РЕД СЛЁРМ

+

Southbridge

Системы контроля версий. Git и Gitlab

slurm.io

Цели и задачи:

-Основные навыки работы в git
-Работа в команде
-Организация процесса разработки

Системы контроля версий

- Локальные rcs
- Централизованные CVS, Subversion, Perforce
- Децентрализованные GIT, Mercurial, Bazaar, Darcs

Преимущества GIT

- скорость
- простая архитектура
- хорошая поддержка нелинейной разработки
- полная децентрализация
- возможность эффективного управления большими проектами

Состояния файлов в GIT

- Commited or Unmodified (Зафиксированный)
- Modified (Измененный)
- Staged (Подготовленный)

Устанавливаем GIT

Пакет-версия	Репозиторий	Адрес репозитория
git-1.8.3	@base	
git-2.12.2	Southbridge-stable	http://rpms.southbridge.ru/southbridge-rhel7-stable.rpm
git2u-2.16.4	IUS.io	https://centos7.iuscommunity.org/ius-release.rpm

Первые шаги

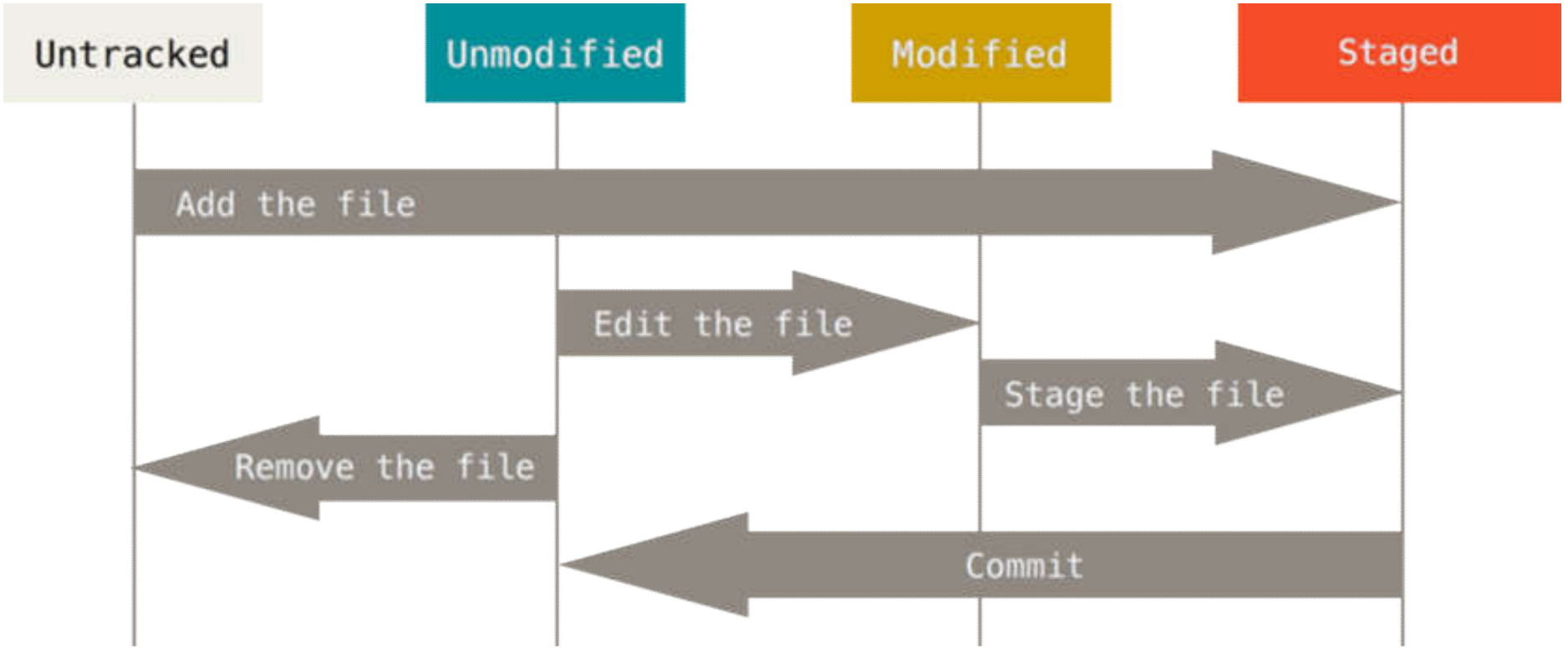
Настройка:

```
git config --global user.name "Your Name"
git config --global user.email "you@yournet.kz"
git config --global http.postBuffer 524288000
```

