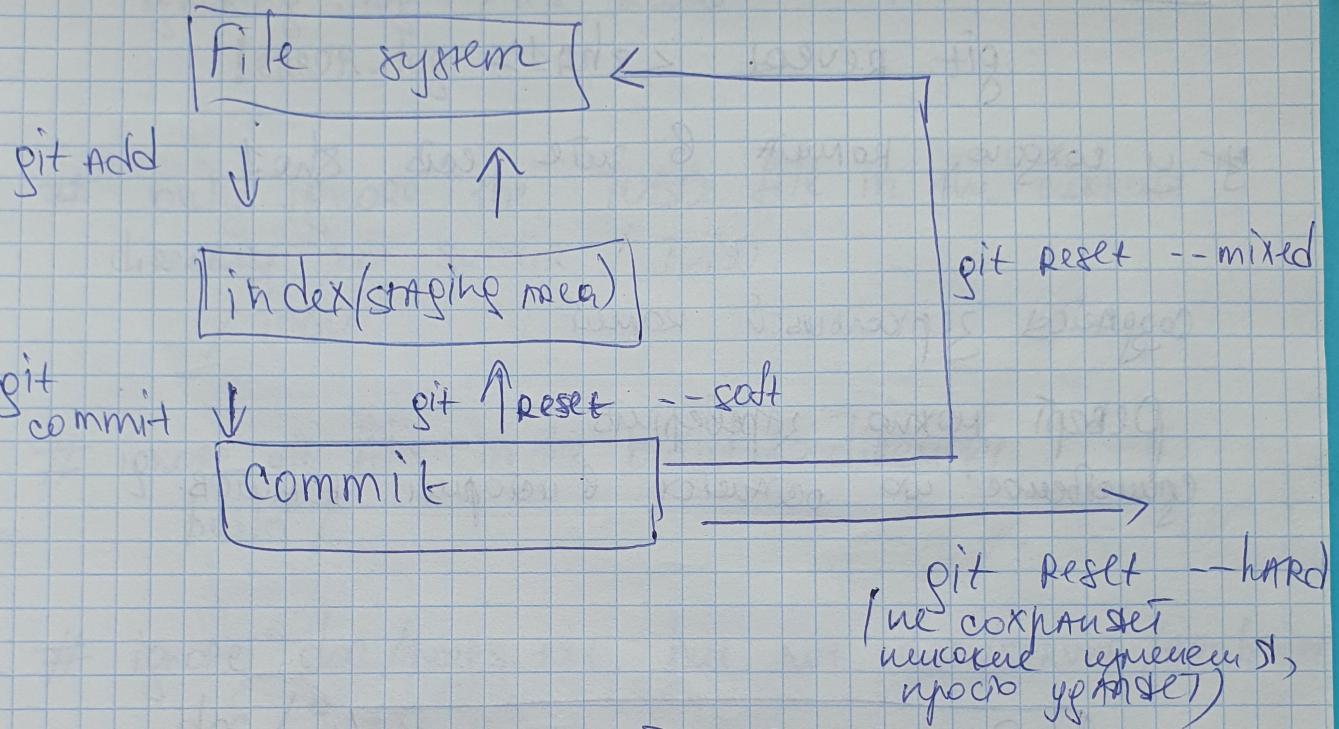


# GIT RESET

3 основных состояния изменений:

[1] Изменения в Index → в ожидании коммита, есть не забытое



git reset номинально не просит изменений  
и не спрашивает у пользователя.

Но умолчанию используется option **mixed**.

**git reset --hard HEAD ^**

всёе откатано  
команда  
уже не  
уберегаем.

## Git revert

Как отменить изменения, которые мы не только  
закомитили, но и уже запустили на PUXAB.

### Remote Repository

git revert <sha1>

У ветви в которую внесли ошибку

создается зеркальный коммит

реверс можно забререть  
единственное что останется в истории - коммитов.

## git .gitignore

Чтобы не загружать изменения в отре-  
вених файлах.

