* Первоначальная настройка git
* Инструкция по подключению репозитория clone\_calculator
* Правила использования репозитория clone\_calculator

### Первоначальная настройка git

Необходимо настроить ваш git в соответствии с файлом <https://github.com/yukht/clone_calculator/blob/dev/.gitconfig>. После клонирования репозитория (шаги для клонирования описаны в главе «Инструкция по подключению репозитория clone\_calculator»), файл также можно скопировать из него. Имя пользователя **name** в блоке [user] измените на ваш логин github или псевдоним, **email** измените на почтовый адрес github или ваш корпоративный email вручную, либо после копирования файла на свой рабочий ПК измените имя пользователя и почтовый адрес следующими командами ( имя и почтовый ящик замените на свои):  
**git config --global user.name "Ivan.Smirnov"  
git config --global user.email "ivan.smirnov@mail.ru"**  
 В зависимости от версии вашей ОС Windows файл может лежать в различных директориях:  
C:\Users\.gitconfig  
C:\Documents and Settings\All Users\Application Data\Git\config  
C:\ProgramData\Git\config   
 Для того, чтобы определить местоположение исходного файла перед заменой, введите команду  
**git config --list —show-origin**  
 Если вы пользуетесь ОС Linux, то файл **.gitconfig** по-умолчанию будет лежать в директории пользователя:  
~/.gitconfig

### Инструкция по подключению репозитория clone\_calculator

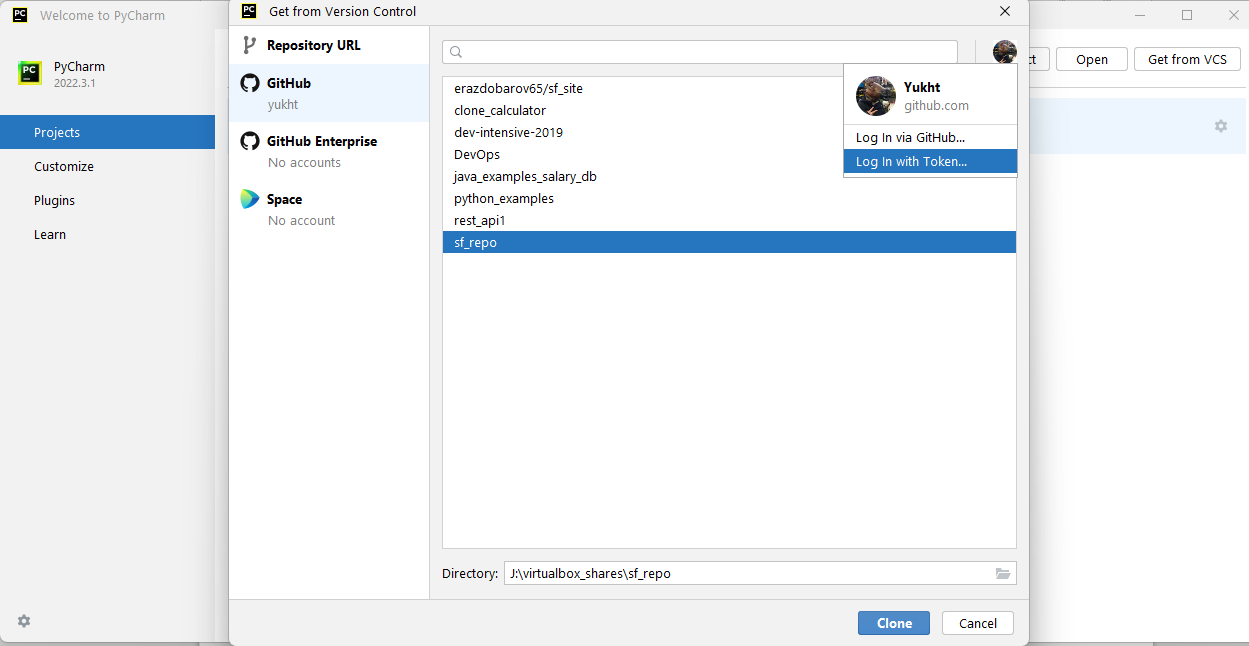
Репозиторий находится по адресу <https://github.com/yukht/clone_calculator>. Для подключения репозитория необходима регистрация на сайте github. Чтобы скачать архив, воспользуйтесь одним из предложенных способов.

