Git vs Github …

- Git est fondamentalement un système de gestion «distribuée» de dépôts
  - Un dépôt peut contenir n'importe quoi (*pas seulement du texte ou du code*)…. Plus l'historique des fichiers (*textes*)
  - Les dépôts peuvent être «locaux» (sur votre ordinateur)
  - … Ou installés dans un «cloud» quelconque
  - Git sait faire communiquer entre eux les dépôts locaux ou distants
- Github est une application Web qui offre
  - Un cloud pour déposer des dépôts
  - Un protocole de communication entre les dépôts locaux/distants
  - Plusieurs services Web
    - Des tableaux de bord pour gérer le travail collaboratif
    - Des wiki
    - Des outils de statistiques, d'automatisation, de sécurisation, …
- Les dépots Github sont organisés en «Organisations»
  - «source-anumby»

https://github.com/anumby-source

Chez nous (anumby)

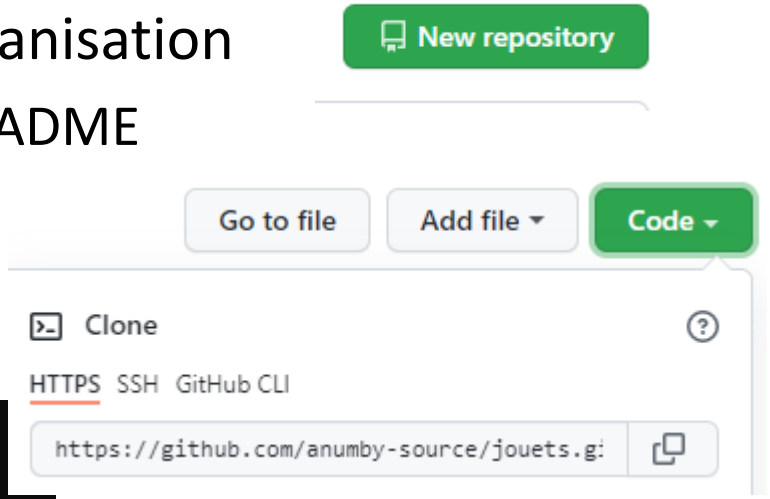
- Nos dépôts actuels sur Github:

Framboise-pico	Voiture2roues-ajax
Jouets	Alarme
Auvent	Voiture-4-roues
Alarme-xiaomi-xavier	Voiture2roues

- Evidemment tout le monde est invité à ajouter des dépôts pour tous les projets
  - Et … on pourrait créer un wiki pour chaque projet … par exemple: https://github.com/anumby-source/jouets/wiki
    - Il y a un petit éditeur wysiwyg
  - Pour être autorisé, il suffit de se créer un compte Github… et on enregistre le compte dans l'organisation.

Créer un dépôt à partir de Github

- ajouter un nouveau dépôt dans l'organisation
  - On ajoute uniquement un fichier README
- construire la référence sur ce dépôt
- clôner ce dépôt localement




```
D:\workspace>git clone https://github.com/anumby-source/jouets.git
Cloning into 'jouets'...
remote: Enumerating objects: 116, done
```

- éditer les documents
- git push
-

```
D:\workspace>cd jouets

D:\workspace\jouets>git remote -v
origin  https://github.com/anumby-source/jouets.git (fetch)
origin  https://github.com/anumby-source/jouets.git (push)
```

Git in a Nutshell

The Problem

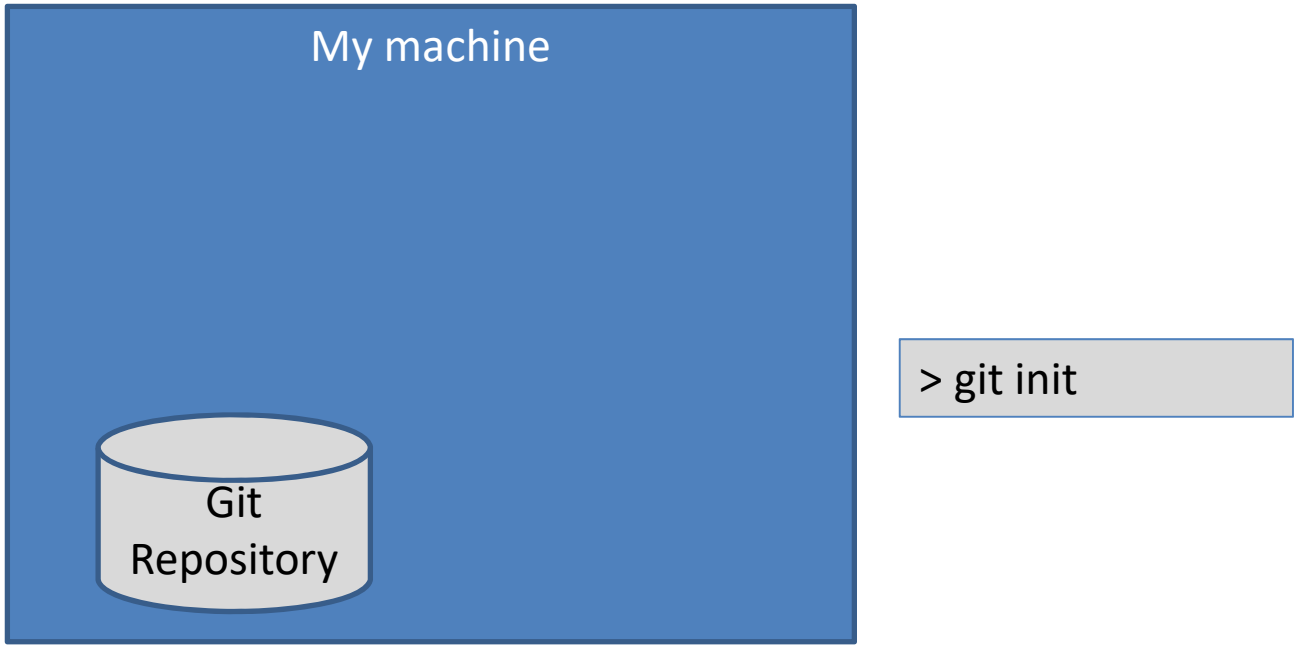
- Concurrent edition of the same file is hazardous
- Need to keep several versions of a code for several reasons:
  - Product versions and their patches.
  - Maintain customization of a product for several clients.
  - Allow the freedom to experiment or make Proof of concept without fear of losing control over code or breaking things
- Keeping multiple copies of the same files is unmanageable
  - Quickly untrackable: remembering what is what becomes a nightmare (or full time job!)
  - Renaming files prevent them from being used (file no longer found by application or compiler)

