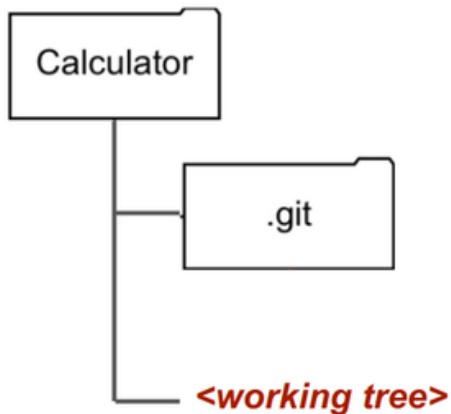


Совместная разработка. Ветвление в GIT

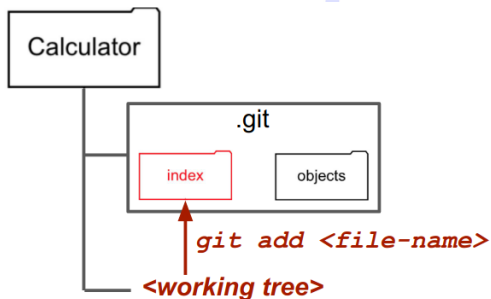
@pvavilin

22 июля 2023 г.

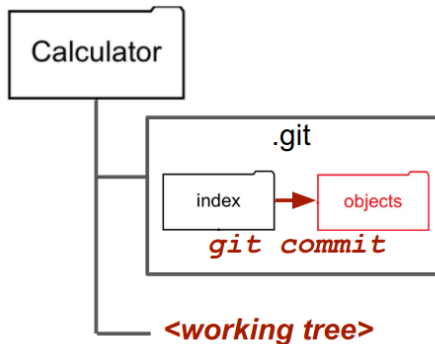
Что внутри .git



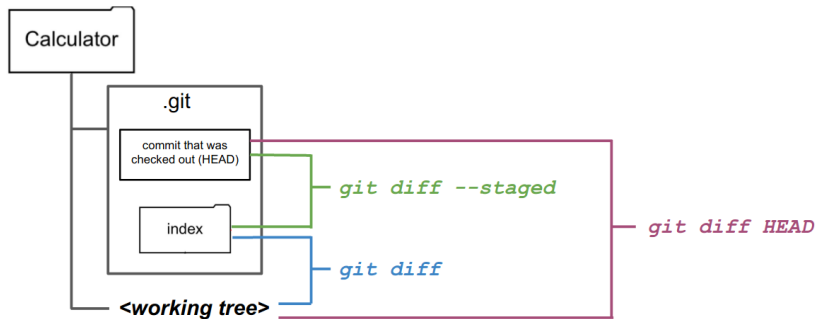
Что внутри .git



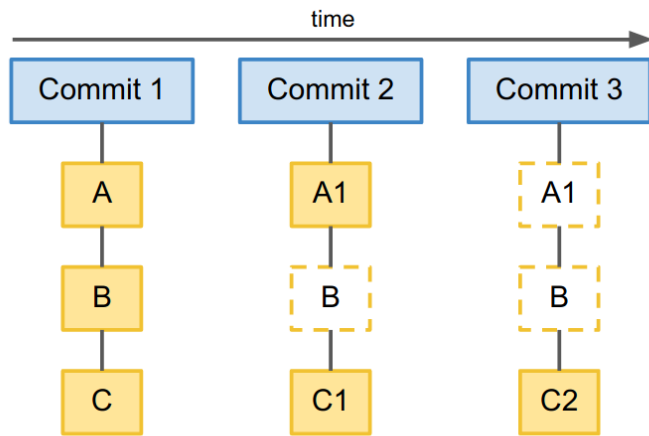
Что внутри .git



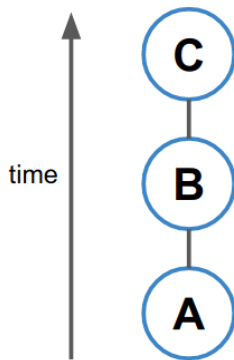
diff



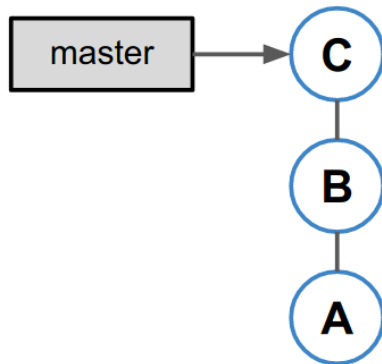
Commits



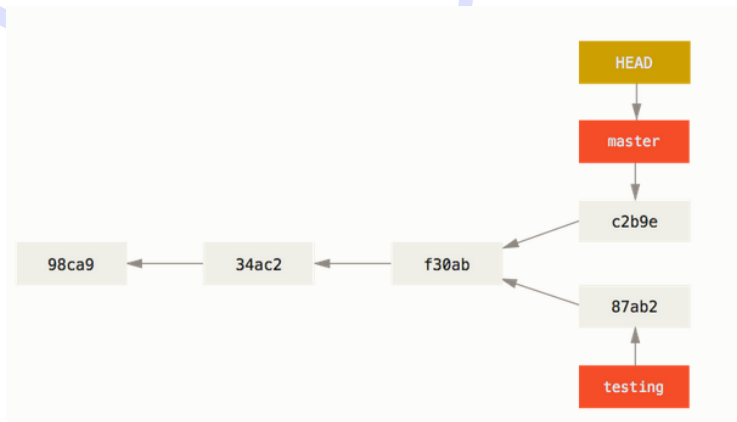
Commits



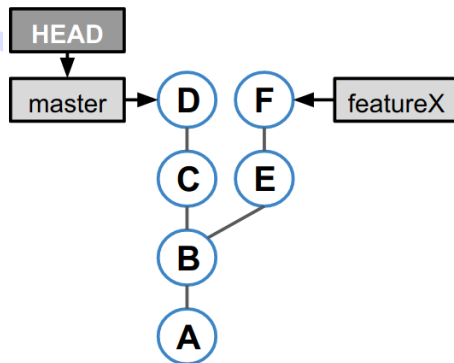
Commits



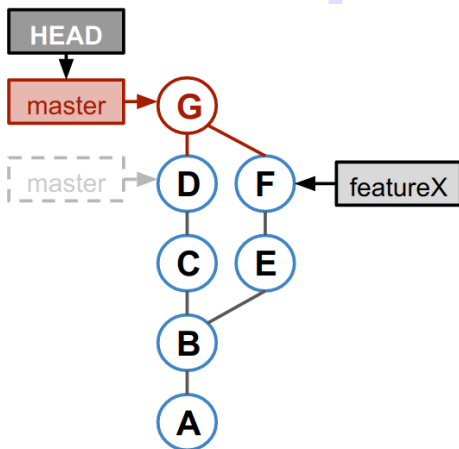
Branches



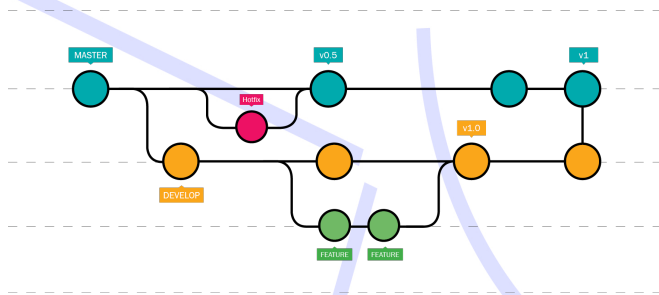
Merge



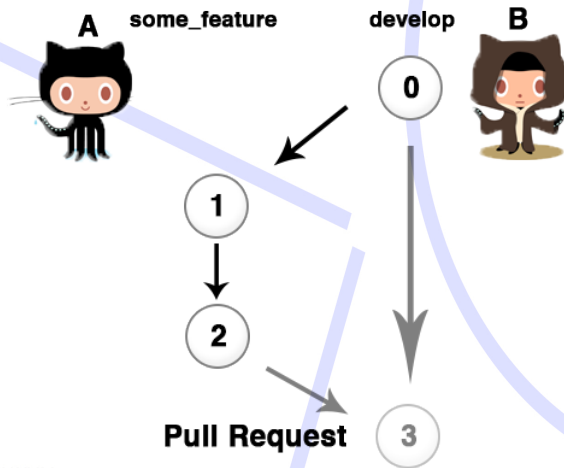
Merge



Merge



Pull Request

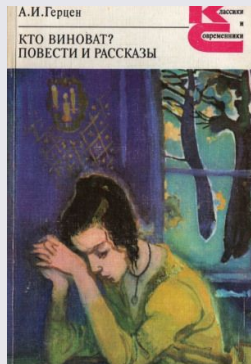


Log

```
git log -p --graph  
# -p : показать правки  
# --graph : показать истории веток
```

Кто виноват?

git blame



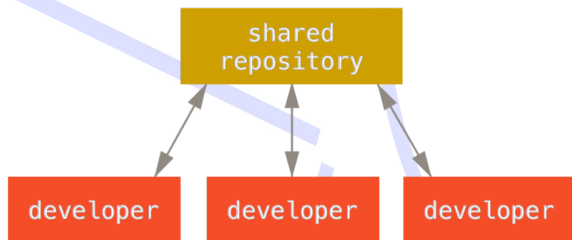
