Git for bzr users

An attempt at making you comfortable when you have to work with a git repository

October 2010 - Aurélien Gâteau <aurelien.gateau@canonical.com>

First time setup

Tell git about you:

```
git config --global user.name “Aurélien Gâteau”

git config --global user.email \
  aurelien.gateau@canonical.com
```

Getting a working copy

git clone <repository-url>

```
git clone git://github.com/agateau/yokadi.git
Initialized empty Git repository in
/home/aurelien/tmp/yokadi/.git/
remote: Counting objects: 3217, done.
remote: Compressing objects: 100% (1121/1121), done.
remote: Total 3217 (delta 2078), reused 3196 (delta 2066)
Receiving objects: 100% (3217/3217), 495.00 KiB | 121
KiB/s, done.
Resolving deltas: 100% (2078/2078), done
```

Inspecting history

git log

```
commit 7ddc7778571965cae842149672b01f4c56052713
Author: Séb... R... <Seb...@d...x.org>
Date:   Tue Jul 6 14:15:39 2010 +0200

    Add n_list simple test case

commit f0cea777729e12720d127cf0557163e52a4048c4
Author: Séb... R... <Seb...@d...x.org>
Date:   Tue Jul 6 14:11:38 2010 +0200

    Fix n_list crash

commit fc28ba71235b35c7676a431665f839654157745
Author: Séb... R... <Seb...@d...x.org>
Date:   Tue Jul 6 14:03:01 2010 +0200

    Add decryption feature to t_list/n_list and textlist
renderer
```

Specifying revisions

- By their SHA1 (no revision numbers)
  Can be truncated as long as it is not ambiguous
- By tag or branch name
- HEAD is the latest commit (= -r-1 in bzr)
- You can append suffixes:
  - ^: parent commit (HEAD^ = -r-2 in bzr)
  - ~N: N-th parent of commit (HEAD~4 = -r-5 in bzr)

man git-rev-parse for more creative way to specify revisions

Displaying diffs

- git diff <commit-range> (= bzr diff -r<range>)
- git show <commit> (= bzr diff -c<revno>, but show message and diff)

```
git diff f0cea77..7ddc777
diff --git a/src/yokadi/tests/tasktestcase.py
b/src/yokadi/tests/tasktestcase.py
index 39c98b7..237eaa7 100644
--- a/src/yokadi/tests/tasktestcase.py
+++ b/src/yokadi/tests/tasktestcase.py
@@ -141,6 +141,18 @@ class TaskTestCase(unittest.TestCase):
                         "@%", "@k%", "!@%", "!@kw1", "-f plain",
"-f xml", "-f html", "-f csv"):
             self.cmd.do_t_list(line)

+    def testNlist(self):
+        tui.addInputAnswers("y")
(...)
```

Editing

git status is the equivalent of bzr status

```
vi NEWS

git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be
committed)
#   (use "git checkout -- <file>..." to discard changes in
working directory)
#
#       modified:   NEWS
#
no changes added to commit (use "git add" and/or "git commit
-a")
```

