GeekBrains

Git. Базовый курс

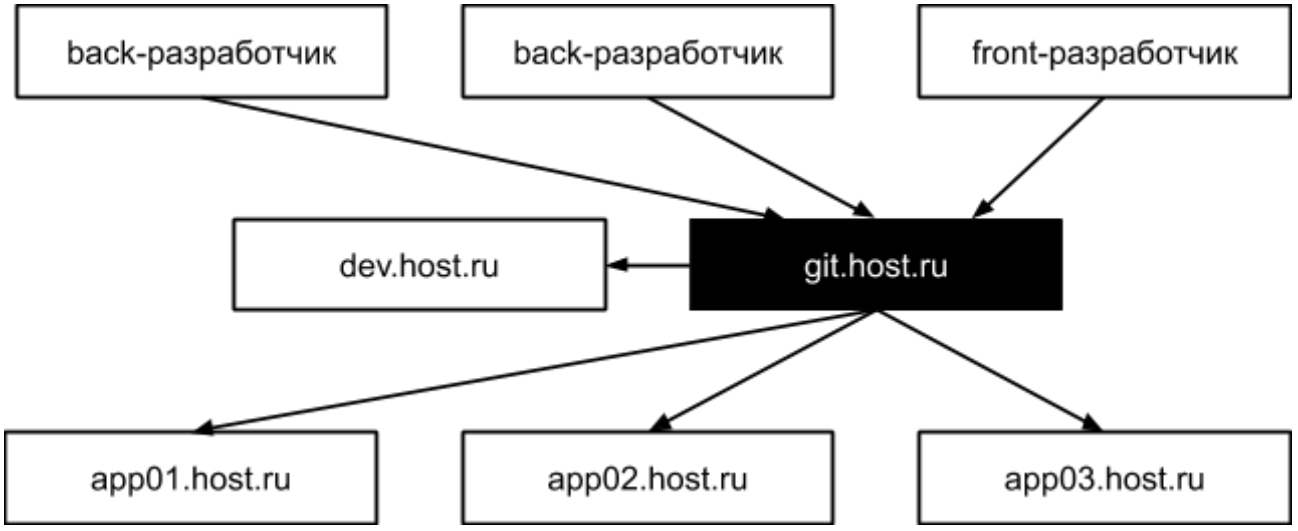
Урок 7

Удаленные Git-репозитории

Удаленный Git-репозиторий

При разработке проекта участники пишут код на своих собственных компьютерах. Чтобы изменения стали доступны другим участникам проекта, их необходимо отправить в удалённый репозиторий, а также иметь возможность загружать изменения, которые сделали другие участники.

Удалённый репозиторий представляет собой компьютер с git, расположенный в интернете или ещё где-то в сети.



Для того, чтобы начать пользоваться удаленным git-репозиторием, проще всего начать с готовых git-хостингов, например, GitHub или BitBucket. Если git-хостинги по каким-то причинам не подходят, можно развернуть свой собственный git-репозиторий: для этого достаточно виртуального сервера с SSH-доступом.

В случае, если помимо консольного доступа необходим веб-интерфейс, можно воспользоваться GitLab, который тоже можно развернуть на собственном сервере. GitLab предоставляет веб-интерфейс для управления git-репозиториями, правами доступа и предоставляет схожий с Git-хостингами функционал.

Git-хостинг GitHub

В последние годы большую популярность приобрел сервис GitHub: он используется как бесплатный git-хостинг для свободных проектов. Допускается размещение любого количества проектов, которые другие пользователи могут клонировать.