Создание репозитория:

```
git init
git add Readme.md
git add *.py
git add .
git commit –m “init”
```

Жизненный цикл файлов



Уверенно идем вперед

Просмотр статуса файлов
git status

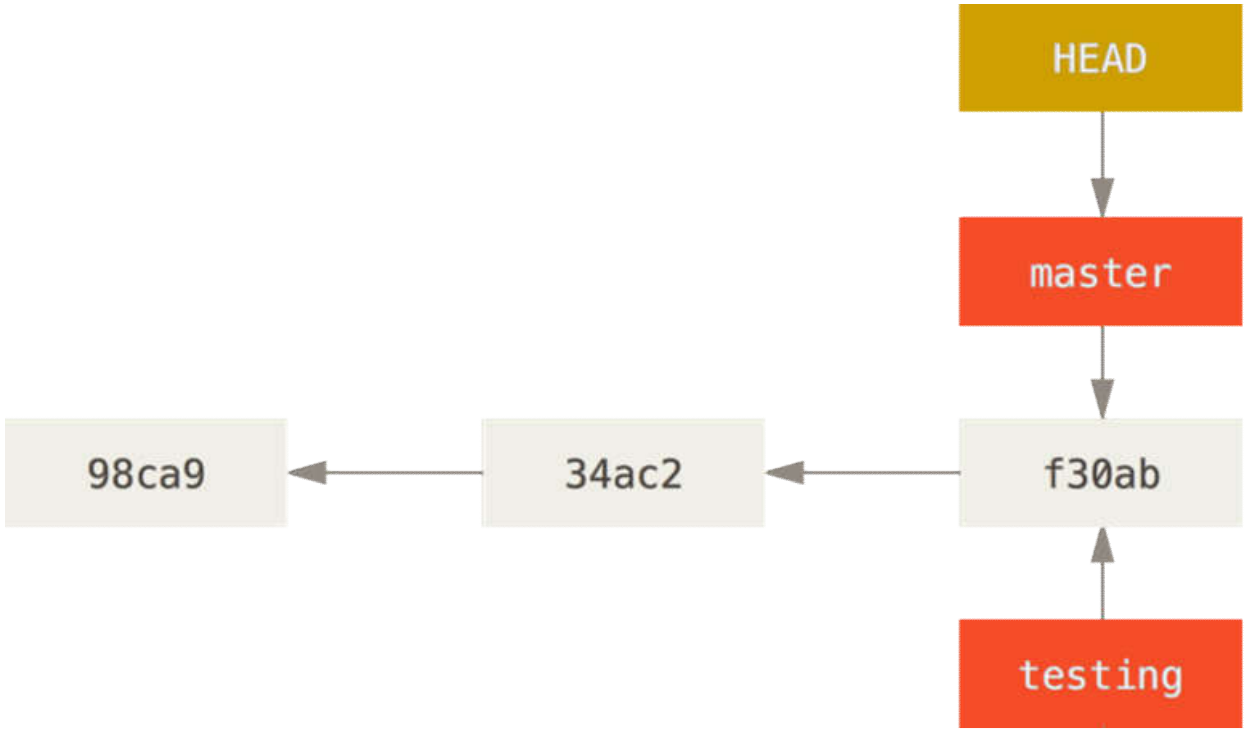
Коммит всех измененных файлов
git commit –a

