

Introduction to GIT

The distributed SCM

Bart Trojanowski
<bart@jukie.net>
<http://www.jukie.net/~bart/>

Overview

Concepts

History

Usage

Wrap up

>

Concepts

- **Source Control Management**
 - track changes to files
 - repository / database of changes
 - working directory / current state

- **Centralized SCM**
 - server: single database
 - client: working directory & state

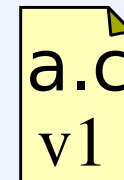
- **Decentralized SCM**
 - anyone can be a server
 - repository coupled with working directory
 - complete history
 - disconnected operation

>

SCM components

- Repository contents

- objects / blobs / diffs / deltas / patches

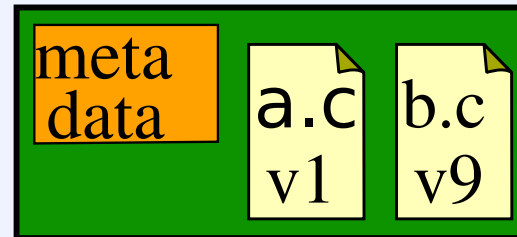


>

SCM components

□ Repository contents

- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions

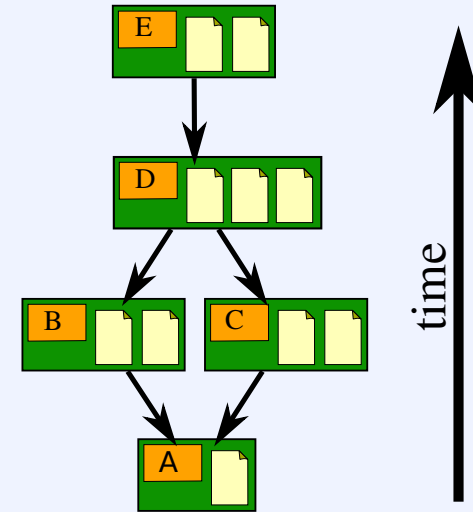


>

SCM components

□ Repository contents

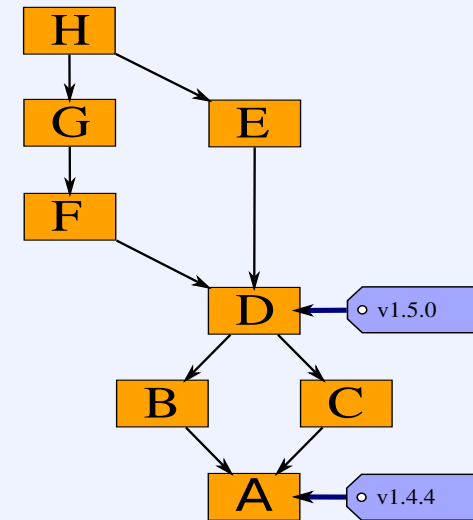
- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history



SCM components

□ Repository contents

- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history
- tags / labels



>

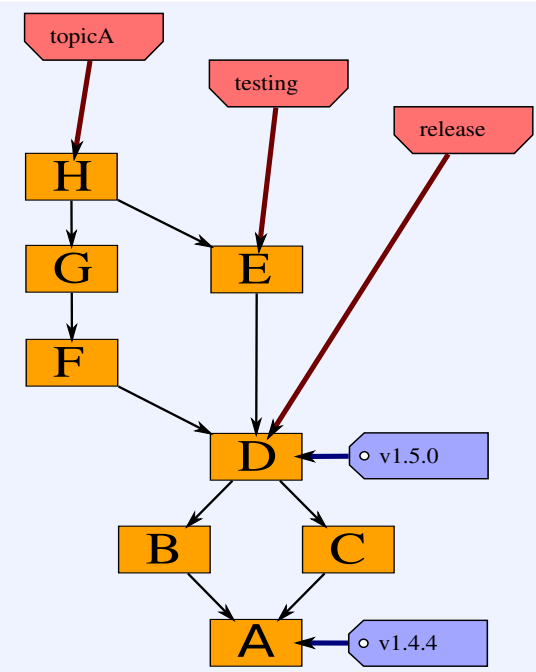
SCM components

□ Repository contents

- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history
- tags / labels
- branches / heads