The Solution : VCS

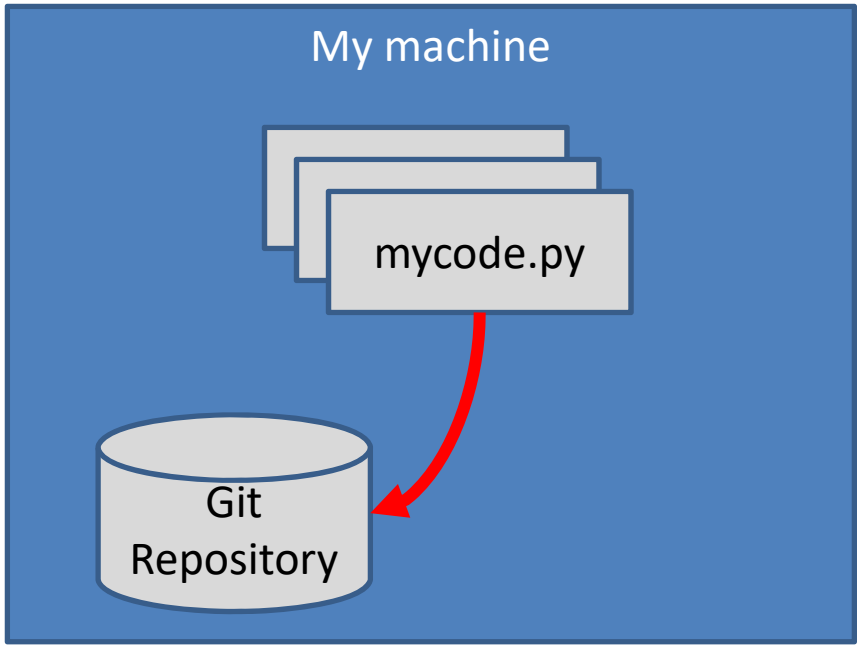
- (Distributed) Version Control System provides the following benefits:
  - Short-Term / Long-Term Undo
  - Backup & Restore
  - Collaborative development
  - Track Changes
  - Track Owner
  - Multiple version support (branches)

