Using git to develop a PostgreSQL patch

pgcon.br 2009
Campinas, Brazil

Magnus Hagander
Redpill Linpro AB

PostgreSQL development

- As you know...
- Master repository CVS
- Limited group of committers
- Patch-on-list based

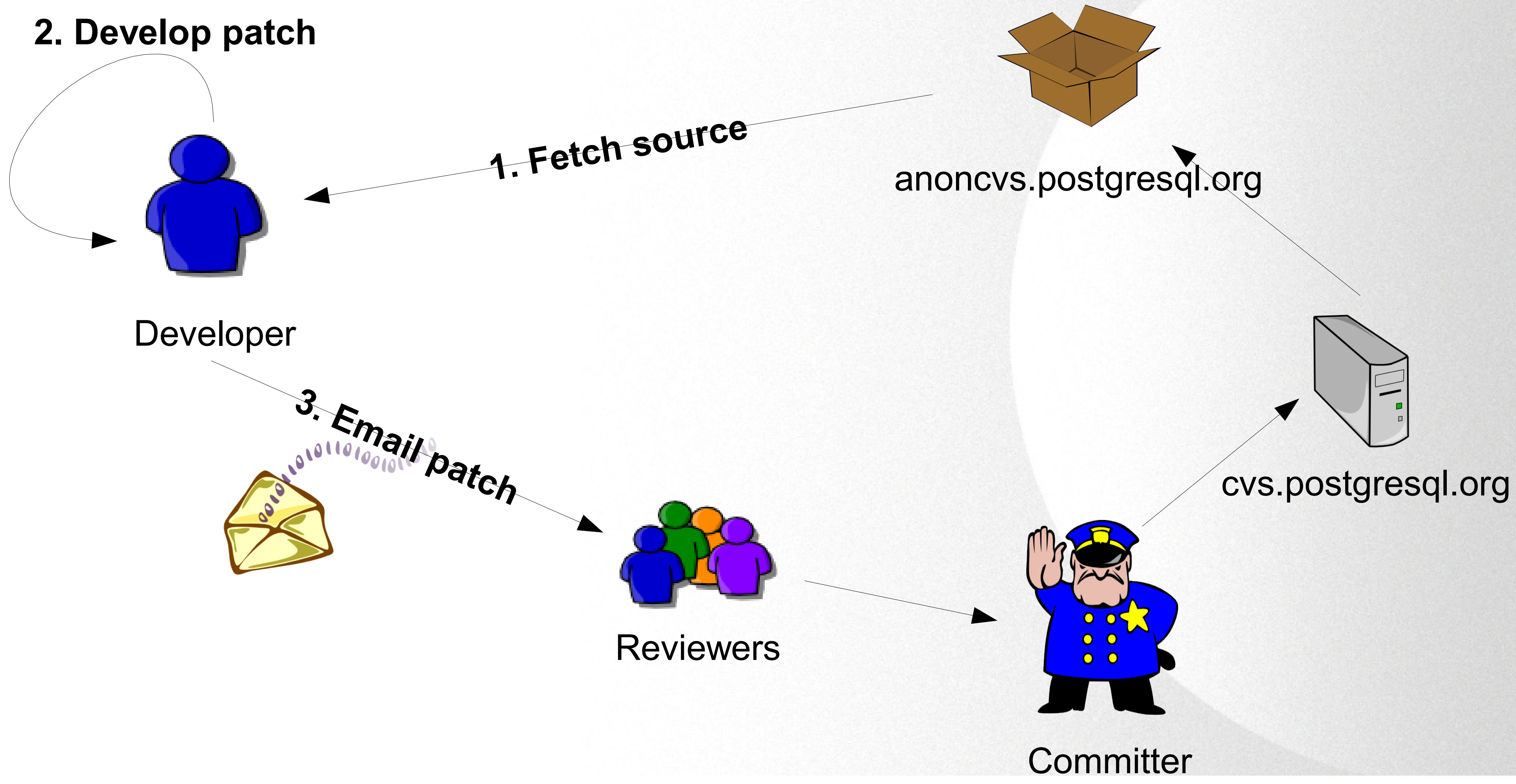
GIT development

- As you may know…
- *Distributed* Version Control
- Each his own master
- Easy branching, easy merging
- push/pull

