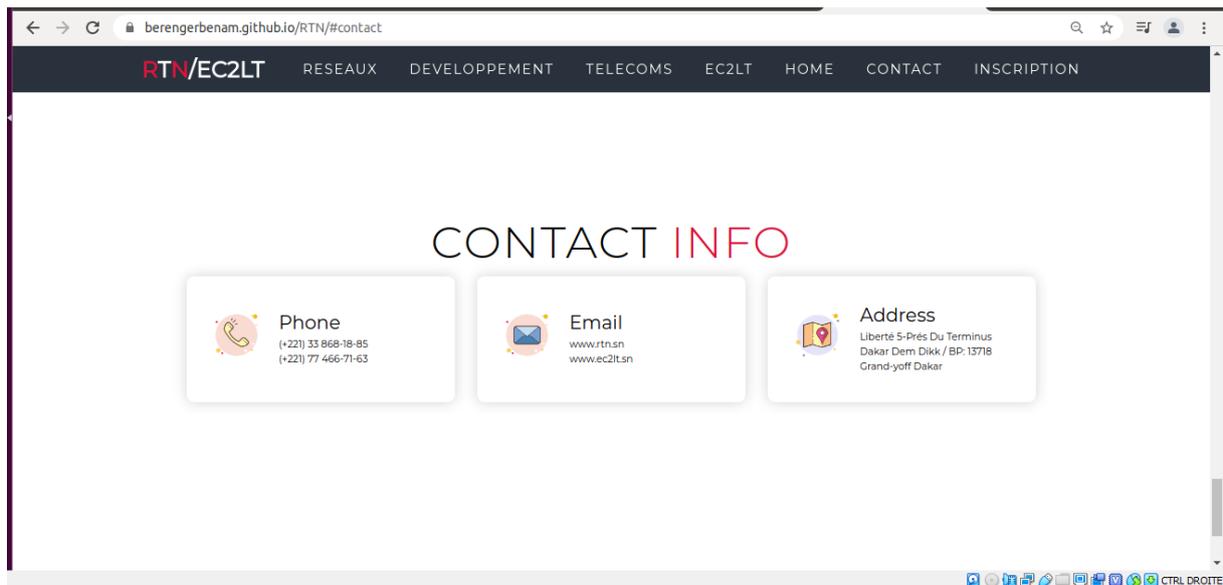
des Interfaces de Programmation d'Applications pour navigateurs (API) — API intégrées aux navigateurs web permettant de créer dynamiquement du HTML, de définir des styles de CSS, de collecter et manipuler un flux vidéo depuis la webcam de

CTRL DROITE

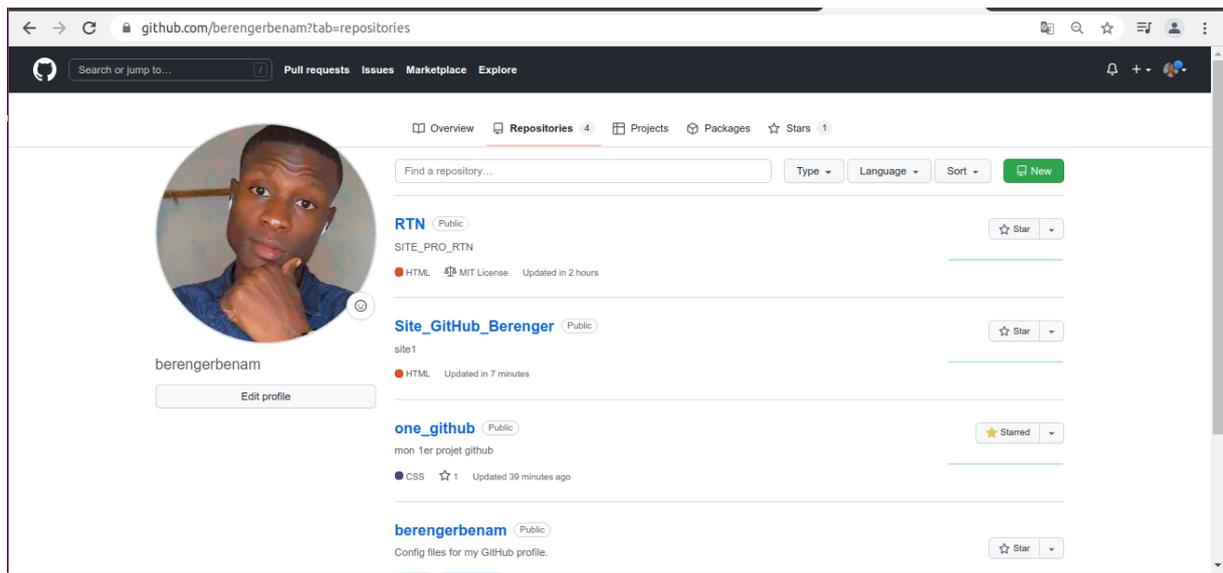
## On clique sur EC2LT



## Page contact



## Voici mes repository



## Conclusion

Git et GitHub offrent des moyens rapides et pratiques de suivre les projets, que le projet soit réalisé par un individu ou une équipe de développeurs de logiciels. Bien que GitHub dispose de nombreuses fonctionnalités complexes, il est facilement accessible pour les projets individuels et de petite taille qui nécessitent une sorte de mécanisme de suivi. En plus du contrôle de version, GitHub fournit aux utilisateurs une plate-forme sociale pour la gestion de projet ainsi que la possibilité pour les utilisateurs de créer des Gists et de stocker GeoJson.