Git is one of the most popular DVCS today

Creating a GIT repository

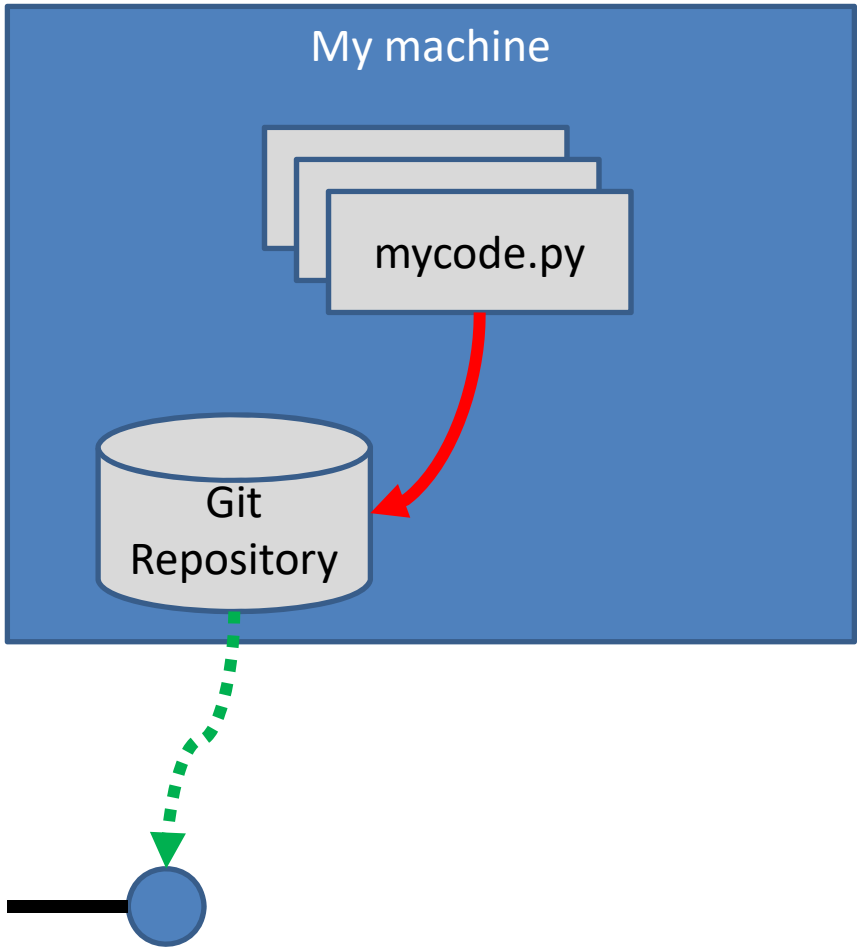


Creating some code, and installing it into the GIT repository



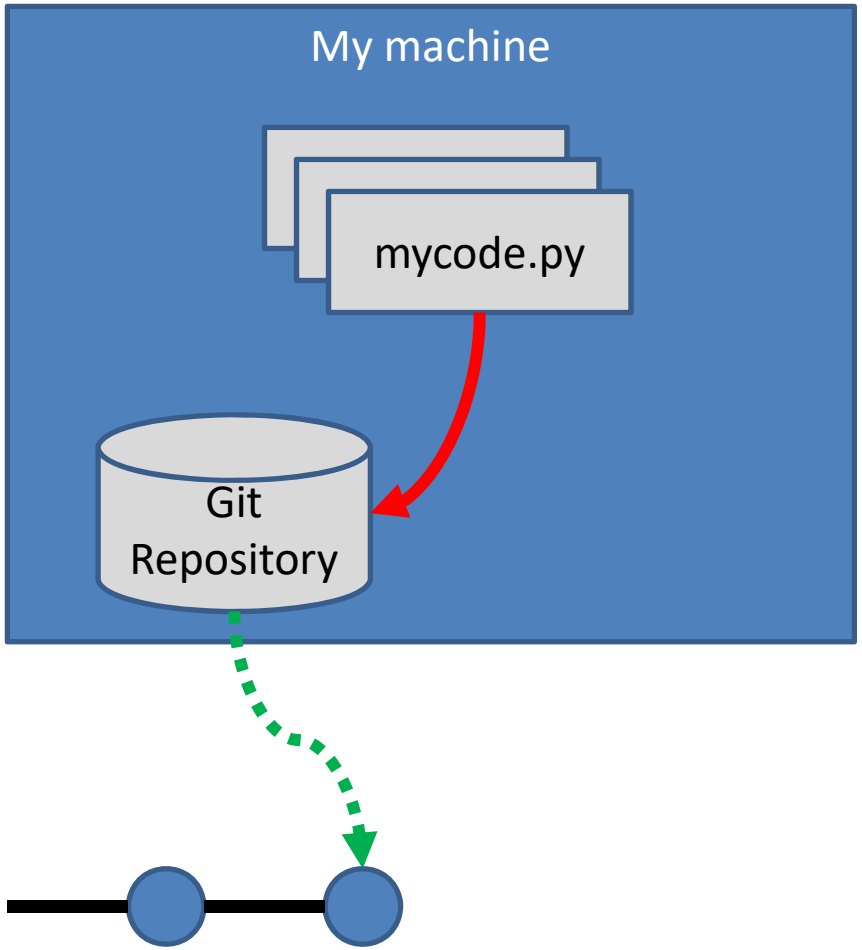

```
> git add .
```

Install (commit) my changes into the GIT repository



> git commit –m ’some change’

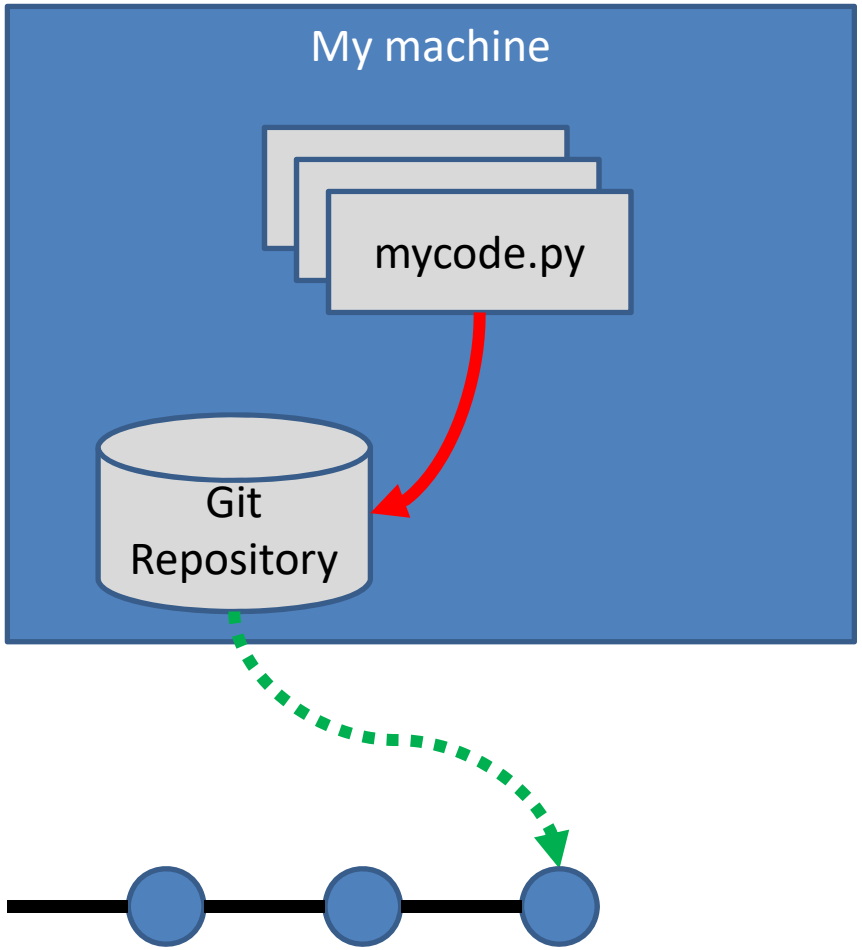
Still working, then commit the new changes…