Просмотр истории коммитов
git log
git log –p

Переключение на коммиты из истории
git checkout <hash commit>

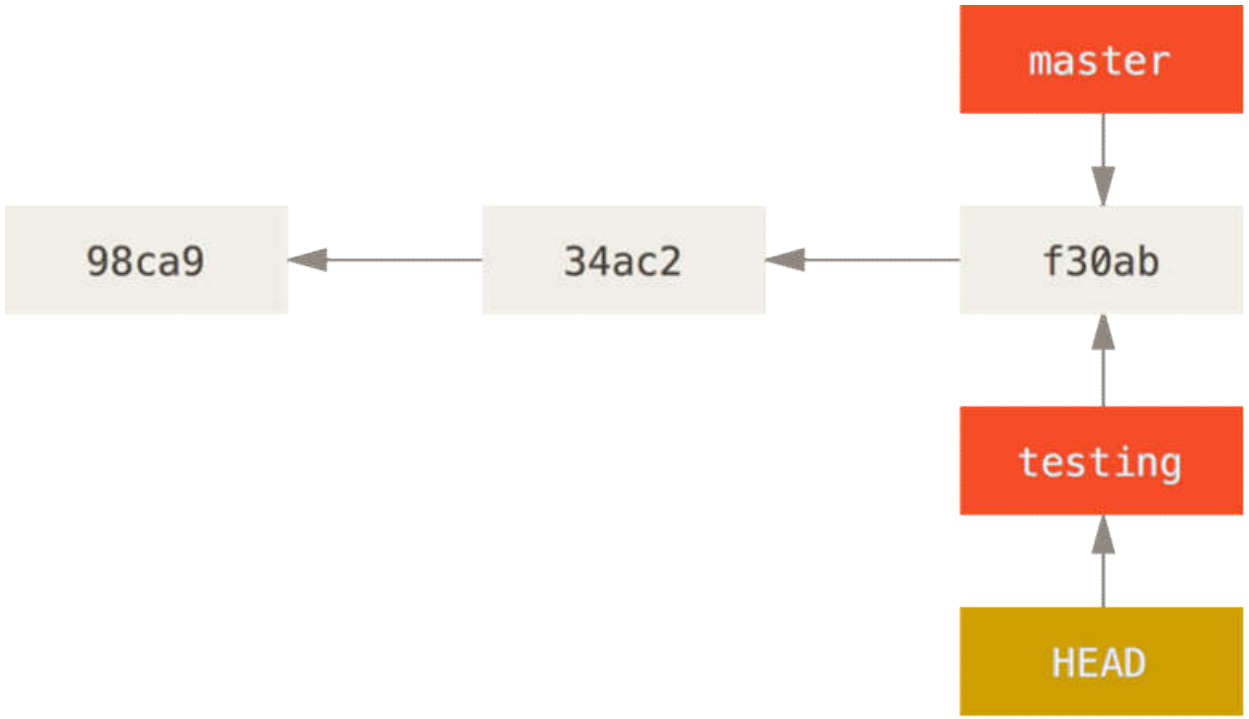
Создание ветки

git branch testing



Переход на ветку

