# Git

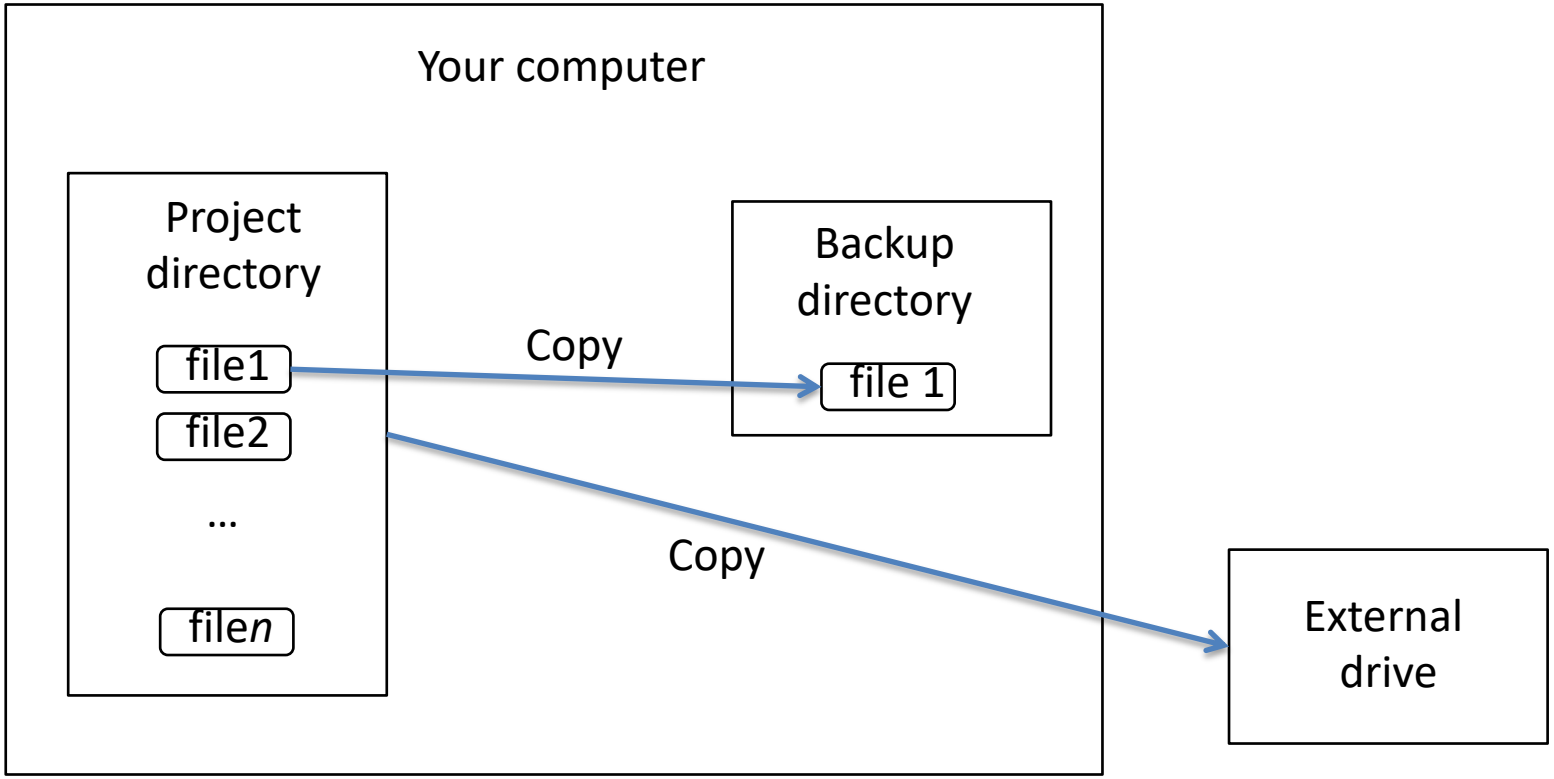
Carlos Cruz
NASA GSFC

Python bootcamp 2016

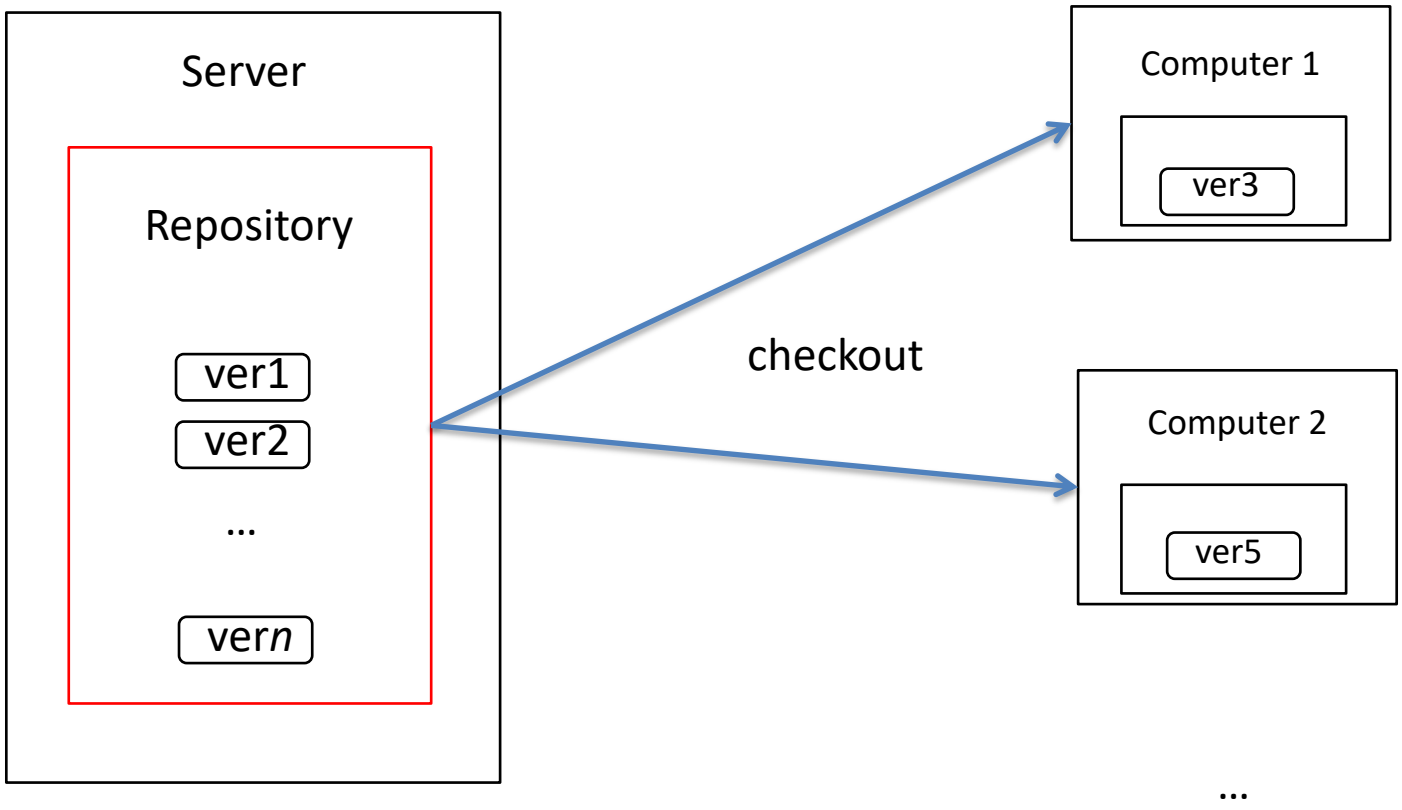
# Version control system

A version control system is a program that can record multiple versions of a source file, storing information such as the creation time of each version, who made it, and a description of what was changed.

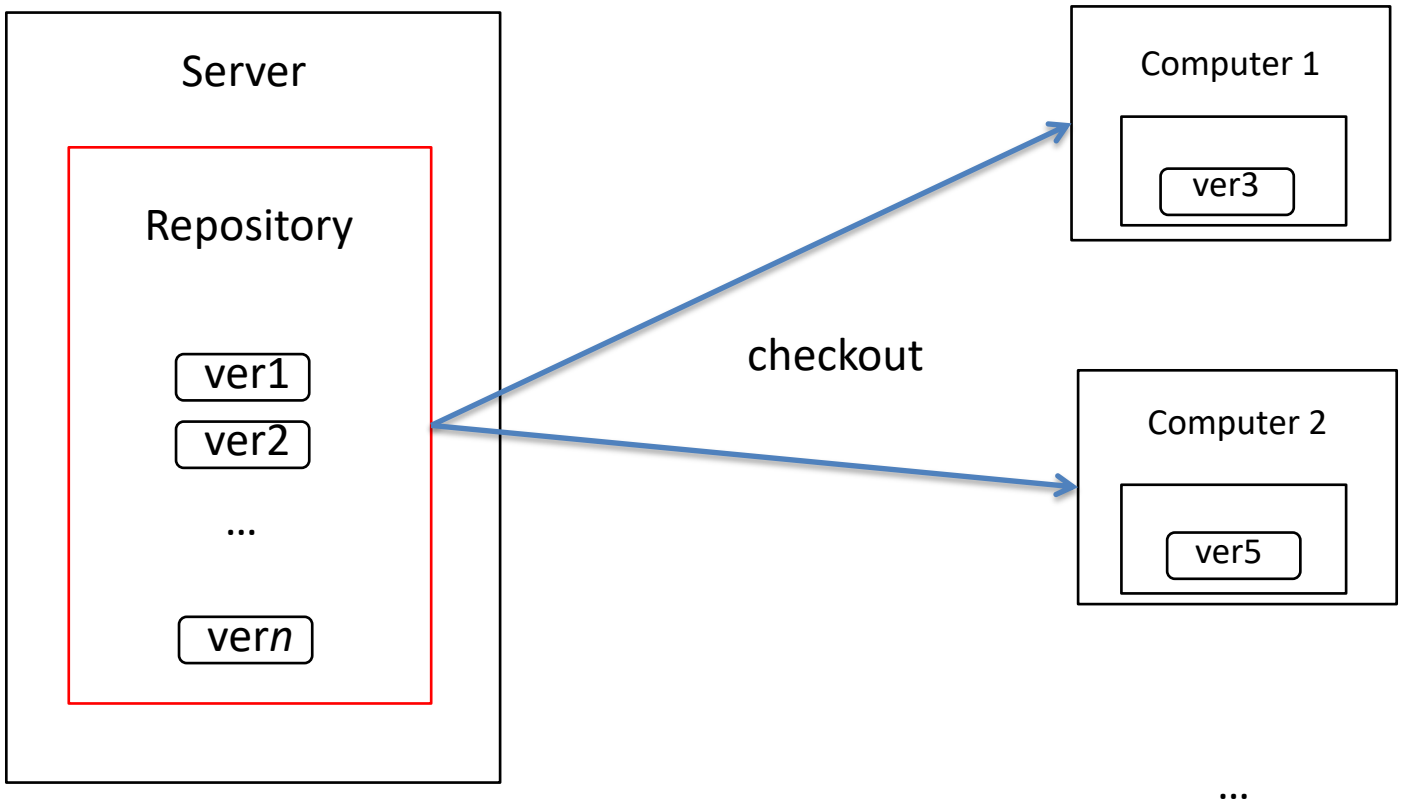
# Why version control?



