Data management on HPC platforms

Transferring data and handling code with Git

scitas.epfl.ch

February 15, 2022

Slides: https://c4science.ch/u/git
Exercices: https://c4science.ch/u/gitexo
What kind of data

Categorizing data to define a strategy

- Based on size?
- Based on format?
- Based on purpose?
What kind of data

Categorizing data to define a strategy

- Based on size? *a few kilo, multiple mega, a tera, ...*
- Based on format?
- Based on purpose?
What kind of data

Categorizing data to define a strategy
- Based on size? a few kilo, multiple mega, a tera, ...
- Based on format? binary/ascii, etc...
- Based on purpose?
What kind of data

Categorizing data to define a strategy

- Based on size? a few kilo, multiple mega, a tera, ... 
- Based on format? binary/ascii, etc... 
- Based on purpose? code/input data/output data, etc...
Different types of data considered

- Simulation input
- Simulation output
- Processed results
- Simulations code, pre/post processing scripts, …
Show me your data, I tell you what you do

Different types of data considered
- Simulation input
- Simulation output
- Processed results
- Simulations code, pre/post processing scripts, ...

Type of tools
- “Big data” type of tools
- “Versioning” type of tools
Clusters folder structure

- /home: User configurations, codes, input files, scripts
- /scratch: Output files from running jobs
- /work: Long term output files storage
- /tmp: Node local space if needed
“Big data”

### SSH based file transfer

- **SSH**: Secure SHell
- **Different ways:**
  - `scp`: secure copy
  - `sftp`: secure file transfer
  - `rsync`: remote synchronization
  - `sshfs`: ssh file system

This data should be on your `/scratch` or `/work`
Move your “Big data” with \texttt{rsync}

\texttt{rsync} Synchronizes two folders, folders could be remote
Could/should be used instead of \texttt{mv}
GUIs exist for all OSes

\url{https://en.wikipedia.org/wiki/Rsync}

\begin{itemize}
  \item \texttt{rsync -auvP <src folder> <dst folder>}
  \item \texttt{rsync -auvP <remote>::<src folder> <dst folder>}
  \item \texttt{rsync -auvP <src folder> <remote>::<dst folder>}
\end{itemize}
Exercise 1: Using rsync

Questions:

- Create a temporary `tmp/` folder in your home folder on the cluster
- Copy the folder `/work/scitas-share/cmake-3/` to this `tmp/` folder
- Create a `backup/` folder in the `tmp/` folder
- Use rsync to copy the `cmake-3` folder in your `tmp/backup` folder
- Modify a file in the code of cmake in `tmp/
- Re-synchronize the files
What is “versioning”

<table>
<thead>
<tr>
<th>What version control means</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source code</strong> change and contribution management</td>
</tr>
<tr>
<td>Keep track of the changes (different versions in time)</td>
</tr>
<tr>
<td>Integrate changes from multiple sources (places or people)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local version control (e.g. RCS)</td>
</tr>
<tr>
<td>Remote on a central server (e.g. CVS, SVN)</td>
</tr>
<tr>
<td>Distributed version control (e.g. Git, Mercurial, Bazaar)</td>
</tr>
</tbody>
</table>
“Versioning”: with Git

**Git:** *the stupid content tracker*

- Distributed revision control
- Originally developed by Linus Torvald
- Named after the *egotistical bastard* Linus
$ git clone <uri repo.git>
git clone

$ git clone <uri repo.git>
Cloning into '<repo>'...
remote: Counting objects: 6940, done.
remote: Total 6940 (delta 0), reused ...
Receiving objects: 100% (6940/6940), ...
Resolving deltas: 100% (3286/3286), done.
$ git clone <uri repo.git>
Cloning into '<repo>'...
remote: Counting objects: 6940, done.
remote: Total 6940 (delta 0), reused ...
Receiving objects: 100% (6940/6940), ...
Resolving deltas: 100% (3286/3286), done.
git status is your friend

$ git status
git status is your friend

$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
Lets add a file: staging/commit
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

  my_code.py

nothing added to commit but untracked files present
Let's add a file: staging/commit

```
REMOTE SERVER

ADD

LOCAL SERVER
.git directory
STAGING AREA

WORKING DIRECTORY

$ git add <filename>
```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  new file: my_code.py
Let's add a file: staging/commit

```
$ git commit -m <message>
```
$ git status

On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
Synchronizing with the remote server

$ git clone <uri>
Synchronizing with the remote server

```
$ git push
```

REMOTE SERVER

LOCAL SERVER
.git directory
STAGING AREA
WORKING DIRECTORY

PUSH
Synchronizing with the remote server

```
$ git pull
```
Exercise 2: First step with Git

Questions:

- If you do not have git installed, get it from https://git-scm.com/downloads or from your package manager.
- Go on https://c4science.ch/ and login with your EPFL account (Login for Swiss Universities).
- Once connected go on the setting page (the wrench on the top right corner).
- In the Authentication > VCS Password menu set a password. This password will be used to connect to the git server through https.
Exercise 3: First step with Git