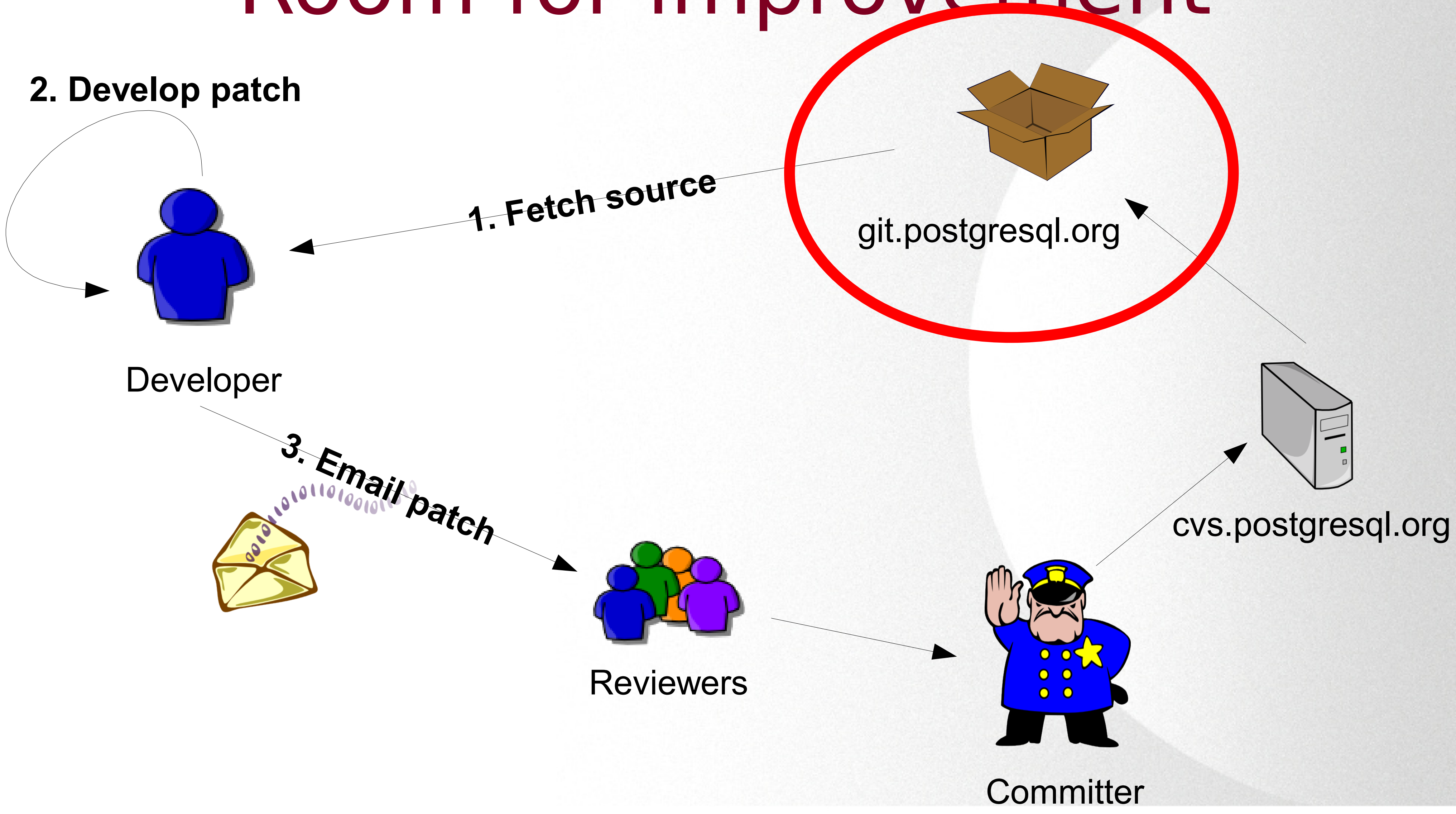
No overlap?!