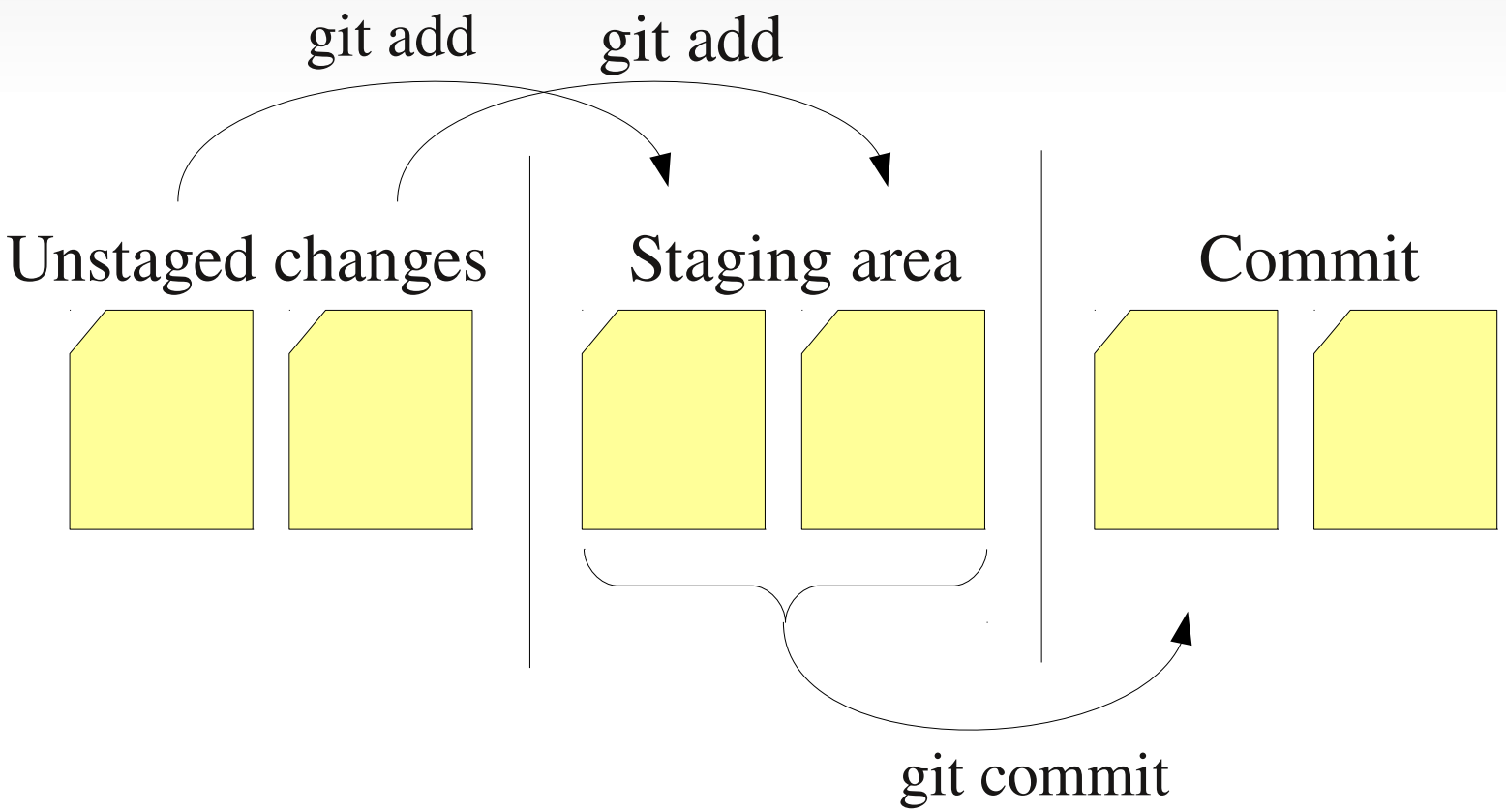
Editing

Even existing files must be added before they are committed:

```
git commit
no changes added to commit (use "git add" and/or "git commit
-a")

git add NEWS

git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   NEWS

git commit -m 'Added latest news'
[master 69b0d3a] Added latest news
 1 files changed, 0 insertions(+), 1 deletions(-)
```

The staging area

- Aka “the index” or “the cache”
- A place to construct your commits



Local branches

- A checkout in bzr is a branch, whereas a checkout in git is a repository (collection of branches)
- Default branch is called “master”
- Main branch commands:
  - git branch: lists available branches
  - git checkout <name>: switches to branch <name>
  - git checkout -b <name>: creates branch <name> and switches to it