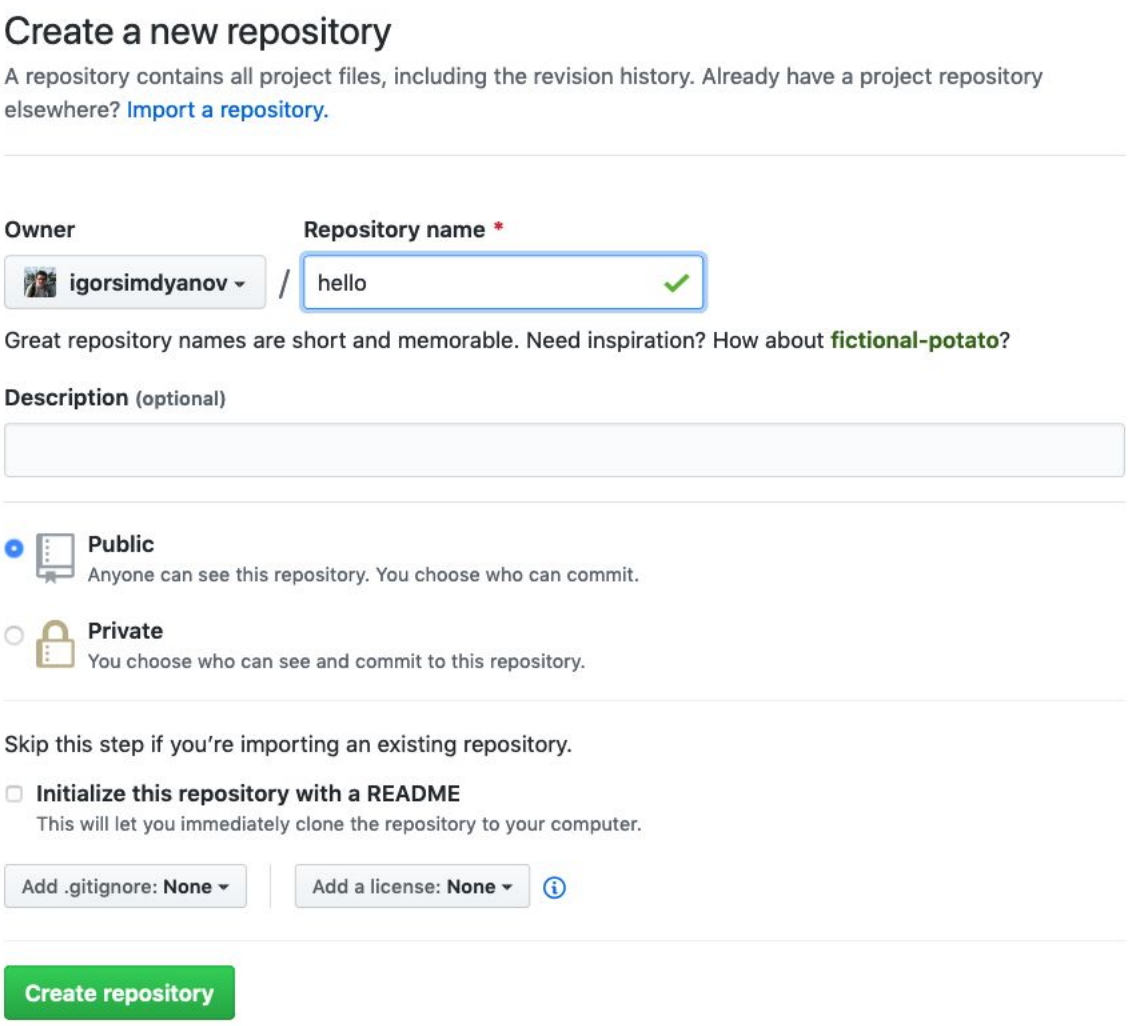
GitHub зарабатывает на предоставлении хост-площадки для закрытых проектов. Если потребуется закрыть код проекта от чужих глаз, придется внести абонентскую плату (впрочем, это касается лишь организаций, где работает более трех человек).

Благодаря политике свободного доступа к открытым проектам, GitHub превратился в площадку для хостинга самых разнообразных свободных проектов, охватывающих все доступные языки программирования.