### ① Создать файл .gitignore

Коммит .nenv → бинарные файлы или скрипты,  
которые не могут быть отслеживаться движком  
коря проекта (скрипты), скрипты можно обрабатывать  
инструктами

Многие из них могут помочь и убедиться в этом

## CUSTOM CUC

# no .log files  
\*.log

# but do track error.log, even though you're  
ignoring .log files above  
!error.log

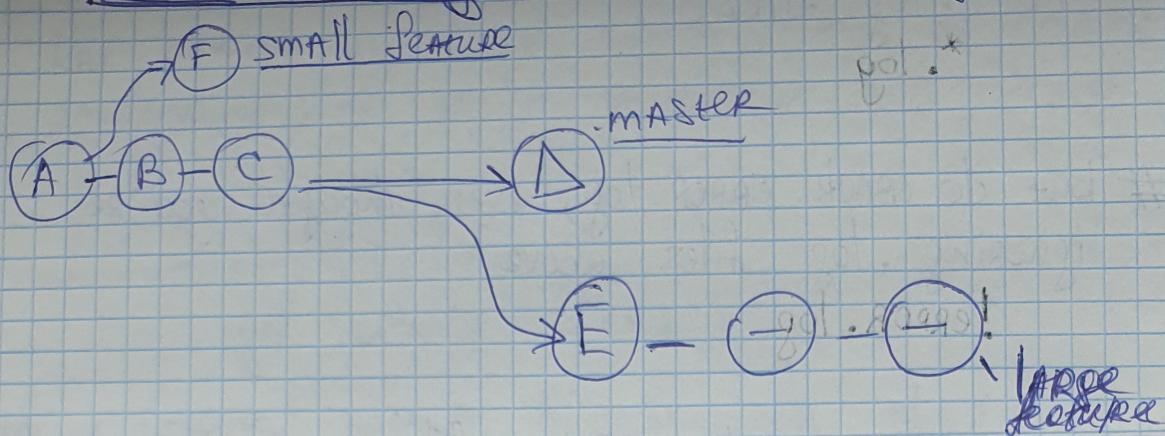
# only ignore the TODO file in the current  
directory, not subdir / TODO  
/TODO

# ignore all files in the build / directory  
build /

# ignore doc/notes.txt, but not doc/server/arch.txt  
doc/\*.txt

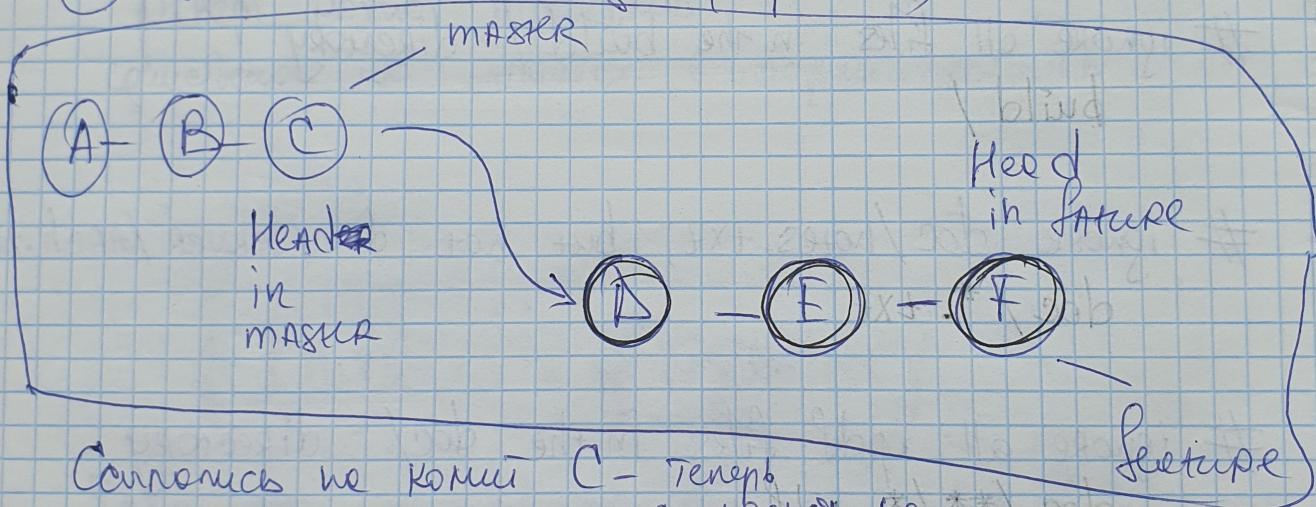
# ignore all .pdf files in the doc/ directory  
doc/\*\*/\*.pdf

