IT/Dev[op]s Army Knives Tools for the researcher

a journey from SSH to Git

Sebastien Varrette

Parallel Computing and Optimization Group (PCOG),

University of Luxembourg (UL), Luxembourg

Summary

Summary

Summary

Seminar Objective

- Review some of the most sensible tools every researcher
  - ↪ . . . in computer science but not only
- Focus on key IT [DevOps] Tools: **SSH, PGP, Vagrant**. . .
- Review also some best-practice for your daily work
  - ↪ Sand-boxing in Python and Ruby for your prototyping
  - ↪ Password Management
- Overview of the Research Computing platforms @ UL
- Overview of Version Control System (VCS) and **Git** in particular

Agenda Part I (9h45-12h00)

Location: room B21, campus Kirchberg

- **SSH** Secure Shell
  - ↪ Overview and Basic usage
  - ↪ Advanced usage (proxy SOCKS, multi-jump w. ProxyCommand)
- **PGP** / GPG: Gnu Privacy Guard
- **Vagrant**: Development environment made easy
- **Puppet**: Configuration Management
- Sandboxed and reproducible running environment across developers
  - ↪ Ruby & Python
- **Markdown**-based documentation, articles and slides
  - ↪ Overview of the Markdown syntax
  - ↪ Git-based Markdown Wiki: gollum, mkdocs
  - ↪ using Markdown with LaTeX and Beamer
- **Password Management**
- Overview of the **Research Computing platforms @ UL**

Agenda Part II (13h15 - 15h00)

Location: room B21, campus Kirchberg

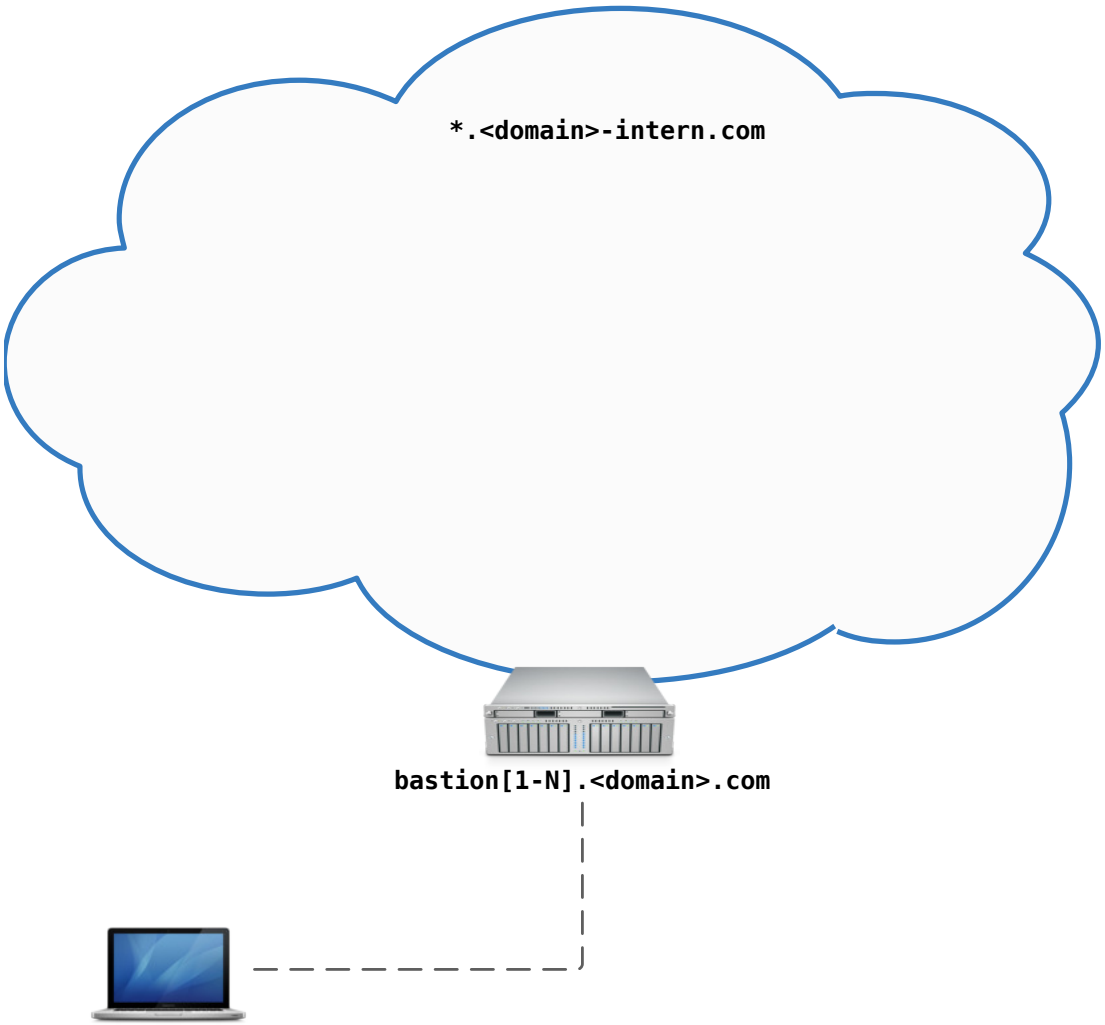
- Introduction to Version Control System (**VCS**)
- Git Basics
  - ↪ Installing Git
  - ↪ Git theory
  - ↪ Basic Commands Branching and Merging
- Collaborating / Working together
- Advanced Git Topics
  - ↪ Git Submodules
  - ↪ Rebasing
  - ↪ Using Git over Subversion Repository
- More Cool stuff

Technical Recommendation

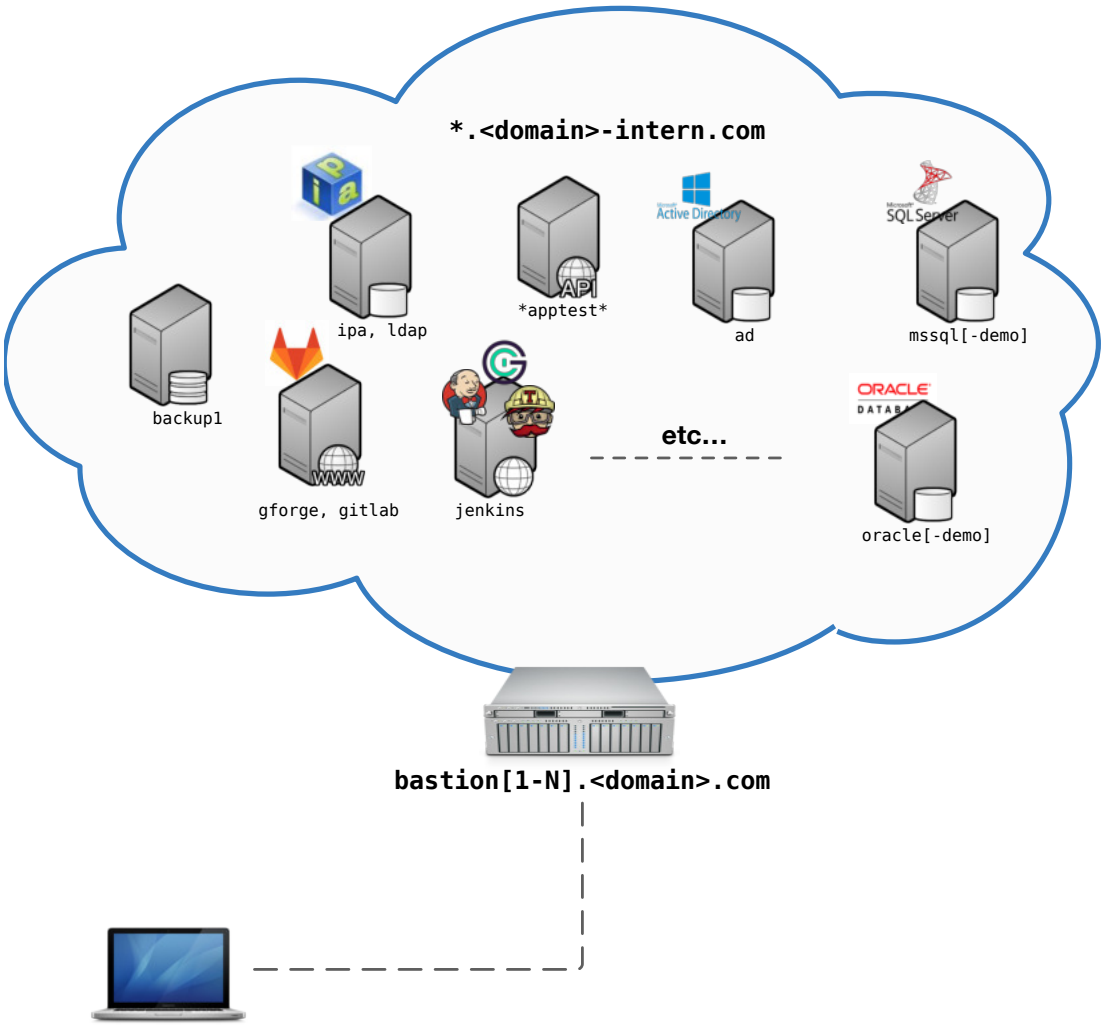
- All attendee are strongly encouraged to bring their computer laptop
  - ↪ the talk integrate a set of hands-on / exercises
- Start to install some components:
  - ↪ Mac OS: Homebrew `http://brew.io`
  - ↪ Virtualbox: `https://www.virtualbox.org/`
  - ↪ Vagrant: `https://www.vagrantup.com/downloads.html`

Summary

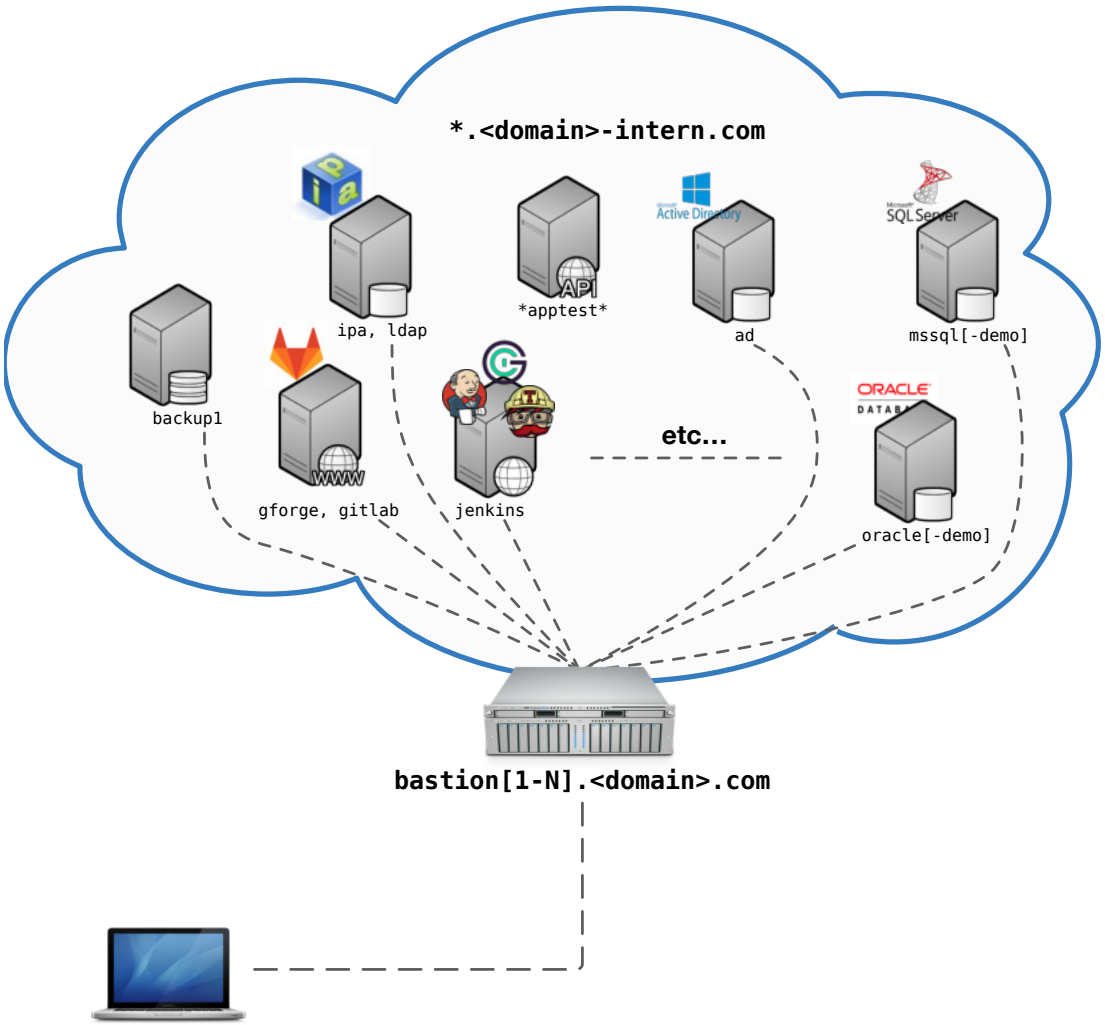
Typical [UL] IT Infrastructure



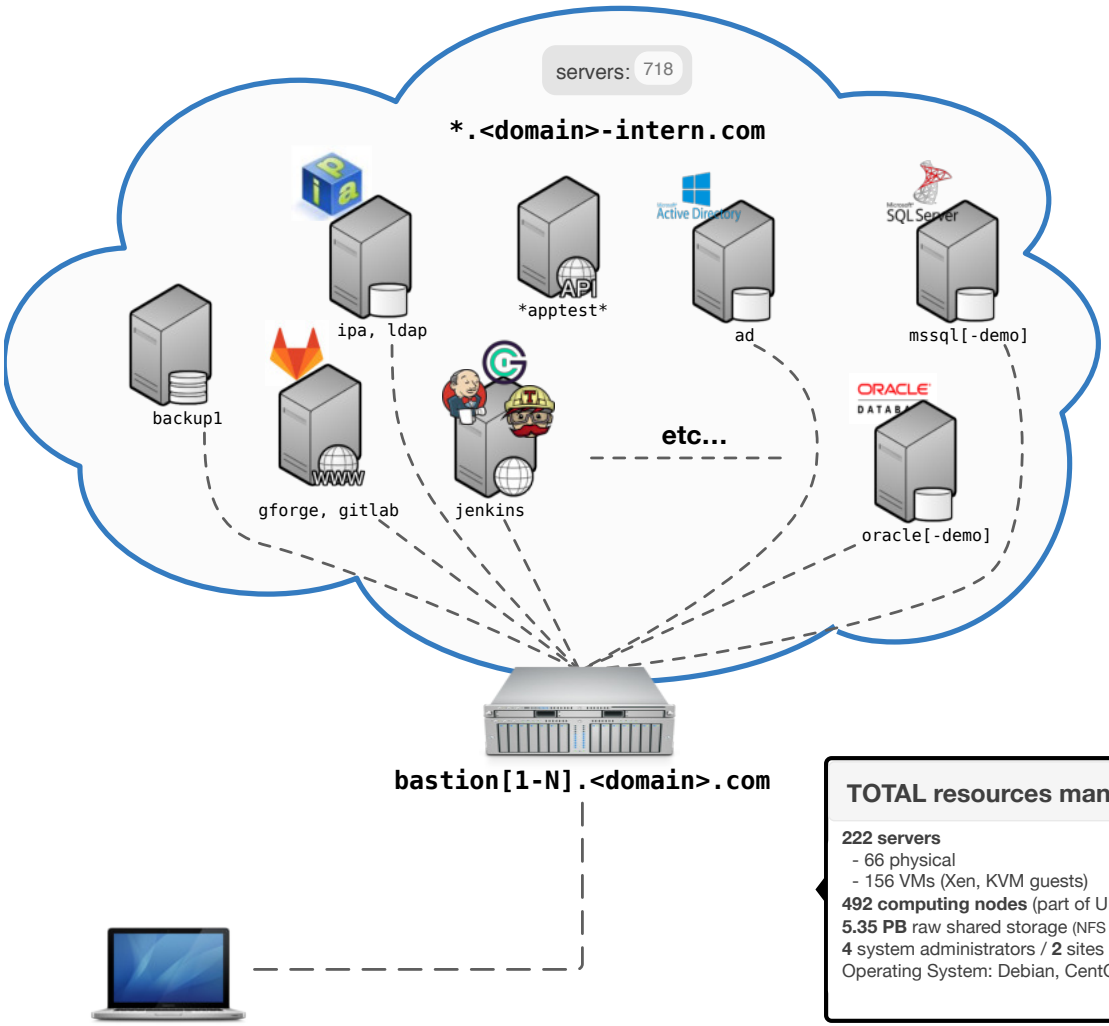
Typical [UL] IT Infrastructure



Typical [UL] IT Infrastructure



Typical [UL] IT Infrastructure



TOTAL resources managed

222 servers
- 66 physical
- 156 VMs (Xen, KVM guests)

492 computing nodes (part of UL HPC facility)
5.35 PB raw shared storage (NFS / GPFS /Lustre)
4 system administrators / **2** sites
Operating System: Debian, CentOS

UL HPC Software Stack

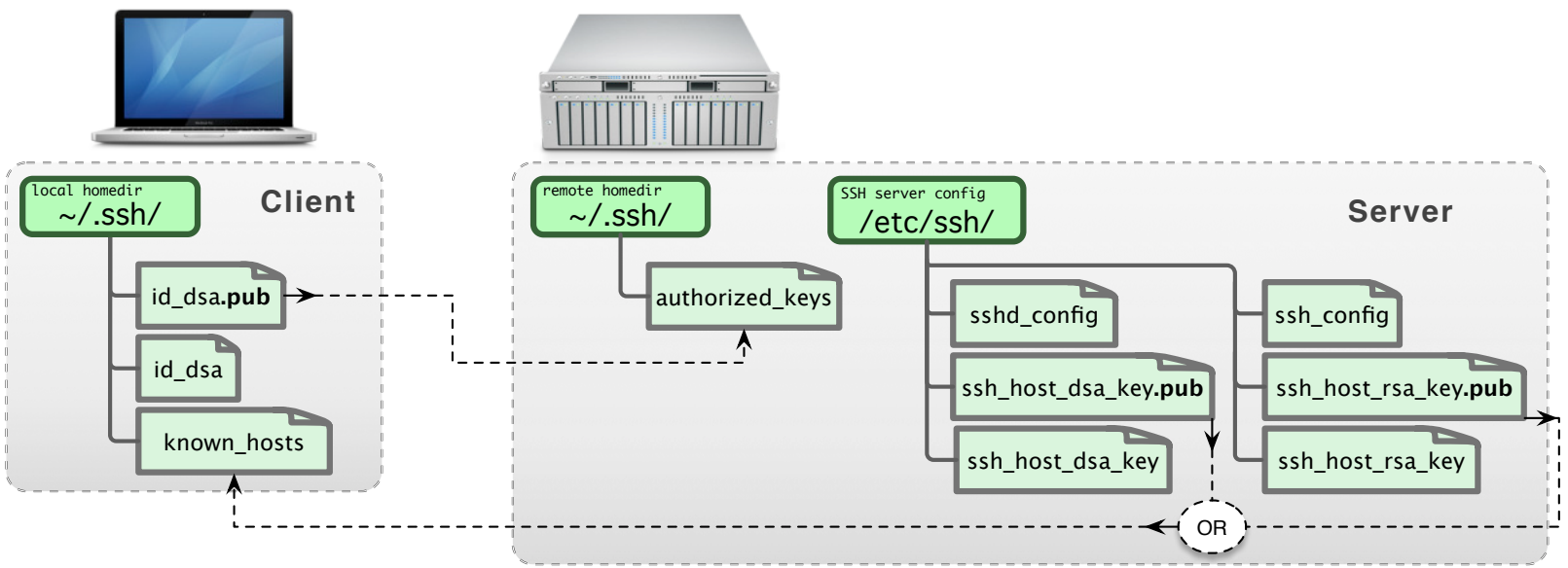
- **Operating System**: Linux Debian (CentOS on storage servers)
- **Remote connection to the platform**: SSH
- **User SSO**: OpenLDAP-based
- **Resource management**: job/batch scheduler: OAR
- **(Automatic) Computing Node Deployment**:
  - ↪ FAI (Fully Automatic Installation)
  - ↪ Puppet
  - ↪ Kadeploy
- **Platform Monitoring**: OAR Monika, OAR Drawgantt, Ganglia, Nagios, Puppet Dashboard etc.
- **Commercial Softwares**:
  - ↪ Intel Cluster Studio XE, TotalView, Allinea DDT, Stata etc.

Summary

Summary

SSH Secure Shell: Overview

- Ensure secure connection to remote (UL) server
- (optional) policy: **restricted** to public key authentication
  - ↪ over non-standard port (8022)
- Rule: 1 machine = 1 key pair (ideally with passphrase protection)



SSH Setup on Linux / Mac OS

- OpenSSH natively supported; configuration directory : `~/.ssh/`
  - ↪ package `openssh-client` (Debian-like) or `ssh` (Redhat-like)

```
$> ssh-keygen -t dsa                    # SSH DSA Key-Pair generation
```

- Public key: `~/.ssh/id_{rsa,dsa}.pub`
  - ↪ **This one is the only one SAFE to distribute.**
- **Private** (identity) key `~/.ssh/id_{rsa,dsa}`
- Configuration: `~/.ssh/config`. Format:

```
Host <shortname>
    Port <port>
    User <login>
    Hostname <hostname>
```

SSH Setup on Windows

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/
  - ↪ PuTTY, the free SSH client
  - ↪ Pageant, an SSH authentication agent for PuTTY tools
  - ↪ PLink, th PuTTy CLI
  - ↪ PuTTYgen, an RSA and DSA key generation utility

SSH Setup on Windows

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/
  - ↪ PuTTY, the free SSH client
  - ↪ Pageant, an SSH authentication agent for PuTTY tools
  - ↪ PLink, th PuTTy CLI
  - ↪ PuTTYgen, an RSA and DSA key generation utility

PuTTY ≠ OpenSSH

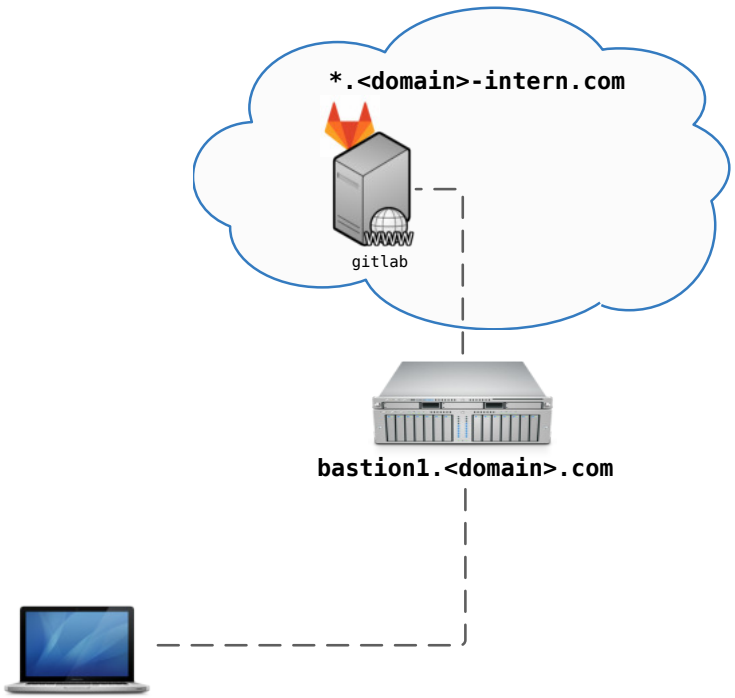
SSH Setup on Windows

- Putty Suite, includes: http://www.chiark.greenend.org.uk/~sgtatham/putty/
  - ↪ PuTTY, the free SSH client
  - ↪ Pageant, an SSH authentication agent for PuTTY tools
  - ↪ PLink, th PuTTy CLI
  - ↪ PuTTYgen, an RSA and DSA key generation utility

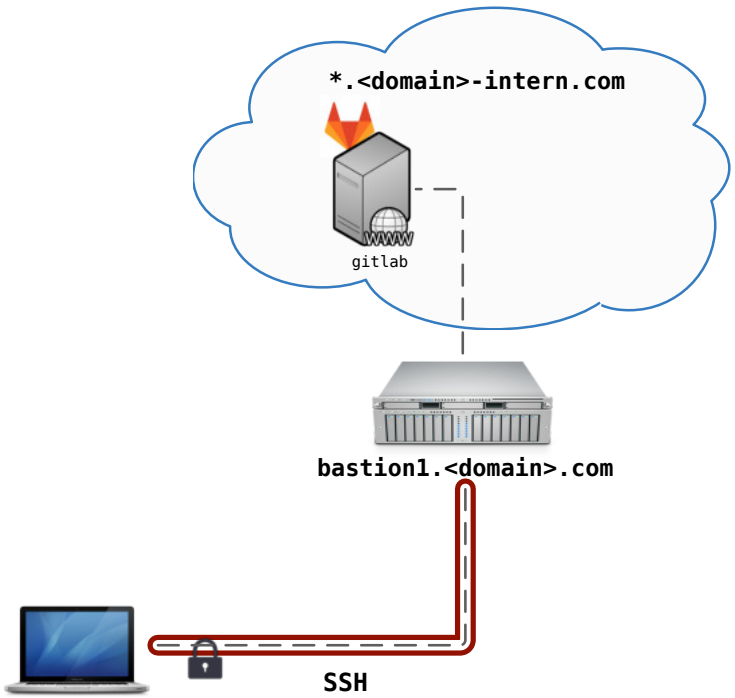
PuTTY ≠ OpenSSH

- Putty keys are **NOT** supported by OpenSSH (yet can be exported)
- Binding Pageant with OpenSSH agent is **NOT** natively supported
  - ↪ Third-party tools like ssh-pageant are made for that
- with PLink, hostnames eventually refer to **PuTTY Sessions**
  - ↪ **NEVER** to SSH entries in `~/.ssh/config`
  - ↪ This usage might be hidden... Ex: `$GIT_SSH` etc.

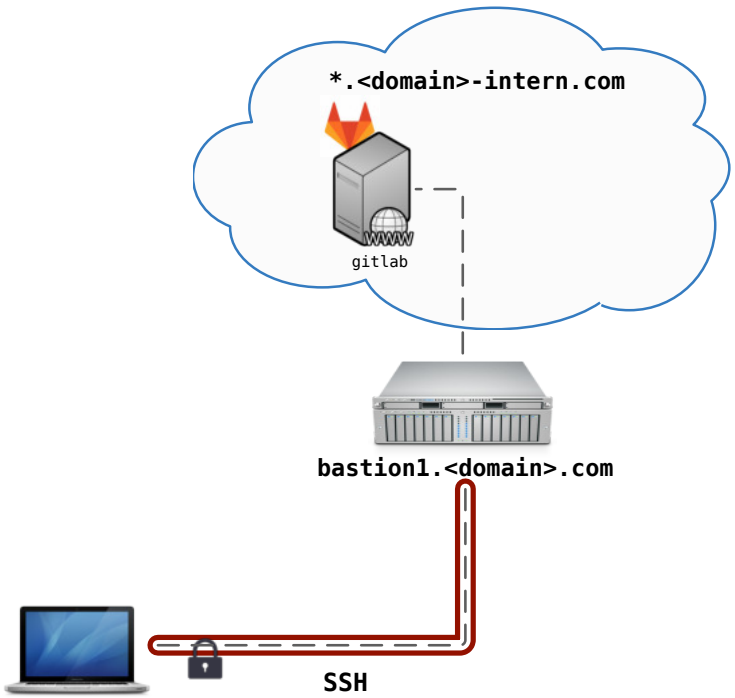
SSH Basic Usage



SSH Basic Usage



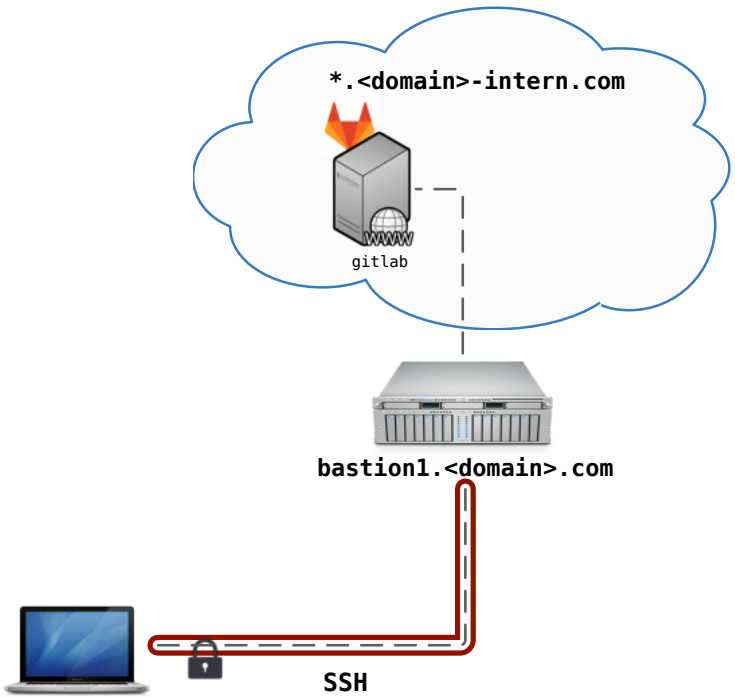
SSH Basic Usage



OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Basic Usage



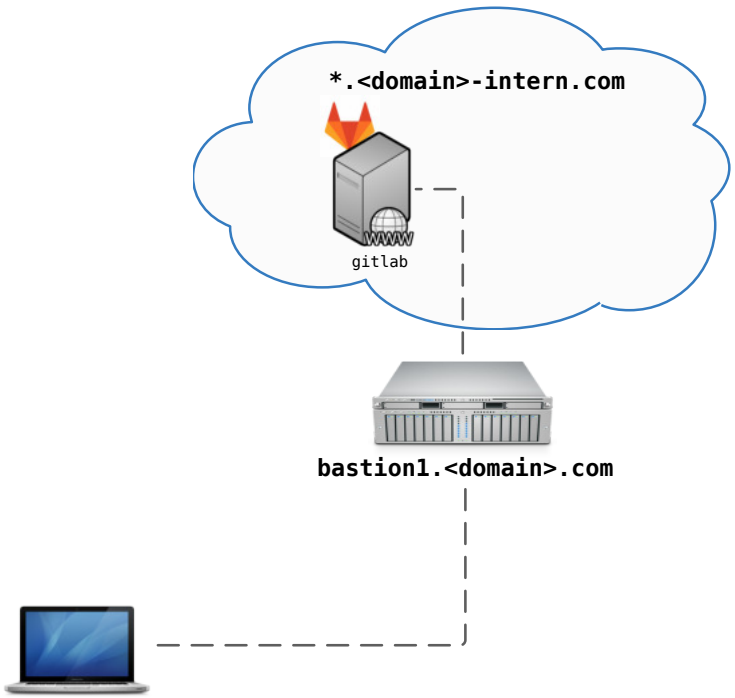
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

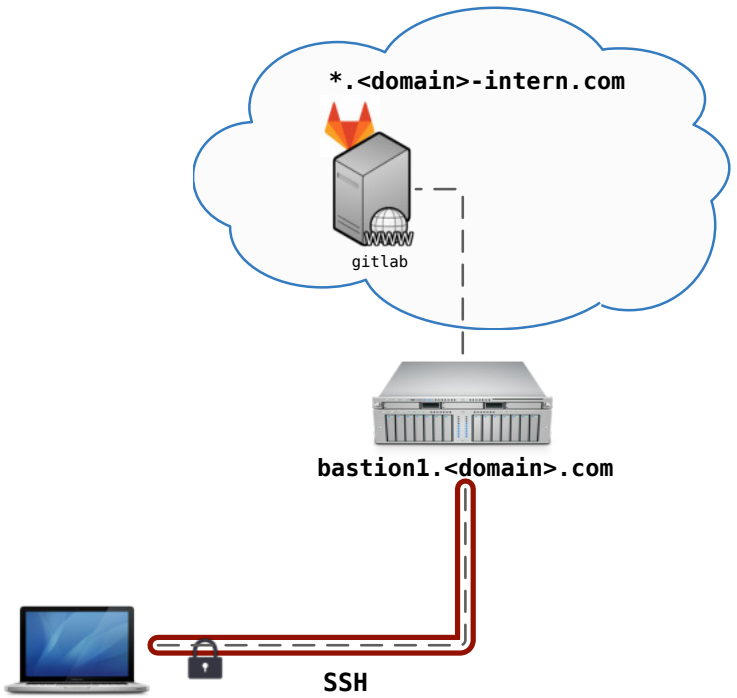
OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



SSH Advanced Usage: SOCKS Proxy



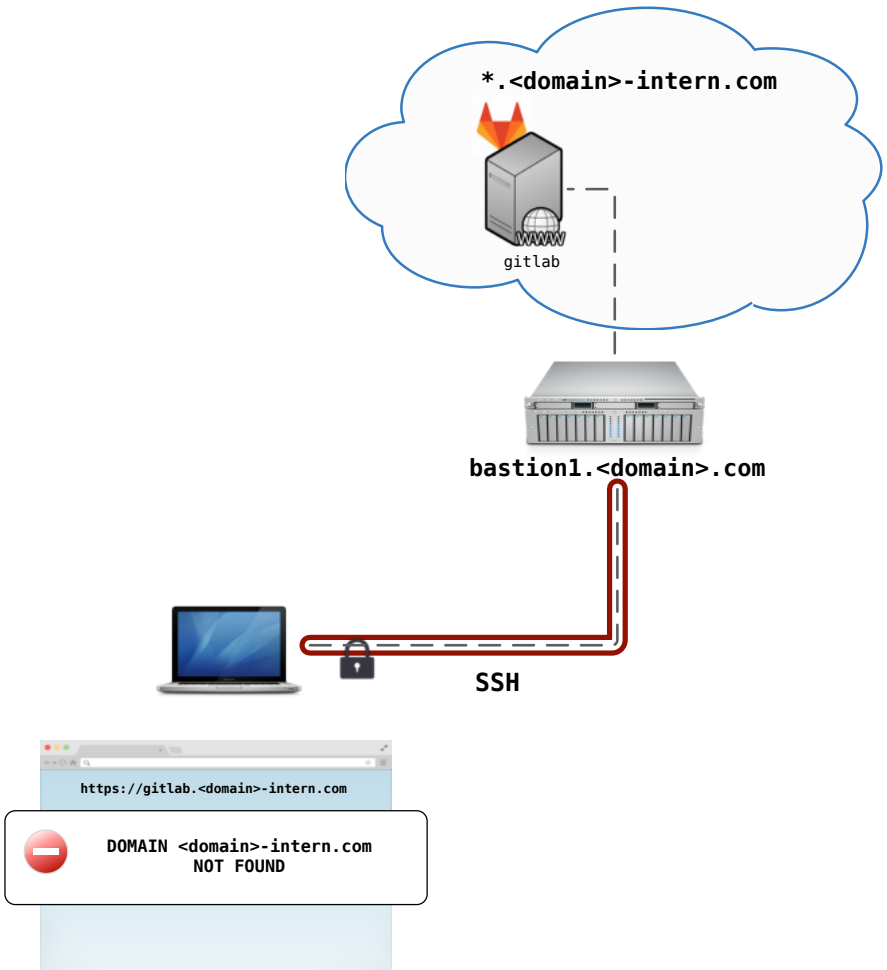
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



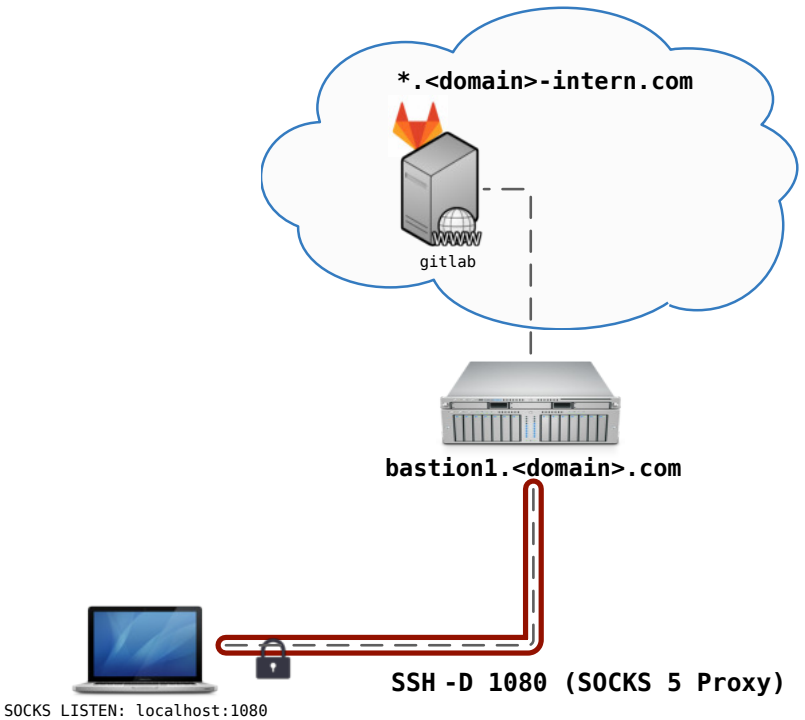
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



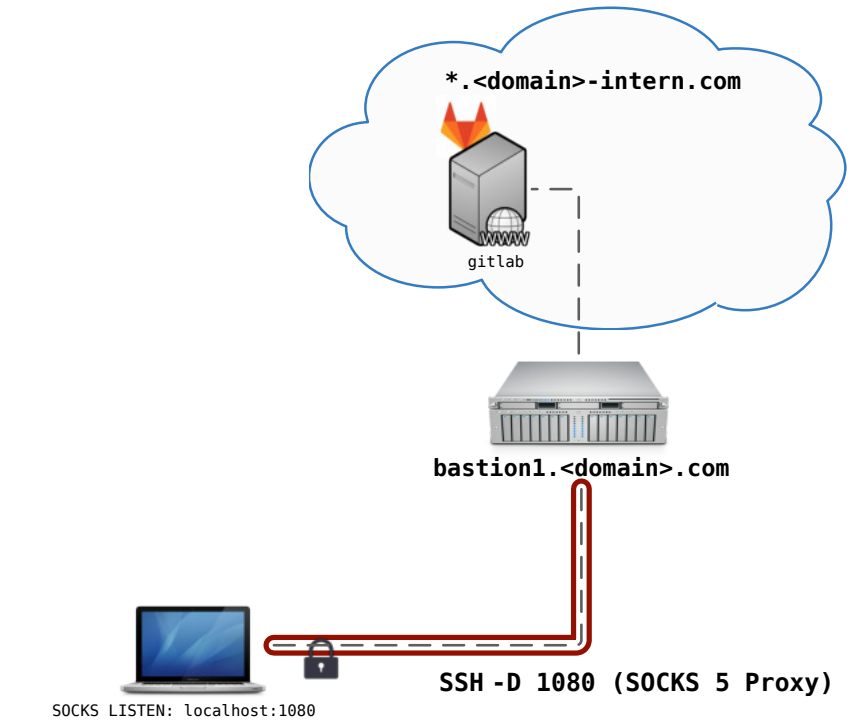
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



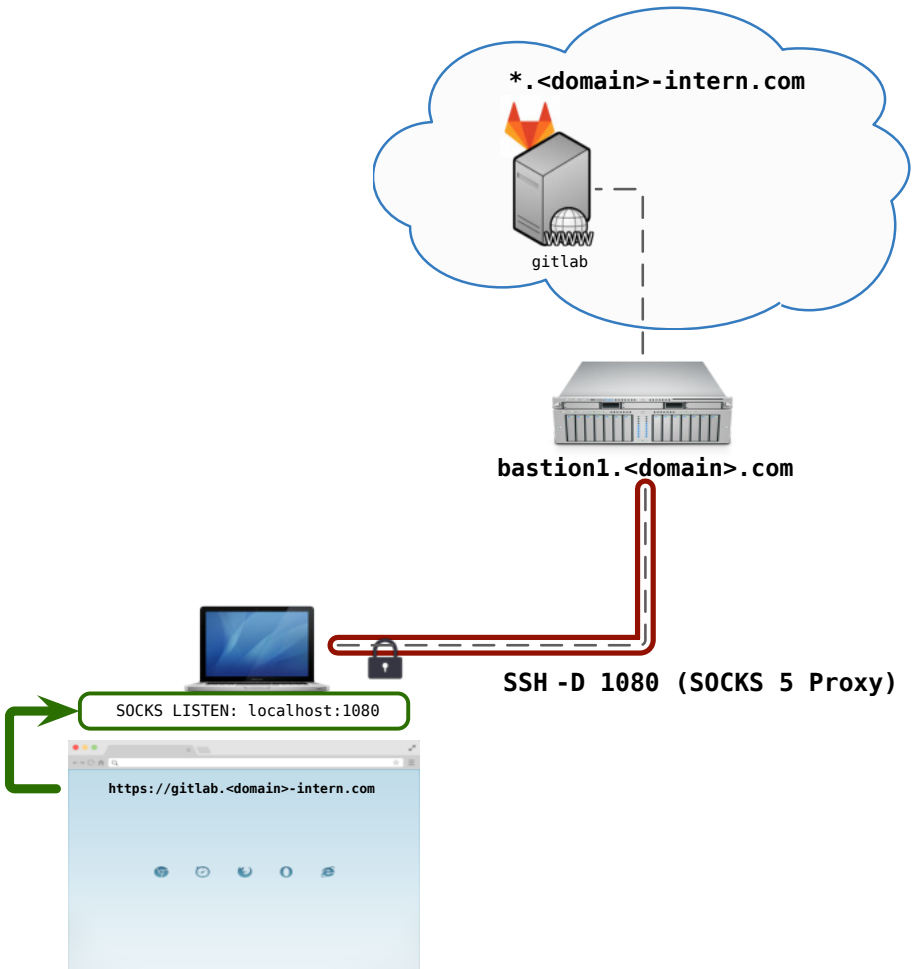
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



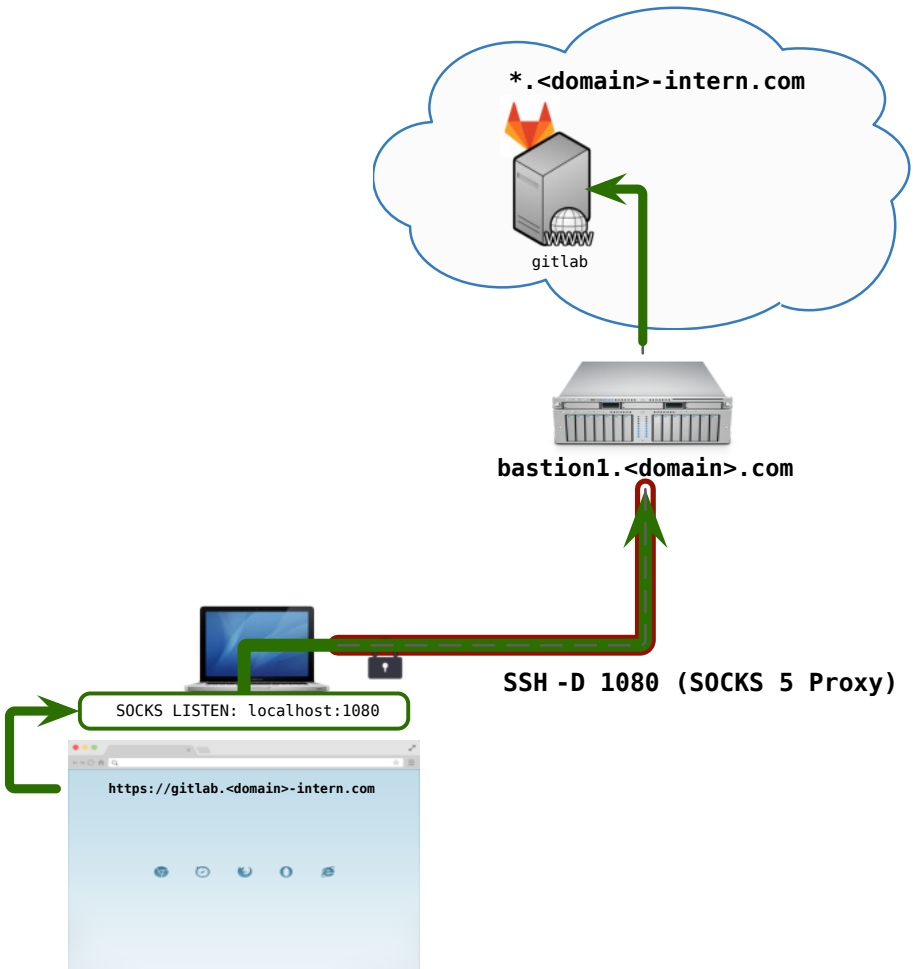
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



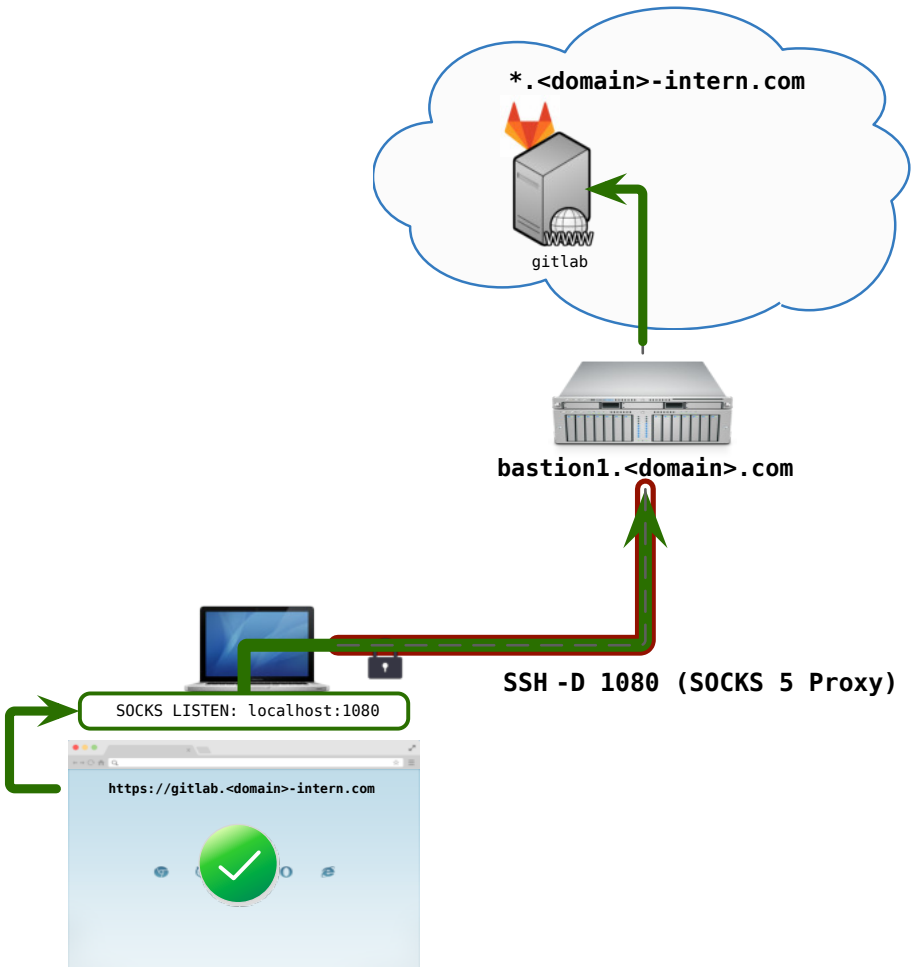
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: SOCKS Proxy



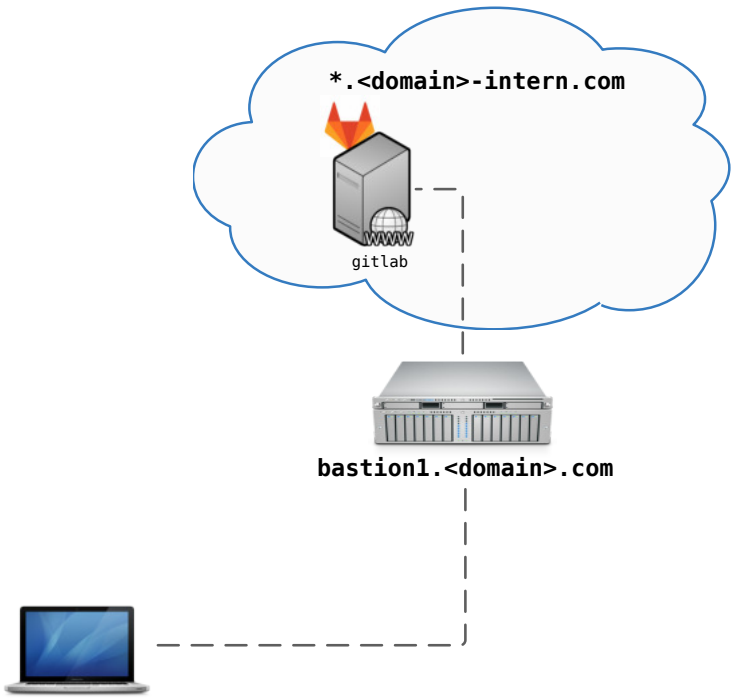
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/SSH/Tunnels: Port 1080, Dynamic
```

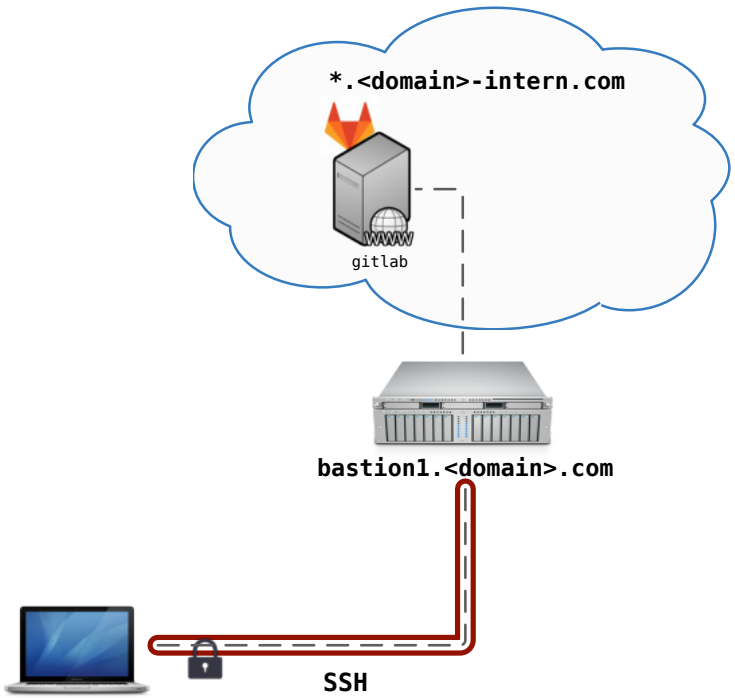
OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: ProxyCommand



SSH Advanced Usage: ProxyCommand



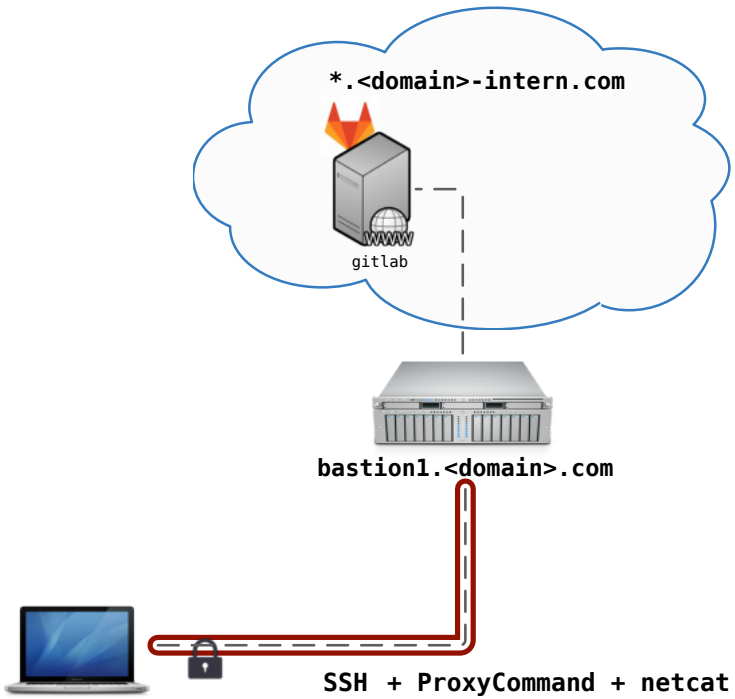
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com
```

SSH Advanced Usage: ProxyCommand



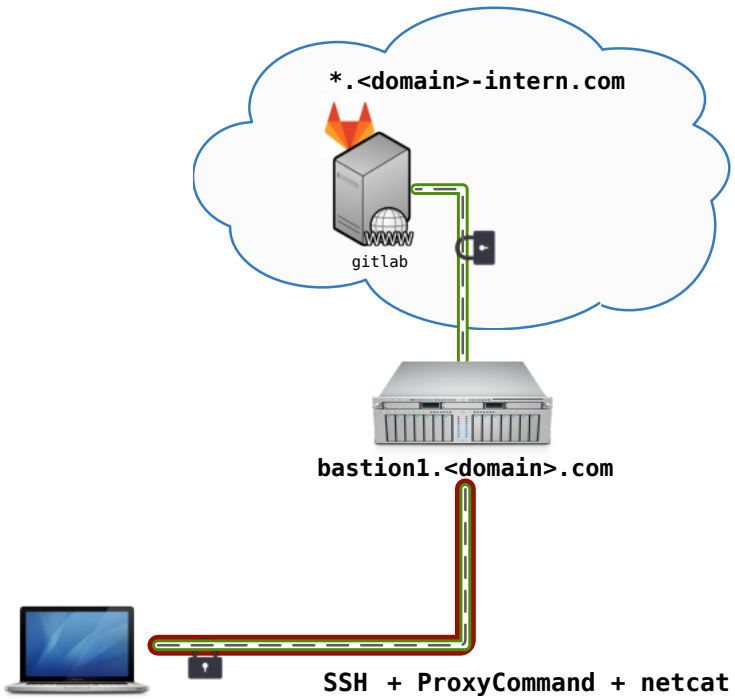
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>

Session "uni_gitlab"
- Hostname: gitlab.<domain>-intern.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/Proxy:
  - type: local
  - Proxy hostname: bastion1.<domain>.com
  - Port: 8022
  - Username: <LOGIN>
  - Local proxy command:
    plink -load "uni_bastion1" -nc %host:%port
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com

Host uni_gitlab
   Hostname gitlab
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