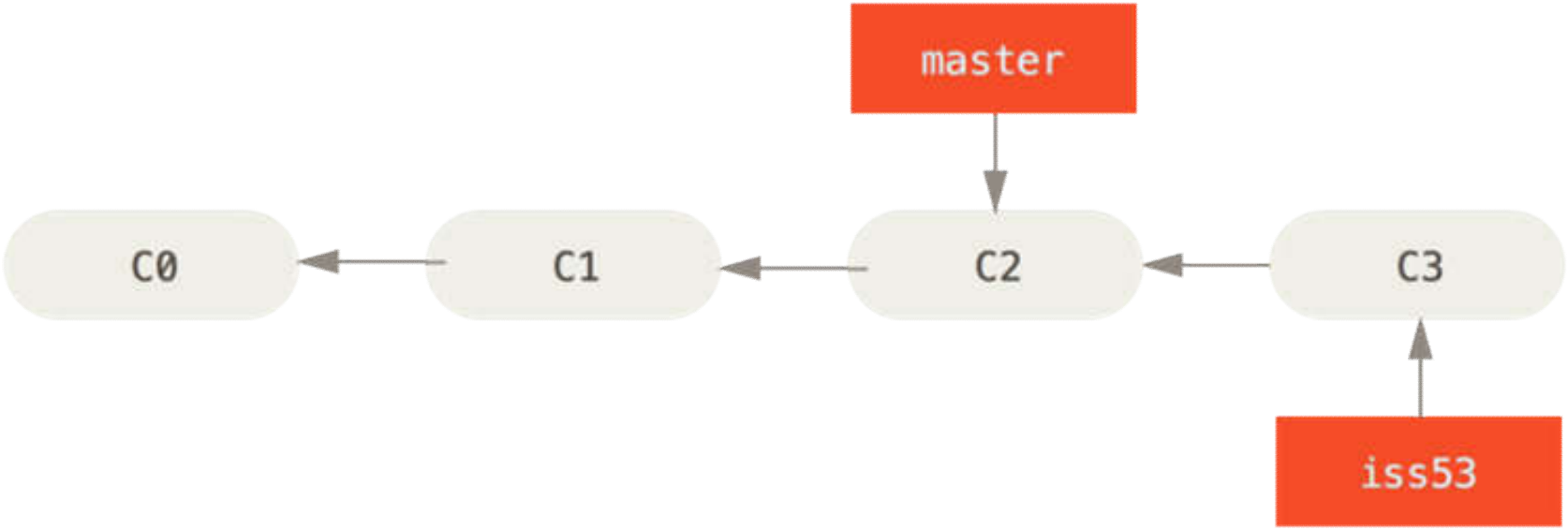
git checkout testing



Основы ветвления и слияния

Работаем над новым функционалом

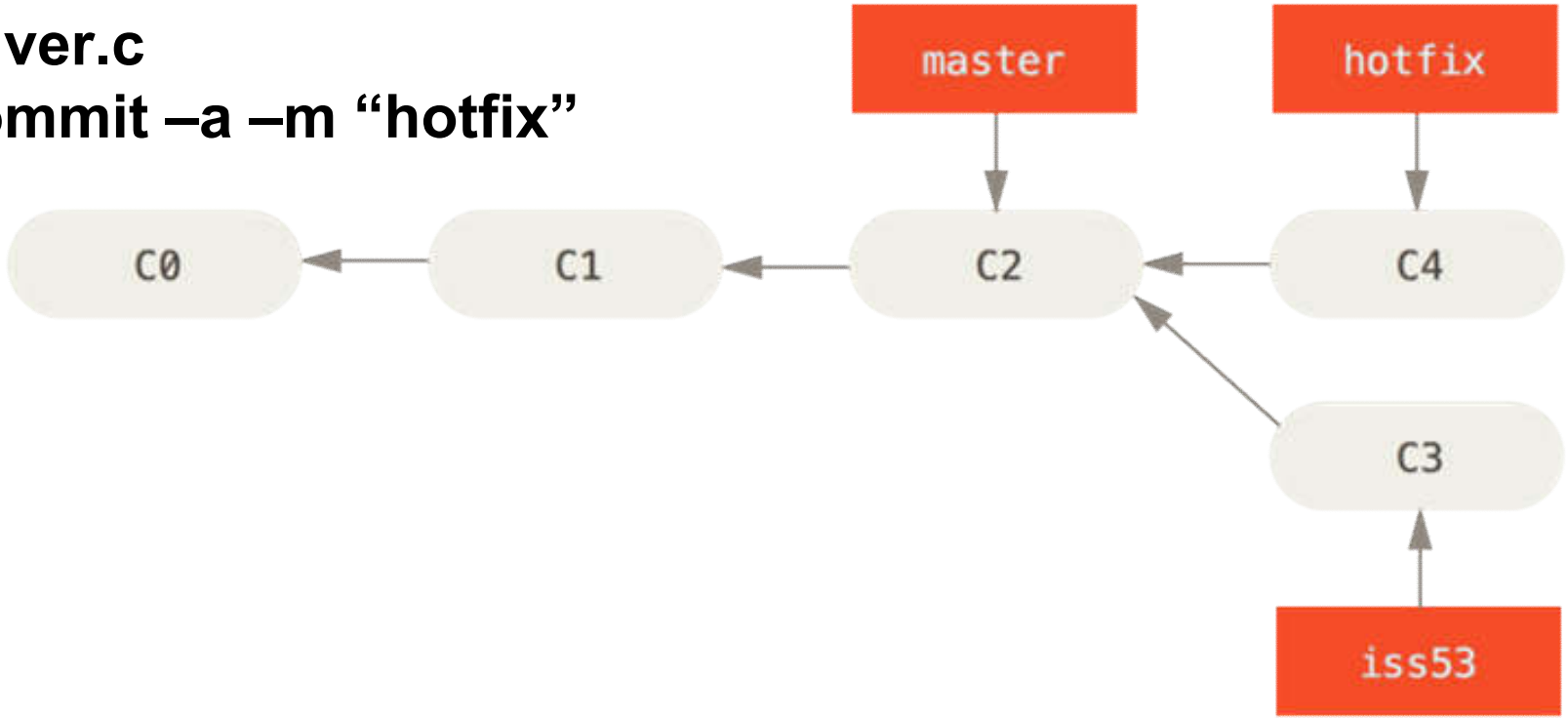
git checkout –b iss53
ee readme
git commit –a –m “new feature”