# Branching and merge



Branch - у нескольких коммитов есть один общий предок.  
Замыв ветки в мастер - merge (слияние)

## I fast forward merge (перемотка)



Со временем коммит C - Теперь  
существует коммит F  
(неподвижно Head с C на F)

Создать ветку и срезу B не переключается  
git checkout -b feature

- ! Переходку в той ветки в которой хочешь
- Внести изменения

В начале сущес master

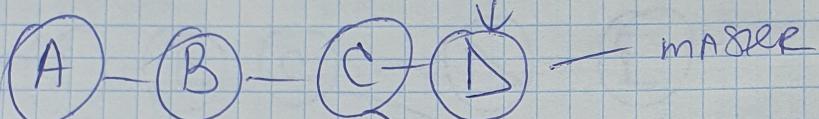
Уз mastera делают git merge

git merge feature

## Non-fast forward merge

но это , но создает дополнительный коммит

head  
in master



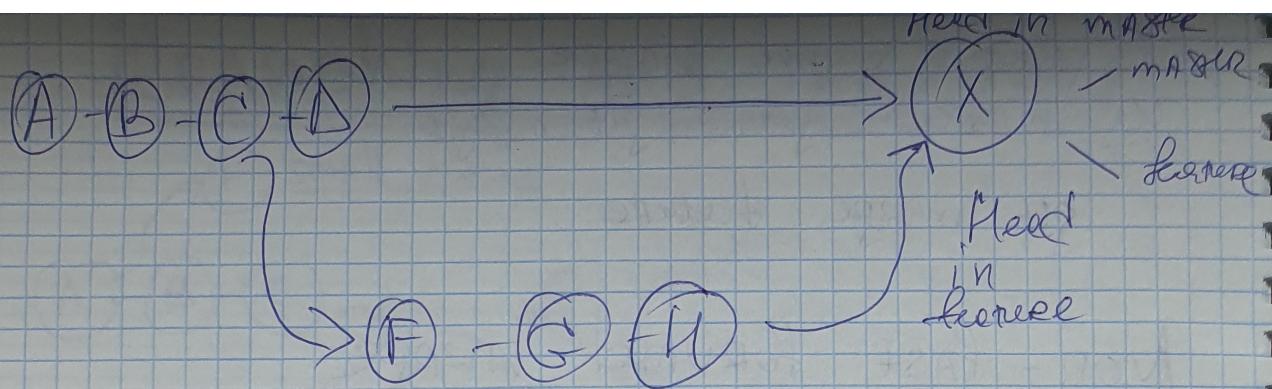
feature

head  
in feature

Если просто перебросим head в master, то  
потеряем коммит D

Решение:

Создаем еще один коммит — результатирующий,  
в который и собираем изменения из  
коммитов F-G-H и D



они будут называться "БУ"

: wq

## Conflict solving

|||| HEAD

некоторые из них могут в мастер

### Solve Conflict

Abort merge

git merge --abort (отменить балды merge)

Resolve by selecting version

git checkout --ours -- theirs

git checkout --ours -- theirs  
отменить изменения  
кто чукали  
правильная  
или  
наоборот