- Anonymous git mirror
  - git.postgresql.org
- Community experience

«Old» development process



Room for improvement



Obvious advantages

- Offline access
- Full history access
- Much much faster

Easy: differences are small

cvs checkout...	git clone...
cvs update	git pull
cvs diff	git diff
cvs log	git log
cvs annotate	git blame

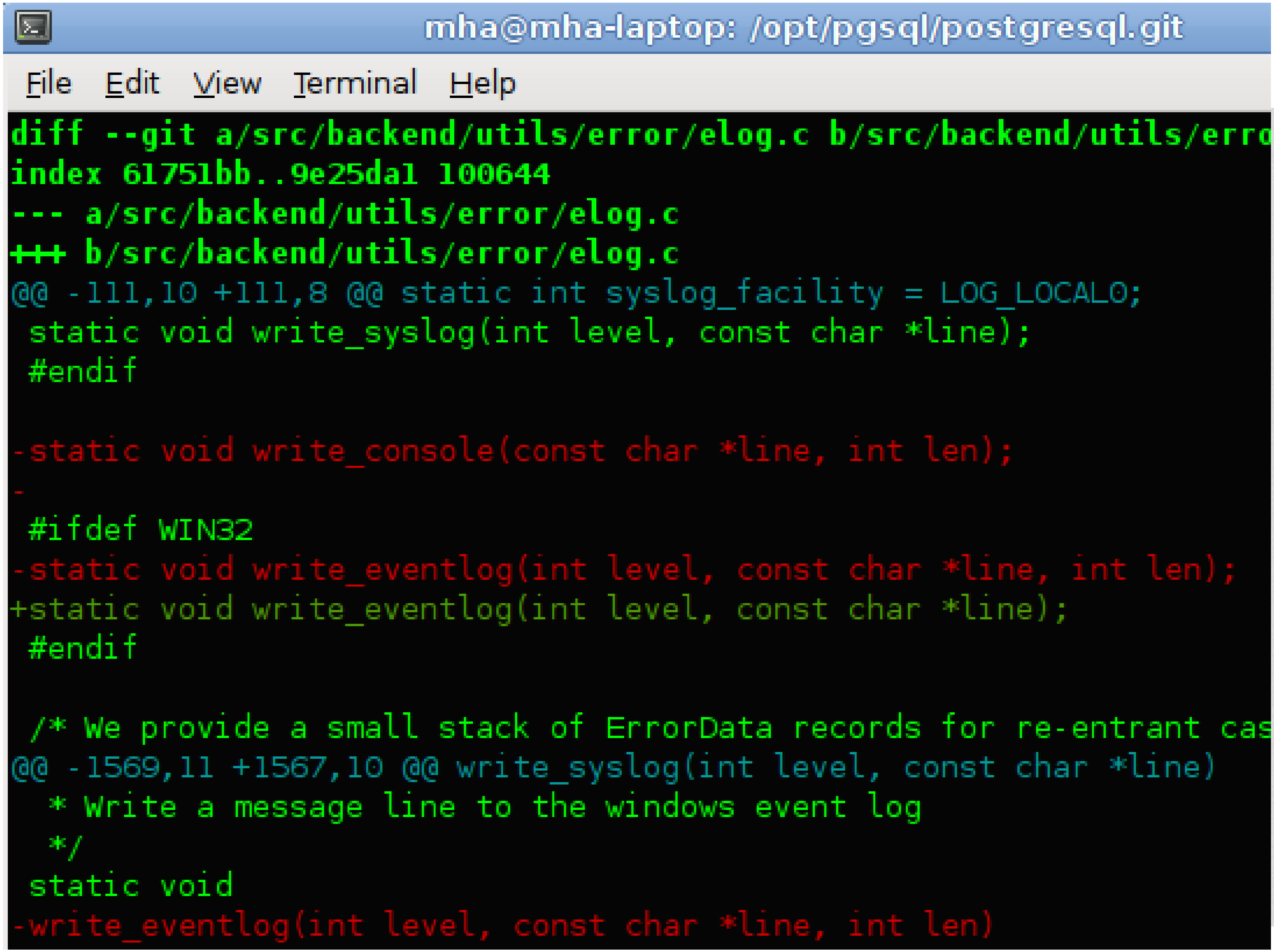
DEMO TIME!