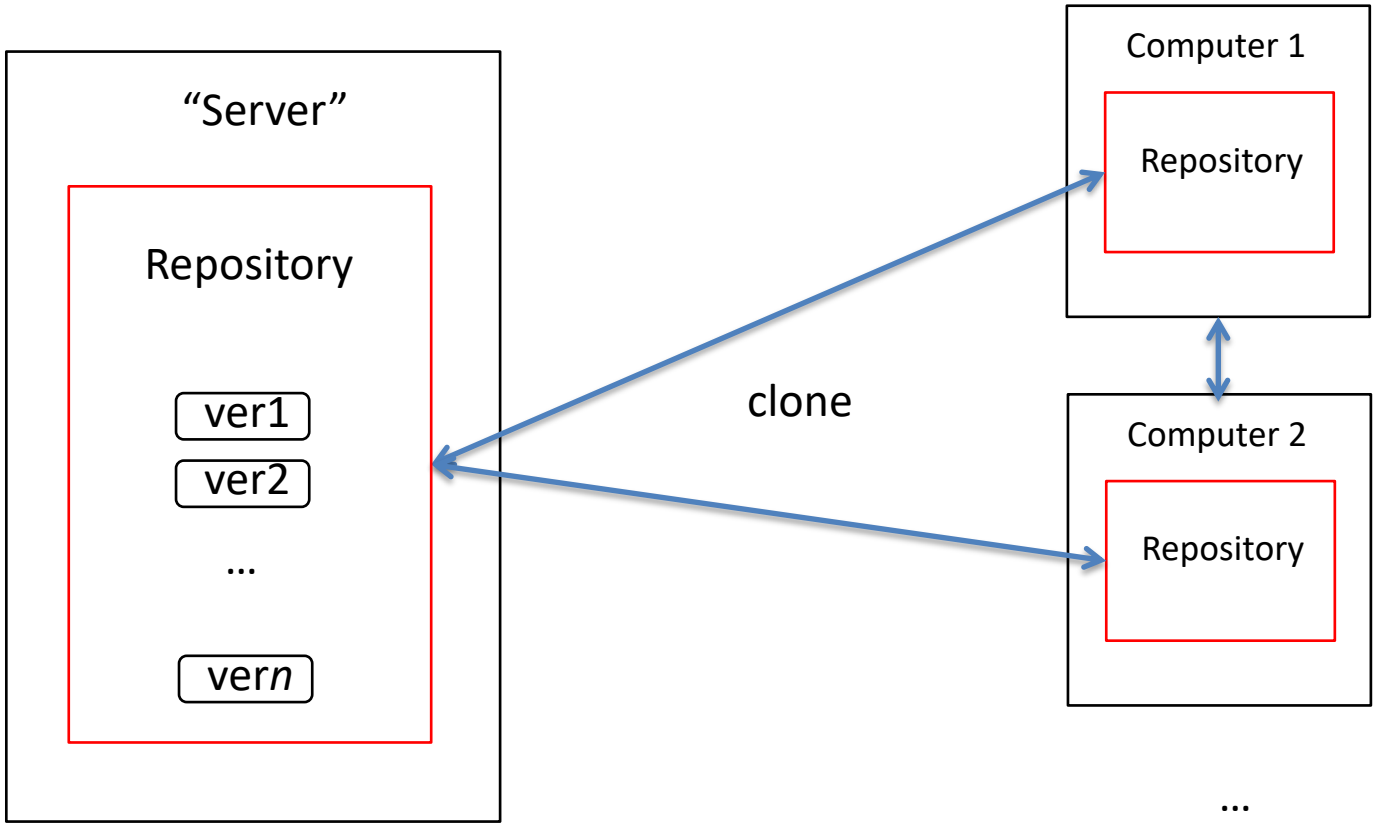
# Centralized VCS



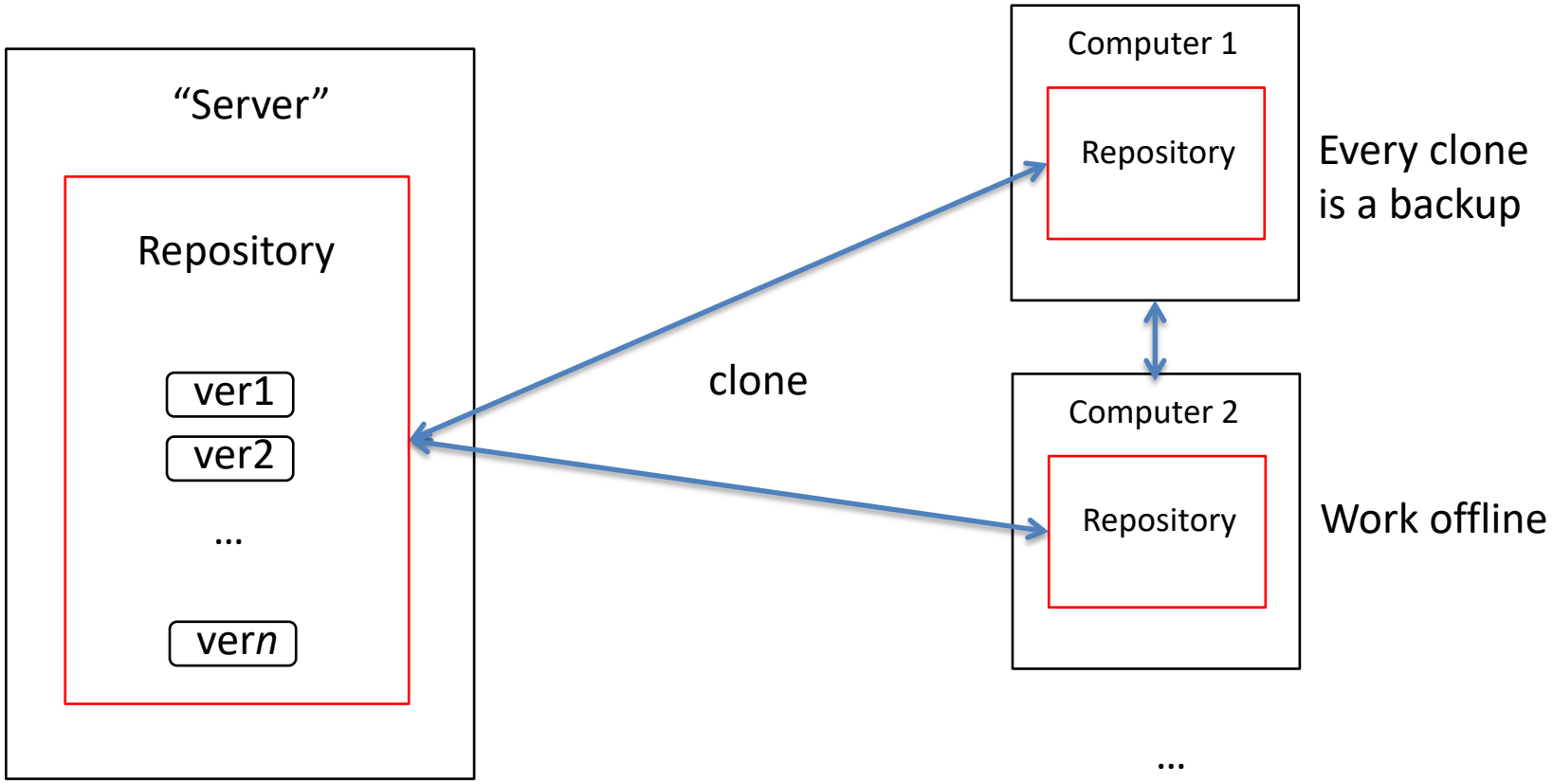
# Centralized VCS



# Distributed VCS



# Distributed VCS




There is by convention – an upstream repo – to stay in sync

# Getting started

- Install git ✓
- Configure git

# Configure git

# Git customization

- System /etc/gitconfig
- User $HOME/.gitconfig
- Project my_project/.git/config

Git commands to edit configuration:

git config --system (system)
git config --global (user)
git config (project)

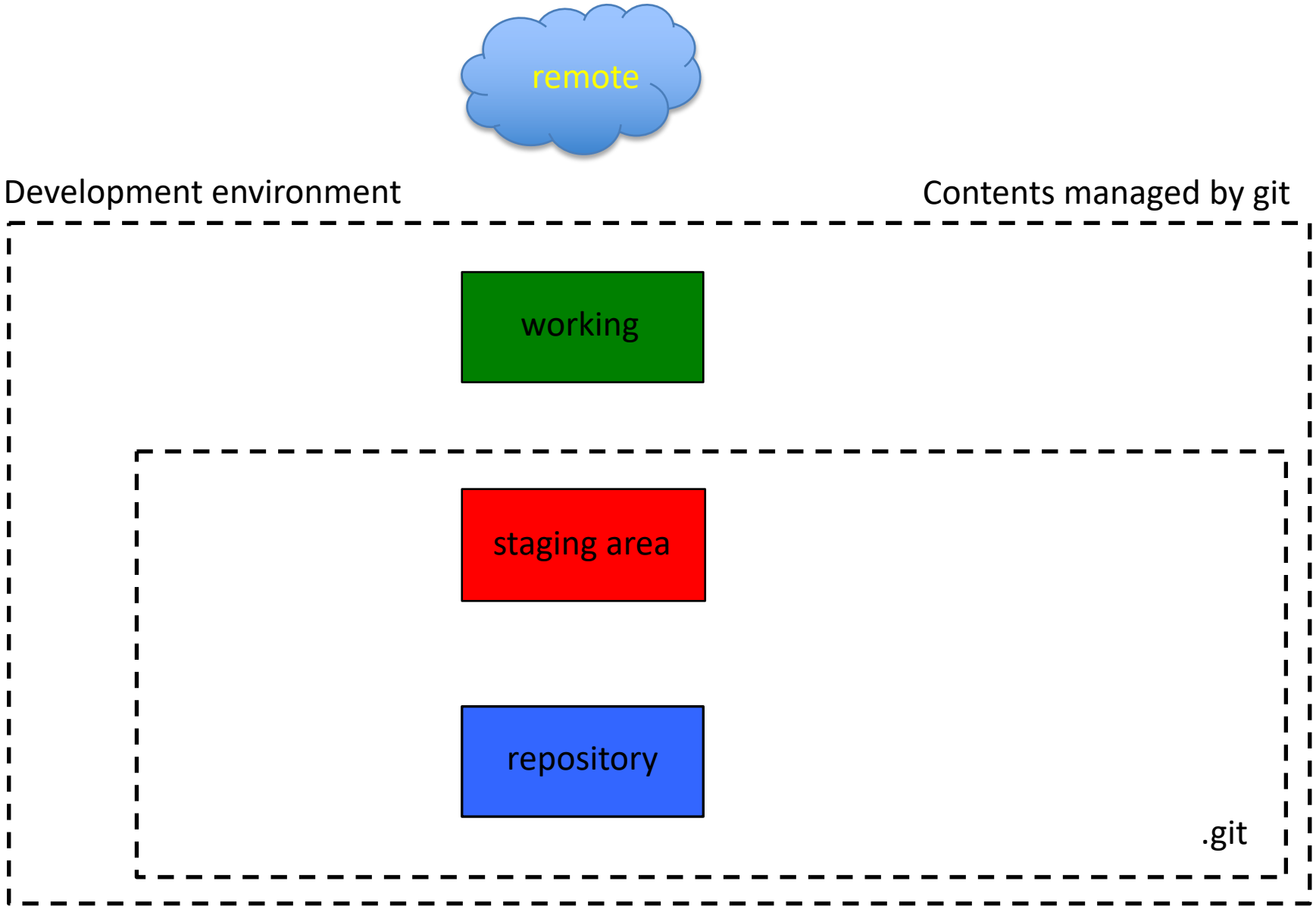
**$ git config --global user.name "[name]"**
Sets the name you want atached to your commit transactions

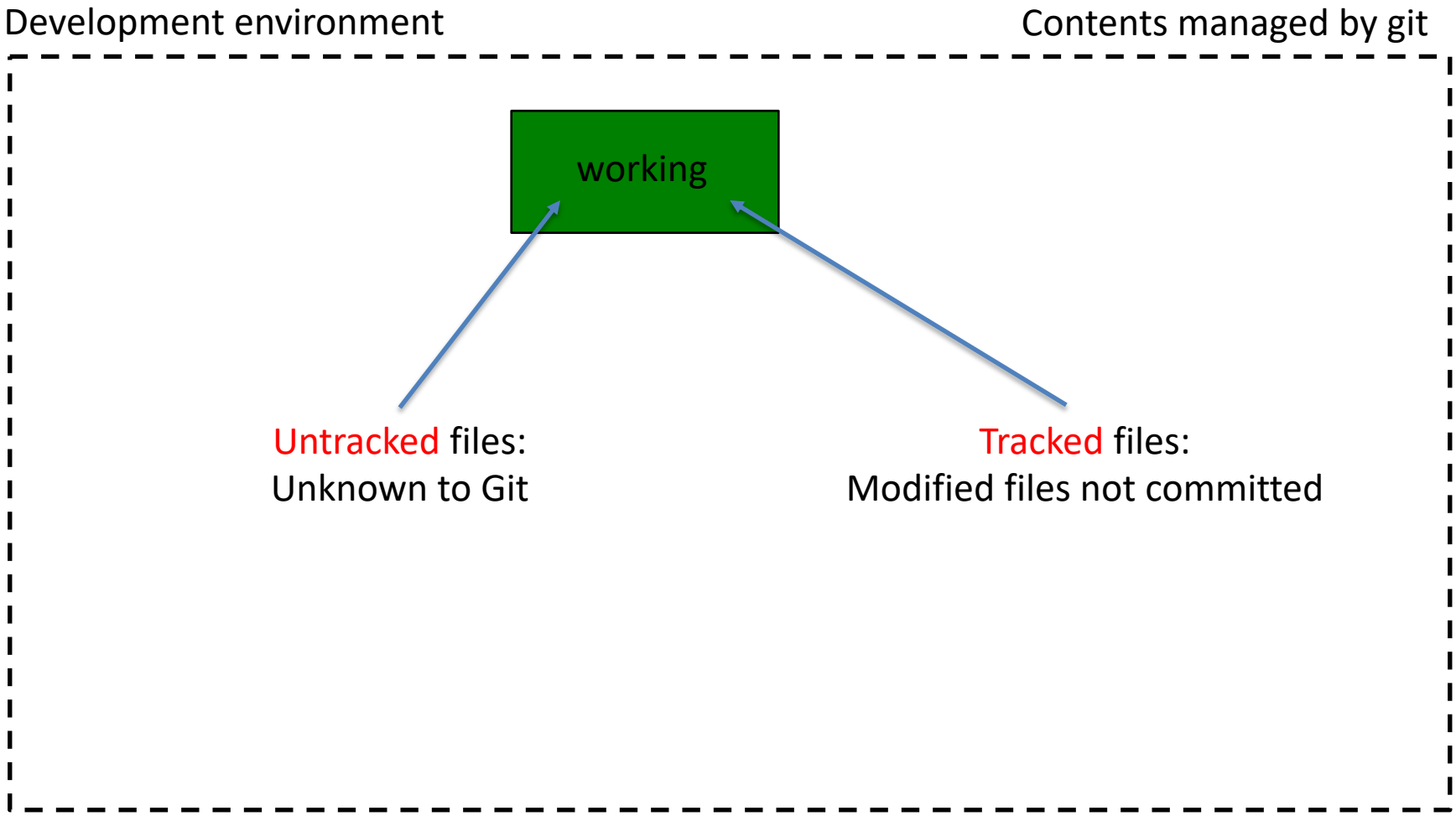
**$ git config --global user.email "[email address]"**
Sets the email you want atached to your commit transactions