□ Working directory

- files
- list of files to add/delete



>

SCM operations

- Modify

- checkout
- add, delete, and rename
- commit

- Information

- status
- diff
- log

- Tagging

- Branching

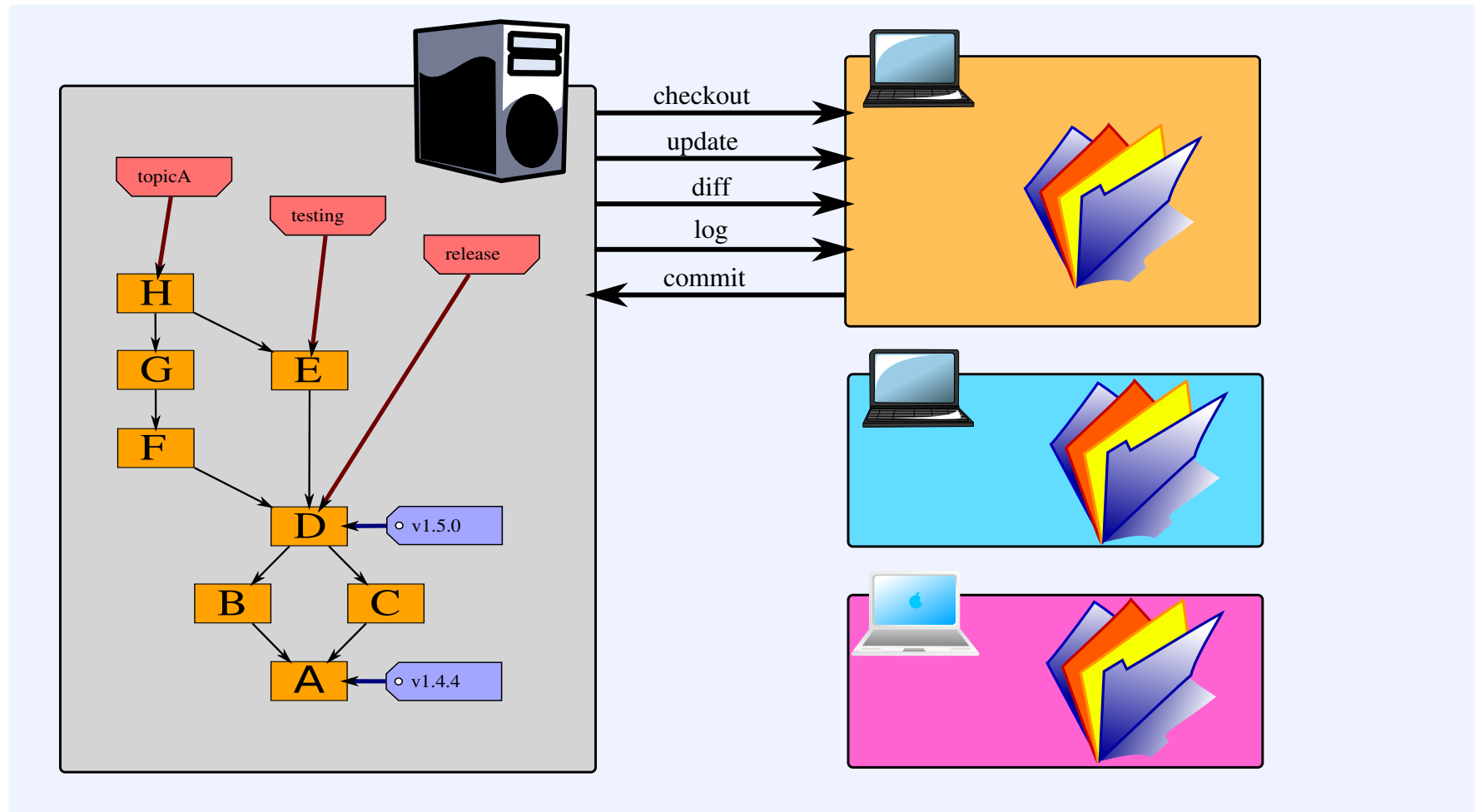
>

Centralized SCM

- Checkout
 - create working directory
 - get state from server
 - populate working directory
- Commit
 - package up all changes
 - send changes to server
- Diff
 - get objects or diffs from server
 - display
- Log
 - get history from server
 - display

>

Centralized SCM



Decentralized SCM

Checkout

- get repository state
- update working directory

Commit

- package changes
- update repository

Diff

- display repository diffs

Log

- display repository history

Clone

- create repository
- create working directory
- get history from server
- populate working directory

Pull / Fetch

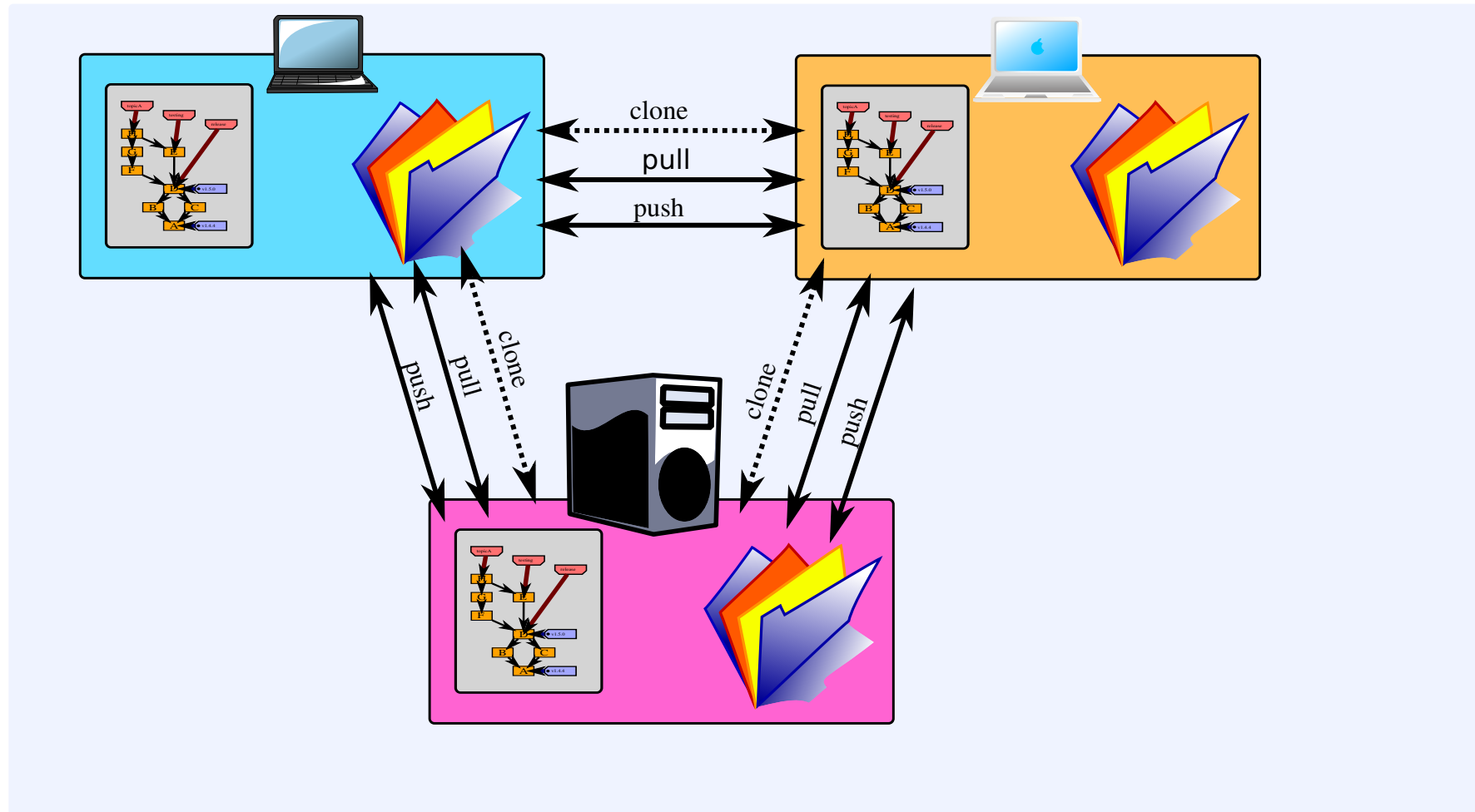
- get changes from server
- update repository