PostgreSQL prefers diff -c

- Git native diff does not produce
- Use filterdiff:

```
git diff | filterdiff -format=context
```

- Alias!
- No more highlighting...

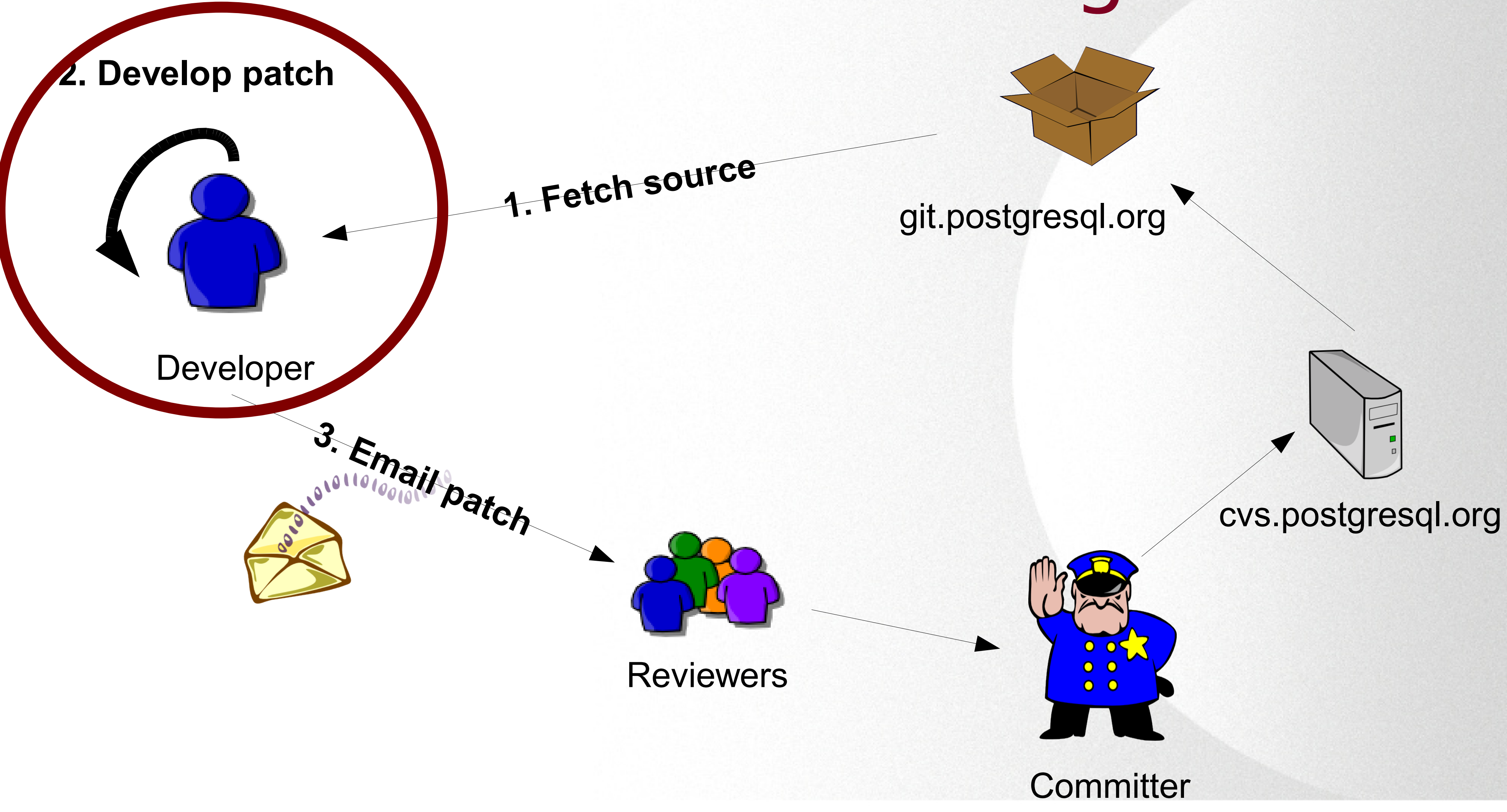

mha@mha-laptop: /opt/pgsql/postgresql.git
File Edit View Terminal Help

```
diff --git a/src/backend/utils/error/elog.c b/src/backend/utils/erro
index 61751bb..9e25da1 100644
--- a/src/backend/utils/error/elog.c
+++ b/src/backend/utils/error/elog.c
@@ -111,10 +111,8 @@ static int syslog_facility = LOG_LOCAL0;
 static void write_syslog(int level, const char *line);
 #endif

-static void write_console(const char *line, int len);
-
 #ifdef WIN32
-static void write_eventlog(int level, const char *line, int len);
+static void write_eventlog(int level, const char *line);
 #endif

 /* We provide a small stack of ErrorData records for re-entrant cas
@@ -1569,11 +1567,10 @@ write_syslog(int level, const char *line)
  * Write a message line to the windows event log
  */
 static void
-write_eventlog(int level, const char *line, int len)
```

The real advantage



Git: Feature branches

- Branch creation is *easy*
- Branch creation is *cheap*
- Branch creation is *fast*
- Conclusion: *create lots of branches*

One branch for each feature

- Commit to local branch
  - Nobody will see it!
- Commit often!
  - Incremental development!
  - Rollback your mistakes
  - Examine incremental changes

DEMO TIME!

One branch for each feature

- Still send a patch to the list, just like before
- PostgreSQL does ***not*** like the one-patch-per-commit format
  - Other git projects do!
  - Notably the Linux kernel

