Совместная разработка веб-проекта с использованием Git в корпоративной среде

Максим Тимохин

Маркетинговая группа Текарт

Собственно, задача

- проекты небольшого и среднего размера, но их много;
- вся разработка – централизованно на сервере проектов;
- большой поток мелких задач – контроль версий необходим;
- разработчики не должны мешать друг другу;
- менеджер проекта контролирует итоговую копию без конфликтов;
- желательно минимизировать накладные расходы.

Система репозиториев:

- рабочая копия для каждого разработчика;
- интеграционная рабочая копия;
- центральный bare-репозиторий: разграничение доступа, исключение конфликтов и т.д.

Стратегии ветвления:

- в репозитории разработчика – task branches обязательно;
- в центральном репозитории – интеграционные ветки для отдельных stages, пока хватает master, но возможны варианты;
- выбор ветки с актуальным состоянием – зависит от специфики проекта.

Центральный bare-репозиторий

DevConf

- разграничение прав доступа;
- единая точка обмена изменениями;
- отображение изменений;
- актуализация интеграционных рабочих копий.

gitolite

Две киллер-фичи:

- контроль доступа не только на уровне репозиториев, но и отдельных веток;
- персональные ветки – обмен кодом между разработчиками через центральный репозиторий без необходимости взаимной авторизации.

Разработчик:

- получает задачу;
- обновление bare/master → local/master
- topic branch;
- решение задачи;
- обновление bare/master → local/master и rebase;
- local/master → bare/dev/developer/master;

Интегратор:

- topic branch;
- fetch bare/dev/developer/master;
- merge с master;
- local/master → bare/master.

Специфика веб-проекта

- необходимо четкое разделение кода и данных, кода мало, данных много, может быть проблемой для legacy проектов;
- для новых проектов – обязательно соблюдать единую структура каталогов;
- код использует контроль версий, данные – нет;
- данные в 80% случаев могут быть расшарены между различными рабочими копиями;
- иногда имеет смысл использовать генерацию кода;

Один из вариантов структуры проекта

DevConf

```
project/
```

Код проекта

```
app/     -- собственно проект
  .git/ -- git-репозитория
  lib/
  www/
    index.php
  Makefile: install, update, check
```

Экземпляр проекта

```
etc/ -- конфигурация экземпляра
var/ -- рабочие файлы
www/ -- реальный DOC_ROOT
Makefile -- symlink или настройки + include.
```

Дерево проектов

```
bin/
etc/
home/           -- рабочие копии разработчиков
  dev1/
    project1/ --  http://project1.dev1.ws/
      .git
stages
  master/       -- мастер-копия проекта
    project1/ -- http://project1.master.ws/
      .git
var/
  project1/     -- статические данные
```

Самое простое – rsync из master-копии, для нас достаточно.

Интересный вариант (мы не пробовали) – система «bare-репозиторий + рабочая копия» на хостинге:

- push в bare;
- hook в bare $\rightarrow$ checkout в рабочую копию;
- изменения в рабочей копии - под контролем версий, можно отправить обратно.

Текущее состояние, проблемы, выводы

DevConf

- в целом жить можно;
- много возни с автоматическим управлением деревом, постепенно доводим до ума;
- с legacy-проектами и проектами на базе готовых решений приходится повозиться;
- хотелось бы проще, но с сохранением git-овских возможностей;
- gitolite – must have для корпоративной разработки;
- разделение данных и кода в проекте – очень важно.

Вопросы?

http://www.techart.ru/

http://github.com/techart/tao-base/