**1. Подключите репозиторий к IDE. Пример настройки IDE PyCharm (JetBrains)**

С начальной страницы PyCharm вы можете настроить его интеграцию с github через

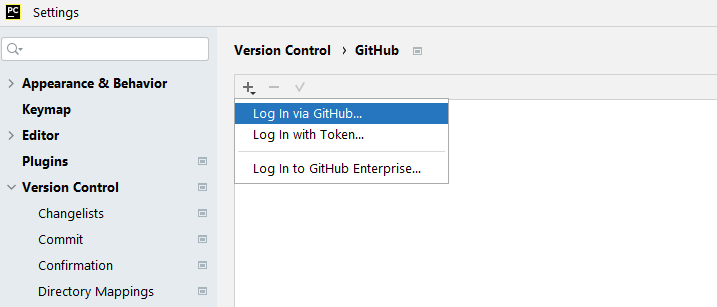
**Projects → Get from VSC → GitHub → Log In with Token**

Вам будет предложено сгенерировать токен. IDE перенаправит вас на сайт github.com и после авторизации запросит подтверждение доступа.



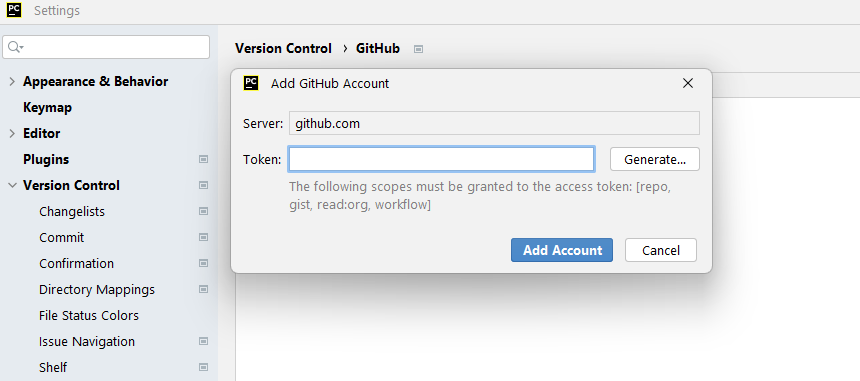
Если вы уже находитесь в открытом проекте, зайдите в

**File → Settings → Version Control → Github**  
Нажмите «**+**» и выберите **Log in with Token**...



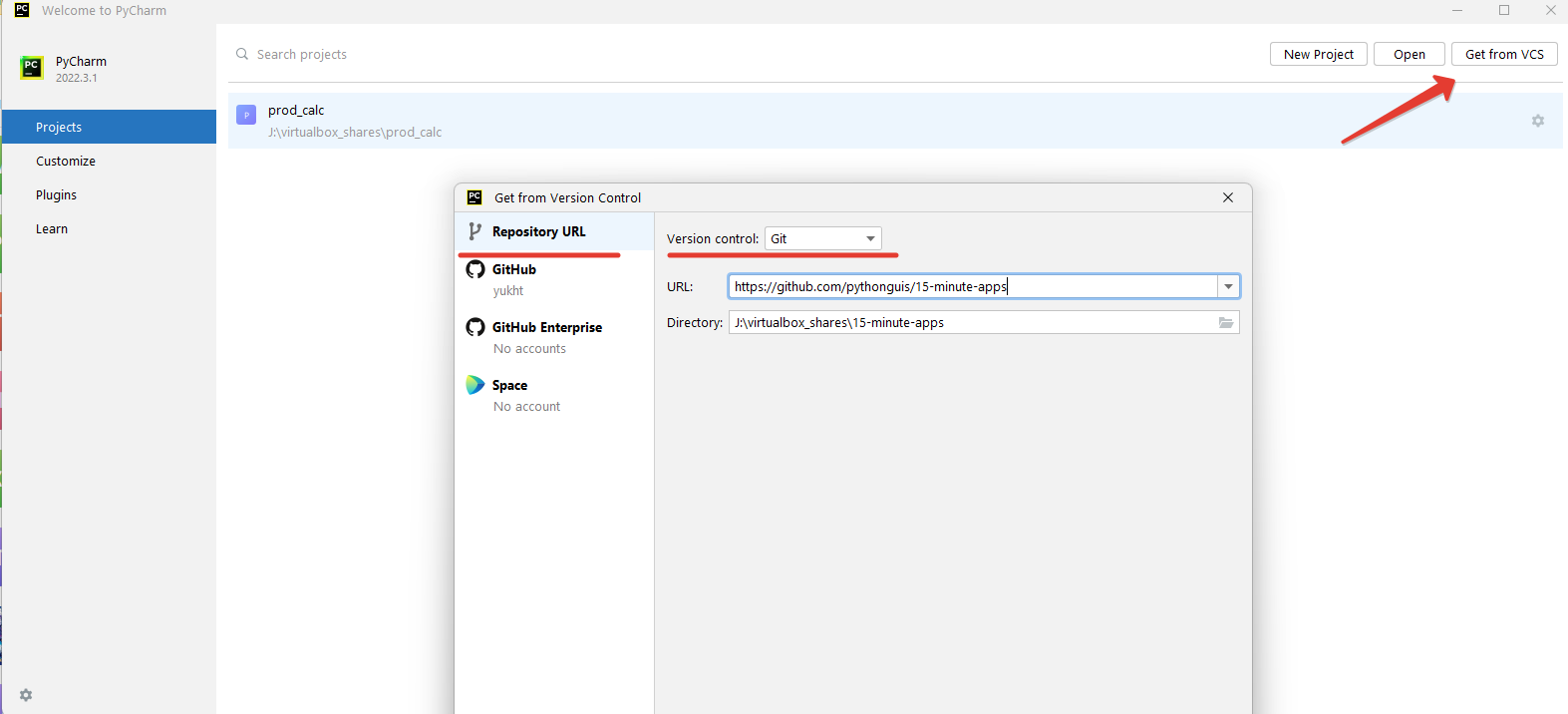
Сгенерируйте токен, приложение перенаправит вас на сайт github.com и запросит