SSH Advanced Usage: ProxyCommand



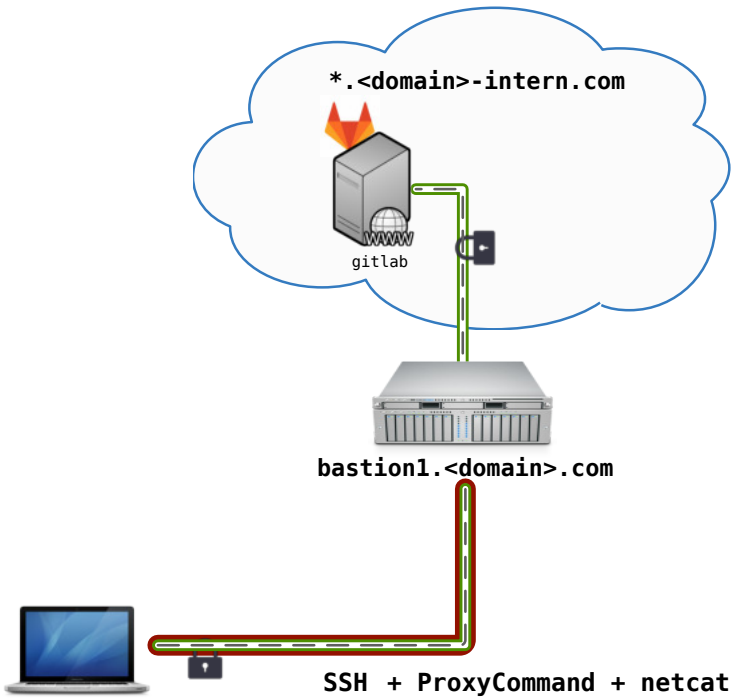
PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>

Session "uni_gitlab"
- Hostname: gitlab.<domain>-intern.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/Proxy:
  - type: local
  - Proxy hostname: bastion1.<domain>.com
  - Port: 8022
  - Username: <LOGIN>
  - Local proxy command:
    plink -load "uni_bastion1" -nc %host:%port
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com

Host uni_gitlab
   Hostname gitlab
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

SSH Advanced Usage: ProxyCommand



PuTTY / PLink / Pageant (Windows)

```
Session "uni_bastion1"
- Hostname: bastion1.<domain>.com
- Port: 8022
- Connection/Data: username: <LOGIN>

Session "uni_gitlab"
- Hostname: gitlab.<domain>-intern.com
- Port: 8022
- Connection/Data: username: <LOGIN>
- Connection/Proxy:
  - type: local
  - Proxy hostname: bastion1.<domain>.com
  - Port: 8022
  - Username: <LOGIN>
  - Local proxy command:
    plink -load "uni_bastion1" -nc %host:%port
```

OpenSSH ~/.ssh/config (Mac / Linux)

```
Host uni_*
    User <LOGIN>
    Port 8022

Host uni_bastion1
   Hostname bastion1.<domain>.com

Host uni_gitlab
   Hostname gitlab
   ProxyCommand ssh -q uni_bastion1 "nc %h %p"
```

DSH – Distributed / Dancer's Shell

http://www.netfort.gr.jp/~dancer/software/dsh.html.en

- SSH wrapper that allows to run commands over multiple machines.
  - ↪ Linux / Mac OS **only**

```
$> { apt-get | yum | brew } install dsh          # Installation
```

- **Configuration**: in `~/.dsh/`
  - ↪ `~/.dsh/dsh.conf`: main configuration file
  - ↪ `~/.dsh/machines.list`: list of **all** nodes
  - ↪ `~/.dsh/group/`: holds group definition
- `<name>` **Group** definition: `~/.dsh/group/<name>`:
  - ↪ simply list **SSH** shortnames (one name by line)
- Bash completion file for DSH: https://gist.github.com/920433.git

DSH configuration ~/.dsh/dsh.conf

```
##########################################################
# ~/.dsh/dsh.conf
# Configuration file for dsh (Distributed / Dancer's Shell).
# 'man dsh.conf' for details
##########################################################
verbose = 0

remoteshell      = ssh
showmachinenames = 1

# Specify 1 to make the shell wait for each individual invocation.
#    See -c and -w option for dsh(1)
waitshell        = 0   # whether to wait for execution

# Number of parallel connection to create at the same time.
#forklimit=8

remoteshellopt   = -q
```

DSH Basic Usage

```
$> dsh [-c | -w] { -a | -g <group> | -m <hostname> } <command>
```

Option	Description
`-c`	run the commands in parallel (default)
`-w`	run the commands in sequential
`-a`	run the command on all nodes listed in `machines.list`
`-g <group>`	restrict the commands to the hosts group `<group>`
`-m <hostname>`	run the command only on `hostname`

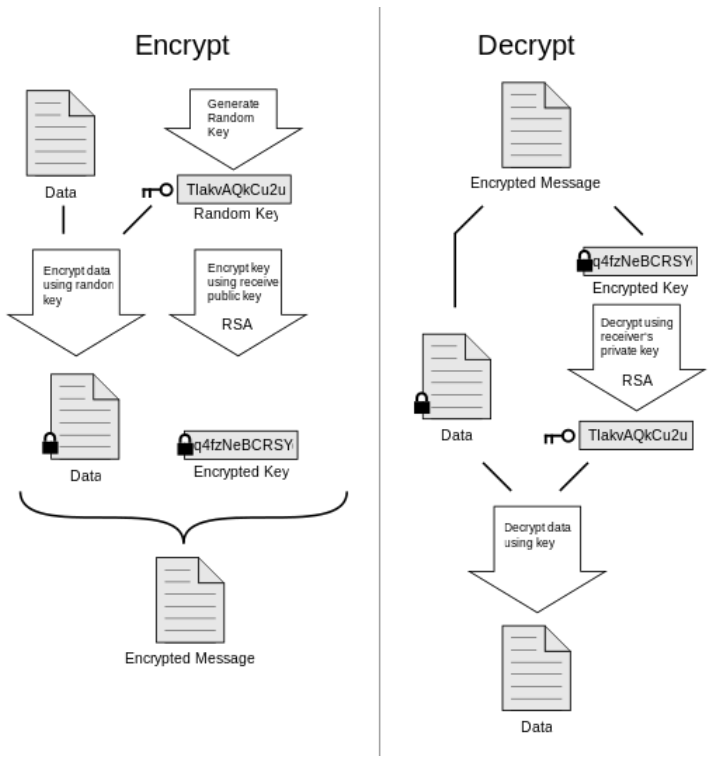
- **FAQ**: `sudo: sorry, you must have a tty to run sudo`
  - ↪ requires to change the default configuration of `sudo`
  - ↪ Ex to **not** requiring a tty to launch a sudo command

```
Defaults:<login> !requiretty
```

Summary

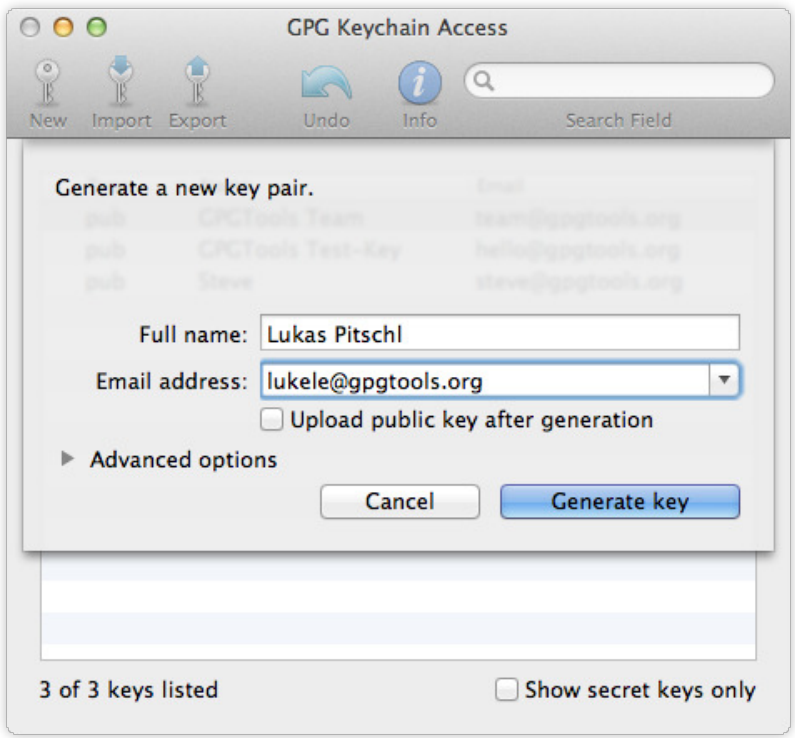
GPG: Gnu Privacy Guard

- GnuPG: implementation of the OpenPGP standard aka RFC4880
  - ↪ Hybrid encryption framework based on Web of Trust
  - ↪ `Mail | Document | Git commit...` encryption / signature



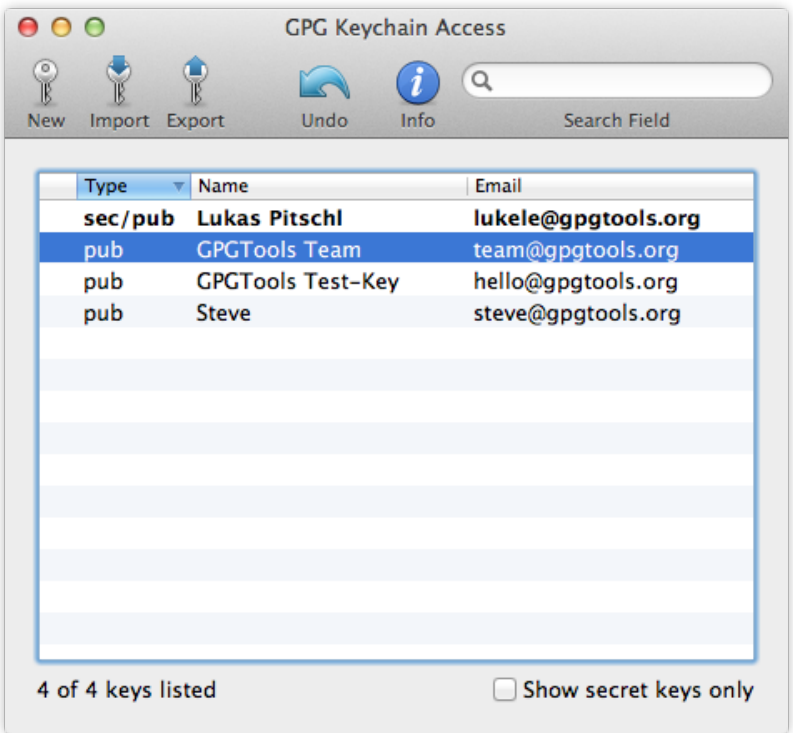
GPG Setup (Mac OS)

- GPGTools Suite
  - ↪ GPG for Apple Mail and GPG Keychain
  - ↪ GPG Services and MacGPG



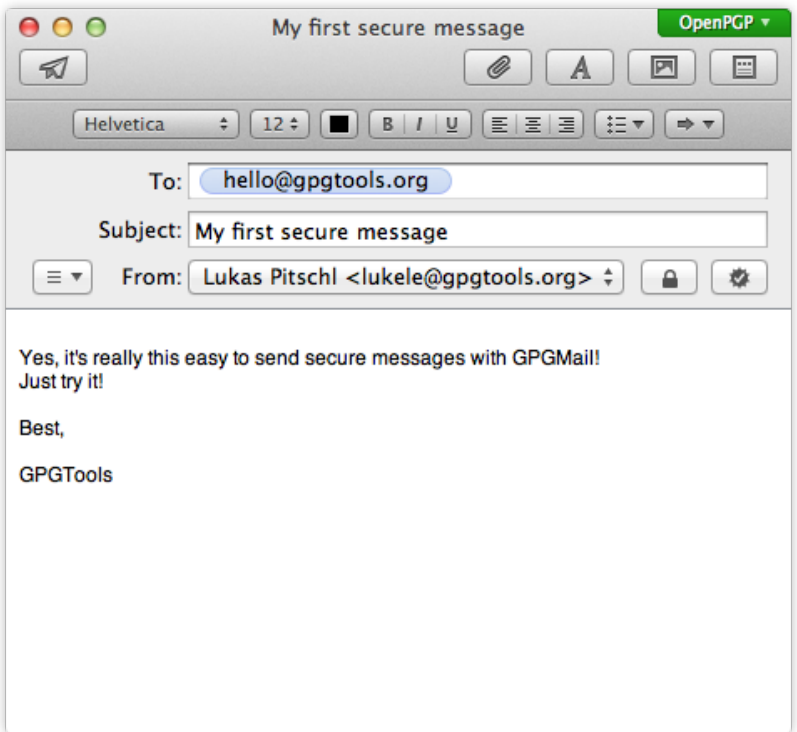
GPG Setup (Mac OS)

- GPGTools Suite
  - ↪ GPG for Apple Mail and GPG Keychain
  - ↪ GPG Services and MacGPG



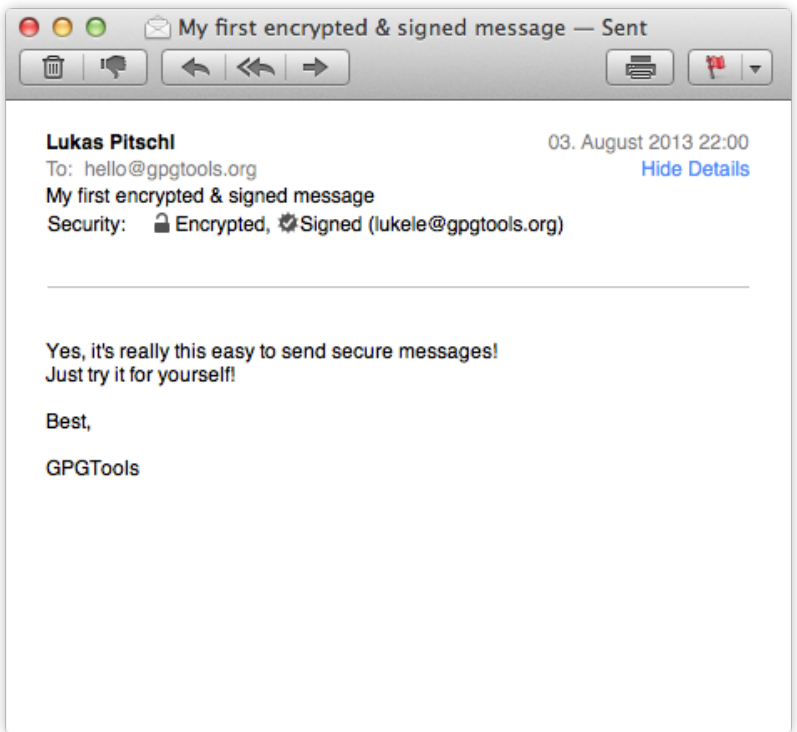
GPG Setup (Mac OS)

- GPGTools Suite
  - ↪ GPG for Apple Mail and GPG Keychain
  - ↪ GPG Services and MacGPG



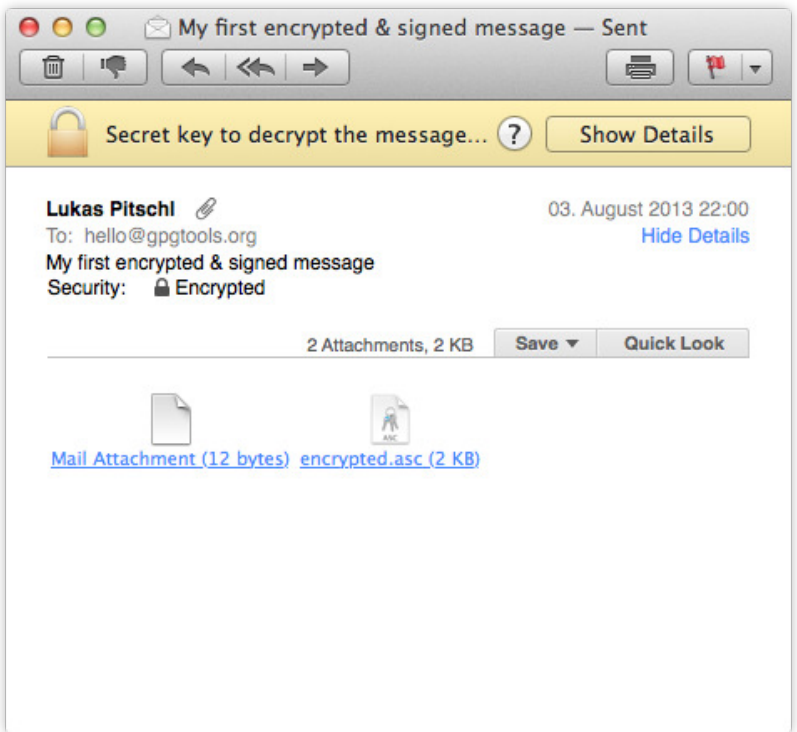
GPG Setup (Mac OS)

- GPGTools Suite
  - ↪ GPG for Apple Mail and GPG Keychain
  - ↪ GPG Services and MacGPG



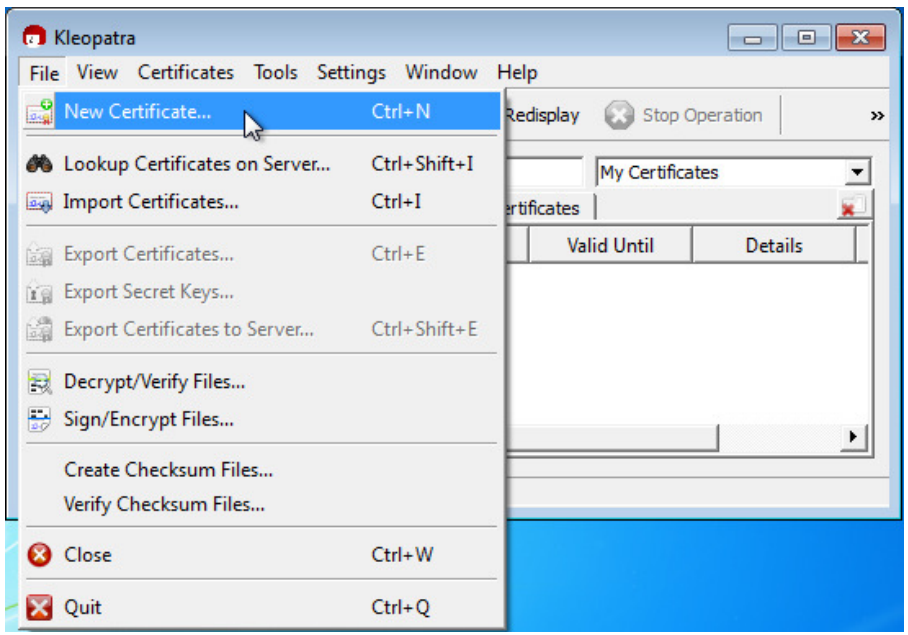
GPG Setup (Mac OS)

- GPGTools Suite
  - ↪ GPG for Apple Mail and GPG Keychain
  - ↪ GPG Services and MacGPG



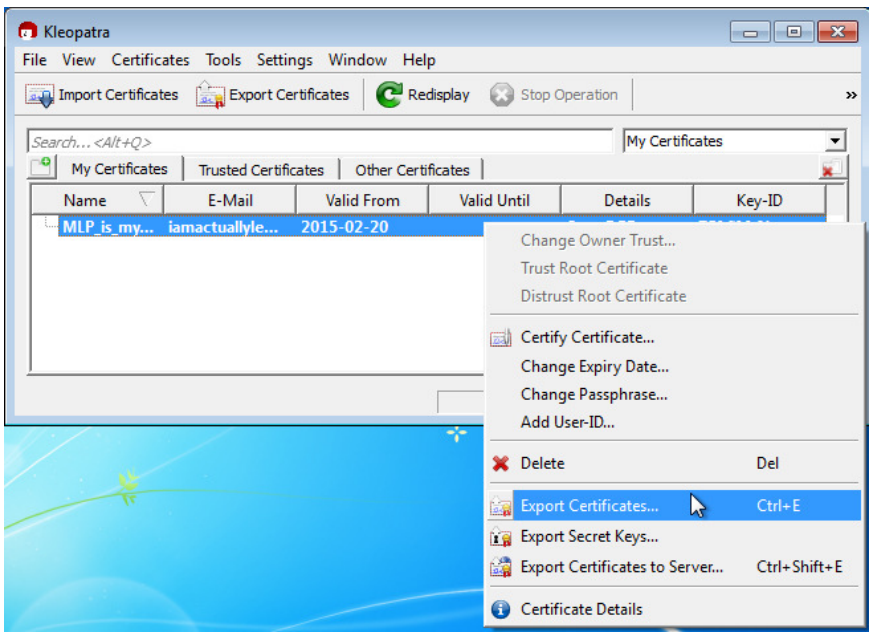
GPG Setup (Windows)

- GPG4Win – Tutorial
  - ↪ GnuPG, GnuPG for Outlook (GpgOL)
  - ↪ **Kleopatra + GNU Privacy Assistant (GPA) (to be checked)**
  - ↪ GPG Explorer eXtension (GpgEX)
- **[All OS]** Thunderbird + Enigmail



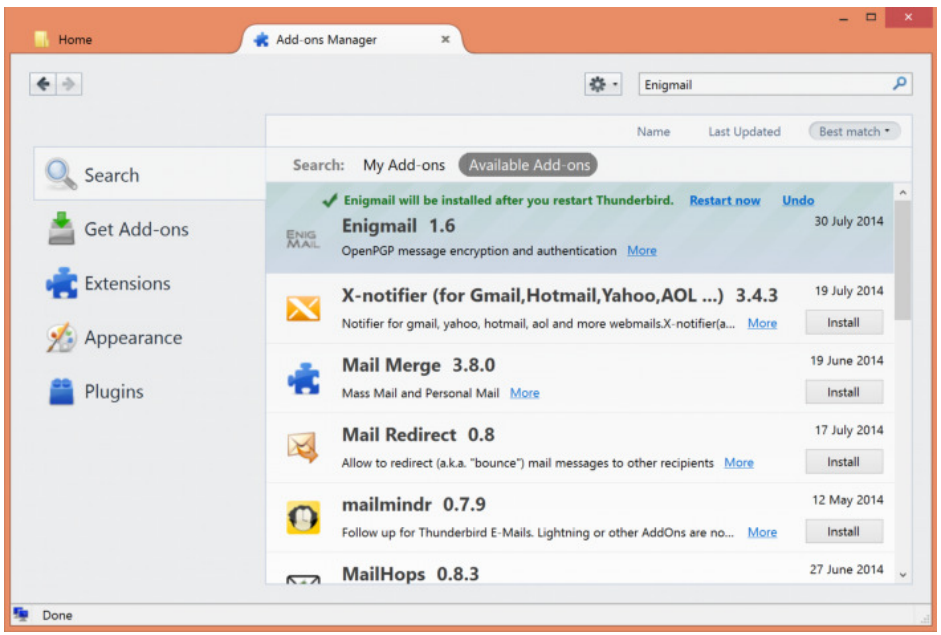
GPG Setup (Windows)

- GPG4Win – Tutorial
  - ↪ GnuPG, GnuPG for Outlook (GpgOL)
  - ↪ **Kleopatra + GNU Privacy Assistant (GPA) (to be checked)**
  - ↪ GPG Explorer eXtension (GpgEX)
- **[All OS]** Thunderbird + Enigmail



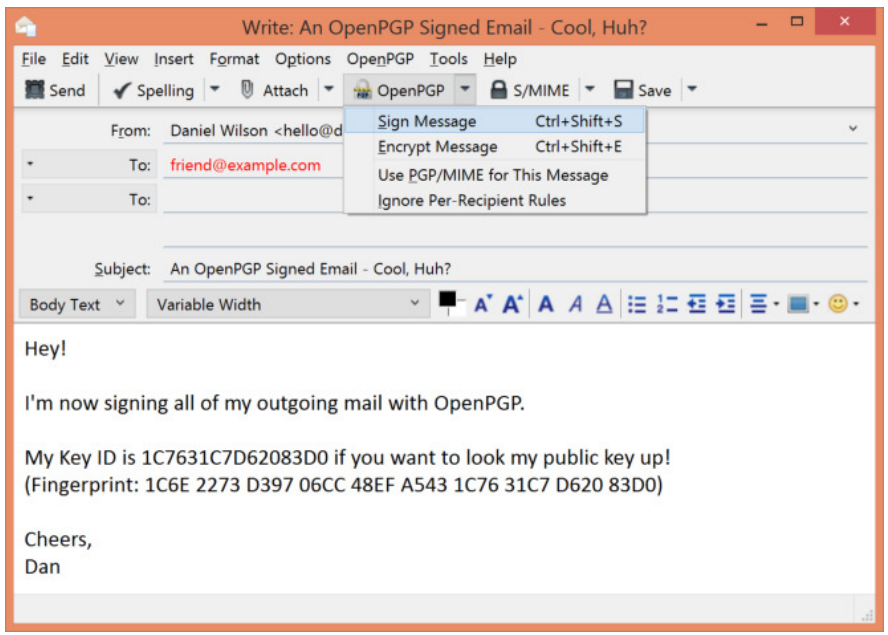
GPG Setup (Windows)

- GPG4Win – Tutorial
  - ↪ GnuPG, GnuPG for Outlook (GpgOL)
  - ↪ **Kleopatra + GNU Privacy Assistant (GPA) (to be checked)**
  - ↪ GPG Explorer eXtension (GpgEX)
- **[All OS]** Thunderbird + Enigmail



GPG Setup (Windows)

- GPG4Win – Tutorial
  - ↪ GnuPG, GnuPG for Outlook (GpgOL)
  - ↪ **Kleopatra + GNU Privacy Assistant (GPA) (to be checked)**
  - ↪ GPG Explorer eXtension (GpgEX)
- **[All OS]** Thunderbird + Enigmail



GPG CLI Usage

```
$> gpg --gen-key                                          # Generate your PGP key
$> gpg --list-keys [pattern]                            # List available PGP key(s)
$> gpg --keyserver pgp.mit.edu --search-keys <pattern> # Search & Import
$> gpg --keyserver pgp.mit.edu --recv-keys <ID>                    # Import
```

GPG CLI Usage

```
$> gpg --gen-key                                          # Generate your PGP key
$> gpg --list-keys [pattern]                            # List available PGP key(s)
$> gpg --keyserver pgp.mit.edu --search-keys <pattern> # Search & Import
$> gpg --keyserver pgp.mit.edu --recv-keys <ID>                    # Import
```

- Send **encrypted** mails to `user@domain.org` ⇔ you **trust** his key
  - ↪ *i.e.* **sign** (after careful check) this key (using GPG Keychain / GPA)

GPG CLI Usage

```
$> gpg --gen-key                                        # Generate your PGP key
$> gpg --list-keys [pattern]                          # List available PGP key(s)
$> gpg --keyserver pgp.mit.edu --search-keys <pattern># Search & Import
$> gpg --keyserver pgp.mit.edu --recv-keys <ID>                  # Import
```

- Send **encrypted** mails to user@domain.org ⇔ you **trust** his key
  ↪ *i.e.* **sign** (after careful check) this key (using GPG Keychain / GPA)

```
$> gpg [-K] --fingerprint <mail>     # Get (with -K) / Check fingerprint
$> gpg --sign-key --ask-cert-level <ID>      # Sign Key <ID> AFTER check
$> gpg --keyserver pgp.mit.edu --send-keys <ID>  # Send back signed key
```

GPG CLI Usage

```
$> gpg --encrypt [-r <recipient>] <file>        # => <file>.gpg
```

- WARNING: encryption does not delete the input (clear-text) file

GPG CLI Usage

```
$> gpg --encrypt [-r <recipient>] <file>        # => <file>.gpg
```

- WARNING: encryption does not delete the input (clear-text) file

```
$> gpg --decrypt <file>.gpg        # Decrypt PGP encrypted file
```

GPG CLI Usage

```
$> gpg --encrypt [-r <recipient>] <file>        # => <file>.gpg
```

- WARNING: encryption does not delete the input (clear-text) file

```
$> gpg --decrypt <file>.gpg        # Decrypt PGP encrypted file
```

```
$> gpg --armor --detach-sign <file>  # Do signature <file>.asc
```

GPG CLI Usage

```
$> gpg --encrypt [-r <recipient>] <file>        # => <file>.gpg
```

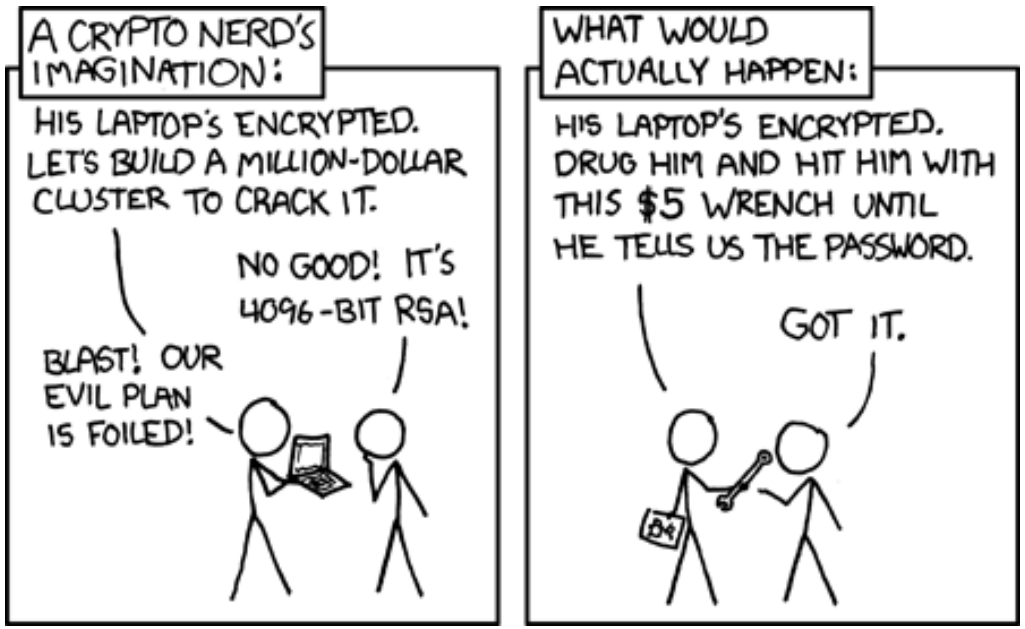
- WARNING: encryption does not delete the input (clear-text) file

```
$> gpg --decrypt <file>.gpg        # Decrypt PGP encrypted file
```

```
$> gpg --armor --detach-sign <file>  # Do signature <file>.asc
```

- **GPG Keychain / Keyring**
  - ↪ Linux / Mac OS: `~/.gnupg/`
  - ↪ Windows: `C:\\Documents and Settings\<LOGIN>\Application Data\gnupg\`

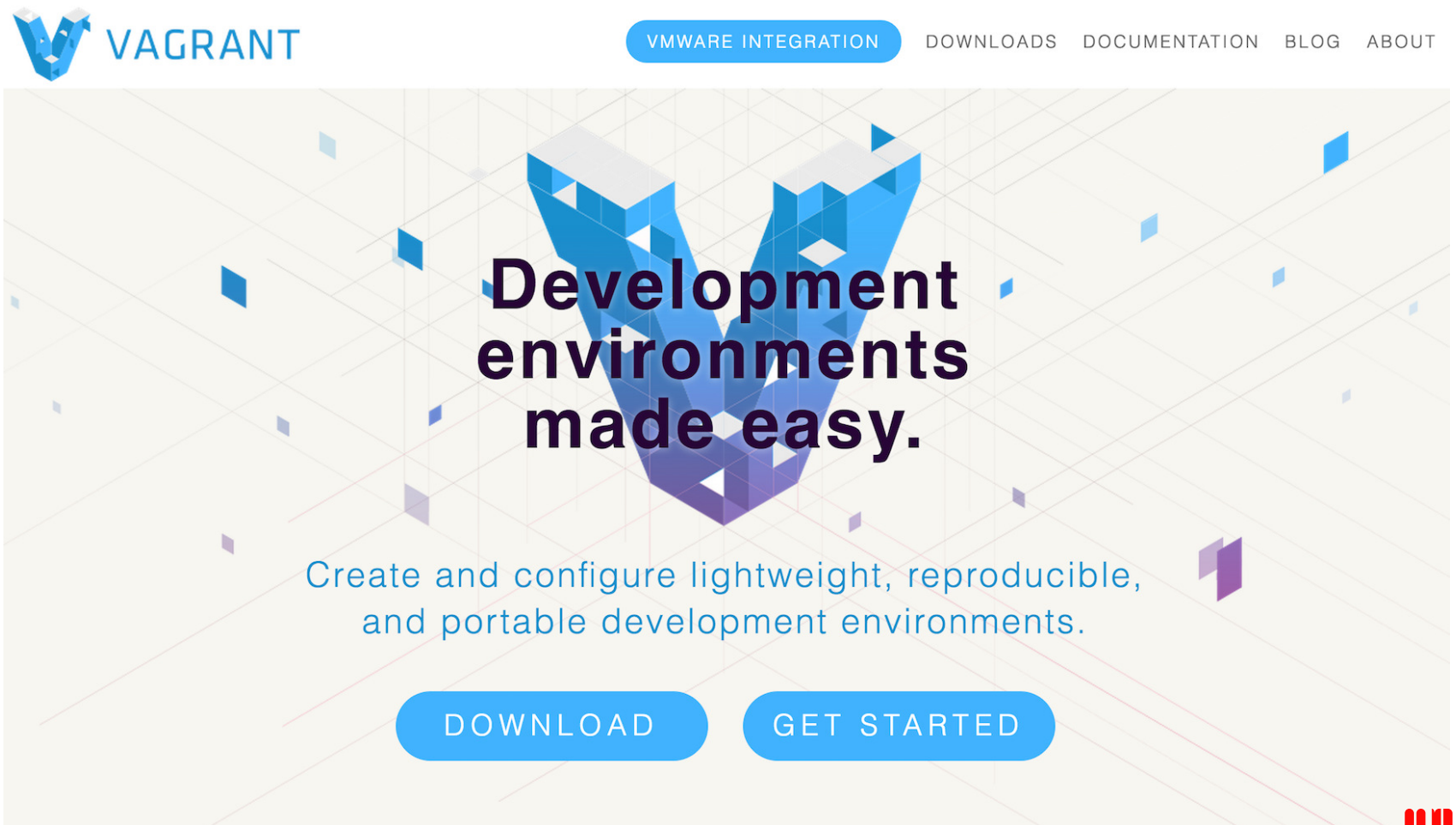
Recall: Security = Noble Goal, yet...



Summary

What is Vagrant ?

http://vagrantup.com/



What is Vagrant ?

Create and configure **lightweight**, **reproducible**, and **portable** development environments

- Command line tool
- Automates VM creation with
  - ↪ VirtualBox
  - ↪ VMWare etc.
- Integrates well with configuration management tools
  - ↪ Shell
  - ↪ Puppet etc.

Runs on Linux, Windows, MacOS

Why use Vagrant?

- Create new VMs quickly and easily: only one command!

```
$> vagrant up
```

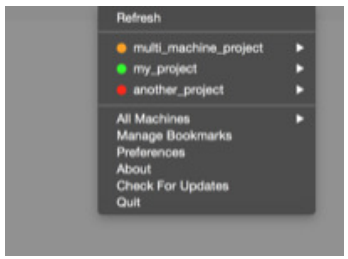
- Keep the number of VMs under control
  - ↪ All configuration in `VagrantFile`
- Reproducability
  - ↪ Identical environment in development and production
- Portability
  - ↪ **avoid** sharing 4 GB VM disks images
  - ↪ Vagrant Cloud to share your images
- Collaboration made easy:

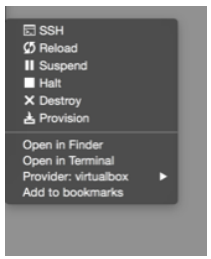
```
$> git clone ...
$> vagrant up
```

Installation Notes: Mac OS

- Best done using Homebrew and Cask

```
$> brew install caskroom/cask/brew-cask
$> brew cask install virtualbox    # install virtualbox
$> brew cask install vagrant
$> brew cask install vagrant-manager # see http://vagrantmanager.com/
```





Installation Notes: Windows / Linux

- Install Oracle Virtualbox
- Go on the Download Page
  - ↪ select the appropriate OS, in 64 bits versions

- **Notes for windows users**:
  - ↪ you will also need both PuTTY and PuTTYGen
  - ↪ Vagrant boxes are located in `%userprofile%/.vagrant.d/boxes`
  - ↪ To configure the appropriate Putty profile:
    - ✓ run `vagrant ssh-config` to collect IP and port (after `vagrant up`)
    - ✓ load `%userprofile%/.vagrant.d/insecure_public_key`
    - ✓ Use `Save Public Key`to convert the OpenSSH key to PPK format
    - ✓ Create the PuttY profile accordingly (username: `vagrant`)

Minimal default setup

```
$> vagrant init [-m] <user>/<name> # setup vagrant cloud image
```

- A `Vagrantfile` is configured

Minimal default setup

```
$> vagrant init [-m] <user>/<name> # setup vagrant cloud image
```

- A `Vagrantfile` is configured

```
$> vagrant up          # boot the box(es) set in the Vagrantfile
```

- The base box is downloaded and stored locally
  ↪ in `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template
- The VM is booted and (eventually) provisioned

Minimal default setup

```
$> vagrant init [-m] <user>/<name> # setup vagrant cloud image
```

- A `Vagrantfile` is configured

```
$> vagrant up          # boot the box(es) set in the Vagrantfile
```

- The base box is downloaded and stored locally
  ↪ in `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template
- The VM is booted and (eventually) provisioned

```
$> vagrant ssh                                  # connect inside it
```

Find a vagrant box

- Vagrant Cloud `https://vagrantcloud.com/`
- VagrantBox.es `http://www.vagrantbox.es/`

Find a vagrant box

- Vagrant Cloud — https://vagrantcloud.com/
- VagrantBox.es — http://www.vagrantbox.es/

Your Turn!

```
$> vagrant init ubuntu/trusty64  # Ubuntu Server 14.04 LTS
$> vagrant up
$> vagrant ssh
```

Box name	Description
ubuntu/trusty64	Ubuntu Server 14.04 LTS
centos/7	CentOS Linux 7 x86_64
debian/jessie64	Vanilla Debian 8 "Jessie"
jhcook/osx-elcapitan-10.11	OS X 10.11 El Capitan

- Once within the box:
  - ↪ `/vagrant`: root directory hosting `Vagrantfile`

Configuring Vagrant

- Minimal `Vagrantfile` (Ruby syntax)

```
VAGRANTFILE_API_VERSION = '2'

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
   config.vm.box = 'svarrette/centos-7-puppet'
   config.ssh.insert_key = false
end
```

- Configure Multiple box within the **same** `Vagrantfile`
  - ↪ See `ULHPC/puppet-sysadmins/Vagrantfile`

Vagrant Box Status / Stop

```
$> vagrant status                    # State of the vagrant box(es)
```

Vagrant Box Status / Stop

```
$> vagrant status                    # State of the vagrant box(es)
```

```
$> vagrant { destroy | halt }                    # destroy / halt
```

- Once you have finished your work within a *running* box
  - ↪ save the state for later with `vagrant halt`
  - ↪ reset changes / tests / errors with `vagrant destroy`
  - ↪ commit changes by generating a new version of the box

Vagrant Box Generation

- You might rely on Falkor/vagrant-vms
  - ↪ use it at your own risks
  - ↪ based on packer and veewee

```
$> git clone https://github.com/Falkor/vagrant-vms.git
$> cd vagrant-vms
$> gem install bundler && bundle install
$> rake setup
```

Vagrant Box Generation

- You might rely on Falkor/vagrant-vms
  - ↪ use it at your own risks
  - ↪ based on packer and veewee

```
$> git clone https://github.com/Falkor/vagrant-vms.git
$> cd vagrant-vms
$> gem install bundler && bundle install
$> rake setup
```

```
# initiate a template for a given Operating System:
$> rake packer:{Debian,CentOS,openSUSE,scientificlinux,ubuntu}:init
# Build a Vagrant box
$> rake packer:{Debian,CentOS,openSUSE,scientificlinux,ubuntu}:build
# If things goes fine:
$> vagrant box add packer/<os>-<version>-<arch>/<os>-<version>-<arch>.box
```

Vagrant Box Customization

- **Obj**: customize / specialize the configuration of a **running** box
- This can be done in two ways:
  1. use **provisionning** within the `Vagrantfile` (using puppet etc.)
  2. re-package the box via `vagrant package`

```
# (1) Vagrantfile with Puppet provisioning
Vagrant.configure(2) do |config|
  config.vm.box = 'svarrette/centos-7-puppet'
  config.vm.provision :puppet  do |puppet|
    puppet.hiera_config_path = 'hieradata/hiera.yaml'
    puppet.working_directory = '/vagrant'
    puppet.manifests_path    = "manifests"
    puppet.module_path       = "modules"
    puppet.manifest_file     = "init.pp"
    puppet.options = [ '-v','--report','--show_diff','--pluginsync' ]
  end
end
```

Box Re-packaging (1/2)

- WARNING: ensure you **DO NOT** reset the (insecure) SSH key
  - ↪ **before** `vagrant up`, use the following `Vagrantfile` configuration:
    `config.ssh.insert_key = false`
- Zero out the free space to save space – run the following script:

```
$> dd if=/dev/zero of=/EMPTY bs=1M
$> rm -f /EMPTY
```

- Ensure Virtualbox Guest additions match using the vbguest plugin

```
$> vagrant plugin install vagrant-vbguest
$> vagrant vbguest --status
GuestAdditions versions on your host (5.0.4) & guest (4.3.26) mismatch
# Upgrade the GuestAdditions
$> vagrant vbguest --do install --auto-reboot
```

Box Re-packaging (2/2)

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"vagrant-vms_default_1431034026308_70455" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

Box Re-packaging (2/2)

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"vagrant-vms_default_1431034026308_70455" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

- Now you can upload the generated box on Vagrant Cloud.
  - ↪ select 'New version', enter the new version number
  - ↪ add a new box provider (Virtualbox)
  - ↪ upload the generated box

Box Re-packaging (2/2)

```
# Locate the internal name of the running VM and repackage it
$> VBoxManage list runningvms
"vagrant-vms_default_1431034026308_70455" {...}
$> vagrant package \
    --base vagrant-vms_default_1431034026308_70455 \
    --output <os>-<version>-<arch>.box
```

- Now you can upload the generated box on Vagrant Cloud.
  - ↪ select '`New version`', enter the new version number
  - ↪ add a new box provider (`Virtualbox`)
  - ↪ upload the generated box
- Upon successful upload: **release** the uploaded box
  - ↪ by default it is unreleased
  - ↪ Now people using the `<user>/<name>` box will be notified of a pending update

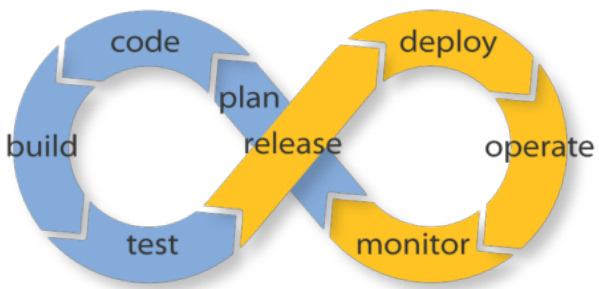
Summary

IT Serv[er|ice] Management: Puppet

Server/Service configuration by Puppet

http://puppetlabs.com

- **IT Automation** for configuration management
  - ↪ idempotent
  - ↪ agent/master OR stand-alone architecture
  - ↪ cross-platform through Puppet's Resource Abstraction Layer (RAL)
  - ↪ Git-based workflow
  - ↪ PKI-based security (X.509)
- DevOps tool of choice for configuration management
  - ↪ Declarative Domain Specific Language (DSL)



Endless Possibilities: DevOps can create an infinite loop of release and feedback for all your code and deployment targets.

IT Serv[er|ice] Management: Puppet

Server/Service configuration by Puppet

http://puppetlabs.com

- **IT Automation** for configuration management
  - ↪ idempotent
  - ↪ agent/master OR stand-alone architecture
  - ↪ cross-platform through Puppet's Resource Abstraction Layer (RAL)
  - ↪ Git-based workflow
  - ↪ PKI-based security (X.509)
- DevOps tool of choice for configuration management
  - ↪ Declarative Domain Specific Language (DSL)

Average server installation/configuration time: $\simeq$ 3-6 min

Configuration Management advantages

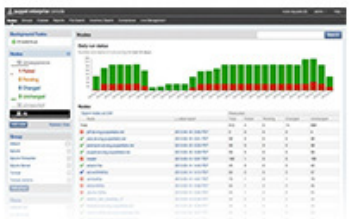
- **Infrastructure as Code**: Track, Test, Deploy, Reproduce, Scale
  - ↪ Code commits log shows the **history of change** on the infrastructure
- **Reproducible setups**
  - ↪ Do once, repeat forever
- **Scale** quickly:
  - ↪ Done for one, use on many
- **Coherent** and consistent server setups
- **Aligned Environments** for `devel`, `test`, `qa`, `prod` nodes

Alternatives to Puppet: Chef, CFEngine, Salt, Ansible

Software related to Puppet

Tool	Description
Facter	Complementary tool to retrieve system's data
MCollective	Infrastructure Orchestration framework
Hiera	Key-value lookup tool where Puppet data can be placed
PuppetDB	Stores all the data generated by Puppet
Puppet DashBoard	A Puppet Web frontend and External Node Classifier (ENC)
The Foreman	A well-known third party provisioning tool and Puppet ENC
Geppetto	A Puppet IDE based on Eclipse

How Puppet work ?

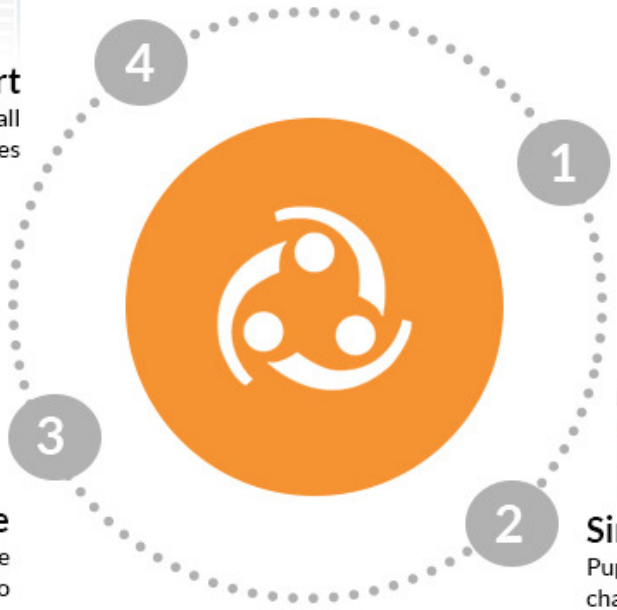


Report

Gain insight into the infrastructure by tracking all changes applied and state of the nodes

Define

Define the desired state of each system using Puppet's declarative language



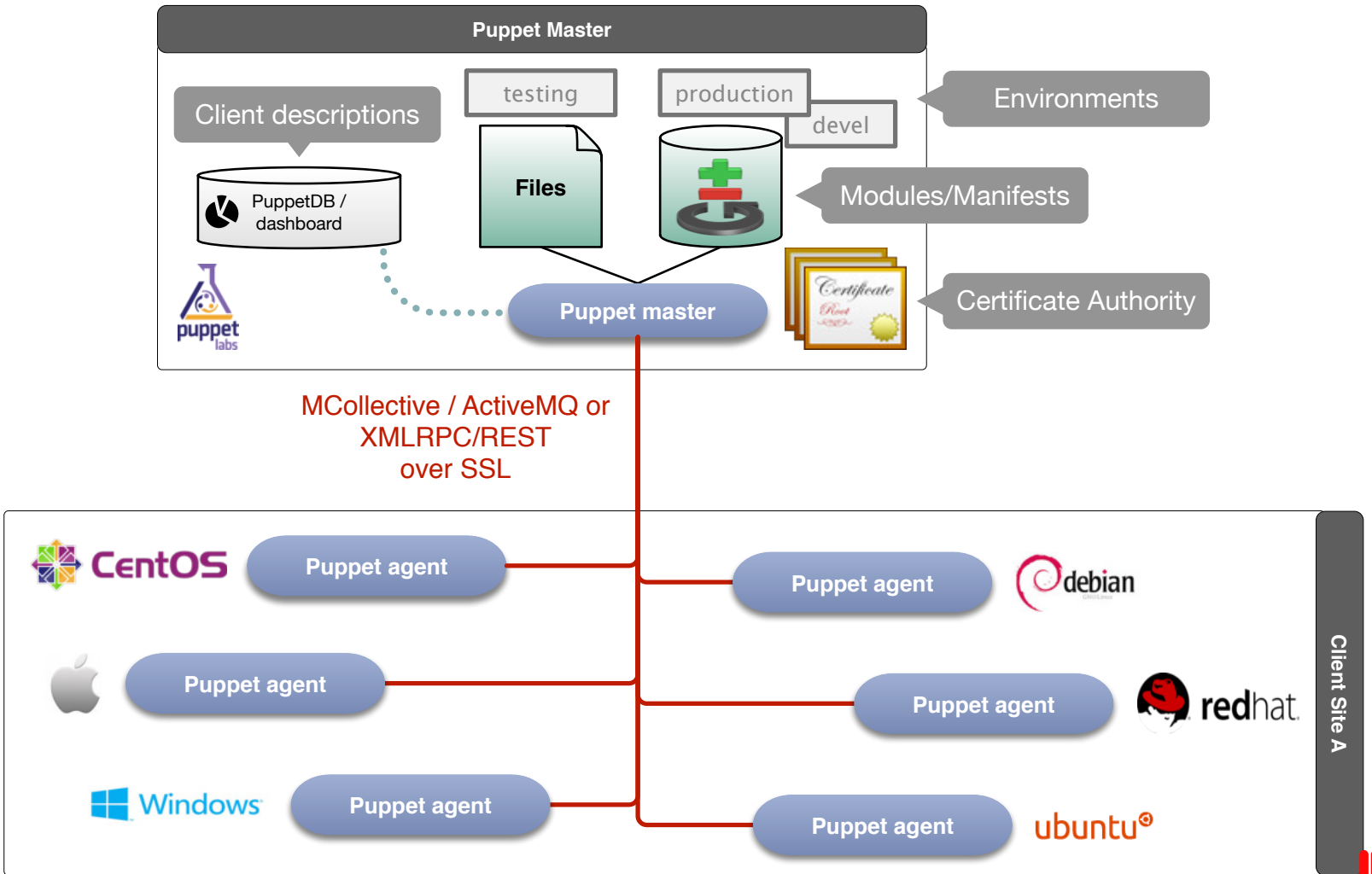
Enforce

Puppet will check the system and updates it to the desired state

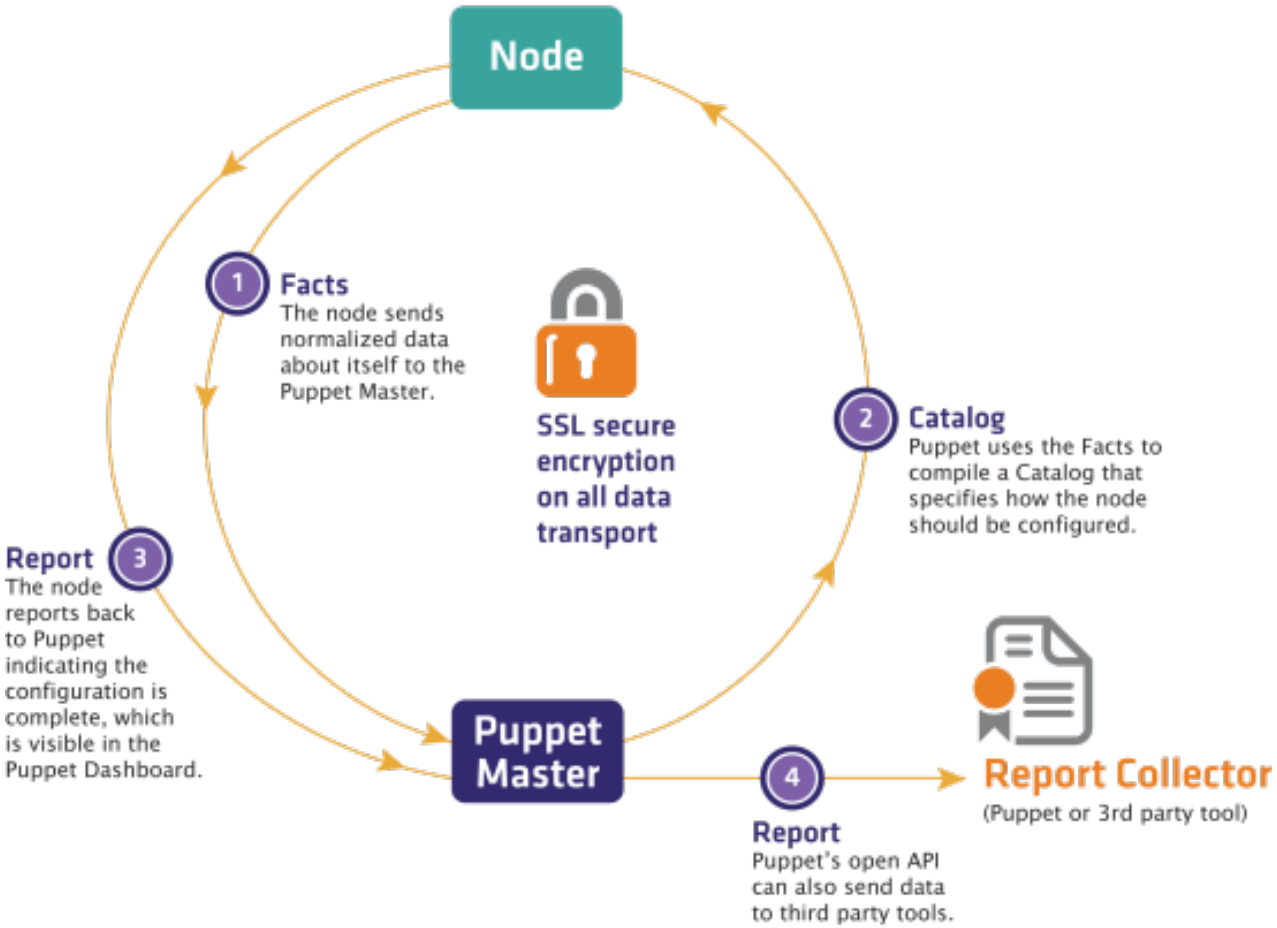
Simulate

Puppet is able to simulate deployment to test the changes without disruption to the infrastructure

