git good

Amy Dickens
@dickensa

(Hacknotts 2015)


Dundee
Edinburgh
Glasgow
United Kingdom
NORTHERN IRELAND
Isle of Man
Dublin
Galway
Ireland
Limerick
Cork
Liverpool
Manchester
ENGLAND
Cambridge
WALES
Oxford
London
Bristol
Cardiff
The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA
UoN


HACK
>SOC
Outreach &
Inclusivity Secretary
hack
notts
Organising Team
UoN Hackathon
wit
Lead Organiser
Tech Conference

🎓 2015 – PhD Computer Science (Mixed Reality Laboratory)

🚩 2016 – GitHub Campus Expert

I  community

Version Control with Git

Why Version Control?

- **History:** remembering what changed over time, who worked on what and when.
- **Backup:** keep copies of your code in multiple places to avoid losing your work.
- **Collaboration:** Work concurrently on the same code with other people.

What will we cover?

 **Gitting Started:** git repositories, staging, committing

 **Gitting Good:** branching, jumping around, merging

 **Gitting Better:** rolling back, using GitHub

My first .git

repository creation, staging, committing

What is a repository?

- A folder containing your code… **but:**
- Has a "**.git**" directory which stores the history of your project
- turn any folder into a **git repository** with `git init`

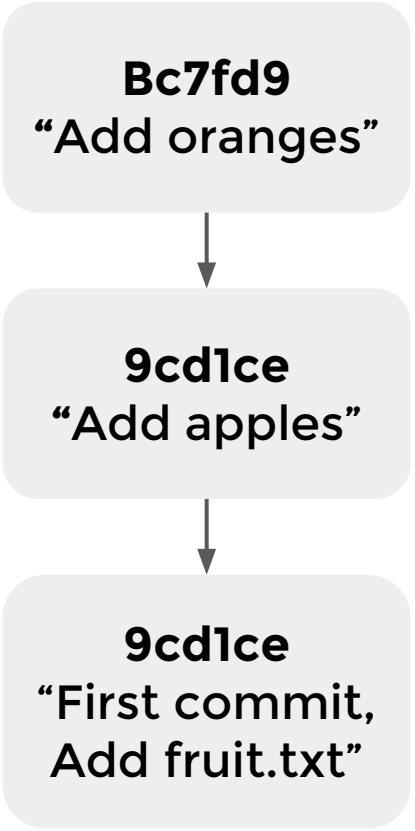
Commits

- **Snapshots** of the **contents (e.g. code)** in your **repository**

- Created with

```
git add .
git commit -m "<commit description>"
```

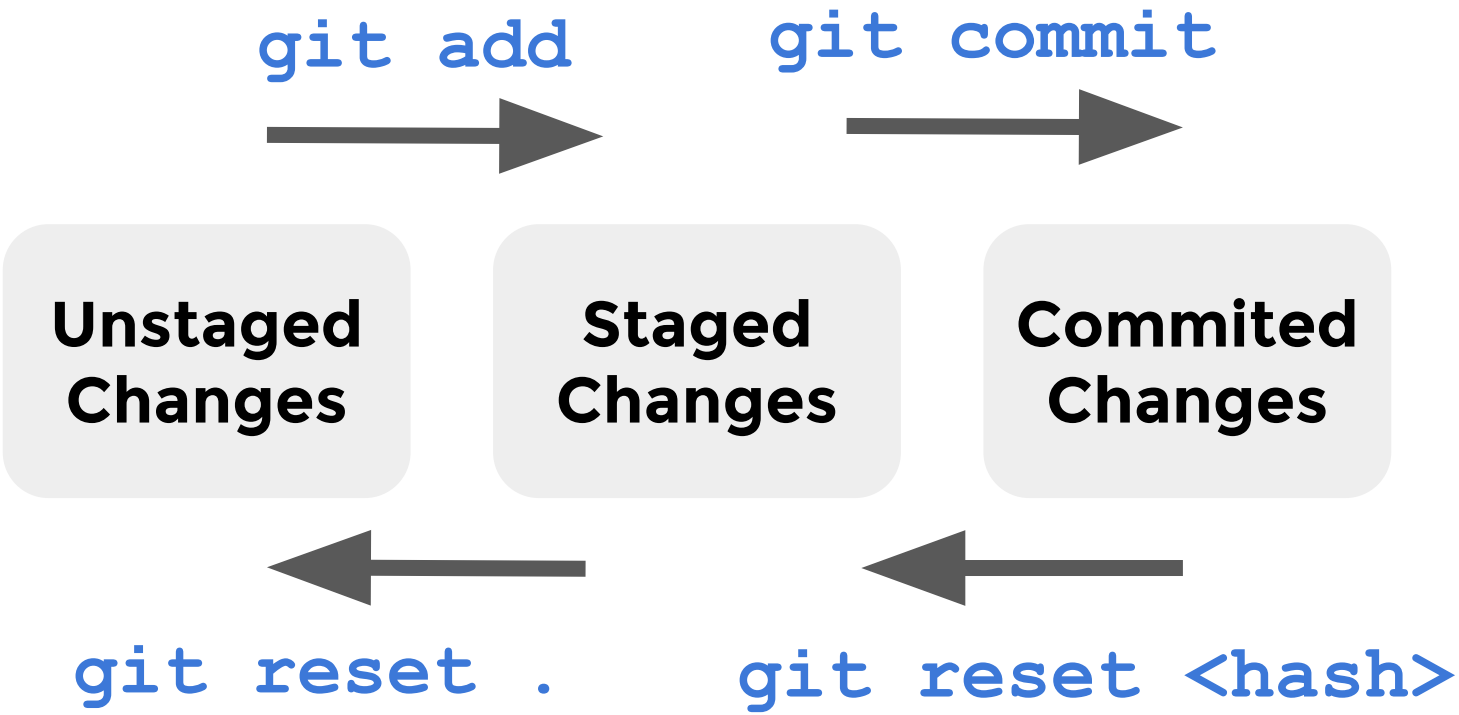
More on Commits



- Commits form a **linked list** structure (for now)

- Run `git log` to see every commit in the repository

Staging



(see what's staged: `git status`)

