Git-borg linker

Incremental backup of your results folder with git and borg

Nicolas Fontrodona

June 30, 2022

Table of contents

Data versioning

In a bioinformatics project:

- Code
- Results (produced by this code)

Saving the results allow to:

- Compare them between code versions
- Have an overview of result files produced by the project code

Data versioning tools can help us to achieve this.

Controlling the version of your data with a tool can help you to

- Save you time in managing and tracking your data versions
- Collaborate with your team members
- Avoid losing data
- Increase the traceability of your results

Data versioning tools

Many data versioning tools are available:

data version control

lakefs

git lfs

... and many others

Data versioning tools

Main advantages

- git integration
- easy to use and install
- Reflink usage, pipeline tools, etc...

Main drawbacks

- No data deduplication
- You can't delete only some versions of a file without deleting them all !

Data versioning tools

Main advantages

- Handle data in the same way you handle code with git
- Recovery from data errors
- Efficient for large data lake

Main drawbacks

- No easy to use (database, UI that you have to manage)
- You can only delete objects on S3
- No git integration

Data versioning tools

Main advantages

- git integration / Same git workflow (no additional commands)
- Store files in a repository dedicated for large files (couples of GBs)

Main drawbacks

- Can only use https:// or file:// endpoints
- You can prune old files, but you can't keep some old version and remove others

Data versioning tools to backup program

When using a data versioning tool, it seems that we can't control precisely specific versions to keep and remove to save some space.

That's why, we started to be interested in backup programs for data versioning.

Backup program: BorgBackup (short Borg)

Borg: Efficient and secure backups. `https://www.borgbackup.org/`

Main advantages

- Easy to use
- Supports deduplication
- Supports compression
- Creates an archive folder (we can handle it like we want)
- No restrictions in backups deletion

Main drawback

- No git integration to link the code with the results

Git Borg Linker

Git Borg Linker (gblk)

Developed to integrate borg with git.

gblk handles data versioning by using git history and uses borg to do the backups. gblk: Borg advantages + git integration

Git Borg Linker

Available at:

https://gitbio.ens-lyon.fr/LBMC/hub/git_borg_linker

Note that git, borg and rust is required to use gblk.

To install rust and gblk, you can use the following commands:

```
$ # Install git
$ apt install git
$ # Install rust on debian & ubuntu
$ apt install borgbackup
$ # install rust
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
$ # install gblk
$ cargo install --git https://gitbio.ens-lyon.fr/LBMC/hub/
    git_borg_linker
$ # Update your bashrc
$ echo "export PATH=$PATH:/home/nicolas/.cargo/bin" > ~/.bashrc
$ source ~/.bashrc
```

Git Borg Linker

To use gblk in a project folder, it must have this minimal structure:

```
project
|- results
|- .git
```

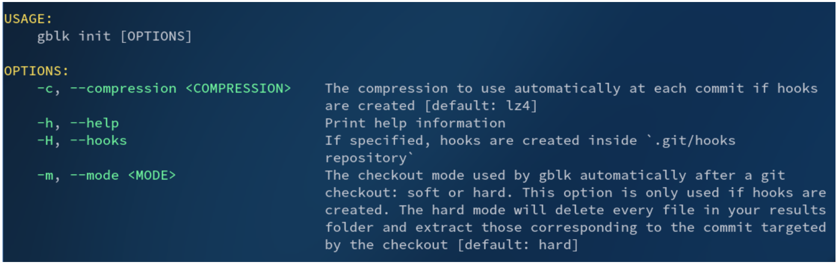
The `results` folder is mandatory to use gblk. This is this folder that will be tracked by gblk.

Git Borg Linker - Example

Let's see how gblk works with an example

```
$ mkdir test_gblk; cd test_gblk
$ mkdir results; src
$ git init
```

Initialize gblk to start using it: Done with `gblk init`:



```
USAGE:
    gblk init [OPTIONS]

OPTIONS:
    -c, --compression <COMPRESSION>    The compression to use automatically at each commit if hooks
                                       are created [default: lz4]
    -h, --help                         Print help information
    -H, --hooks                        If specified, hooks are created inside `.git/hooks
                                       repository`
    -m, --mode <MODE>                  The checkout mode used by gblk automatically after a git
                                       checkout: soft or hard. This option is only used if hooks are
                                       created. The hard mode will delete every file in your results
                                       folder and extract those corresponding to the commit targeted
                                       by the checkout [default: hard]
```