> git commit –m ’some change’

> git add .
> git commit –m ’new change’

another change…

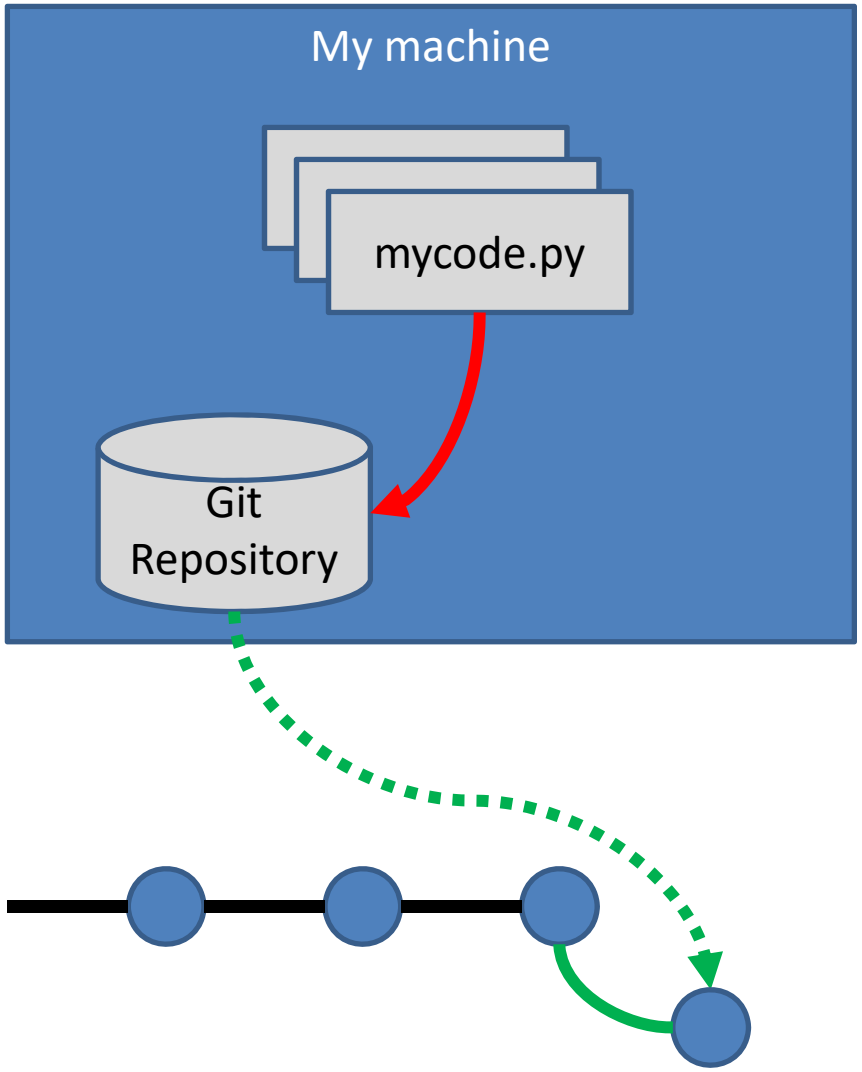


> git commit –m ’some change’

> git add .
> git commit –m’new change’

> git add .
> git commit –m ’another change’

Starting developping a new feature without disturbing the main thread (branch)