Resolve manually

git diff

Undo merge

git revert d8fe472

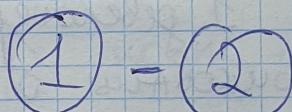
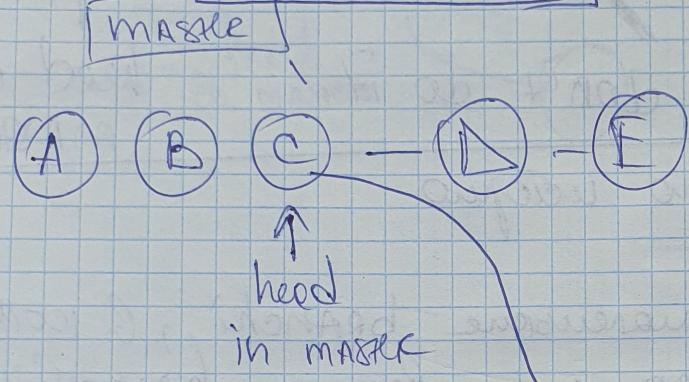
User merge tool

## Avoid conflict

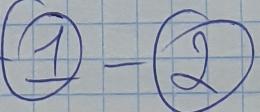
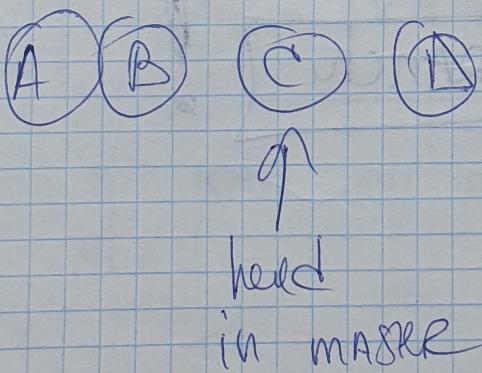
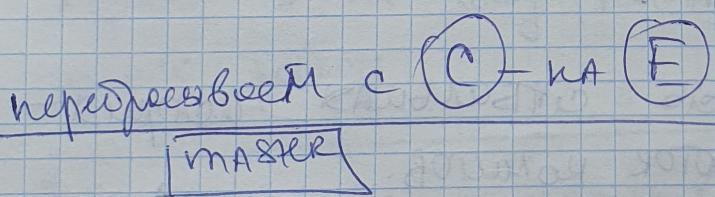
- Short commits
- No edits to whitespace
- Merge often.

## REBASE

against Dots)



feature



feature

of  
head  
in feature

КАК ~~НЕ~~ нужно делать Rebase:

Хорошо работает только в своем малом окружении

MASTER

A B C → ① - ② - ③ - E



Just don't do it

↑  
need in  
MASTER

Изменение в уже используемой истории

Rebase мержит только мелкие branchи, с которыми работает 1 единичная feature branch  
которые относятся к исчезающим

Но мы в коем случае нельзя Rebase используемых  
branchи таких MASTER.

Люди у которых есть относимые исходи-  
тельно близкие в них комитов.

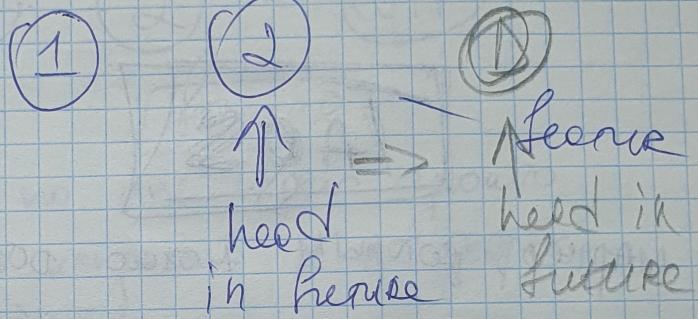
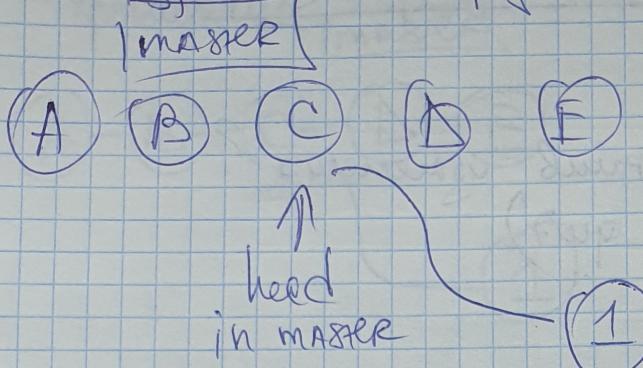
The Rebases public branch!