Push

- send changes to server
- update repository

>

Decentralized SCM



Decentralized SCM operations

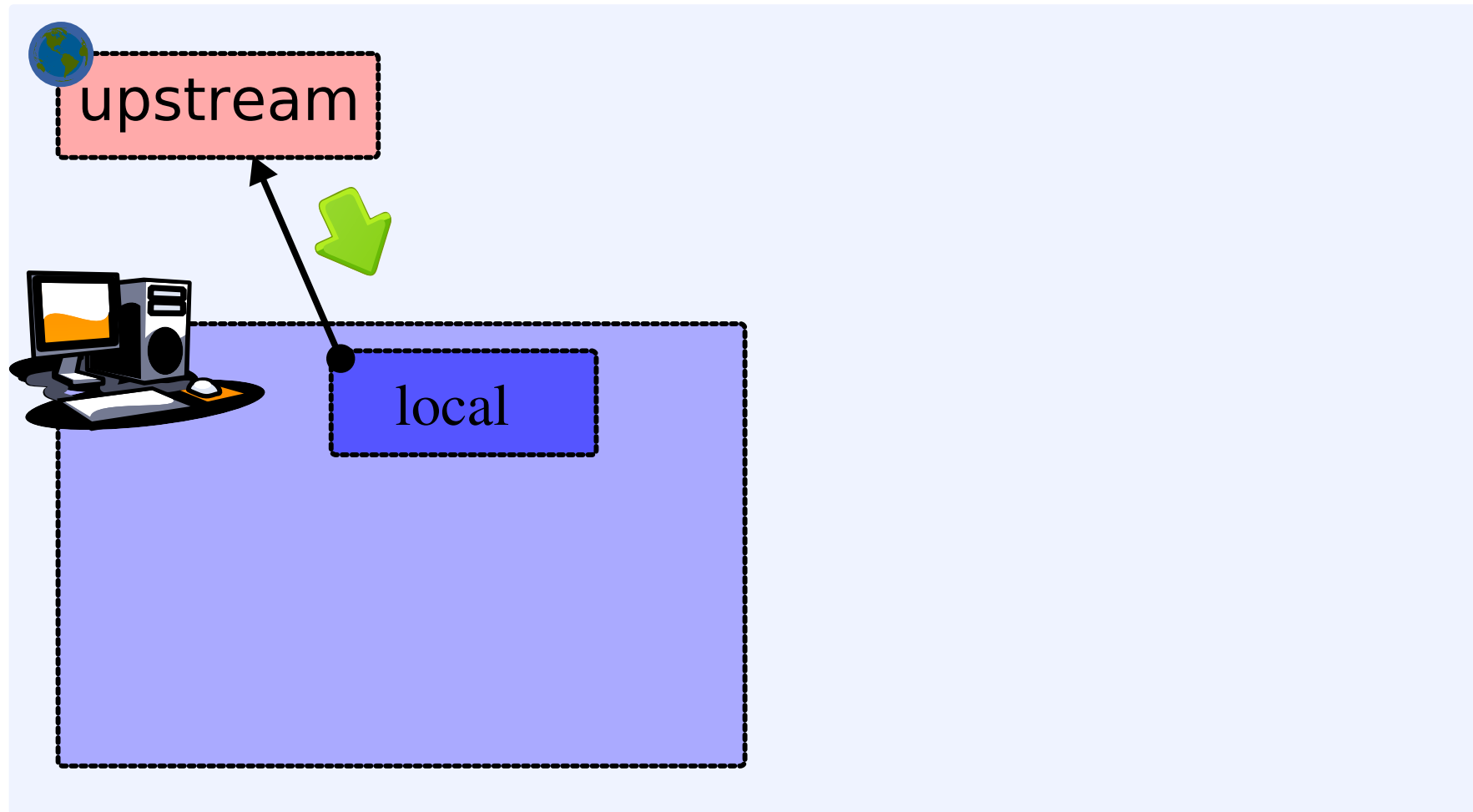


upstream

public repository

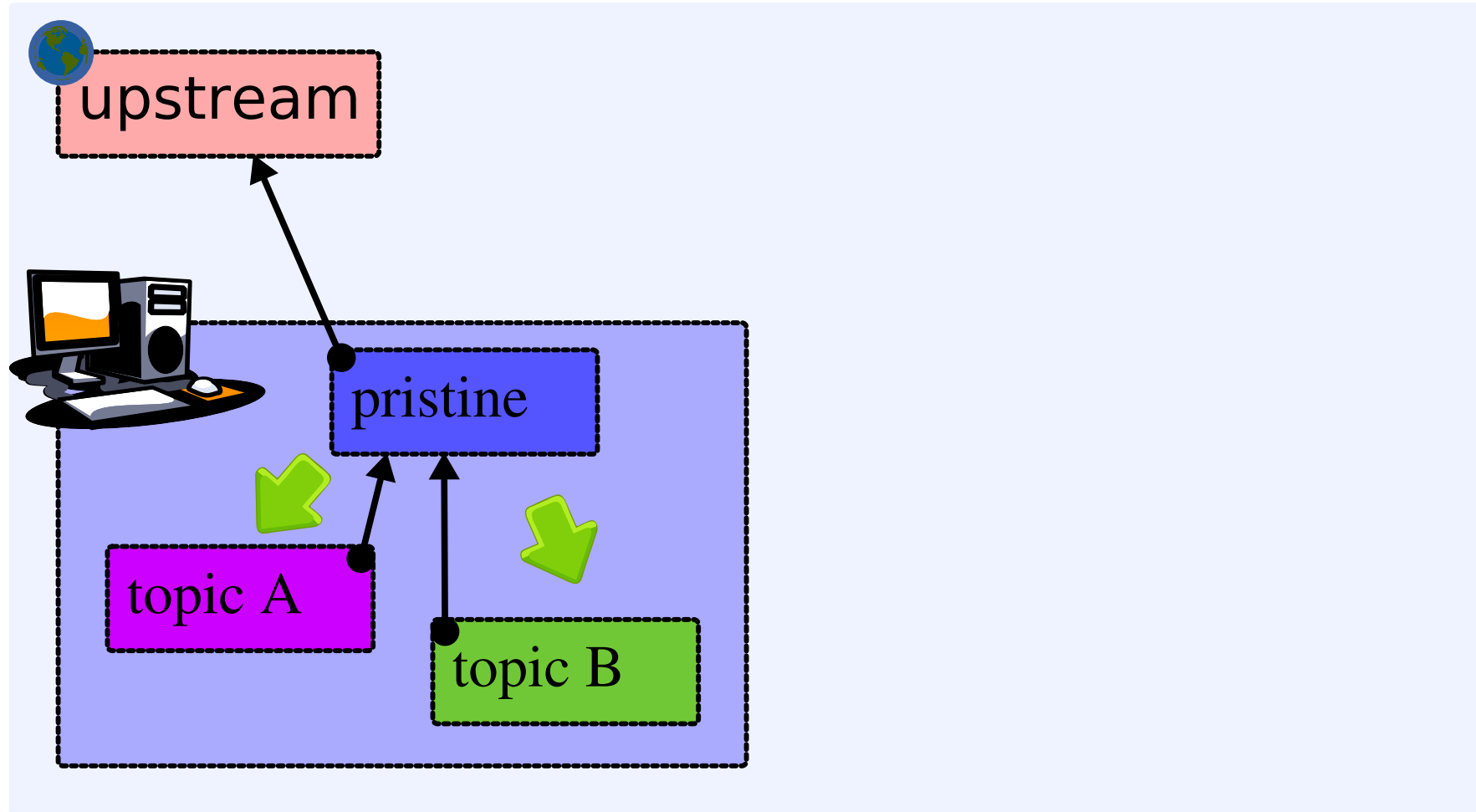


Decentralized SCM operations



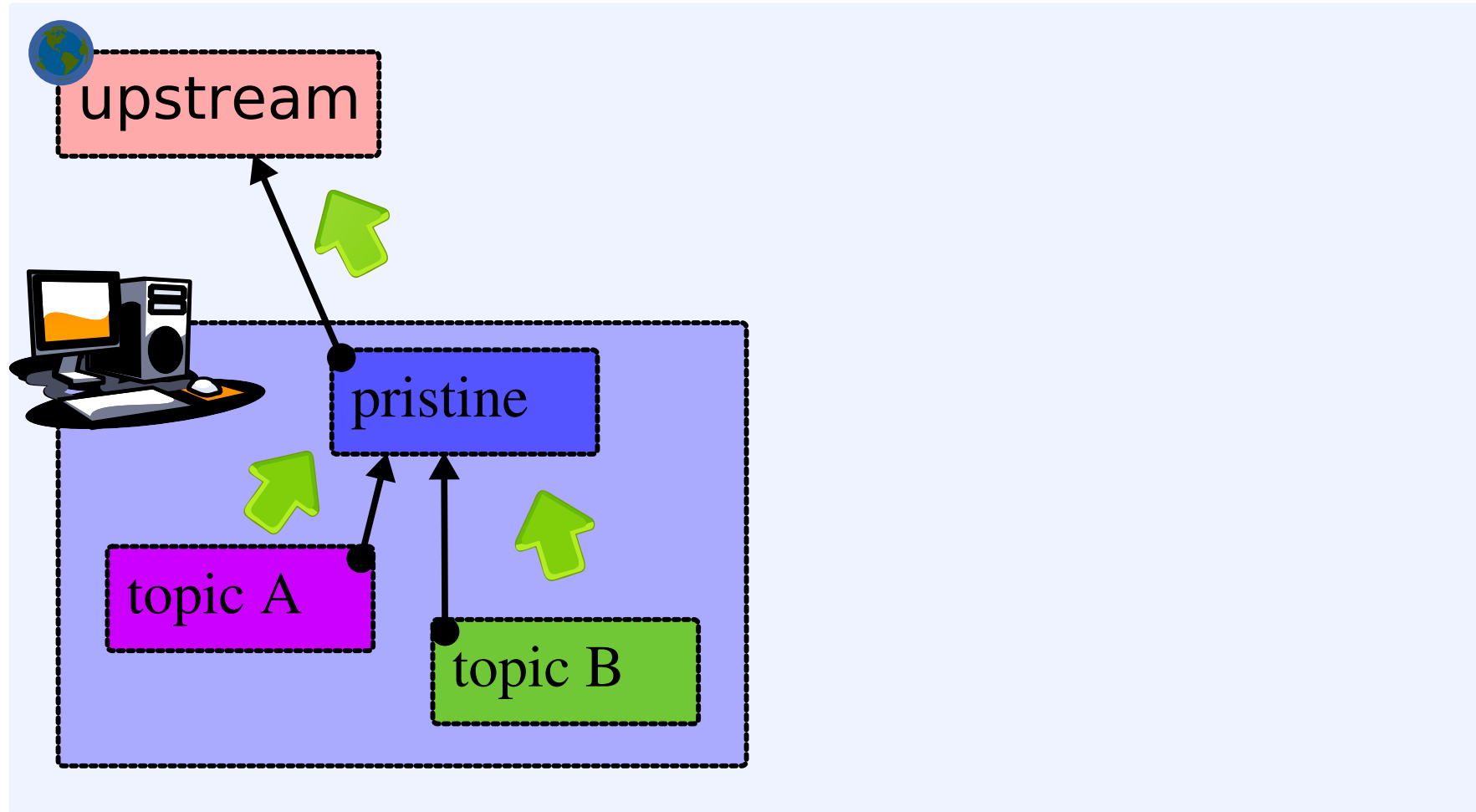
make a local clone

Decentralized SCM operations



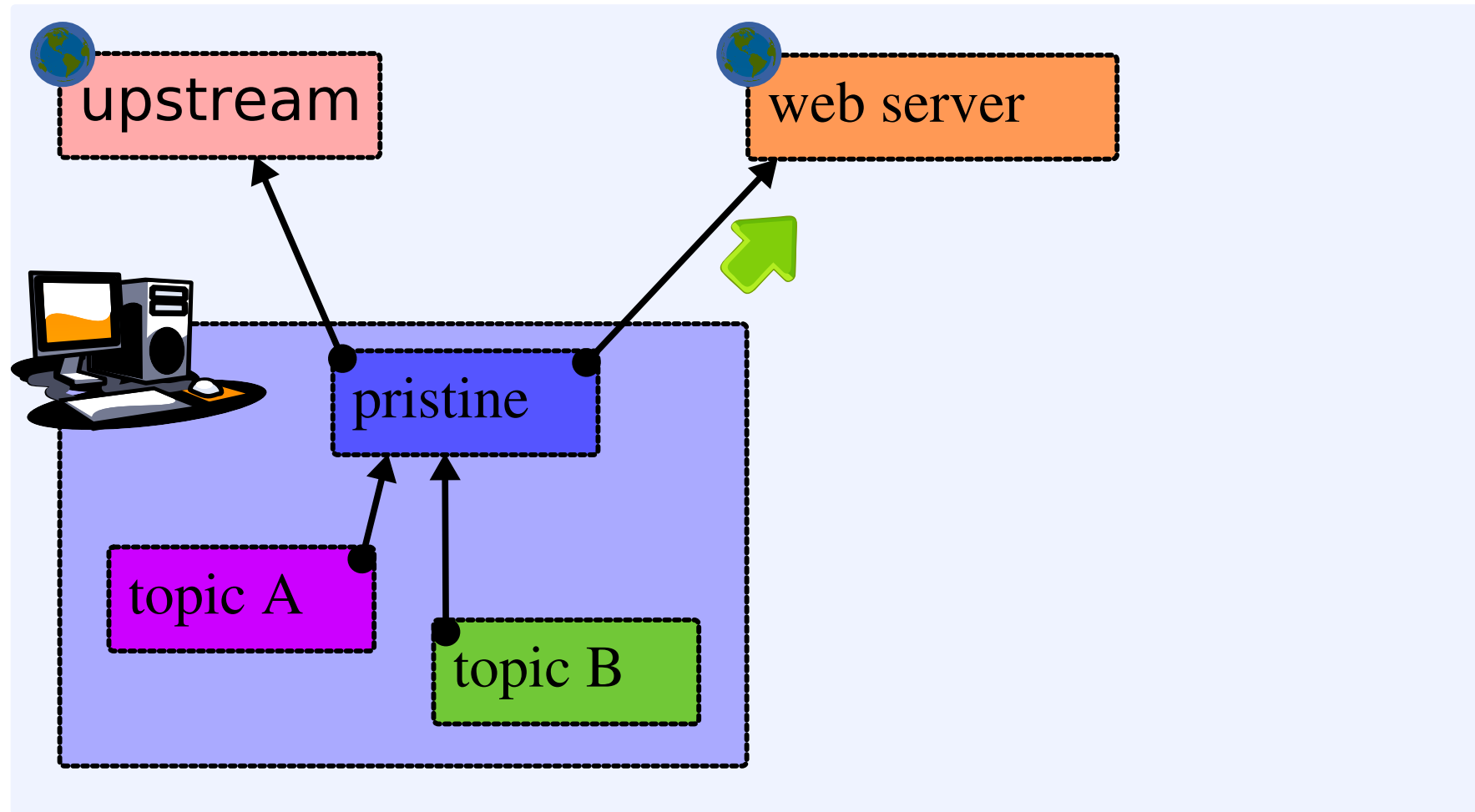
create topic branches

Decentralized SCM operations



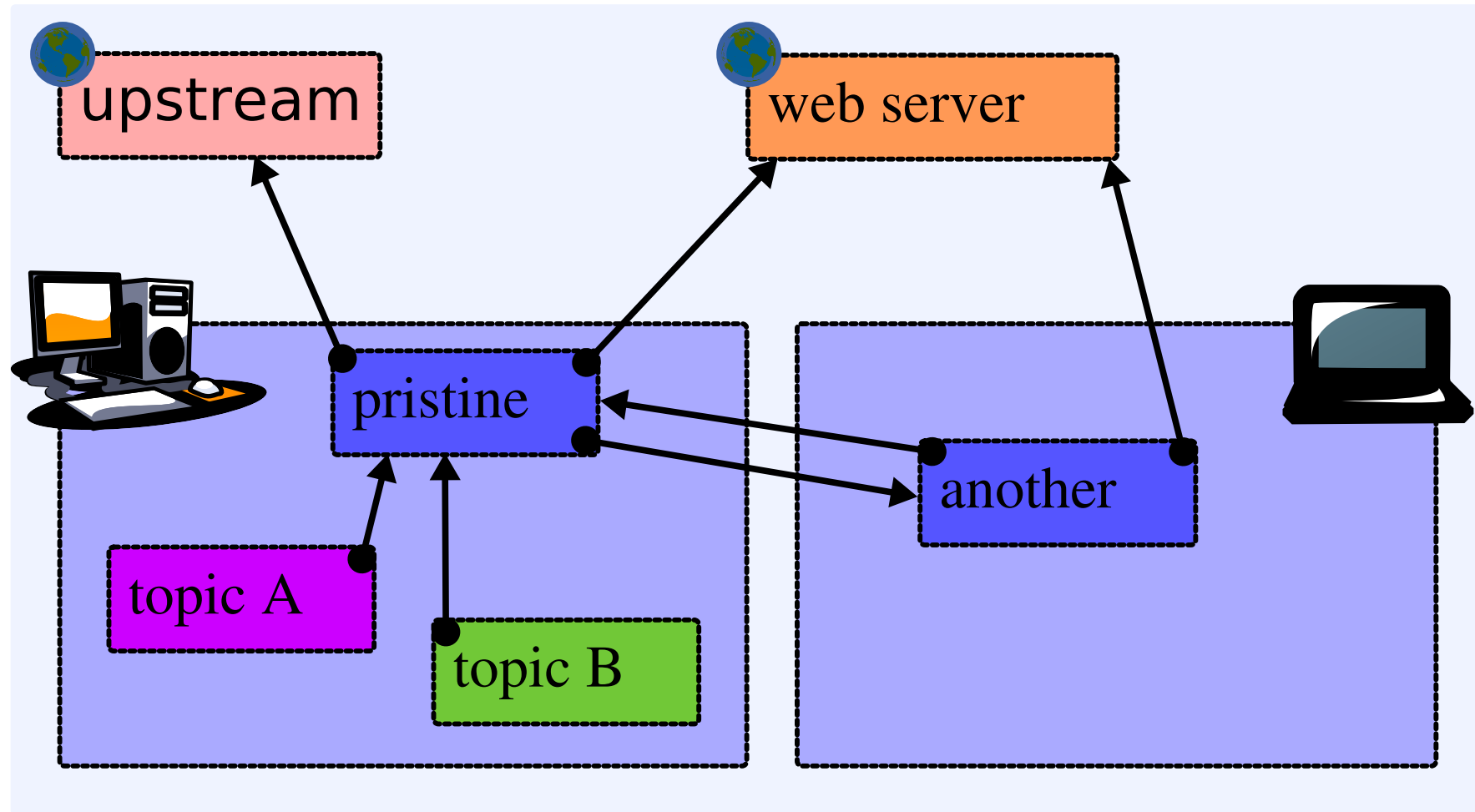
push changes between any repositories

Decentralized SCM operations



publish changes to public server

Decentralized SCM operations



share changes with trusted peers

Is Decentralized good?

YES

Is Decentralized good?

- non-intrusive micro-commits
- detached operation
- no single point of failure
- backups are trivial

>

Birth of GIT

- 2002
 - Linus uses BitKeeper for tracking Linux
 - BK gets better
 - Linux development scales better
- April 6, 2005
 - BitMover drops free license
 - Linus writes his own SCM, GIT
- April 18, 2005
 - GIT can merge
- June 16, 2005