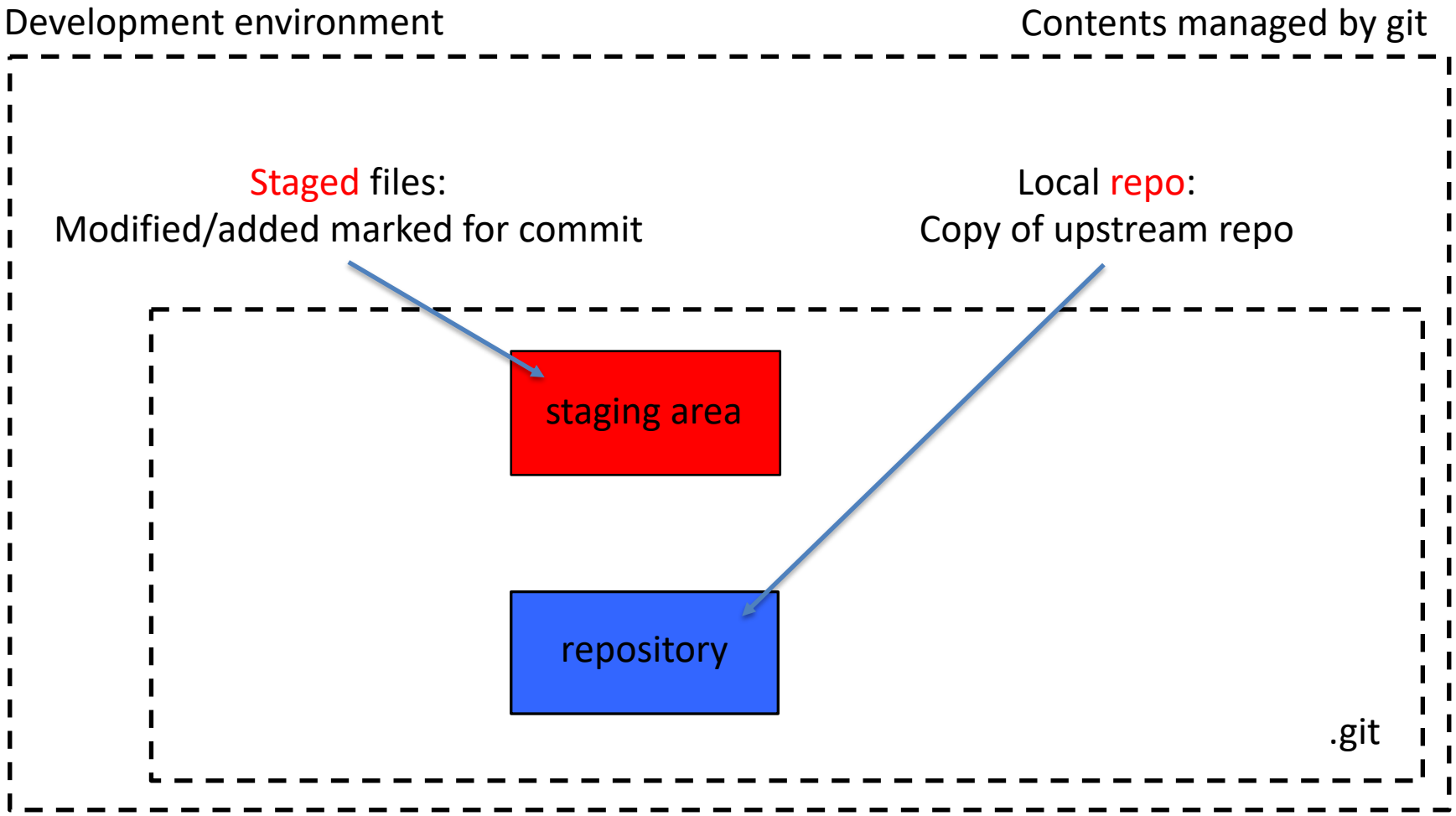
**$ git config --global color.ui auto**
Enables helpful colorization of command line output

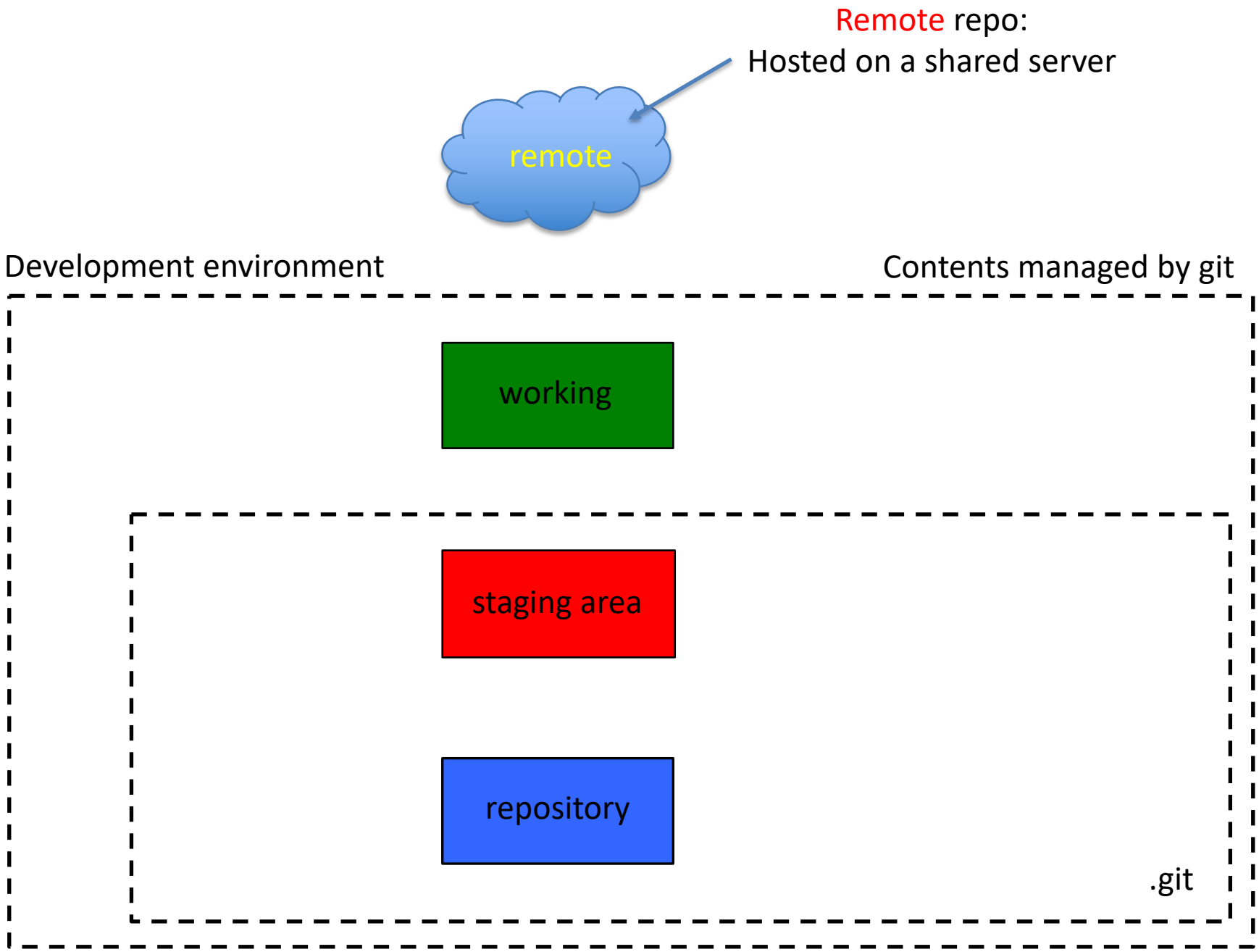
# Getting started

- Install git ✓
- Configure git ✓
- Git concepts
- Demo
  - Initialize a repository
    - Working with repository
    - add, commit, etc…
  - Working with a remote repository
    - Pushing changes

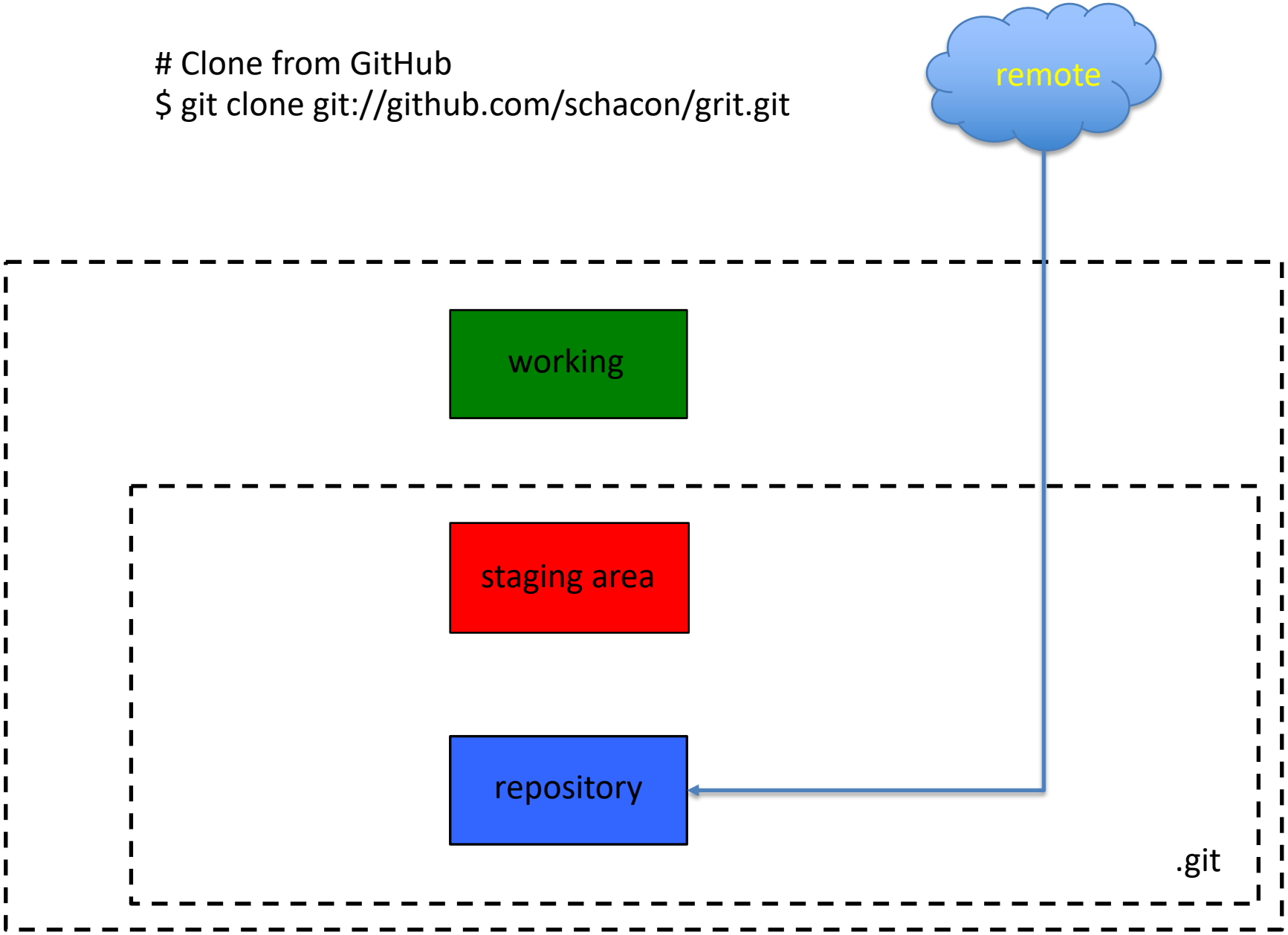

remote
Development environment
Contents managed by git
working
staging area
repository
.git