Available compressions: no, lz4, zstd, zlib or lzma. See Borg create page

Git Borg Linker - init

gblk can optionally create git hooks (with `gblk init` command or `create-hooks`) to automatically execute commands:

Git Borg Linker - init

gblk can optionally create git hooks (with `gblk init` command or `create-hooks`) to automatically execute commands:

- after a commit to automatically save you data
- after a checkout to:
  1. Revert the checkout
  2. Check if there is unsaved/missing data in the `results` folder (compared to the archive of the current commit). If so, stops the checkout.
  3. Perform git checkout
  4. Perform a gblk checkout (restore your results folder as it was at the destination commit).

Git Borg Linker - init

gblk can optionally create git hooks (with `gblk init` command or `create-hooks`) to automatically execute commands:

- after a commit to automatically save you data
- after a checkout to:
  1. Revert the checkout
  2. Check if there is unsaved/missing data in the `results` folder (compared to the archive of the current commit). If so, stops the checkout.
  3. Perform git checkout
  4. Perform a gblk checkout (restore your results folder as it was at the destination commit).

The hooks created by those commands can be found in `.git/hooks` folder and are name `post-commit` and `post-checkout`.

Git Borg Linker - init

`gblk init --hooks` and `create-hooks` commands will also create 3 git aliases:

- `co`: for checkout. It allow to make the git checkout quiet (because it will be reverted)

Git Borg Linker - init

`gblk init --hooks` and `create-hooks` commands will also create 3 git aliases:

- `co`: for checkout. It allow to make the git checkout quiet (because it will be reverted)
- `conh`: For checkout. It allow to make a checkout without using the hooks. It is useful to bypass the check of unsaved/missing data in the `results` folder. You have to combine it with `gblk checkout --mode hard` to checkout your `results` folder to the new current commit.

Git Borg Linker - init

`gblk init --hooks` and `create-hooks` commands will also create 3 git aliases:

- `co`: for checkout. It allow to make the git checkout quiet (because it will be reverted)
- `conh`: For checkout. It allow to make a checkout without using the hooks. It is useful to bypass the check of unsaved/missing data in the `results` folder. You have to combine it with `gblk checkout --mode hard` to checkout your `results` folder to the new current commit.
- `cnh`: For commit. It allows to make a simple git commit without saving your result repository (may lead to data loss)

Note: Those aliases are valid only for the current project. They can be found in the file `.git/config`

Git Borg Linker - init

Let's init gblk with hooks:

```
$ gblk init --hooks
borg repository initialised at .borg
```

Creates an empty `.borg` repository were the backups of your results folder will be saved.

Git Borg Linker - commit

Let's add a code that will produce a result file.

```
$ echo "echo 'result line' > results/result.txt" > src/script.sh # creates a script file that produces a result file
$ bash src/script.sh # creation of a results/result.txt
$ git add src/script.sh && git commit -m "src/script.sh: initial commit" # commit the change
Repository: /~/test_gblk/.borg
Archive name: b1da0e305c906fb242bc8ef5699edeaa8c2a6d64
...
------------------------------------------------------------------------------

Original size      Compressed size    Deduplicated size
This archive:                618 B              551 B              551 B
All archives:                 12 B               15 B              738 B

Unique chunks         Total chunks
Chunk index:                     3                    3
------------------------------------------------------------------------------

[master (commit racine) b1da0e3] src/script.sh: initial commit
1 file changed, 1 insertion(+)
create mode 100644 src/script.sh
```

Git Borg Linker - commit

Because a post-commit hook was created, the command

```
git commit -m "src/script.sh:  initial commit"
```

is equivalent to:

```
$ git commit -m "src/script.sh: initial commit" # commit the change
$ gblk commit # commit the results, we have to use this command
    after commit if gblk hooks are not enabled
```

Git Borg Linker - commit

Because a post-commit hook was created, the command

```
git commit -m "src/script.sh:  initial commit"
```

is equivalent to:

```
$ git commit -m "src/script.sh: initial commit" # commit the change
$ gblk commit # commit the results, we have to use this command
    after commit if gblk hooks are not enabled
```