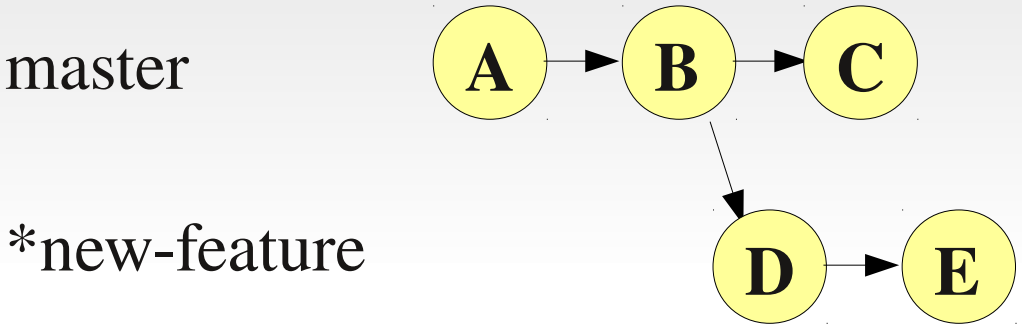
Local branches

```
git branch
* master

git checkout -b new-feature
Switched to a new branch 'new-feature'

git branch
  master
* new-feature

git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
```

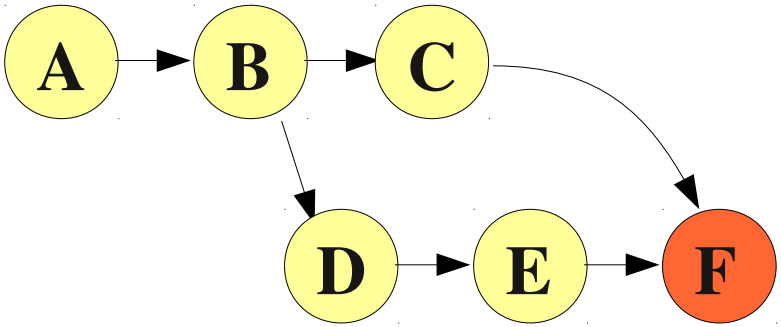
Note the message from "git checkout master". This is because we have a change which has not been pushed yet.

Merging

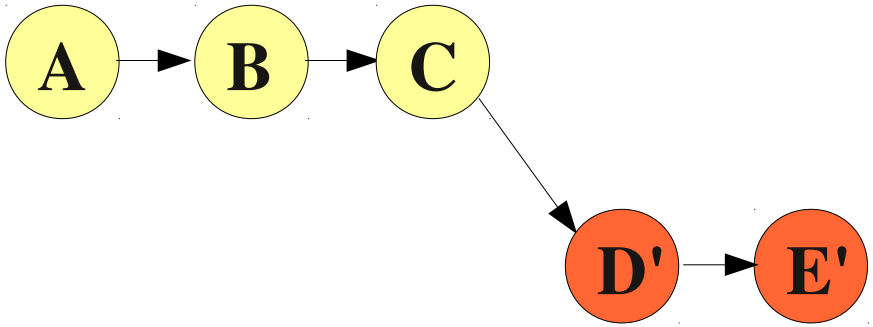
Two ways to merge: merge and rebase



Merging (git merge master)



Rebasing (git rebase master)



Do not rebase published commits!