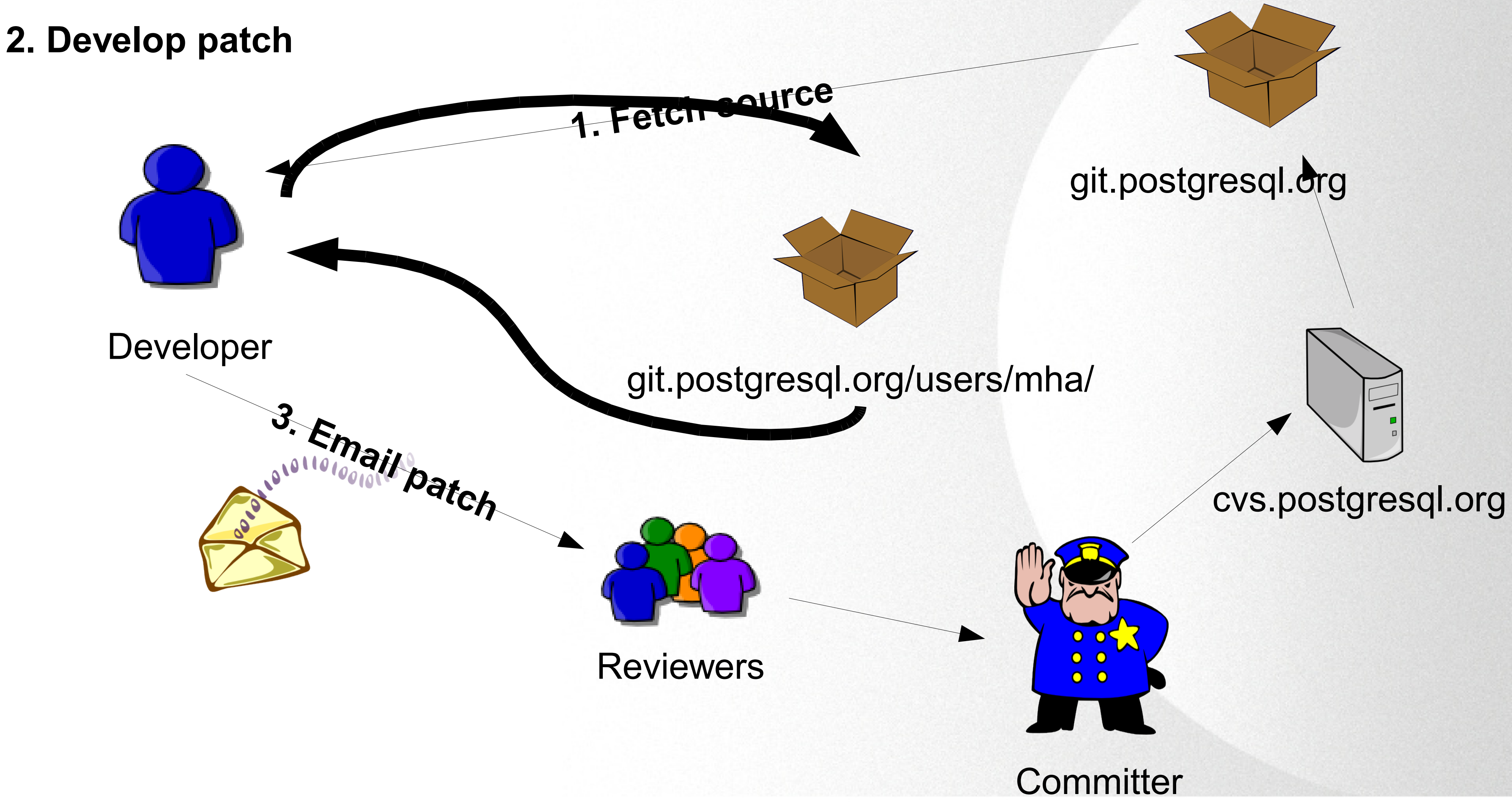
Merge or rebase

- Upstream changes during development
- Maybe different files, maybe same
- Update *often* to avoid conflicts

Merge or rebase

- «git pull» will do:
  - fetch
  - merge
- «git pull –rebase» will do:
  - store all changes
  - fetch
  - update
  - re-apply your changes

Next step: sharing branches



Sharing your branches

- Set up a repository on *git.postgresql.org*
- *Push* your branch
- Others can *pull* your branch
- Suddely, you're sharing!

Not sharing your mistakes?

- Once pushed, you can never remove it
    - Well, you *should* never...
- What about all those tiny commits?

Rebase to the rescue

- Use

```
git rebase origin/master -interactive
```

- Squash commits into single ones
- Edit commit messages
- *ONLY* before you push!

DEMO TIME!

Using git for testing

- A good way to get cross platform
- For example, testing on Windows
  - Consider Amazon EC2, see

    `http://blog.hagander.net/archives/151-Testing-PostgreSQL-patches-on-Windows-using-Amazon-EC2.html`

Redpill Linpro

Using git to develop a PostgreSQL patch

Questions?

magnus@hagander.net
Twitter: @magnushagander
http://blog.hagander.net