Понимание Git

E-PAGES, Апрель 2016

План

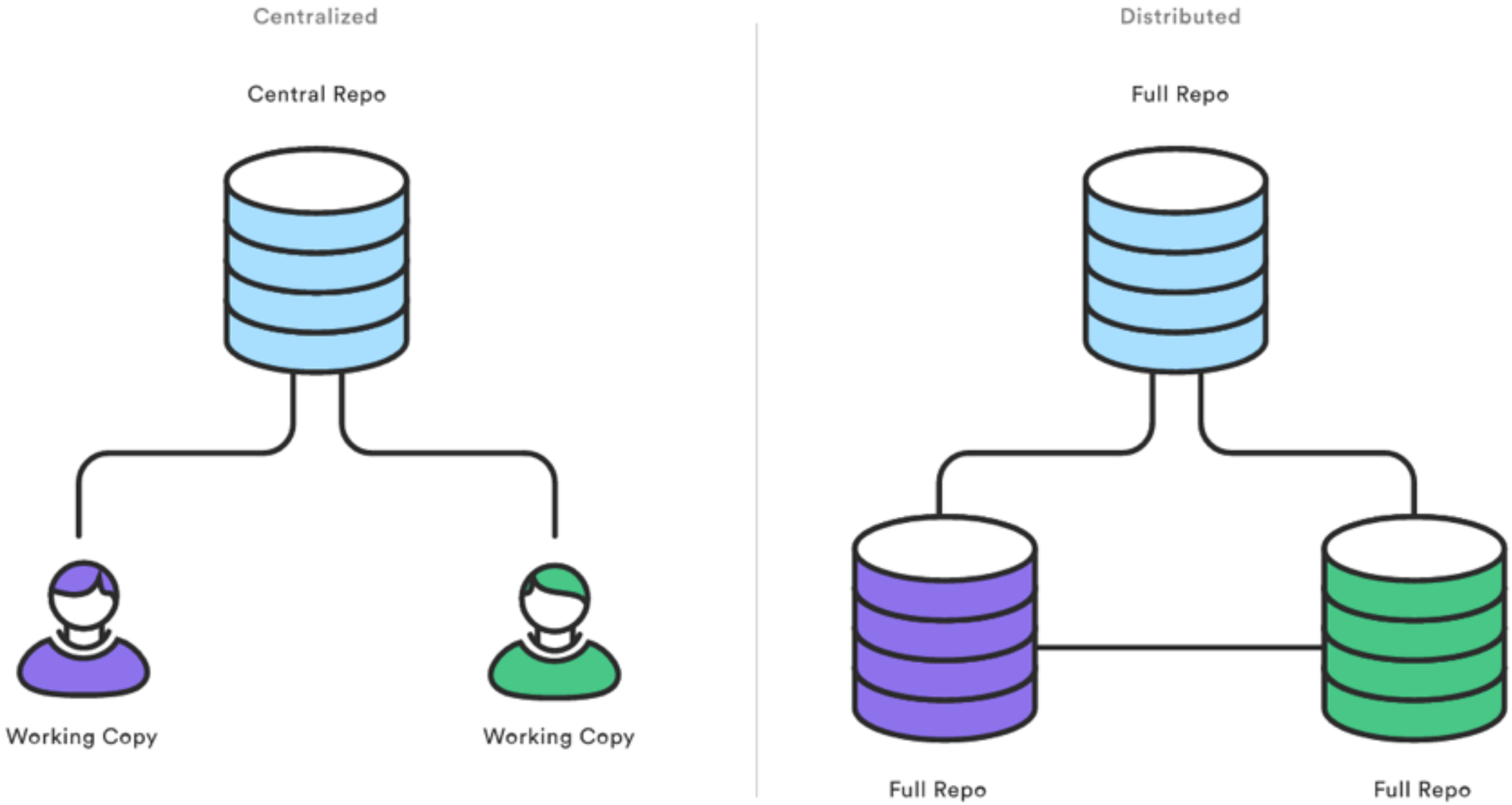
- Понятие распределенной VCS
- Работа с удаленными репозиториями
- Workflow
- Пример команд при работе по нашему workflow

Распределенная разработка

- Git - распределенная система контроля вресий, а SVN - централизованная.

- В SVN есть центральный сервер с репозиторием, а разработчики работают с рабочими копиями. Все операции проводятся с участием сервера.

- В Git каждая рабочая копия есть полноценным репозиторием со всей историей коммитов, которые могут работать сами по себе и “расшаривать" изменения между собой.