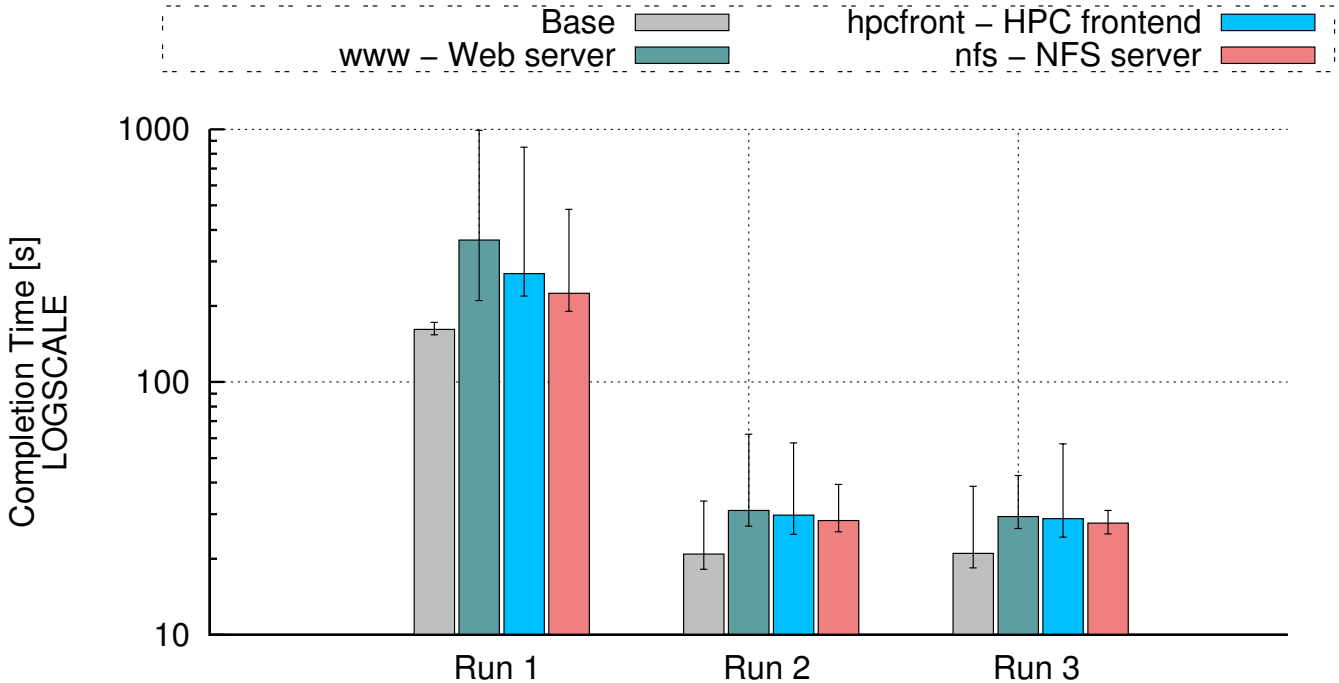
General Puppet Infrastructure



Puppet Data Flow



Puppet Performances



- between 161s and 364s to completely bootstrap a virgin node
  - ↪ between 20s and 31s to later check/correct the config

Now proposed as an IT service to external consumers

Puppet Installation

- Debian, Ubuntu (available by default)

```
$> apt-get install puppet       # On clients (nodes)
$> apt-get install puppetmaster # On server (master)
```

- RedHat, Centos, Fedora
  - ↪ Add EPEL repository or RHN Extra channel

```
$> rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-el-<version>.noarc
$> yum install puppet         # On clients (nodes)
$> yum install puppet-server # On server (master)
```

- Other OS: https://docs.puppet.com/puppet/3.8/reference/pre_install.html

Puppet DSL

- A Declarative Domain Specific Language (DSL)
  - ↪ defines **STATES** (and **not** procedures)
- Puppet code is written in **manifests** `<file>.pp`
  - ↪ **declare resources** that affect elements of the system
    - ✓ each resource has a type (`package`, `service`, `file`, `user`, `exec` ...)
    - ✓ each resource has a **uniq** title
  - ↪ resources are grouped in **classes**
- Classes and configuration files are organized in **modules**
- **Example** of resources types:

```
file { '/etc/motd':
  content => "Toto"
}
```

```
package { 'openssh':
  ensure => present,
}
```

```
service { 'httpd':
  ensure => running,
  enable => true,
}
```

Puppet Classes

- **Containers** of different resources
  - ↪ Can have parameters since Puppet 2.6

```
class mysql (
  $root_password = 'default_value',
  $port          = '3306',
) {
  package { 'mysql-server':
    ensure => present,
  }
  service { 'mysql':
    ensure    => running,
  }
  [...]
}
```

Puppet Classes Declaration

- To use a class previously defined, we **declare** it
- "Old style" class declaration, without parameters:

```
include mysql
```

- "New style" (from Puppet 2.6) with explicit parameters:

```
class { 'mysql':
  root_password => 'my_value',
  port          => '3307',
}
```

- A class is **uniq** to a given node

Puppet Defines

- Similar to parametrized classes . . .
  - ↪ . . . but can be used multiple times (with different titles).

```
# Definition of a define
define apache::virtualhost (
  $ensure   = present,
  $template = 'apache/virtualhost.conf.erb' ,
  [...] ) {
  file { "ApacheVirtualHost_${name}":
    ensure  => $ensure,
    content => template("${template}"),
  }
}
# Declaration of a define:
apache::virtualhost { 'www.uni.lu':
  template => 'site/apache/www.uni.lu-erb'
}
```

Puppet Variables and Facts

- Can be defined in different places and by different actors:
  - ↪ by client nodes as facts
  - ↪ defined by users in Puppet code, on Hiera on in the ENC
  - ↪ built-in and be provided directly by Puppet
- Facts using `facter`:
  - ↪ runs on clients and collects facts that the server can use as variables

```
$> facter
architecture => x86_64
fqdn => toto.uni.lu
hostname => toto
kernel => Linux
memorytotal => 16.00 GB
netmask => 255.255.255.0
operatingsystem => Centos
operatingsystemrelease => 6.3
osfamily => RedHat
virtual => physical
[...]
```

Puppet User Variables

- In Puppet manifests:

```
$role = 'mail'

$package = $::operatingsystem ? {
    /(?i:Ubuntu|Debian|Mint)/ => 'apache2',
    default                   => 'httpd',
}
```

- In an External Node Classifier (ENC)
  - ↪ Commonly used ENC are Puppet DashBoard, the Foreman, Puppet Enterprise.
- In an Hiera backend

```
$syslog_server = hiera(syslog_server)
```

Puppet Nodes

- A node is identified by the PuppetMaster by its **certname**
  - ↪ defaults to the node's fqdn

```
node 'web01' {
  include apache
}
```

```
node /^www\d+$/ {
  include apache
}
```

- Nodes classification can be done by External Node Classifier (ENC)
  - ↪ Puppet DashBoard, The Foreman and Puppet Enterprise
- Nodes classification can be done also by Hiera
  - ↪ In `/etc/puppet/manifests/site.pp`

```
hiera_include('classes')
```

Puppet Operational modes

- **Masterless** - apply Puppet manifests directly on the target system.
  - ↪ No need of a complete client-server infrastructure.
  - ↪ Have to distribute manifests and modules to the managed nodes.

```
$> puppet apply --modulepath /modules/ /manifests/file.pp
```

Puppet Operational modes

- **Masterless** - apply Puppet manifests directly on the target system.
  - ↪ No need of a complete client-server infrastructure.
  - ↪ Have to distribute manifests and modules to the managed nodes.

```
$> puppet apply --modulepath /modules/ /manifests/file.pp
```

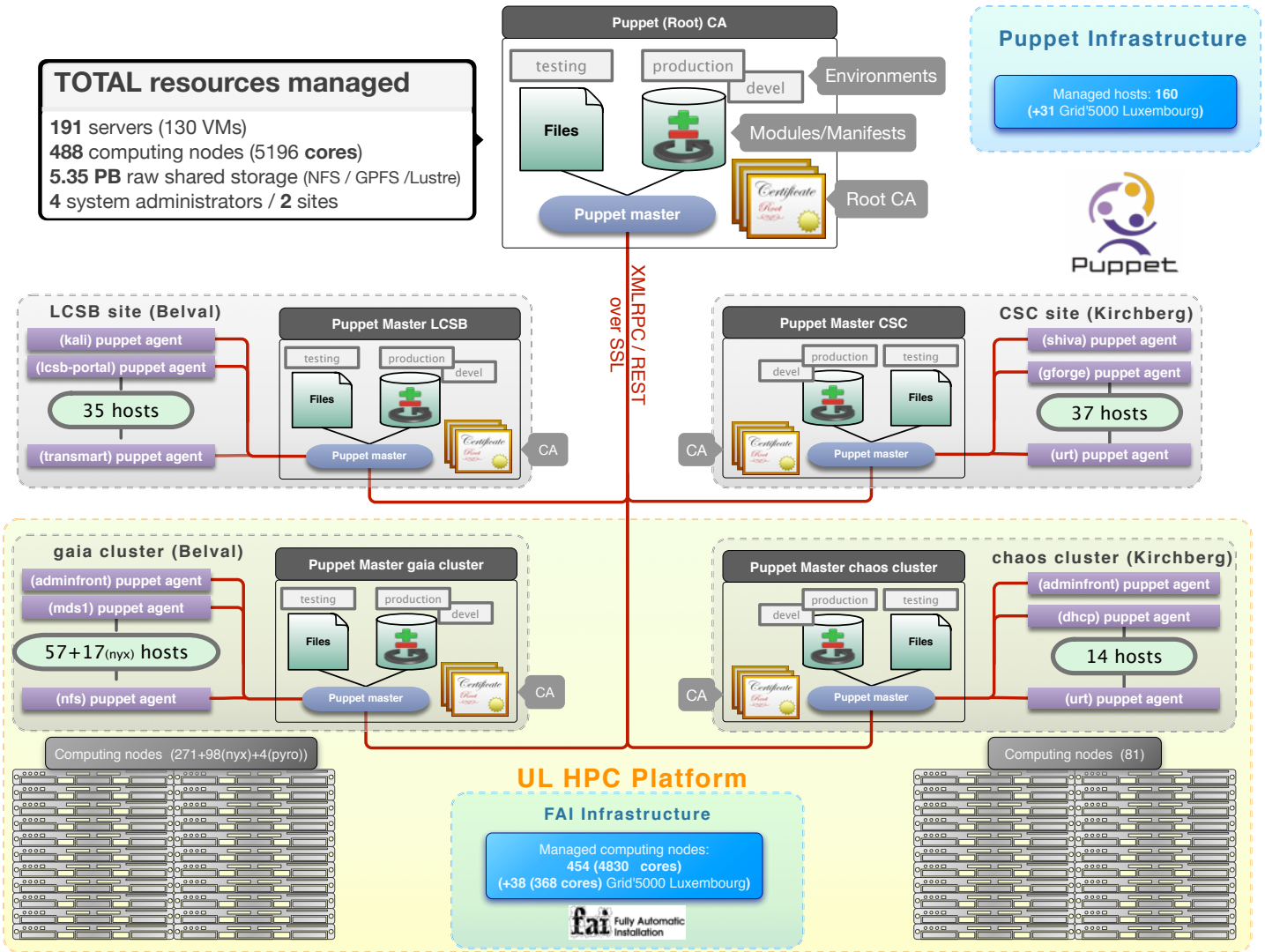
- **Master / Client** Setup
  - ↪ server (running as `puppet`) listening on 8140 on the Puppet Master
  - ↪ client (running as `root`) on each managed node.
    - ✓ Run as a service (default), via cron (with random delays), manually or via MCollective
  - ↪ Client and Server have to share SSL certificates
    - ✓ certificates must be signed by the Master CA

```
$> puppet agent --test [--noop] [--environment <environment>]
```

Components of a Puppet architecture

- **Tasks to be deal with**:
  - ↪ definition of the classes to be included in each node
  - ↪ definition of the parameters to use for each node
  - ↪ definition of the configuration files provided to the nodes
- **Components**
  - ↪ Master, CA, and agents
  - ↪ (optional) ENC - External Node Classifier
  - ↪ (optional) ldap/IPA backend
  - ↪ Hiera - Data key-value backend
  - ↪ Public modules - Public shared modules — Puppet Forge
  - ↪ Site modules - Local custom modules

ULHPC Puppet Infrastructure



Summary

Ruby / RVM / Bundler

- Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
- Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Ruby / RVM / Bundler

- Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
- Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Typical setup of a freshly cloned project:

```
$> gem install bundler # assuming it is not yet available
$> bundle              # clone ruby deps/env as defined in Gemfile*
$> rake -T             # To list the available tasks
```

Ruby / RVM / Bundler

- Bring the flexibility of `Rakefile` (`Makefile` + Ruby)
- Bundler: **reproducible** running environment **across** developpers
  - ↪ easy configuration through `Gemfile[.lock]` + `bundle` command
- RVM: sandboxed environment per project (**alternative**: rbenv)
  - ↪ easy configuration through `.ruby-{version,gemset}` files

Typical setup of a freshly cloned project:

```
$> gem install bundler # assuming it is not yet available
$> bundle              # clone ruby deps/env as defined in Gemfile*
$> rake -T             # To list the available tasks
```

Recommended Gems

`falkorlib`, `rake`, `bundler`, `git_remote_branch`

Python / Pip

- `pip`: Python package manager
  - ↪ "nice" python packages: `mkdocs`...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

Python / Pip

- `pip`: Python package manager
  - ↪ "nice" python packages: `mkdocs`...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

```
$> pip install -U pip                       # upgrade on Linux/Mac OS
```

Python / Pip

- pip: Python package manager
  - ↪ "nice" python packages: mkdocs...
  - ↪ Windows: install via Chocolatey

```
$> pip install <package>                    # install <package>
```

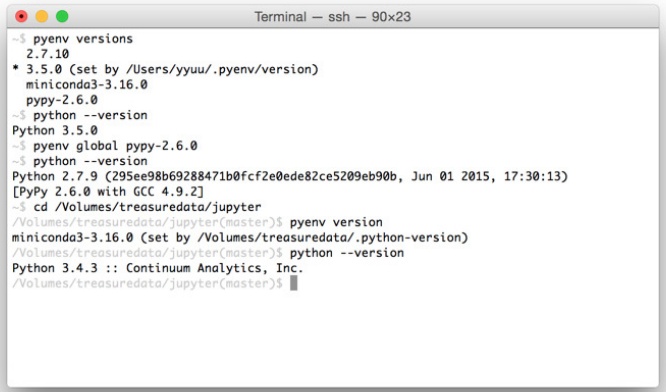
```
$> pip install -U pip                       # upgrade on Linux/Mac OS
```

- Dump python environment to a requirements file

```
$> pip freeze -l > requirements.txt         # as Ruby Gemfiles
```

Pyenv / VirtualEnv / Autoenv

- pyenv: ≃ RVM/rbenv for Python
- virtualenv ≃ RVM Gemset
- (optional) autoenv
  - ↪ Directory-based shell environments
  - ↪ easy config through .env file. **Ex**:




```
# (rootdir)/.env : autoenv configuration file
pyversion=‘head .python-version‘
pvenv=‘head .python-virtualenv‘

pyenv virtualenv --force --quiet ${pyversion} ${pvenv}-${pyversion}
# activate it
pyenv activate ${pvenv}-${pyversion}
```

Documentation

- Priviledge Markdown-based documentation
  - ↪ easy to track over Git (`text` files, **not** Word/RFT etc.)
  - ↪ easy to export to any format using `pandoc` / `multimarkdown`
  - ↪ **focus on writing**, viewers for all platform
    - ✓ Mac OS: MOU, Marked 2
    - ✓ Linux: Remarkable, Retext
    - ✓ Windows: MarkdownPad, Remarkable

Git-based Markdown Wiki

- Gollum, as embedded in GitLab
- Mkdocs

Gollum / MkDocs

- Advantage: possibility to serve the HTML **locally**
  - ↪ Gollum: `gollum` (from root directory) `http://localhost:4567`
  - ↪ Mkdocs: `mkdocs serve` (from root directory) `http://localhost:8000`

Gollum / MkDocs

- Advantage: possibility to serve the HTML **locally**
  - ↪ Gollum: `gollum` (from root directory) `http://localhost:4567`
  - ↪ Mkdocs: `mkdocs serve` (from root directory) `http://localhost:8000`

```
$> mkdocs new     # initialize 'mkdocs.yml' and docs/ directory
```

Gollum / MkDocs

- Advantage: possibility to serve the HTML **locally**
  - ↪ Gollum: `gollum` (from root directory) `http://localhost:4567`
  - ↪ Mkdocs: `mkdocs serve` (from root directory) `http://localhost:8000`

```
$> mkdocs new    # initialize 'mkdocs.yml' and docs/ directory
```

```
# mkdocs.yml -- MkDocs configuration, all *.md files relative to docs/
site_name: UL HPC Developpers Documentaion
pages:
- Home: 'index.md'
- Tools:
    - SSH: 'tools/ssh.md'
    - Git: 'tools/git.md'
- Configuration:
    - CA Certificates: 'config/certificates/README.md'
theme: readthedocs
```

Summary

Password Management

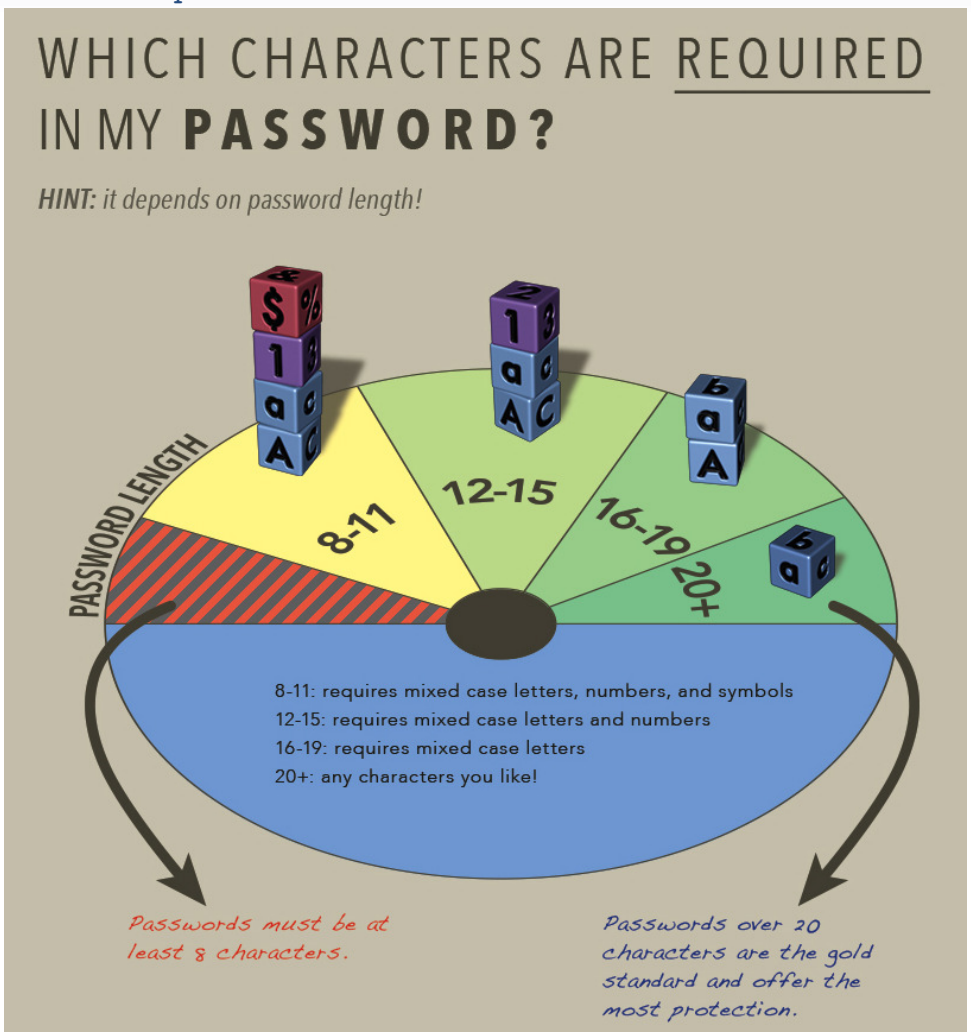
Traditional [Strong] Password policy

- $\geq$ **15** characters, including digits, special chars (#,&,@,$ etc.)
  - ↪ mix upper/lower case
- **avoid** matching dictionary/personal/company/dates info
- renew **periodically**, typically after 180 days.

- Build by selecting words / sentence easy to remember
  - ↪ combine them to respect the above rules

Stanford Password Policy

https://itservices.stanford.edu/service/accounts/passwords/quickguide



Password Manager

Password Manager

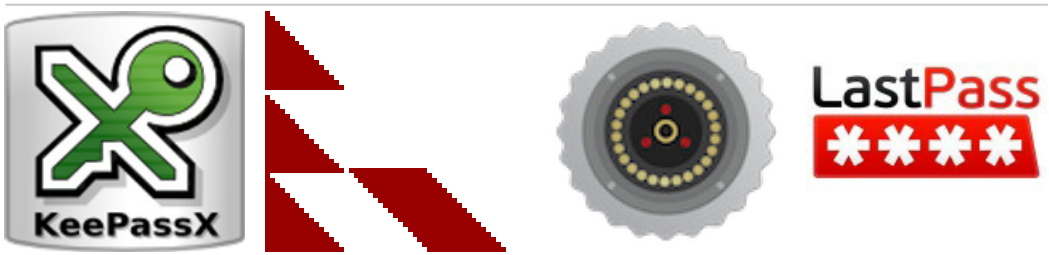
- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Password Manager

Password Manager

- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Open-Source / Cloud-based

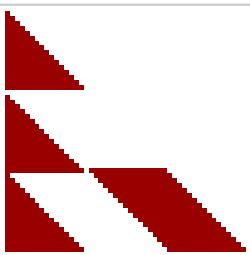


Password Manager

Password Manager

- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Open-Source / Cloud-based



Commercial

GPG+Git Password Management: pass

- pass: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: ~/.password-store/
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: { brew | yum | apt-get } install pass

```
$> pass init <ID> && pass git init # Create the store over git
```

GPG+Git Password Management: `pass`

- `pass`: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: `~/.password-store/`
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: `{ brew | yum | apt-get } install pass`

```
$> pass init <ID> && pass git init # Create the store over git
```

```
$> pass insert <domain>/<name>      # store <domain>/<name>.gpg
```

GPG+Git Password Management: pass

- pass: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: ~/.password-store/
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: { brew | yum | apt-get } install pass

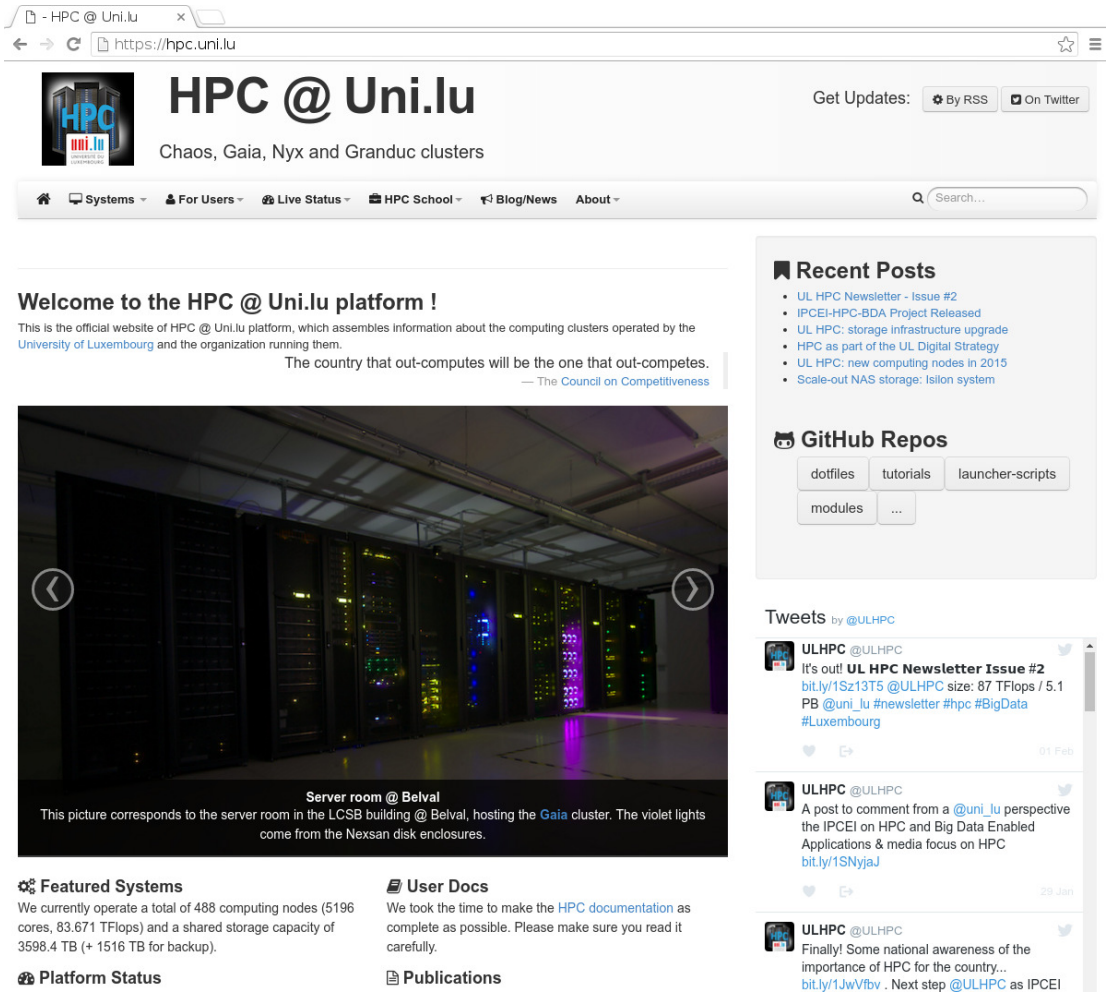
```
$> pass init <ID> && pass git init # Create the store over git
```

```
$> pass insert <domain>/<name>      # store <domain>/<name>.gpg
```

```
$> pass [<domain>/<name>]     # list / retrieve password <name>
```

Summary

High Performance Computing @ UL

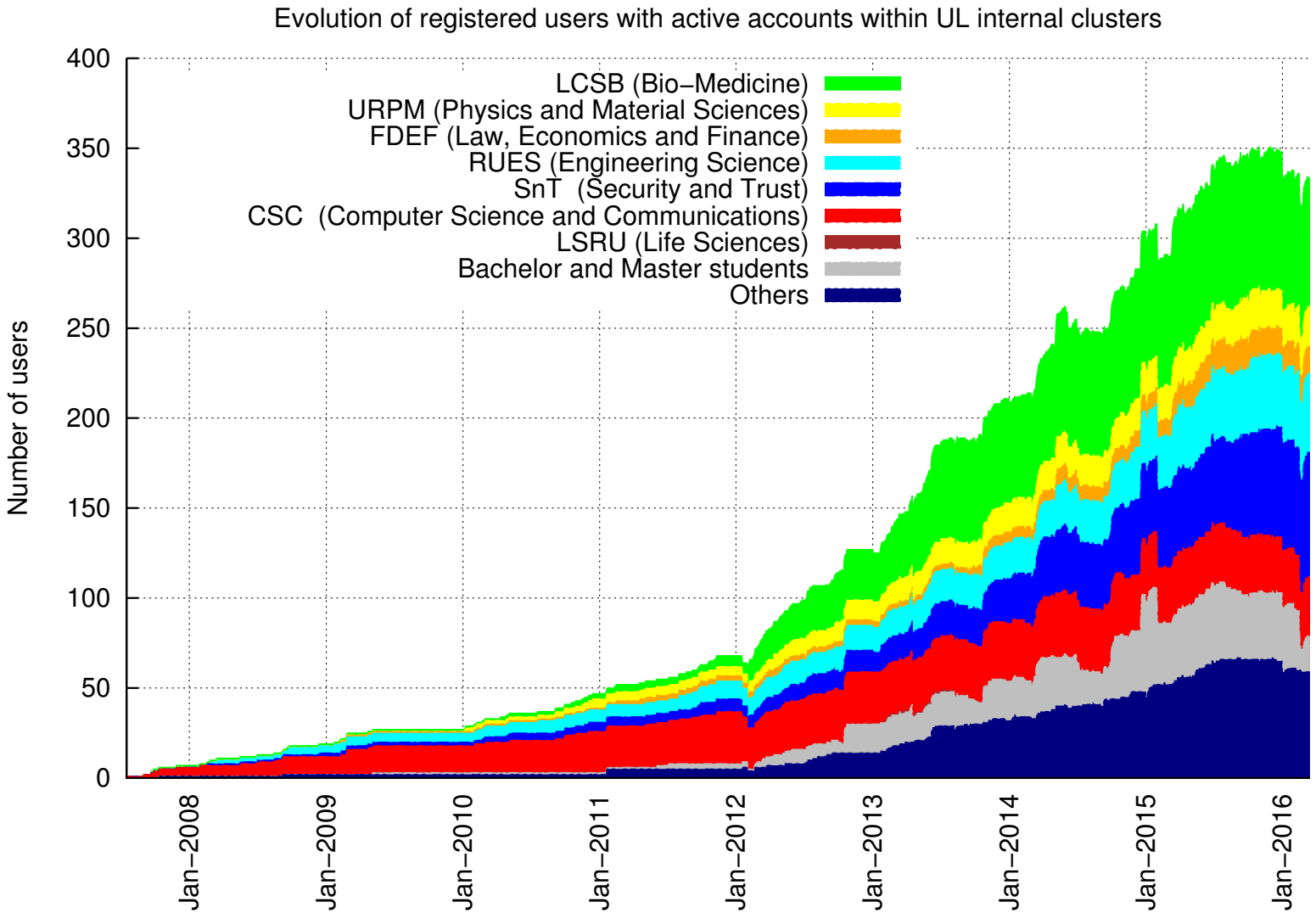


Key numbers

- 344 users
- 98 servers
- 492 nodes
  - ↪ 5300 cores
  - ↪ **85.543 TFlops**
- **5354.4 TB**
- 4 sysadmins
- 2 sites
  - ↪ Kirchberg
  - ↪ Belval

http://hpc.uni.lu

High Performance Computing @ UL



High Performance Computing @ UL

- **Enables & accelerates** scientific discovery and innovation
- **Largest facility** in Luxembourg (after GoodYear R&D Center)

			TFlops	TB	FTEs
Country	**Name/Institute**	**#Cores**	**R_{peak}**	**Storage**	**Manpower**
Luxembourg	UL	5300	85.543	5354.4	2
	CRP GL	800	6.21	144	1.5
France	TGCC Curie, CEA	77184	1667.2	5000	n/a
	LORIA, Nancy	3724	29.79	82	5.05
	ROMEO, UCR, Reims	564	4.128	15	2
Germany	Juqueen, Juelich	393216	5033.2	448	n/a
	MPI, RZG	2556	14.1	n/a	5
	URZ, (bwGrid),Heidelberg	1140	10.125	32	9
Belgium	UGent, VCS	4320	54.541	82	n/a
	CECI, UMons/UCL	2576	25.108	156	> 4
UK	Darwin, Cambridge Univ	9728	202.3	20	n/a
	Legion, UCLondon	5632	45.056	192	6
Spain	MareNostrum, BCS	33664	700.2	1900	14

UL HPC Services

Horizontal HPC & storage services

- for the three UL Faculties and their Research Units
- for the two UL Inter-disciplinary Centres
  - ↪ LCSB, SnT
- . . . and their external partners
- on UL **strategic research priorities**
  - ↪ computational sciences
  - ↪ systems biomedicine
  - ↪ security, reliability and trust
  - ↪ finance

UL HPC Services

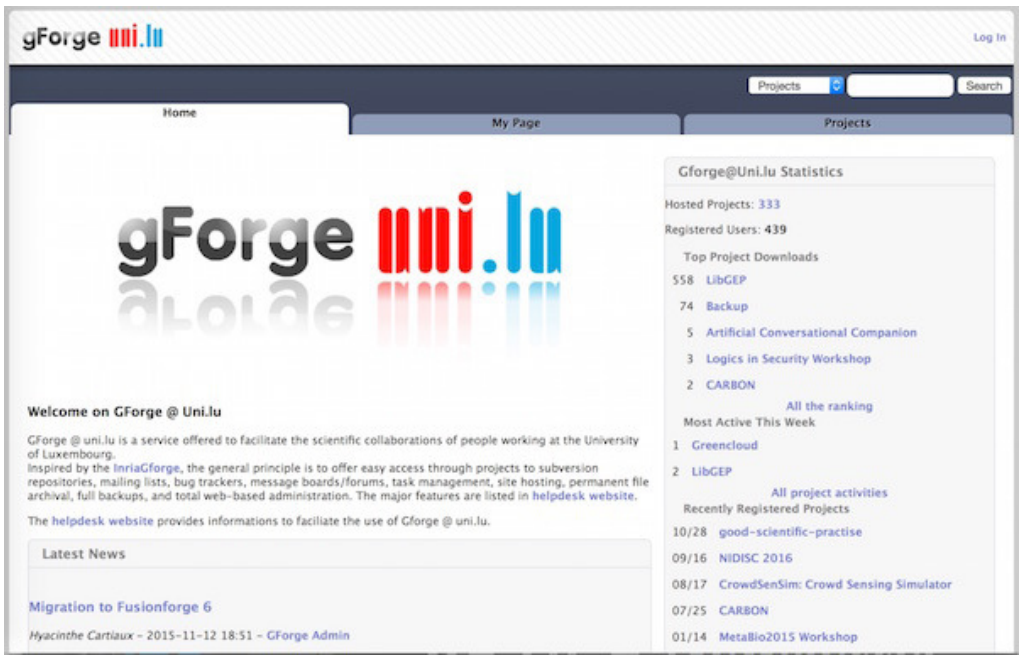
Horizontal HPC & storage services

- for the three UL Faculties and their Research Units
- for the two UL Inter-disciplinary Centres
  - ↪ LCSB, SnT
- . . . and their external partners
- on UL **strategic research priorities**
  - ↪ computational sciences
  - ↪ systems biomedicine
  - ↪ security, reliability and trust
  - ↪ finance

- **Complementary research related services** **Total**: 80 servers
  - ↪ On demand VM hosting for development, frontends, etc.
  - ↪ Project management & collaboration (GForge, GitLab. . .)
  - ↪ Cloud storage (OwnCloud) . . . and many others!

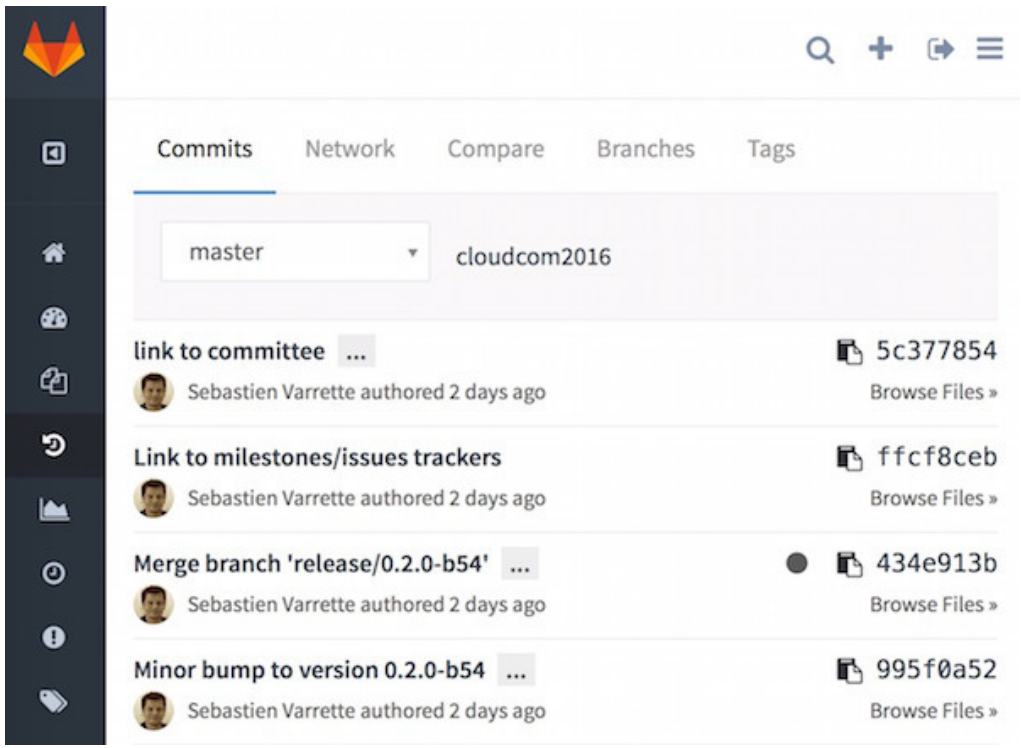
Gforge @ Uni.lu – https://gforge.uni.lu

- Long-running collaboration system, featuring:
  - ↪ static web hosting for projects
  - ↪ Git or Subversion repositories etc.
- Get an account / information: https://helpdesk.gforge.uni.lu/
- Open to anybody **but** separate authentication base



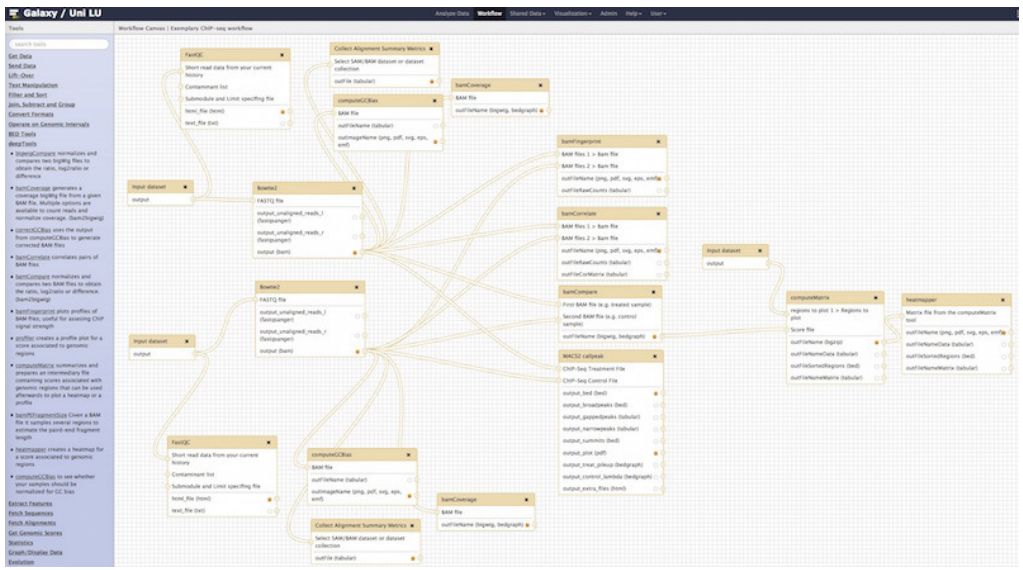
Gitlab @ Uni.lu – https://gitlab.uni.lu

- Similar to Github
  - ↪ **advanced Git repository management**
  - ↪ . . . incl. **private** projects
- Open to _UL staff___ with an **HPC account**



Galaxy – http://galaxy-server.uni.lu

- Web-based platform
- Simplified interface to many popular **bioinformatics tools**
  - ↪ . . . and generation of **reproducible workflows**.



Summary

Summary

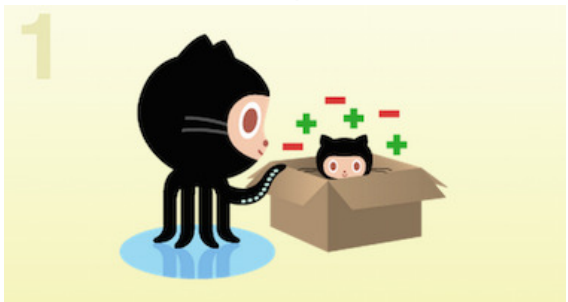
What Git will now mean to you...

Github

github.com

(Reference) web-based Git repository hosting service

Set up Git



Create Repository

Fork repository

Work together

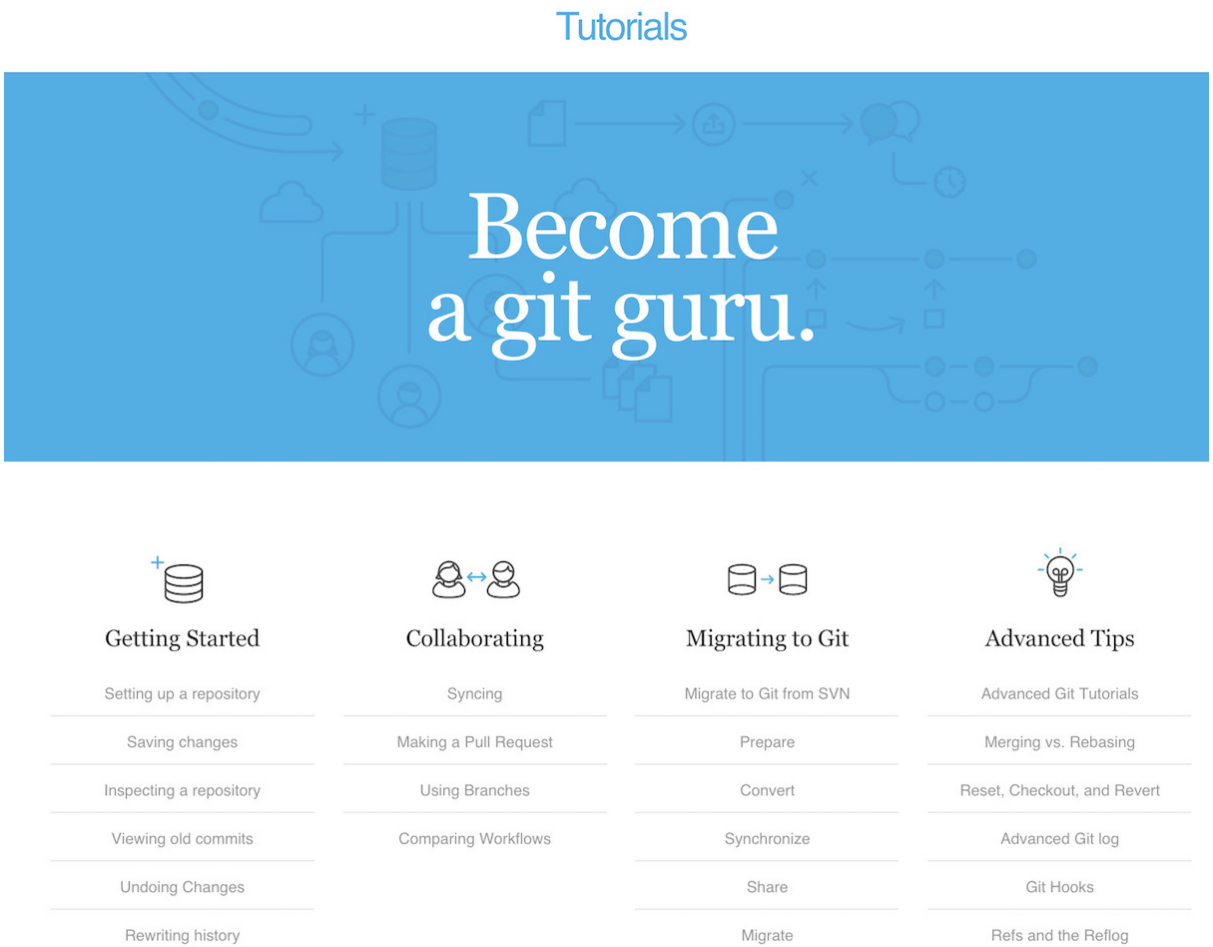
git-scm.com --everything-is-local

Git – the simple Guide

http://rogerdudler.github.io/git-guide/

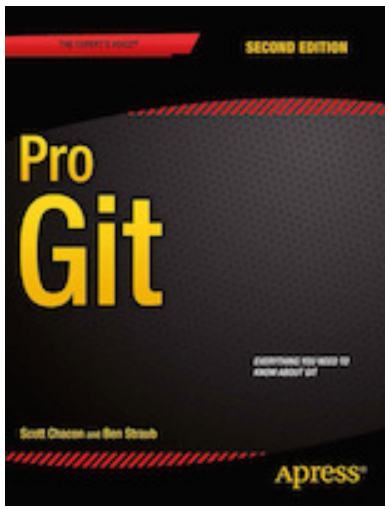
Atlassian Tutorials

https://www.atlassian.com/git/tutorials/



Pro Git Book – progit.org

- Open-Source Book on Git by S. Chacon and B. Straub
  - ↪ Sources (on Github)
  - ↪ Online Reading – PDF
- See also Git Internal, also by S. Chacon

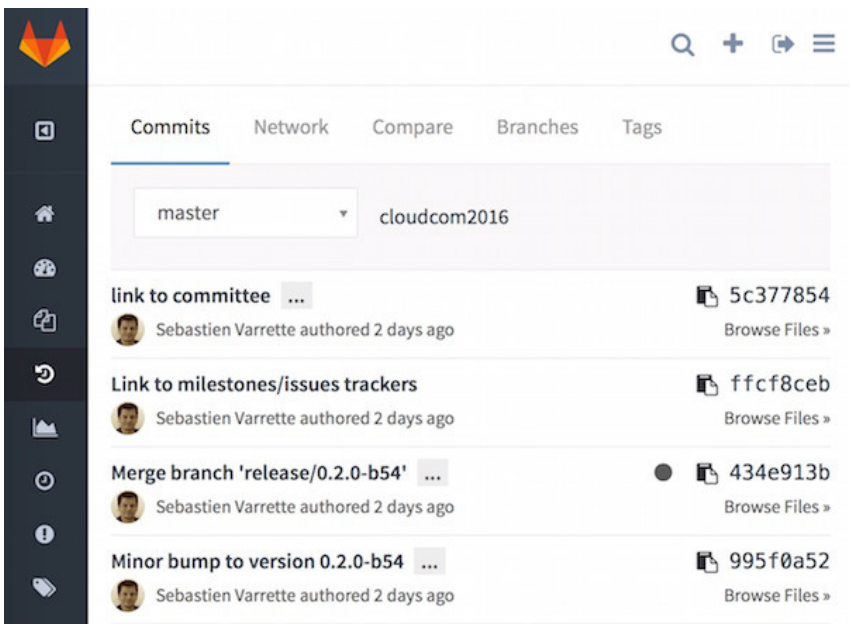


- **Note**: Most images of this talk comes from this book
  - ↪ more precisely the first edition

Gitlab @ Uni.lu

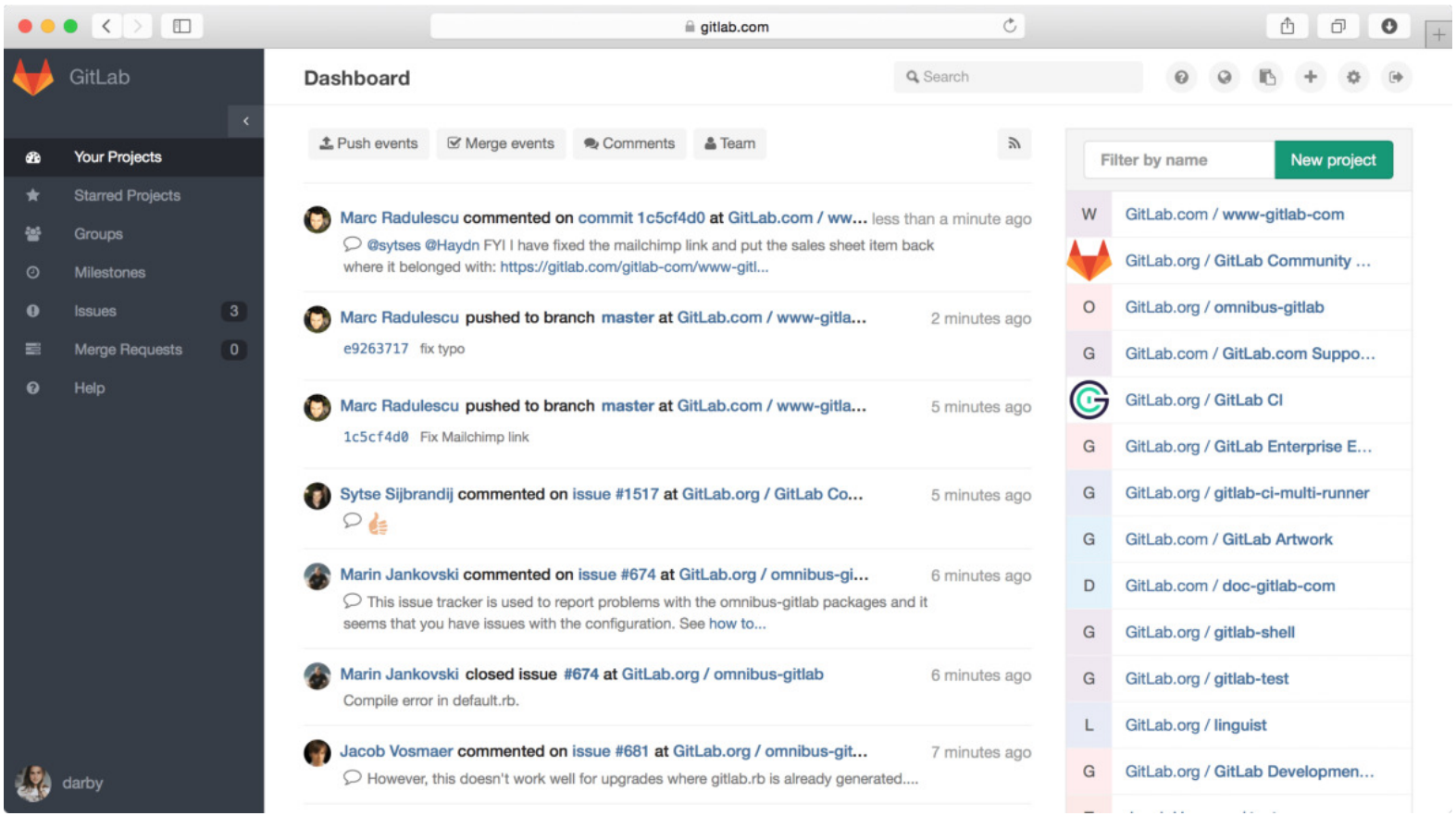
https://gitlab.uni.lu .t

- Gitlab $\simeq$ Github Clone, for deployment on internal servers
  - ↪ web-based Git repository manager, wiki & issue tracking
  - ↪ GitLab CI for continuous integration and delivery.
- Open to **UL staff** with an **HPC account**