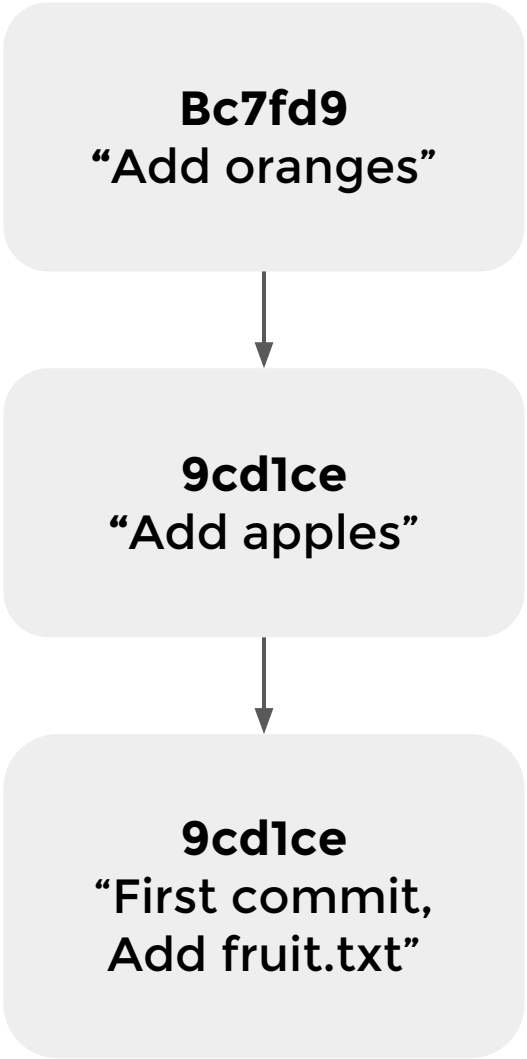
Baby Git: Recap

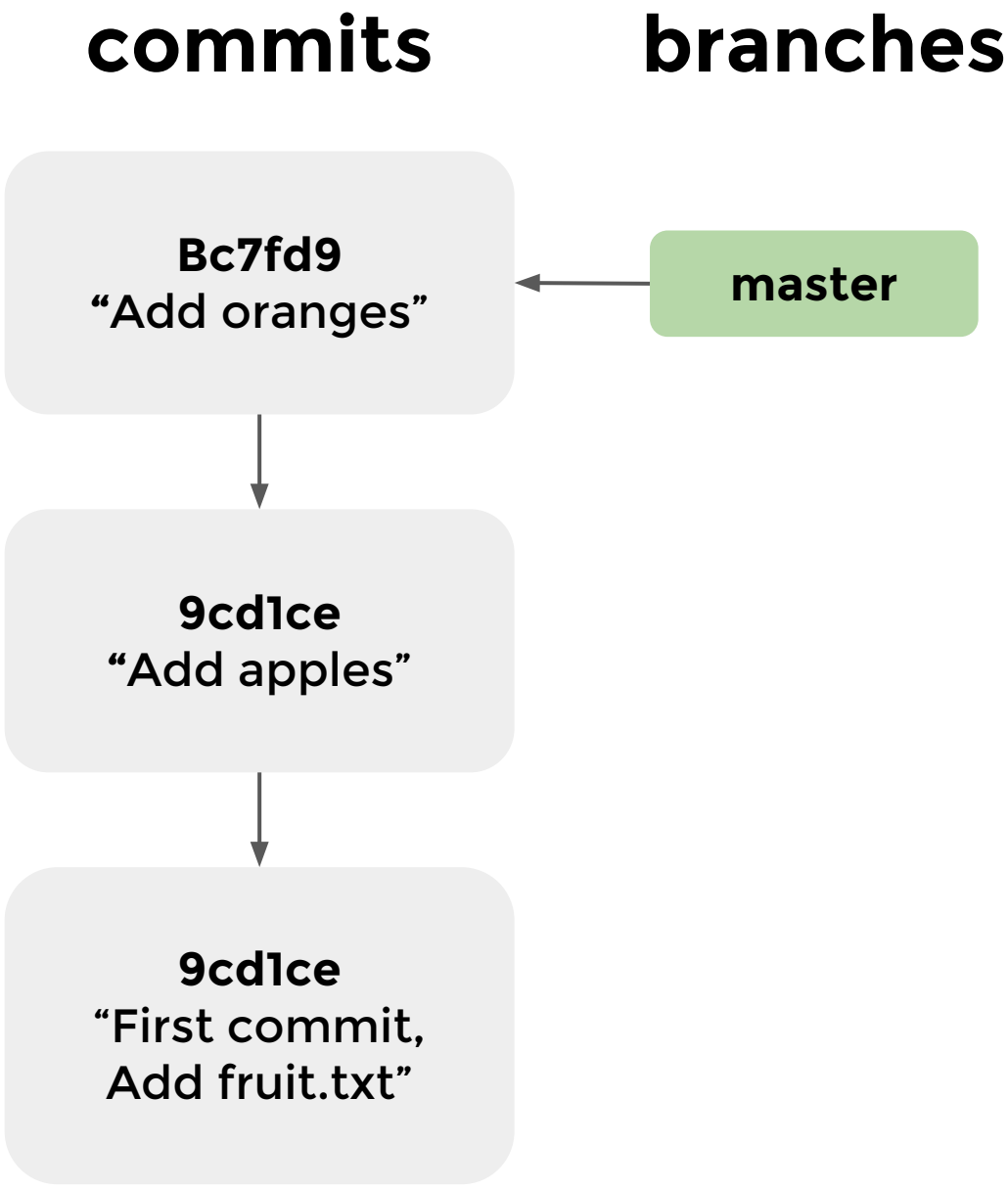
- **git init** - turn any folder into a super smart git folder
- **git add** - stage changes to files
- **git commit** - create a snapshot of staged changes
- **git reset** - unstage, uncommit
- **git log (or ll)** - see all commits
- **git status** - see what is staged