Development environment
Contents managed by git
working
Untracked files:
Unknown to Git
Tracked files:
Modified files not committed

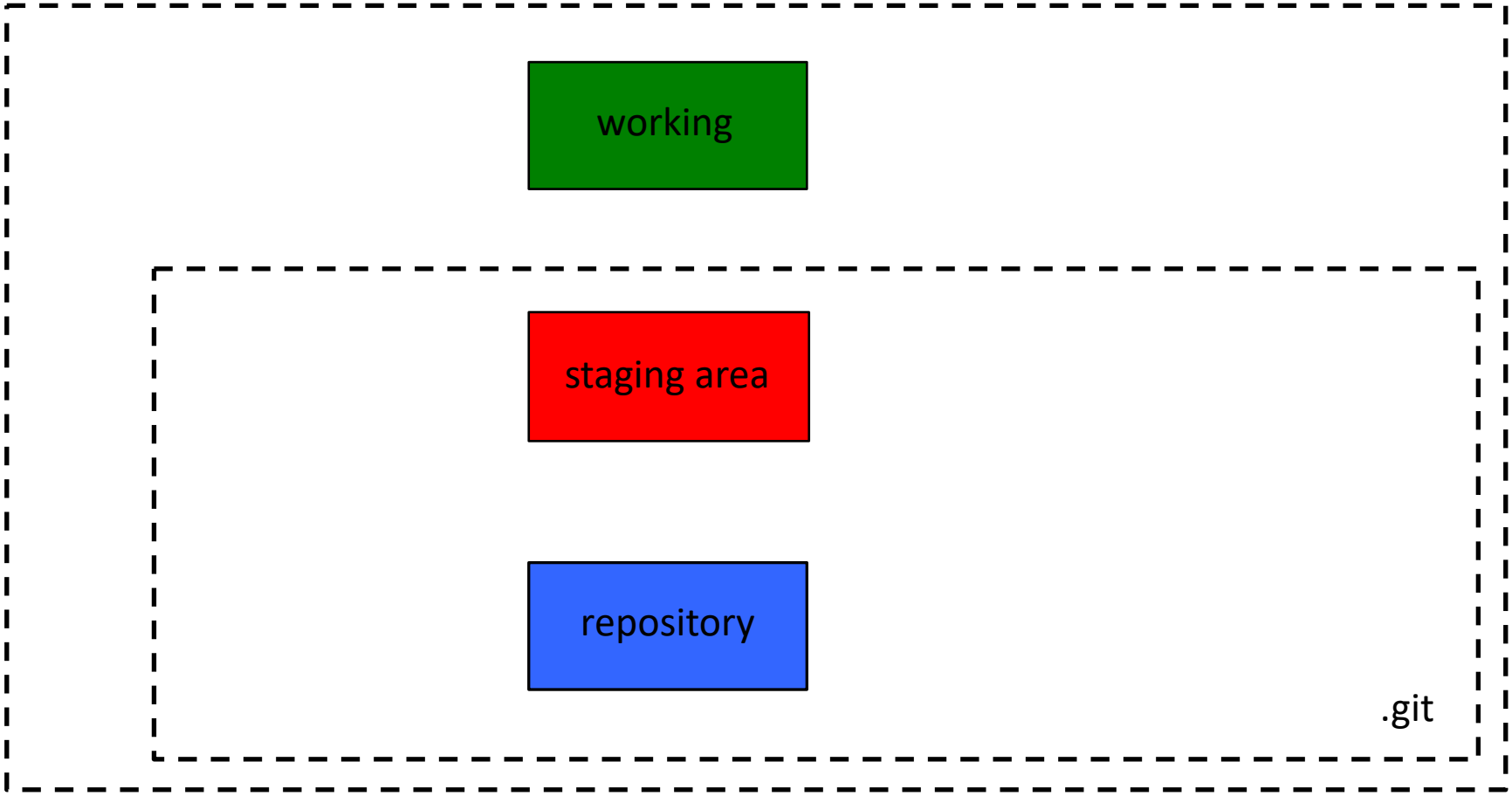

Development environment
Contents managed by git
Staged files:
Modified/added marked for commit
Local repo:
Copy of upstream repo
staging area
repository
.git


Remote repo:
Hosted on a shared server
remote
Development environment
Contents managed by git
working
staging area
repository
.git

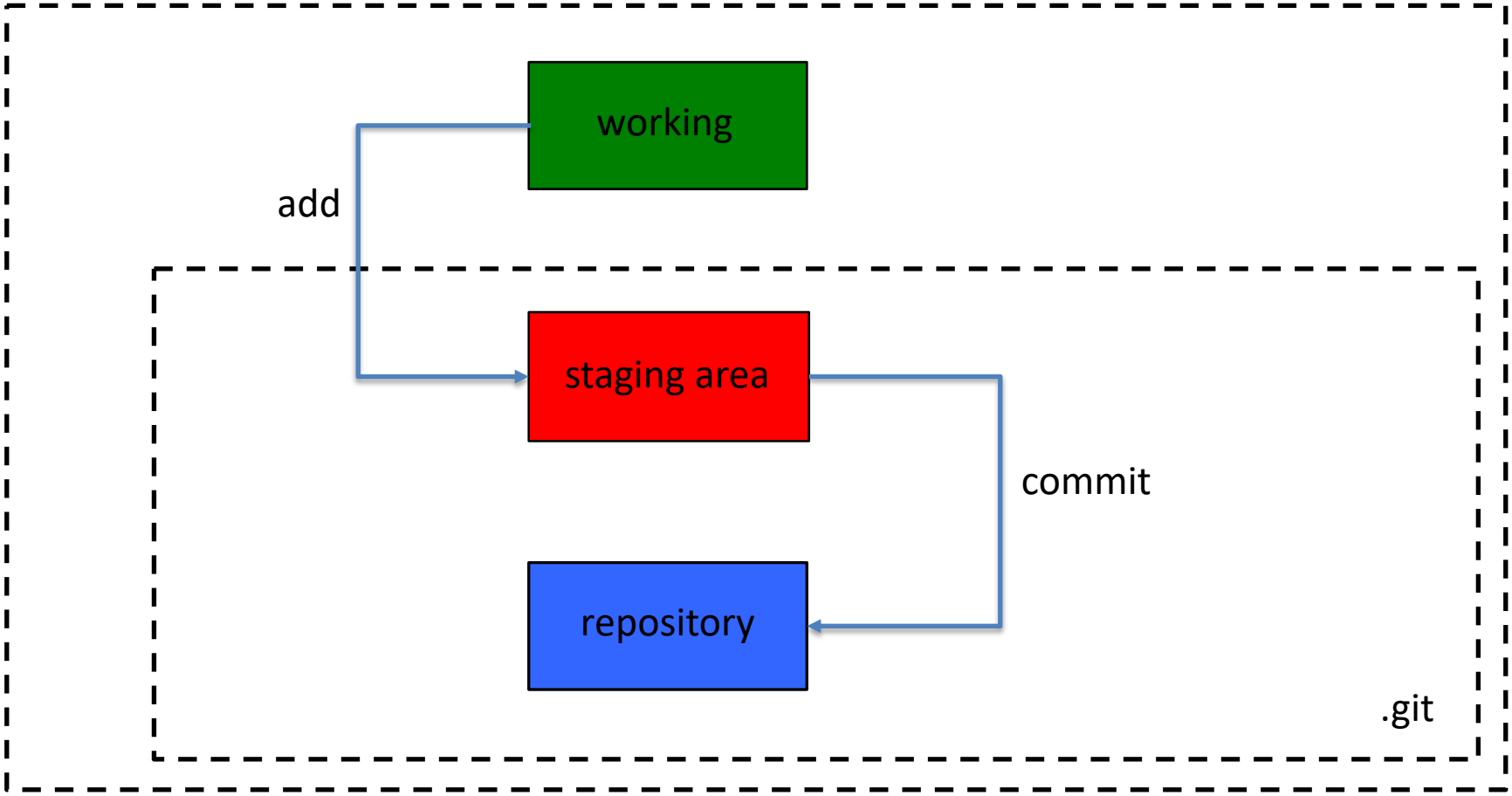
# Clone from GitHub
$ git clone git://github.com/schacon/grit.git



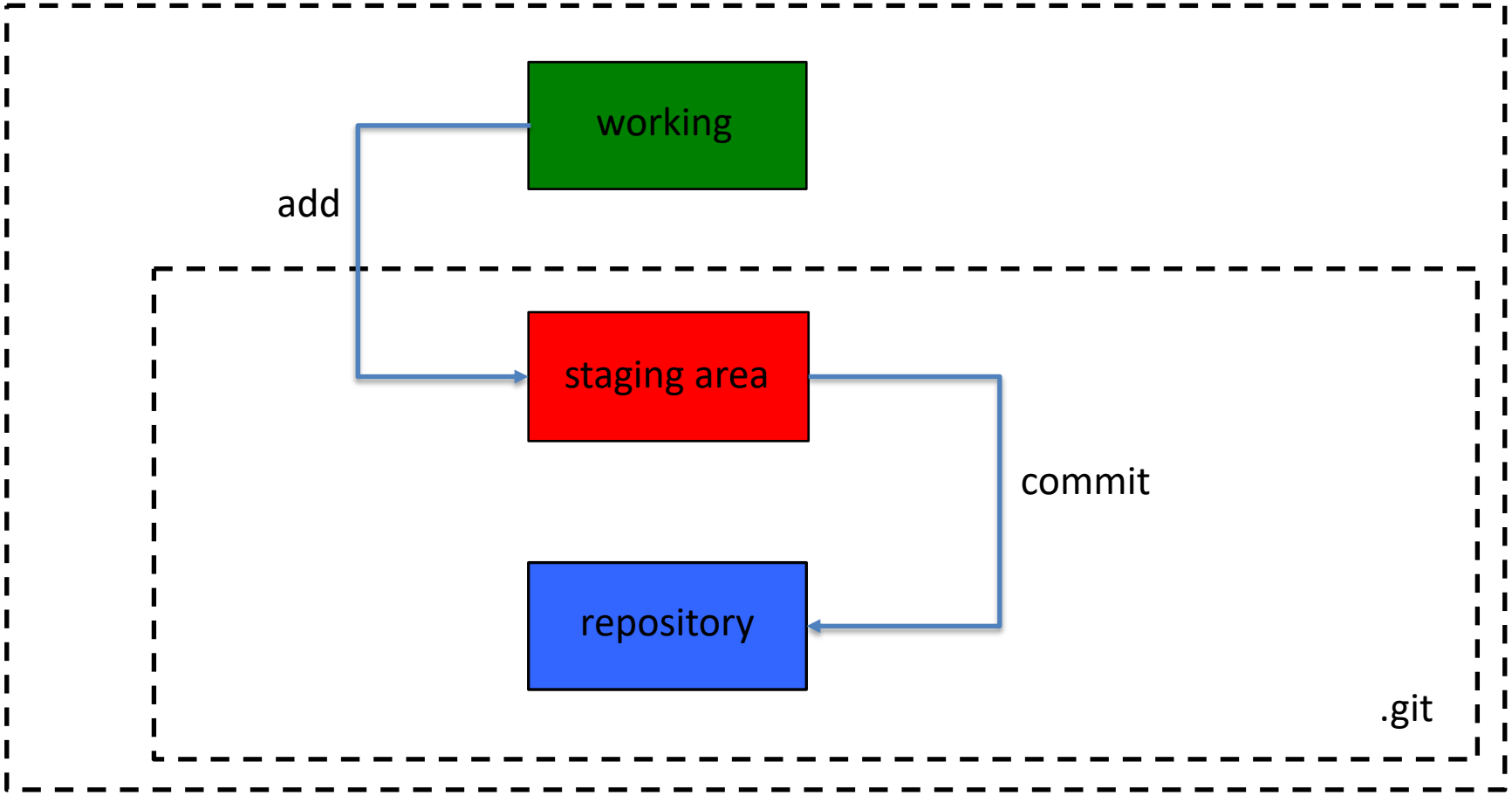
```
$ mkdir myproj
$ cd myproj
# Put myproj under version control
$ git init
Initialized empty Git repository in /some/path/myproj
```