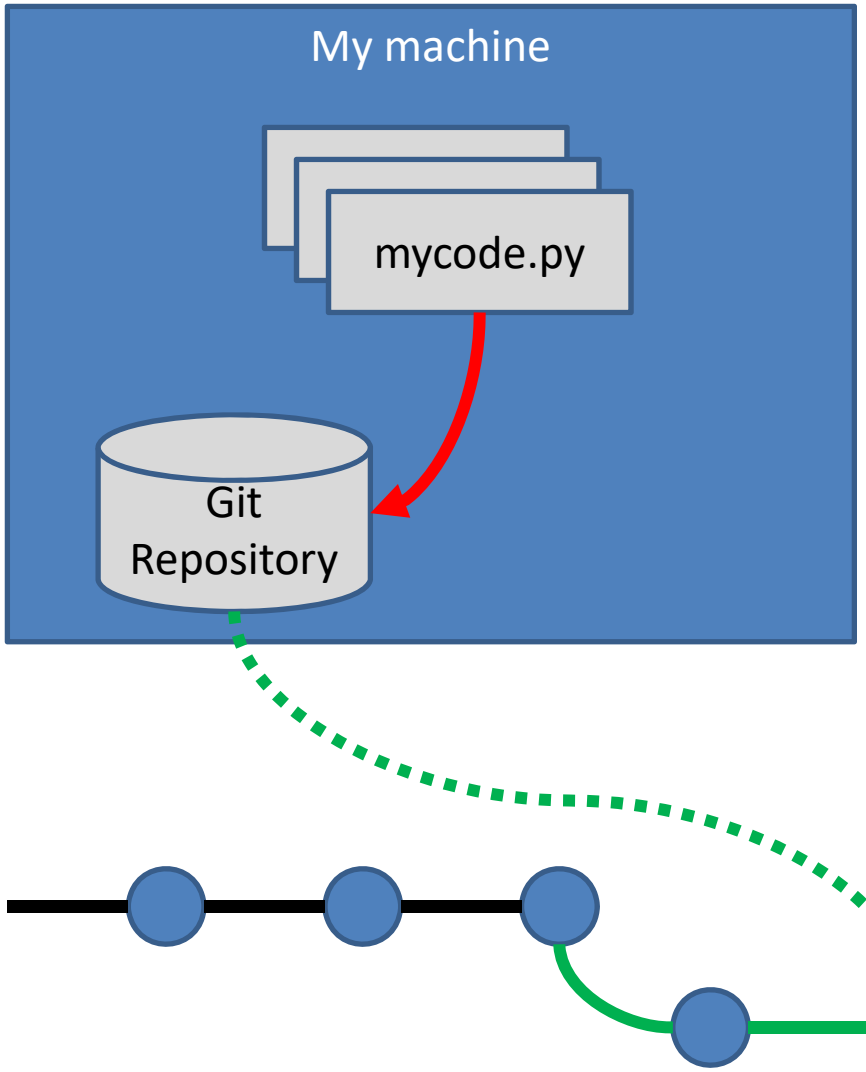
> git commit –m ’some change’

> git add .
> git commit –m’new change’

> git add .
> git commit –m ’another change’

> git checkout –b ‘feature’

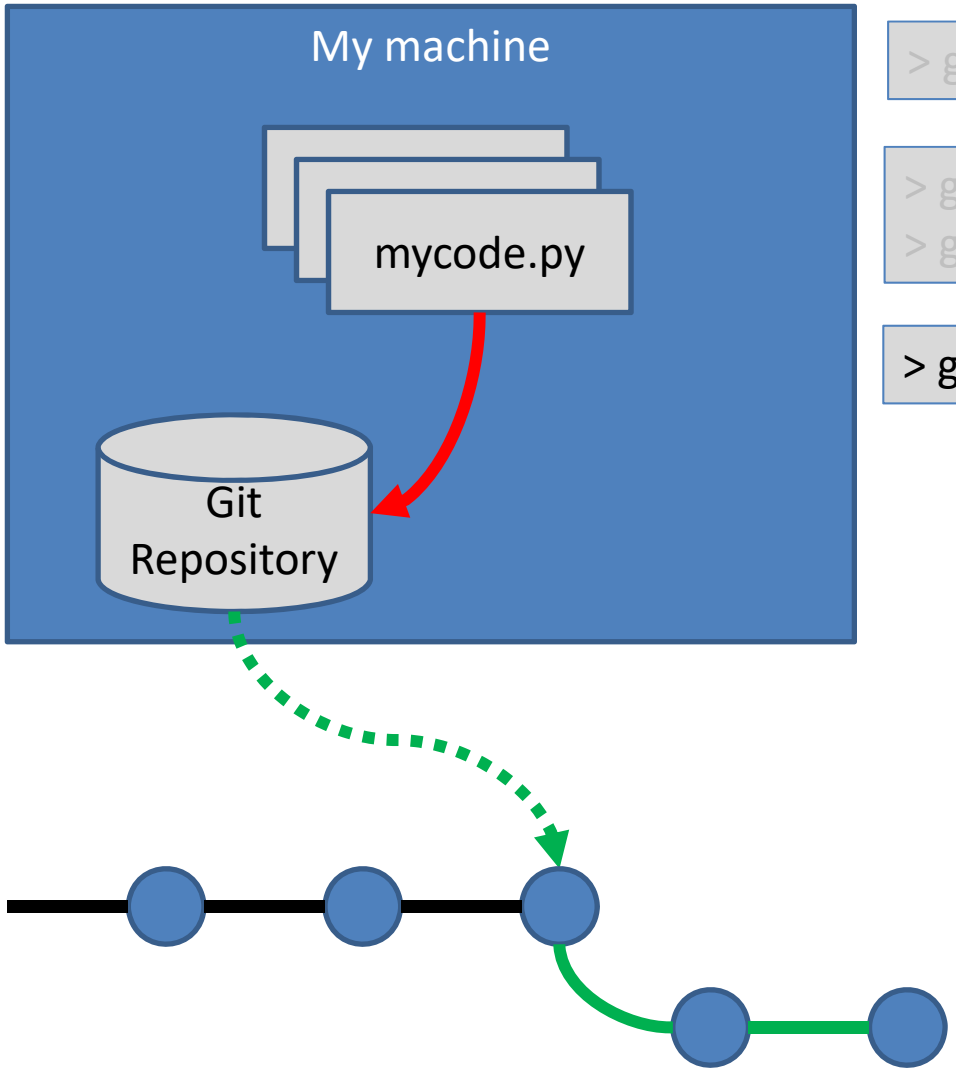
We can commit developments for the feature



