

TUTORIAL GIT

the fast version control system

IONUT NICU

developer LiSA Project

BLUG · OS · CON

we're open

What is Git?

- **Distributed** version control system (VCS) or source code management system (SCM)
- **Efficient** handling of very large projects.
- Strong support for **non-linear development**.
Easy branching and merging
- **Cryptographic authentication** of history
- **Toolkit design**
- Protocols: git, rsync, http, git over ssh

Git Facts

- Original author: Linus Torvalds
- Current maintainer: Junio Hamano
- License: GPL
- Written in: C, Bash, Perl
- Portability:
 - POSIX based systems: Linux, BSD, Solaris, Darwin.
 - Windows:
 - Cygwin
 - Native port: msysgit

Projects using git

- Git
- Linux Kernel
- Perl
- Gnome
- Qt
- Ruby on Rails
- X.org
- Fedora
- Debian
- LiSA ;-)

First step – global config

- Global config file: `$HOME/.gitconfig`

- Sign-off information:

```
git config –global user.name 'Ionut Nicu'
```

```
git config –global user.email 'ionut.nicu@mindbit.ro'
```

- Configure global settings:

```
git config –global color.diff auto
```

```
git config –global color.status auto
```

```
git config –global color.branch auto
```

```
git config –global core.autocrlf input
```

Git concepts (1)

- **Remotes:** git remote show, git remote show origin, git remote add -f git_url:
 - origin – remote repository
 - origin/master – remote branch
 - git remote [add|rm|update|show|prune]
- **Branches:** local/remote
 - show: git branch (local), git branch -r (remote), git branch -a (all).
 - move head: git checkout <existing_local_branch>
 - create new branch:
 - git checkout -b <local_branch> <commit>
 - git checkout -b <local_branch> <remote>/<remote_branch>
 - **master** branch: like CVS mainline, just a convention, nothing special about it.

Git concepts (2)

- **Commits:** uniquely identified by SHA1 id (20 bytes). Visualize history on a branch: `git log [file-name]`, `gitk --all`:
 - Create a commit: change / `git add` (stage changes to index) / `git commit [-s] [-m]`. Log messages should be properly formatted for e-mail submission.
 - HEAD, HEAD~1 (HEAD^), HEAD~2 (HEAD^^), HEAD~5 (5 commits before HEAD): `git log HEAD~5..HEAD^^`
- **Tags:** CVS style tags, annotated tags:
 - human friendly name for a commit
 - `git tag -a` (annotated) `-s` (GPG signed)
 - `git tag -l`, `git tag -d`
 - `git describe`: *lisa_2.6.21-174-gf461349*
- **Index:** staging area for commits (preparing a commit):
 - Files in repository: added to index, locally modified, untracked
 - `git add`, `git reset --mixed`, `--hard` (be careful with this!)

Cloning or creating a repository

- Work on a public project:

```
git clone git://lisa.mindbit.ro/lisa.git
```

```
cd lisa
```

```
vim Makefile
```

```
git add Makefile
```

```
git commit -s
```

```
git format-patch origin/master
```

```
git send-email -to lisa-devel@lisa.mindbit.ro 0001-My-First-Patch
```

- Create your own project:

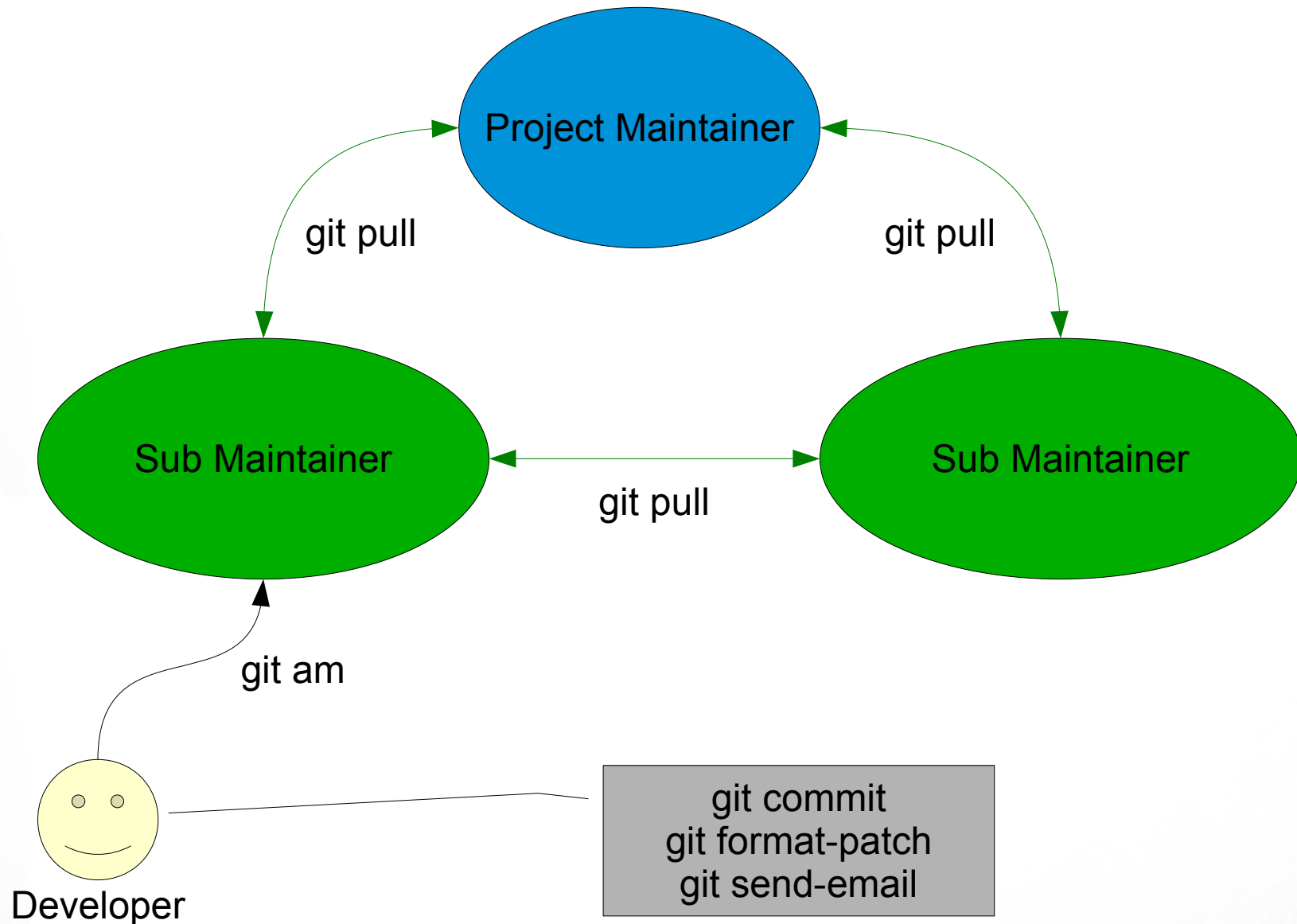
```
cd hello
```

```
git init .
```