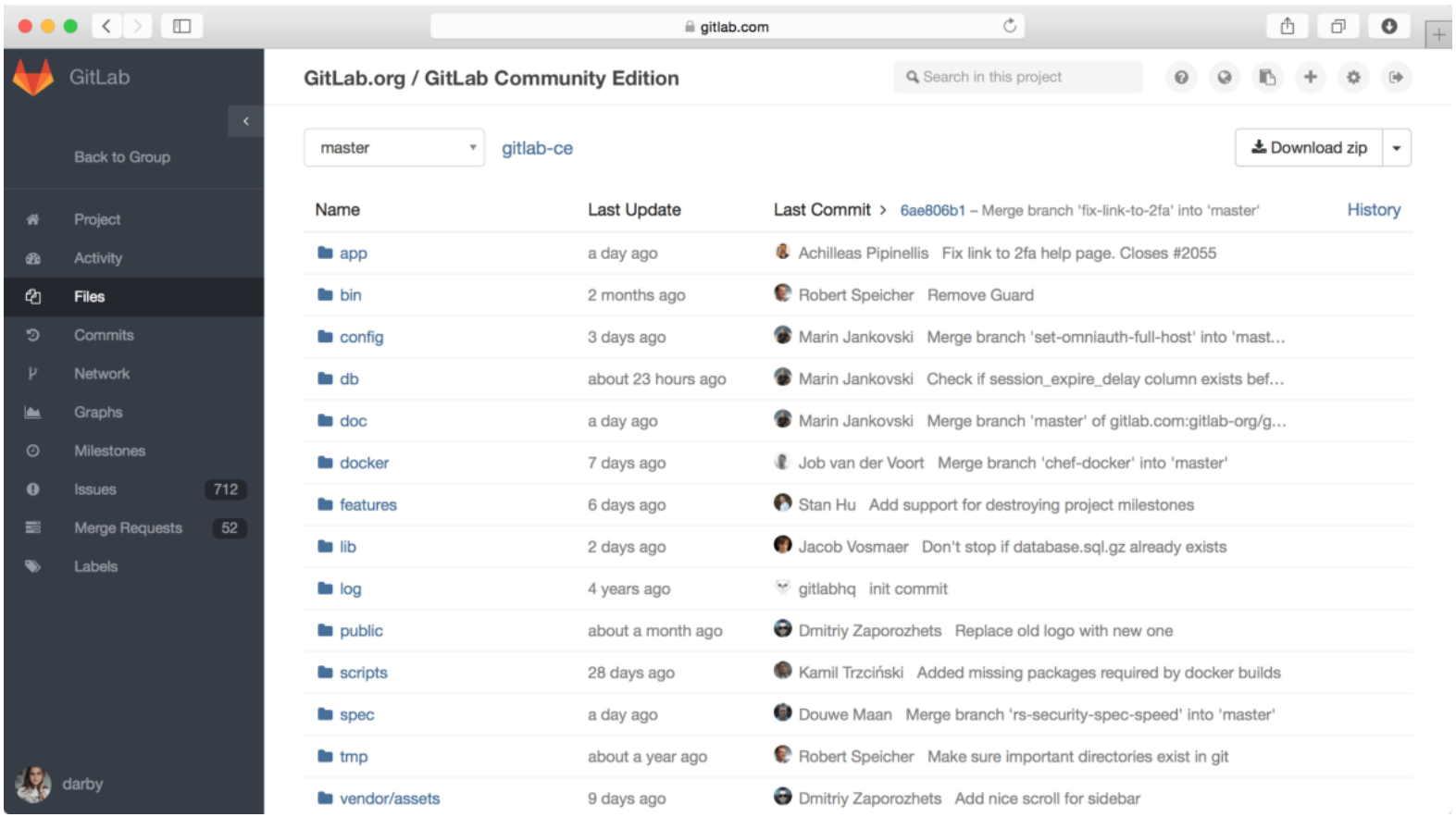
Gitlab Features

- **Activity Stream**



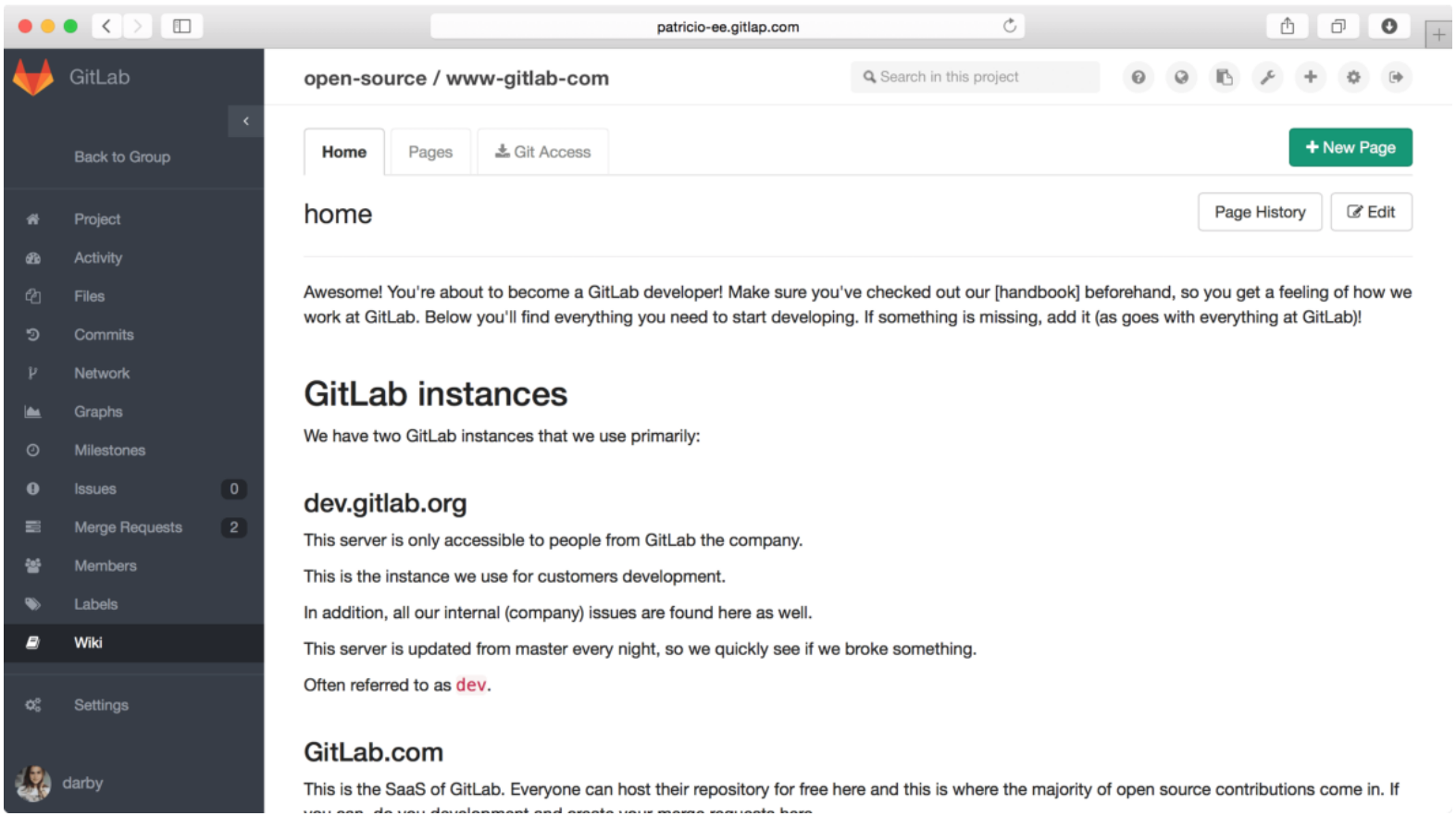
Gitlab Features

- **File Browser**



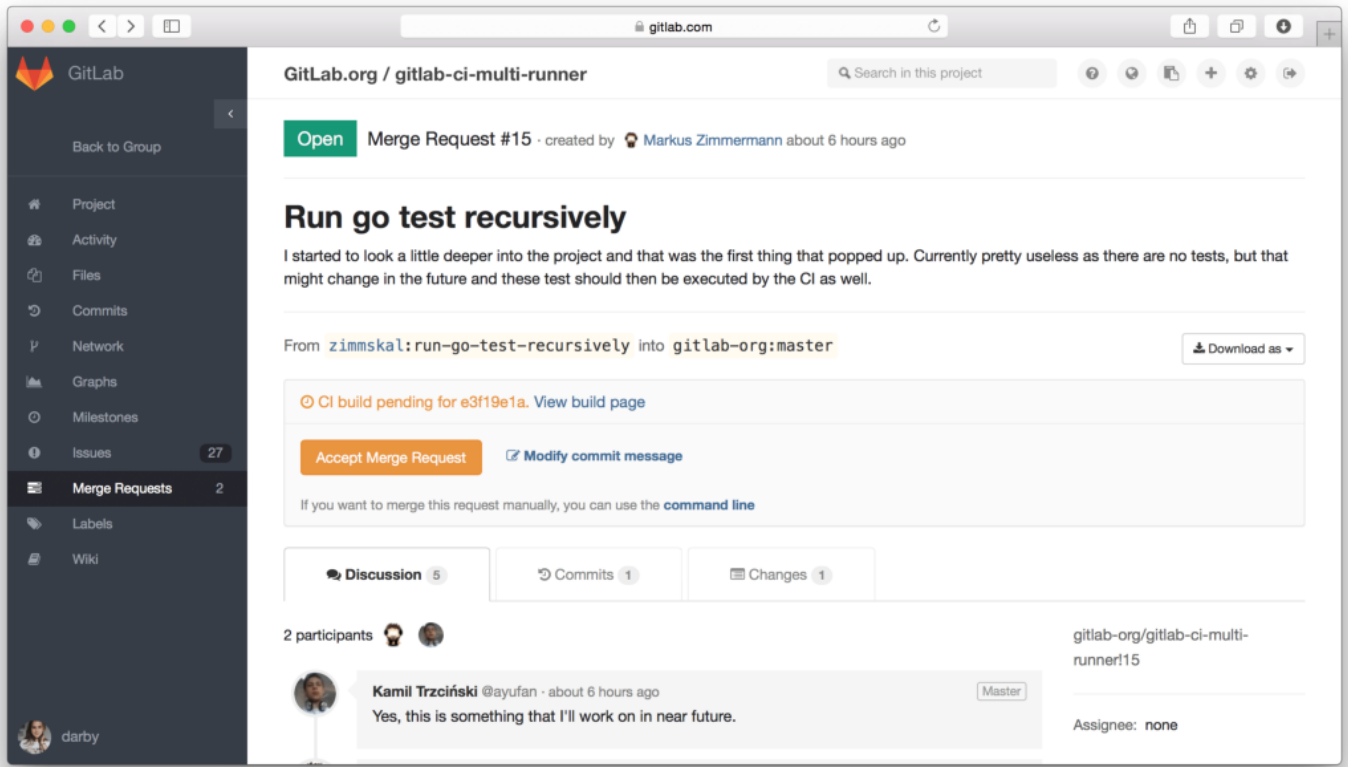
Gitlab Features

- **Git/Markdown powered Wiki**



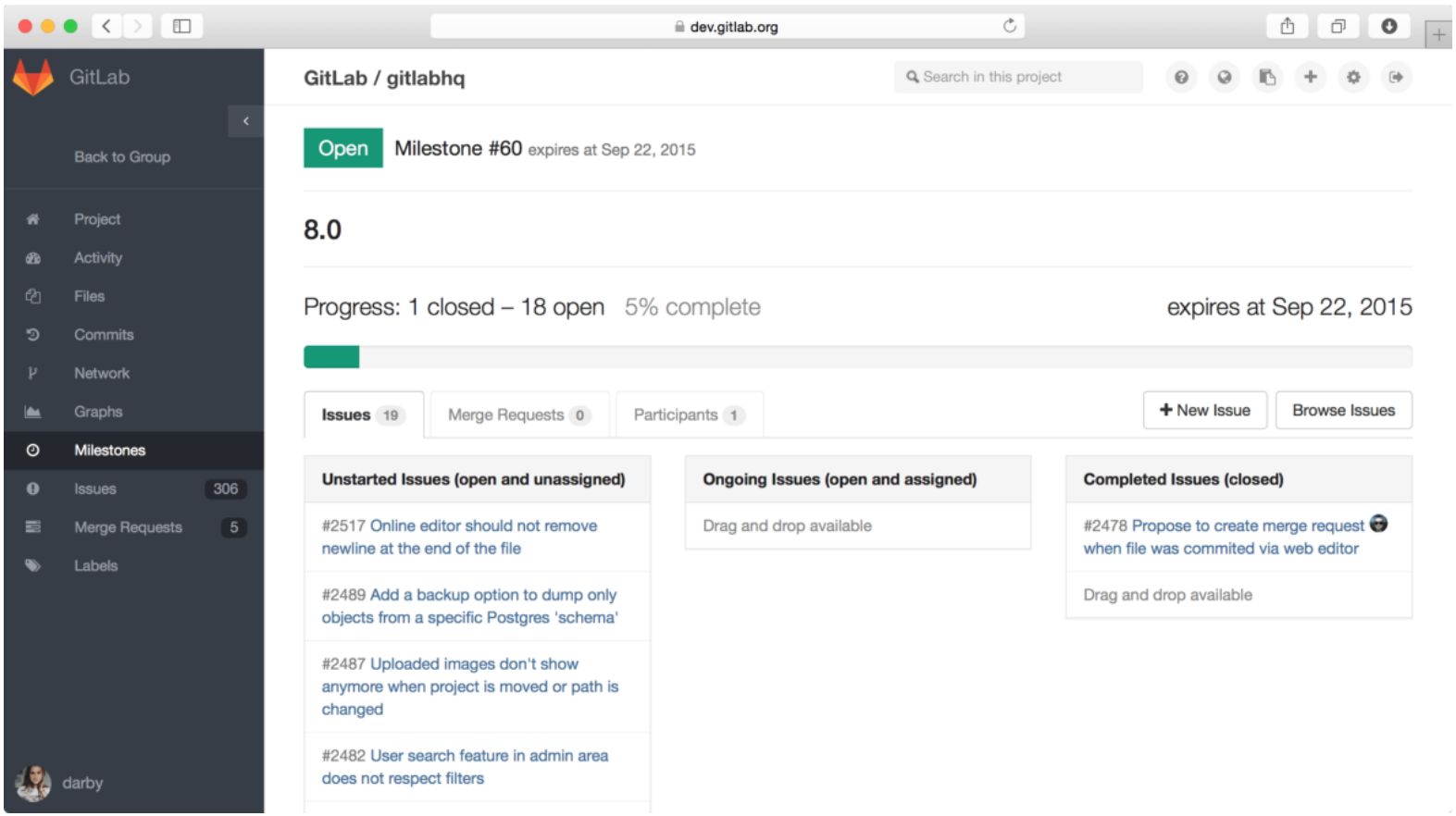
Gitlab Features

- **Powerful Code Review**



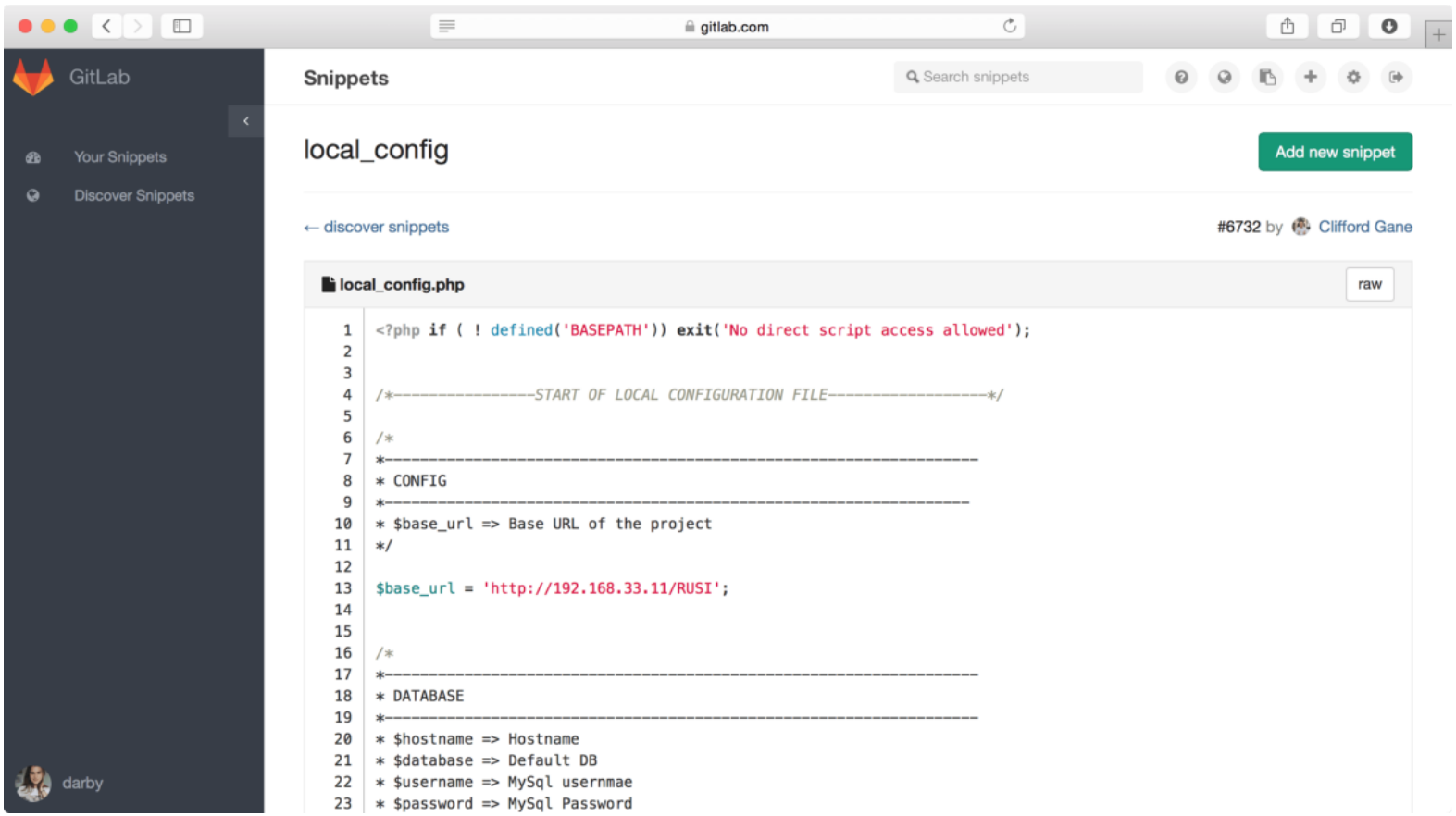
Gitlab Features

- **Issue Management**



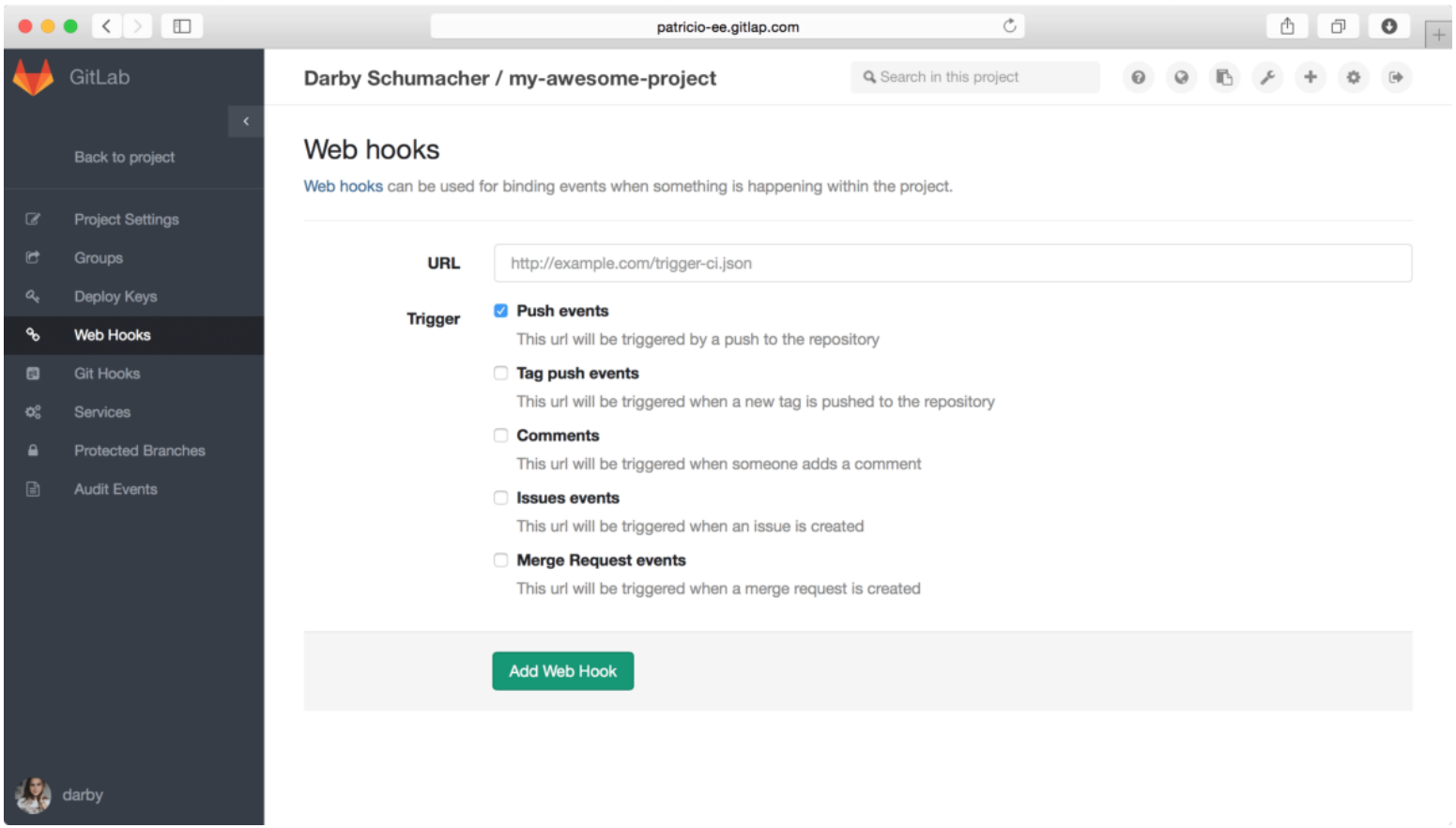
Gitlab Features

- **Code Snippets**



Gitlab Features

- **Web/Service Hooks**



Summary

Why use Version Control?

- Version Control = Revision Control = Source Control
  - ↪ lets you track your files over time.
- you probably cooked up your own!
  - ↪ ever get files like `main-v2.tex`, `CORE-proposal.doc.old` or `2015-03-cv.pdf`?

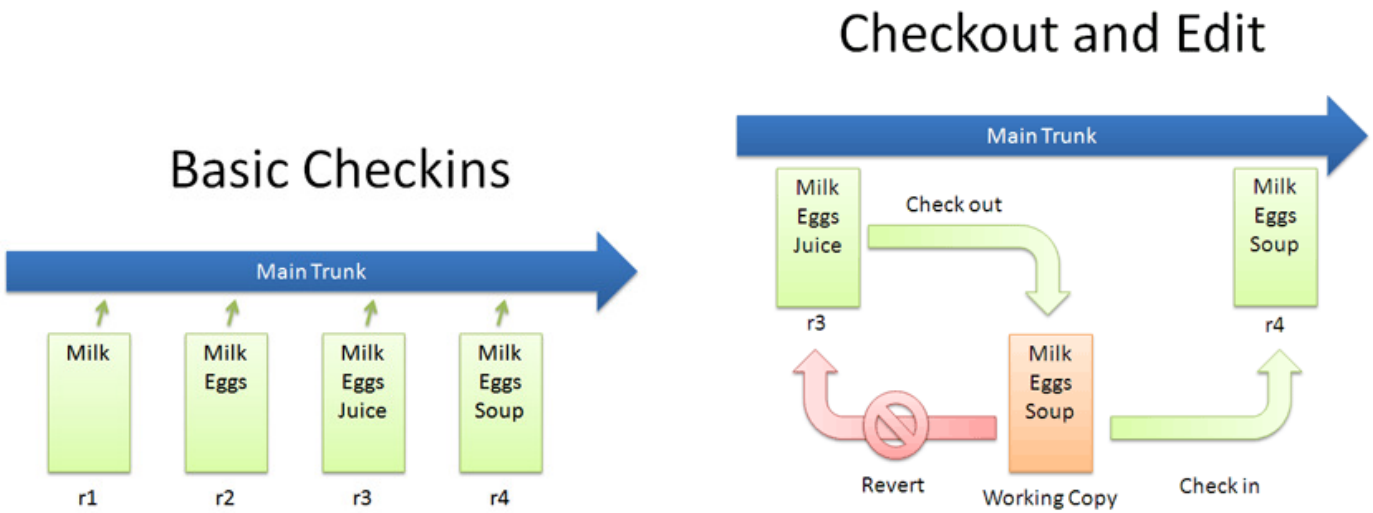
Why use Version Control?

- Version Control = Revision Control = Source Control
  - ↪ lets you track your files over time.
- you probably cooked up your own!
  - ↪ ever get files like `main-v2.tex`, `CORE-proposal.doc.old` or `2015-03-cv.pdf`?

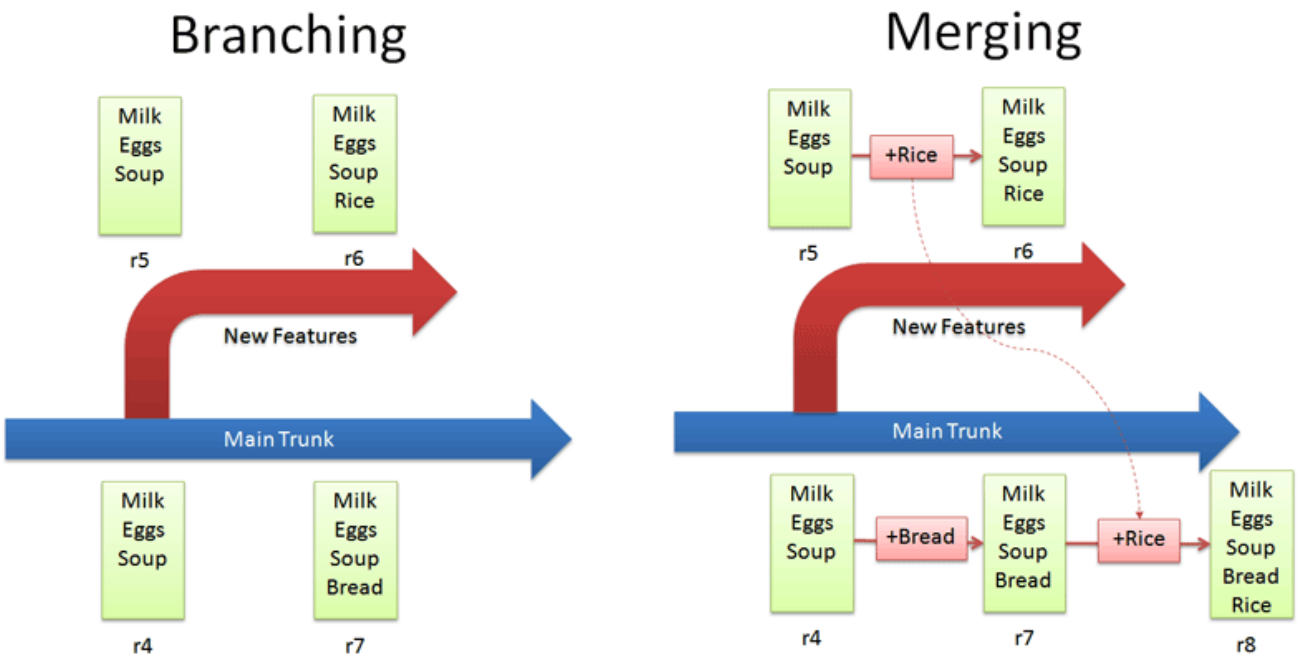
Version Control System (VCS)

- Integrated fool-proof framework for:
  - ↪ Backup and Restore
  - ↪ Synchronization / Collaborating
  - ↪ Short and long-term undo / Tracking changes
  - ↪ Sandboxing

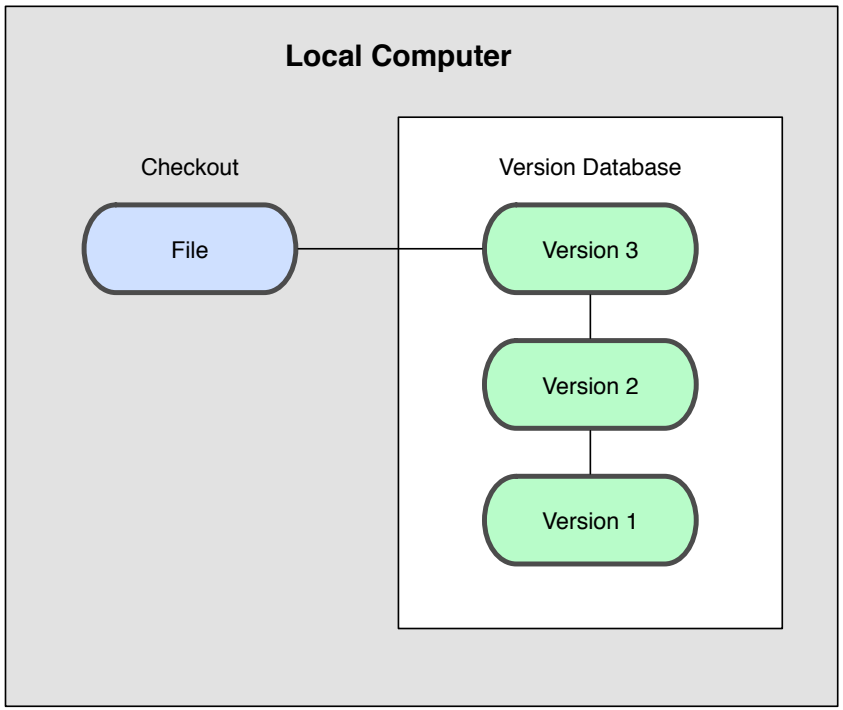
Typical VCS Workflow



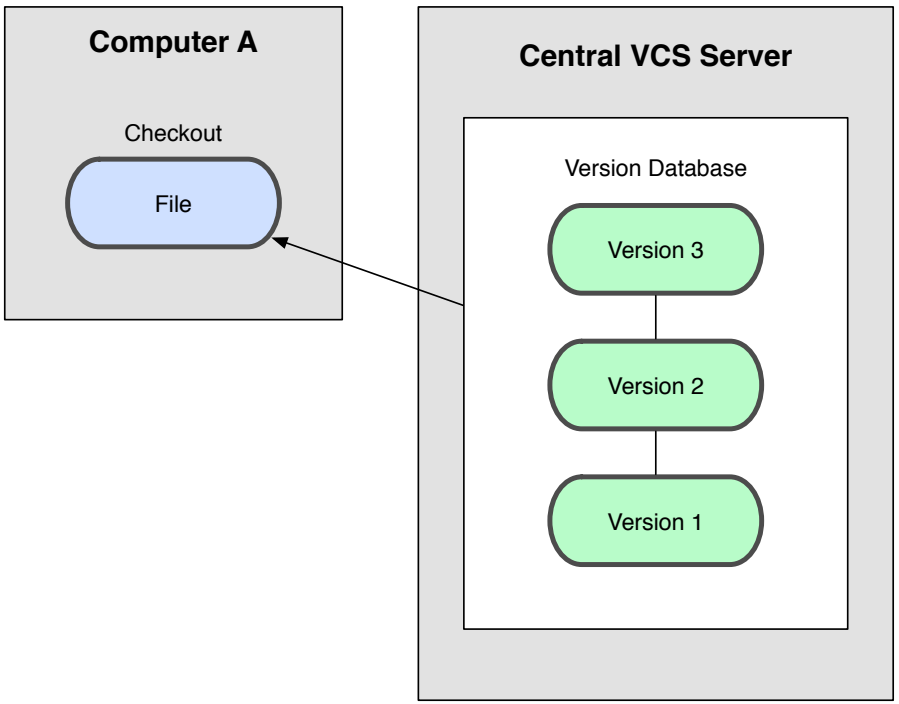
Typical VCS Workflow



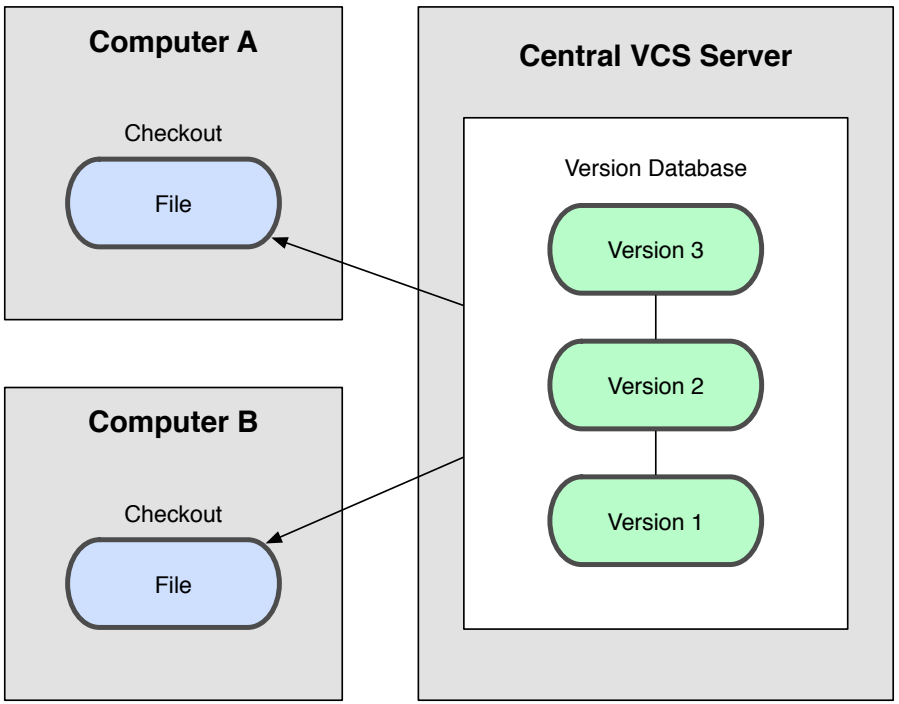
Local VCS – RCS, Mac OS Versions



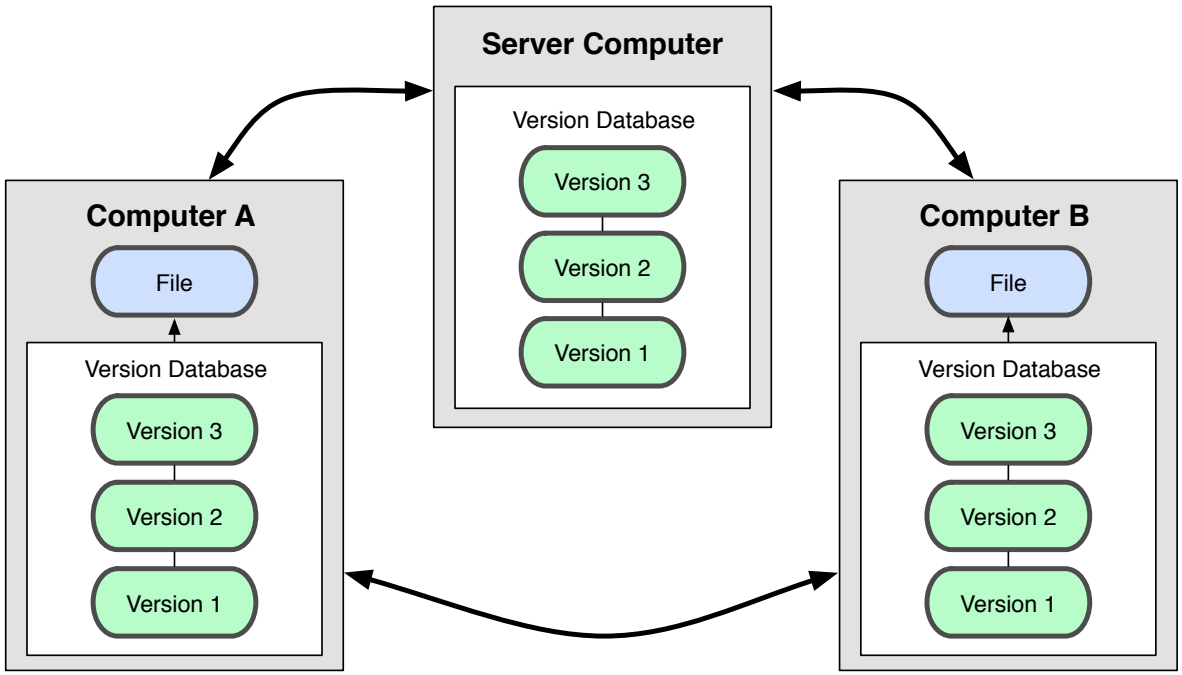
Centralized VCS – CVS, SVN



Centralized VCS – CVS, SVN

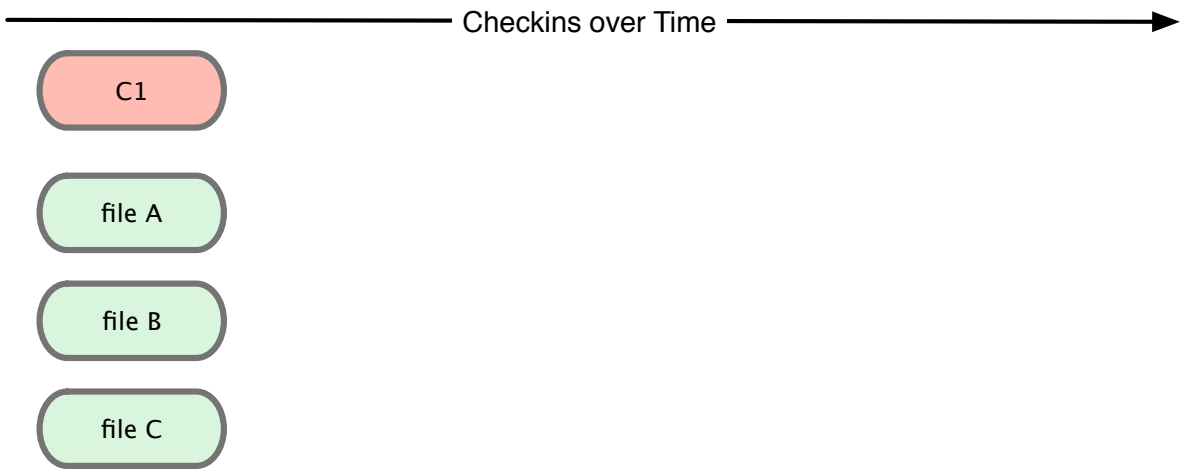


Distributed VCS – Git

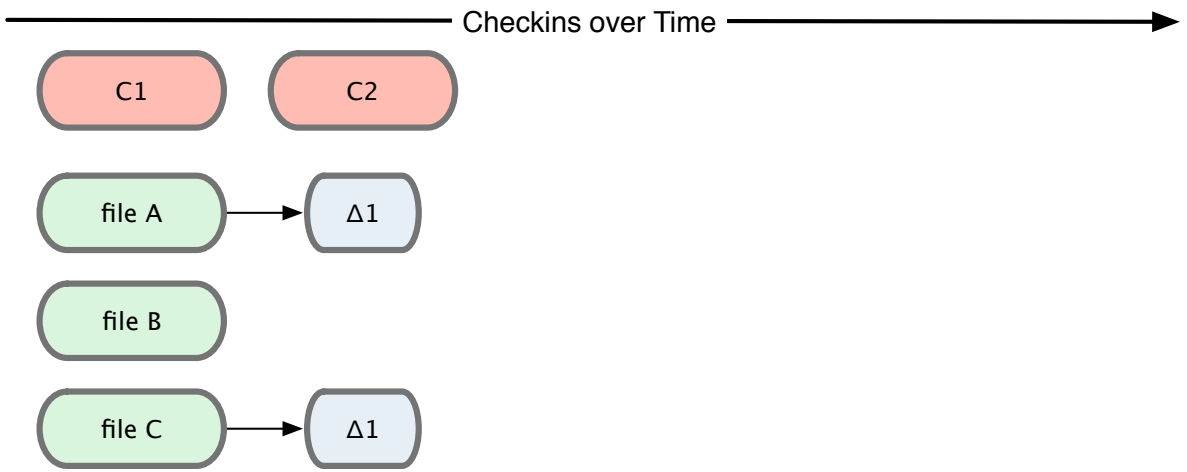


Everybody has the full history of commits

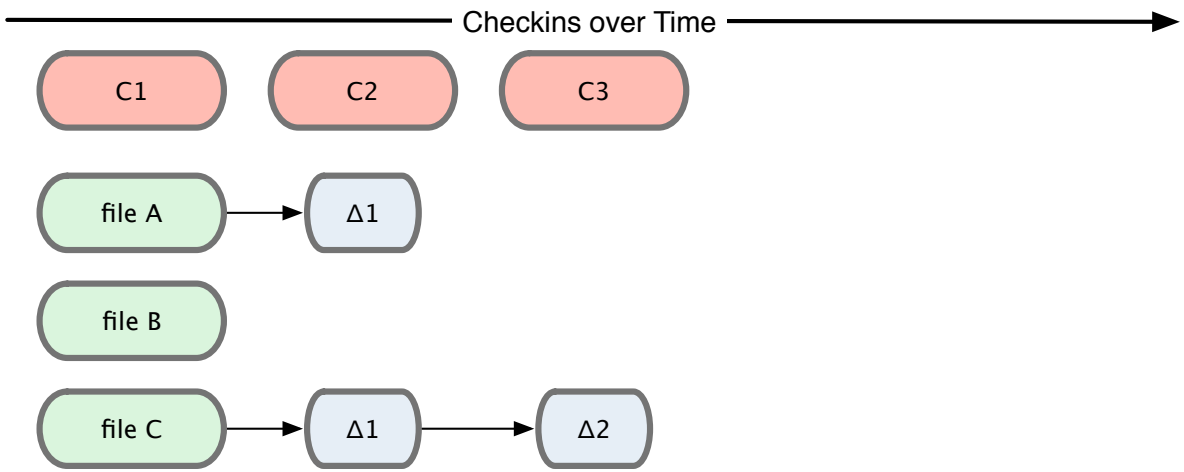
Tracking changes (most VCS)



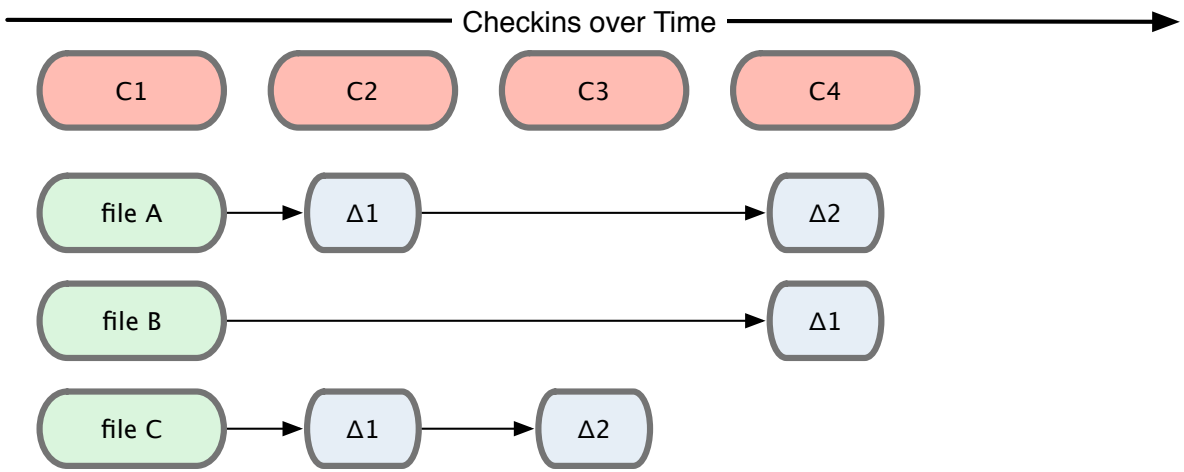
Tracking changes (most VCS)



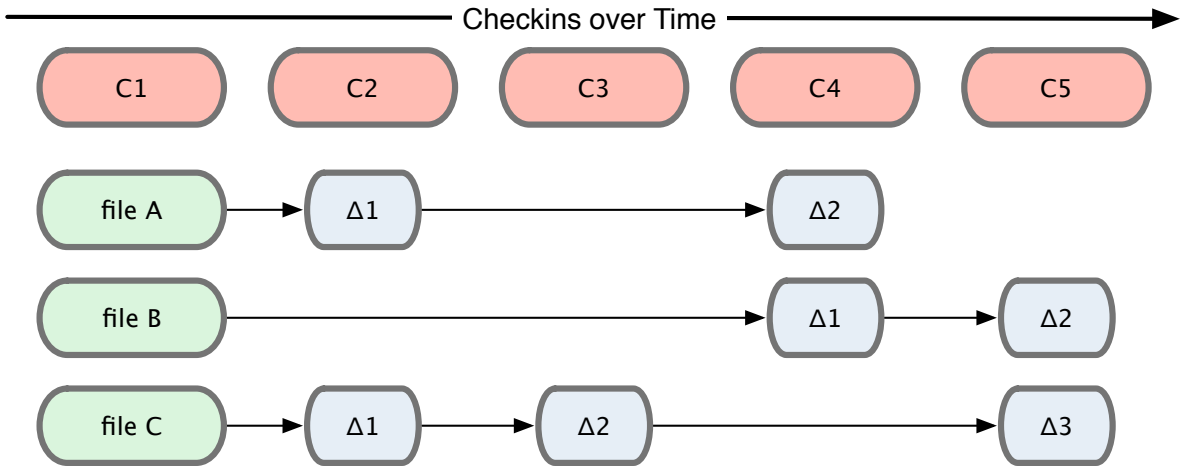
Tracking changes (most VCS)



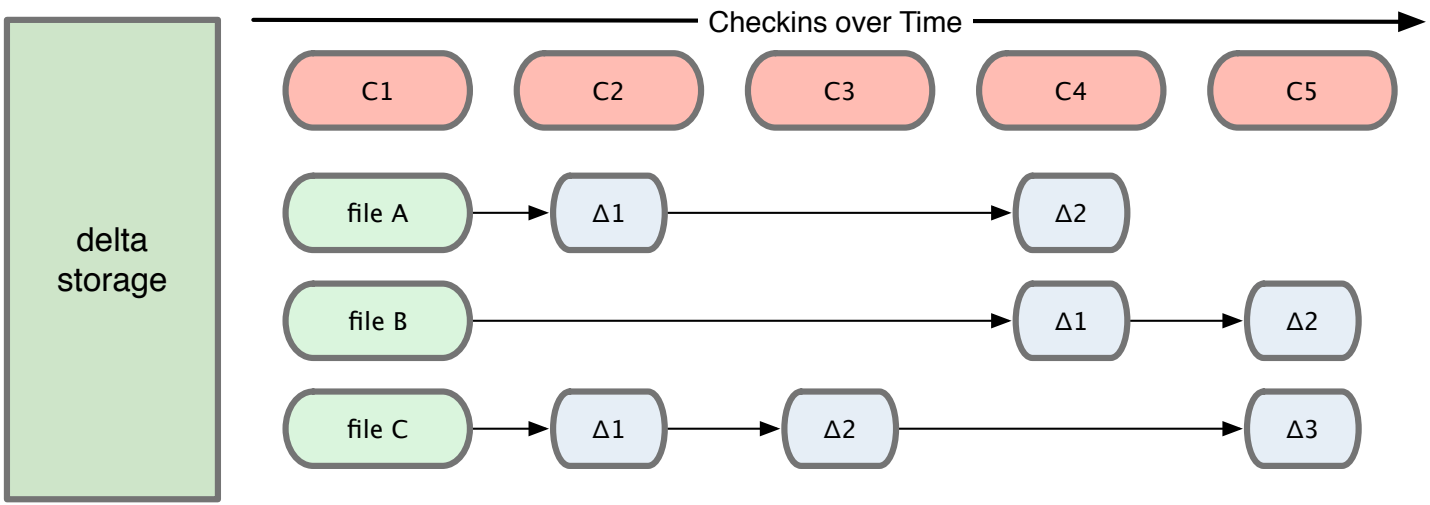
Tracking changes (most VCS)



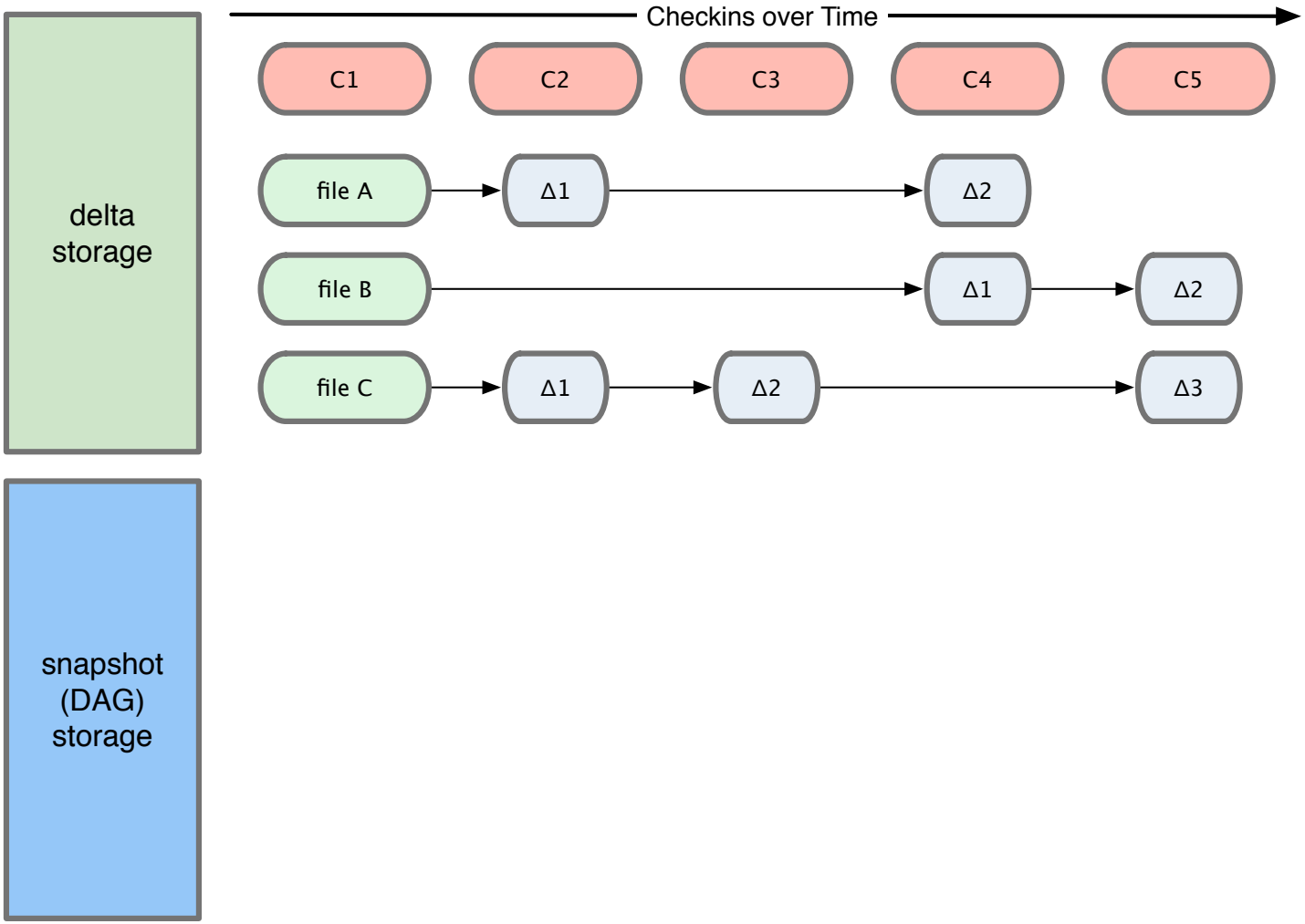
Tracking changes (most VCS)



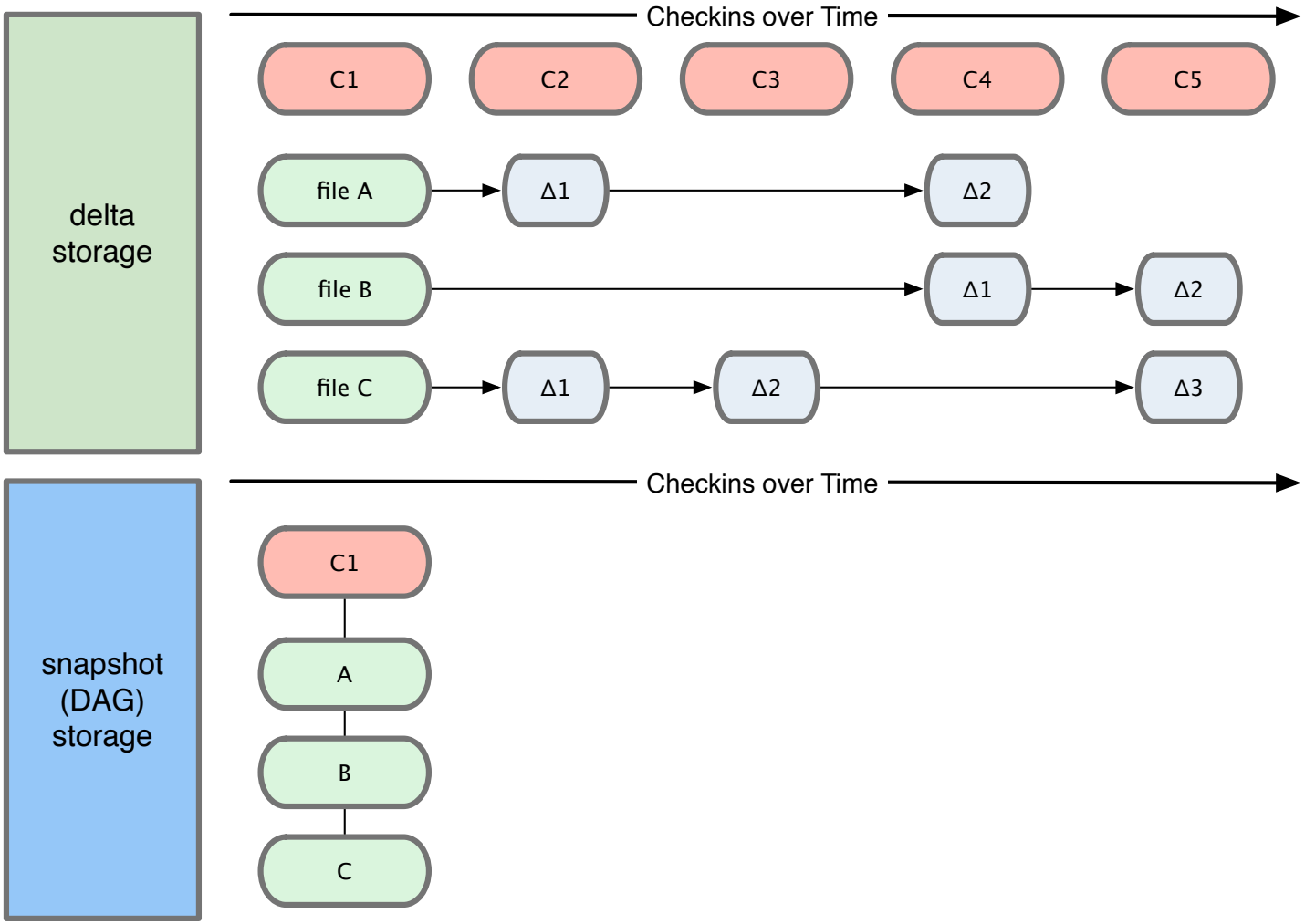
Tracking changes (most VCS)



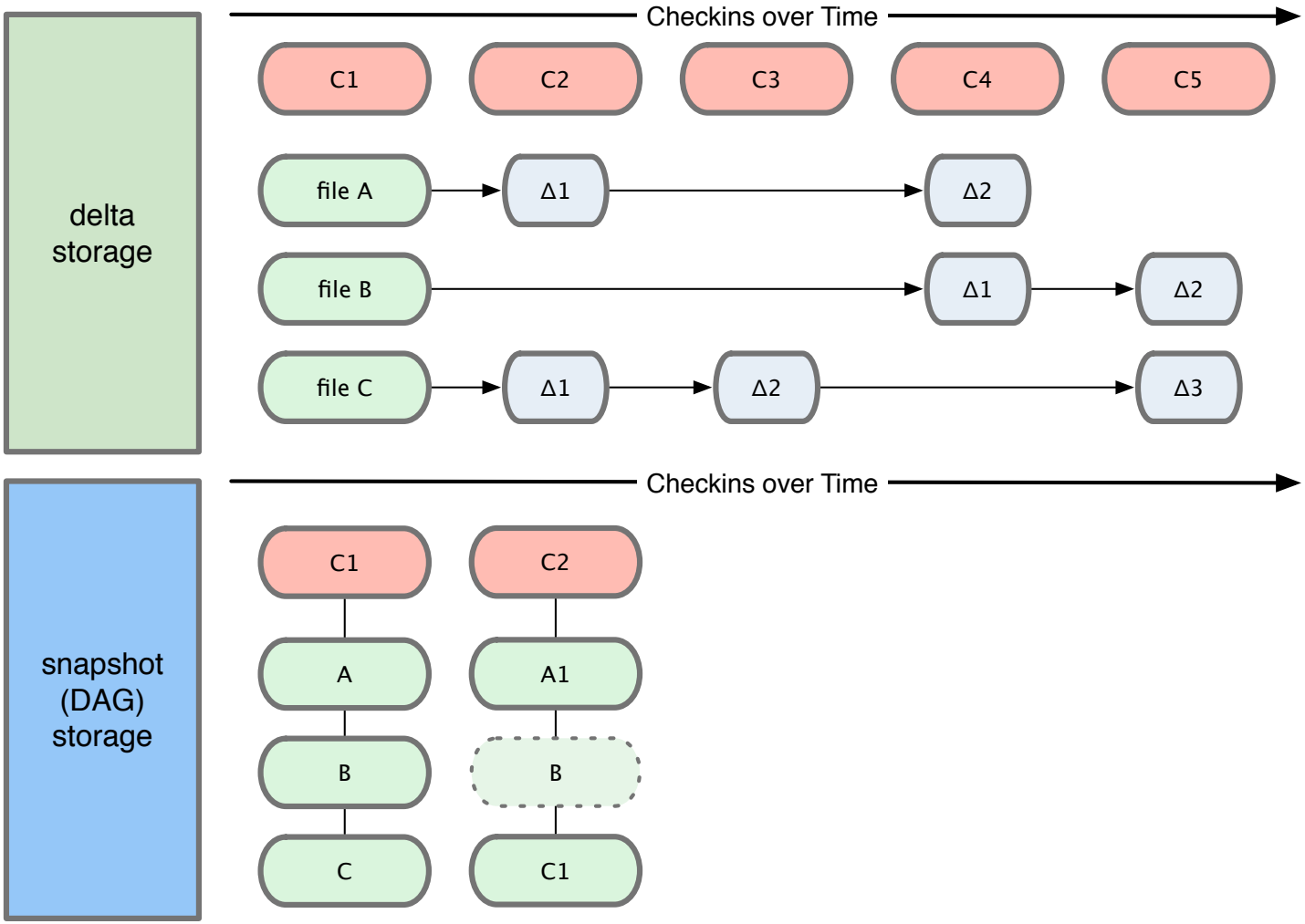
Tracking changes (Git)