> git checkout –b ‘feature’

> git add .
> git commit –m ’feature change’

Let’s go back to main thread

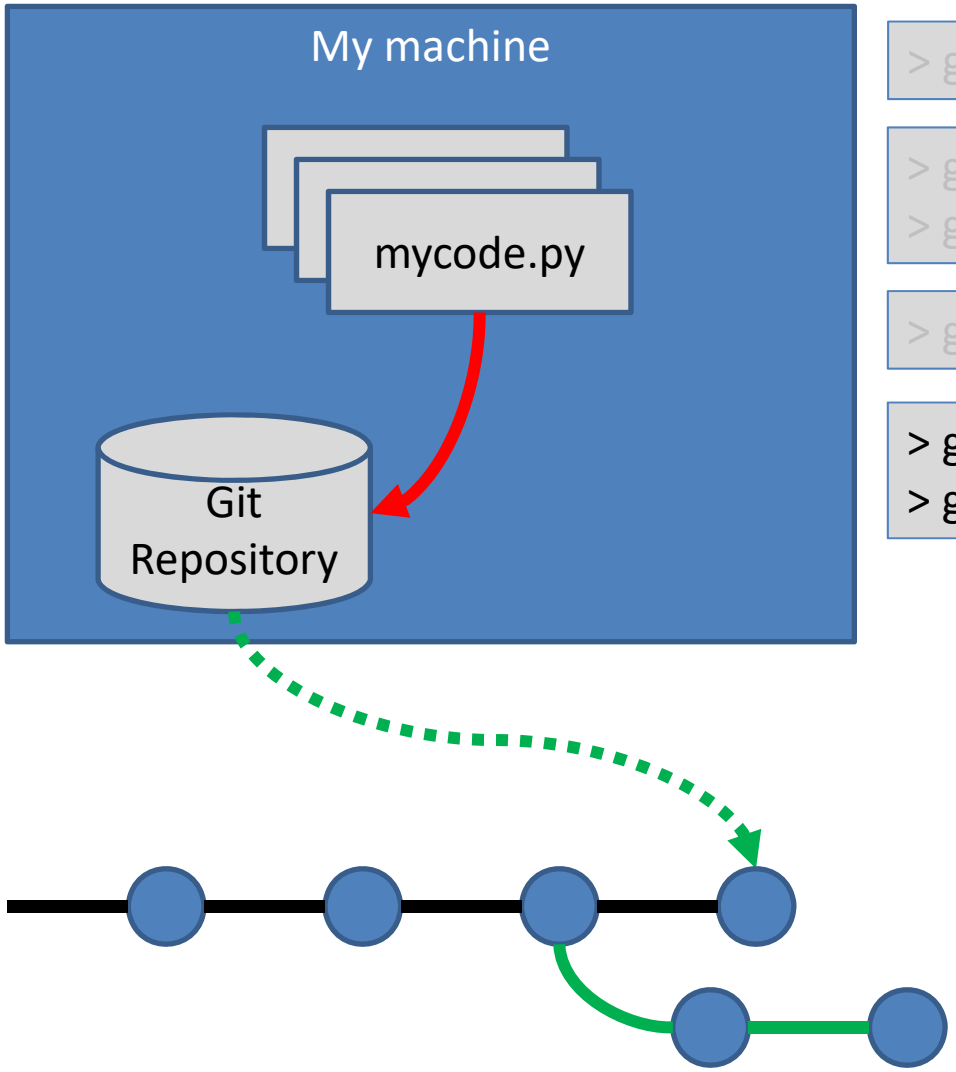


> git checkout –b ‘feature’

> git add .
> git commit –m ’feature change’