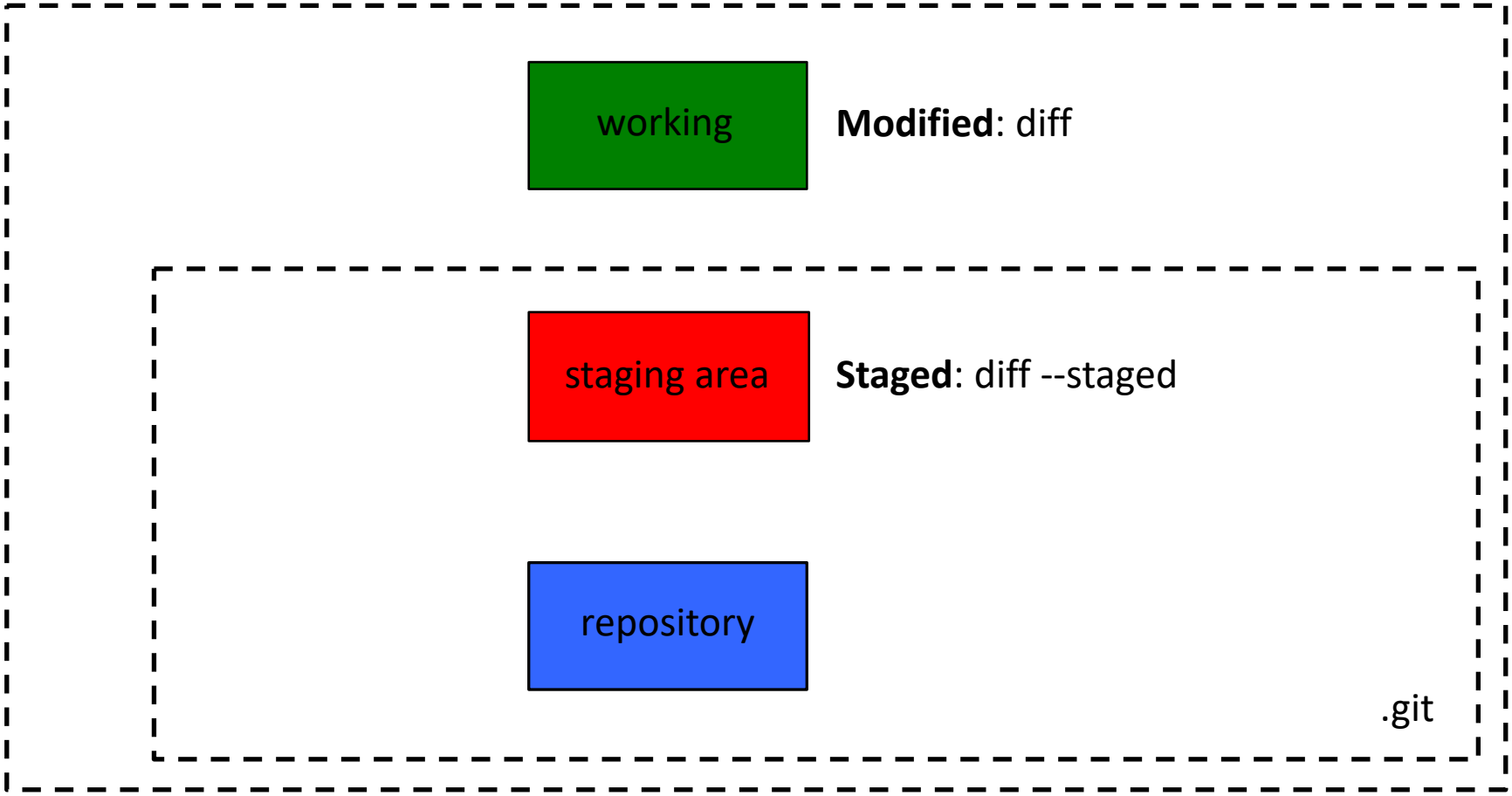
```
# Add a file
$ git add README
# Save a file
$ git commit –m ’Add a file’ README
```

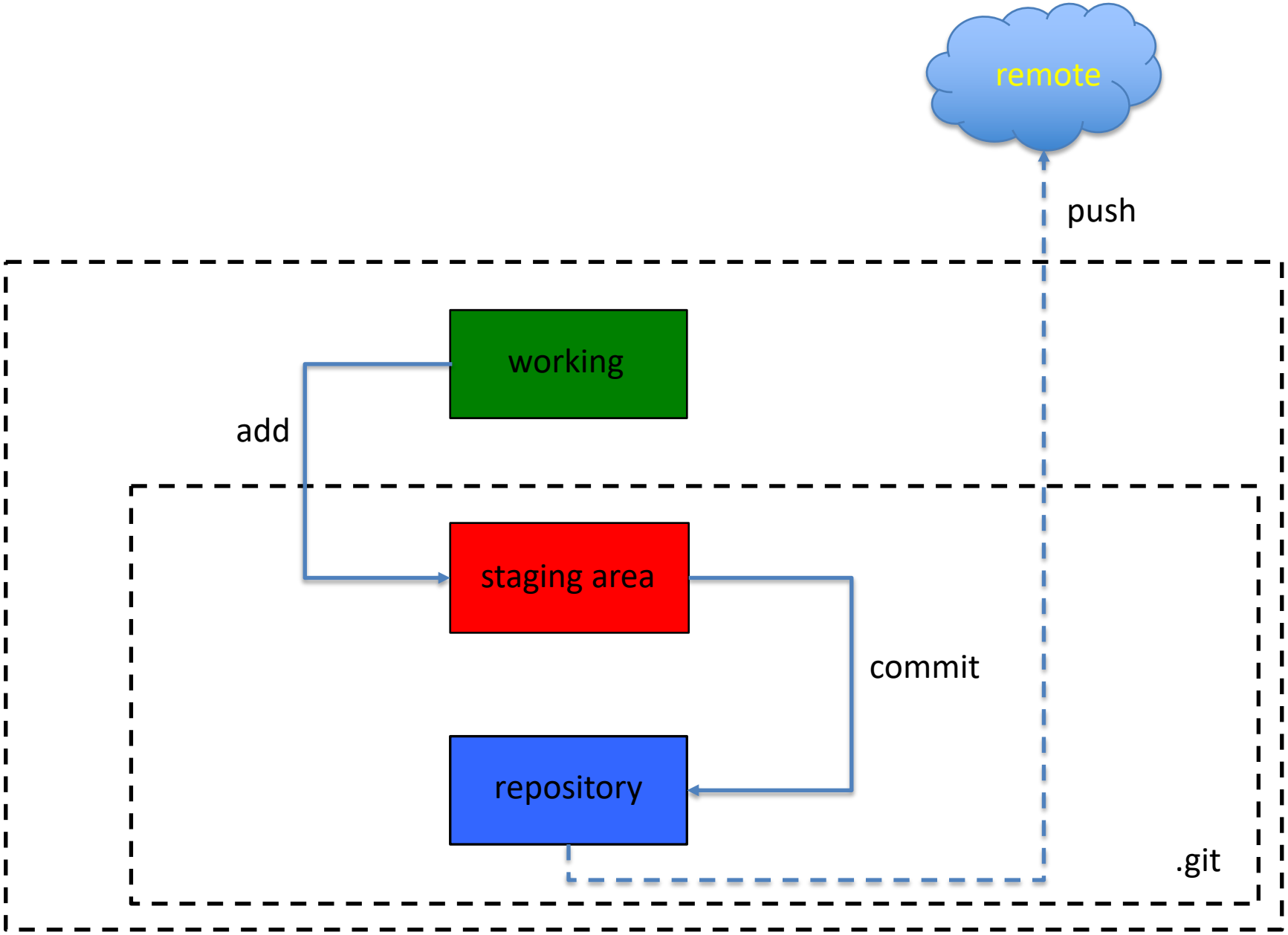


# Show commit log
$ git log
# Show status of repo
$ git status

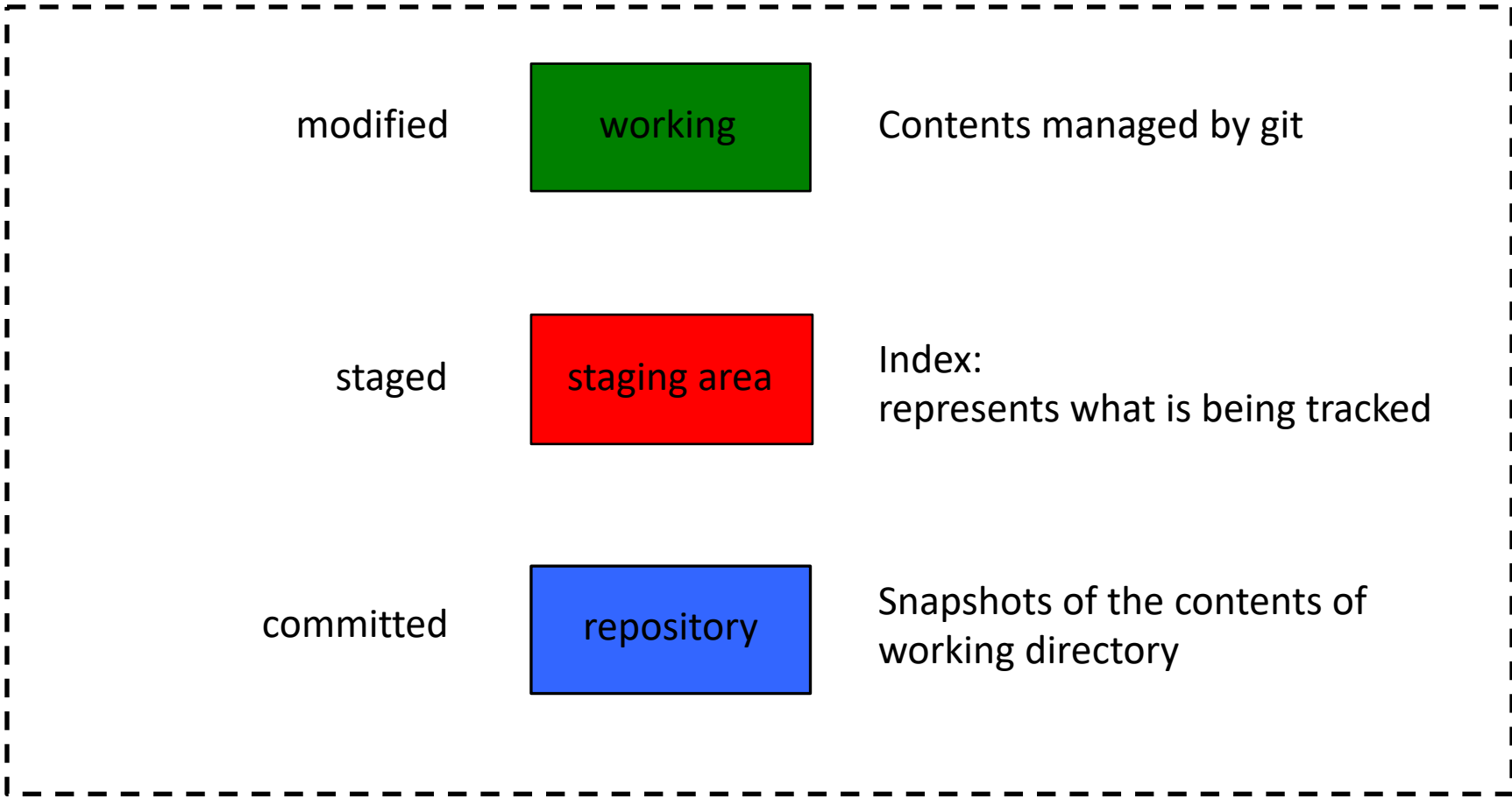


```
# Show commit log
$ git log
# Show status of repo
$ git status
# Differences
$ git diff
$ git diff --staged
```