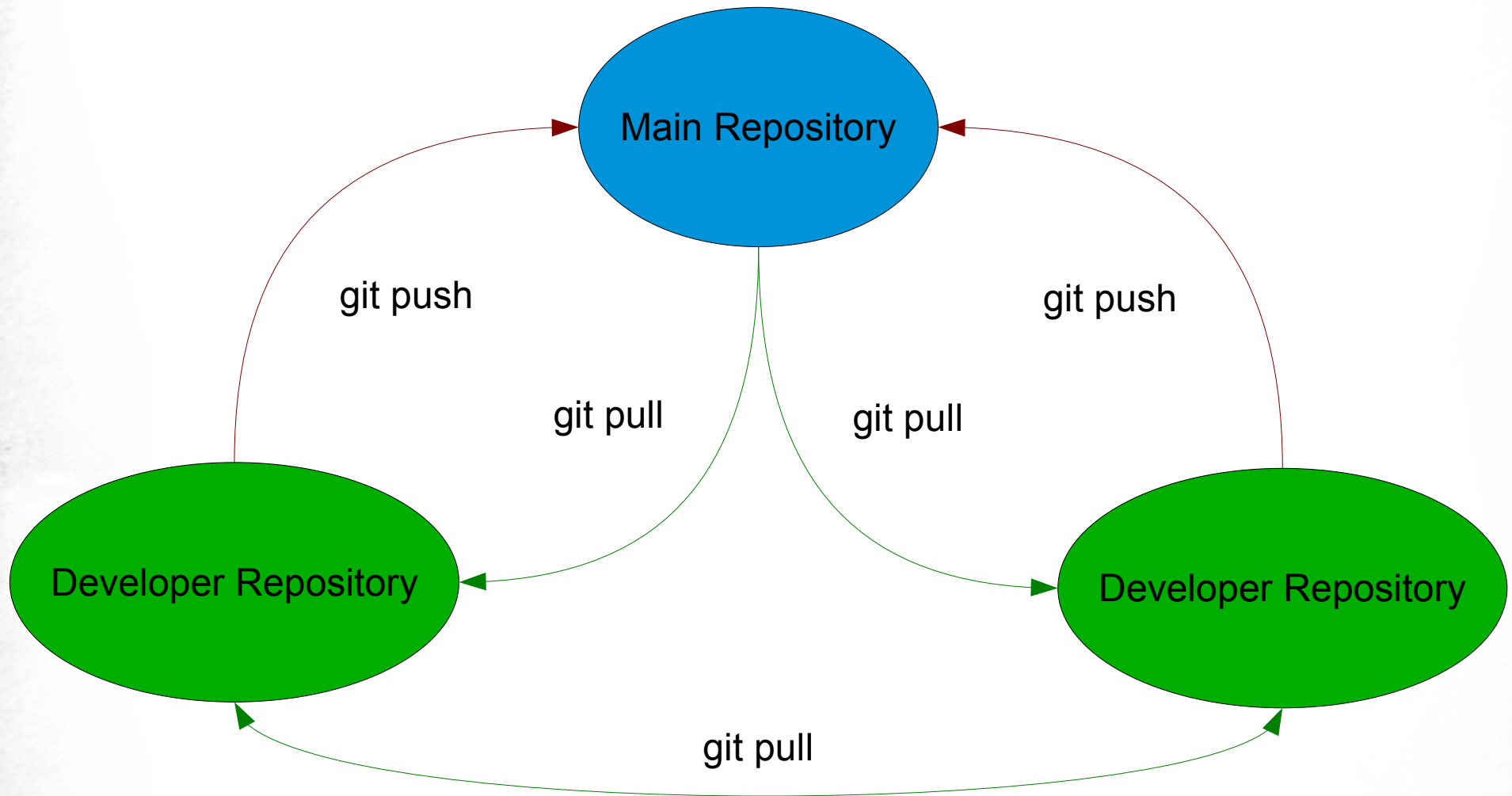
```
git add *.c *.h
```

```
git commit -s
```