> git checkout master

Making changes to the main development



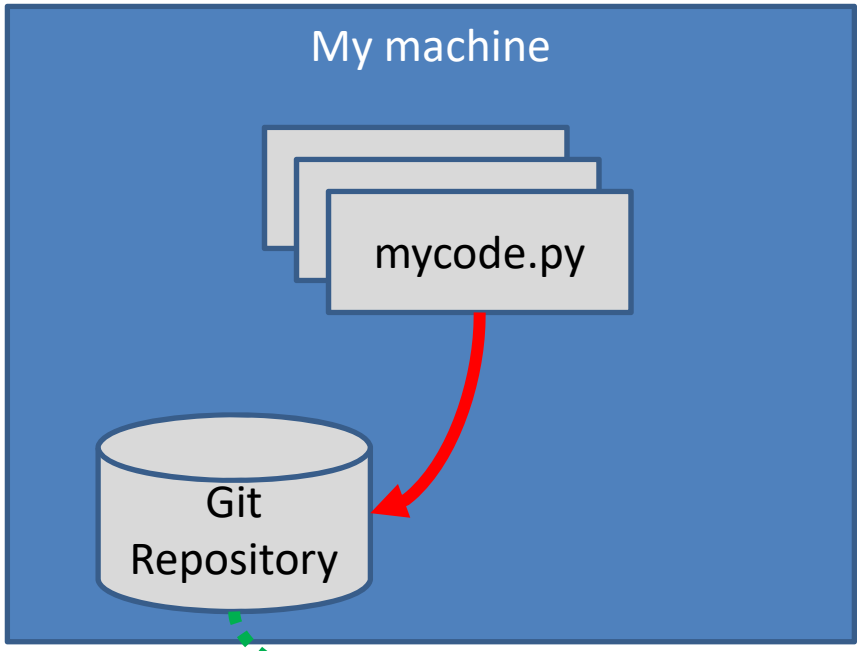
> git checkout –b ‘feature’

> git add .
> git commit –m ’feature change’

> git checkout master

> git add .
> git commit –m ’change again’

Merge the feature to the main development



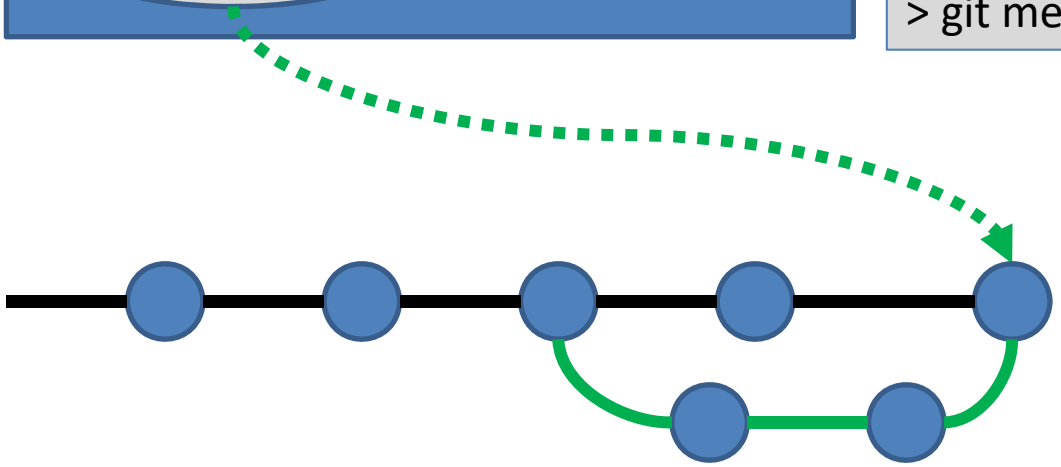

```
> git checkout –b ‘feature’
```

```
> git add .
> git commit –m ’feature change’
```

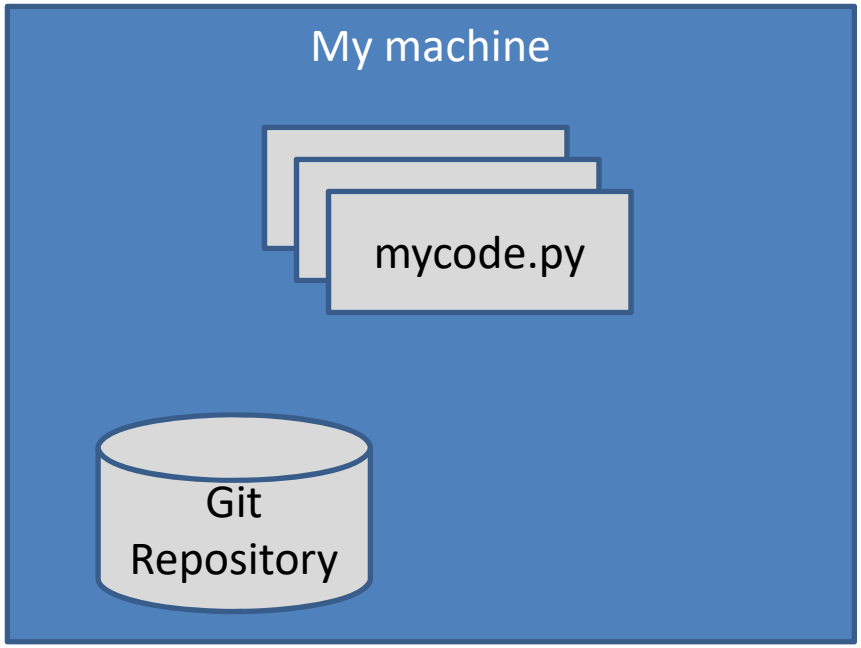
```
> git checkout master
```

```
> git add .
> git commit –m ’change again’
```