!

## CHERRY - pick

Беремся б Rebase.

называем основу Negro.



Я хочу нанести комит D, но не хочу генерировать Rebase, потому что мне не нужен E. А

нужен я не хочу, потому что не сделаю это above today.

Cherry - pick - копирование комита б отрасли

Rebase это ошибка и заменяет комит, т.к. с ним SHA 1

28A5 - 181d - f19

Наша фраза f19

Что такое BRANCH в git и как с ним работать?

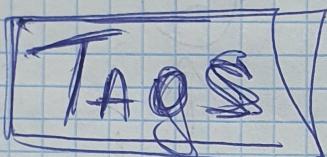
Ответ: Git BRANCH

Как просмотреть список файлов в папке с помощью git?

git branch --all

Как выйти из VI редактора в git?

:wq (write and quit)



\* МАКСИМ, который может дод

git tag version1

~~затем~~ ТЕПЕРЬ можем использовать Branch SHA1

использовать git checkout version1 - не нужно б  
тот коммит, который останется ТЕПЕРЬ version1

MARK commit with TAG

git tag ver1

View TAGS

git tag --list

Push

git push --tags

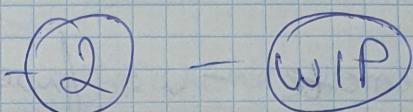
Check it out

git checkout ver1

# Stashing

возможность сохранять общий историю по времени

Хранение в MASTER



feature

→ поддано в WIP, но мне нужно спасти неиспользованные на мастер ветвь для проекта.  
(но я не могу это сделать)

! Git предоставляет хранение Stash.

Tl, чтобы сортировать или отменить Git  
команды в Stash.

Save working directory

некоторые из них

git stash & save "description"

View stashes

git stash list

Bring them back

снимая с stash

git stash pop (удалить из stash) но удаляя

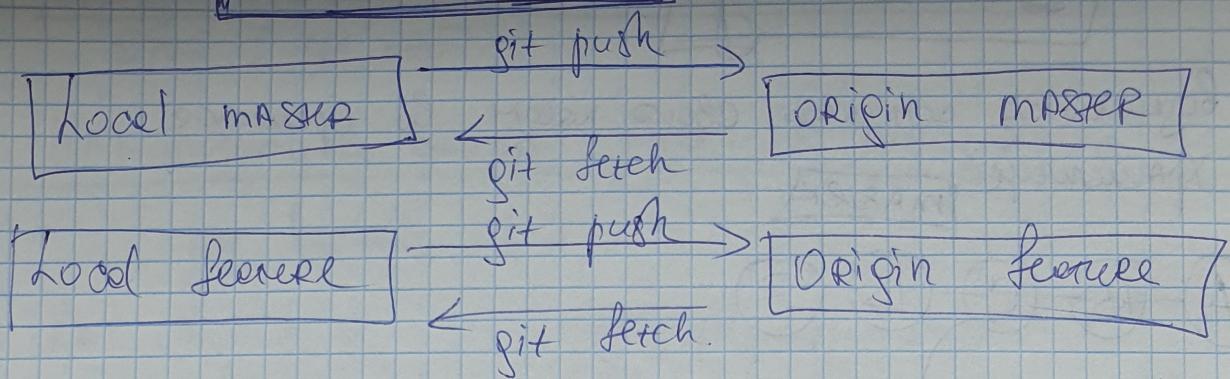
git stash apply (оставить в stash)

Remove

→ удалить because we're good now.

git stash drop (clear)

# Remotes



git remote -v //смотрят с каким origin  
(если у них есть падение)  
//использовать рекурсию

git remote remove origin

// избавляемся от него добавим new

git remote -v

// избавляемся от него

git remote add origin git@github.com:...