git **good**

branching, context switching, merging

commits

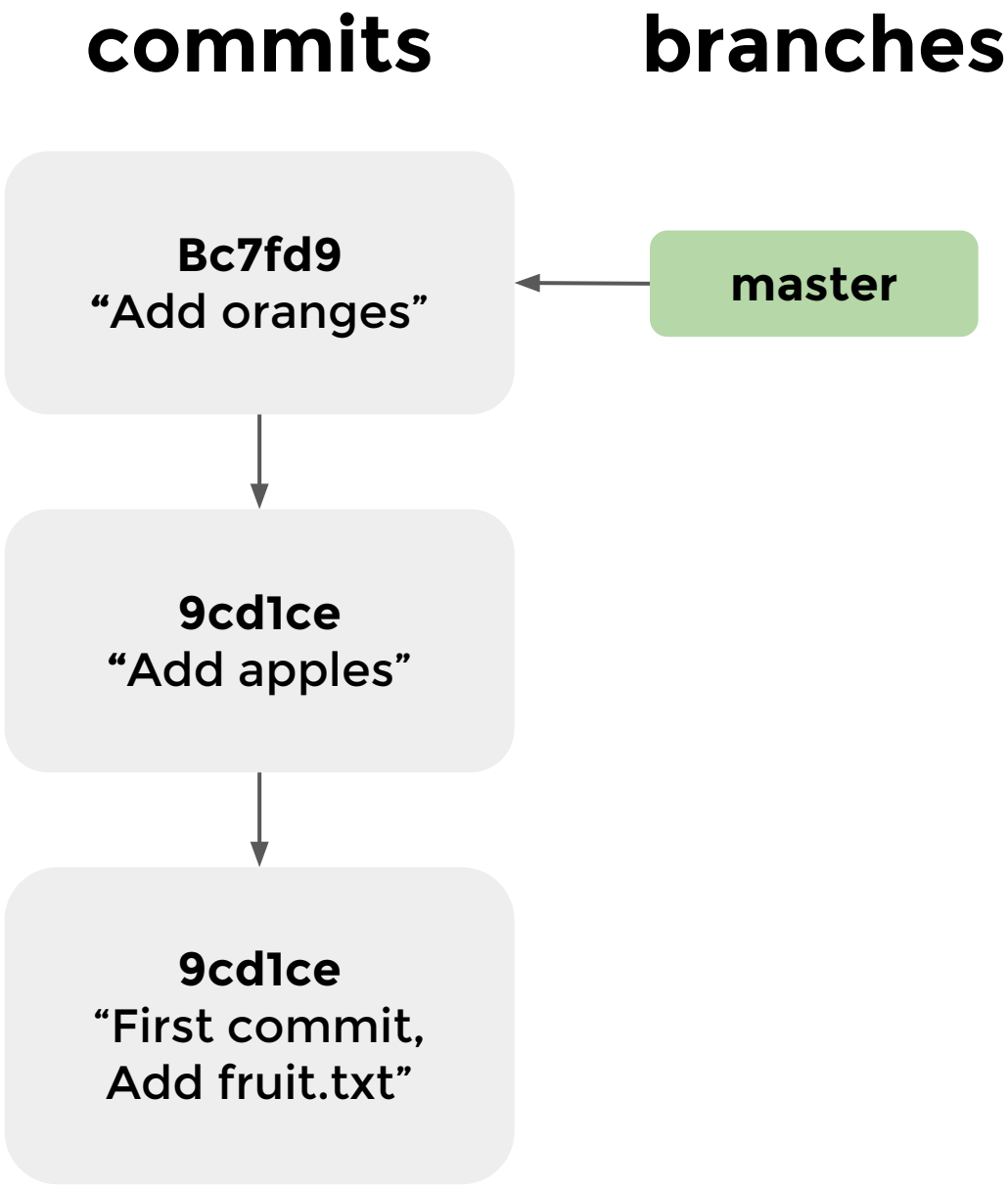

Bc7fd9
“Add oranges”
9cd1ce
“Add apples”
9cd1ce
“First commit,
Add fruit.txt”