Принцип работы централизованной и распределенной VCS

Работа с удаленными репозиториями

- **origin** - это имя удаленного сервера по умолчанию, с которого вы сделали себе клон проекта

- **master** - имя ветки по умолчанию в репозитории

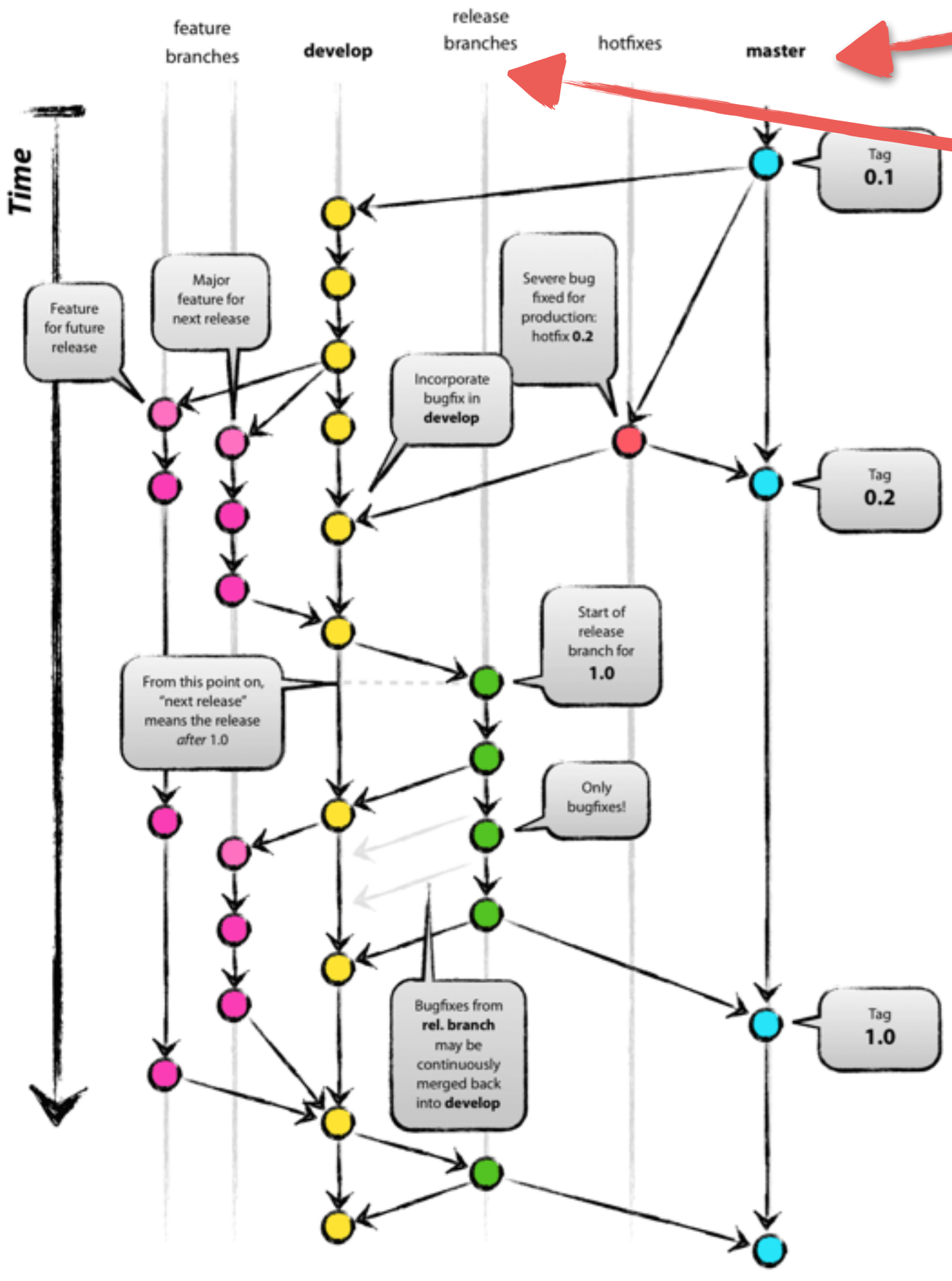
- **fetch** - команда, которая идет на удаленный сервер и получает все изменения в репозитории, которых у вас нет в вашем репозитории, но никак не пытается применить их в локальную рабочую копию, т.е. в ваши файлы.

- **pull** - команда, которая делает все то же, что и **fetch**, но применяет полученные изменения в рабочую копию. Могут быть конфликты, если Git сам не может смержить изменения. Нужно решить их вручную и сделать отдельный коммит.

- **commit** - коммит есть коммит, но если вы его сделали это не значит, что его увидят сразу все разработчики в команде (как в SVN). Коммиты делаются в ваш локальный репозиторий и удаленные репозитории о них ничего не знают, пока вы их не покажете миру.

- **push** - собирает все ваши локальные коммиты и отправляет их на удаленный сервер и тогда их видят все.