// избавляемся от него в ed

git remote -v

// заменить

git push // запишем

Add

git remote add <name> <url>

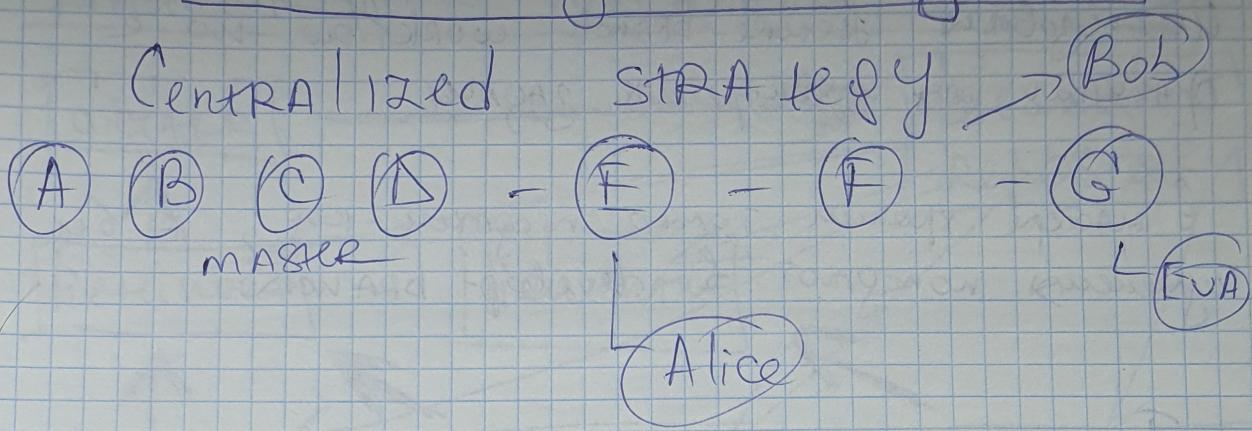
git remote add origin git@github.com:username

View

git remote -v

git remote show <name>

# Branching STRATEGIES



Беј подготвот б огнов branche.

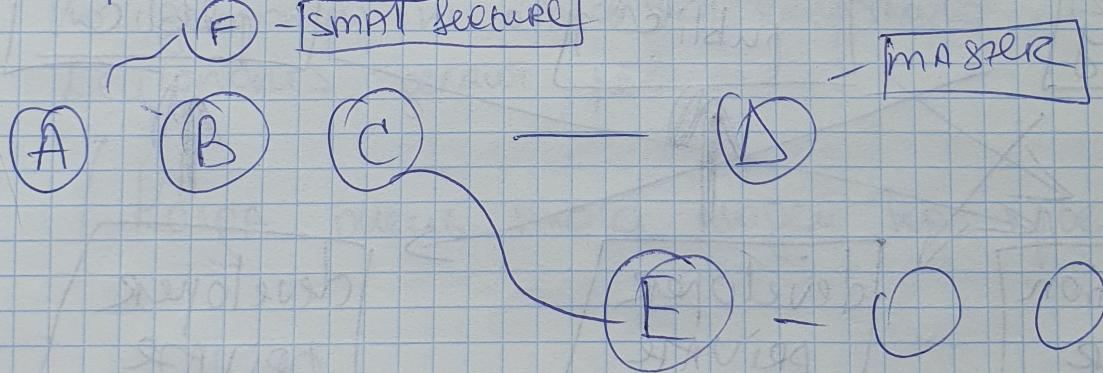
Беј комист и пуштят б мастер.

Беј конфигурираат б мастер

Сми комист мако заси Git

Сми комист подготвят на XAKATONE.

## Feature - branch workflow



согрем бачи

генерен недор

мержим б античка оптина (MASTER)