A backup with the name current git commit was created. We can use `gblk list` to list the backups:

```
$ gblk list
b1da0e305c906fb242bc8... Mon, 2022-06-13 15:47:24 [2960...]
$ git rev-parse --verify HEAD # Show current commit
b1da0e305c906fb242bc8...
```

Git Borg Linker - commit

Because a post-commit hook was created, the command

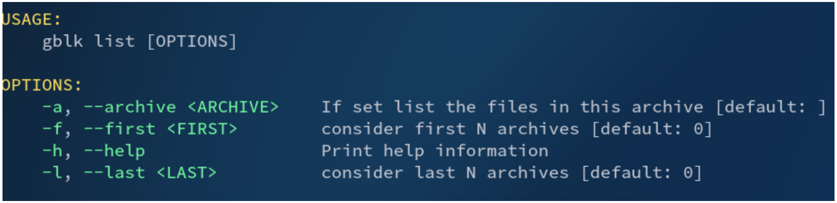
`git commit -m "src/script.sh: initial commit"`

is equivalent to:

```
$ git commit -m "src/script.sh: initial commit" # commit the change
$ gblk commit # commit the results, we have to use this command
    after commit if gblk hooks are not enabled
```

A backup with the name current git commit was created. We can use `gblk list` to list the backups:

```
$ gblk list
b1da0e305c906fb242bc8... Mon, 2022-06-13 15:47:24 [2960...]
$ git rev-parse --verify HEAD # Show current commit
b1da0e305c906fb242bc8...
```



```
USAGE:
    gblk list [OPTIONS]

OPTIONS:
    -a, --archive <ARCHIVE>    If set list the files in this archive [default: ]
    -f, --first <FIRST>        consider first N archives [default: 0]
    -h, --help                 Print help information
    -l, --last <LAST>          consider last N archives [default: 0]
```

Git Borg Linker - checkout

Let's commit the new changes and then checkout.

```
$ echo "echo 'newresult line' > results/newresult.txt" >> src/script
    .sh
$ bash src/script.sh
$ ls results
result.txt   newresult.txt
$ git add src/script.sh && git commit -m "src/script.sh: second
    commit" # commit the change
...
$ git co b1da0e305c906fb242bc8... # checkout to the first commit
$ ls results
result.txt
```

Git Borg Linker - checkout

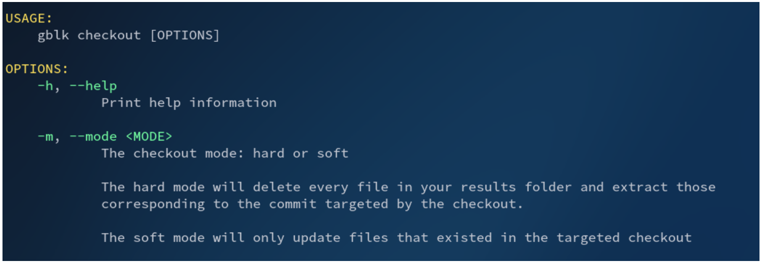
Because a post-checkout hook was created, the command `git co b1da0e305c906fb242bc8...` is equivalent to:

```
$ git chekout --quiet b1da0e305c906fb242bc8...
$ git chekout source_commit # it cancels the first commit because
    pre-checkout hook doesn't exist in git
$ gblk pre-co # checks if there is no new data inside the results
    folder otherwize it stops the checkout.
$ git checkout b1da0e305c906fb242bc8...
$ gblk checkout --mode hard # hard is the default mode used when
    hooks are created. With hard mode, the results folder is
    completly deleted, this action is skipped if with --mode soft.
    Then the backup of the first commit is extracted into the
    results folder.
```

Git Borg Linker - checkout

Because a post-checkout hook was created, the command `git co b1da0e305c906fb242bc8...` is equivalent to:

```
$ git chekout --quiet b1da0e305c906fb242bc8...
$ git chekout source_commit # it cancels the first commit because
    pre-checkout hook doesn't exist in git
$ gblk pre-co # checks if there is no new data inside the results
    folder otherwize it stops the checkout.
$ git checkout b1da0e305c906fb242bc8...
$ gblk checkout --mode hard # hard is the default mode used when
    hooks are created. With hard mode, the results folder is
    completly deleted, this action is skipped if with --mode soft.
    Then the backup of the first commit is extracted into the
    results folder.
```



```
USAGE:
    gblk checkout [OPTIONS]

OPTIONS:
    -h, --help
            Print help information

    -m, --mode <MODE>
            The checkout mode: hard or soft

            The hard mode will delete every file in your results folder and extract those
            corresponding to the commit targeted by the checkout.

            The soft mode will only update files that existed in the targeted checkout
```