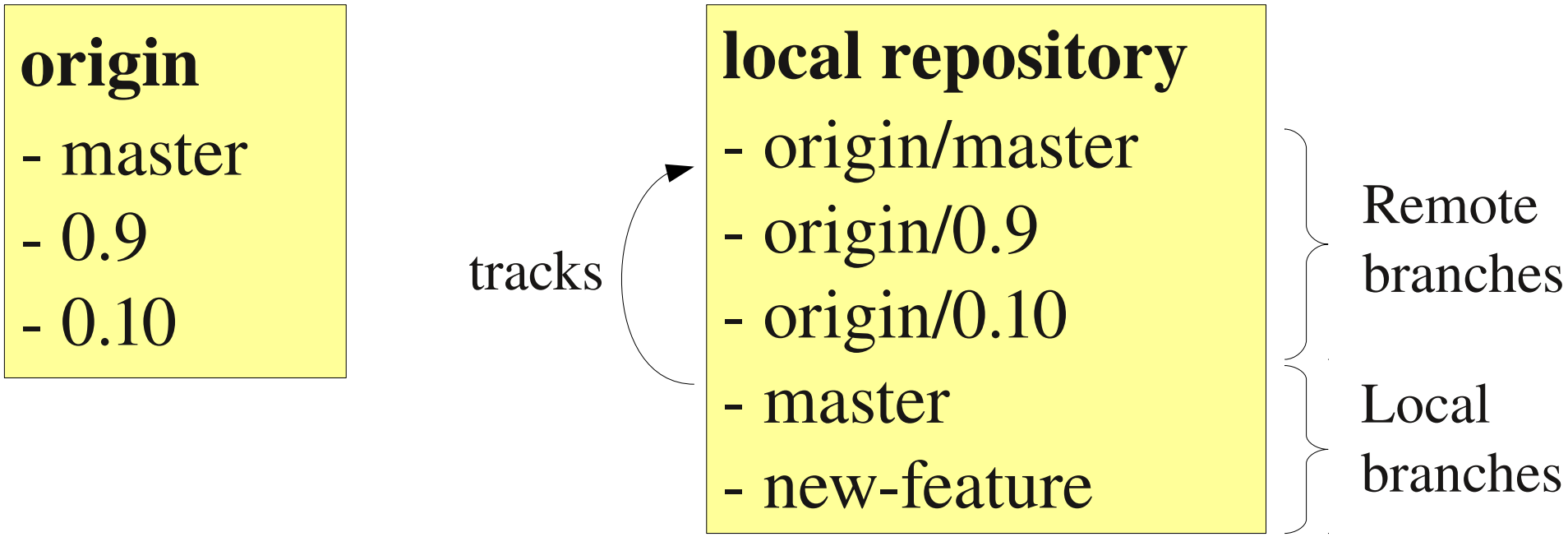
Remotes

- A remote is a “bookmark” to a distant repository
- Default remote is “origin”, set when repository was cloned
- Use “git remote -v” to list remotes with the urls they point to (two urls per remote):

```
git remote -v
origin  git://github.com/agateau/yokadi.git (fetch)
origin  git://github.com/agateau/yokadi.git (push)
```

Remote branches

- “git clone” does two things:
  - clones all branches as “remote” branches
  - creates a local branch named “master”, tracking “origin/master”



Remote branches

git branch -a: lists all branches (local and remote)