LARGE  
feature

## Gitflow

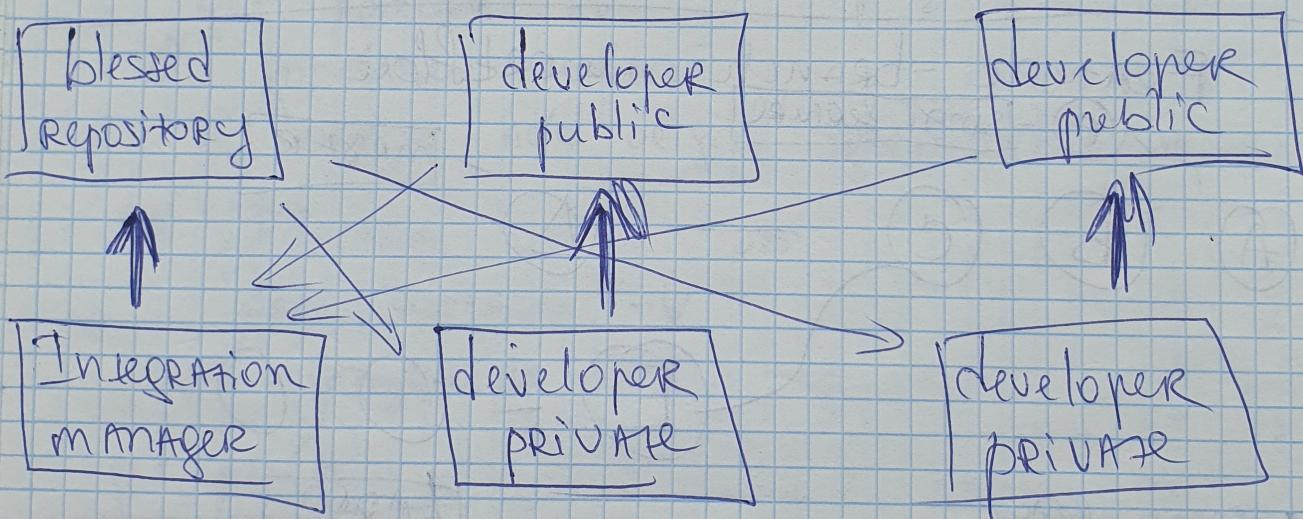
Часто используемое feature-branch workflow на Git  
Для решения конкретных задач

в начале хранят только прототипы, а ближе к окончанию наносят в develop-branch

## Integration manager workflow

использовался в больших проектах и игр.  
составлял пакеты для распространения.

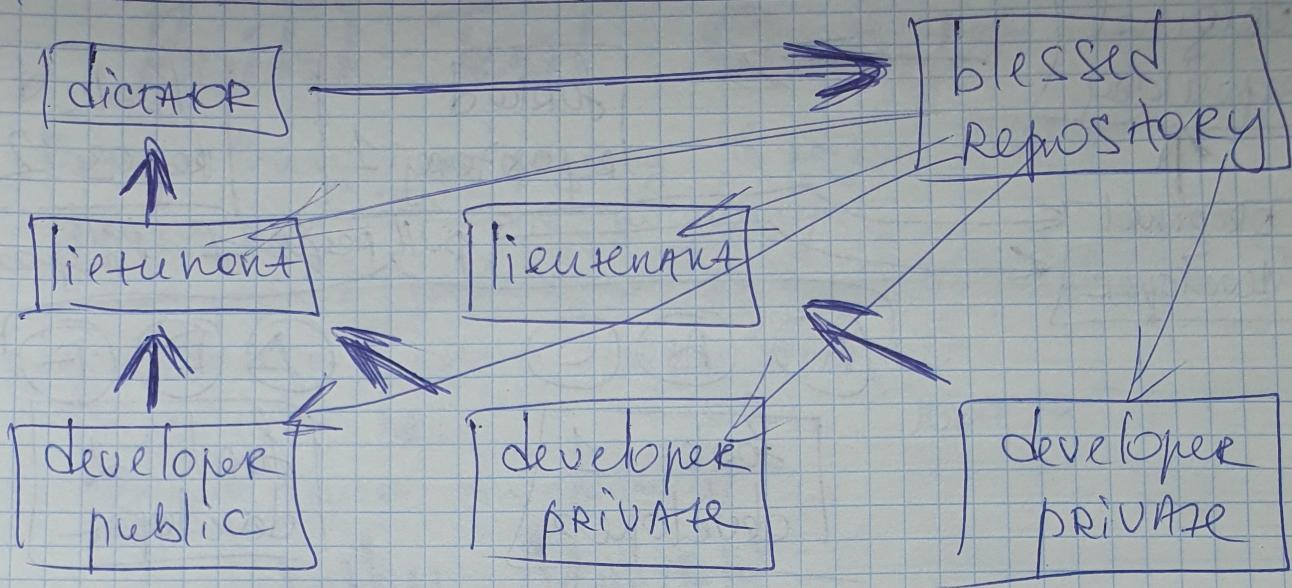
\* Использована не концепция пакетов Remote



Developer / нынешний в blessed repository, а  
pushat в developer public.