Git Borg Linker - checkout

What happens if the results folder contain unsaved changes ?

```
$ git co master
$ echo "new endline" >> results/newresult.txt
$ git co b1da0e305c906fb242bc8... # checkout to the first commit
+27 B      -15 B results/newresult.txt

Your results folder contains unsaved changes!
Please update your current commit with:  gblk commit --update
Or revert it back to it's previous state with gblk commit --revert
```

To avoid losing data, gblk pre-co command will stop the checkout if new data is found in the `results` folder.

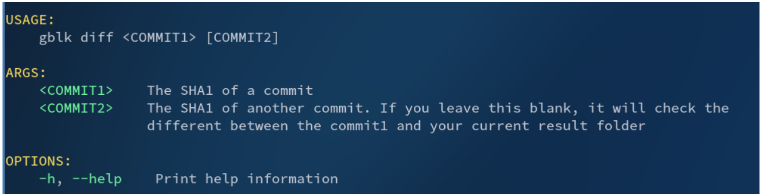
You can either:

- remove the change (`gblk commit --revert`)
- save the changes (`gblk commit --update`).

And proceed to checkout

Git Borg Linker - diff

Check for differences between backups using `gblk diff`.



```
USAGE:
    gblk diff <COMMIT1> [COMMIT2]

ARGS:
    <COMMIT1>    The SHA1 of a commit
    <COMMIT2>    The SHA1 of another commit. If you leave this blank, it will check the
                 different between the commit1 and your current result folder

OPTIONS:
    -h, --help    Print help information
```

Note that you can only use the name of backups (that corresponds to the SHA1 of a commit) saved in `.borg` folder. Branch names can't be used. Example:

```
$ gblk list
b1da0e305c9... Mon, 2022-06-13 15:47:24 [2960ccbfea14d4...]
25fdb6808cd... Mon, 2022-06-13 16:05:55 [e8fddf7eba8019...]
$ gblk diff b1da0e305c9... 25fdb6808cd...
added          15 B results/newresult.txt
$  gblk diff 25fdb6808cd...
+27 B     -15 B results/newresult.txt
```

Git Borg Linker - delete

For now, gblk does not handle backup deletion. To delete a backup, you can use borg

Use borg delete to remove specific commits or the N first or last commits

```
$ borg delete .borg::b1da0e305c906fb242bc8ef5699edeaa8c2a6d64 #
    deletion of a selected commit the disk space is not freed
$ borg compact .borg
$ gblk list
$ # deletion of the two first commits
$ borg delete .borg --first 2 && borg compact
```

You can also remove backups based on their creation date using `borg prune`

```
# Keep 7 end of day and 4 additional end of week archives.
# Do a dry-run without actually deleting anything.
$ borg prune --keep-daily=7 --keep-weekly=4 .borg && borg compact
```

You can see borg documentation to learn more about `borg delete`, `prune` and `compact`

Git Borg Linker - borgignore

To tell gblk to don't track some files in the `results` folder, a `.borgignore` file can be defined.

Example:
To avoid tracking all files in the folder `results/test` and to avoid tracking txt file in the subfolder `results/notxt` you can write the following .borgignore file

```
results/test/*
results/notxt/*.txt
```

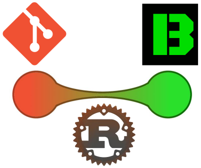
Note : When checking out with hard mode, ignored files won't be deleted.

Warning : If you put a blanck line at the end of `.borgignore` file: Results file that didn't previously exists on the destination commit won't be deleted even with `-mode hard`.

Git Borg Linker - Still under development

- Still needs a feedback of user for:
  - improvements/new features
  - Bug fix

Conclusion



Git borg linker

- Integrates git and borg so it can be used for data versioning in a bioinformatic projects.
- Is easy to use
- Handles data deduplication to reduce the cost in storage space when large files are generated

But it can be slow when the results folder is big