Основы ветвления и слияния

Исправляем критическую ошибку

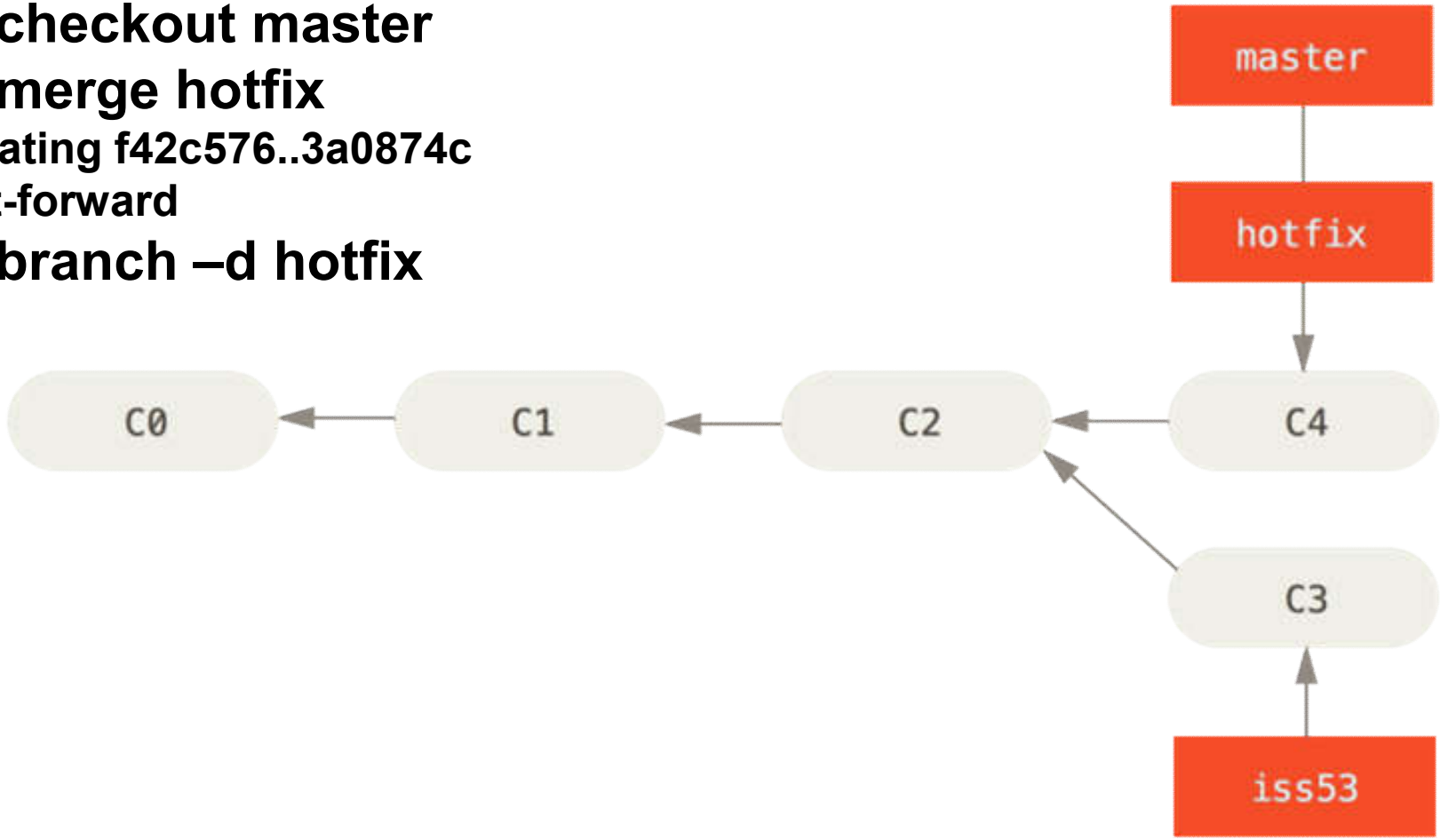
```
git checkout master
git checkout –b hotfix
ee driver.c
git commit –a –m “hotfix”
```