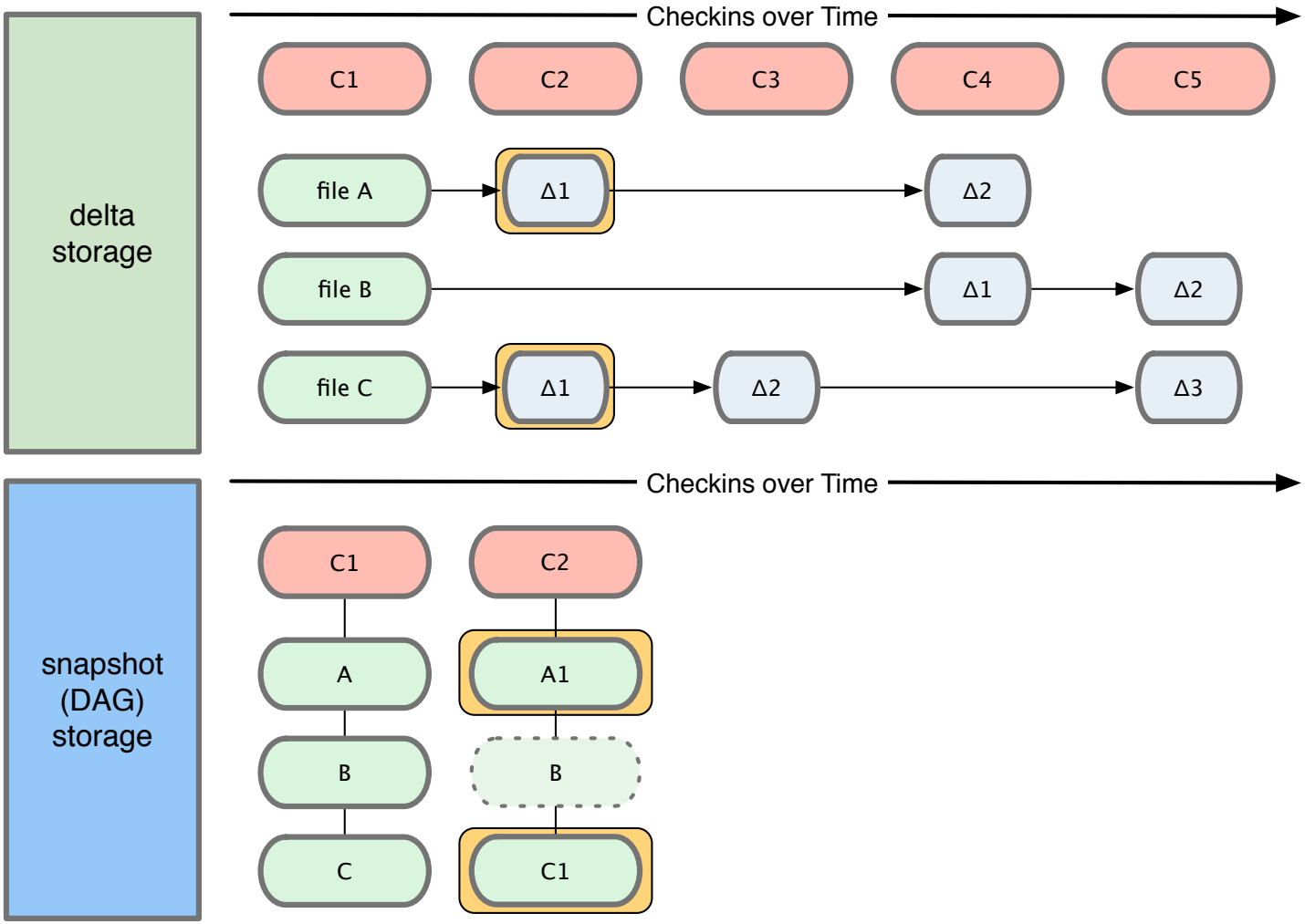
Tracking changes (Git)



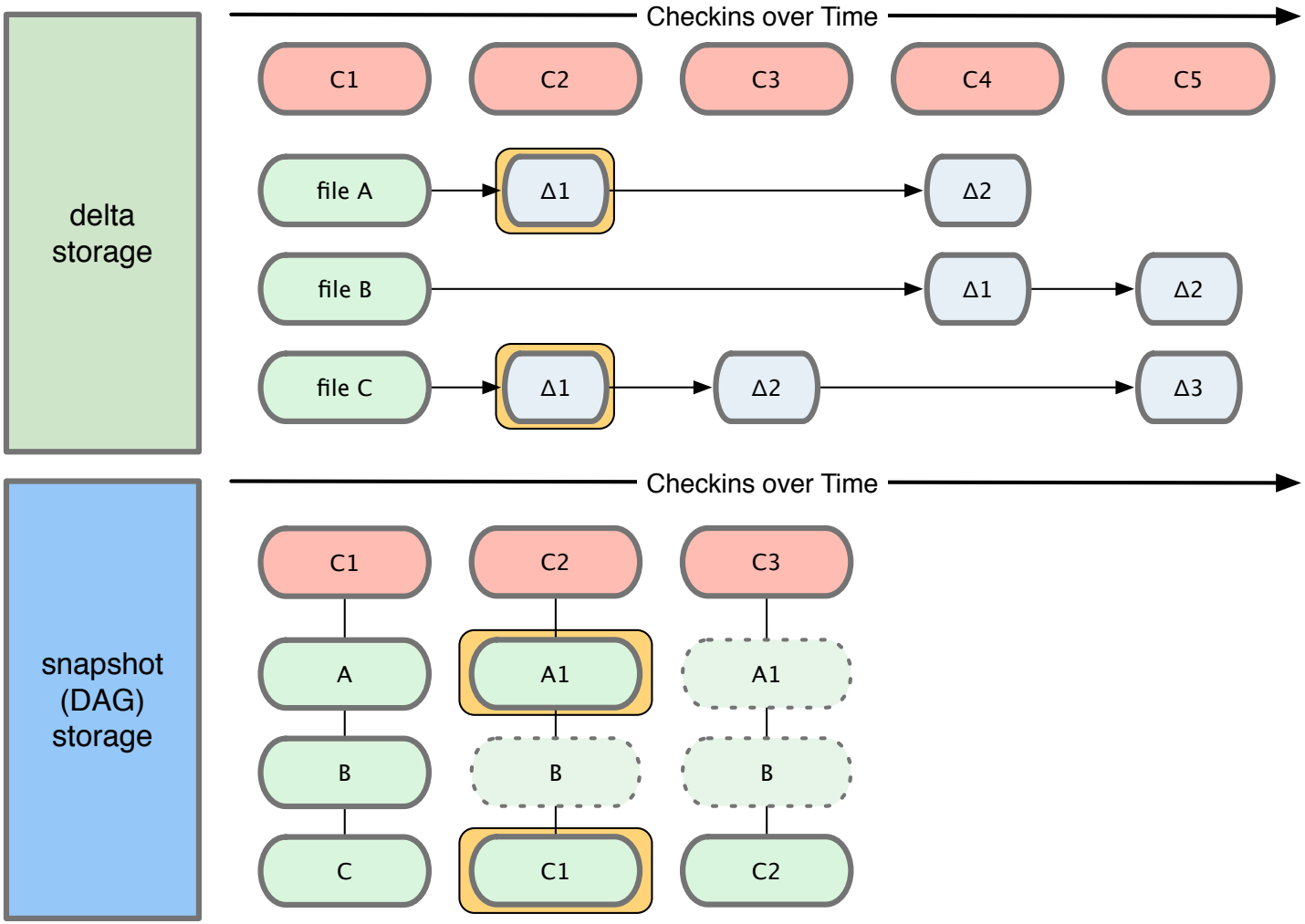
Tracking changes (Git)



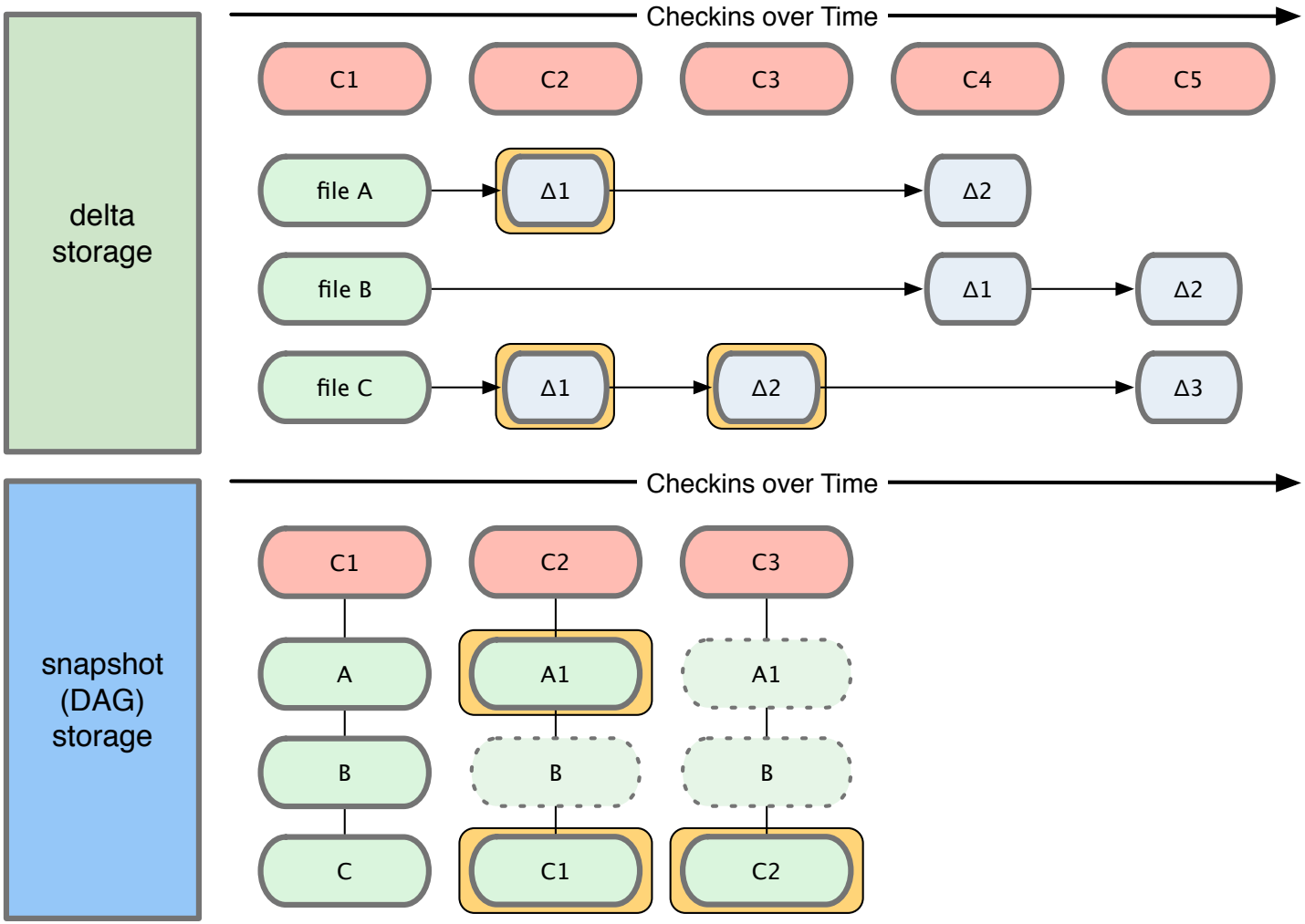
Tracking changes (Git)



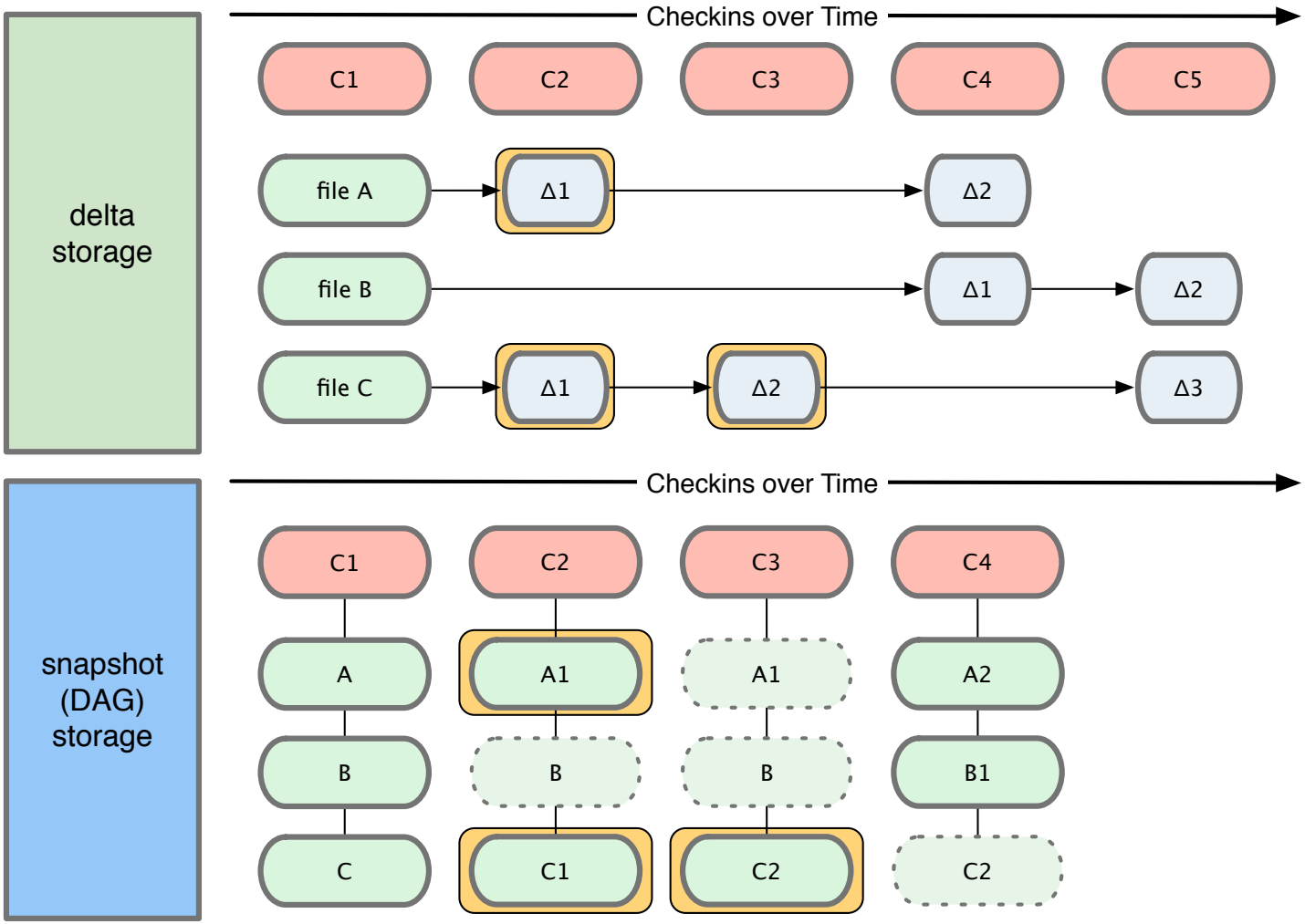
Tracking changes (Git)



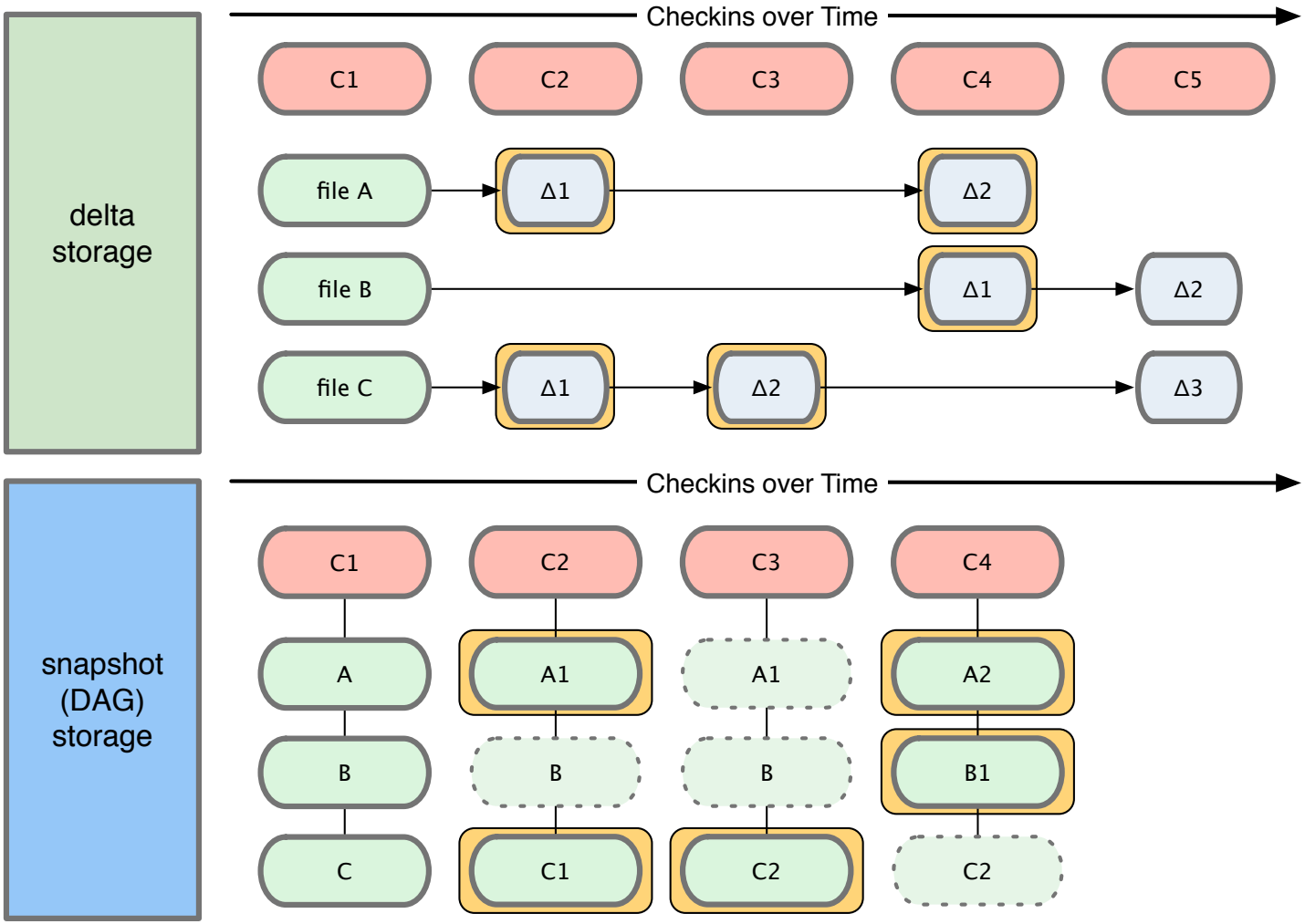
Tracking changes (Git)



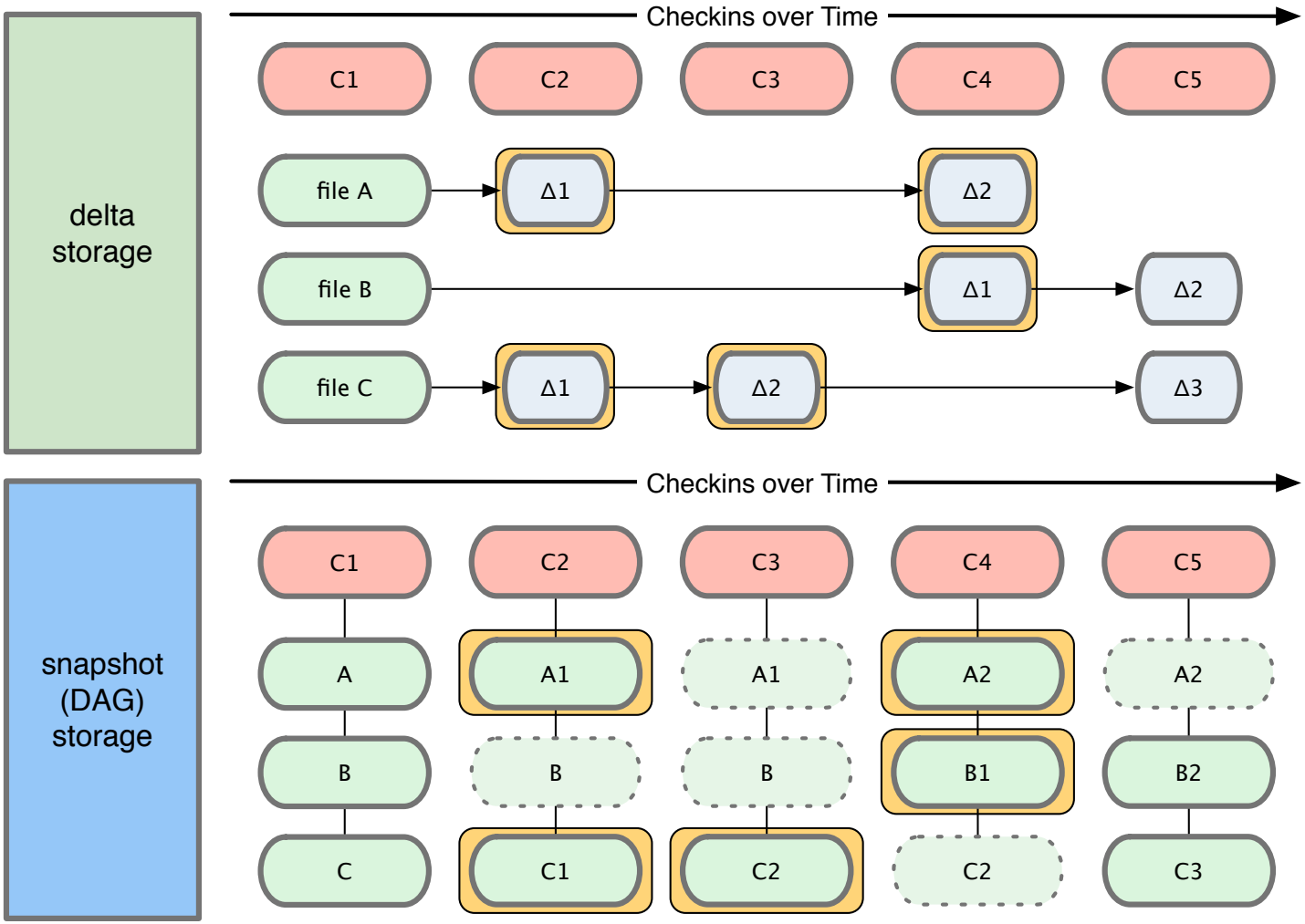
Tracking changes (Git)



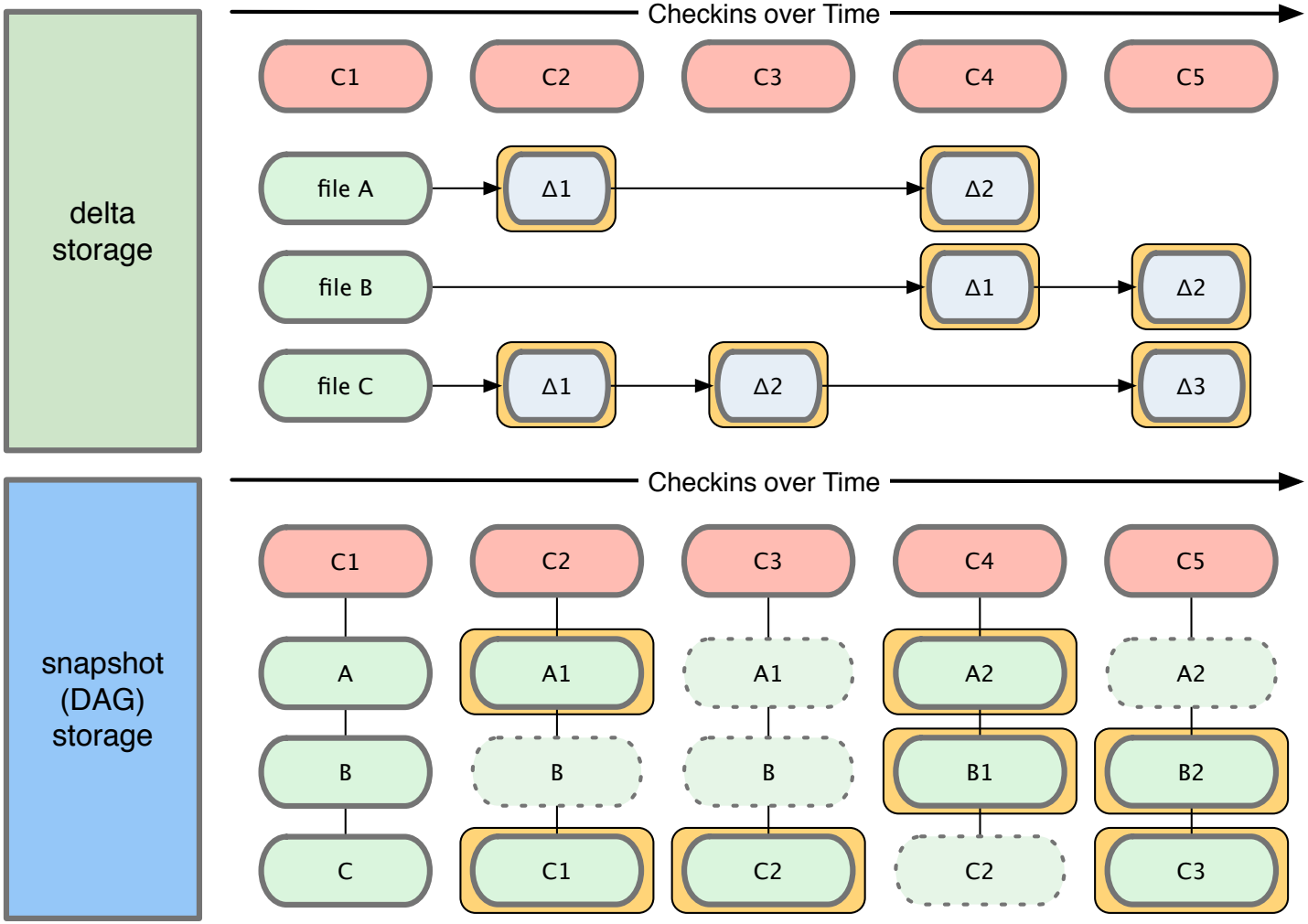
Tracking changes (Git)



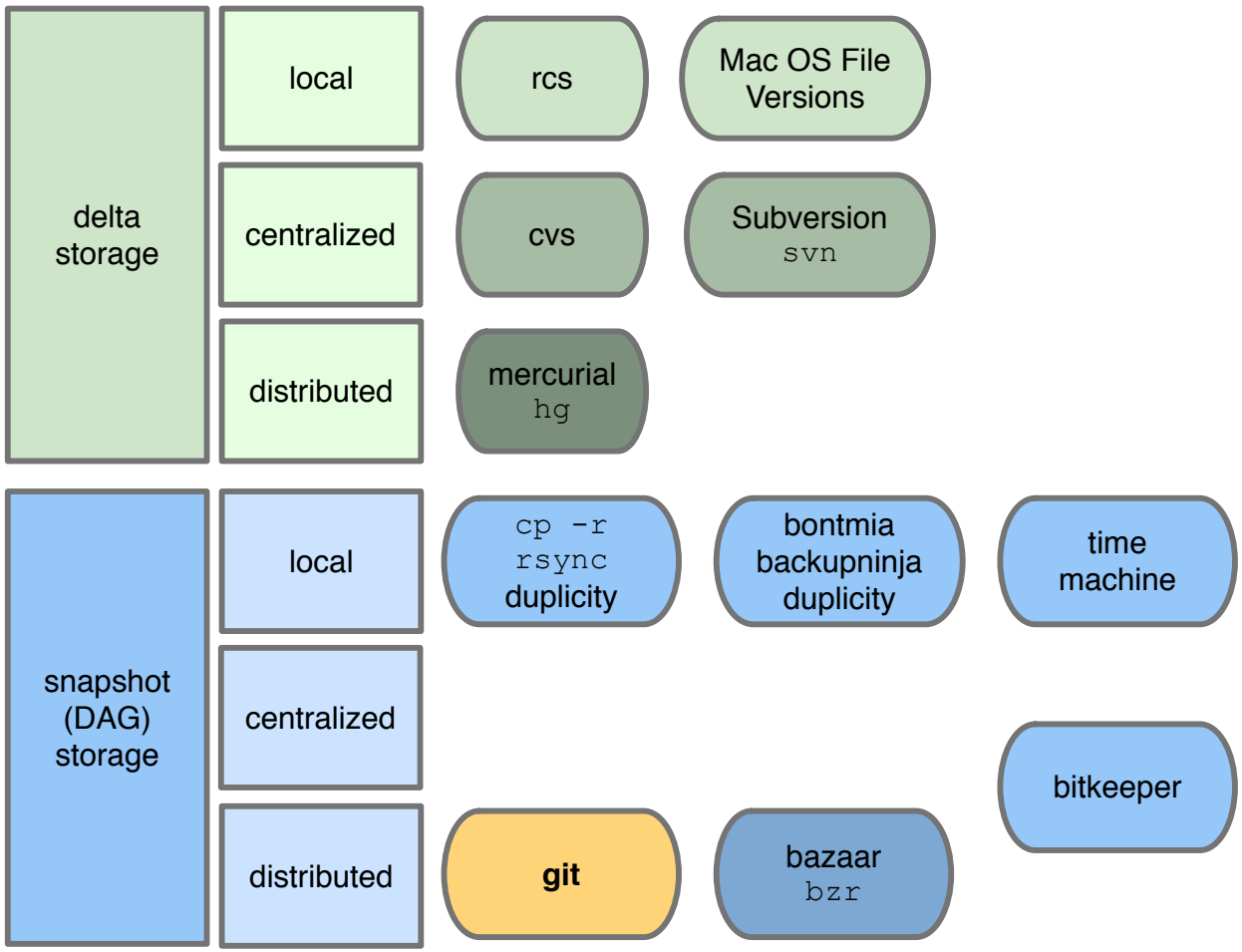
Tracking changes (Git)



Tracking changes (Git)



VCS Taxonomy



So what makes Git so useful?

(almost) Everything is local

- everything is fast
- every clone is a backup
- you work **mainly offline**

Ultra Fast, Efficient & Robust

- Snapshots, not patches (deltas)
- **Cheap branching and merging**
  - ↪ Strong support for thousands of parallel branches
- Cryptographic integrity everywhere

Other Git features

- **Git doesn't delete**
  - ↪ **Immutable** objects, Git generally only adds data
  - ↪ If you mess up, you can usually recover your stuff
    - ✓ Recovery can be tricky though

Other Git features

- **Git doesn't delete**
  - ↪ **Immutable** objects, Git generally only adds data
  - ↪ If you mess up, you can usually recover your stuff
    - ✓ Recovery can be tricky though

Git Tools / Extension

- cf. **Git submodules** or **subtrees**
- **Introducing git-flow**
  - ↪ workflow with a strict branching model
  - ↪ offers the `git` commands to follow the workflow

```
$> git flow init
$> git flow feature { start, publish, finish } <name>
$> git flow release { start, publish, finish } <version>
```

Summary

Summary

Installation Notes

git-scm.com

Linux / Mac OS

```
$> apt-get install git-core git-flow          # On Debian-like systems
$> yum install git gitflow                    # On CentOS-like systems
$> brew install git git-flow                  # On Mac OS, using Homebrew
```

Installation Notes

git-scm.com

Linux / Mac OS

```
$> apt-get install git-core git-flow        # On Debian-like systems
$> yum install git gitflow                  # On CentOS-like systems
$> brew install git git-flow                # On Mac OS, using Homebrew
```

Windows MsysGit

- Incl. Git Bash/GUI & Shell Integration
  - ↪ use PLINk from Putty
  - ↪ install **Git bash** + command prompt
  - ↪ select checkout windows / commit unix

Installation Notes

git-scm.com

Linux / Mac OS

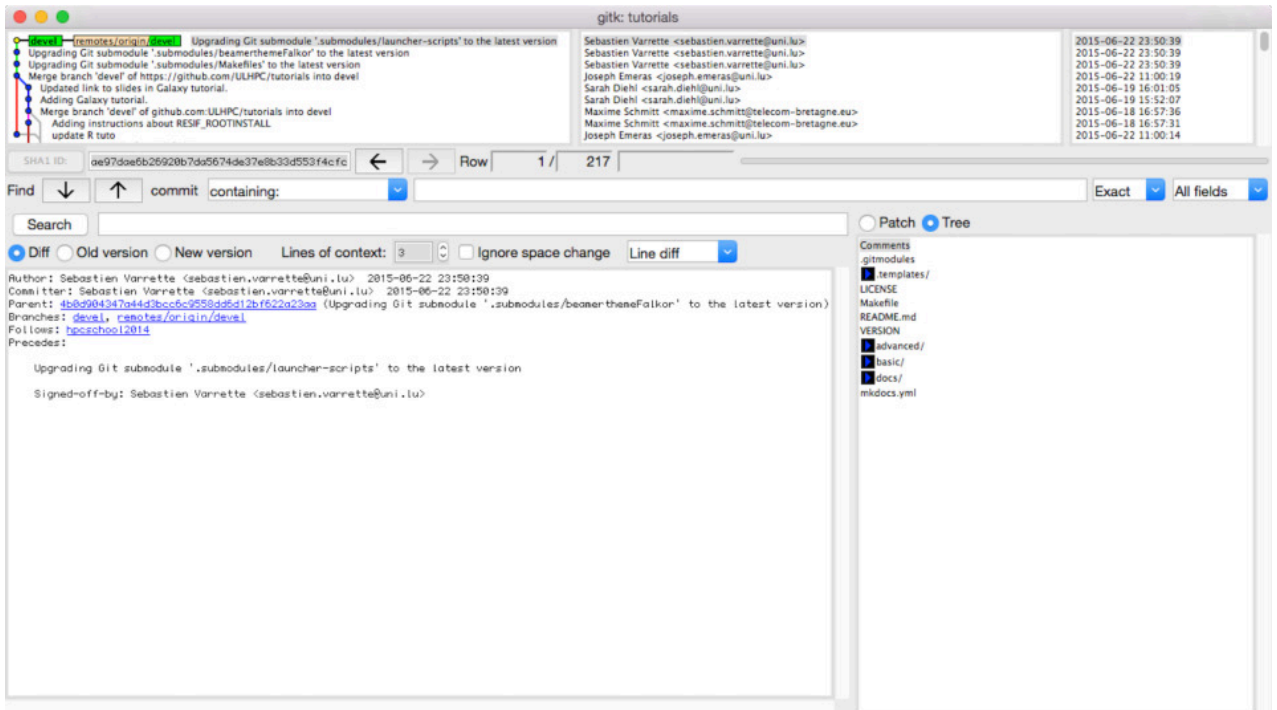
```
$> apt-get install git-core git-flow        # On Debian-like systems
$> yum install git gitflow                  # On CentOS-like systems
$> brew install git git-flow                # On Mac OS, using Homebrew
```

Windows MsysGit

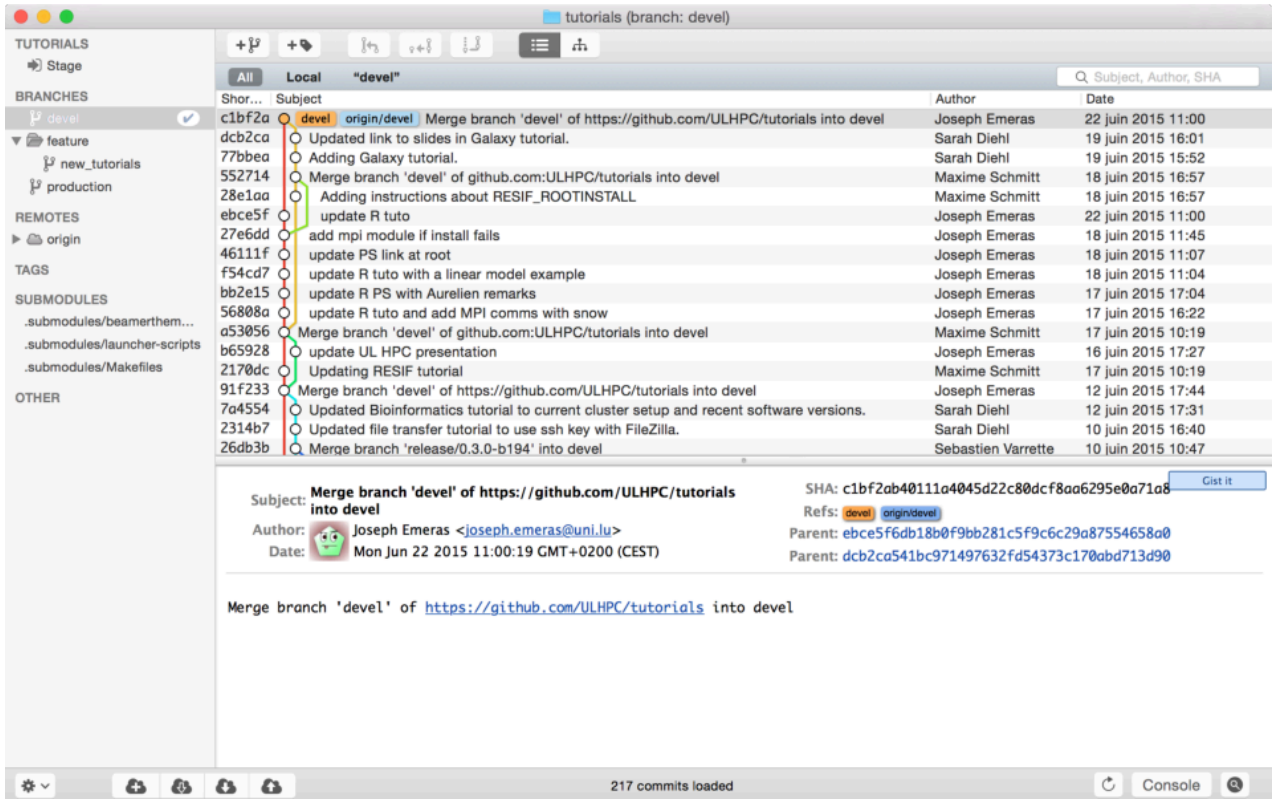
- Incl. Git Bash/GUI & Shell Integration
  - ↪ use PLINk from Putty
  - ↪ install **Git bash** + command prompt
  - ↪ select checkout windows / commit unix

Your Turn! Ensure you have git installed

Git GUI (default) Gitk

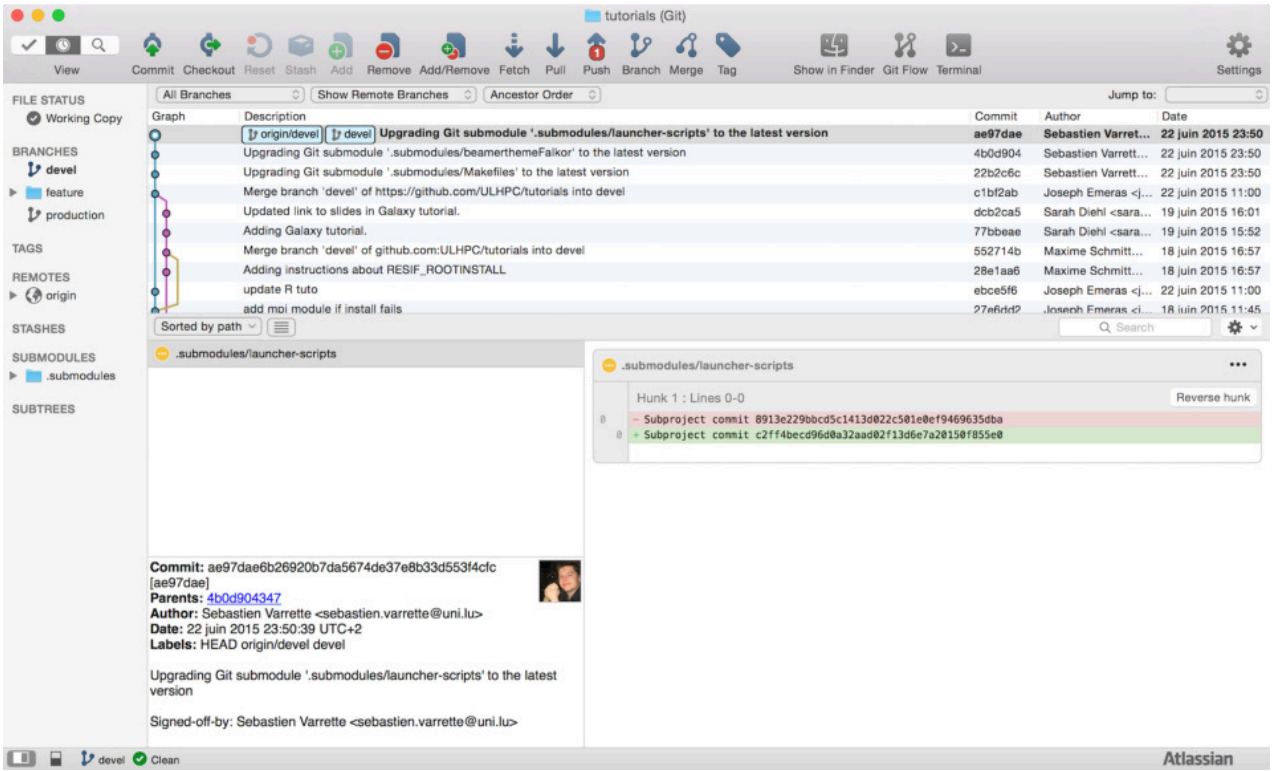


Git GUI (Mac OS) GitX-dev



http://rowanj.github.io/gitx/

Git GUI (Windows/Mac) SourceTree



http://www.sourcetreeapp.com/

1. Let it install a default git ignore file
2. make it load your SSH key created with Putty

Preliminary Configurations

- Global Git configuration are stored in `~/.gitconfig`
  - ↪ **Ex**: see my personal `.gitconfig`
- You **SHOULD** at least configure your name and email to commit
  - ↪ open a terminal (Git bash under windows) for the below commands

```
$> git config --global user.name "Firstname LastName"
$> git config --global user.email "Firstname.LastName@uni.lu"
$> git config --global color.ui true                    # Colors
$> git config --global core.editor vim                  # Editor
```

Preliminary Configurations

- Global Git configuration are stored in `~/.gitconfig`
  - ↪ **Ex**: see my personal `.gitconfig`
- You **SHOULD** at least configure your name and email to commit
  - ↪ open a terminal (Git bash under windows) for the below commands

```
$> git config --global user.name "Firstname LastName"
$> git config --global user.email "Firstname.LastName@uni.lu"
$> git config --global color.ui true                    # Colors
$> git config --global core.editor vim                  # Editor
```

Your Turn!

- Then check the changes by: `git config -l | grep user`

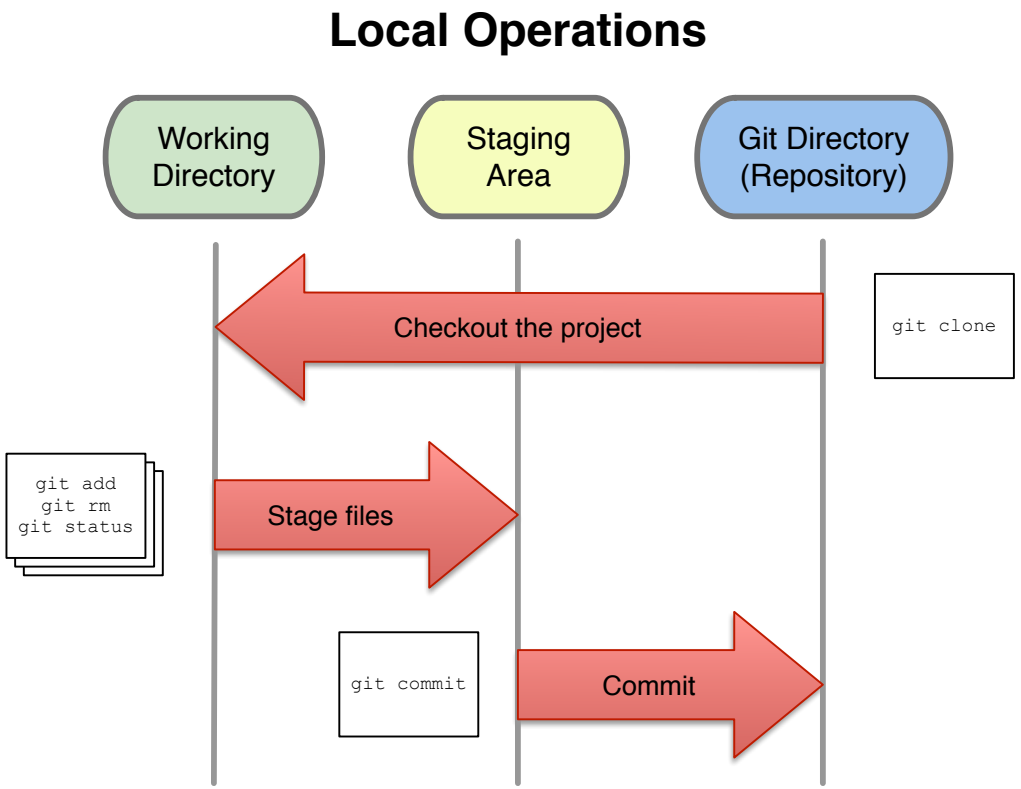
Git Commands Aliases

- You can also create git command aliases in `~/.gitconfig`.
  - ↪ **Ex** copy/paste from my personal `.gitconfig`

```
[alias]
    up = pull origin
    pu = push origin
    st = status
    df = diff
    ci = commit -s
    co = checkout
    br = branch
    w  = whatchanged --abbrev-commit
    ls = ls-files
    gr = log --graph --oneline --decorate
    amend = commit --amend
```

Summary

The Three (local) States



- The local repository lives in the `.git` directory.
- The **staging area** tracks what will go into the next commit
  - ↪ AKA "the index"

Summary

Creating a Repository

```
$> git [flow] init
```

- Initializes a new git (flow) repository in the current directory

Creating a Repository

```
$> git [flow] init
```

- Initializes a new git (flow) repository in the current directory

Your Turn!

```
$> cd /tmp
$> mkdir firstproject
$> cd firstproject

$> git init
Initialized empty Git repository in /private/tmp/firstproject/.git/
```

Cloning a Repository

```
$> git clone [--recursive] <url> [<path>]
```

Type	URL Format / Example	Port
Local	`/path/to/project.git`	n/a
SSH	`git+ssh://user@server:port/project.git`	22
Git	`git://server/project.git`	9418
HTTPS	`https://github.com/Falkor/falkorlib.git`	443

Cloning a Repository

```
$> git clone [--recursive] <url> [<path>]
```

Your Turn!

```
$> cd /tmp
$> git clone https://github.com/ULHPC/tutorials.git
Cloning into 'tutorials'...
remote: Counting objects: 1247, done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 1247 (delta 32), reused 0 (delta 0), pack-reused 1181
Receiving objects: 100% (1247/1247), 15.74 MiB | 3.08 MiB/s, done.
Resolving deltas: 100% (588/588), done.
Checking connectivity... done.
$> git clone --recursive \
       https://github.com/ULHPC/tutorials.git /tmp/tutorials2
```

Inspecting a Repository

```
$> git status [-s]                    # -s:  short / simplified output
```

Inspecting a Repository

```
$> git status [-s]                    # -s:  short / simplified output
```

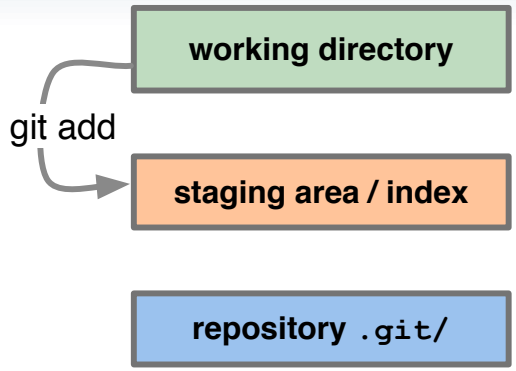
Your Turn!

```
$> cd /tmp/firstproject
$> git status
On branch master

Initial commit

nothing to commit

# Create an empty file
$> touch README.md
```

```
$> git status
On branch master

Initial commit

Untracked files:
    README

nothing added to commit but untracked
files present
$> git status -s
?? README
```

Add / Tracking [new] file(s)

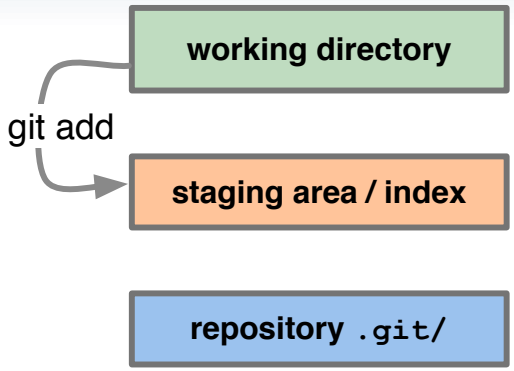
```
$> git add [-f] <pattern>
```



- Adds changes to the index
  - ↪ Add a specific file: `git add README`
  - ↪ Add a set of files: `git add *.py`
- Beware that empty directory cannot be added **directly**
  - ↪ due to the internal file representation (**blobs**)
  - ↪ **Tips**: add an hidden file `.empty` (or `.gitignore`)

Add / Tracking [new] file(s)

```
$> git add [-f] <pattern>
```



- Adds changes to the index
  - ↪ Add a specific file: `git add README`
  - ↪ Add a set of files: `git add *.py`
- Beware that empty directory cannot be added **directly**
  - ↪ due to the internal file representation (**blobs**)
  - ↪ **Tips**: add an hidden file `.empty` (or `.gitignore`)

Your Turn!

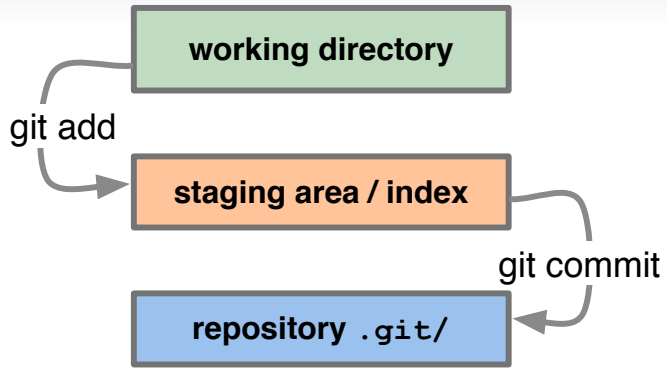
```
$> cd /tmp/firstproject
$> git status -s
?? README
```

```
$> git add README
$> git status -s
A  README
```

Committing your changes

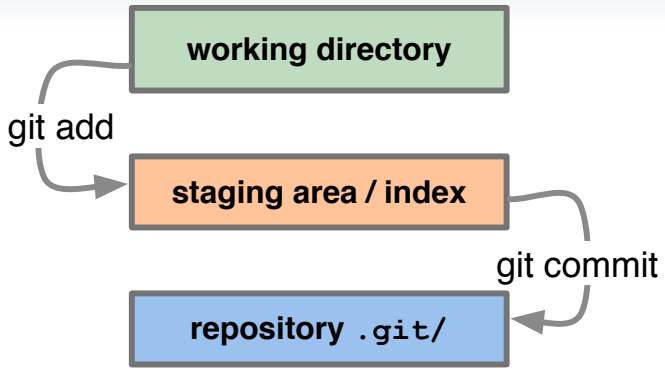
```
$> git commit [-s] [-m "msg"]
```

- Commit all changes: `git commit -a`



Committing your changes

```
$> git commit [-s] [-m "msg"]
```



- Commit all changes: `git commit -a`

Your Turn!

```
$> cd /tmp/firstproject
$> git commit -s -m "add README"    # OR git ci -m "add README"
[master (root-commit) ee60f53] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README
$> git status               # OR git st
On branch master
nothing to commit, working directory cleant
```

Removing Files

```
$> git rm [-rf] [--cached] <file>
```

- `--cached`: remove from Staging area
  - ↪ otherwise (default): from index **and** file system

Ignoring Files

Ignoring files from staging: '.gitignore'

- you can create a `.gitignore` file listing patterns to ignore
  - ↪ Blank lines or lines starting with \# are ignored
  - ↪ End pattern with slash (/) to specify a directory
  - ↪ Negate pattern with exclamation point (!)
- Collection of useful `.gitignore` templates

```
.DS_Store
*~
```

```
*.asv
*.m~
*.mex*
tmp/*
```

- LaTeX `.gitignore`
- Python `.gitignore`
- Ruby `.gitignore`

Moving Files

```
$> git mv <source> <destination>
```

```
# Equivalent of:
mv <source> <destination>
git rm <source>
git add <destination>
```

Moving Files

```
$> git mv <source> <destination>
```

```
# Equivalent of:
mv <source> <destination>
git rm <source>
git add <destination>
```

Your Turn!

```
$> cd /tmp/firstproject
$> git mv README README.md
$> git status
On branch master
Changes to be committed:
    renamed:    README -> README.md
$> git commit -m "a first move"
```

Check the Commit History

```
$> git log [-p] [--stat] [--graph --oneline --decorate]
```

- `-p` / `--stat`: show the differences introduced in each commit
- You can also perform some date filtering

  ```
  $> git log --since=2.weeks
  ```

- Ncurses-based text-mode interface: `tig`

Check the Commit History

```
$> git log [-p] [--stat] [--graph --oneline --decorate]
```

- `-p` / `--stat`: show the differences introduced in each commit
- You can also perform some date filtering

  ```
  $> git log --since=2.weeks
  ```

- Ncurses-based text-mode interface: `tig`

Your Turn!

```
$> cd /tmp/firstproject
$> git log --oneline --graph --decorate     # OR git gr
* f1f0c27 (HEAD -> master) a first move
* ee60f53 add README
$> git log -p -1          # only the last commit OR git show
$> tig
```

Show differences

```
$> git diff [--cached] [<ref>]
```

- Check **un-staged** changes: `git diff`
  - ↪ `--cached`: check **staged** changes
- Relative to a specific revision:

```
$> git diff 1776f5
$> git diff HEAD^
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

- Restore to the last committed/cloned version: **all** changes are lost!

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

```
$> git revert <commit>                   # revert a <commit>
```

- Make a new commit that undoes all changes made in <commit>

Undoing Things

```
$> git commit --amend                    # Change the last commit
```

```
$> git unstage <file>                    # or git reset HEAD <file>
```

```
$> git checkout -- <file>    # DANGER! Un-modify modified file
```

```
$> git revert <commit>                   # revert a <commit>
```

Your Turn!