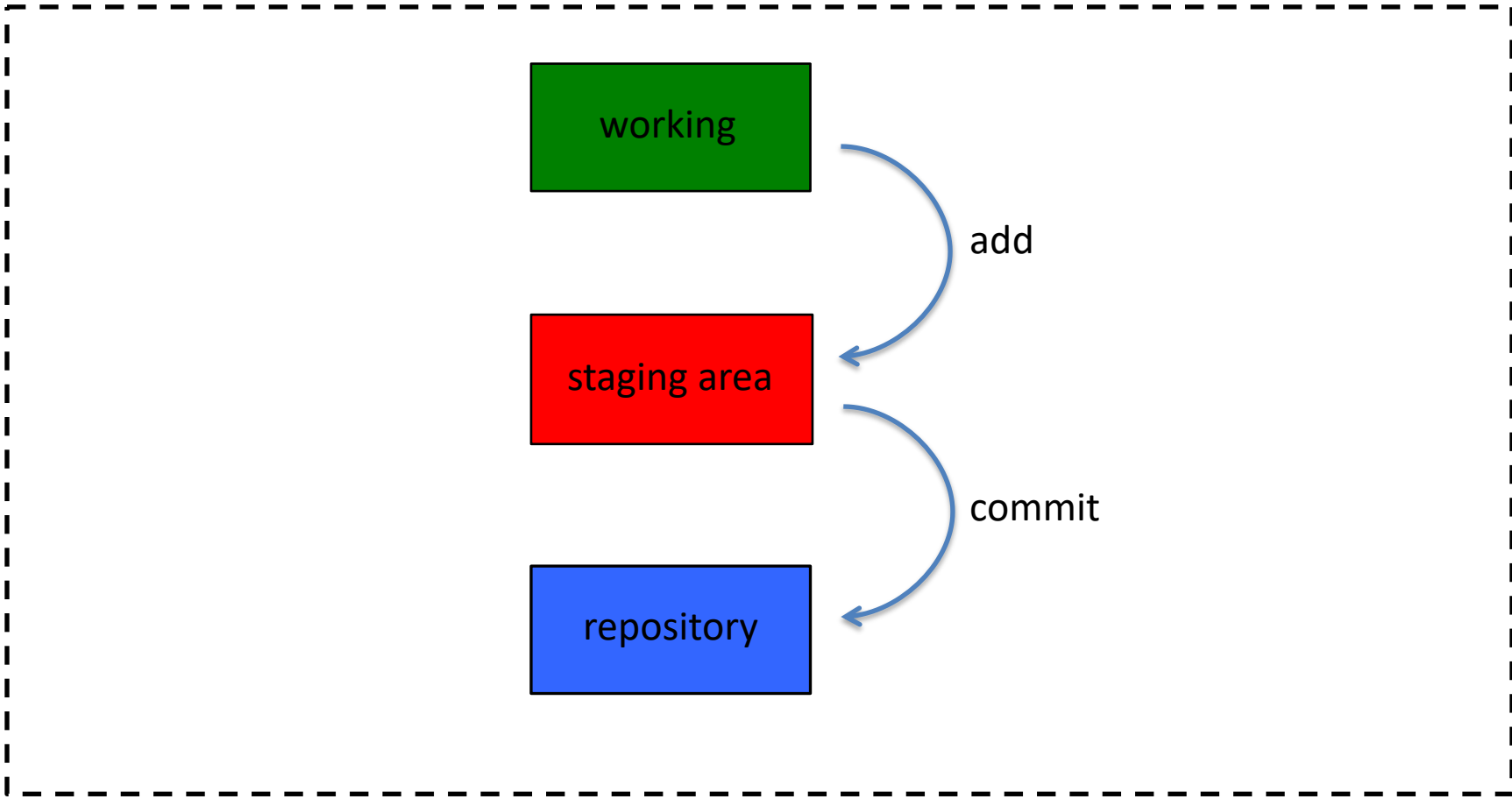



remote
push
working
add
staging area
commit
repository
.git

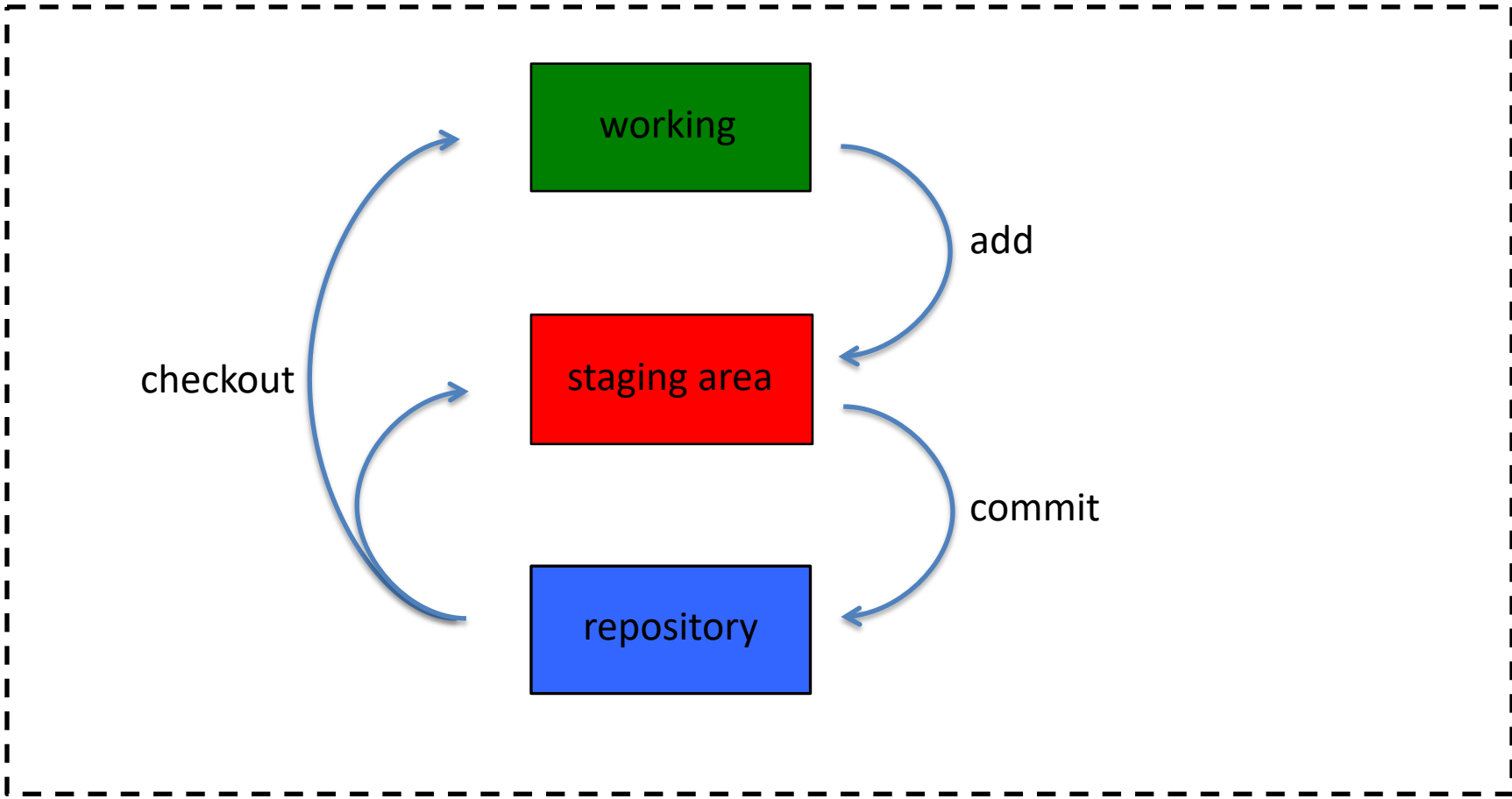
# Git concepts



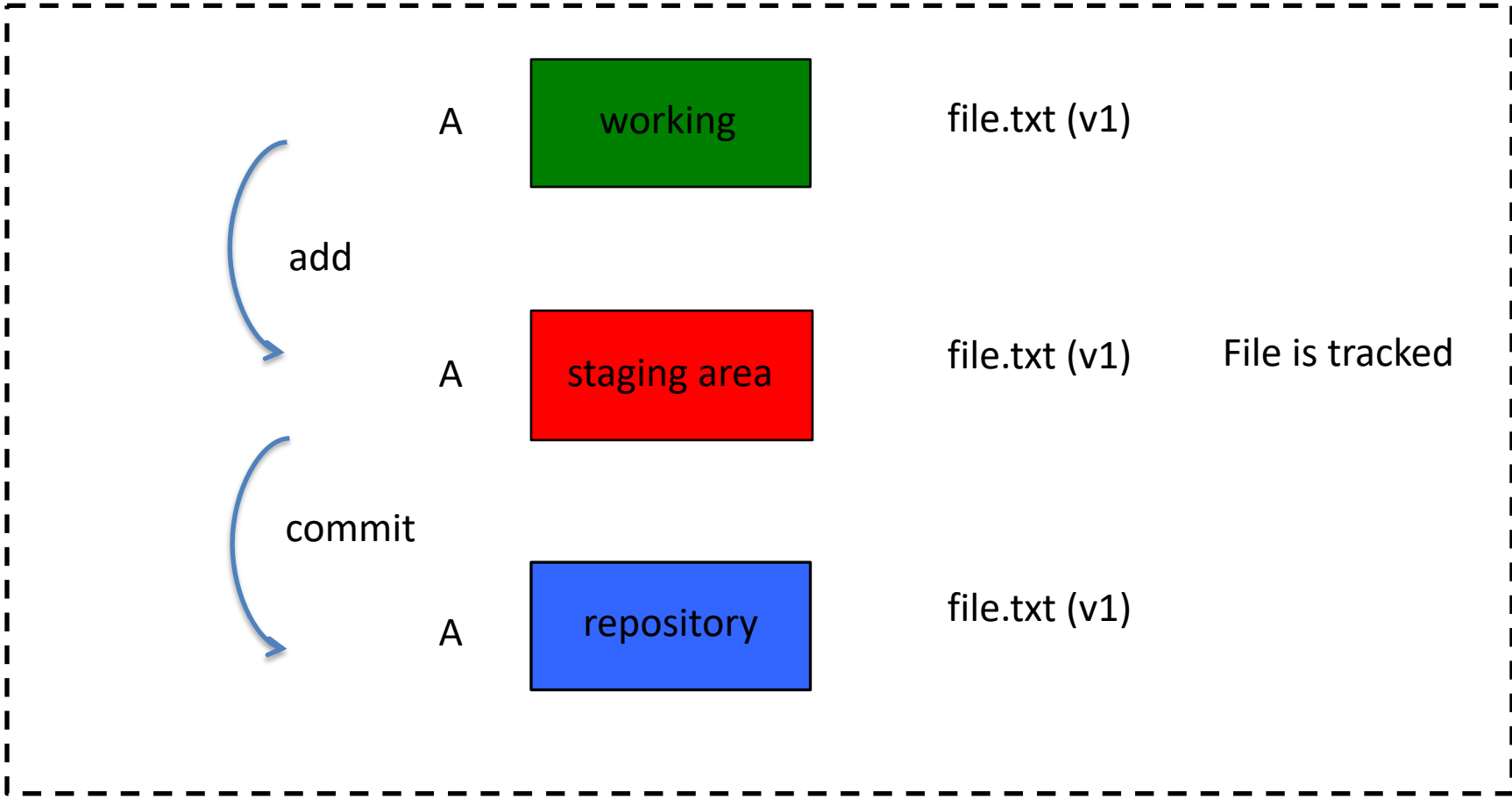
# Git concepts



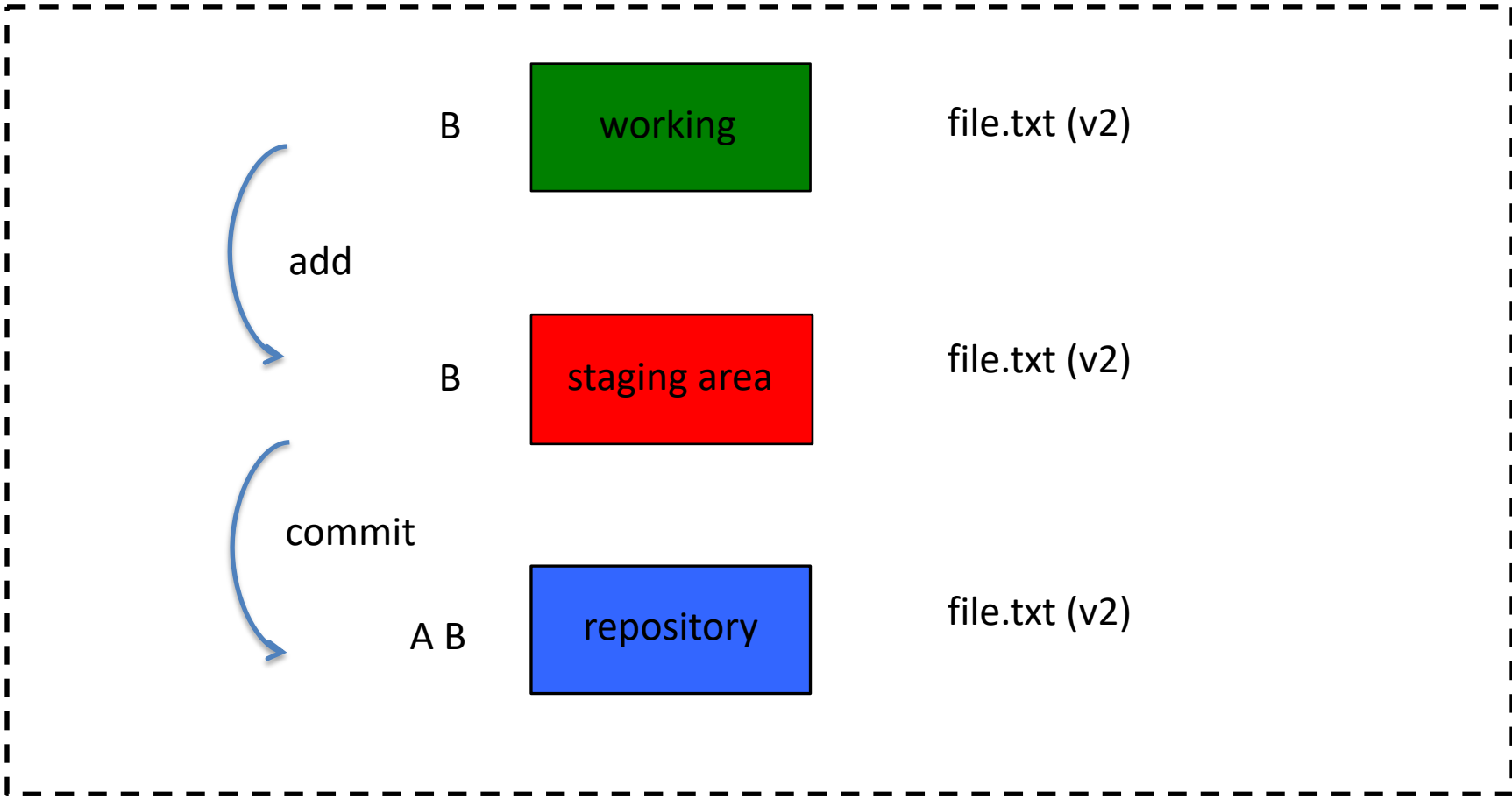
# Git concepts



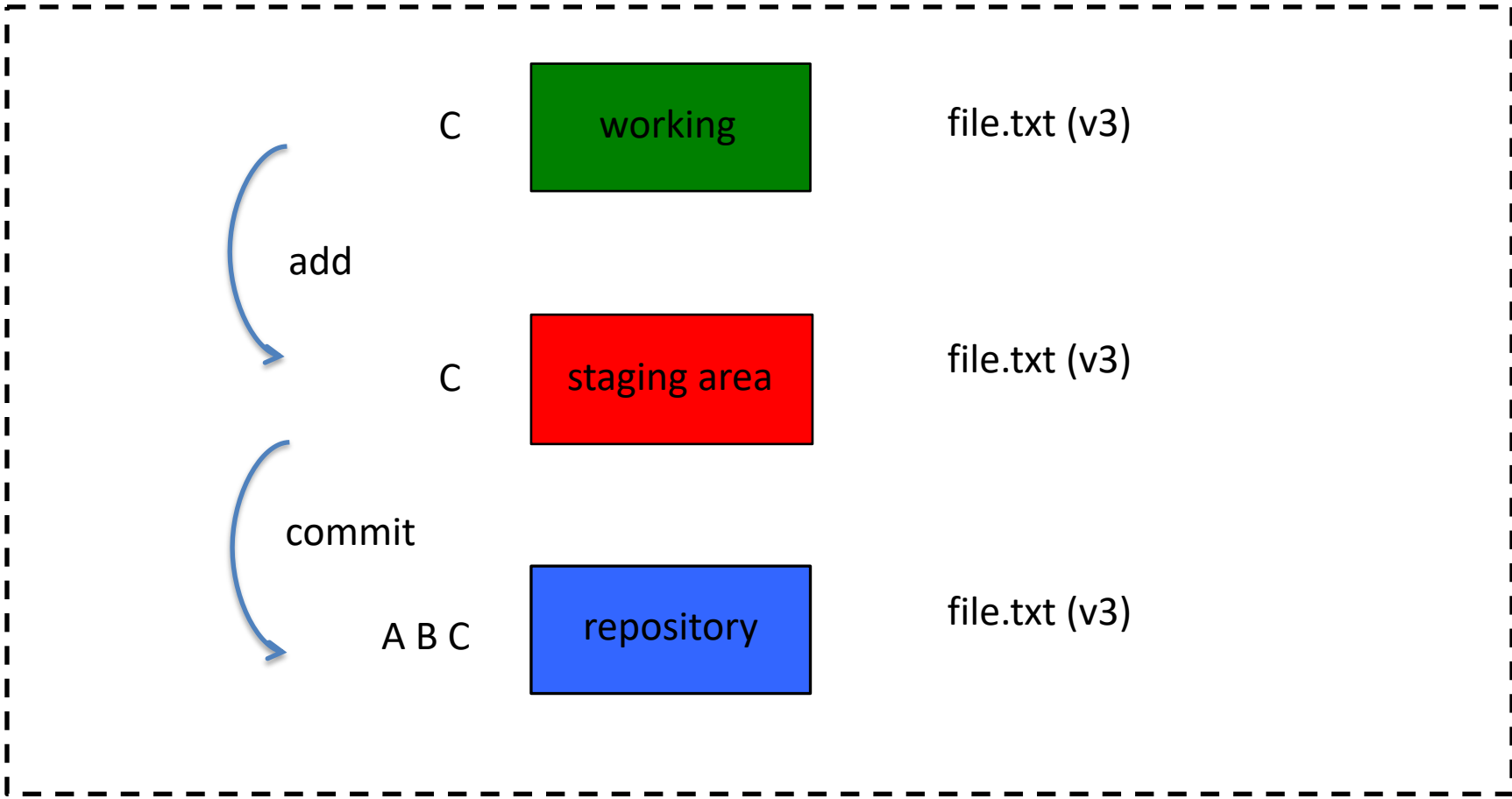
# Git concepts



# Git concepts



# Git concepts



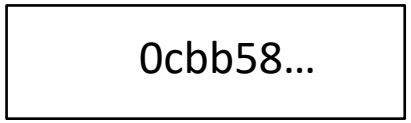