Основы ветвления и слияния

Вливаем хотфикс в основную ветку

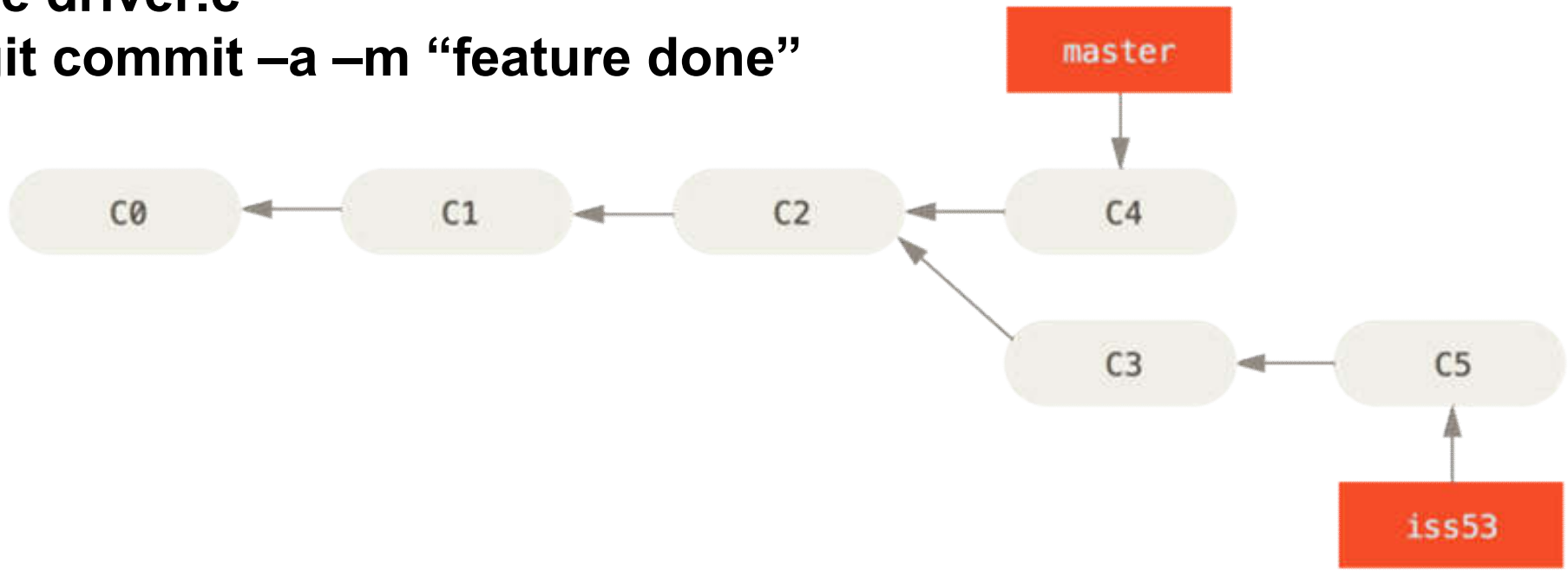
git checkout master
git merge hotfix
Updating f42c576..3a0874c
Fast-forward
git branch –d hotfix