```
$> cd /tmp/firstproject
$> git commit --amend
$> echo 'toto' >> README.md
```

```
$> cat README.md && git status
$> git checkout -- README
$> git status
```

Summary

Basic Workflow

Edit files	`vim` / `emacs` / `subl` ...
Stage the changes	`git add`
Review your changes	`git status`
Commit the changes	`git commit`

Summary

For cheaters: A Basicerer Workflow

Edit files	`vim` / `emacs` / `subl` ...
Stage & commit the changes	`git commit -a`

Summary

For cheaters: A Basicerer Workflow

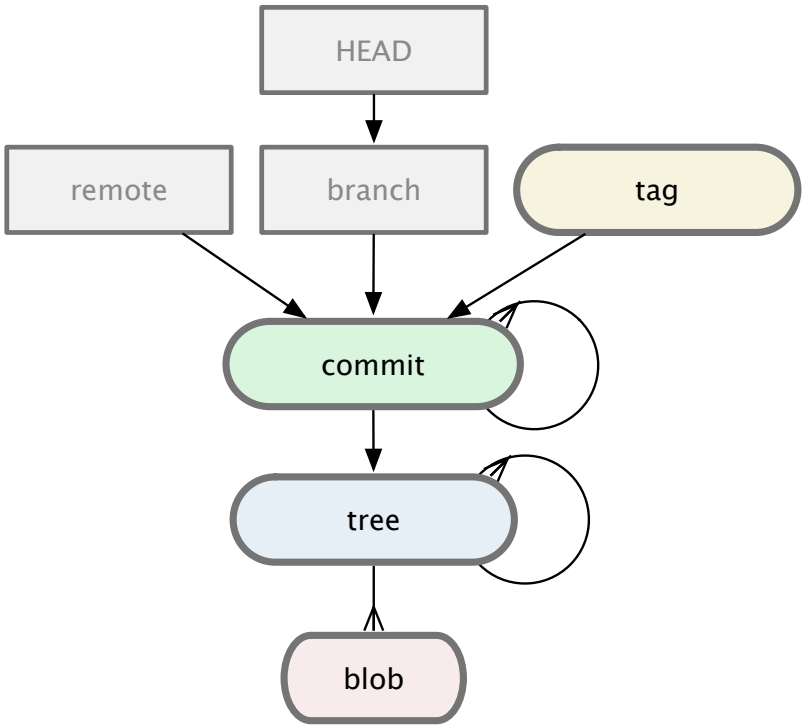
Edit files	`vim` / `emacs` / `subl` ...
Stage & commit the changes	`git commit -a`

Advices

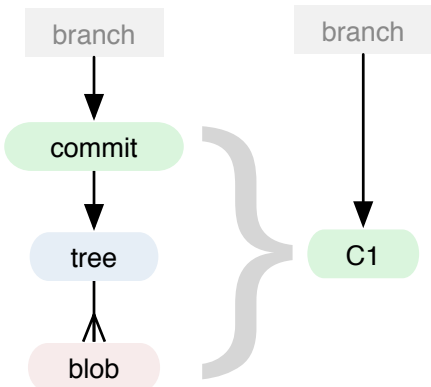
- **Commit early, commit often!**
  - ↪ commits = save points
  - ↪ use descriptive commit messages
- Don't get out of sync with your collaborators
- Commit the sources, not the derived files

Summary

Data Model



- **Immutable** objects
  - ↪ **Blob**: File content
  - ↪ **Tree**: Directory List
  - ↪ **Commit**: Pointer to a snapshot / tree
  - ↪ **Tag**: Pointer to commit
- **Git Branch**: Lighweight, movable pointer to a commit (`HEAD`: current branch)



Data Model Example

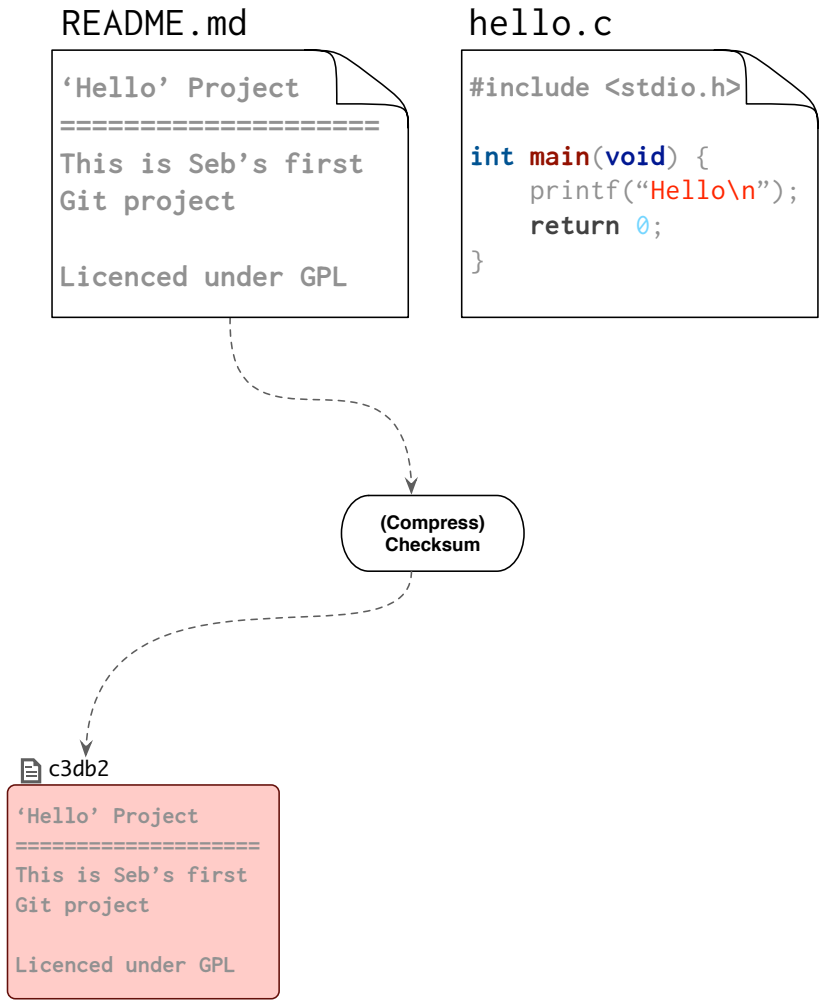
README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

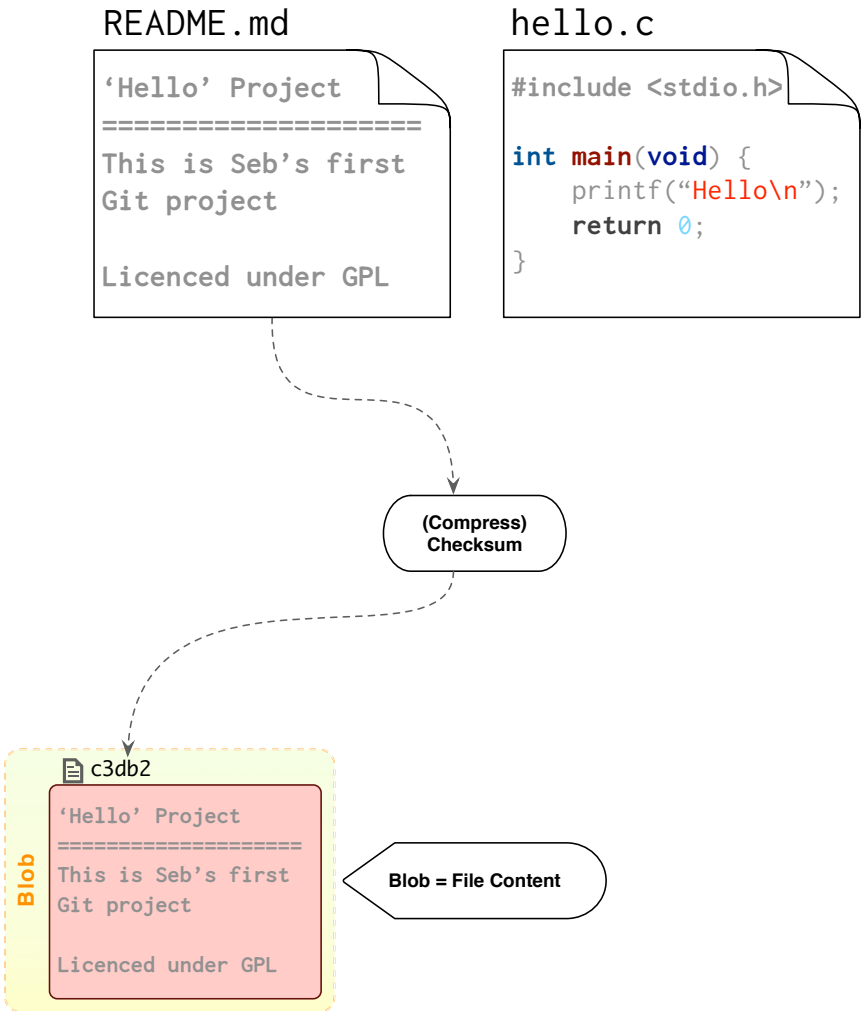
hello.c

```c
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```

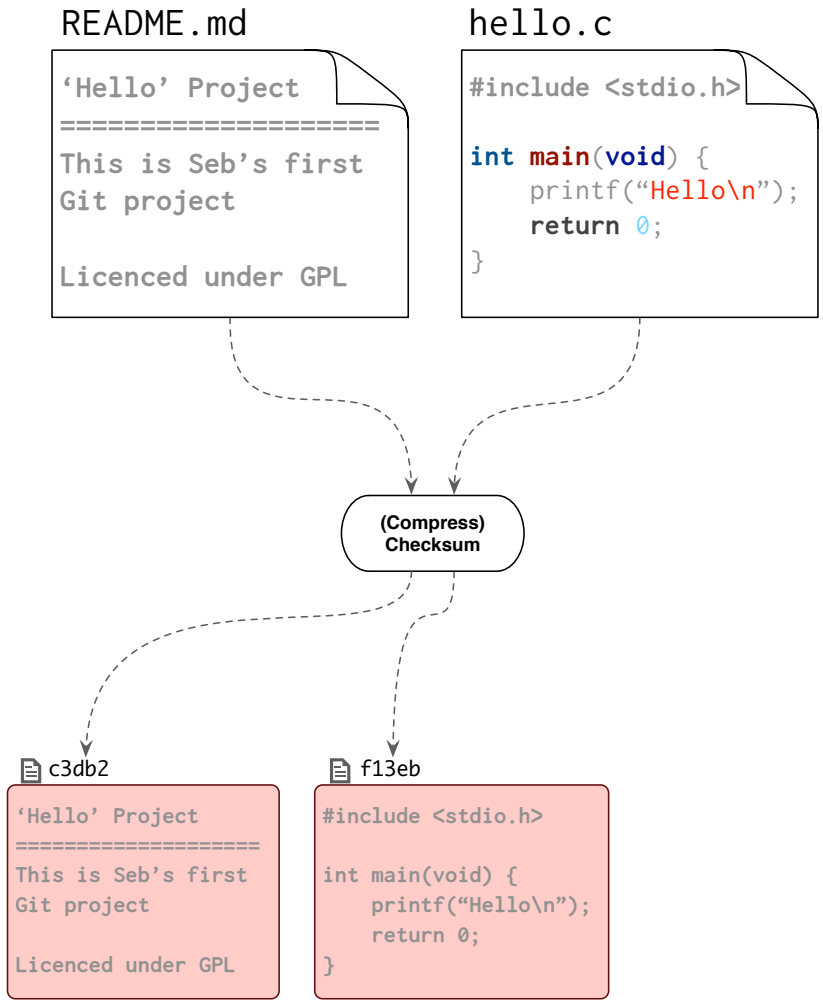
Data Model Example



Data Model Example



Data Model Example



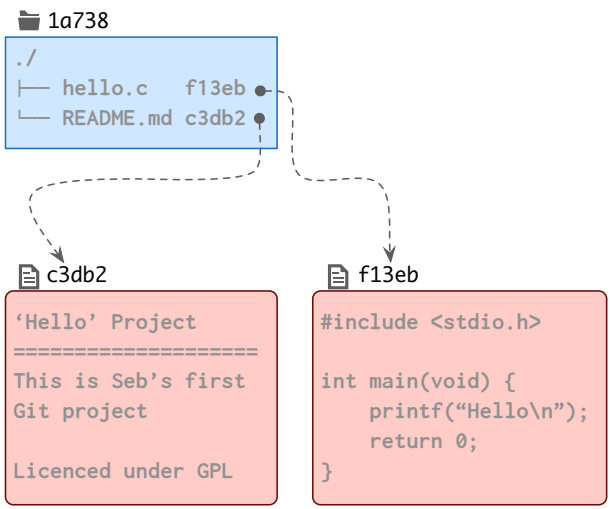
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```c
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



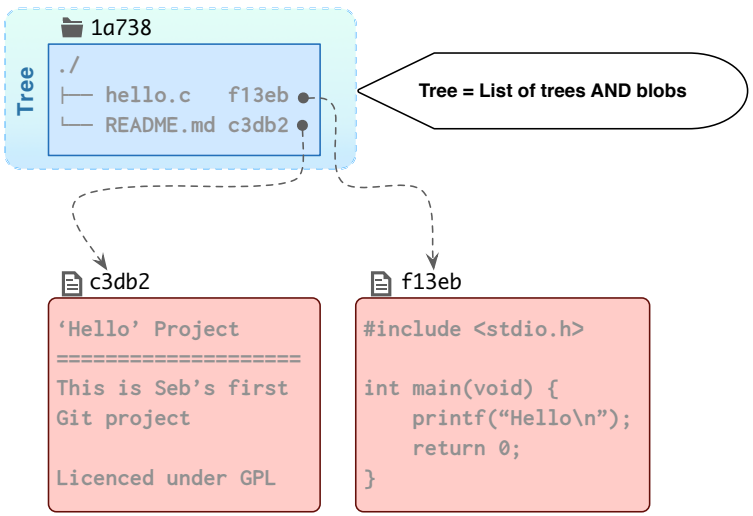
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```c
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



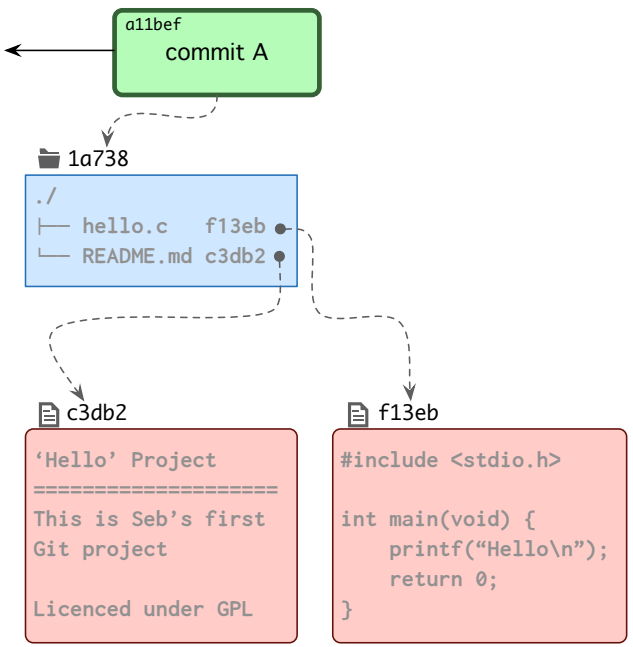
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```c
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



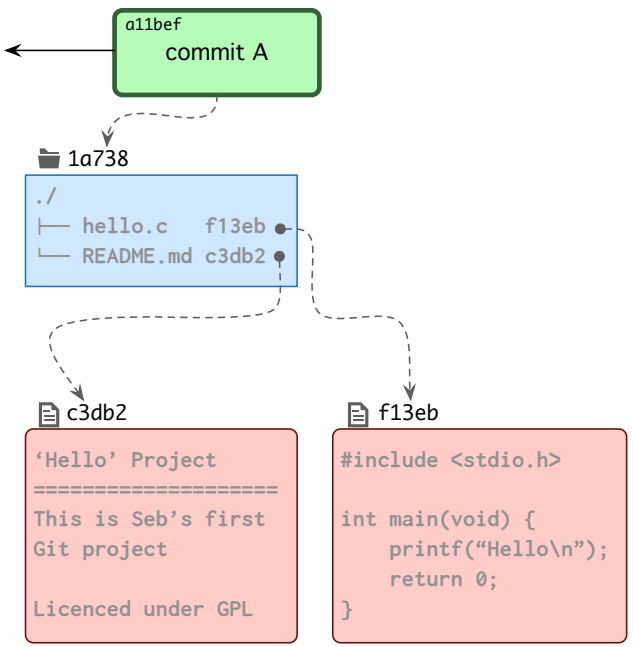
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```c
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



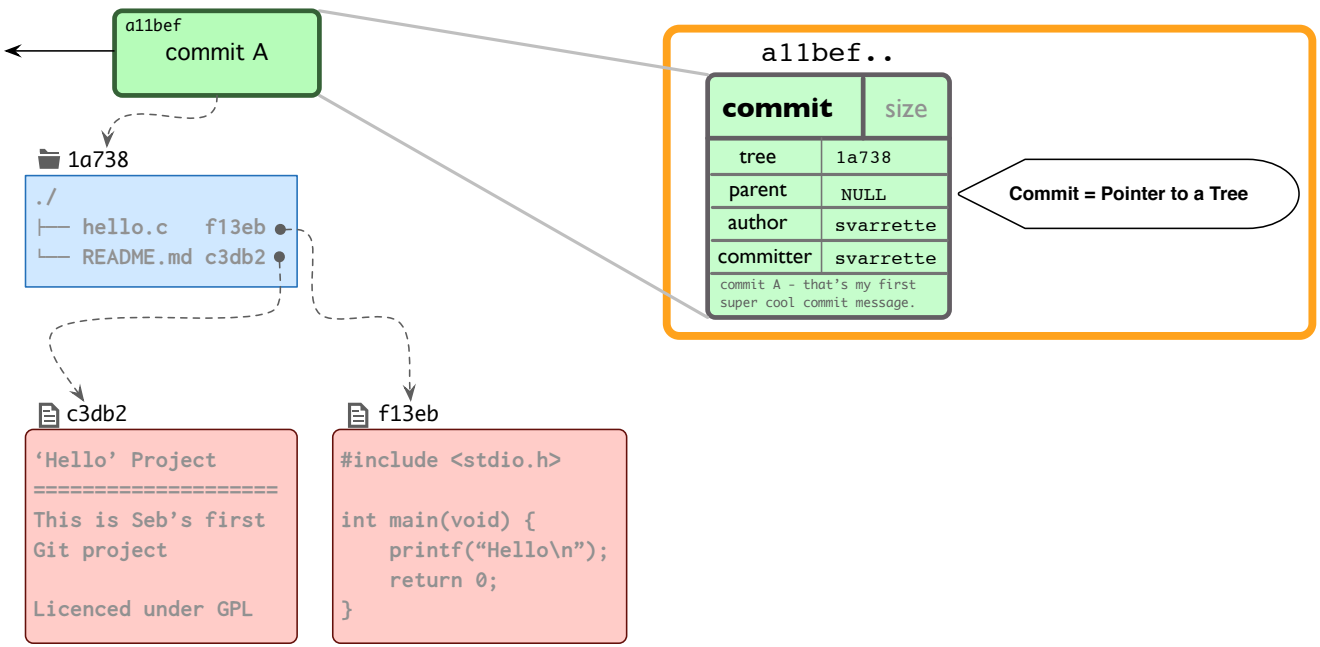
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



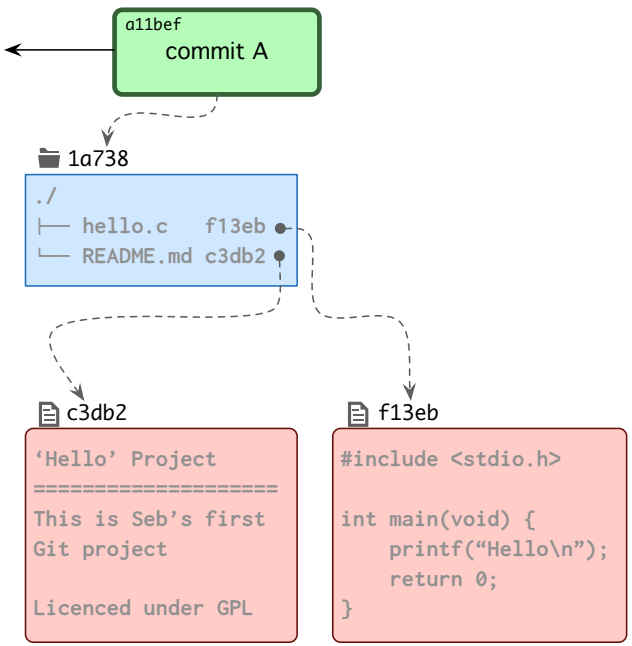
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hello\n");
    return 0;
}
```



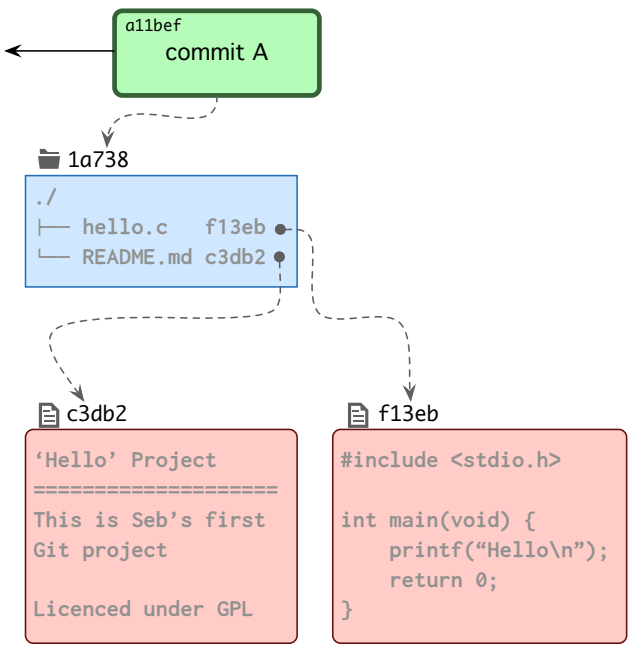
Data Model Example

README.md

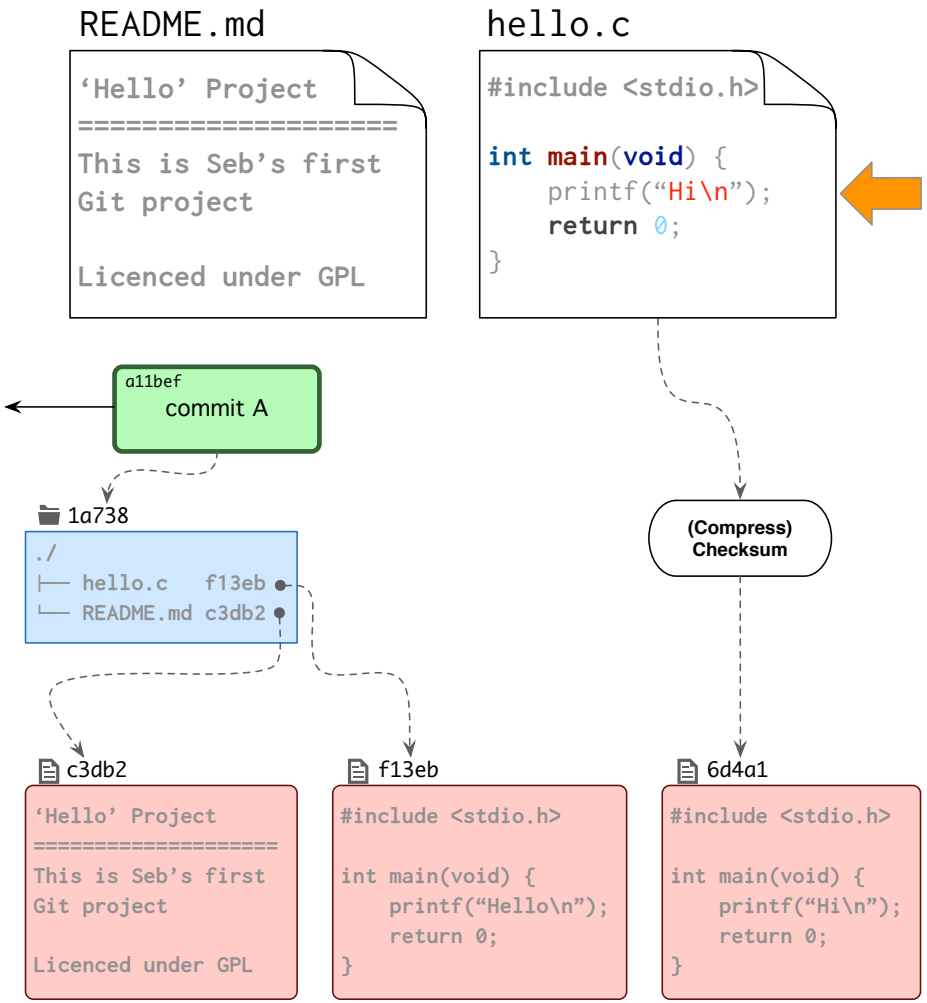
```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



Data Model Example



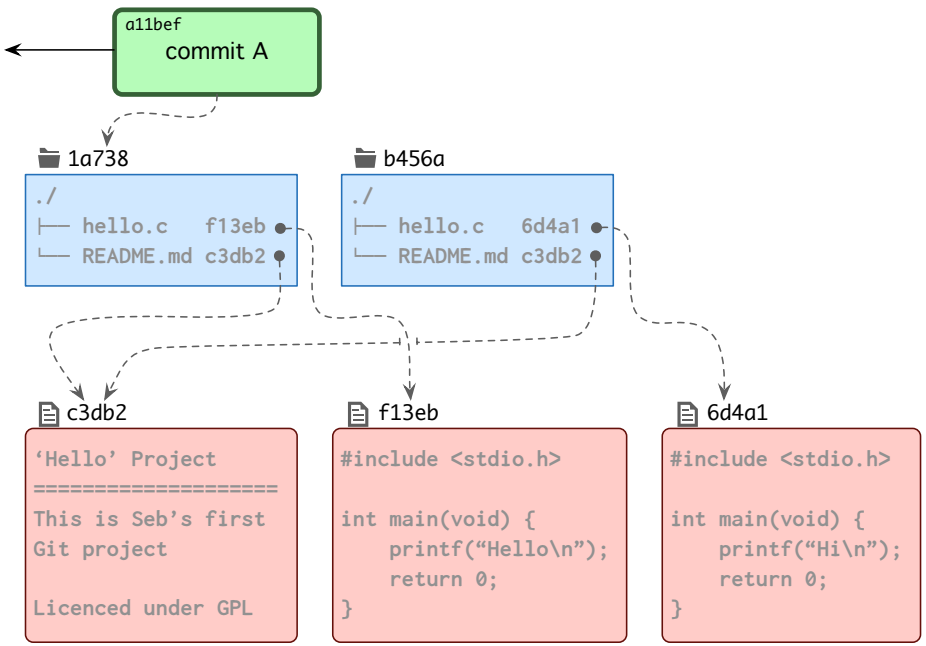
Data Model Example

README.md

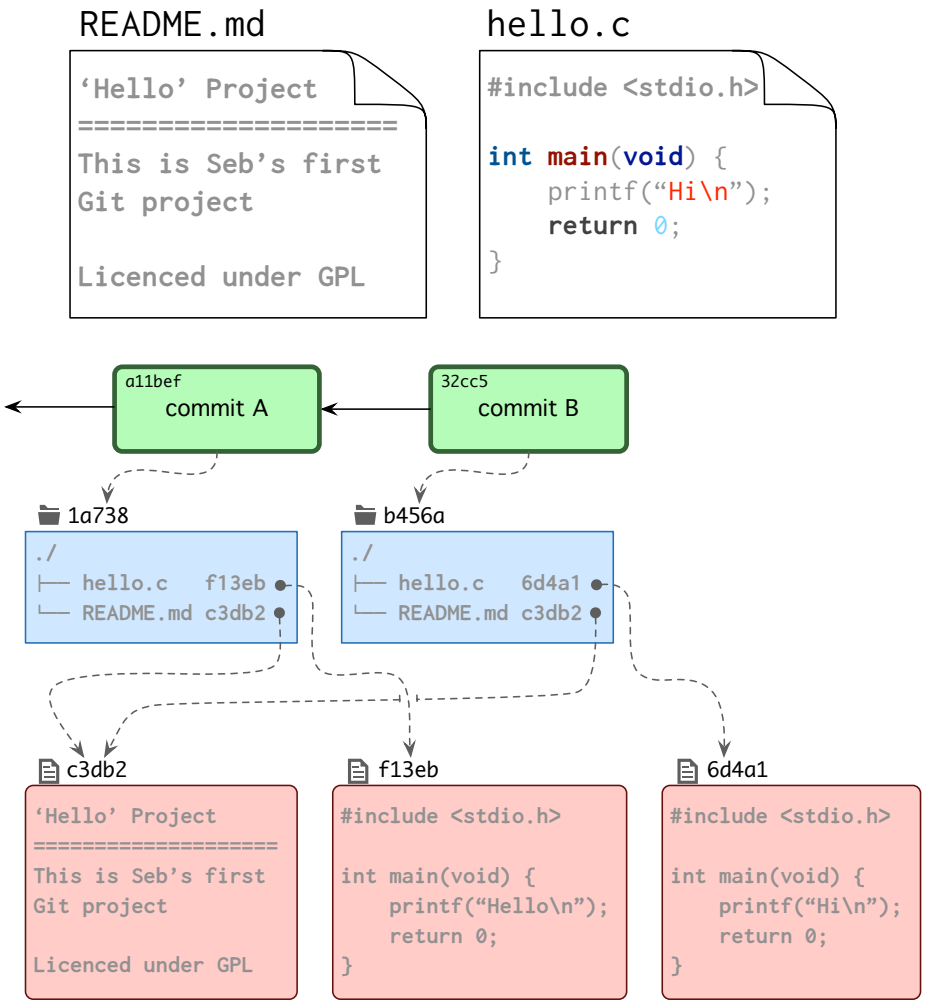
```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



Data Model Example



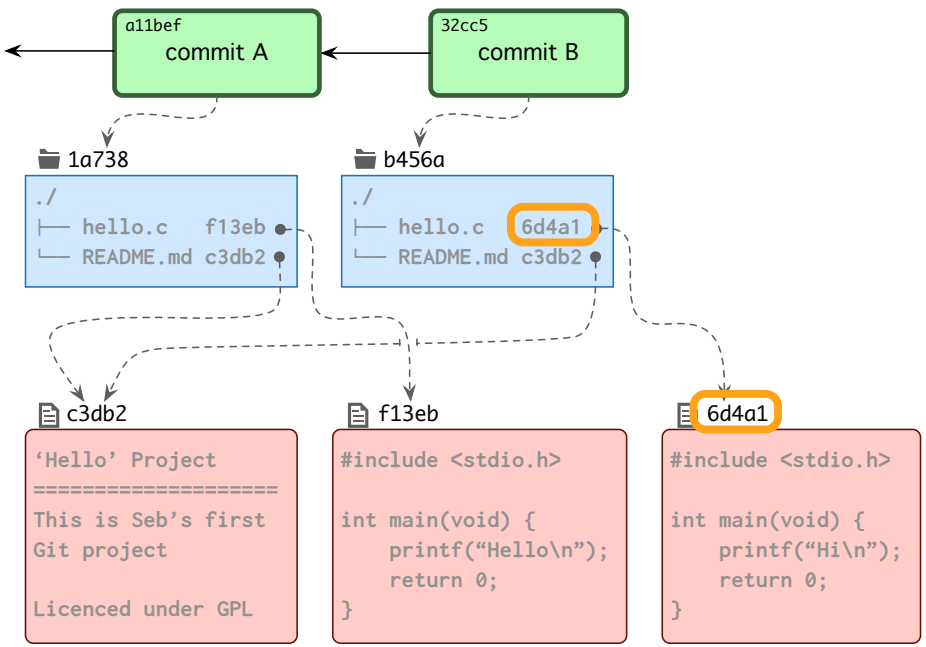
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



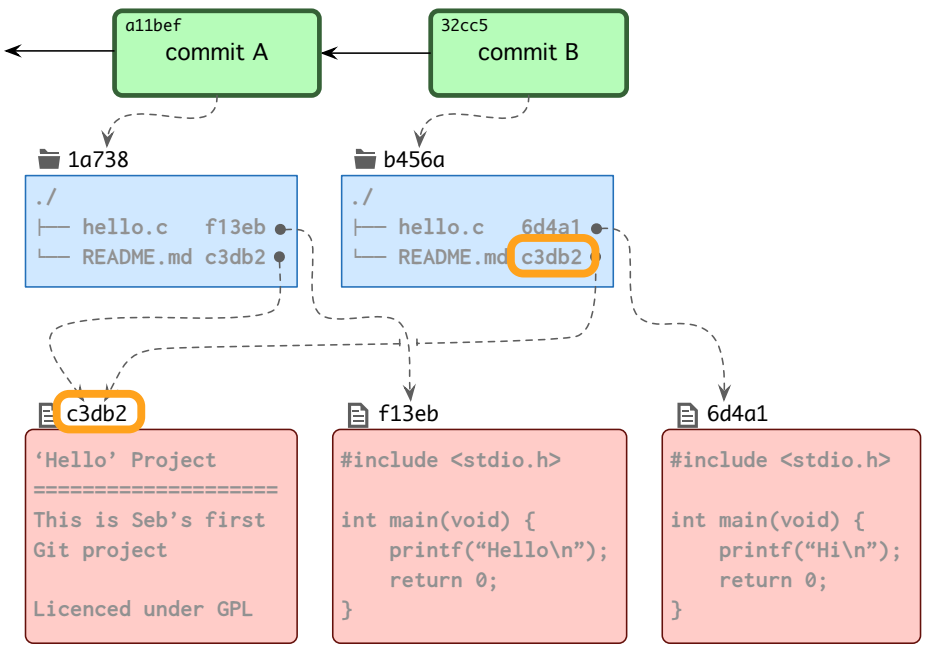
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



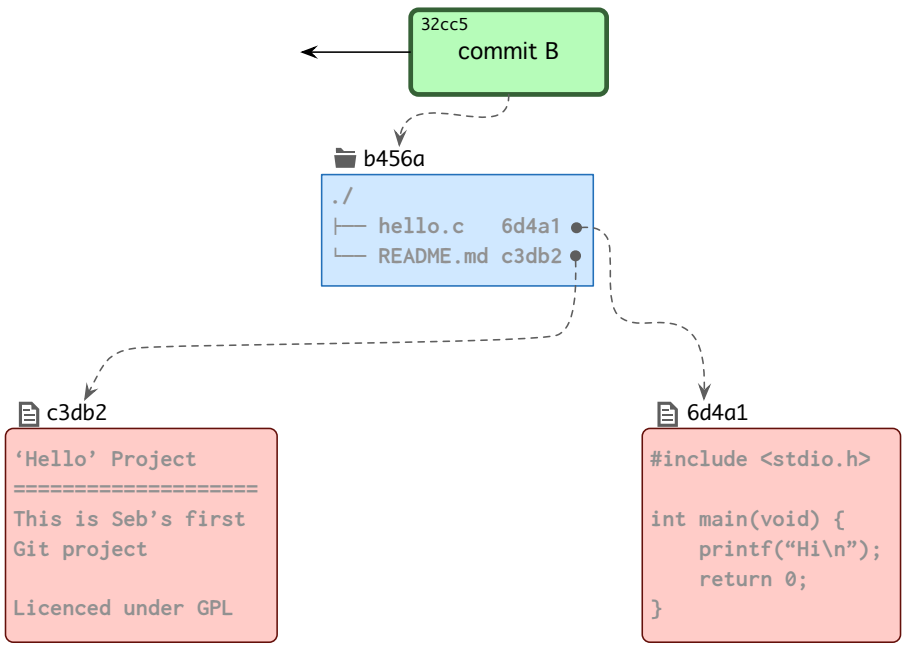
Data Model Example

README.md

```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



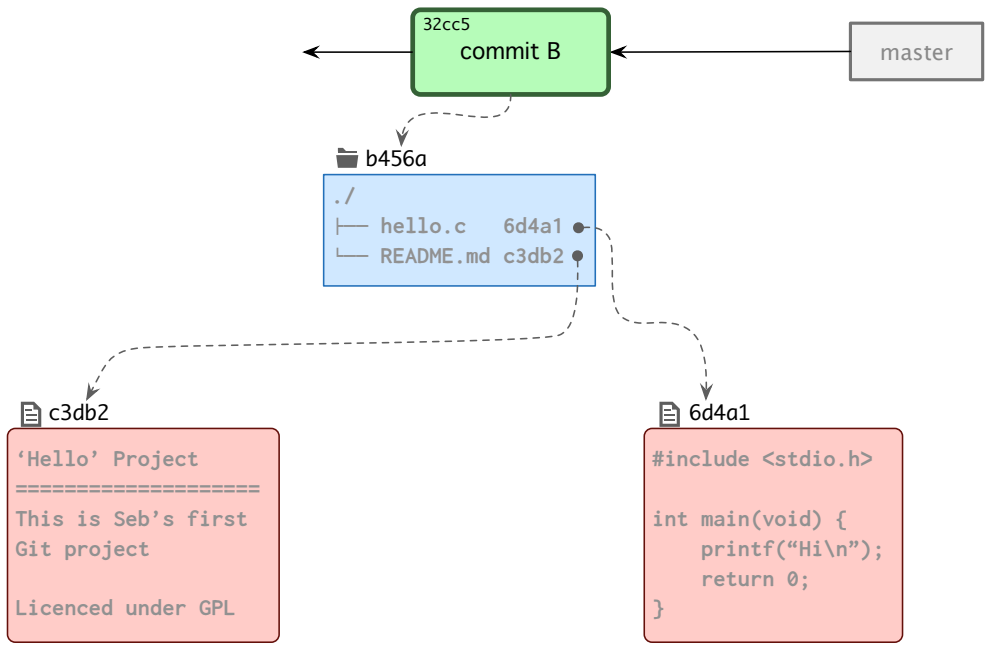
Data Model Example

README.md

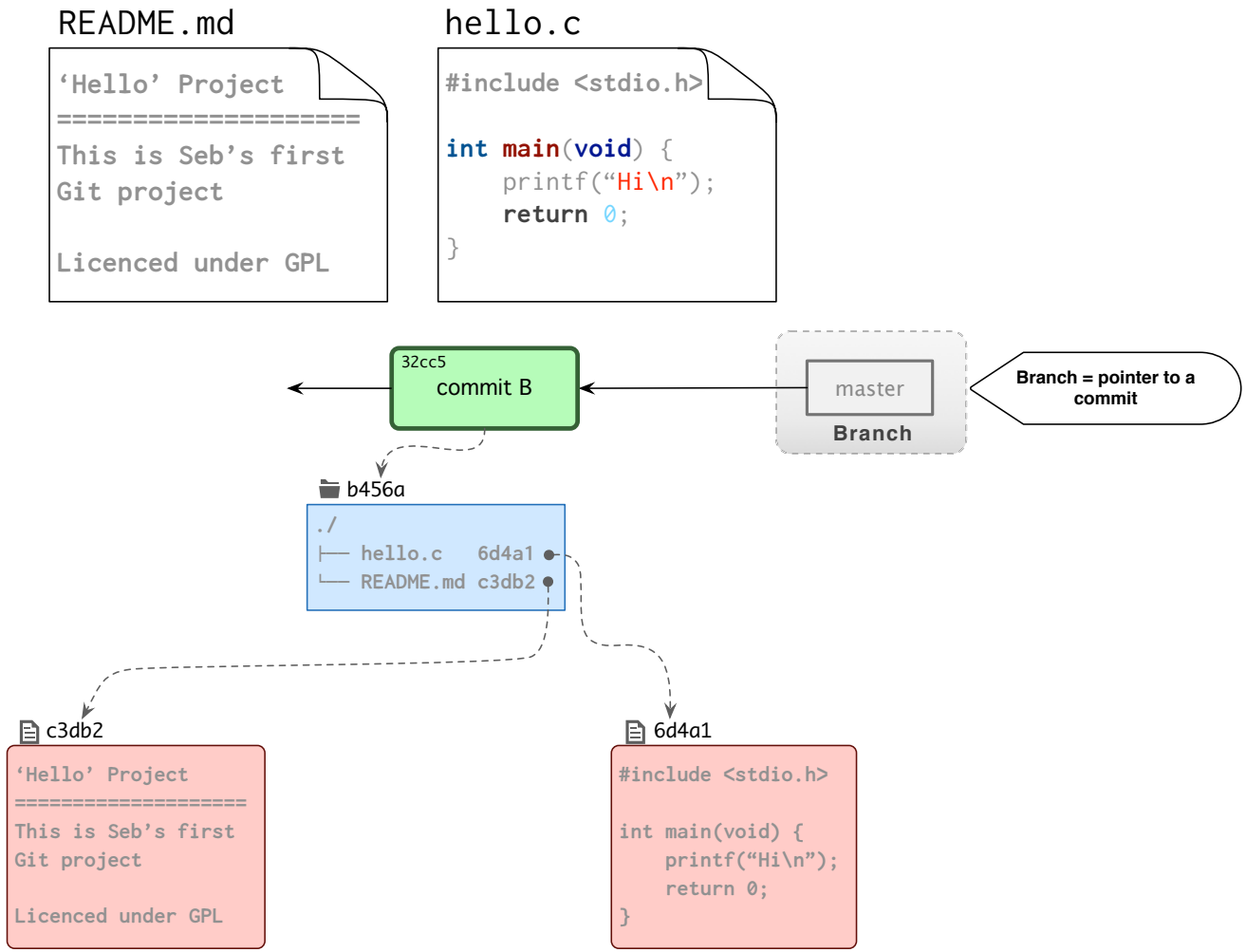
```
'Hello' Project
===============
This is Seb's first
Git project

Licenced under GPL
```

hello.c

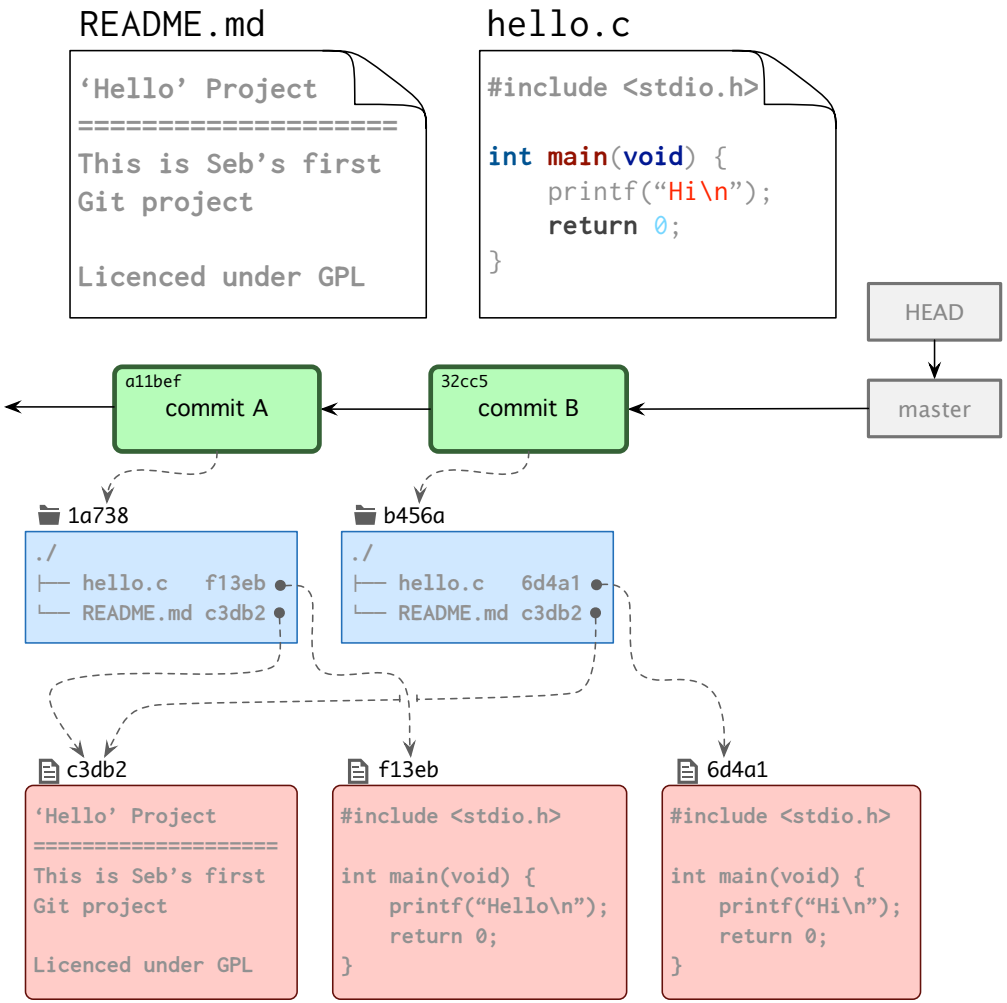
```
#include <stdio.h>

int main(void) {
    printf("Hi\n");
    return 0;
}
```



Data Model Example



Data Model Example



Branching

```
$> git branch <name>                    # create a branch <name>
```

Branching

```
$> git branch <name>                    # create a branch <name>
```

```
$> git branch -d <name>                 # delete the branch <name>
```

- use `-D` instead of `-d` to force deletion

Branching

```
$> git branch <name>                    # create a branch <name>
```

```
$> git branch -d <name>                 # delete the branch <name>
```

```
$> git branch [-a]                      # List [all] the branches
```

Branching

```
$> git branch <name>                    # create a branch <name>
```

```
$> git branch -d <name>                 # delete the branch <name>
```

```
$> git branch [-a]                      # List [all] the branches
```

```
$> git checkout [-b] <name>             # swicth to the branch <name>
```

- -b: create the branch before switching
- changes commited through `git commit` are committed to `HEAD`

Branching

```
$> git branch <name>                    # create a branch <name>
```

```
$> git branch -d <name>                 # delete the branch <name>
```

```
$> git branch [-a]                      # List [all] the branches
```

```
$> git checkout [-b] <name>             # swicth to the branch <name>
```

- **Switching** branches **changes** the files in your Working directory
  - ↪ since you change the `HEAD` snapshot...

Tags

```
$> git tag [-s] <name> [-m 'msg']
```

- `-s`: GPG-signed tag, assuming you have configured your signing key

  ```
  $> git config --global user.signingkey 0xDD01D5C1
  ```

Tags

```
$> git tag [-s] <name> [-m 'msg']
```

- `-s`: GPG-signed tag, assuming you have configured your signing key

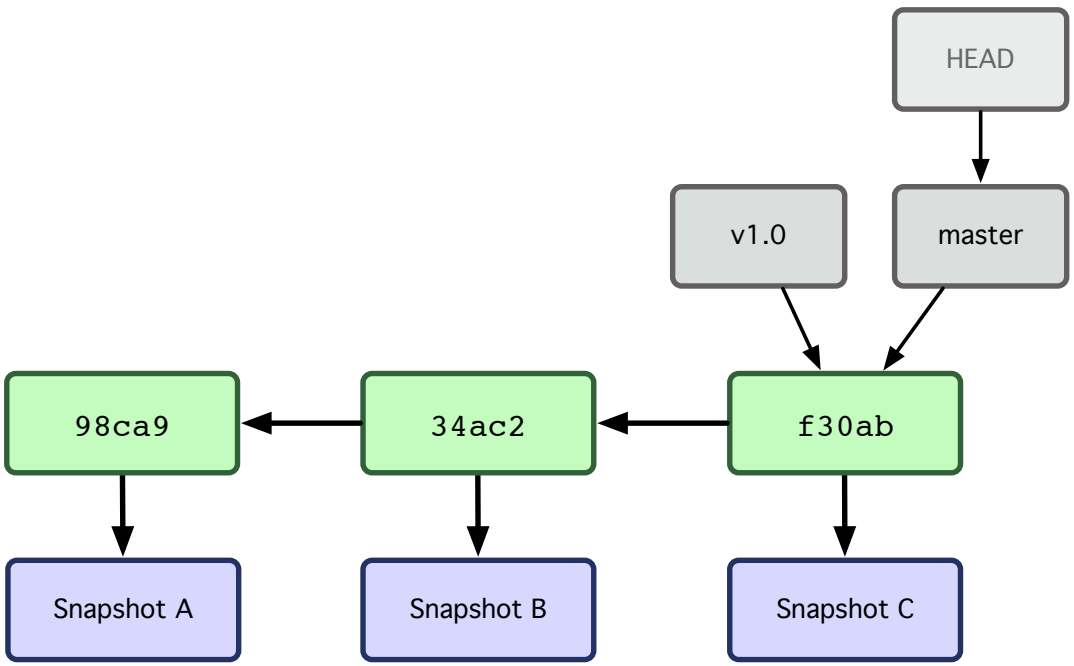
  ```
  $> git config --global user.signingkey 0xDD01D5C1
  ```

Your Turn!