Кто виноват?

```
git blame
```

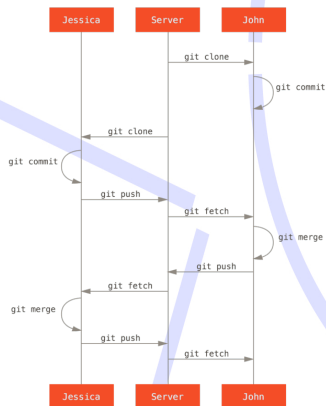
```
git blame <file>
```

```
blame cPython
```


Совместная разработка



Совместная разработка



GIT. что почитать?

Классика

GIT

Советую (english)

GIT Concepts and Workflows

Continuous Integration

Все пушат прямо в *master (main)*

```
git clone ssh://user@host/path/to/repo
git add <some-file>
git commit
git push origin main
```

Важно всегда работать с последней версией *master (main)* чтобы не было конфликтов.

```
git pull
# vi /path/to/file
git add /path/to/file
git commit && git push origin main
```

pull rebase

Используя *rebase* во время *pull* вы не будете создавать дополнительный merge-коммит.

```
git pull --rebase origin main
```

Или настроить автоматический ребейз

```
git config pull.rebase true
```

Разрешение конфликтов

```
git status  
# редактируете файлы с конфликтами  
git add <file>  
git rebase --continue
```

Git Feature Branch Workflow

Для каждой новой *фичи* или задачи создаётся новая ветка. Затем ветка сливается в основной код в *master* (*main*)

```
git checkout -b features/feature-xxx  
# редактируем код  
git add <files>  
git commit  
git push -u origin features/feature-xxx
```

Правила работы с ветками

Должно быть общее правило именования feature веток.

feature/<feature_name>

task/<task-ID>

Правила работы с ветками

Всегда начинайте feature ветку от текущего состояния основной ветки.

```
git pull    # git pull --rebase  
git branch -b feature/<feature_name>
```

Правила работы с ветками

Перед созданием PR вливайте изменения из основной ветки. Все конфликты должны быть решены в вашей ветке!

```
git fetch origin main  
git rebase
```

Правила работы с ветками

Долгоживущие feature ветки зло.

Держите не более 1-2 дней. Для этого четко описывайте задачу, которую хотите решить в ветке.