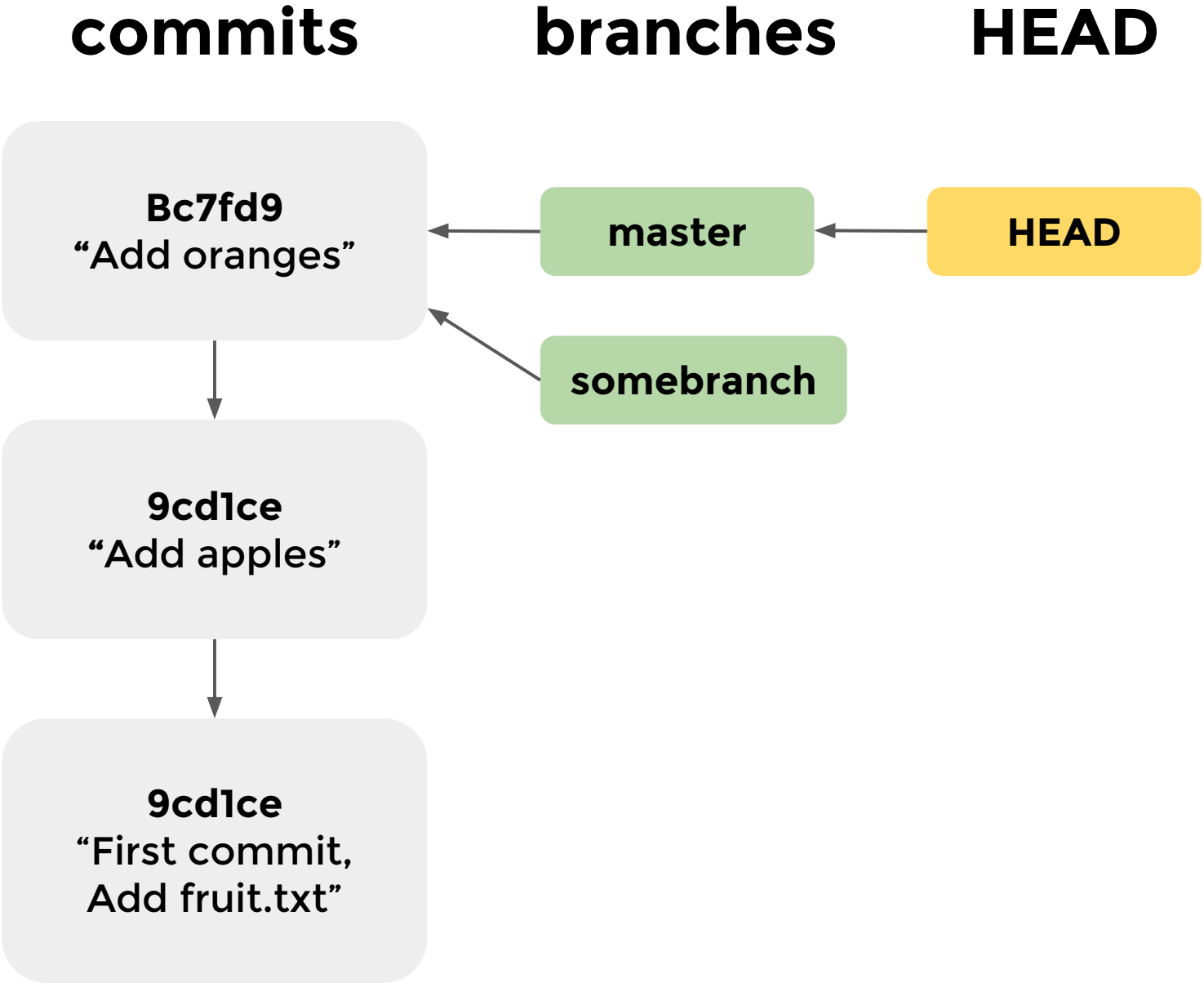

commits
branches
Bc7fd9
“Add oranges”
master
9cd1ce
“Add apples”
9cd1ce
“First commit,
Add fruit.txt”

Working with branches

`git branch` - see the names of all availables branches

`git branch branchname` - create a new branch called “branchname”


commits
branches
Bc7fd9
“Add oranges”
master
9cd1ce
“Add apples”
9cd1ce
“First commit,
Add fruit.txt”


commits
branches
HEAD
Bc7fd9
“Add oranges”
master
HEAD
somebranch
9cd1ce
“Add apples”
9cd1ce
“First commit,
Add fruit.txt”

Working with HEAD

`git checkout branchname` - move HEAD to point to a branch.

In simple terms: Make my working repository look like this branch

Merging - combining commits

`git merge branchname` - create a new commit, combining the latest commit pointed to by HEAD, with the latest commit pointed to by the other branch.

Gitting Good: Recap

- **`git branch`** - list all branches
- **`git branch name`** - create a branch
- **`git checkout`** - jump to a branch
- **`git merge --no-ff branchname`**