доступ для интеграции IDE

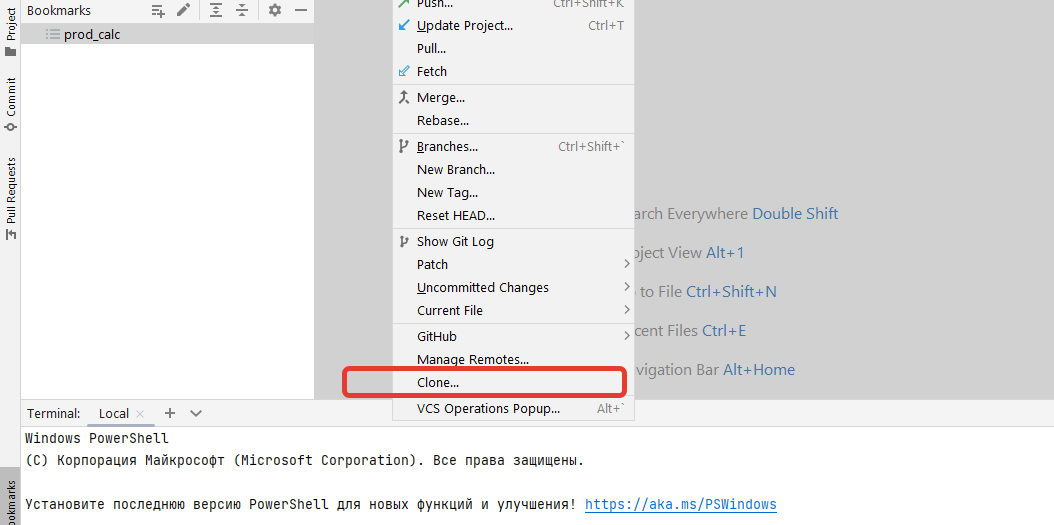


Вы можете клонировать репозитории со стартовой страницы, выбрав  
**Projects → Get from VCS → Repository URL**

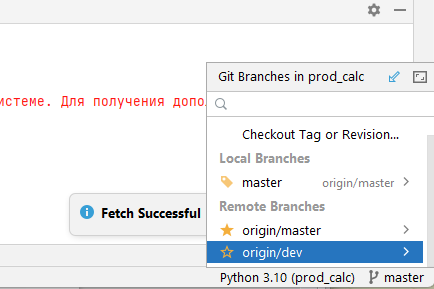
Выбрите **Git** из выпадающего списка «**Version Control**» и укажите url репозитория, начинающийся с указания на протокол HTTPS («**https://**»)