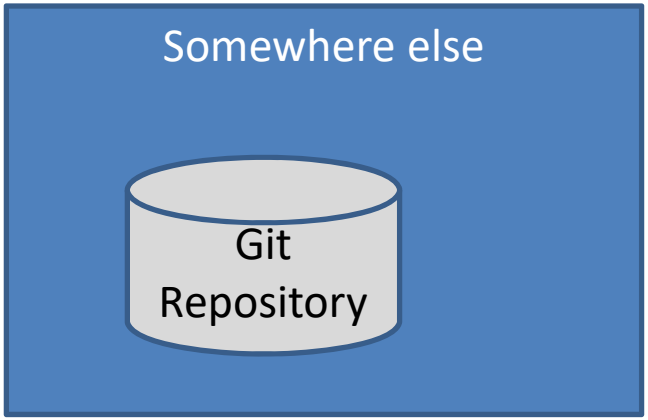
```
> git merge feature
```



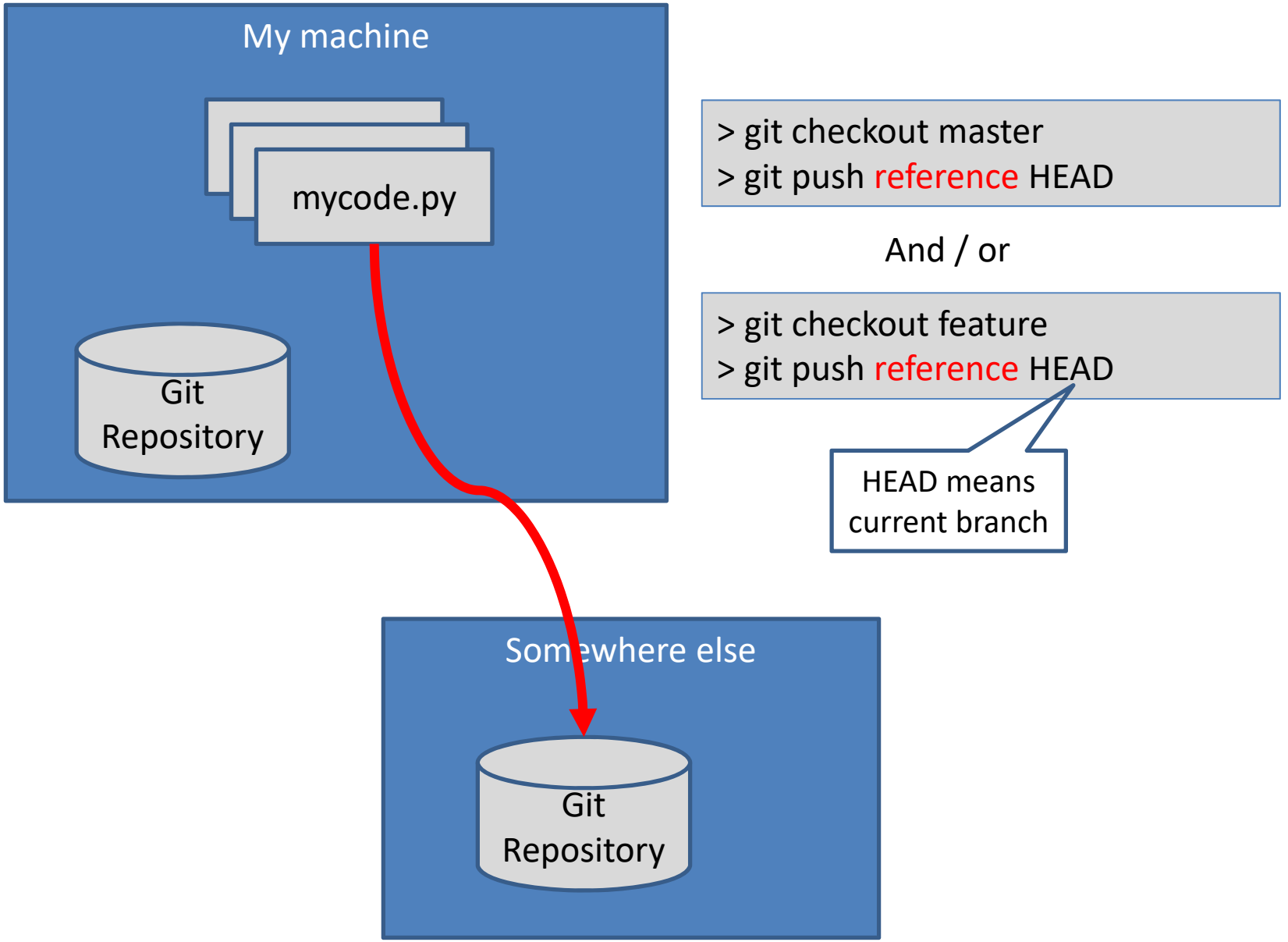
Declare some GIT repository to make it a reference repository



> git remote add reference http:…



Whatever is available in the our current branch will be pushed to *reference*



Updating from Reference Repository

It is sometimes necessary to import in the local repository (on 'My machine') changes pushed by others into the reference repository. The commands below illustrate what needs to be done, assuming that you want to import changes from branch 'master' of remote 'reference'.

```
# Save your uncommitted changes, if any
git stash

# Import all remote changes into your local repository (no local change)
git fetch --all

# Update your current local branch with master branch from reference
git rebase reference/master

# Restore your saved changes (except if none were found by initial 'git stash'
git stash pop
```

To Learn The Details…

- LAL Introduction to Git
  - http://ens.lal.in2p3.fr/NPAC/git/git_introduction.html
- The Definitive Git Book
  - http://git-scm.com/book/en/v2