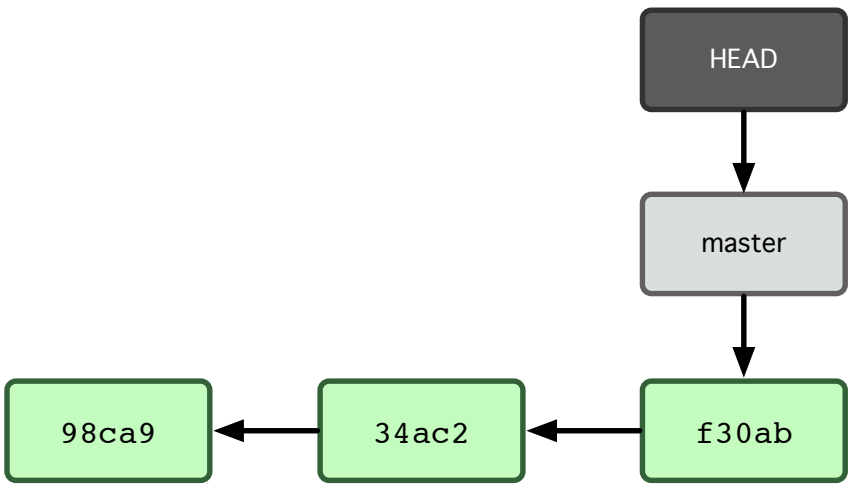
Branch and Tags Hands-on

```
$> cd /tmp/firstproject
$> git branch
* master
$> git tag v1.0 -m 'first tag'
$> git gr
* f31c173 (HEAD -> master, tag: v1.0) a first move with amend
* ee60f53 add README
$> git branch testing
$> git checkout testing   # Move to the 'testing' branch
$> echo 'testing' >> README.md && git commit -a -m "testing 1"
[testing 7afa96d] testing 1
 1 file changed, 1 insertion(+)
$> git checkout master   # return to 'master'
$> echo 'master' >> README.md && git commit -a -m "master"
[master 72d4d5f] master
 1 file changed, 1 insertion(+)
$> git gr
$> gitx   # or gitk
```

Daily Branching Example

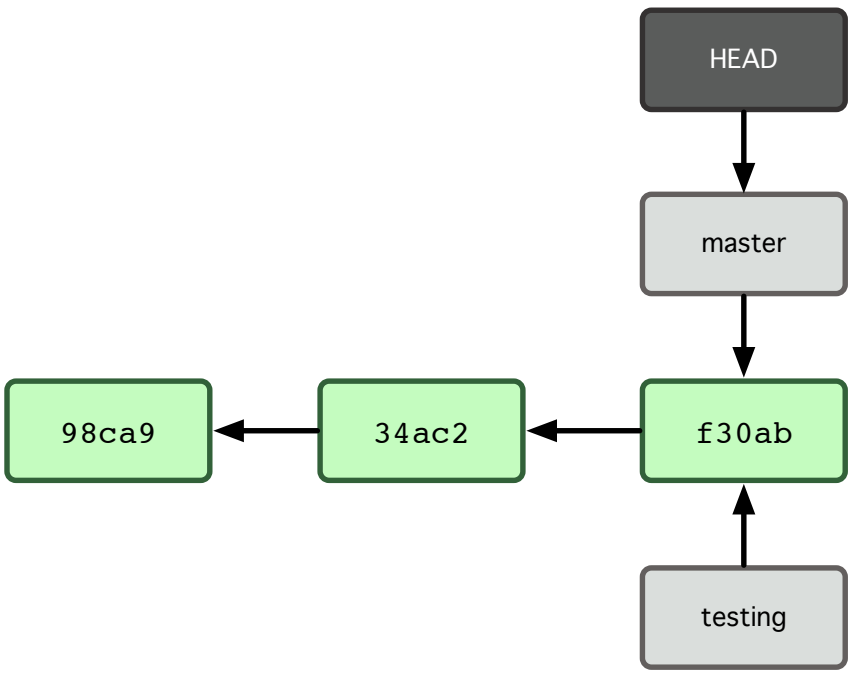


Daily Branching Example



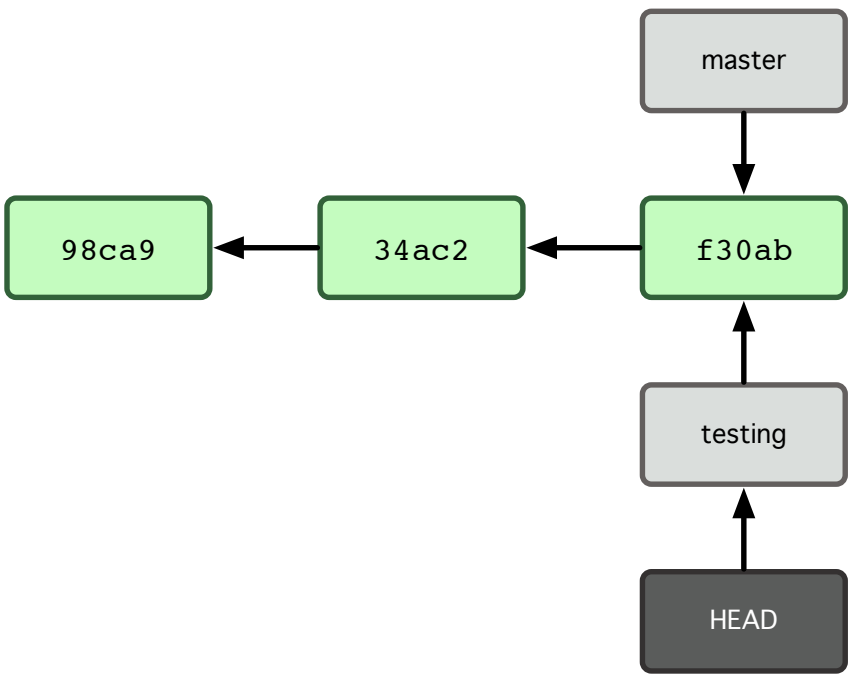
Daily Branching Example

```
(master)$> git branch testing          # create a branch named 'testing'
```



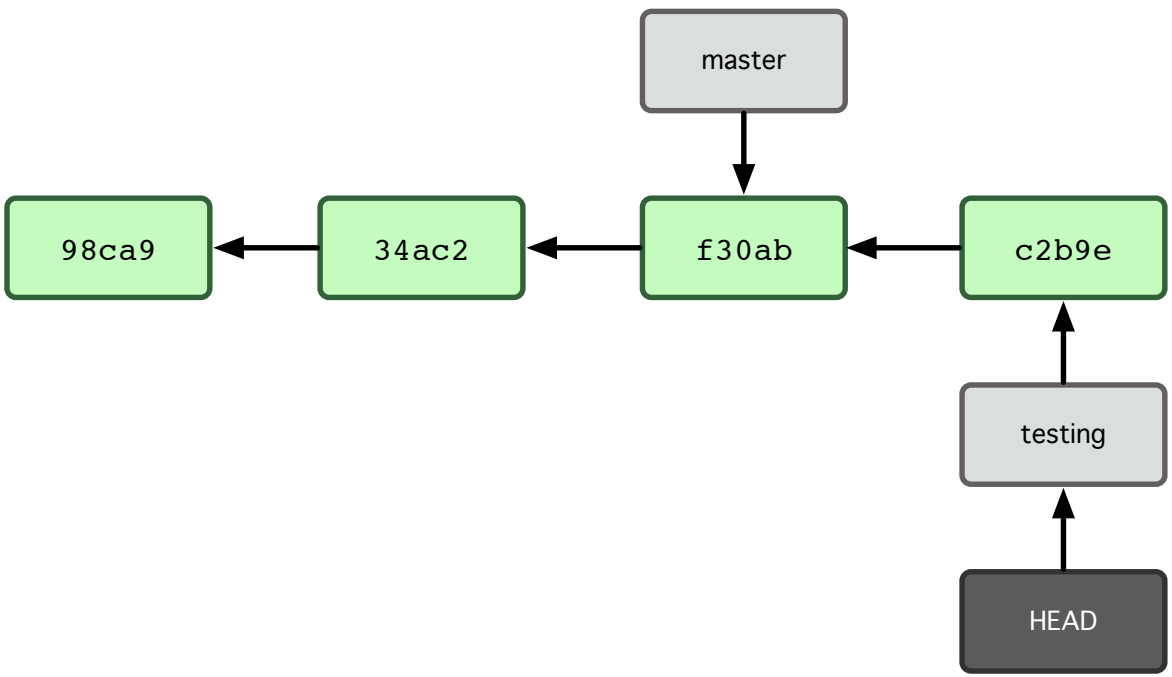
Daily Branching Example

```
(master)$> git checkout testing          # switch to the 'testing' branch
```



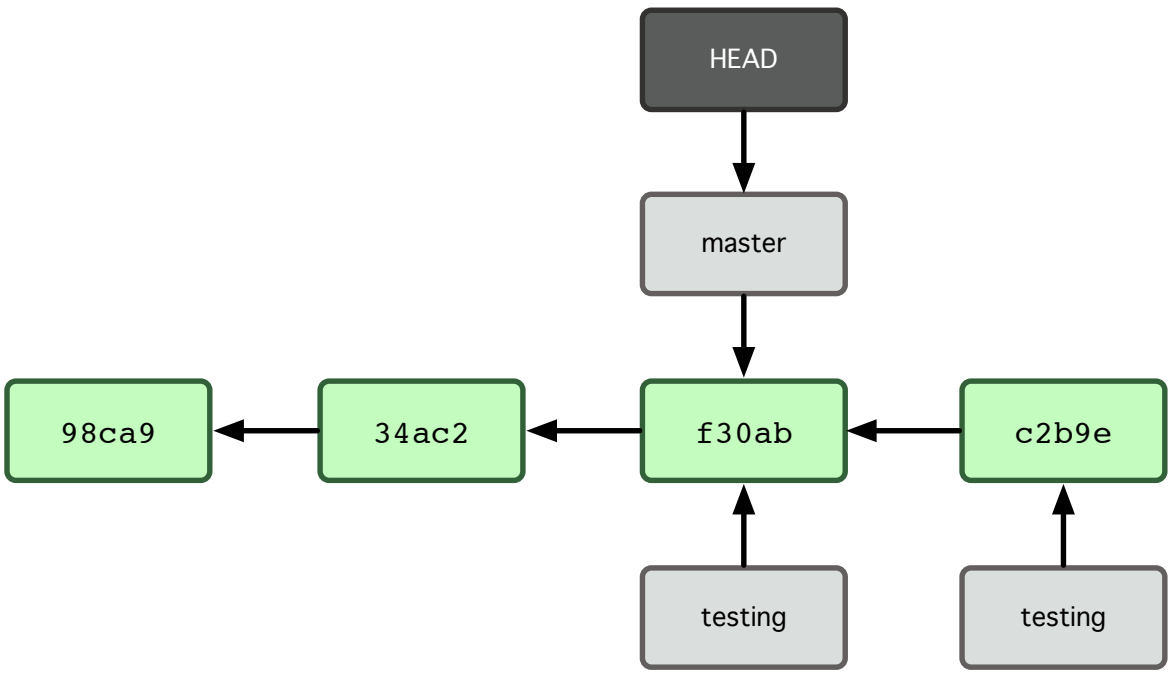
Daily Branching Example

```
(testing)$> vim README.md                          # make some edits...
(testing)$> git commit -a -m "made a change"       # and commit them
```



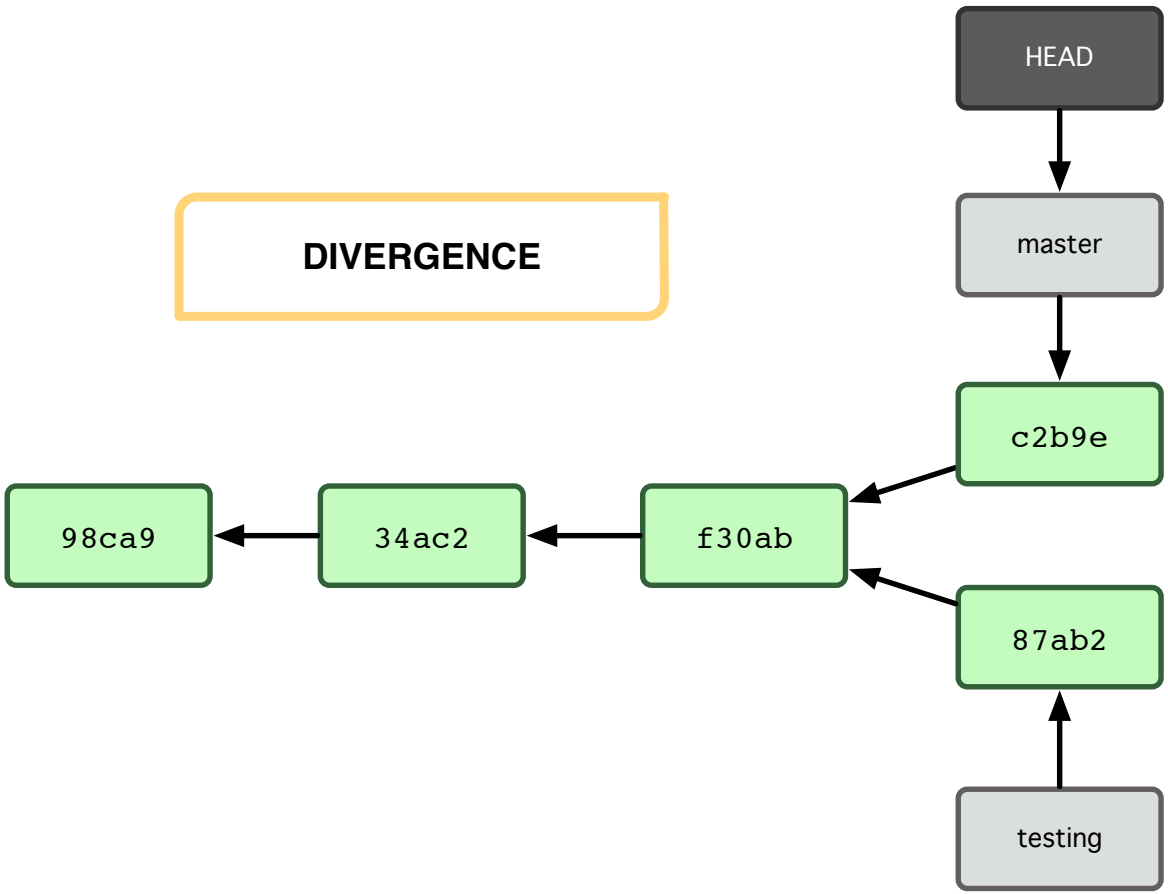
Daily Branching Example

```
(testing)$> git checkout master          # switch back to 'master' branch
```



Daily Branching Example

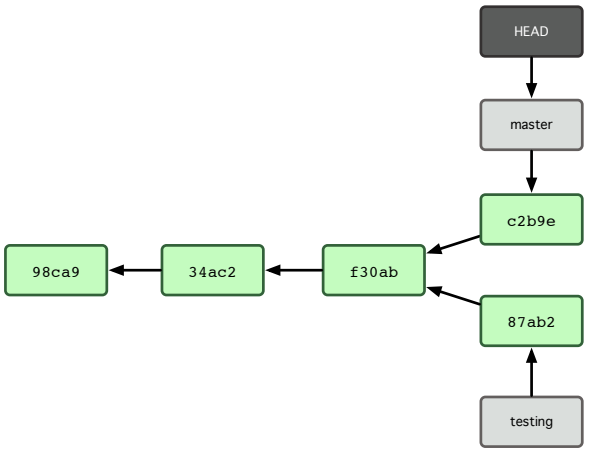
```
(testing)$> vim README.md                          # make some edits
(testing)$> git commit -a -m "intro"               # introduce divergence!
```



Diverging / Converging (Fork-Join)

Diverging

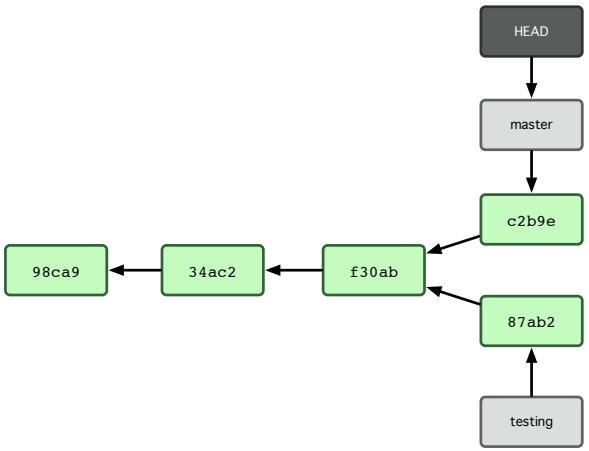
- Changes are committed into two branches independently
  - ↪ Then the branches **diverge**



Diverging / Converging (Fork-Join)

Diverging

- Changes are committed into two branches independently
  - ↪ Then the branches **diverge**



Converging to join branches

1. `merge` (if possible in **fast-forward** mode)
2. `rebase`

Merging

```
$> git merge [--no-ff] <branch>
```

- Different auto-merge strategies
  - ↪ **fast-forward** (if possible)
  - ↪ **3-ways** (regular)
- Usually painless ;)

Merging

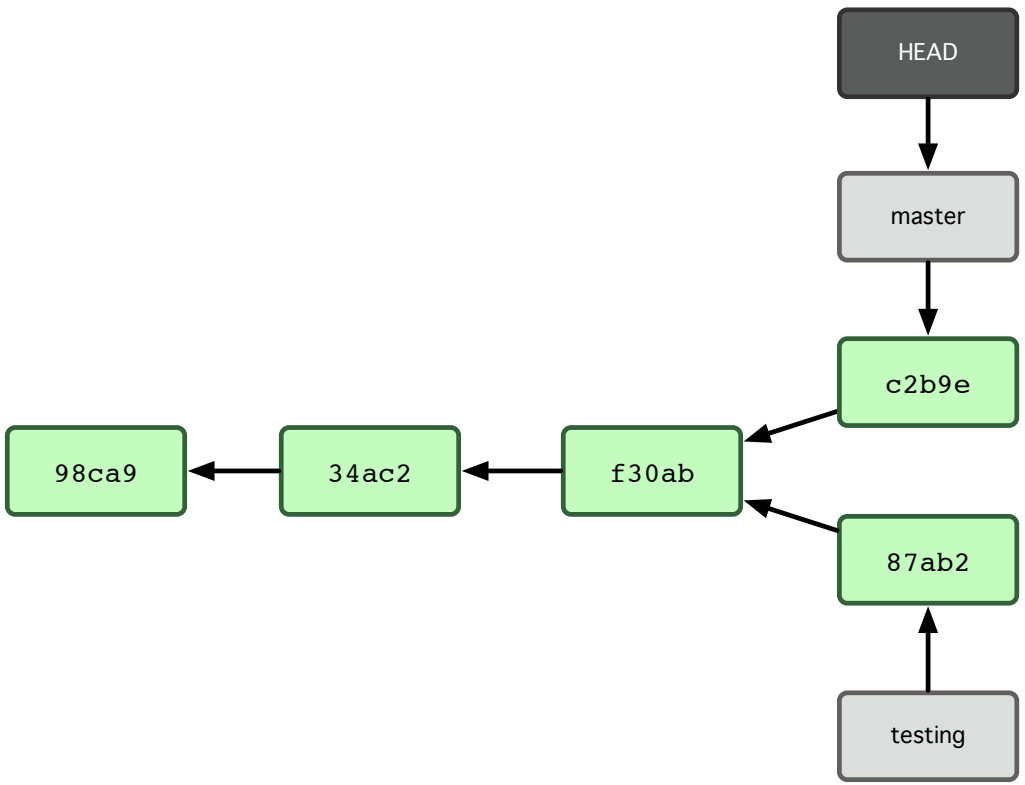
```
$> git merge [--no-ff] <branch>
```

- Different auto-merge strategies
  - ↪ **fast-forward** (if possible)
  - ↪ **3-ways** (regular)
- Usually painless ;)

```
<<<<<<< HEAD
master
=======
testing
>>>>>>> testing
```

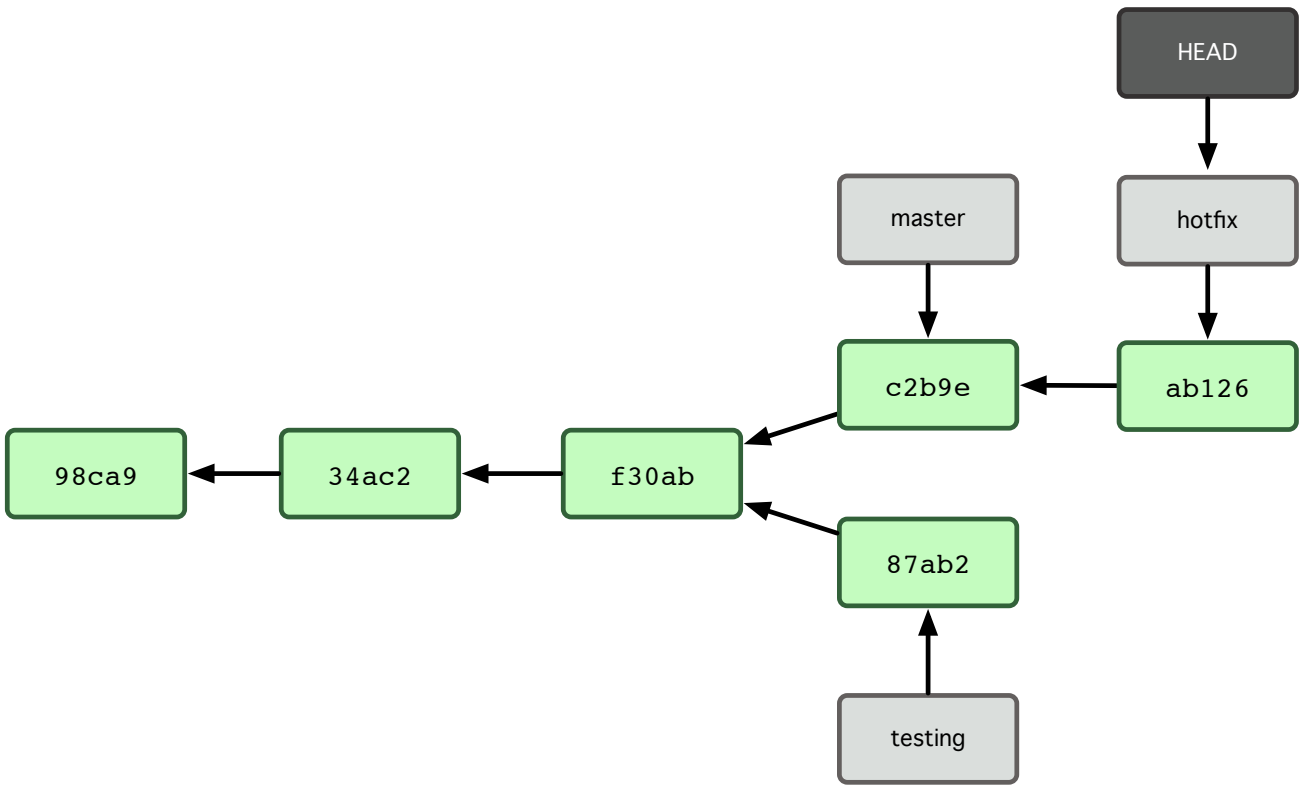
- **In case of conflicts**:
  - ↪ Resolve the conflicts manually — `vim` / `emacs` / `subl` ...
    - ✓ check for the sequence <<< in the text
  - ↪ then mark as resolved — `git add <file>`
  - ↪ and trigger the merge commit — `git commit`

Daily Branching Example



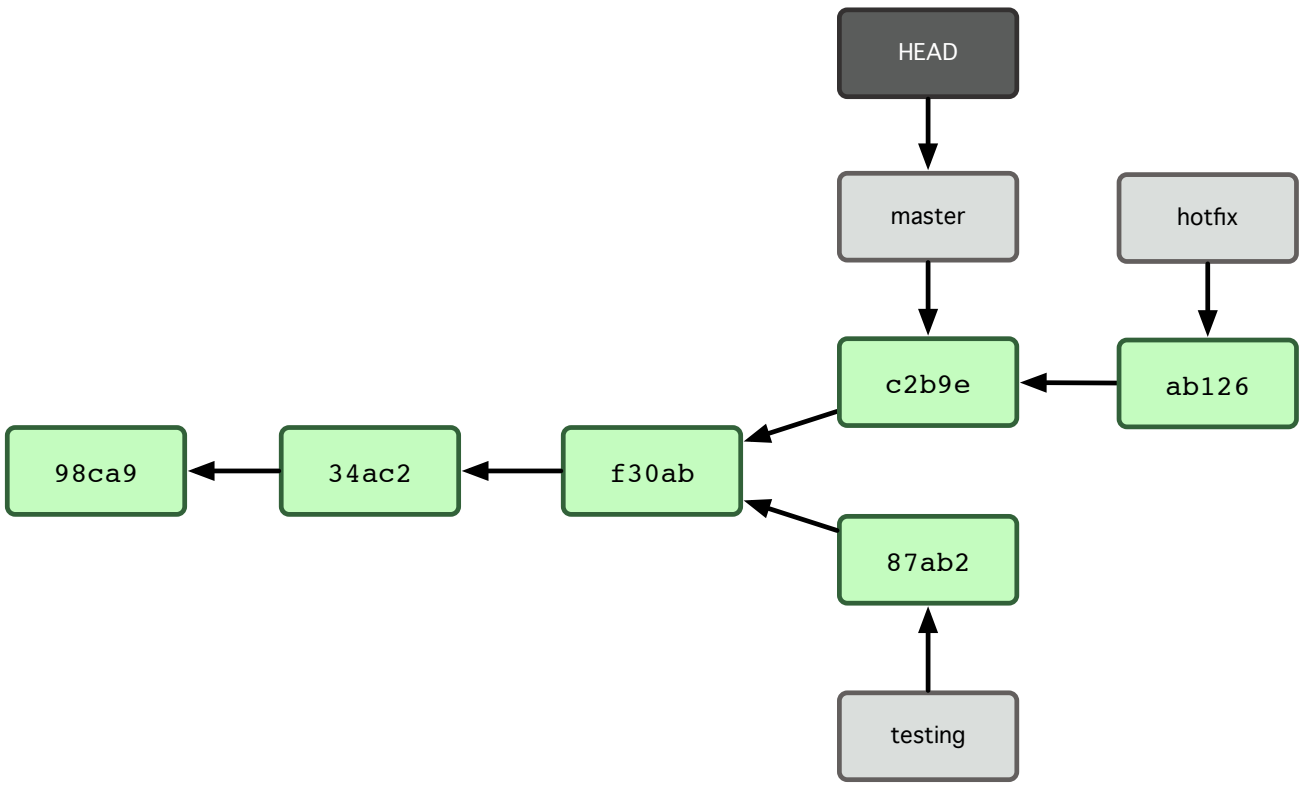
Daily Branching Example

```
(master)$> git checkout -b hotfix        # create and switch to 'hotfix'
(hotfix)$> vim test.rb                      # make some edits...
(hotfix)$> git commit -a -m "hotfix"           # and commit them
```



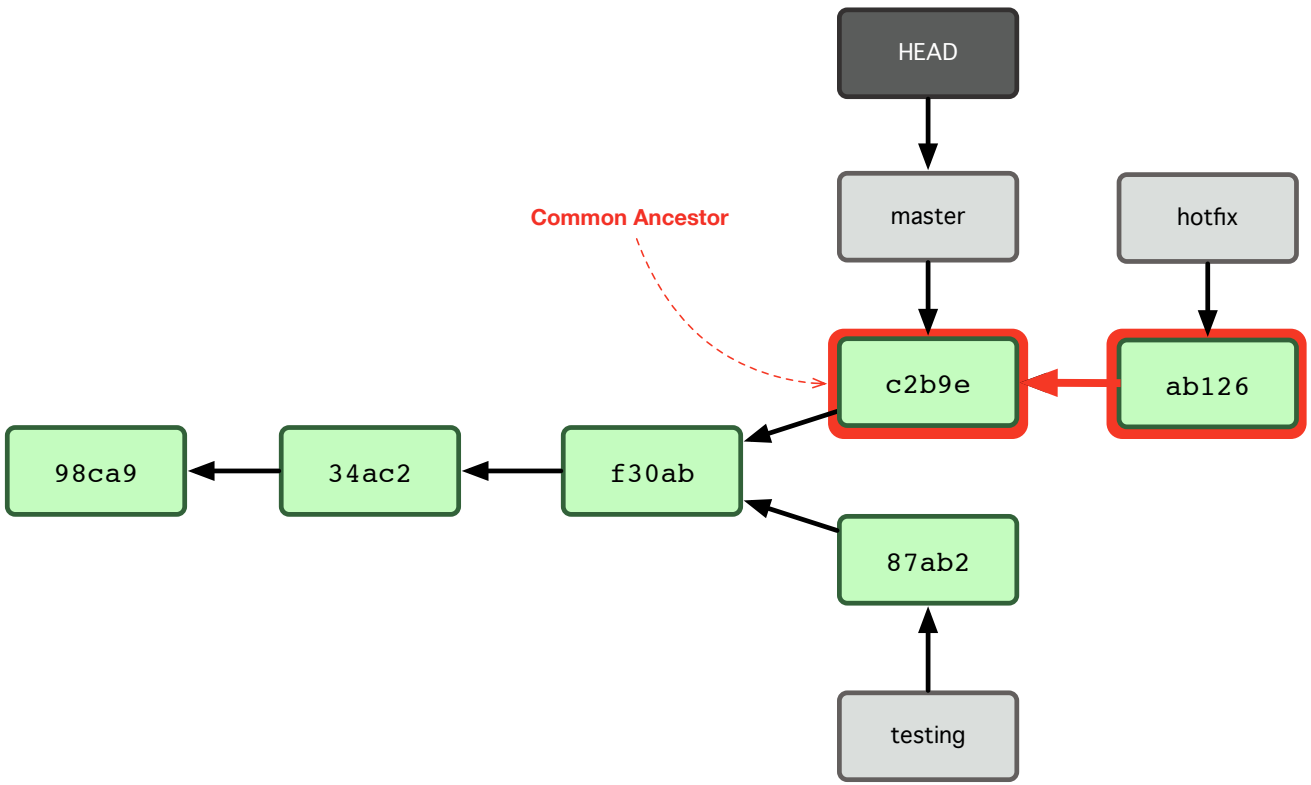
Daily Branching Example

```
(hotfix)$> git checkout master          # swicth back to 'master' branch
```



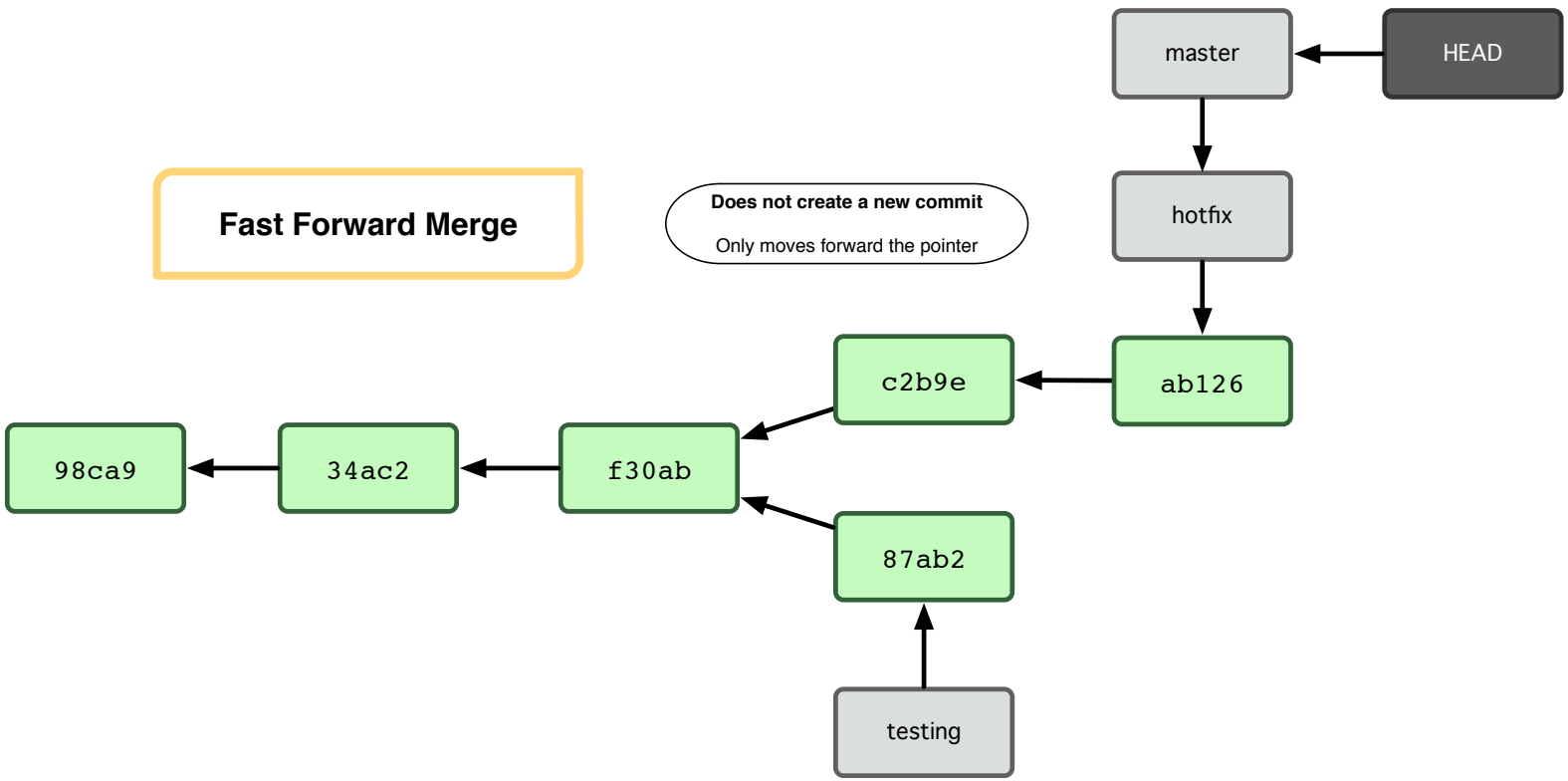
Daily Branching Example

```
(master)$> git merge hotfix     # merge the 'hotfix' branch (fast-forward)
```

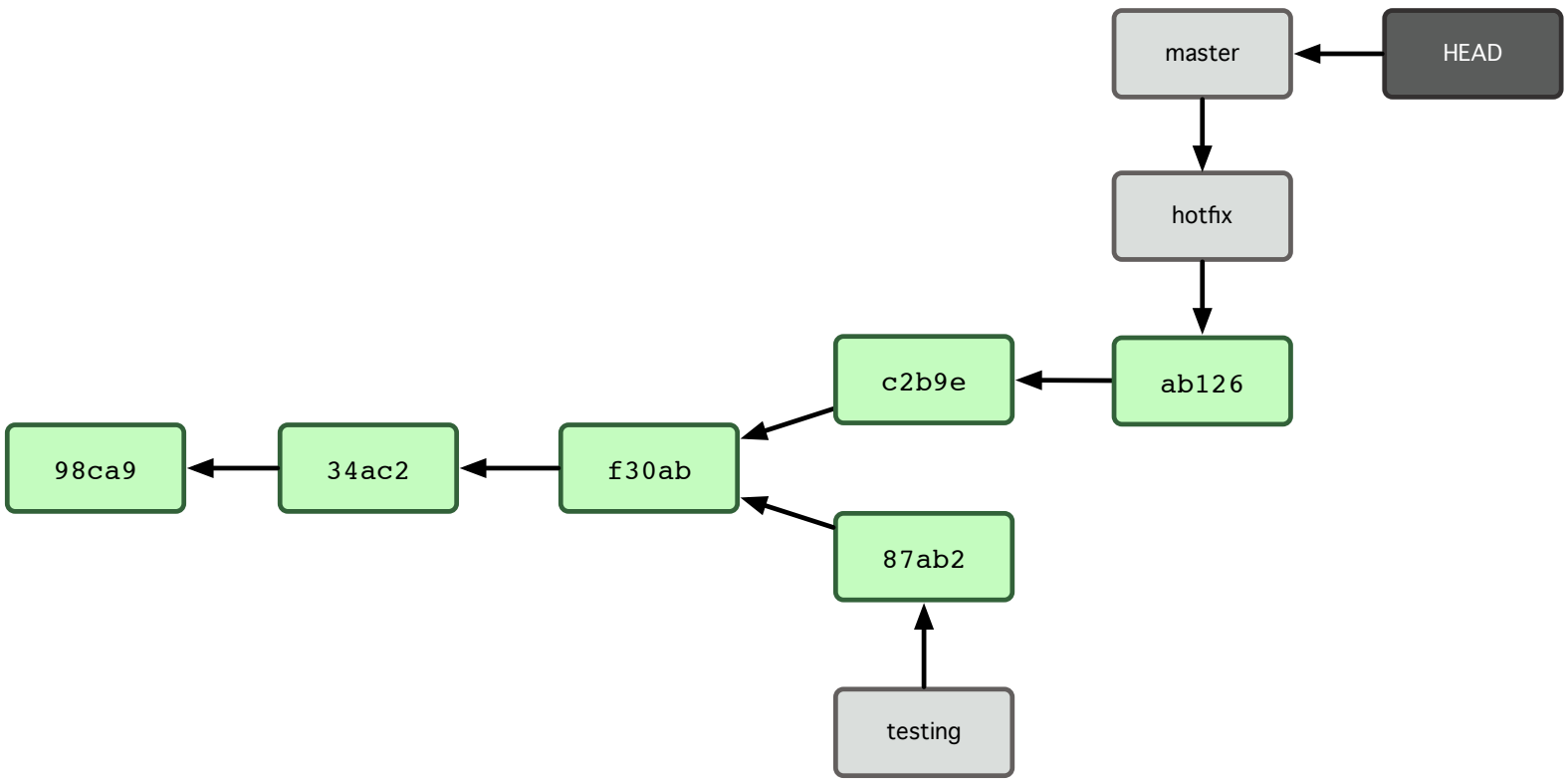


Daily Branching Example

```
(master)$> git merge hotfix     # merge the 'hotfix' branch (fast-forward)
```

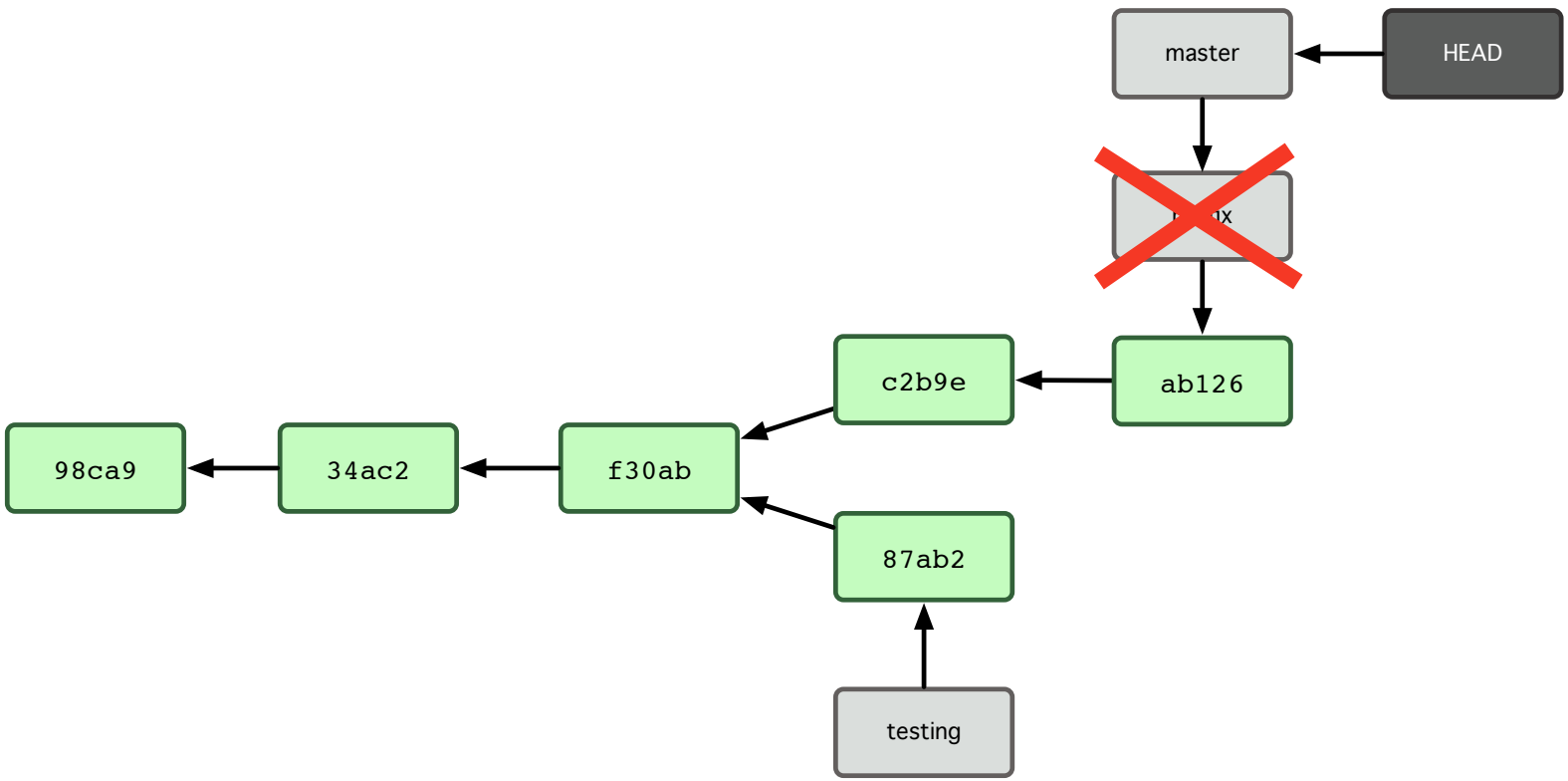


Daily Branching Example



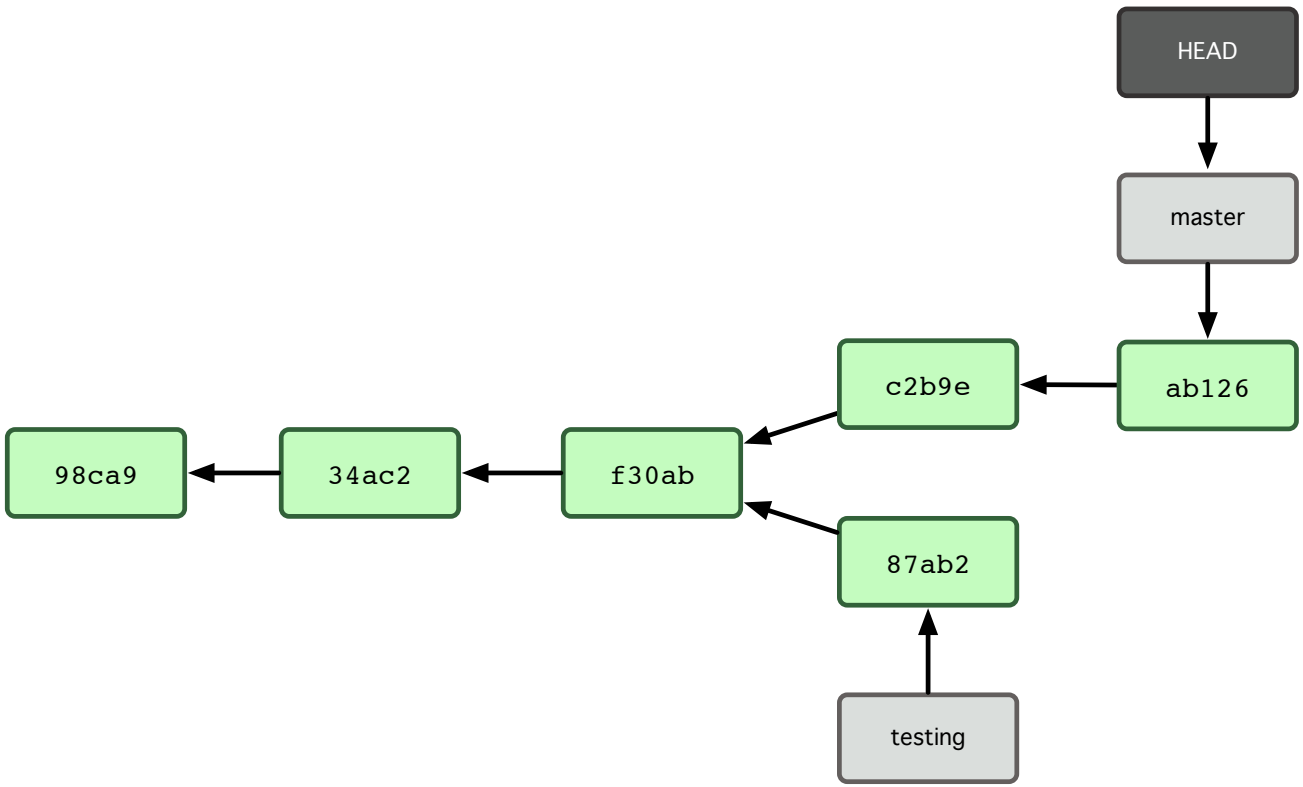
Daily Branching Example

```
(master)$> git branch -d hotfix     # delete the (useless) 'hotfix' branch
```

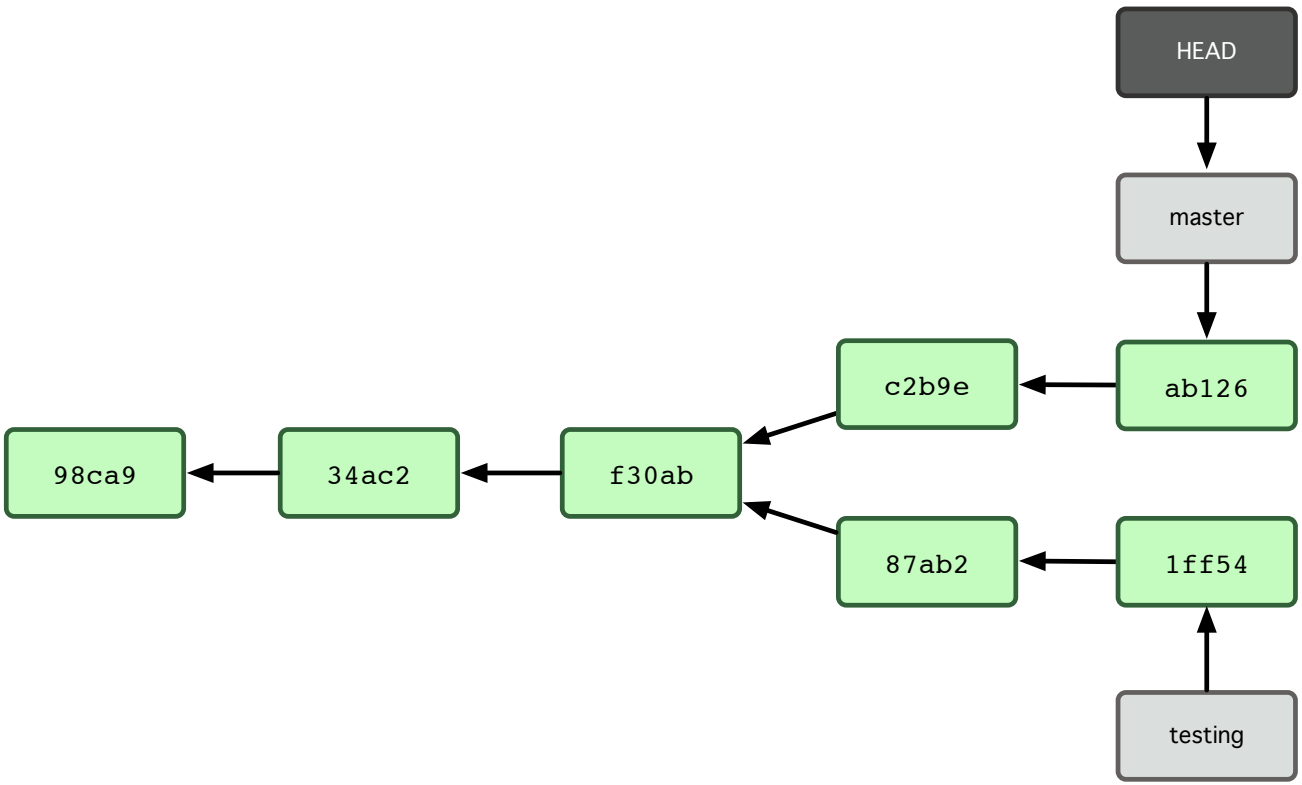


Daily Branching Example

```
(master)$> git branch -d hotfix     # delete the (useless) 'hotfix' branch
```

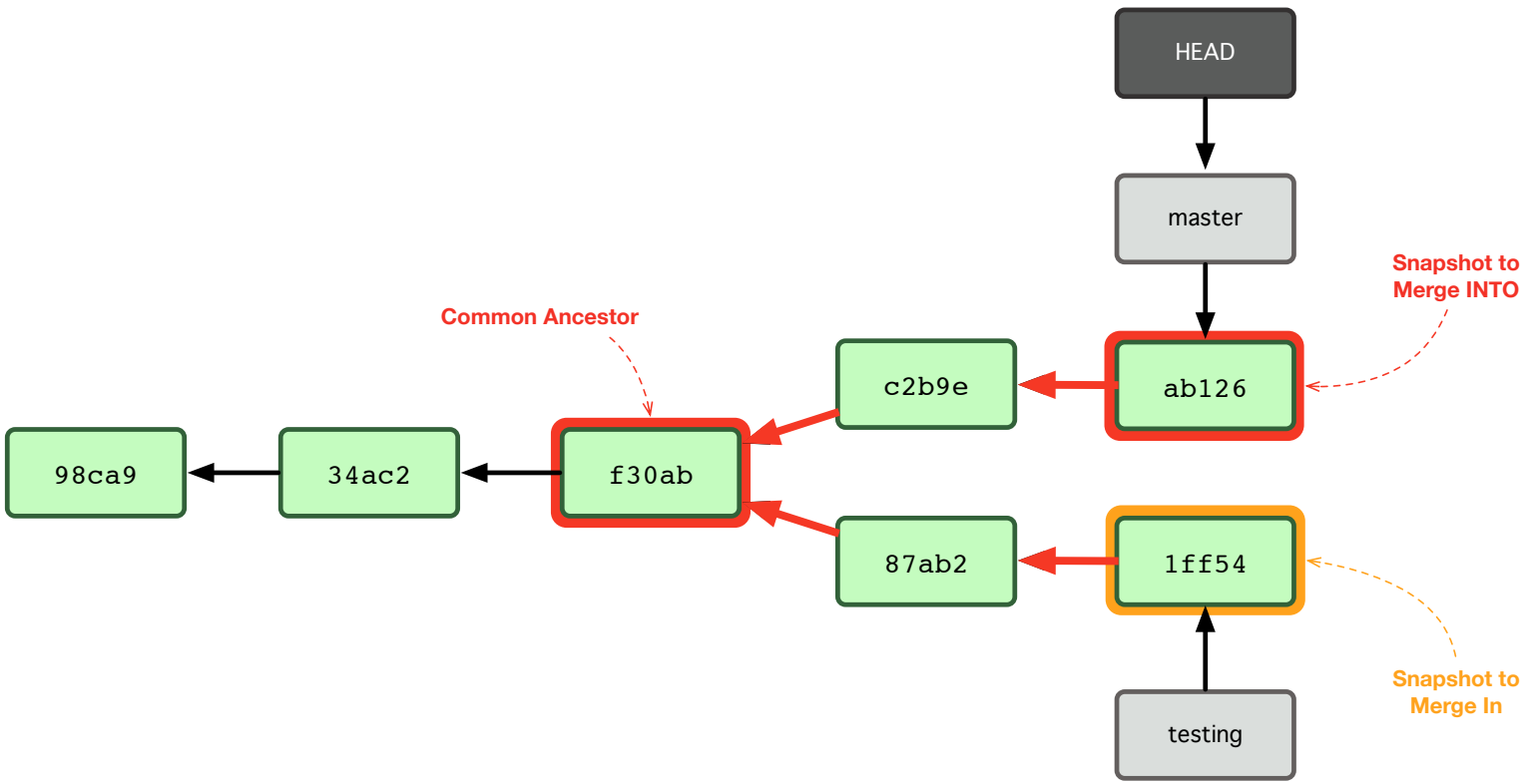


Daily Branching Example



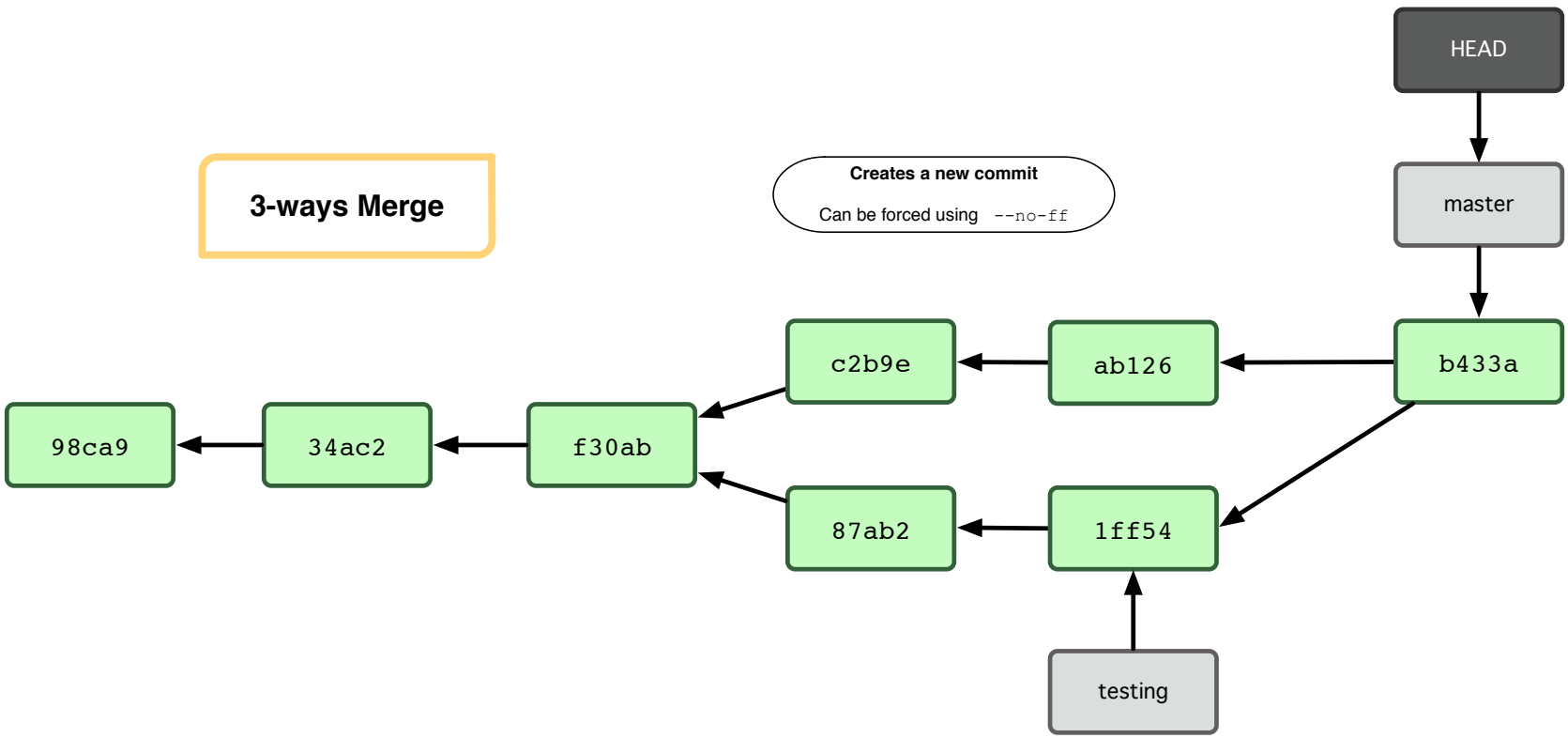
Daily Branching Example

```
(master)$> git merge testing          # merge the 'testing' branch (3-ways)
```



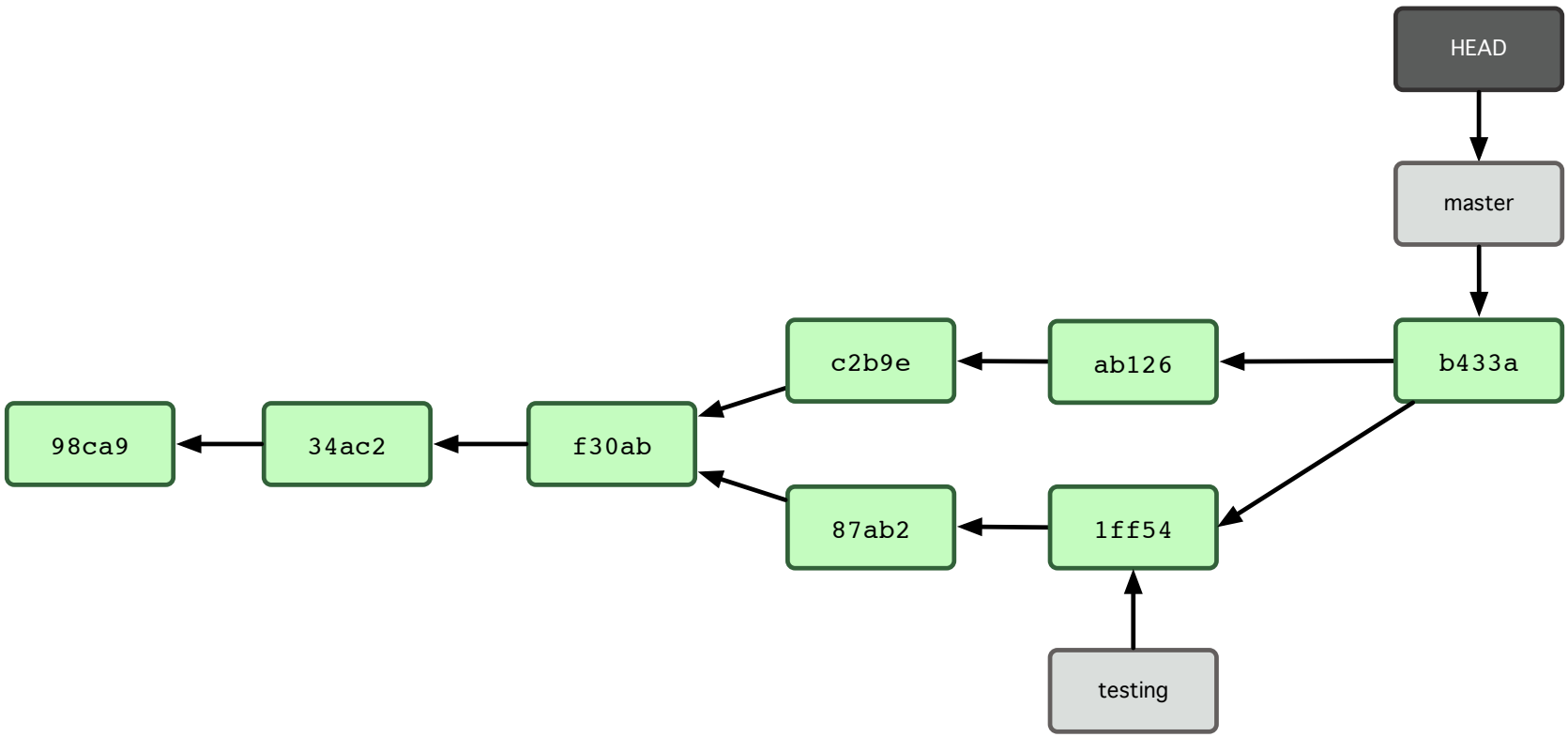
Daily Branching Example

```
(master)$> git merge testing          # merge the 'testing' branch (3-ways)
```



Daily Branching Example

```
(master)$> git merge testing          # merge the 'testing' branch (3-ways)
```



Merging and solving conflicts Hands-on

Your Turn!