Правила работы с ветками

Пишите тесты на свой код!

После разрешения конфликтов слияния они вам здорово помогут.

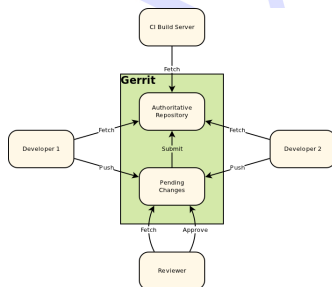
Правила работы с GitFlow

- 1 Релизная ветка (master, main, trunk).**
 - Всегда стабильна, готова к работе в любой момент времени.
 - Работают только лиды/синьоры.
 - Изменения только через Pull Request (PR) из develop или hot-fix веток.
- 2 Основная ветка разработки (develop).**
 - Допускается краткосрочная неработоспособность.
 - Работает вся команда.
 - Изменения напрямую или через PR из функциональных веток.
- 3 Функциональные ветки (feature/<name>), они же фича ветки.**
 - Под каждого разработчика/фичу.
 - Изменения напрямую.
 - Порождается от develop ветки.

Gerrit

Gerrit это надстройка над GIT сервером. Он дополнительно даёт вам

- Code Review
- Контроль доступа к бранчам
- История правок не засоряет log



Gerrit UI

The screenshot displays the Gerrit web interface for reviewing a code change. At the top, the change is identified as '208892: added margin bottom to the paragraph before the create change button'. The interface includes a sidebar on the left with metadata and a main review area on the right.

Metadata (Left Sidebar):

- Updated:** Jan 03
- Owner:** Thomas Shafer (h)
- Assignee:** Set assignee...
- Reviewers:** Dave Borowitz (x), GerritForge CI (x), Mona El Mahdy (x). Includes links for 'AND 2 MORE' and 'ADD REVIEWER'.
- CC:** ADD CC
- Repo:** gerrit
- Branch:** master
- Parent:** bc63487
- Topic:** ADD TOPIC
- Strategy:** Merge if Necessary
- Hashtags:** ADD HASHTAG

Review Status (Left Sidebar):

- Verified:** +1 (GerritForge CI)
- Code-Review:** No votes.
- Code-Style:** +1 (GerritForge CI)

Main Review Area (Right):

- Change-Id:** [I353B49ec8a9d8314450358ff3acf1ce6759c319d9](#)
- EDIT** button
- Files:** Base → Patchset 2 | [bdc5b7e](#) | NO PATCHSET DESCRIPTION
- Commit message:**

```
M polygerrit-ui/avo/elements/change-list/create-change.html | 1 + -
```
- No Tricium Results** (with [More info/File bug](#) link)
- Change Log / Comment Threads:**
 - Only comments** (toggle)
 - Thomas Shafer (h)** Uploaded patch set 1. (Dec 29 00:42)
 - Thomas Shafer (h)** added to REVIEWER: Patrick Hiesel, Arnab Banerjee (j). (Dec 29 00:42)
 - GerritForge CI** added to REVIEWER: GerritForge CI. (Dec 29 01:59)
 - GerritForge CI** **Code-Style +1** Patch Set 1: Code-Style+1 ✓ All files are correctly formatted (<https://gerrit-ci.gerritforge.com/job/Gerrit-codestyle/35111/console...>) (Dec 29 01:59)

Литература

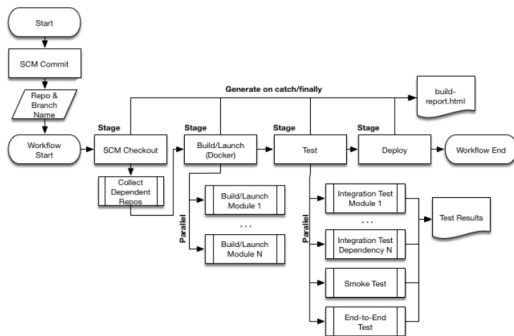
- GIT Concepts and Workflows
- GIT branches
- GIT workflows
- GitHub flow
- Gerrit