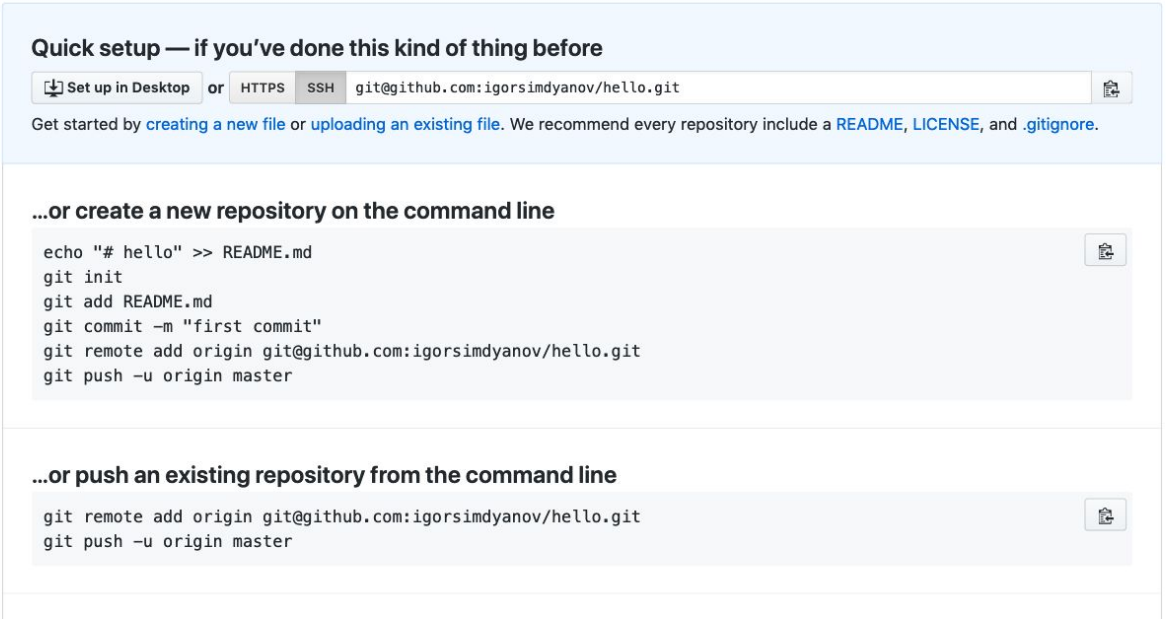
Кроме хостинга, GitHub предоставляет мощный веб-интерфейс для управления Git-репозиторием, трекер задач, Wiki-редактор документации, поиск по проектам, статистику по кодовой базе и многое другое.

Чтобы получить доступ к созданию собственных репозиториев, необходимо зарегистрироваться на сайте http://github.com. Далее следует перейти в редактирование профиля на странице https://github.com/settings/profile. В разделе SSH Key следует загрузить свой публичный ключ. Как его получить в разных операционных системах, вы узнаете на уроке 12.

После этого можно приступать к созданию собственных проектов. В верхнем меню следует найти значок + и выбрать в выпадающем списке пункт New repository, на странице создания будет предложено назвать проект.



Пусть проект будет называться hello. Открывшаяся пустая страница проекта предлагает два варианта инициализации: развертывание проекта с нуля или подключение уже готового репозитория.



У нас уже есть репозиторий, давайте воспользуемся вторым вариантом:

```
$ git remote add origin git@github.com:igorsimdyanov/hello.git
```