  - merge another branch into current

git **better**

rolling back, using GitHub

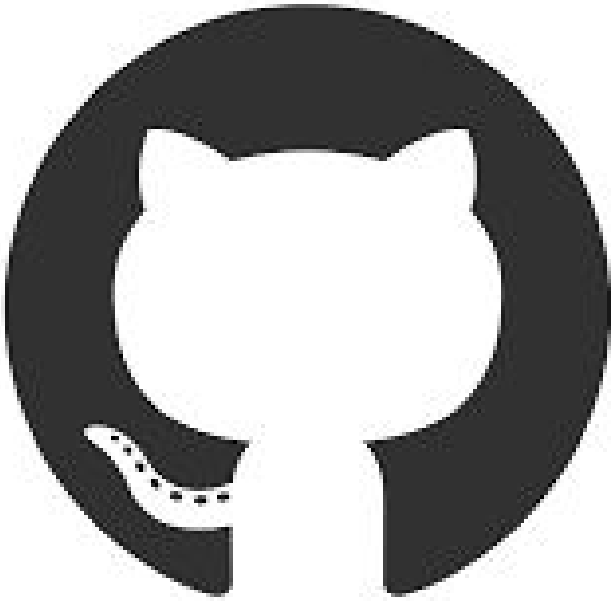
Rolling Back - reflog & reset

`git reflog` - view the history of moves the current branch pointer made.

`git reset --HARD <hash>` - move the branch pointer to a particular hash

`git reset --HARD HEAD@{n}` - move the pointer n steps back.

Backup to GitHub



Downloading/Uploading Repos

`git clone <url>` - download a copy of a git repository from a website.

`git pull` - merge the current branch with the most up-to-date version of this branch on the remote server.

`git push` - tell the remote server to to merge your latest version of the branch into its own.

Recap

- **Baby steps:** repository creation, staging, committing
- **Basic git:** creating branches, jumping between branches, merging
- **Intermediate Git:** rolling back with the reflog, pushing/pulling from remote servers

Get your DojoCat

Go forth & git on!

We are here to help:
Amy @dickensa
Nick @nicktikhonov