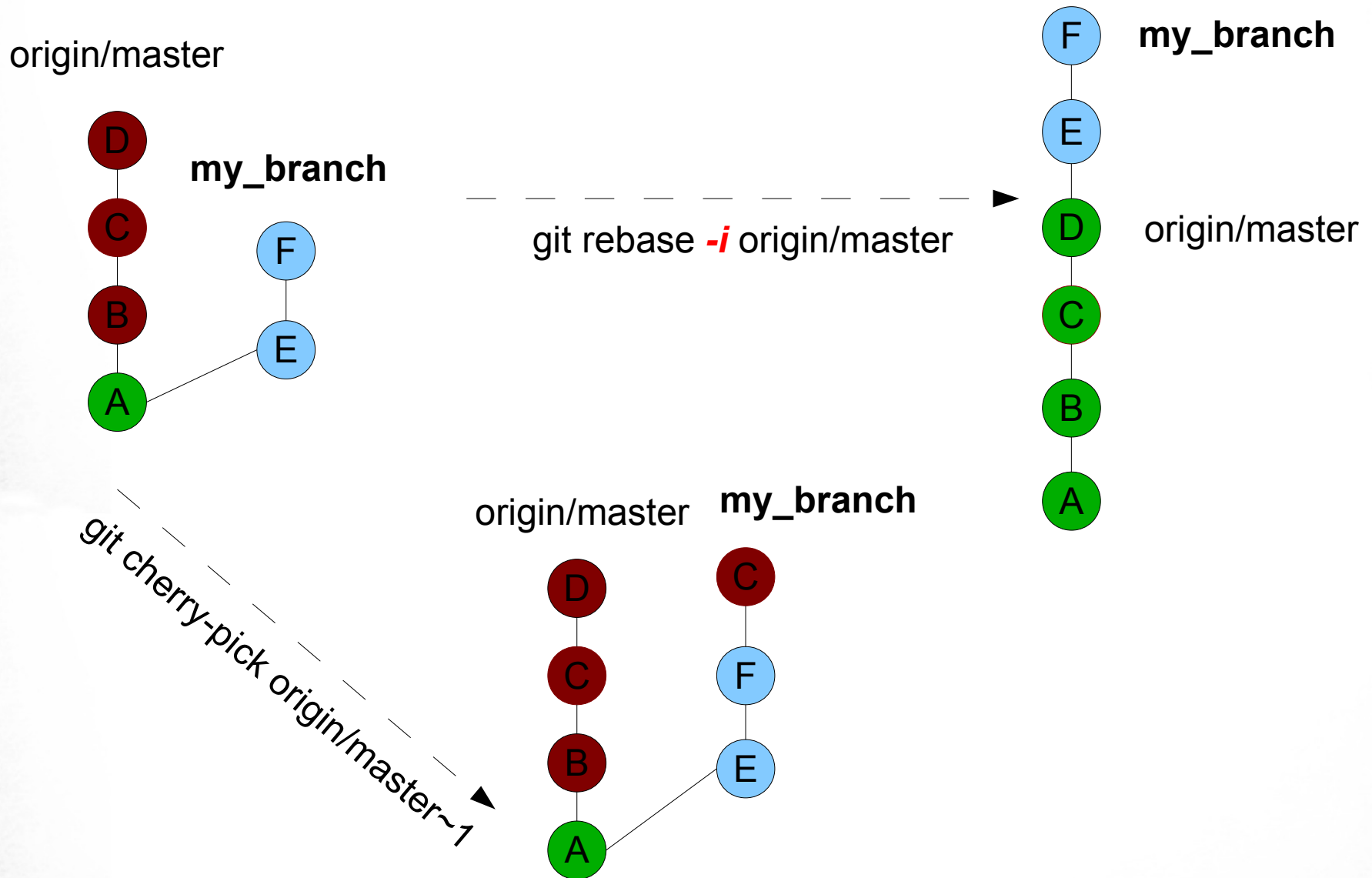
Development models (1)



Development models (2)



git rebase / git cherry-pick



Git goodies

•Stashes

- WIP on dev branch, urgent patch required on stable (release) branch
- git stash, git stash show, git stash apply <stash>, git stash clear

•Fixing regressions

- git bisect start
- git bisect bad (current version is bad)
- git bisect good v2.1
 - *Bisecting: 675 revisions left to test after this*
 - git bisect visualize: view suspects in gitk
- Compile/test: if test fails, then git bisect bad, else git bisect good
- git bisect run:
 - git bisect start HEAD v2.1 – (HEAD is bad, v2.1 is good).
 - git bisect run make test – (make test builds and test).

?