 - GIT is officially used to track Linux

>

GIT gets better

And then realize that nothing is perfect.
Git is just *closer* to perfect than any
other SCM out there.

-Linus

Using GIT

Basics

- Creating a repository
- Creating commits
- Looking at the history

Tagging & Branching

- Creating
- Merging
- Rebase

Patches

- Generating
- Accepting

Decentralized

- Cloning
- Fetch / Pull
- Push
- Daemon
- Remotes

Tools

- Working with other SCMs
- Housekeeping
- Visualization

Tricks

>

Basics : Commands

□ format

```
$ git <options> <command> <cmd-options>
```

□ list of commands

```
$ git help
```

□ usage info

```
$ git init -h
```

```
usage: git-init [--template=...] [--shared]
```

□ detailed help

```
$ git help init
```

```
$ git init --help
```

```
$ man git-init
```

>

Basics : GIT 1.5.0

- easier merging
 - use `git-merge` command

- better remote management
 - remote vs local branch separation
 - common usage patterns are automated
 - new `git-remote` command

- lots of minor changes
 - heavy use of `.git/config` for configuration
 - improvements to `git-add` and `git-rm` commands
 - better documentation
 - ...

>

Basics : Configuration

- global configuration is in `~/.gitconfig`

```
$ git config --global --list
user.name=Bart Trojanowski
user.email=bart@jukie.net
color.branch=auto
color.diff=auto
color.pager=true
color.status=auto
```

- repository configuration is in `repo/.git/config`

```
$ git config --list
section.variable=value
...
```

- changing settings

```
$ git config --global user.name "Bart Trojanowski"
$ git config --global user.email bart@jukie.net
```

Basics : Creating a repository

```
$ mkdir my-project  
$ cd my-project  
$ git init  
Initialized empty Git repository in .git
```

□ Initial .git directory

```
$ du -sh .git  
60K      .git
```

>

Basics : Creating commits

□ adding files

```
$ echo this is a test > test.1
$ echo this is also a test > test.2
$ git add test.1 test.2
$ git commit -a -m "adding some files"
Created initial commit 4bba26ed61c918687d9b34e9fcbef12a6d5bf01a
2 files changed, 2 insertions(+), 0 deletions(-)
create mode 100644 test.1
create mode 100644 test.2
```

□ add everything, recursive

```
$ git add .
```

□ commit just one file, interactive

```
$ git commit file.1
```

>

Basics : Creating commits

□ removing files

```
$ git rm test.2
```

```
$ git commit -a -m"removing a file"
```

```
Created commit b4e0bebe216fd53b540134a659254e192cbce57c  
1 files changed, 0 insertions(+), 1 deletions(-)  
delete mode 100644 test.2
```

>

Basics : Altering commits

□ altering last commit

- make some changes...

```
$ edit file
```

```
$ git commit -a
```

- realize you made a mistake

- fix the code...

```
$ edit file
```

- recreate the commit...

```
$ git commit --amend
```

- avoid if commit was already pushed

- can modify commit message

>

Basics : Current state

□ Show uncommitted file changes

```
$ echo more text >> test.1
$ git status
# On branch master
# Changed but not updated:
#   modified:   test.1
```

□ show the differences

```
$ git diff
--- a/test.1
+++ b/test.1
@@ -1,2 @@
   this is a test
+more text
```