Основы ветвления и слияния

Возвращаемся к работе над новым функционалом

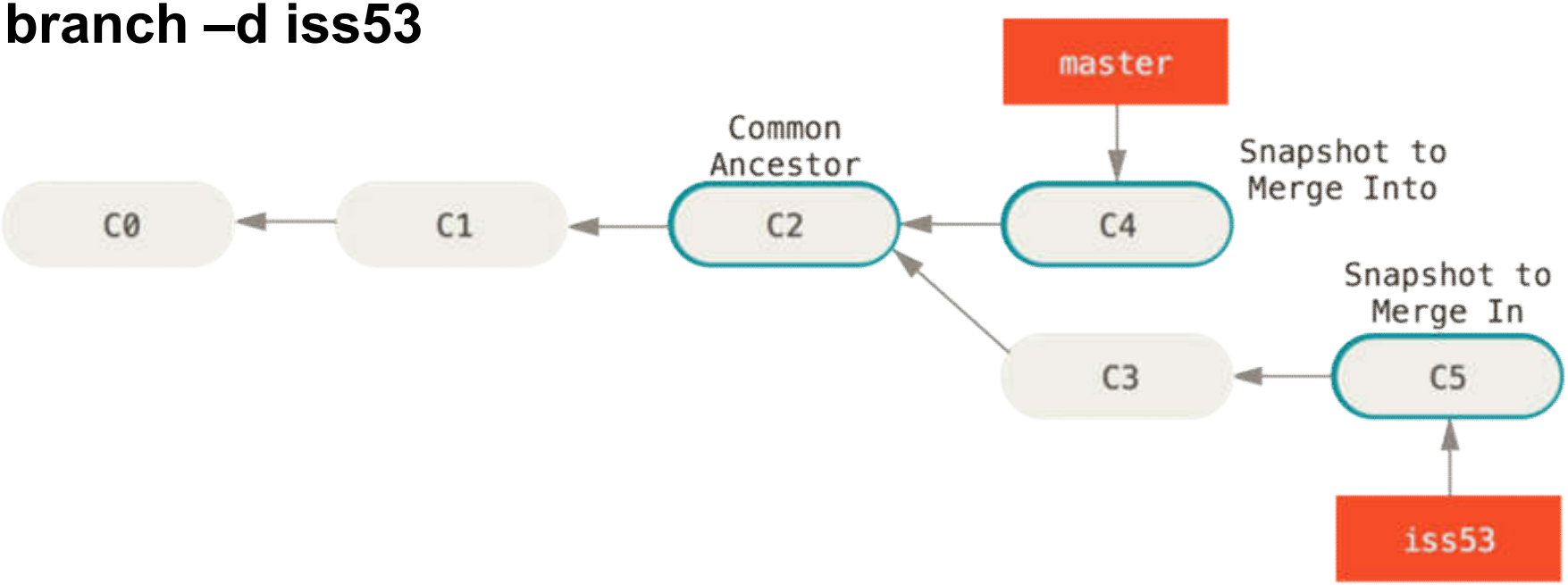
git checkout iss53
ee driver.c
git commit –a –m “feature done”



Основы ветвления и слияния

Вливаем новый функционал в основную ветку

```
git checkout master
git merge iss53
  Merge made by the 'recursive' strategy
git branch –d iss53
```



Удалённый репозиторий

Клонируем к себе:
git clone https://github.com/git/git

Создаем новый и подключаем к нему:
git init
git remote add upstream git@gitlab.slurm.io:red/slurm.git

Удалённый репозиторий

Создаем по шагам:
git init
git remote add origin git@gitlab.slurm.io:red/slurm.git
git fetch origin
git branch –av
git checkout –b master origin/master
git branch –avv

Одной командой:
git clone git@gitlab.slurm.io:red/slurm.git

Удалённый репозиторий

Получаем данные с сервера:
git fetch origin
git checkout master
git merge origin/master

Одной командой:
git pull

Отправляем данные на сервер:
git push

Удалённый репозиторий

Связываем локальные и удалённые ветки:
git checkout –b bname origin/bname
git checkout bname

Создаем ветку и отправляем на сервер:
git checkout –b newbranch
git push –u origin newbranch