Questions:
- Now you should be able to clone a repository
  Either create a repository or clone
  https://c4science.ch/source/scitas-test-repo.git
- Create a file, use a filename that will not clash with the others
- Check the state of your working copy
- Add the file to the repository
- Commit your modifications
- Clone the same repository in a different folder
- Pull the potential modifications from the server
- Push your changes to the server
Collaborative work with potential problems

REMOTE SERVER

LOCAL SERVER

.GIT DIRECTORY
STAGING AREA

WORKING DIRECTORY
Collaborative work with potential problems

REMOTE SERVER

LOCAL SERVER
.stgit directory
STAGING AREA

WORKING DIRECTORY

ADD
COMMIT
PUSH

$ git add <filename>
$ git commit -m <message>
$ git push
Collaborative work with potential problems

$ git add <filename>
$ git commit -m <message>
$ git push
$ git push
To <repo>
  ! [rejected] master -> master (fetch first)
error: failed to push some refs to '<repo>'
hint: ...

Collaborative work with potential problems
Collaborative work with potential problems

$ git pull
Collaborative work with potential problems

$ git pull
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From <repo>
  fe22d81..0bcfb99 master    -> origin/master
Auto-merging my_code.py
CONFLICT (content): Merge conflict in my_code.py
Automatic merge failed; fix conflicts and then commit the result.
Collaborative work with potential problems

$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)

Unmerged paths:
(use "git add <file>..." to mark resolution)

both modified: my_code.py
Collaborative work with potential problems

Correct the conflict:

<<<<<<<<<<<
One version
=========
Other version
>>>>>>>>>>>
Collaborative work with potential problems

REMOTE SERVER

COMMIT
PUSH

LOCAL SERVER
.git directory
STAGING AREA

WORKING DIRECTORY

$ git commit -a
$ git push
Exercise 4: Generate and solve conflicts

Questions:

- Modify the file created in the previous exercise in both clones
- Commit this both modifications
- Pull and push in one of the clone
- Pull in the second clone, You should get a conflict

<<<<<<<
One version
=======
Other version
>>>>>>>

- Check the local status
- Correct the conflict and commit using `git commit -a`
- Push the modifications
Introduction to branches

$ git clone <uri repo.git>
Introduction to branches

```
$ git checkout -b feature
```
Introduction to branches

```
$ git commit -m <message>
```
Introduction to branches

$ git commit -m <message>
Introduction to branches

`$ git commit -m <message>`
Introduction to branches

$ git checkout master
Introduction to branches

$ git commit -m <message>
Introduction to branches

$ git commit -m <message>
Introduction to branches

$ git commit -m <message>
Introduction to branches

$ git merge feature
Introduction to branches

$ git commit -m <message>
Exercise 5: Branches/merges

Questions:
- Create a branch with the name of your choice
- Modify a file and commit the changes
- Checkout the master branch
- Modify a file and commit the changes
- Merge the branch previously created in the master branch
- List all branches
- Print the logs of the different modifications
- Delete the merged branch
Workflow: feature branch
Workflow: gitflow
Workflow: gitflow
Workflow: gitflow
Workflow: gitflow

- Master
- Hotfix
- Release
- Develop
- Feature

Versioning:
- v0.1
- v0.2
- v1.0
Multiple servers for one project

Remote server 1

File versions DB
- Version 3
- Version 2
- Version 1
Multiple servers for one project

Remote server 1
File versions DB
  Version 3
  Version 2
  Version 1

Computer
File
File versions DB
  Version 3
  Version 2
  Version 1

Command
On Computer:

```bash
git clone <remote url 1>
```
Multiple servers for one project

Remote server 1
- File versions DB
  - Version 3
  - Version 2
  - Version 1

Remote server 2
- File versions DB

Computer
- File
- File versions DB
  - Version 3
  - Version 2
  - Version 1

Command
On Remote server 2:
```
git init --bare
```
Multiple servers for one project

- Remote server 1
  - File versions DB
    - Version 3
    - Version 2
    - Version 1

- Remote server 2
  - File versions DB

- Computer
  - File
    - File versions DB
      - Version 3
      - Version 2
      - Version 1

Command:

On Computer:

```
  git remote add server2 \<remote url 2>
```
Multiple servers for one project

Remote server 1
File versions DB
Version 3
Version 2
Version 1

Remote server 2
File versions DB
Version 3
Version 2
Version 1

Computer
File
File versions DB
Version 3
Version 2
Version 1

Command
On Computer:
git push server2
Exercise 6: Handle remotes

Questions:

- Connect on the front node of your favorite cluster
- Create a new folder that will contain your server
- In this folder initialize a new git server
- In one of the former clone of scitas-test add the new remote URL
  
  \texttt{ssh://<username>:@<cluster name>/<path to repo>}

- List the remotes to see if everything looks correct
- Push the local content to the new server
- On the cluster clone this new server URL \texttt{<path to repo>}

**Note:** The access permission on this new server are based on the file system permissions
Sources

- Wikipedia
- http://git-scm.com
- Manpages: rsync, git
- https://www.atlassian.com/git/
- http://nvie.com/posts/
  a-successful-git-branching-model/

Learn more

- Research Data management
  http://library.epfl.ch/research-data/
- Git with a game: http://learngitbranching.js.org/