Basics : Cleanup

- revert changes to a file

```
$ git checkout file
```

- forget newly added files

```
$ git reset
```

- drop all changes

```
$ git reset --hard
```

- undo last commit, keeping changes

```
$ git reset --soft HEAD^
```

>

Basics : More cleanup

- delete all untracked files (dangerous)

```
$ git clean -d -x
```

- delete last three commits (dangerous)

```
$ git reset --hard HEAD~3
```

>

Basics : That never happened

- new commit undoing changes of another commit

```
$ git revert HEAD~10
```

○ this is short form for...

```
$ git diff HEAD~10..HEAD~9 | patch -R -p1
```

```
$ git commit -a -m"reverted commit"
```

>

Basics : Looking at the history

□ review commit messages

```
$ git log
```

○ with ranges

```
$ git log -10
```

```
$ git log --since="2 weeks ago"
```

```
$ git log tag..
```

```
$ git log tag1..tag2
```

```
$ git log branch1 branch2 ^common
```

○ limit to commits that alter a path

```
$ git log -- file
```

○ varying levels of information

```
$ git log --no-merges
```

```
$ git log --pretty=oneline
```

```
$ git log --pretty=email
```

```
$ git log --stat --summary
```

>

Basics : Looking at the history (2)

- review changes in commits

 - \$ git whatchanged

 - like git-log, but also lists files that changed

- limit as with log

 - \$ git whatchanged --since="May 10"

 - \$ git whatchanged tag1..tag2 -- file

>

Basics : Looking at the history (3)

- see what changes one commit made (diff)

```
$ git show <commit>
```

- like git-log, but also lists diff's

- look at diffs for all/some commits

```
$ git log -p
```

```
$ git log -p tag..tag1
```

>

Basics : Looking at the history (4)

- who made changes to a file?

```
$ git blame filename
```

- list commits which made a particular change

```
$ git log --pretty=oneline -grep 'sprintf'
```

- shows all commits that add/remove a pattern

```
$ git log --pretty=oneline -S'sprintf' -- onefile
```

- can be limited to a file

>

Basics : Looking at the history (5)

□ searching for a change

```
$ git log \  
    --since="June 5, 2005" \  
    --grep="find this" \  
    --author="bob@domain.com" \  
    --pretty=oneline \  
some-branch -- crypto/*.c
```

□ who did what?

```
$ git log [...] | git shortlog  
○ lists commits made by each author
```

□ what can I search for?

```
$ man git-rev-list
```

>

Commit IDs

□ reference by

○SHA1 sum

```
git show b4e0bebe216fd53b540134a659254e192cbce57c
```

○short SHA1 sum

```
git show b4e0bebe
```

○a tag or branch name

```
git show something
```

○current commit

```
git show HEAD
```

○relative to a commit

```
git show HEAD^
```

```
git show HEAD^^^^^^^^^^
```

```
git show HEAD~10
```

○ranges and exclusions

```
git log HEAD~10..HEAD
```

```
git log --max-count=100 --skip=100
```

```
git log tag branch --not ^something
```

>

Tagging

- symbolic name for a commit
- shorter than SHA1

- **tag a commit**

 - \$ git tag some-name

 - \$ git tag -f some-name

 - \$ git tag another-name b4e0bebe...

- **list existing tags**

 - \$ git tag -l

 - some-name

 - another-name

- **remove a tag**

 - \$ git tag -d some-name

Branching

- always on a branch
- branch is a pointer to a commit
- two types of branches
 - local branches
 - master
 - whatever-you-want
 - remote branches
 - origin/master
 - some-host/their-branch

Branching : creating

creating a branch

```
$ git branch this
$ git branch another b4e0bebe...
$ git branch four-back HEAD~4
```

show local

```
$ git branch
  another
* master
```

show remote

```
$ git branch -r
  origin/master
  origin/foo
```

>

Branching : switching

- switch to an existing branch

 - \$ git checkout some-branch

 - blow away uncommitted changes:

 - \$ git checkout -f some-branch

- switch to a new branch

 - \$ git branch new-branch a-commit

 - \$ git checkout new-branch

 - ... Or ...

 - \$ git checkout -b new-branch a-commit

>

Branching : merging

Merges suck.
Unless you're smart, and use git.

-Linus

Branching : merging

- merge another branch

```
$ git merge some-branch
```

- expecting bugs?

```
$ git merge --no-commit some-branch
```

```
$ fix
```

```
$ git commit -a
```

- resolving conflicts

```
$ git merge some-branch
```

```
$ git status
```

```
$ fix
```

```
$ git add resolved-file
```

```
$ git commit -a
```

Branching : cherries

- have some work on a branch

```
    A---B---C    <- some-work
    /
D---E---F---G    <- master
```

- want to bring one commits to master

```
$ git cherry-pick C
```

```
    A---B---C    <- some-work
    /
D---E---F---G---C' <- master
```

- cherry-pick lets you edit the commit

```
$ git cherry-pick --edit <commit>
```

```
$ git cherry-pick --no-commit <commit>
```

>

Branching : rebase

- have some work on a branch

```
    A---B---C    <- some-work
   /
  D---E---F---G    <- master
```

- could merge

```
$ git checkout -b new-work master
$ git merge some-work
```

```
    A---B---C    <- some-work
   /             \
  D---E           H    <- new-work
   \             /
    F-----G    <- master
```

- or rebase

```
$ git checkout some-work
$ git rebase master

    A---B---C    <- some-work
   /
  D---E---F---G    <- master
```

Branching : changing history with rebase

- made some commits

```
A--B--C--D--E(master)
```

- realized you made a mistake in commit 'B'

- go back

```
$ git checkout HEAD~3
```

```
$ git commit --amend
```

```
  .B'(HEAD)
```

```
 /
```

```
A--B--C--D--E(master)
```

- bring back the other commits

```
$ git rebase HEAD master
```

```
A--B'--C--D--E(master)
```

>

Patches

□ standard diff

```
$ diff -u a b
--- a      2007-03-04 22:40:11.473402253 -0500
+++ b      2007-03-04 22:40:25.365464311 -0500
@@ -1 +1 @@
-old text
+new text
```