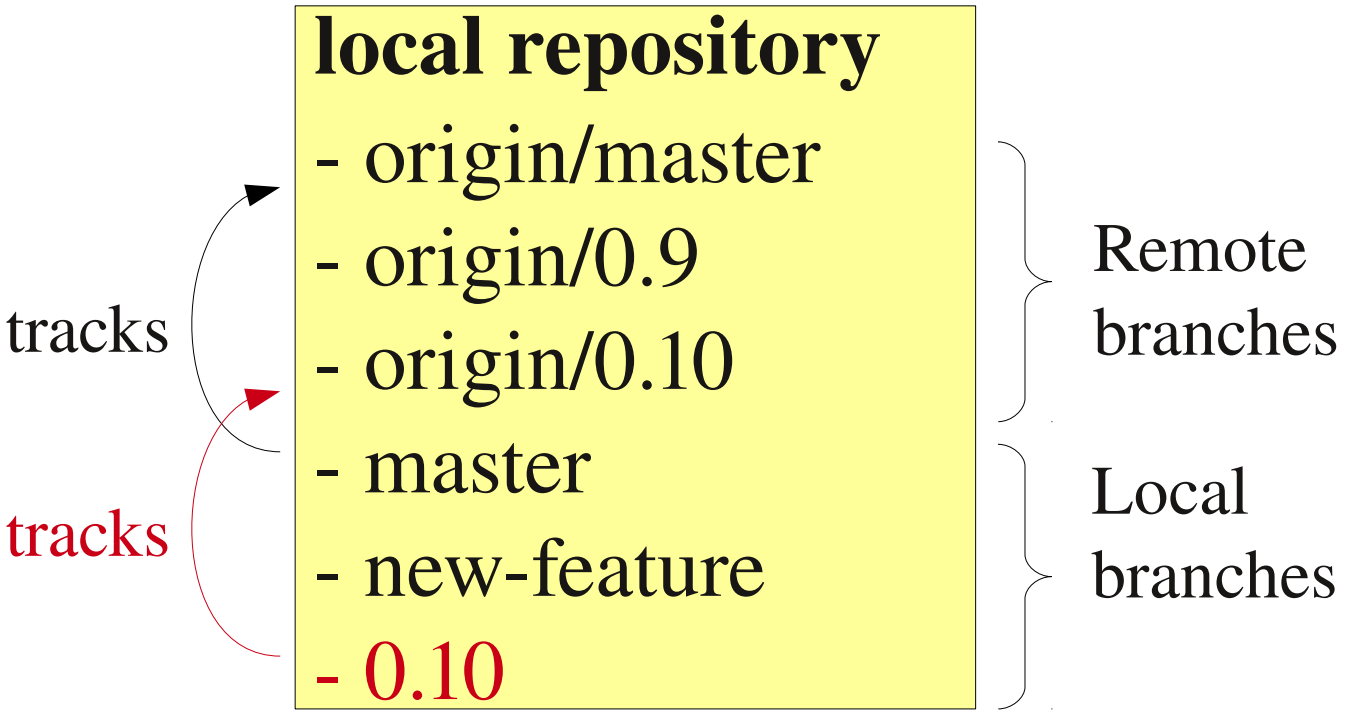
```
git branch -a
  master
* new-feature
  remotes/origin/0.10
  remotes/origin/0.9
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

remotes/origin/HEAD is a shortcut which makes it possible to refer to origin/master as origin

Working on a remote branch

You need to create a tracking local branch

```
git checkout -t origin/0.10
Branch 0.10 set up to track remote branch 0.10 from origin.
Switched to a new branch '0.10'
```



Updating from a repository

- Simple version: git pull or git pull --rebase
- What happens under the hood:
  - git fetch: updates the remote branches from the remote repository
  - git merge: updates the remote-tracking local branches from the remote branches

Git pull will stop if your working copy contains uncommitted changes

Publishing changes

- Simplistic version: git format-patch <range>
  - Generates one .patch file per commit for <range>, named according to the first line of the commit message
- Can then be applied with git am *.patch

```
git format-patch origin/master
0001-Updated.patch
0002-Fixed-the-editor.patch
```

Publishing changes

- To push changes to another repository, use git push
- git push without any argument: push all changes from tracking branches to default repository (origin)
- git push <repository> <branch-name>: push changes from <branch-name> to <repository>
  - example: “git push origin master” push all changes from master to the master branch in the origin repository

Tools

- Lots of tools exist around git
  - tig: command line tool to browse history
  - gitk: tcl/tk tool to browse history
  - git gui: tcl/tk tool to create commits
  - giggle: gtk+ tool to browse history
  - git grep: greps through whole repository, faster than grep
  - Show current git branch in your prompt (search for “git branch in prompt” on the internet for several examples)

Configuration

- Options can be set per repository (.git/config) or per account (~/.gitconfig)
- Git uses .ini style files
- Files can be edited by hand or with git config:
  - git config <key> <value> edits .git/config
  - git config --global <key> <value> edits ~/.gitconfig
  - if <value> is empty, it prints the current value

Useful configuration setup

- Paste this in your ~/.gitconfig

```
[color]
branch = auto
diff = auto
status = auto

[alias]
pr = pull --rebase
st = status
ci = commit
log1 = log --pretty=oneline --decorate
```

The end

You are now git savvy!

(hopefully)

Questions?