[Integration manager] нынешний в гитбендер  
создает, если код хороший вывести в blessed  
repository]

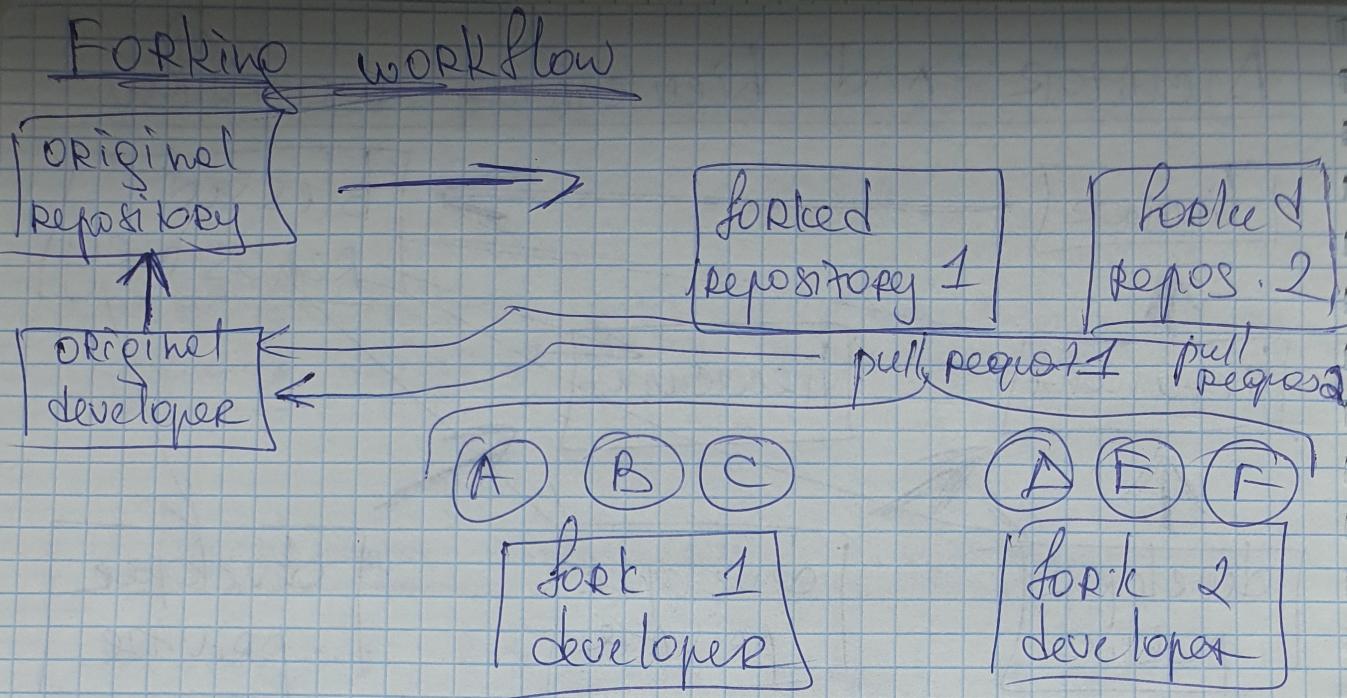
## Dictator and lieutenants workflow



Также Integration managerов несколько.  
Соб Blessed Repository, в который имеет право  
пушить один человек (это генератор).

Но генераторы не отвечают за версию Dictator.  
Lieutenants делают ребус

(Когда меня много людей не хватает)



Pewnej poprzedni rysunek ułożyłeś w połyne zgodnie z zamierzeniem.

Rzucić:

Pro Git (we call it Git)

commit 31e886

commit ea1e5e3c