□ git diff

```
$ git diff HEAD~1
diff --git a/file b/file
index acc4bcb..eee417f 100644
--- a/file
+++ b/file
@@ -1 +1 @@
-old text
+new text
```

Patches : Generating

□ just diffs

```
$ git diff -1 > last.patch  
$ git diff tag~1..tag > before-tag.patch  
$ git diff ^good > since-good.patch
```

□ git patches

○ file per commit

```
$ git format-patch -2  
00001-this-is-my-first-patch.txt  
00002-this-is-my-most-recent-patch.txt
```

○ an mbox file

```
$ git format-patch a..b --stdout -k > patches
```

○ send email

```
$ git send-email --to bart@jukie.net patches
```

>

Patches : Accepting

□ apply patch

```
$ git apply changes-from-bob.patch
$ git add new-files...
$ git rm old-files...
$ git commit -a \
    -m"fixes soemthing" \
    --author="Bob <bob@gmail.com>"
```

□ apply and commit a git patch

```
$ git am -3 -s -k 00001-first-patch.txt
$ git am -3 -s -k patches.mbox
```

>

Decentralized : Cloning

□ cloning local repository

```
$ git clone /var/git/our-repo.git my-work  
$ git clone ~bob/export.git working-with-bob  
$ git clone -l -s my-work my-work-2
```

□ cloning remote repository

```
$ git clone ssh-server:~bart/something my-work  
$ git clone http://web-server/git-repo my-work  
$ git clone git://git-server/git-repo my-work
```

□ with reference

```
$ git clone --reference existing-full-clone \  
    git://git-server/git-repo quick-test
```

>

Decentralized : Fetch / Pull

- get updates from server

 - \$ git fetch origin

 - \$ git merge origin/master

- ... or ...

 - \$ git pull origin

- 'origin' is the default remote

 - \$ git pull

- NOTE: the merge only happens to tracking branches

>

Decentralized : Tracking branches

□ when it matters

```
$ git checkout some-branch
```

```
$ git pull remote
```

if 'some-branch' tracks a remote branch, then
git-pull implicitly calls git-merge

□ creating tracking branches

```
$ git checkout --track new-branch remote/branch
```

... or ...

```
$ git branch --track new-branch remote/branch
```

>

Decentralized : Tracking branches (2)

□ adding them manually

```
$ git config branch.mywork.remote myremote
```

```
$ git config branch.mywork.merge refs/heads/master
```

when current branch is 'mywork', a git-pull from 'myremote' will merge the remote 'master' branch into 'mywork'

```
$ git checkout mywork
```

```
$ git pull myremote
```

□ setting the default

```
$ git config --global branch.autosetupmerge true
```

any new branch off a remote will automatically track it
default is off

>

Decentralized : Push

- publish updates to 'origin'

```
$ git push
```

- push to some other destination

```
$ git push some-remote
```

- push to an untracked location

```
$ git push some-remote local-branch:remote-branch
```

>

Decentralized : Daemon

- git supports

- /local

- ssh://

- rsync://

- http://

- git://

- run git:// server on 9418

- ```
$ git daemon /src/software.git
```

- from inetd

- see: `man git-daemon`

>

# Decentralized : Remotes

---

## □ list remotes

```
$ git remote
origin
```

## □ new remotes

```
$ git remote add bob git://server/bob.git/
```

## □ update tracked branches

```
$ git remote update
$ git branch -r
bob/one
bob/two
```

>

# Tools : Finding regressions

---

- found a regression

- HEAD is bad
- known last good point
- want to find first bad commit

- binary search

- \$ git bisect start
- \$ git bisect bad
- \$ git bisect good my-last-release
- finds commit to test next
- test and tell it 'good' or 'bad'
- repeat

>

# Tools : Working with other SCMs

---

- can interoperate with

- SVN
- CVS
- arch
- darcs
- p4
- hg
- bk

... others

- tailor

- Swiss Army knife
- anything <---> anything

>

# Tools : Working with SVN

---

```
$ git-svn clone svn+ssh://host/project/trunk project.git
$ cd project.git
 ... work work work ...
$ git commit
 ...
$ git-svn fetch
$ git-svn rebase
$ git-svn dcommit
```

>

# Tools : Housekeeping

---

- repository cleanup

```
$ git prune -d
```

```
$ git fsck
```

... or ...

```
$ git gc
```

- lost commits

```
$ git lost-found
```

>

# Tools : Visualization

---

- commit tools

  - `$ git gui`

  - `$ gct`

- mergeing

  - `$ git mergetool`

- revision tools

  - `$ gitk`

  - `$ qgit`

  - `$ gitview`

>

# Tools : Build farms

---

- create a snapshot of a revision

```
$ git archive --format=tar \
 --remote=git://server/repo.git \
 --prefix=build-dir \
 tag-to-build | tar xf -
$ cd build-dir
$ make release
```

>

# Tools : gitweb

---

## □ use bare repositories

```
$ ssh www
$ mkdir /home/git/project.git
$ cd /home/git/project.git
$ git --bare init
```

## □ setup an additional remote to push to

```
$ cd my-work.git
$ git remote add www server:/www/git/project.git
$ git config remote.www.push master:refs/heads/master
$ git config branch.master.remote www
$ git config branch.master.merge refs/heads/master
$ git push server
```

## □ you cannot clone from gitweb URLs

```
$ edit project.git/description
... add the git:// url to it
```

>

# Tools : misc

---

- cogito
- bash and zsh completion
- quilt like functionality
  - STGIT
  - guilt

>

# References

---

- Getting started

- <http://git.or.cz/>

- Download

- <http://www.kernel.org/pub/software/scm/git/>

- GIT release notes

- 1.5.0 - <http://lwn.net/Articles/222086/>

- 1.5.1 - <http://lwn.net/Articles/229669/>

- 1.5.2 - <http://lwn.net/Articles/235109/>

# Questions

---