Gitlab

Как установить:
https://about.gitlab.com/installation/

Добавляем репозиторий гитлаб:
curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

Устанавливаем:

EXTERNAL_URL="http://gitlab.r01.slurm.io" yum install -y gitlab-ce

Уровни привилегий Gitlab

Guest - могут только создавать Issues

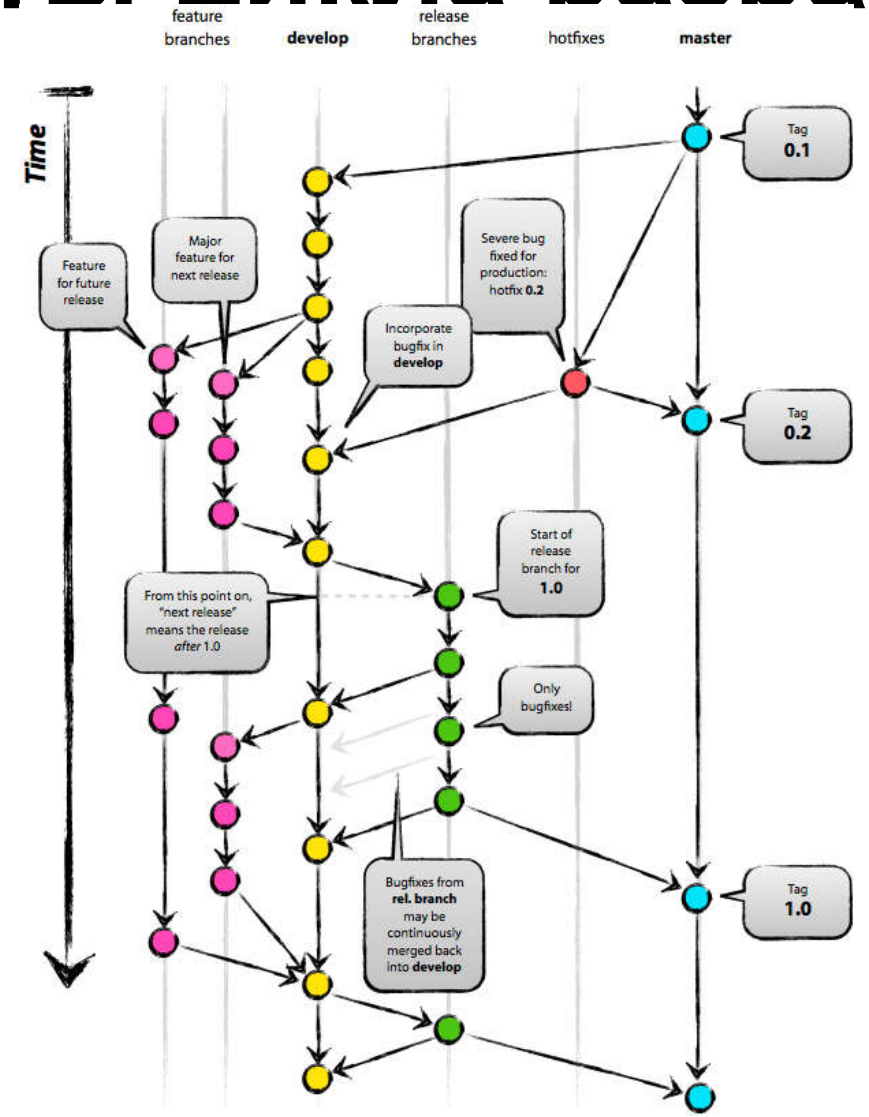
Reporter - могут смотреть код

Developer - могут создавать новые ветки и мерджреквесты

Mantainer - могут принимать мерджреквесты, коммитить в защищенные ветки

Owner - могут назначать привилегии

Варианты цикла разработки



Теги в git

Создаем тег:
git tag v1.0
git tag v1.0 <hashcommit>

Отсылаем теги в удалённый репозиторий:
git push --tags

Получаем описание текущего коммита:
(можно использовать как текущую версию)
git describe --tags

РЕД СЛЁРМ

+

Southbridge

slurm.io