A, B and C are change sets

# Git concepts

For each change set (i.e. each commit A,B,C):

- Git generates a checksum
- Git uses SHA-1 algorithm to create checksums
  - 40-character hexadecimal string
  - e.g. 1fbb5af06b9e4facff4170fc687ecdd143daad50

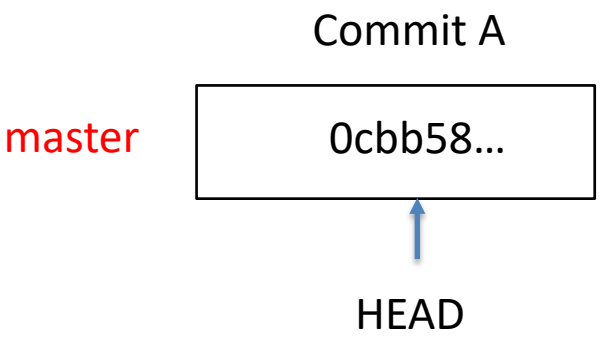
# Git concepts



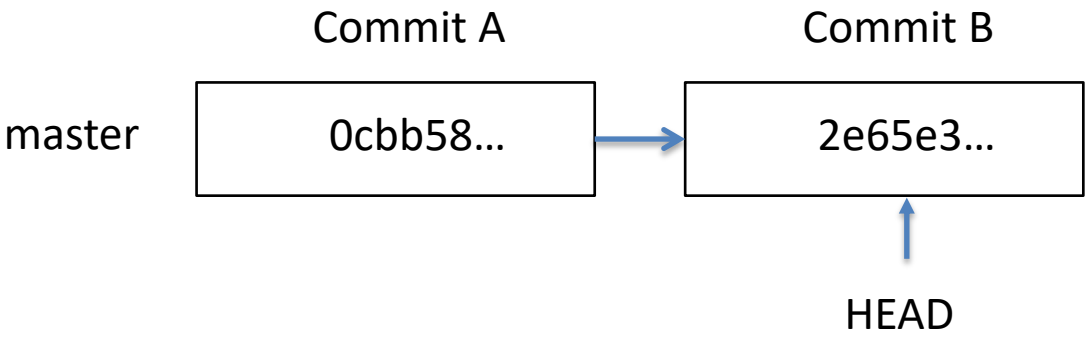

Commit:

Represented by 40-char hex string
Author
Timestamp
Message
Parent

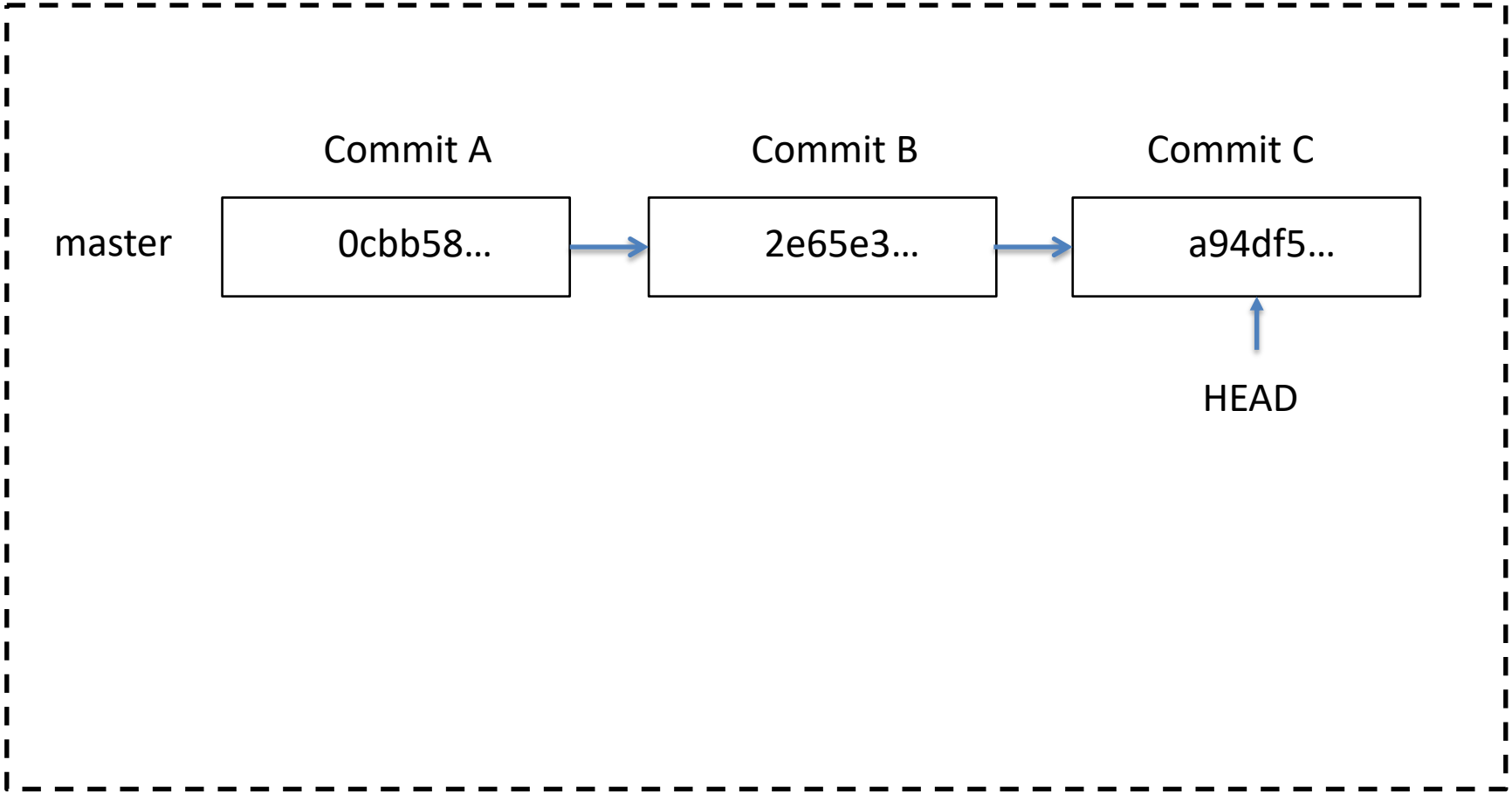
# Git concepts



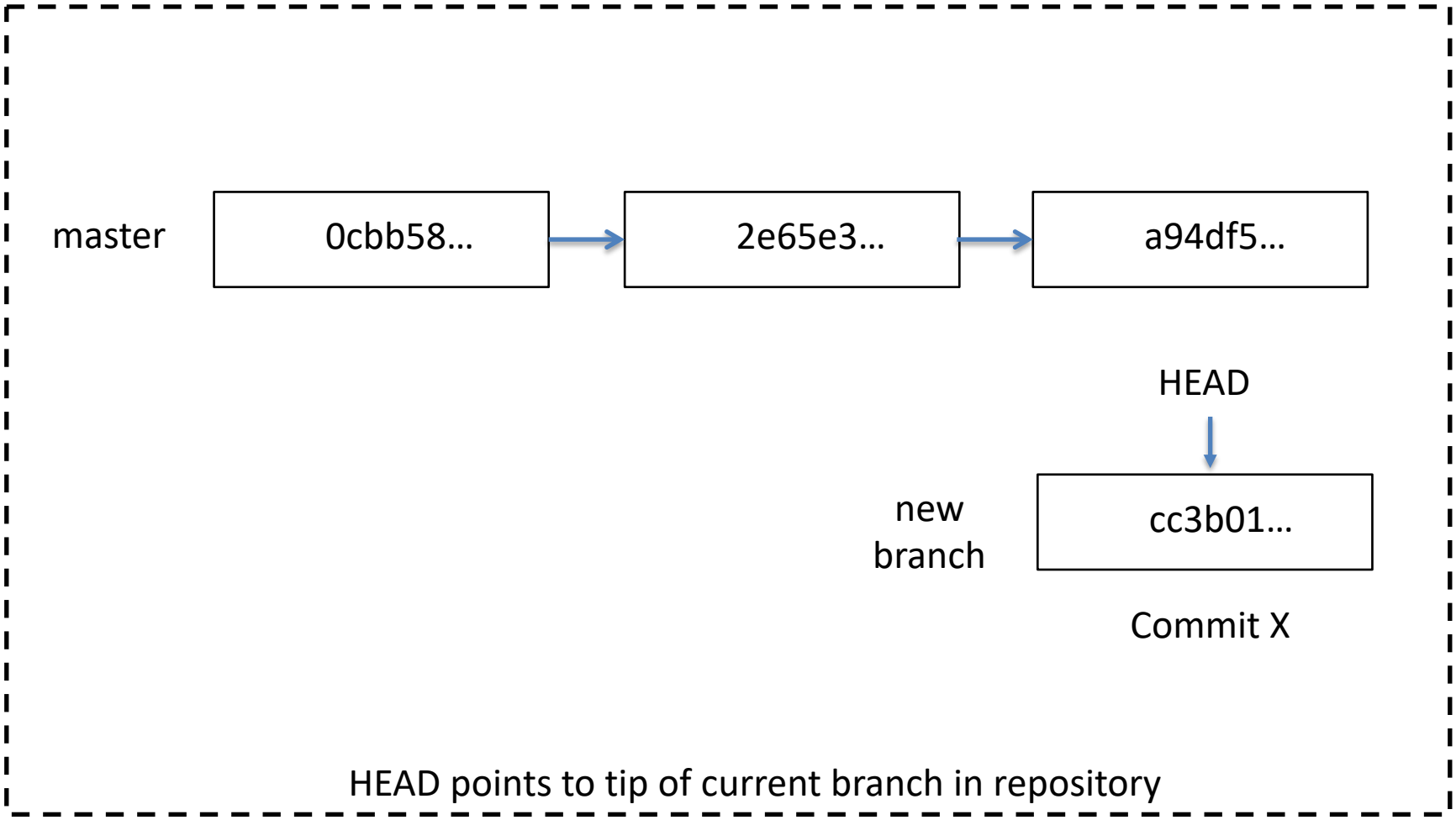
# Git concepts



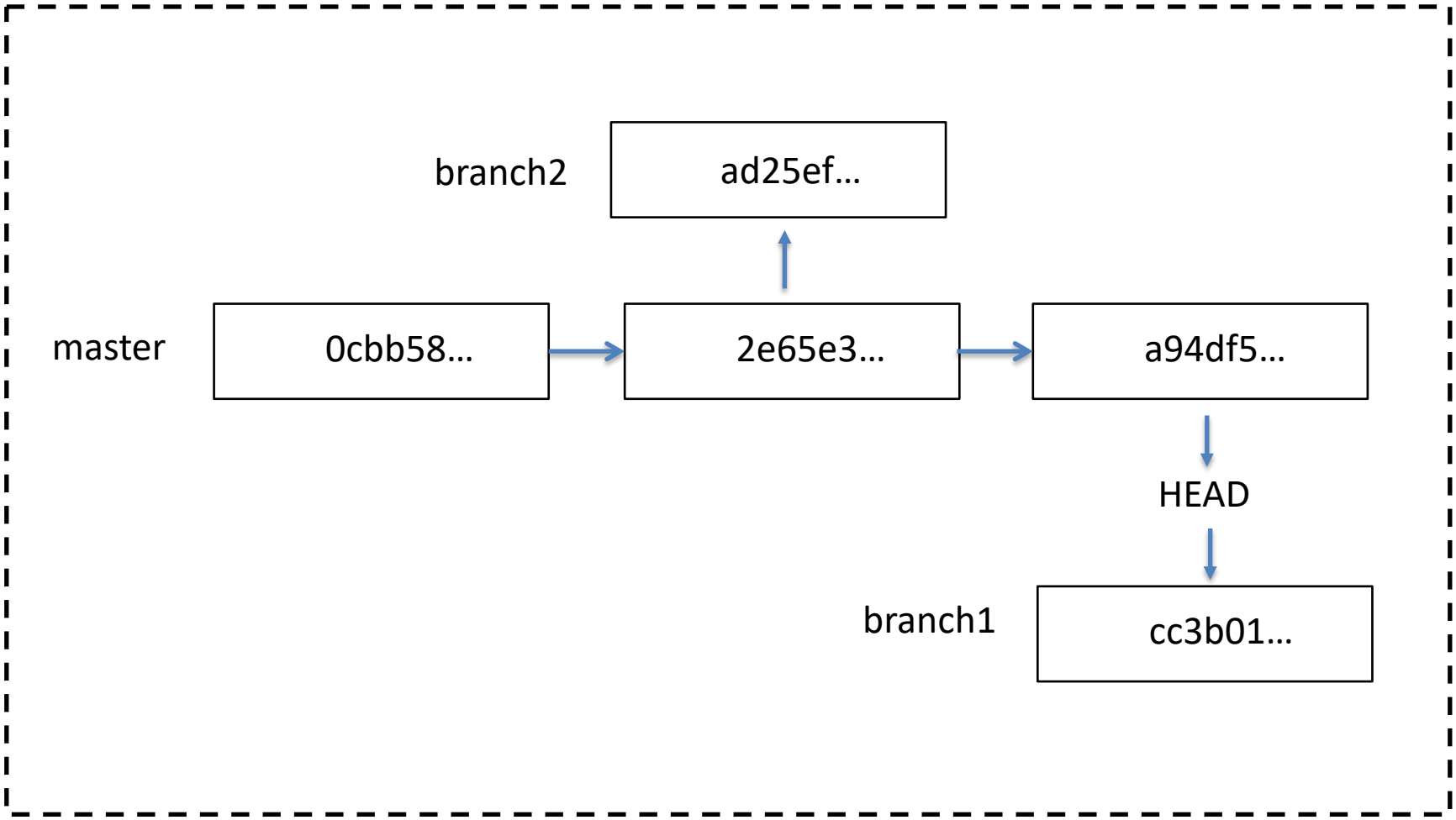
# Git concepts



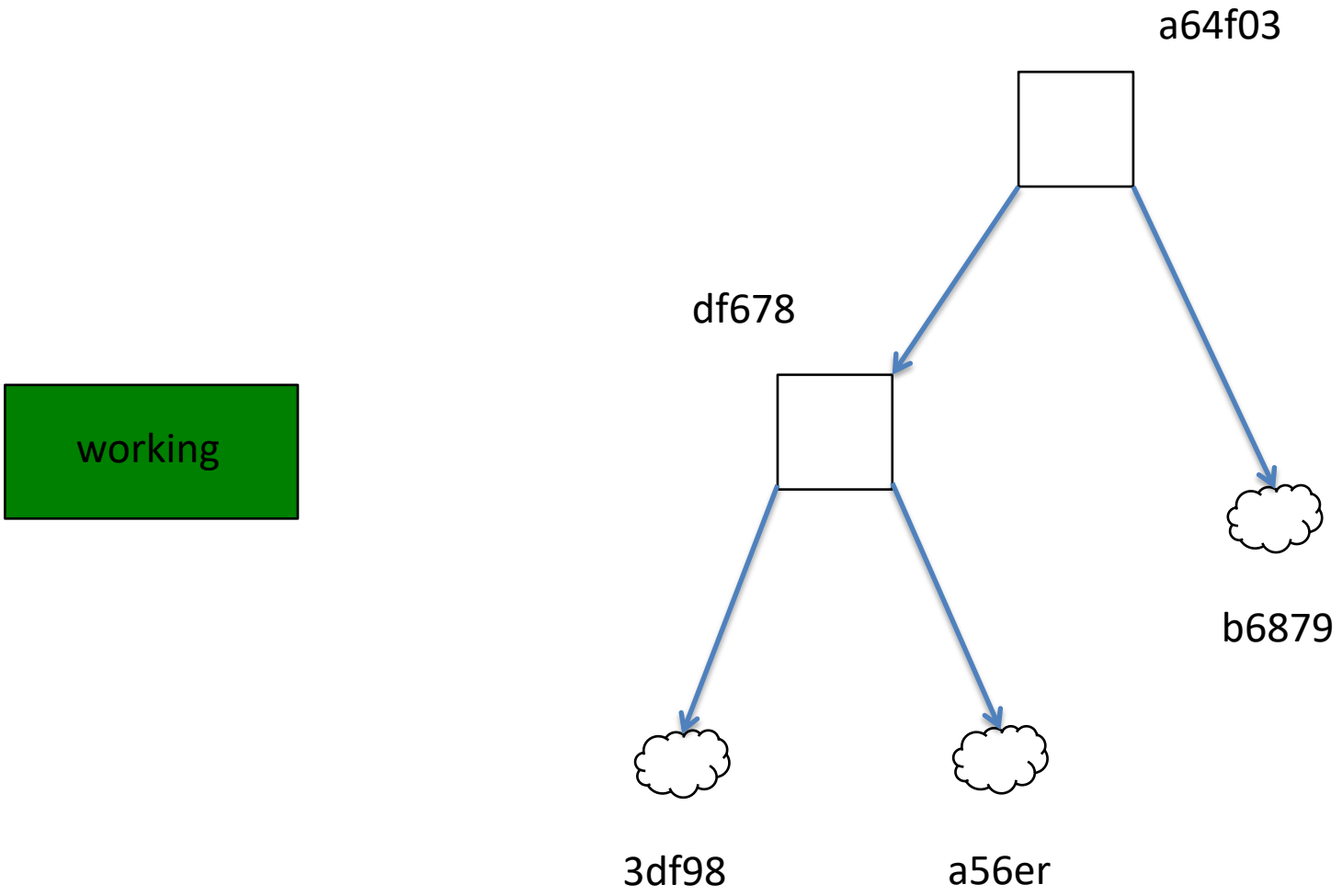
# Git concepts



# Git concepts



# Git concepts



# Demo

# Common commands

Create repositories

**$ git init [project-name]**

Creates a new local repository with the specified name

**$ git clone [url]**

Downloads a project and its entire version history

Make changes

**$ git status**

Lists all new or modified files to be commited

**$ git add [file]**

Snapshots the file in preparation for versioning

**$ git reset [file]**

Unstages the file, but preserve its contents

**$ git diff**

Shows file differences not yet staged

**$ git diff --staged**

Shows file differences between staging and the last file version

**$ git commit -m "[descriptive message]"**

Records file snapshots permanently in version history

Group changes

**$ git branch**

Lists all local branches in the current repository

**$ git branch [branch-name]**

Creates a new branch

**$ git checkout [branch-name]**

Switches to the specified branch and updates the working directory

**$ git merge [branch]**

Combines the specified branch's history into the current branch

**$ git branch -d [branch-name]**

Deletes the specified branch

Review history

**$ git log**

Lists version history for the current branch

**$ git log --follow [file]**

Lists version history for a file, including renames

**$ git diff [first-branch]...[second-branch]**

Shows content differences between two branches

**$ git show [commit]**

Outputs metadata and content changes of the specified commit

# References

- https://git-scm.com/doc
- http://gitref.org/

# The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

The following paragraphs are extracted from the report of the Inquiry Board. An interesting article on the accident and its implications by James Gleick appeared in The New York Times Magazine of 1 December 1996. The CNN article reporting the explosion, from which the above graphics were taken, is also available.

*On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.*

*The failure of the Ariane 501 was caused by the complete loss of guidance and altitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.*

*The internal SRI* software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer.*

*SRI stands for Système de Référence Inertielle or Inertial Reference System.

Software error…due to conversion of floating point to integer