Здесь git — имя пользователя на сервере, github.com — адрес сервера, а igorsimdyanov/hello.git — путь к репозиторию на сервере.

```
$ git push -u origin master
Warning: Permanently added the RSA host key for IP address '140.82.118.4' to the list
of known hosts.
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 12 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (35/35), 3.22 KiB | 1.07 MiB/s, done.
Total 35 (delta 8), reused 0 (delta 0)
remote: Resolving deltas: 100% (8/8), done.
To github.com:igorsimdyanov/hello.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Команда **git push** отправляет изменения на git-сервер. При первом обращении может быть задан вопрос, хотите ли вы подсоединиться к новому ssh-серверу. На него следует ответить утвердительно,

набрав yes. При последующих выполнениях команды **git push** данный вопрос задаваться уже не будет.

Если мы сейчас перезагрузим страницу репозитория, то сможем увидеть файлы нашего проекта на github.com.

Команда **git remote add origin** регистрирует удаленный сервер в файле .git/config. Ниже представлено содержимое этого файла.

Можно видеть, что после выполнения команды в нем появляется секция remote с адресом удаленного Git-репозитория.

Файл .git/config можно исправлять вручную.

```
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
    ignorecase = true
    precomposeunicode = true
[remote "origin"]
    url = git@github.com:igorsimdyanov/hello.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
```

Название origin, хотя и предлагается по умолчанию, не является обязательным — его можно изменить. В этом случае в команде git push вместо origin следует указывать соответствующее название репозитория.

Давайте добавим псевдоним **github**:

```
$ git remote add github git@github.com:igorsimdyanov/hello.git
```

Теперь отправлять изменения в репозиторий можно при помощи команды:

```
$ git push github master
```

Т. е., после команды git push сначала указываем имя удаленного репозитория, а затем название ветки.

Список доступных удаленных репозиториев можно узнать при помощи команды:

```
$ git remote
github
origin
```

Клонирование репозитория

После того, как проект развернут в удаленном Git-репозитории, его можно клонировать при помощи команды **git clone**.

Давайте переместимся на уровень выше при помощи команды **cd**:

```
$ cd ..
```

Набираем команду git clone

```
$ git clone git@github.com:igorsimdyanov/hello.git
```

После выполнения команды у нас будет две папки: test и hello. Они позволят нам смоделировать ситуацию командной работы.

Обмен изменениями

Перейдем в папку hello и создадим коммит с файлом index.php:

```
<?php
  echo "Hello world!";
```

Зафиксируем изменения в виде коммита:

```
$ git add index.php
$ git commit -m "Индексный файл проекта"
```

Отправить изменения на сервер можно при помощи команды **git push**, первый параметр в которой соответствует удаленному репозиторию, а второй параметр — ветке.

```
$ git push origin master
```

Так как удаленный репозиторий у нас один, а работа идет с основной веткой master, параметры можно опустить и записать команду в короткой форме:

```
$ git push
```

Сейчас у нас изменения нет, поэтому команда сообщает, что отправлять в центральный репозиторий нечего.

На странице проекта в GitHub можно убедиться, что файл успешно добавлен в репозиторий.

Теперь перейдем в проекте в папку test и создадим коммит с файлом phpinfo.php:

```
<?php
   phpinfo();
```

Переходим в папку test:

```
$ cd ../test
```

Запрашиваем статус проекта:

```
$ git status
```

Добавляем файл в индекс:

```
$ git add .
```

И формируем коммит

```
$ git commit -m "Добавление phpinfo-отчета"
```

Давайте попробуем отправить коммит на сервер:

```
$ git push origin master
```

Попытка завершается неудачей, сервер сообщает, что в удаленном репозитории есть незагруженные изменения, и предлагает их предварительно загрузить. Git предотвращает конфликтные ситуации: если при слиянии возникнет конфликт, Git предложит их разрешить.

Давайте выполним команду **git pull** и скачаем изменения:

```
$ git pull origin master
```

Вот теперь можно отправлять свои изменения на сервер:

```
$ git push origin master
```

Если теперь перейти на страницу проекта в GitHub и перезагрузить страницу, можно увидеть, что файл успешно доставлен в удаленный репозиторий.

Команда **git pull**, как правило, применяется для загрузки лишь одной указанной ветки, для того, чтобы получить все изменения с удаленного сервера, следует воспользоваться командой **git fetch**.

```
$ cd ../hello
$ git fetch
```

Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git: **https://git-scm.com/book/ru/v2.**