DevOps

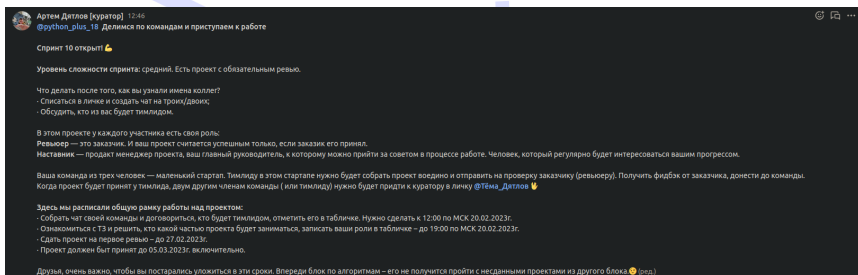
- Coding
- Building
- Testing
- Deploying (packaging, releasing)

Автоматизация DevOps

Jenkins



Спринт 10: Командная работа



Арте́м Де́тлов [куратор] 12:46
@python_plus_18 Делимся по командам и приступаем к работе

Спринт 10 открыт! 🍀

Уровень сложности спринта: средний. Есть проект с обязательным ревью.

Что делать после того, как вы узнали имена коллег?

- Списатьсь в личке и создать чат на троих/двоих;
- Обсудить, кто из вас будет тимлидом.

В этом проекте у каждого участника есть своя роль:

Ревьюер — это заказчик. И ваш проект считается успешным только, если заказчик его принял.

Наставник — продакт менеджер проекта, ваш главный руководитель, к которому можно прийти за советом в процессе работы. Человек, который регулярно будет интересоваться вашим прогрессом.

Ваша команда из трех человек — маленький стартап. Тимлиду в этом стартапе нужно собрать проект воедино и отправить на проверку заказчику (ревьюеру). Получить фидбэк от заказчика, донести до команды. Когда проект будет принят у тимлида, двум другим членам команды (или тимлиду) нужно будет прийти к куратору в личку @Тема_Детлов 🍀

Здесь мы расписали общую рамку работы над проектом:

- Собрать чат своей команды и договориться, кто будет тимлидом, отметить его в табличке. Нужно сделать к 12:00 по МСК 20.02.2023г.
- Ознакомиться с ТЗ и решить, кто какой частью проекта будет заниматься, записать ваши роли в табличке — до 19:00 по МСК 20.02.2023г.
- Сдать проект на первое ревью — до 27.02.2023г.
- Проект должен быть принят до 05.03.2023г. включительно.

Друзья, очень важно, чтобы вы постарались уложиться в эти сроки. Впереди блок по алгоритмам — его не получится пройти с несданными проектами из другого блока. 🍀 (ред)

Избегаем блокировок

- 1 В первую очередь пишем модели БД.
 - Лучше всего, если каждая модель в отдельном файле.
 - Если модель для `ForeignKey` не существует, допустимо сделать *IntegerField*, затем заменить на *ForeignKey*.
- 2 Если ваша ветка живет более 4-х часов, то вливайте изменения из основной ветки дважды в день. Все конфликты должны быть решены в вашей ветке!
- 3 Поддерживайте тесную связь. Синхронизация Pull Request-ов, правок кода сильно ускорит работу над проектом. Задача лида быть всегда в курсе что происходит с проектом, кто над чем работает.
- 4 Никогда не изменяйте историю

```
git push -f
```

Командные процессы

- 1 Ежедневные встречи (daily meetings). Если нужна помощь в проведении, приглашайте наставников.
- 2 Планирование в начале проекта и по необходимости в ходе работы.
- 3 Совместная работа в трекере задач (Trello/Jira и т.п.).
- 4 Команда обсуждает вопросы, по которым требуется помощь и делегирует лиду обсуждение этих вопросов с наставником.

Вопросы-ответы