Как с этим работать, какой workflow?


feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.1
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Прод

Тест

Пример работы по workflow

- Дано:
  - гит репозиторий (уже склонирован локально)
  - в нем ветки
    - master
    - development
    - release-candidate (rc)
- Задача:
  - Выполнить новую задачу по проекту, выкатить на тестовую площадку для теста, после теста - выкатить на прод

Выполнение задачи (1-4)

1. **Переключиться на ветку development**

   ✓ *git checkout development*

2. **Создать новую фича-ветку**

   ✓ *git checkout -b jira-task*

3. **Выполнить задачу в фича-ветке**

   ✓ *git commit -m 'Add new feature’*

   ✓ *git commit -m 'Add new feature #2’*

   ✓ *git push origin jira-task*

4. **Смержить изменения в ветку development**

   ✓ *git checkout development*

   ✓ *git merge jira-task (будет fast-forward)*

   ✓ *git push origin development*

Выполнение задачи (5-6)

5. **Смежить в ветку rc. На тестовой площадке появляется эта задача для теста.**

   ✓ *git checkout rc*

   ✓ *git merge development*

   ✓ *git push origin rc*

6. **Проходит тест - приходят правки. Правки делаются в ветке rc (после мержа в rc делать коммиты в фича-ветку уже нельзя)**

   ✓ *git commit -m 'New feature fixes'*

   ✓ *git commit -m 'New feature fixes #2'*

   ✓ *git checkout development*

   ✓ *git merge rc*

   ✓ *git push origin development rc*

Выполнение задачи (7-8)

7. **После теста изменения выливаются на прод мержем в ветку master**

   ✓ *git checkout master*

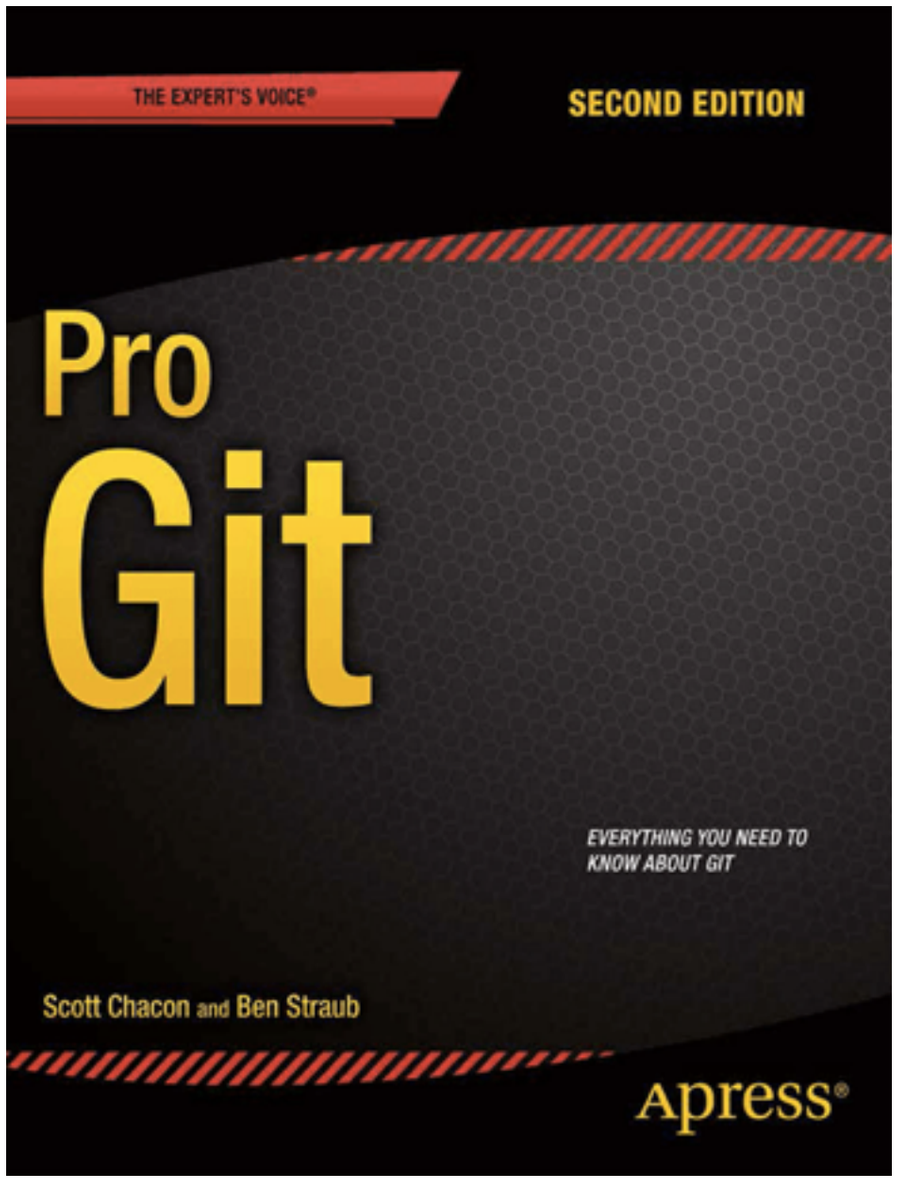
   ✓ *git merge rc (будет fast-forward)*

8. **Делается новый tag с номером версии с применением семантического версирования**

   ✓ *git tag -l ‘1.0.1'*

   ✓ *git push origin master*

Литература



https://git-scm.com/book/en/v2

Вопросы