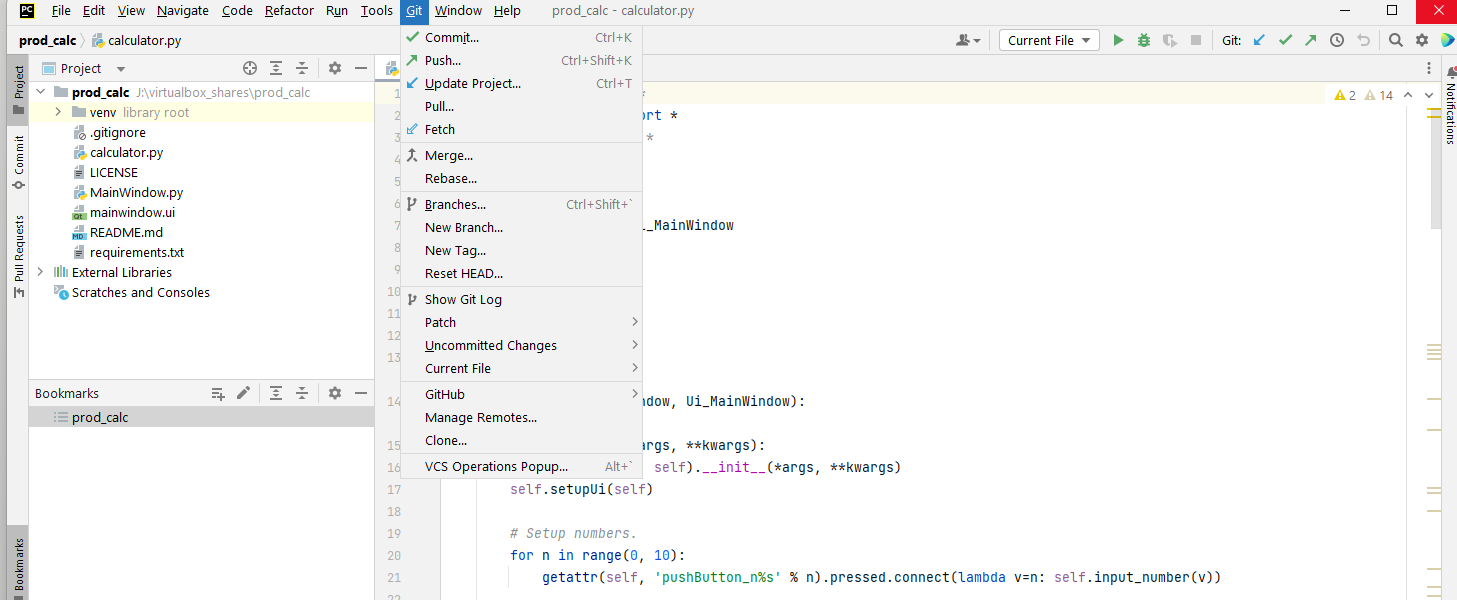


Если вы находитесь в уже открытом проекте, нажмите **Git → Clone…** и откроется то же окно «Get from Version Control»



После скачивания и автоматической установки недостающих библиотек средствами IDE вы окажитесь в локальной ветке Master, сможете увидеть код приложения, скопированный из удаленной ветки Master. Если после **Git → Pull…** скачались не все ветки репозитория (например, отсутствует ветка dev или необходимая вам ветка коллеги), попробуйте загрузить их через **Git → Fetch**



Интерфейс PyCharm интуитивно понятен для дальнейшего использования git. Названия команд git в IDE совпадают с теми, которые будут приведены в п.2 (скачивание и использование репозитория через консоль linux)

**Сразу переключитесь в нужную вам ветку или создайте новую, следуя регламенту, изложенному в главе «Правила использования репозитория clone\_calculator»!**

**2. Скачайте архив, используя терминал ОС linux**

Для этого необходимо сгенерировать пару ключей командой

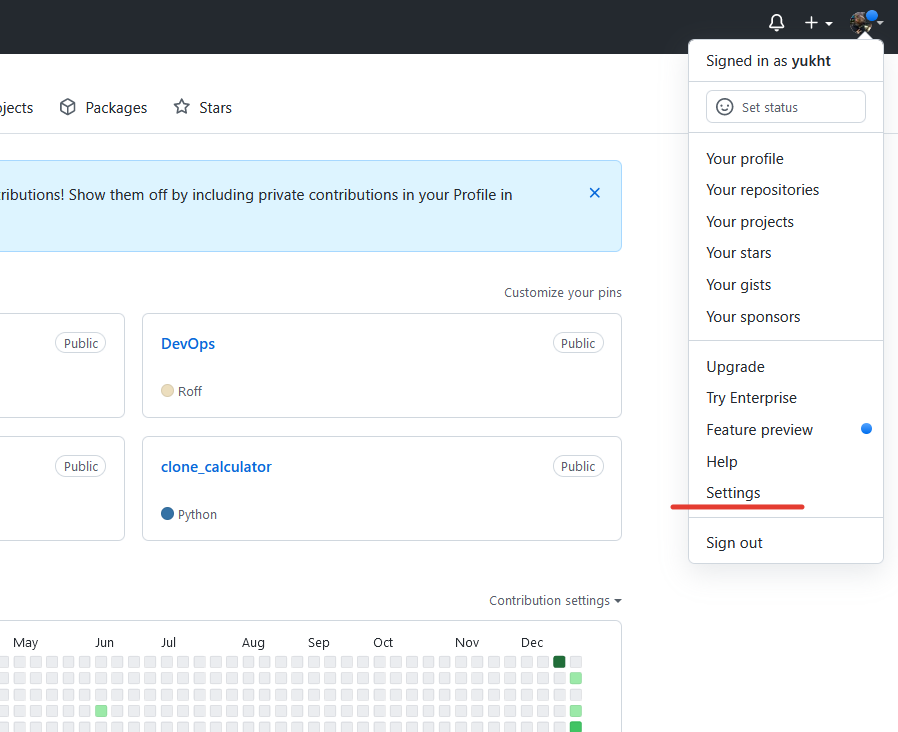
**ssh-keygen**

Ключи генерируются в интерактивном режиме. Вам будет предложено указать имя приватного ключа и путь, по которому его следует сохранить. Укажите следующий путь:

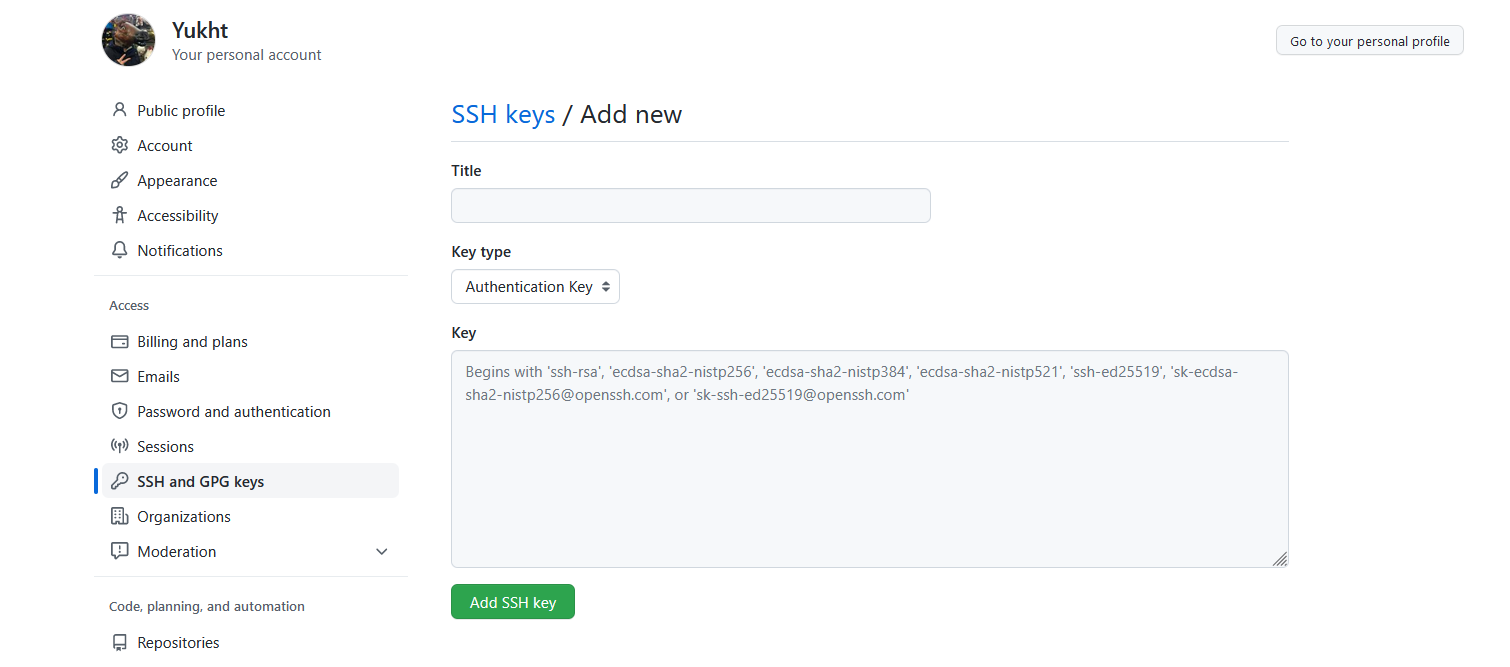
**~/.ssh/id\_rsa**

В директории **/home/XXX/.ssh/** вашего пользователя появится пара ключей: **id\_rsa** и **id\_rsa.pub**. Теперь необходимо зайти на сайт github.com и добавить содержимое файла **id\_rsa.pub** в раздел «SSH keys» вашего аккаунта.

Авторизуйтесь на github.com и выберите **Settings** из выпадающего меню в правом верхнем углу экрана



Нажмите **SSH and GPG keys** → **New SSH key**, введите в появившемся окне ваш публичный ключ **id\_rsa.pub** и нажмите **Add SSH key**.

 Теперь у вас появилась возможность клонировать репозитории, используя протокол SSH. Зайдите в директорию, в которую хотите скачать архив, например, **~/dev\_calculator**

**cd ~/dev\_calculator**

Склонируйте архив.

**git clone** [**https://github.com/yukht/clone\_calculator**](https://github.com/yukht/clone_calculator) **.**

Измените правило доступа к репозиторию (вместо https:// мы хотим использовать протокол ssh):

**git remote set-url origin git@github.com:yukht/clone\_calculator**

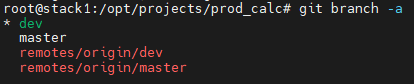
Проверьте возможность скачивания данных с сайта github.com:

**git pull**

**git status**

Если все шаги выполнены успешно, git подскажет вам, что вы находитесь в локальной ветке master. Просмотрите существующие ветки:

**git branch -a**



Переключитесь в ветку dev и создайте новую ветку, или переключитесь в другую целевую ветку (см. правила использования репозитория) и создайте новую. В данном примере предлагается разработка новой фичи через ветвление от ветки dev.

**git checkout dev**

**git pull**

**git status**

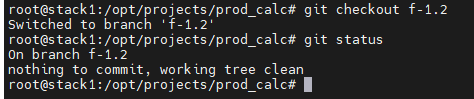
**ls -lah**

Вы должны увидеть в консоли те версии файлов, которые соответствуют ветке dev. Для создания новой ветки и переключения на нее введите следующие команды

**git branch f-1.2**

**git checkout f-1.2**  
 В данном примере f-1.2 это имя новой ветки, которая содержит те же файлы, что и ветка dev. Убедитесь, что вы находитесь в ветке f-1.2 и можете приступать к разработке новой фичи

**git status**



Прочитайте правила использования репозитория (третья глава данного руководства), создайте, если это предусматривает регламент, новые ветки, скоординируйтесь с коллегами по команде. Дальнейшие шаги мы будем выполнять в ветке f-1.2 и результат вливать в dev, чтобы проиллюстрировать пример использования команд git, но эта схема управления ветвлением репозитория не является руководством к действию, если у вас недостаточно полномочий.

После изменения файлов в папке, где у вас был развернут репозиторий и проверки выполненной работы, добавьте все измененные файлы в локальный трекинг (tracking), выполните commit с осмысленным комментарием:

**git add my\_file1 my\_file2 my\_file3**

**git commit -m “Добавил файлы my\_file1, my\_file2, my\_file3, в которых содержатся трактаты Платона”**

Когда изменения достигнут состояния готовности, влейте их в родительскую ветку, выполнив следующие шаги.

Сначала залейте вашу текущую ветку разработки на удаленный сервер

**git push -u origin f-1.2**

Переключитесь на родительскую ветку и выполните слияние (dev 1.2 используется здесь только в качестве примера)

**git checkout dev**

**git merge f-1.2**

**git tag 1.2**

**git push -u origin 1.2**

**git push**

### Правила использования репозитория clone\_calculator

1. Разрабатываемые исправления вливаются членами команды в ветку dev, из нее же создаются новые ветки для выполнения поставленных задач. Master-ветка не должна подвергаться изменениям, пока новая функциональность продукта не будет протестирована и с руководителем проекта не будет согласовано решение о слиянии с master с dev. **Ветка master защищена от перезаписи на github, для слияния с другими ветками потребуется запрос (Pull Request), просмотр изменений руководителем проекта (Review) и согласование**.

1.1 Ветка dev предназначена для разработки, компоновки и полного тестирования новой функциональности продукта, которая в рамках выполняемых членами команды задач достигла статуса готовности. При изменении ветки dev (слияния с другими ветками в процессе текущей разработки) необходимо опубликовать тэг, соответствующий версии изменения.

2 Версии продукта именуются по шаблону 1.[0-9], 2.[0-9] ... N.[0-9] в порядке возрастания. Например, версия 1.6 следует после версии 1.5, в ветке release будет выглядеть как «release-1.6» (см. п. 2.1), при вливании в ветку dev будет установлен тэг "1.6".

2.1 Перед слиянием с dev, новая версия продукта должна пройти промежуточную стадию release-X.X и быть минимально протестирована (как минимум, вручную). Это необходимо для проверки корректности взаимодействия между всеми компонентами, разработанными ранее, с совместными правками от различных исполнителей и/или групп разработки, которые могут работать одновременно над выпуском нескольких новых функций продукта на текущем этапе разработки. Правила управления веткой release представлены в следующих пунктах данного документа.

2.2 Версии могут состоять из нескольких фич (отдельных, полезных с точки зрения потребителя функций). Если новая версия продукта должна включать в себя нескольких фич,

допускается дробление веток на

**dev → f-XXX → [u-bob | u-mary] → release-X.X**

Здесь в имени ветки f-XXX метка "XXX" - имя в произвольной форме отдельно взятой функции, которую планируется включить в релиз, u-bob и u-mary могут быть ветками с именами/псевдонимами отдельных разработчиков/команд, если фича требует привлечения большого количества специалистов. Либо над f-XXX могут работать сразу несколько специалистов в ОДНОЙ отдельной ветке (например, u-all), если в разработке данной фичи участвуют 1-3 человека, т.к. небольшое количество участников снижает количество

вероятных конфликтов при push каждого участника u-XXX в ветку f-XXX. Тогда схема ветвления будет выглядеть, например, так:

**dev-X.X → f-XXX → u-all → release-X.X**

Как уже было упомянуто раньше, веток f-XXX может быть несколько. Итоги совместной работы должны быть влиты сначала в release-X.X, после чего должена быть выполнена первичная проверка работоспособности всех компонентов продукта.

2.2.1 При планировании небольших объемов доработки до новой версии продукта допускается также упрощенное дробление разрабатываемой версии на

**dev → [u-bob | u-mary] → release-X.X**

или

**dev → [u-all] → release-X.X**

То есть разделение dev на одну промежуточную ветку (например, u-all), если в разработке релиза участвуют 1-3 человека и совместно работают с ней, либо на несколько промежуточных веток, имена которых основаны на именах/псевдонимах участников разработки данного релиза.

При упрощенном варианте иерархии сохраняется правило по предварительному тестированию совместной разработки в release-X.X перед слиянием с dev. Если небольшие изменения разработывались в одной ветке по схеме **dev-X.X** **→ [u-all] → release-X.X,** то

в release-X.X выливается готовая версия ветки u-all, также переходит в состояние первичного тестирования перед слиянием с dev. Если проверка прошла успешно, изменения могут быть слиты с веткой dev и подвергнуться дальнейшему тестированию, в т.ч. тестированию временем и тестированию с увеличением нагрузки на сервисы продукта.

2.2.2 Если в результате полной проверки при разработке по п.2.2, п.2.2.1 после слияния с dev обнаружились проблемы, от ветки dev создается ответвление на ветку фикса, которая должна быть названа по шаблону "fix-XXX", где "XXX" - номер исправляемой версии. Ветки dev и release откатывают к предыдущему рабочему коммиту, ветка "fix-XXX" дорабатывается до ожидаемого состояния, после чего снова вливается в release (текущей новой версии) и dev (снова устанавливается тэг текущей новой версии) по описанному выше алгоритму. Таким образом, при возникновении проблем на последнем этапе цикла разработки перед слиянием с dev и master ветки f-XXX становятся избыточными и их можно не использовать (очистить)