Merging and solving conflicts Hands-on

```
$> cd /tmp/firstproject
$> git checkout master
Switched to branch 'master'
$> git checkout -b hotfix
Switched to a new branch 'hotfix'
$> touch test.rb && git add test.rb && git commit -m "hotfix"
[hotfix ac188bd] hotfix
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.rb
$> git checkout master
Switched to branch 'master'
$> git gr
* 72d4d5f (HEAD -> master) master
* f31c173 (tag: v1.0) a first move with amend
* ee60f53 add README
```

Merging and solving conflicts Hands-on

- Fast-Forward Merge

```
$> git merge hotfix
Updating 72d4d5f..ac188bd
Fast-forward
 test.rb | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.rb
$> git gr
* ac188bd (HEAD -> master, hotfix) hotfix
* 72d4d5f master
* f31c173 (tag: v1.0) a first move with amend
* ee60f53 add README

$> git branch -d hotfix
Deleted branch hotfix (was ac188bd)
```

Merging and solving conflicts Hands-on

- Solving conflicts

```
$> git merge testing
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
$> cat README.md
<<<<<<< HEAD
master
=======
testing
>>>>>>> testing
$> vim README.md    # Edit to solve the conflicts
$> cat README.md
master corrected
```

Merging and solving conflicts Hands-on

- Solving conflicts

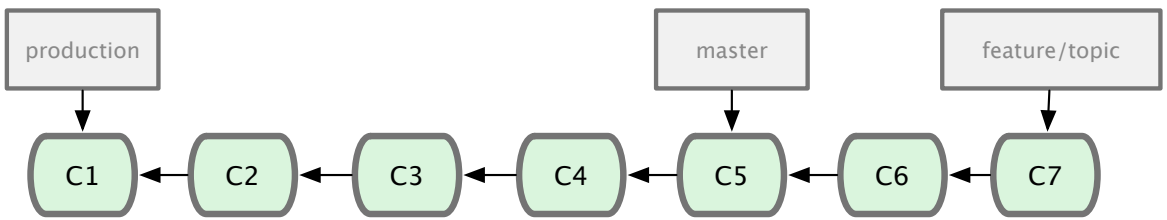
```
$> git status    # OR git st
On branch master
You have unmerged paths.

Unmerged paths:
    both modified:   README.md

no changes added to commit
```

Merging and solving conflicts Hands-on

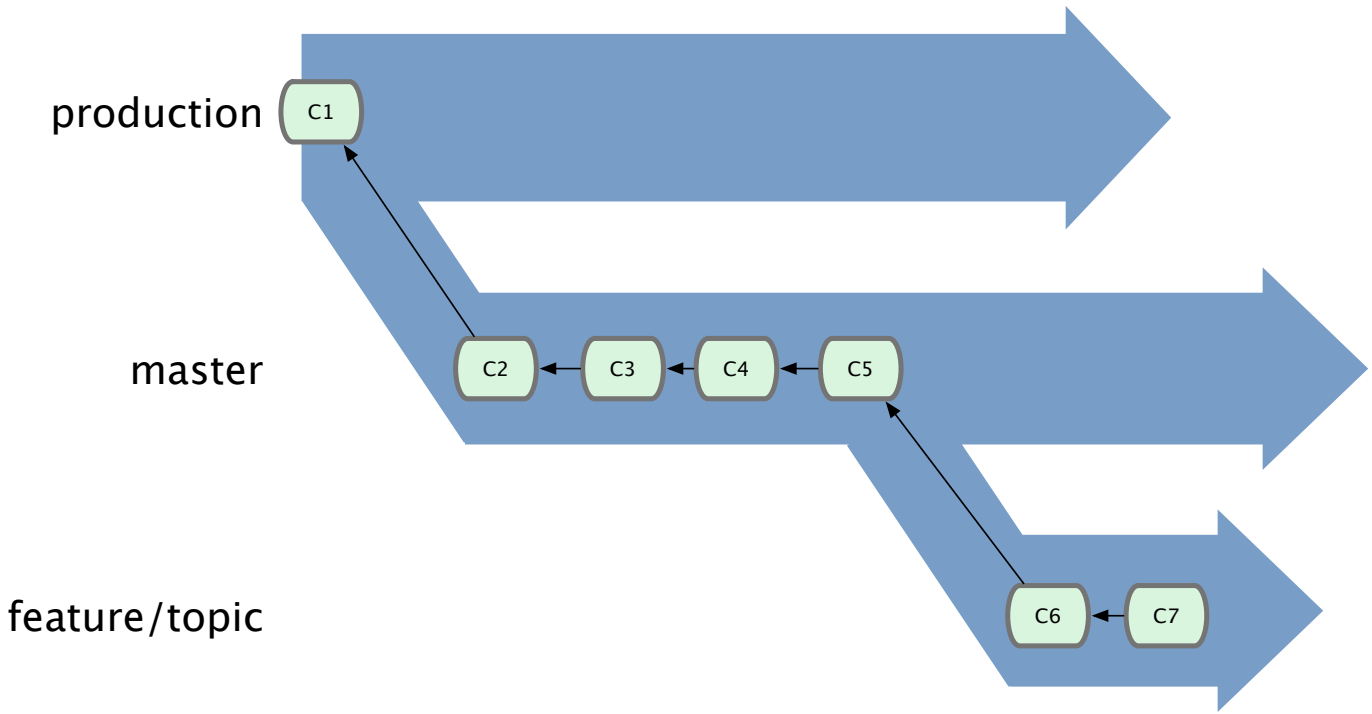
- Solving conflicts and 3-way merge

```
$> git add README.md     # Mark as corrected / conflict solved
$> git commit
Recorded resolution for 'README.md'.
[master ef299b7] Merge branch 'testing'
$> git gr
*   ef299b7 (HEAD -> master) Merge branch 'testing'
|\
| * 7afa96d (testing) testing 1
* | ac188bd hotfix
* | 72d4d5f master
|/
* f31c173 (tag: v1.0) a first move with amend
* ee60f53 add README
```

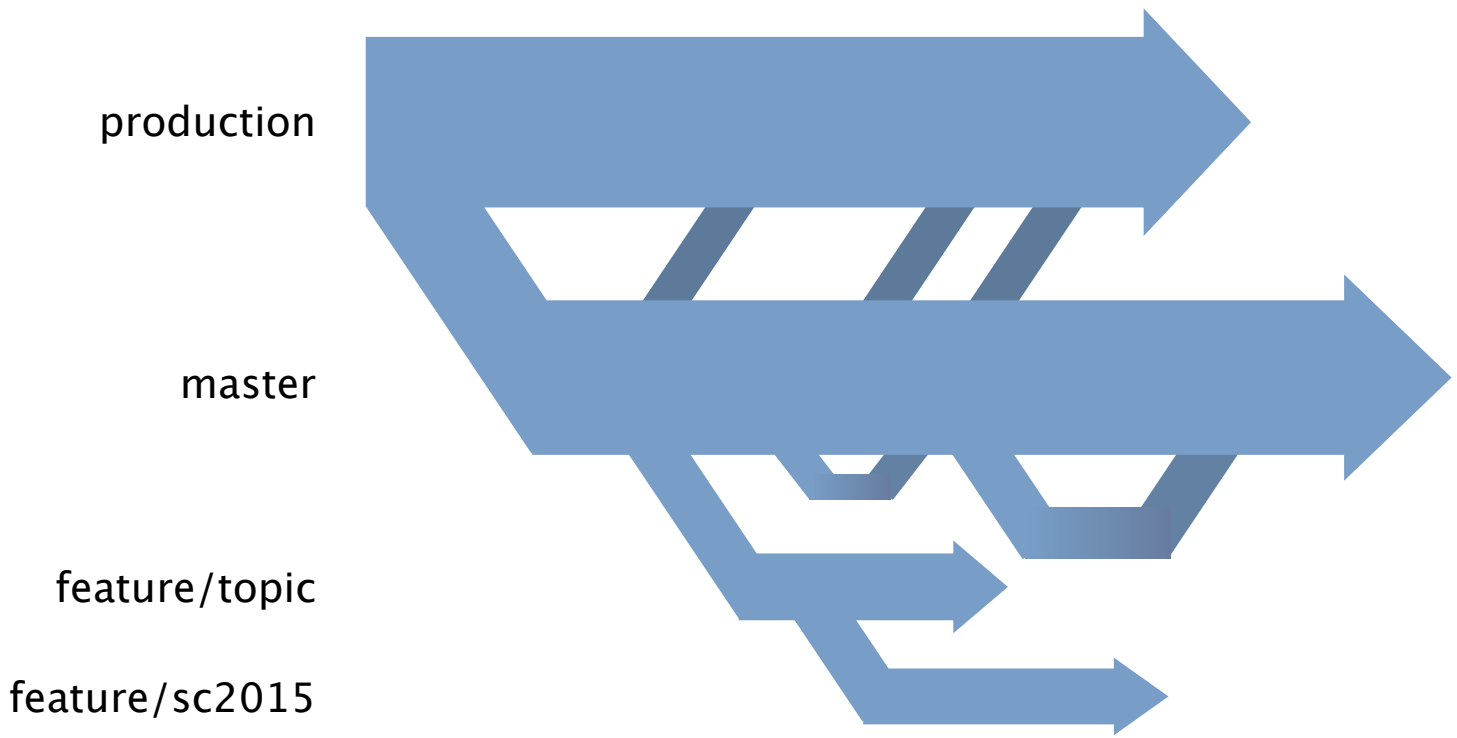
Branching Workflow



Branching Workflow



Branching Workflow



Git-flow to the rescue

http://nvie.com/posts/a-successful-git-branching-model/

```
$> git flow init
```

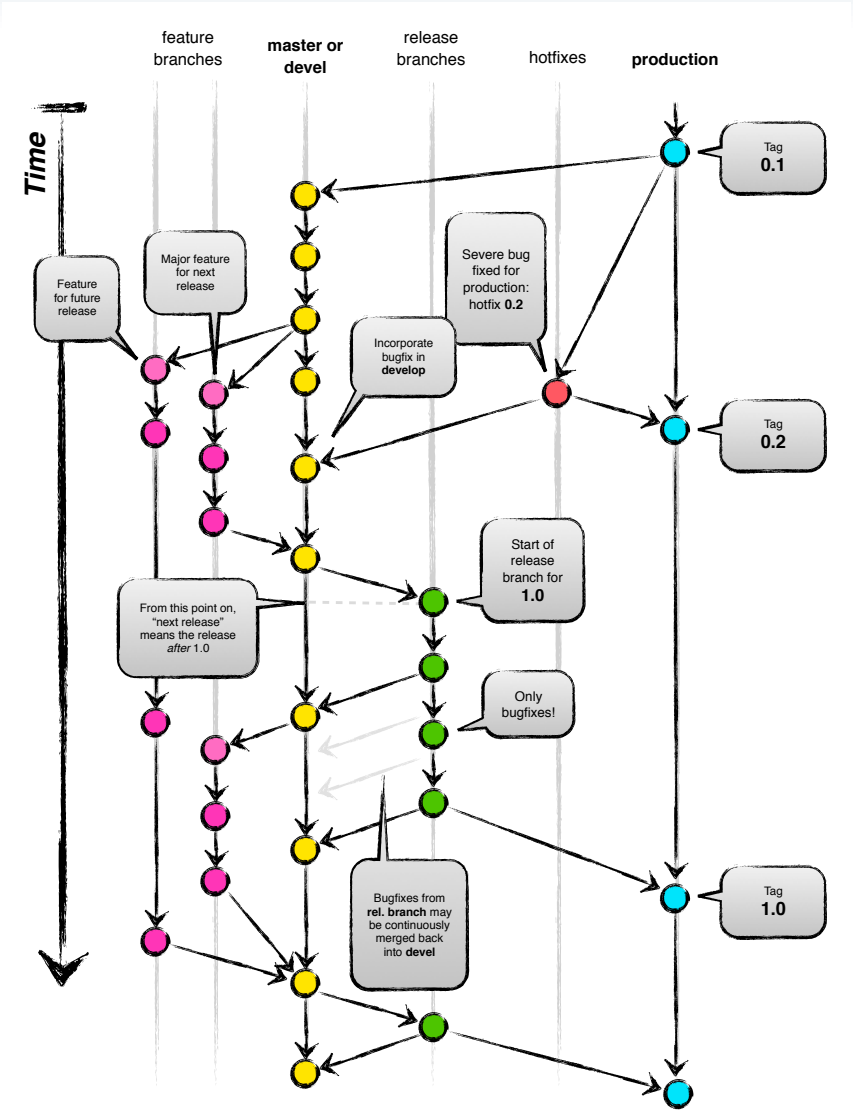
```
$> git flow feature { start, publish, finish } <name>
```

```
$> git flow release { start, publish, finish } <version>
```

- Ensure two long running branches
  - ↪ `production` : the stable branch
    - ✓ ideally holding **only** tags of the successive releases
  - ↪ `master` / `devel`: the **main** branch where the developments occurs
- On demand: make a new feature branch `feature/<name>`
- From time to time, release your code into `production` and tag

Git-flow Illustrated

[Source : Nvie]



Git-flow Setup using FalkorLib

- Initiate a Git-flow-ready repository using FalkorLib
  - ↪ [Personnal] Ruby Library offering the `falkor` binary

```
$> falkor new repo [--rake]      # setup the current directory
```

- The repository is fed with a root `Makefile` (or `Rakefile`)
  - ↪ facilitate repository setup upon cloning

```
$> git clone <url> && cd <cloned_dir>
$> make setup
```

  - ↪ project releasing using Git-flow made easy

```
$> make start_bump_{major,minor,patch}      # bump version with git-flow
$> make release
```

Summary

Working Together

- Sign-up on Github
  - ↪ `http://github.com`
  - ↪ Best place to share **public** repository

Working Together

- Sign-up on Github
  - ↪ `http://github.com`
  - ↪ Best place to share **public** repository

- Alternative for **private** projects:
  - ↪ Gitlab @ Uni.lu `https://gitlab.uni.lu`

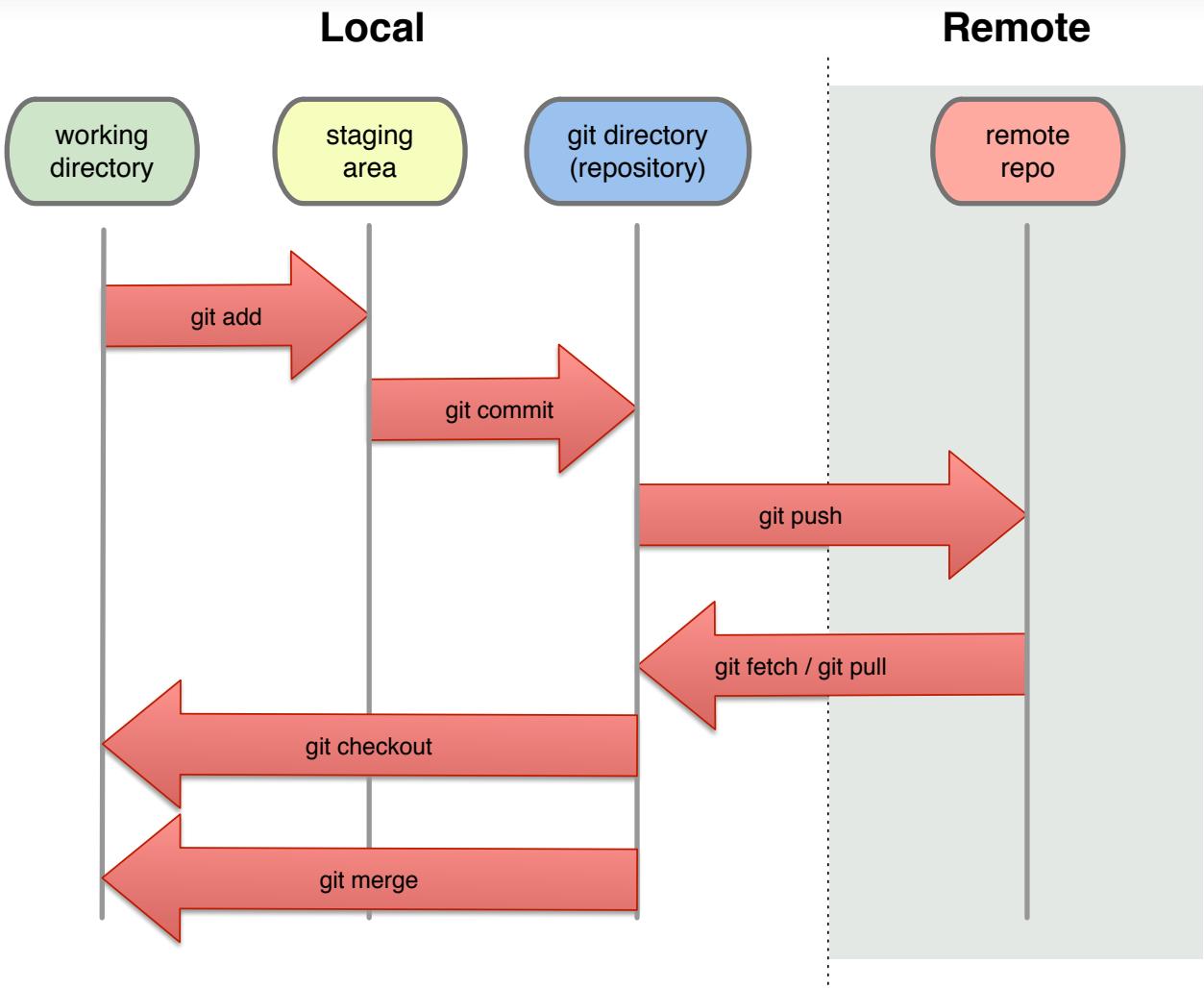
Working Together

- Sign-up on Github
  - ↪ `http://github.com`
  - ↪ Best place to share **public** repository

- Alternative for **private** projects:
  - ↪ Gitlab @ Uni.lu `https://gitlab.uni.lu`

- Setup your own Git server: gitolite `https://github.com/sitaramc/gitolite`
  - ↪ Management through the `gitolite-admins` Git repository (!)
  - ↪ A single user (`git`) to interact with all repositories
    - ✓ map users though their (multiple) SSH keys
    - ✓ fine-grained access control

Working with remotes



Remotes

```
$> git remote [-v]
```

- Other clones of the same repository
  - ↪ Can be local (another checkout) or remote (coworker, central server)
  - ↪ default remotes for push and pull actions: `origin`
    - ✓ `origin` is set upon clone

Remotes

```
$> git remote [-v]
```

- Other clones of the same repository
  - ↪ Can be local (another checkout) or remote (coworker, central server)
  - ↪ default remotes for push and pull actions: `origin`
    - ✓ `origin` is set upon clone

Your Turn!

```
$> cd /tmp/tutorials
$> git remote
origin
$> git remote -v
origin  https://github.com/ULHPC/tutorials.git (fetch)
origin  https://github.com/ULHPC/tutorials.git (push)
```

Adding Remotes

```
$> git remote add <name> <url>
```

Adding Remotes

```
$> git remote add <name> <url>
```

Your Turn!

- Fork the ULHPC/tutorials (as <yourlogin>/tutorials)
- Clone and add the upstream remote to the original repository

```
$> git clone https://github.com/<yourlogin>/tutorials.git /tmp/fork
$> cd /tmp/fork
$> git remote add upstream  https://github.com/ULHPC/tutorials.git
$> git remote -v
origin      https://github.com/<yourlogin>/tutorials.git (fetch)
origin      https://github.com/<yourlogin>/tutorials.git (push)
upstream    https://github.com/ULHPC/tutorials.git (fetch)
upstream    https://github.com/ULHPC/tutorials.git (push)
```

Removing Remotes

```
$> git remote rm <name>
```

Removing Remotes

```
$> git remote rm <name>
```

Your Turn!

```
$> cd /tmp/fork
$> git remote
origin
upstream
$> git remove rm upstream
```

Remote Branches

- Branches on remotes are represented locally as: `<remote>/<branch>`
  - ↪ **Ex**: `origin/master`

Remote Branches

- Branches on remotes are represented locally as: `<remote>/<branch>`
  - ↪ **Ex**: `origin/master`

Tracking Remote Branches

- You can track a remote branch `<remote>/<branch>`
  - ↪ assuming you have previously `fetch` the remote `origin`
  - ↪ creates the local branch `<branch>`

```
$> git branch --track <branch> origin/<branch>
```

Tracking Remote Branches

```
$> git branch --track <branch> origin/<branch>
```

Your Turn!

```
$> cd /tmp/tutorials
$> git branch -a
* devel
  remotes/origin/HEAD -> origin/devel
  remotes/origin/devel
  remotes/origin/production
$> git branch --track production origin/production
Branch production set up to track remote branch production from
origin.
$> git branch
* devel
  production
```

Pushing to your remote

```
$> git push [<remote>]
```

- Transfer local commits of the **current** branch to a remote.
  ↪ push to `origin` by default, assuming the current branch is tracked

Pushing to your remote

```
$> git push [<remote>]
```

- Transfer local commits of the **current** branch to a remote.
  ↪ push to `origin` by default, assuming the current branch is tracked

Your Turn!

```
$> cd /tmp/fork
$> git remote
origin
upstream
$> touch new-file
$> git add new-file
$> git commit -m "add"
```

```
$> git push
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), done.
Total 10 (delta 4), reused 0 (delta 0)
To git@github.com:<yourlogin>/documents.git
   671eb88..c798919  devel -> devel
```

Pulling from remotes

```
$> git pull [--rebase] [<remote>]          # --rebase = DANGER!
```

- fetch all commits from the remote **and** merge (or rebase)
  - ↪ allows for easy to use, equivalent to the **advanced** alternative:

```
$> git fetch [<remote>]
$> git merge <remote>/<branch>      # 'git rebase' if --rebase
```

  - ↪ `fetch`: allows for inspection and manual merging of remote changes

Pulling from remotes

```
$> git pull [--rebase] [<remote>]          # --rebase = DANGER!
```

- fetch all commits from the remote **and** merge (or rebase)
  - ↪ allows for easy to use, equivalent to the **advanced** alternative:

```
$> git fetch [<remote>]
$> git merge <remote>/<branch>      # 'git rebase' if --rebase
```

  - ↪ `fetch`: allows for inspection and manual merging of remote changes

Your Turn!

```
$> cd /tmp/tutorials
$> git pull        # OR git up
Updating ae97dae..06576e0
[...]
2 files changed, 4 insertions(+), 5 deletions(-)
```

Publish a (local) branch on a remote

```
$> git push -u origin <branch>
```

```
$> git flow feature publish <name>
```

Publish a (local) branch on a remote

```
$> git push -u origin <branch>
```

```
$> git flow feature publish <name>
```

- If you want to **delete** a **remote** branch

```
$> git push origin --delete <branch>                    # DANGER!
```

Publish a (local) branch on a remote

```
$> git push -u origin <branch>
```

```
$> git flow feature publish <name>
```

- If you want to **delete** a **remote** branch

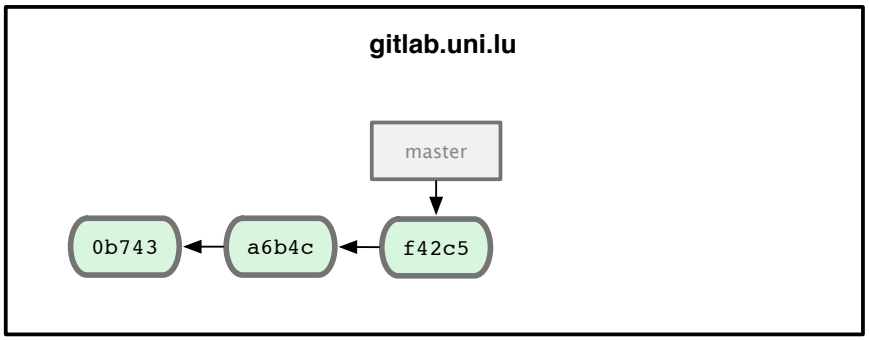
```
$> git push origin --delete <branch>                    # DANGER!
```

Your Turn!

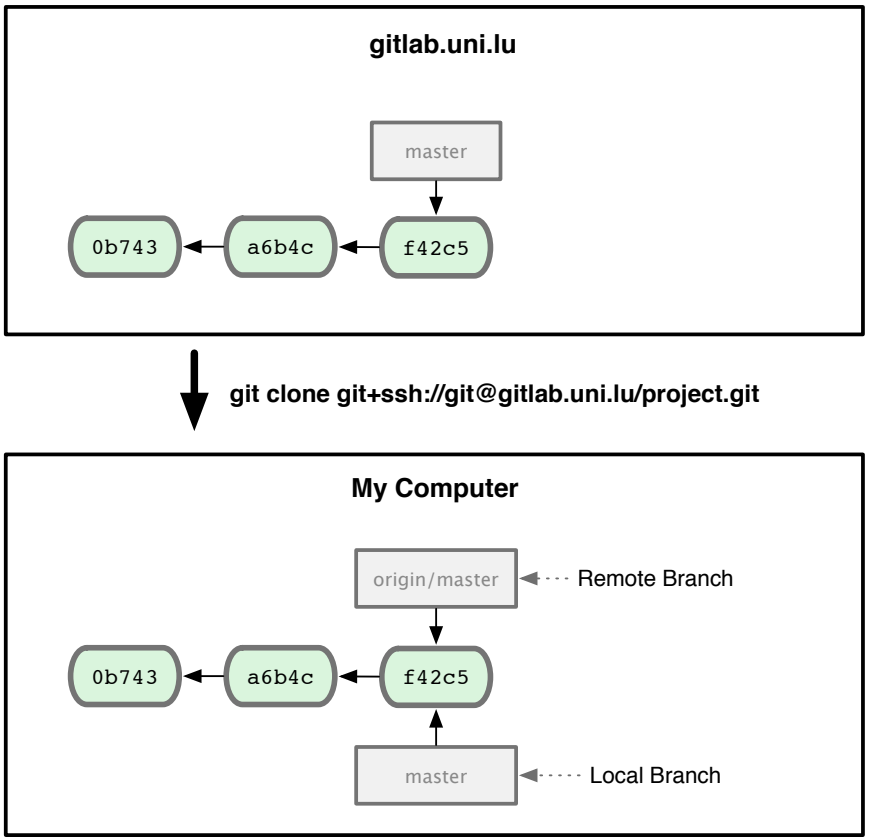
```
$> cd /tmp/fork
$> git flow feature start toto
$> git flow feature publish toto
```

```
$> git branch -a
$> git push origin --delete \
      feature/toto
```

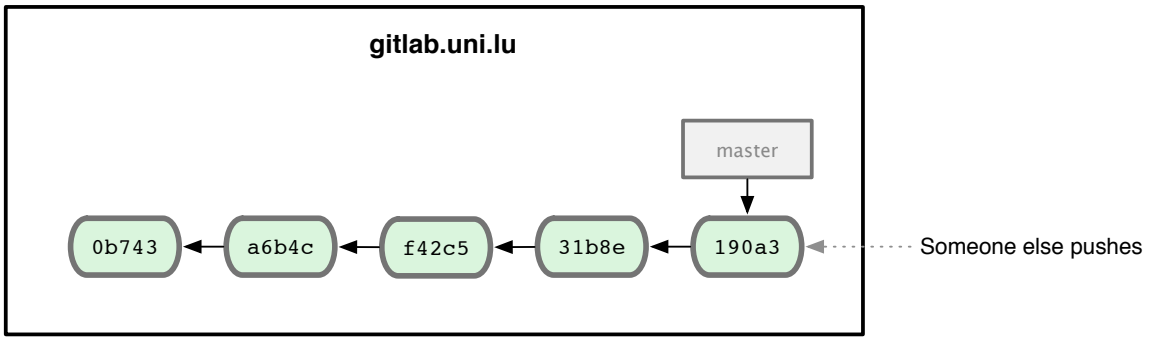
Putting it all together

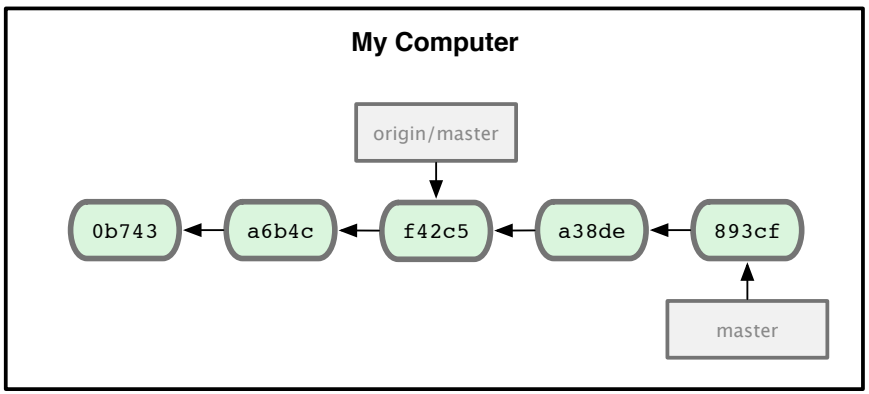


Putting it all together

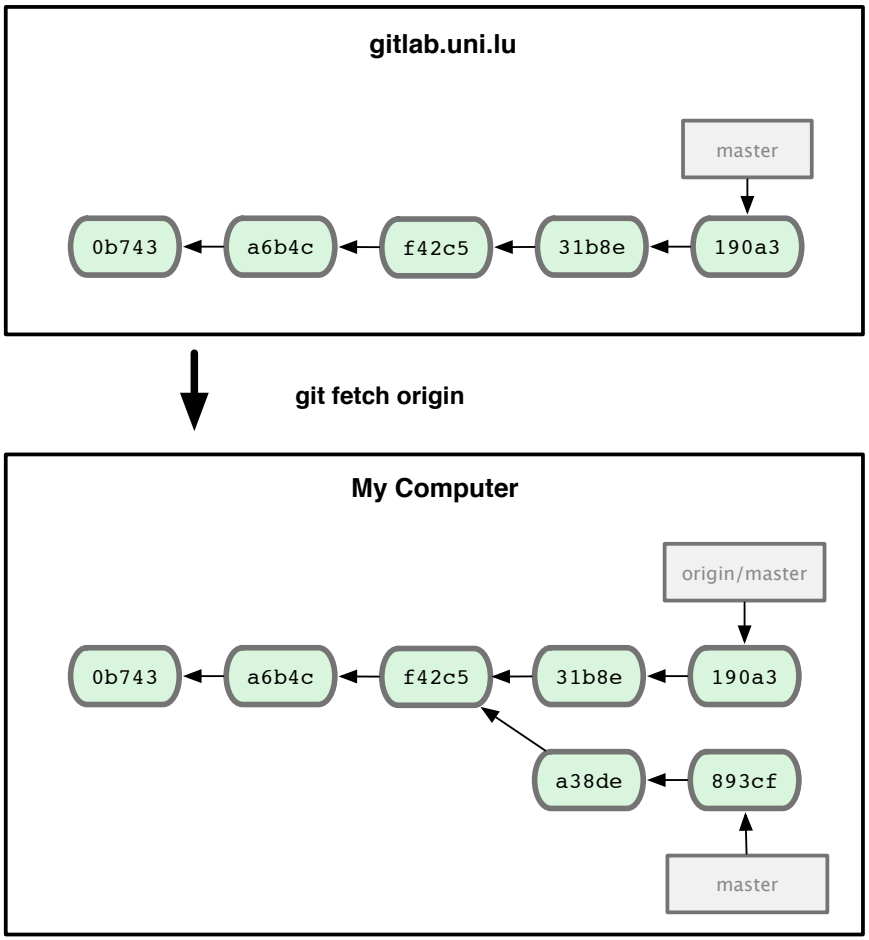


Putting it all together

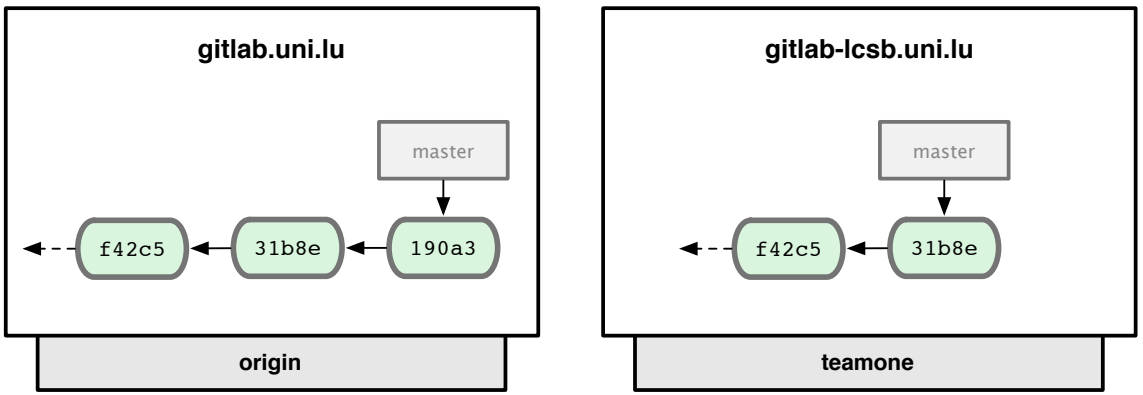




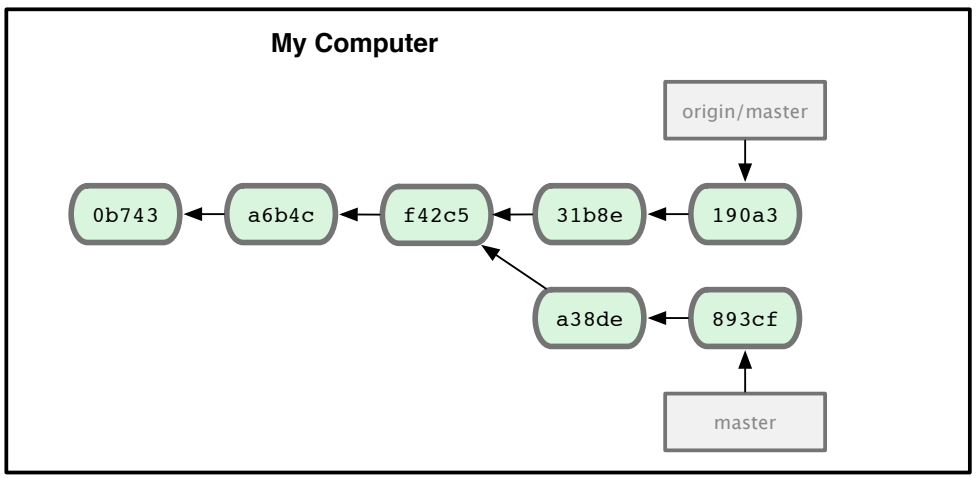
Putting it all together



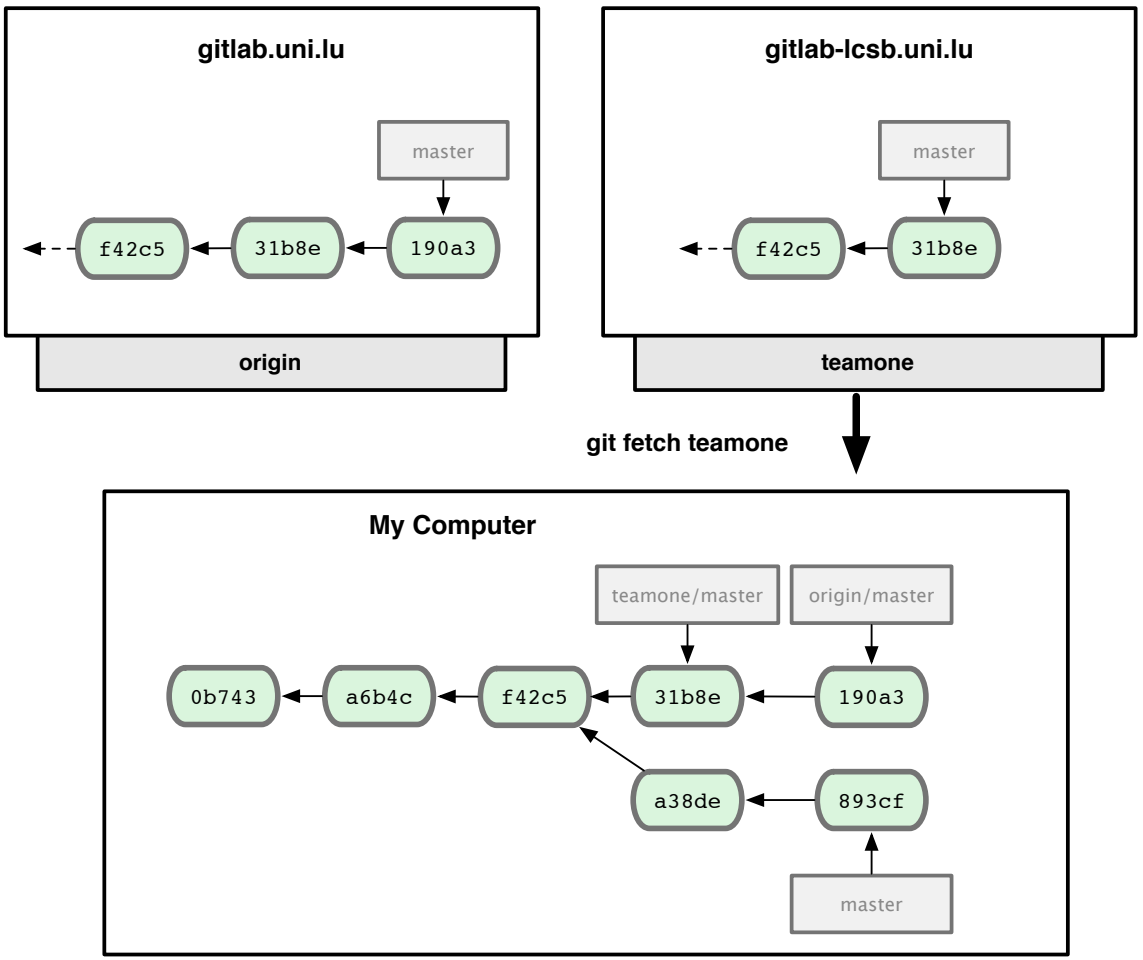
Putting it all together



git remote add teamone git://gitlab-lcsb.uni.lu



Putting it all together



Summary

Summary

Git Submodules

```
$> git submodule add [-b <branch>] <url> <subdir>
```

- **Git submodule**: repository nested within another repository.
  - ↪ see it as a read-only snapshot
  - ↪ make symbolic links to the submodules files
- State saved in `.gitmodules` (git root directory)

```
[submodule ".submodules/Makefiles"]
    path = .submodules/Makefiles
    url = https://github.com/Falkor/Makefiles
```

- Explicit initialization is **mandatory**
  - ↪ **before** cloning `git clone --recursive`
  - ↪ **after** cloning `git submodule init && git submodule update`

Git Submodules - Update

```
$> git submodule add \
       https://github.com/Falkor/Makefiles .submodules/Makefiles
```

- You might need to **update** the submodules after `fetch` / `pull`
- You might wish to **upgrade** the submodules to the latest version

```
$> git submodule init
$> git submodule update
$> git submodule foreach \
     'git fetch origin; \
      git checkout $(git rev-parse --abbrev-ref HEAD); \
      git reset --hard origin/$(git rev-parse --abbrev-ref HEAD); \
      git submodule update --recursive; git clean -dfx'
```

- See `make upgrade` of this Makefile for repositories

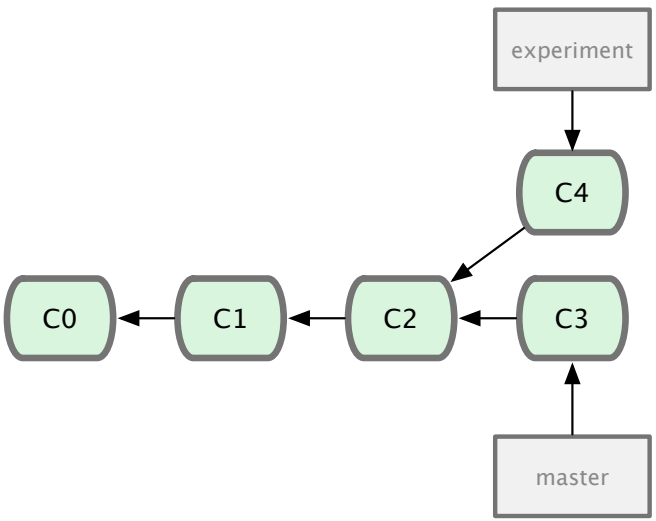
Summary

Rebasing

```
$> git rebase <branch>                    # DANGER! Rewrites the tree
```

- Basic (**3-ways**) merging via `git merge` creates a new commit

```
(master)$> git merge experiment
```

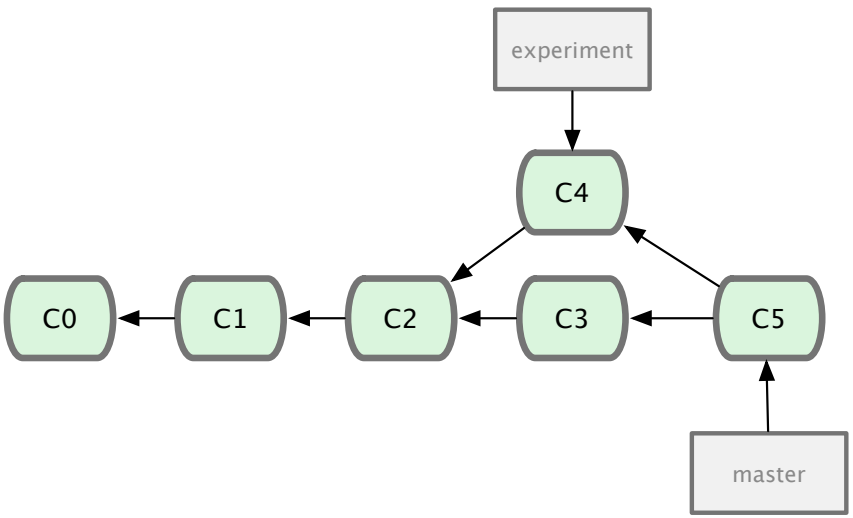


Rebasing

```
$> git rebase <branch>                    # DANGER! Rewrites the tree
```

- Basic (**3-ways**) merging via `git merge` creates a new commit

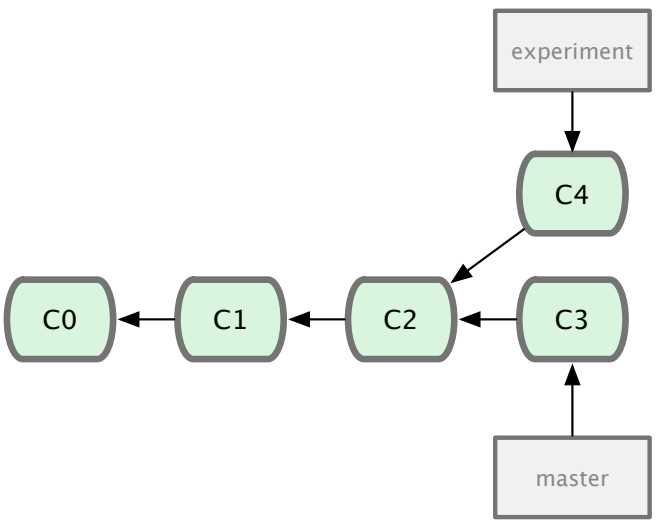
```
(master)$> git merge experiment
```



Rebasing

```
$> git rebase <branch>                    # DANGER! Rewrites the tree
```

- **Rebasing**: **Linear** alternative to merging
  - ↪ create a patch of the introduced change (in C4)
  - ↪ reapply it on top (of C3) to create C4'

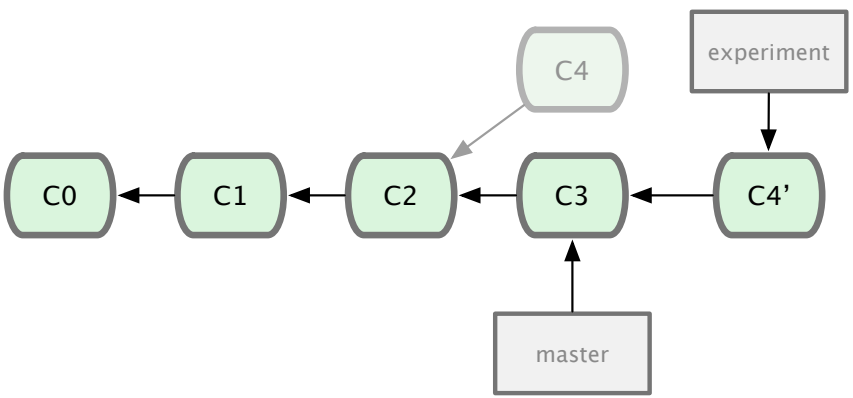


Rebasing

```
$> git rebase <branch>                    # DANGER! Rewrites the tree
```

- **Rebasing**: **Linear** alternative to merging
  - ↪ create a patch of the introduced change (in C4)
  - ↪ reapply it on top (of C3) to create C4'

```
(master)$> git checkout experiment
(experiment)$> git rebase master
```

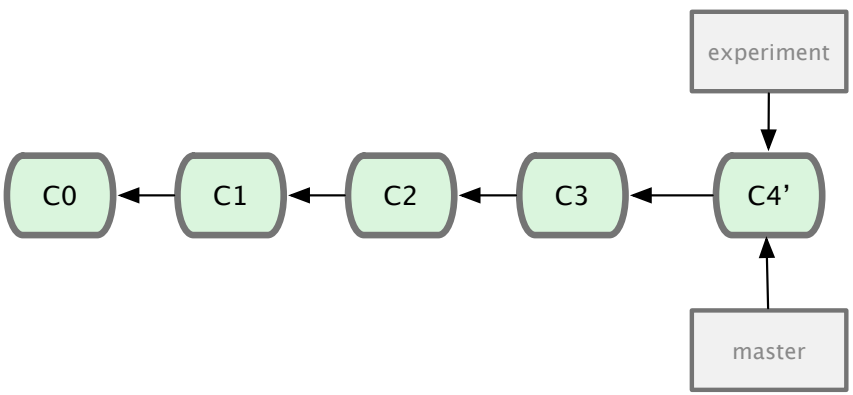


Rebasing

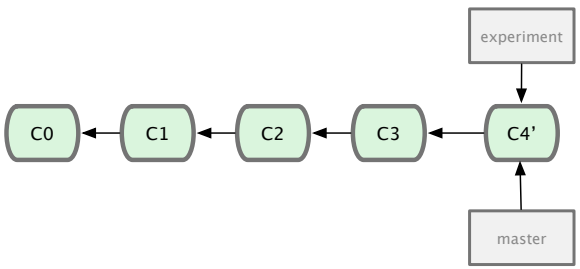
```
$> git rebase <branch>                    # DANGER! Rewrites the tree
```

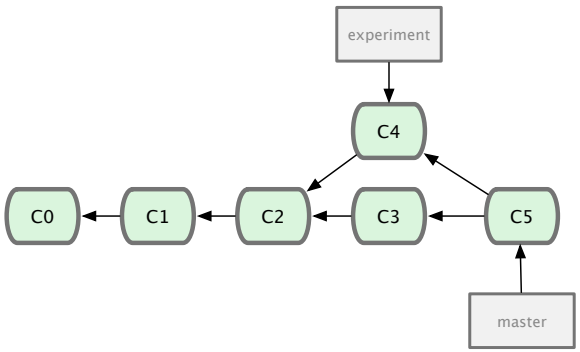
- **Rebasing**: **Linear** alternative to merging
  - ↪ create a patch of the introduced change (in C4)
  - ↪ reapply it on top (of C3) to create C4'

```
(experiment)$> git checkout master
(master)$> git merge experiment
```



Rebasing (left) vs. Merging (right)





- Rebasing ensure your commits apply cleanly on a (remote) branch

Never rebase published code!

Summary

Git-svn: using Git with Subversion

```
$> git svn clone [-s] <svn-url>   # checkout an SVN repository
```

- `-s`: standard SVN layout with `trunk/`, `branches/` and `tags/`
- clone into `master` – you **shall** work in another branch **Ex**: `work`

  ```
  $> git checkout -b work
  ```

  ↪ delegate all interactions with SVN repository with `master`
  ↪ thus make all your (local) commits into the `work` branch

Git-svn: using Git with Subversion

```
$> git svn clone [-s] <svn-url>   # checkout an SVN repository
```

- `-s`: standard SVN layout with `trunk/`, `branches/` and `tags/`
- clone into `master` – you **shall** work in another branch **Ex**: `work`

  `$> git checkout -b work`

  ↪ delegate all interactions with SVN repository with `master`

  ↪ thus make all your (local) commits into the `work` branch

```
$> git svn rebase          # fetch revisions from SVN and rebase
```

- **Important**: always do that from the `master` branch!

  `(work)$> git checkout master`

  `(master)$> git svn rebase`

Git-svn: commit to Subversion

```
$> git svn dcommit     # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(work)$> git checkout master
(master)$> git svn rebase
```

1. rebase the master branch with the SVN repository

Git-svn: commit to Subversion

```
$> git svn dcommit     # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(master)$> git checkout work
(work)$> git rebase master
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master

Git-svn: commit to Subversion

```
$> git svn dcommit     # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(work)$> git log -graph -oneline -decorate                    # OR git gr
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master
3. ensure everything is fine

Git-svn: commit to Subversion

```
$> git svn dcommit    # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(work)$> git checkout master
(master)$> git merge -no-ff work
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master
3. ensure everything is fine
4. force 3-ways merge your local commit

Git-svn: commit to Subversion

```
$> git svn dcommit     # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(master)$> git commit -amend
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master
3. ensure everything is fine
4. force 3-ways merge your local commit
5. edit (amend) the last commit for your SVN dudes

Git-svn: commit to Subversion

```
$> git svn dcommit    # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(master)$> git svn dcommit
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master
3. ensure everything is fine
4. force 3-ways merge your local commit
5. edit (amend) the last commit for your SVN dudes
6. **Finally** commit on the SVN server

Git-svn: commit to Subversion

```
$> git svn dcommit    # create an SVN revision for each commit
```

AFTER you sanitize the 'master' branch!

```
(master)$> git checkout work
```

1. rebase the master branch with the SVN repository
2. go back to the work branch and rebase with master
3. ensure everything is fine
4. force 3-ways merge your local commit
5. edit (`amend`) the last commit for your SVN dudes
6. **Finally** commit on the SVN server
7. Go back to the 'work' branch!

Summary

Shell Integration

- Git Completion – Git flow completion

Colored PS1

- bash: integrate `__git_ps1()` function in your `PS1` variable
  - ↪ normally part of the `bash-completion` package
  - ↪ See integration in the ULHPC/dotfiles repository

```
$> export GIT_PS1_SHOWDIRTYSTATE=1      # you probably want that
```

- zsh: agnoster theme / powerline
  - ↪ Mac OS instructions

- On **CentOS/Redhat**, you have to source the correct file

```
$> ln -s /usr/share/git-core/contrib/completion/git-prompt.sh /etc/profile.d/
```

Revision Selection and Log Filtering

- When referring to a commit <commit>:

Symbol	Description
<commit>^	Parent of the commit <commit>
HEAD^	previous commit (parent of HEAD)
<commit>~<n>	<n>-th parent of <commit>

```
$> $ git log --pretty=format:'%h %s' --graph
* 734713b fixed refs handling, added gc auto, updated tests  # HEAD
*   d921970 Merge commit 'phedders/rdocs'                    # HEAD^
|\
| * 35cfb2b Some rdoc changes
* | 1c002dd added some blame and merge stuff     # d921970^ OR HEAD~2
|/
* 1c36188 ignore *.gem                          # HEAD~3
```

External Merge and Diff Tools

- Git offers visual diff/merge tools, assuming you configured it:

```
$> git config --global merge.tool sourcetree
```

```
$> git difftool [<commit>]                                    diff GUI
```

External Merge and Diff Tools

- Git offers visual diff/merge tools, assuming you configured it:

```
$> git config --global merge.tool sourcetree
```

```
$> git difftool [<commit>]                                    diff GUI
```

```
$> git mergetool [<path>...]          # resolving merge conflicts
```

External Merge and Diff Tools

- Git offers visual diff/merge tools, assuming you configured it:

```
$> git config --global merge.tool sourcetree
```

```
$> git difftool [<commit>]                                    diff GUI
```

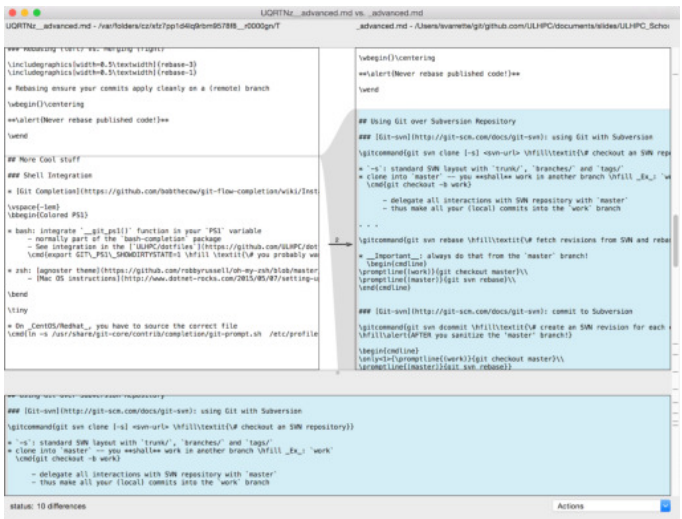
```
$> git mergetool [<path>...]          # resolving merge conflicts
```

- You can set up **another** graphical merge-conflict-resolution tool
  - ↪ List the available tools: `git mergetool --tool-help`
  - ↪ **Mac OS**: `git config --global merge.tool opendiff`
  - ↪ **Linux**: `git config --global merge.tool kdiff3`
  - ↪ Cross-platform: P4Merge (download)

```
$> brew cask install p4merge # on Mac OS, using Homebrew and Cask
```

Using P4Merge as diff/merge tool

```
git config --global merge.tool p4mergetool
git config --global mergetool.p4mergetool.trustexitcode   false
git config --global mergetool.p4mergetool.keeptemporaries false
git config --global mergetool.p4mergetool.keepbackup      false
git config --global mergetool.p4mergetool.cmd \
    $BASE $LOCAL $REMOTE $MERGED
```



- **Alternatives** (mostly Mac OS)
  - ↪ Kaleidoscope
  - ↪ Araxis Merge
  - ↪ DeltaWalker
  - ↪ DiffMerge (free)
  - ↪ SourceTree (free)

Interesting Git plugins

- Git-extra: Additionnal GIT utilities https://github.com/tj/git-extras
  - ↪ repo summary, repl, changelog population, author commit...
- Git-crypt: Transparent file encryption in git
  - ↪ file to automatically encrypt specified in `.gitattributes` file
  - ↪ beware of a hook to check status

```
$> git-crypt init          # Initialize repo with YOUR GPG key
```

Interesting Git plugins

- Git-extra: Additionnal GIT utilities https://github.com/tj/git-extras
  - ↪ repo summary, repl, changelog population, author commit...
- Git-crypt: Transparent file encryption in git
  - ↪ file to automatically encrypt specified in `.gitattributes` file
  - ↪ beware of a hook to check status

```
$> git-crypt init              # Initialize repo with YOUR GPG key
```

```
$> git-crypt [un]lock                    # Lock/Unlock the files
```

Git-crypt

- You need also to enable a git pre-commit hook
  - ↪ to avoid accidentally adding unencrypted files – see issue #45.
  - ↪ Example of such a pre-commit hook: this gist – raw version
    - ✓ to be placed as `.git/hooks/pre-commit`
- **note**: these hooks are **local** to your working directory

```
$> curl <url/to/raw/gist> -o .git/hooks/pre-commit
$> chmod +x .git/hooks/pre-commit
```

Git-crypt

```
$> git-crypt add-gpg-user USER_ID      # Add (GPG) collaborator
```

```
$> git-crypt status          # Status - raise WARNING on problem
```

Git-crypt

```
$> git-crypt add-gpg-user USER_ID      # Add (GPG) collaborator
```

```
$> git-crypt status          # Status - raise WARNING on problem
```

```
# .gitattributes
# specify which files to encrypt using git-crypt
# see https://www.agwa.name/projects/git-crypt/

# Certificate private keys
*.key filter=git-crypt diff=git-crypt
# Host SSH private keys
*ssh_*_key filter=git-crypt diff=git-crypt
```

Other Cool Stuff

Stashing

- Move changes to a separate “stash”.

```
$> git stash
$> git stash pop
$> git stash list
$> git stash apply
$> git stash drop
$> git stash clear
```

Interactive Rebase

```
$> git rebase -i <branch>
```

Other Cool Stuff

Stashing

- Move changes to a separate “stash”.

```
$> git stash
$> git stash pop
$> git stash list
$> git stash apply
$> git stash drop
$> git stash clear
```

Interactive Rebase

```
$> git rebase -i <branch>
```

- **Git hooks**:
  - ↪ Located in `.git/hooks/`
  - ↪ scripts run at various stages of Git operation
  - ↪ useful to perform lint actions for instance before pushing

Thank you for your attention...

Questions?

Sebastien Varrette
mail: Sebastien.Varrette@uni.lu
Office E-007
Campus Kirchberg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg