

<epam>

# VCS concept

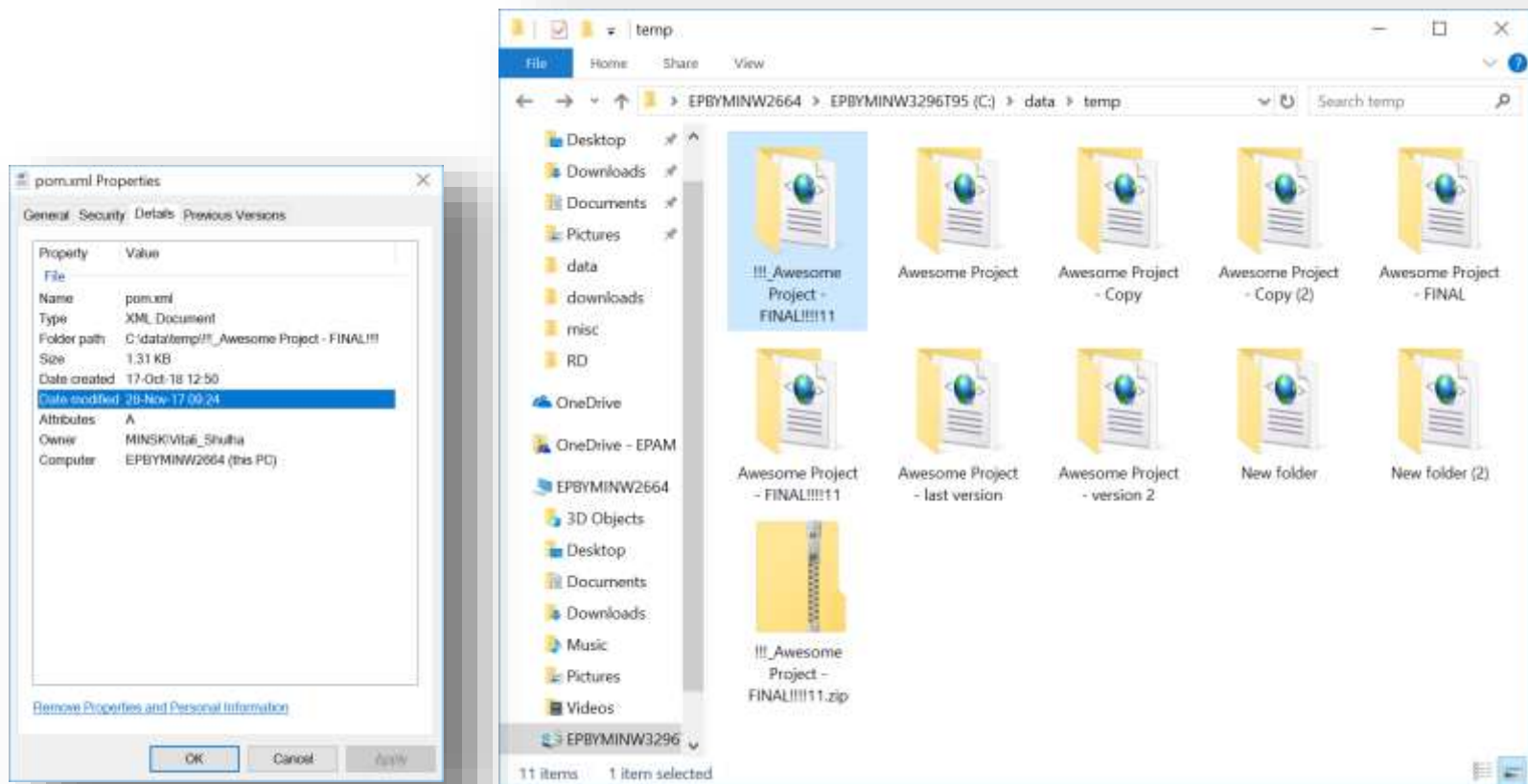
Version Control with Git. DevTestOps training.



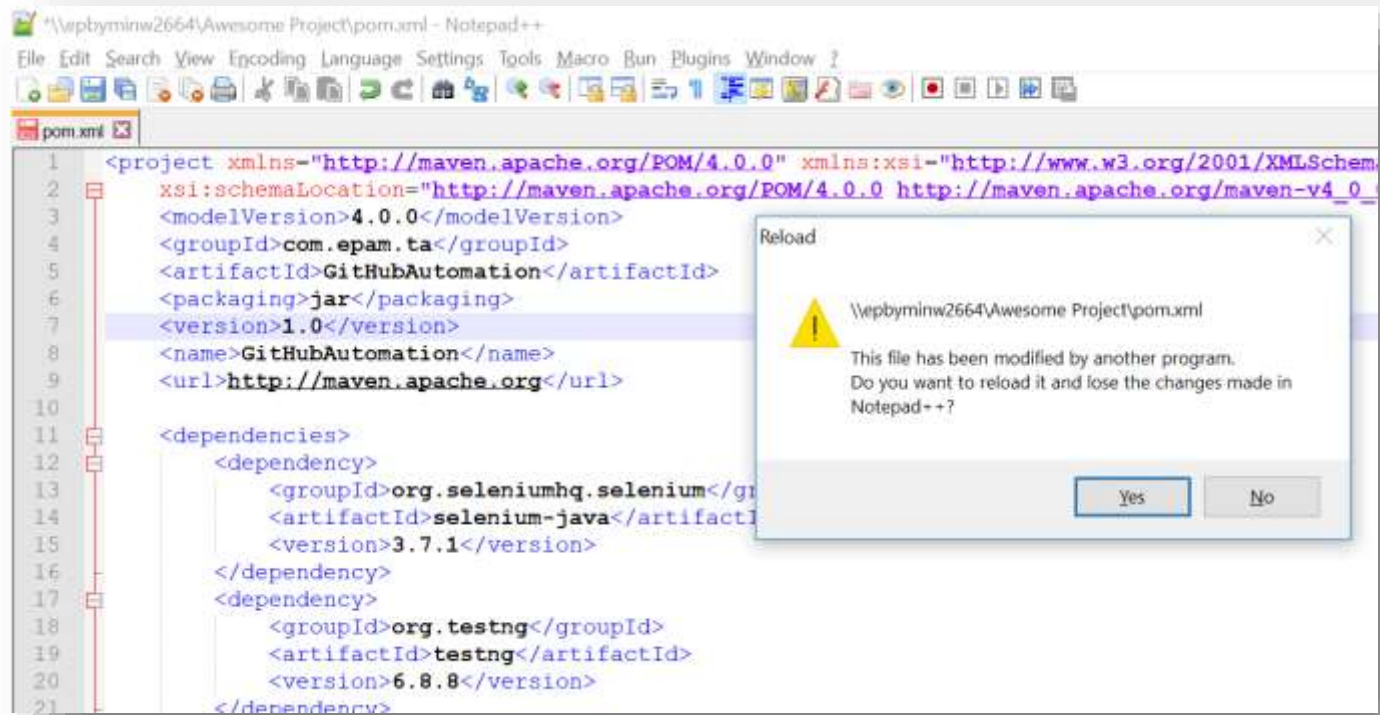
**TRAINING**  
CENTER

— <epam> —

# Standalone work. Level 1 - beginner



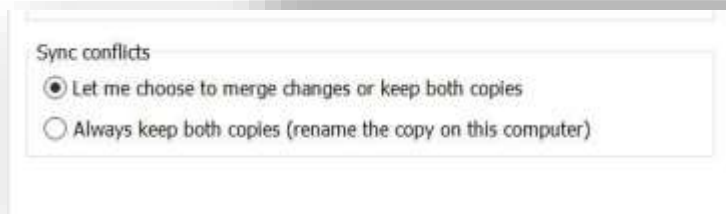
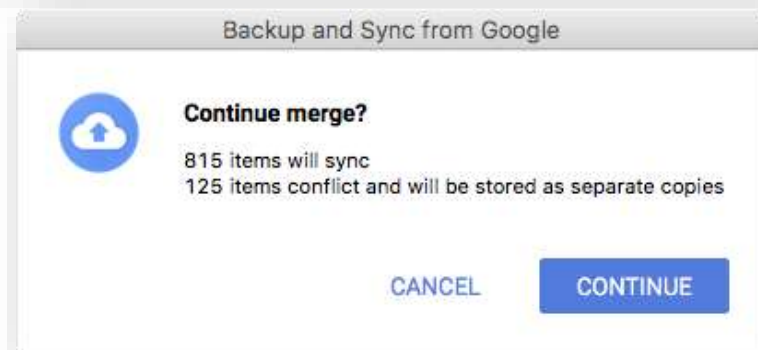
## Team work. Level 2 – network share



## Standalone/Team work. Level 3 - cloud



Name	Date Modified
 Sample File	 Today at 10:54
 Sample File (hanz-mbp's conflicted copy 2018-10-12)	 Today at 10:54



# VCS goals

---

1 **BACKUP AND RESTORE**

2 **SYNCHRONIZATION**

3 **UNDO**

4 **TRACK CHANGES AND OWNERSHIP**

5 **SANDBOXING**

6 **BRANCHING**

<epam>

# Version control types

Version Control with Git. DevTestOps training.

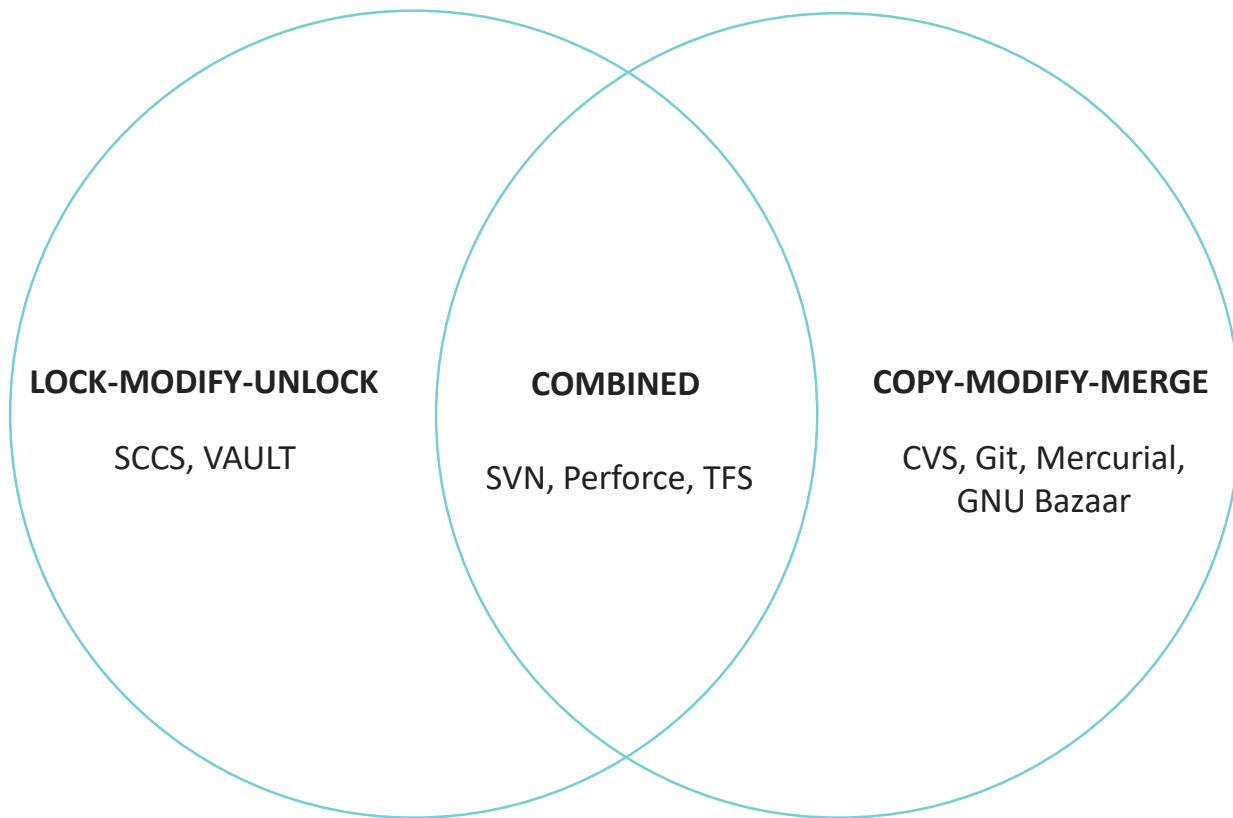


**TRAINING**  
CENTER

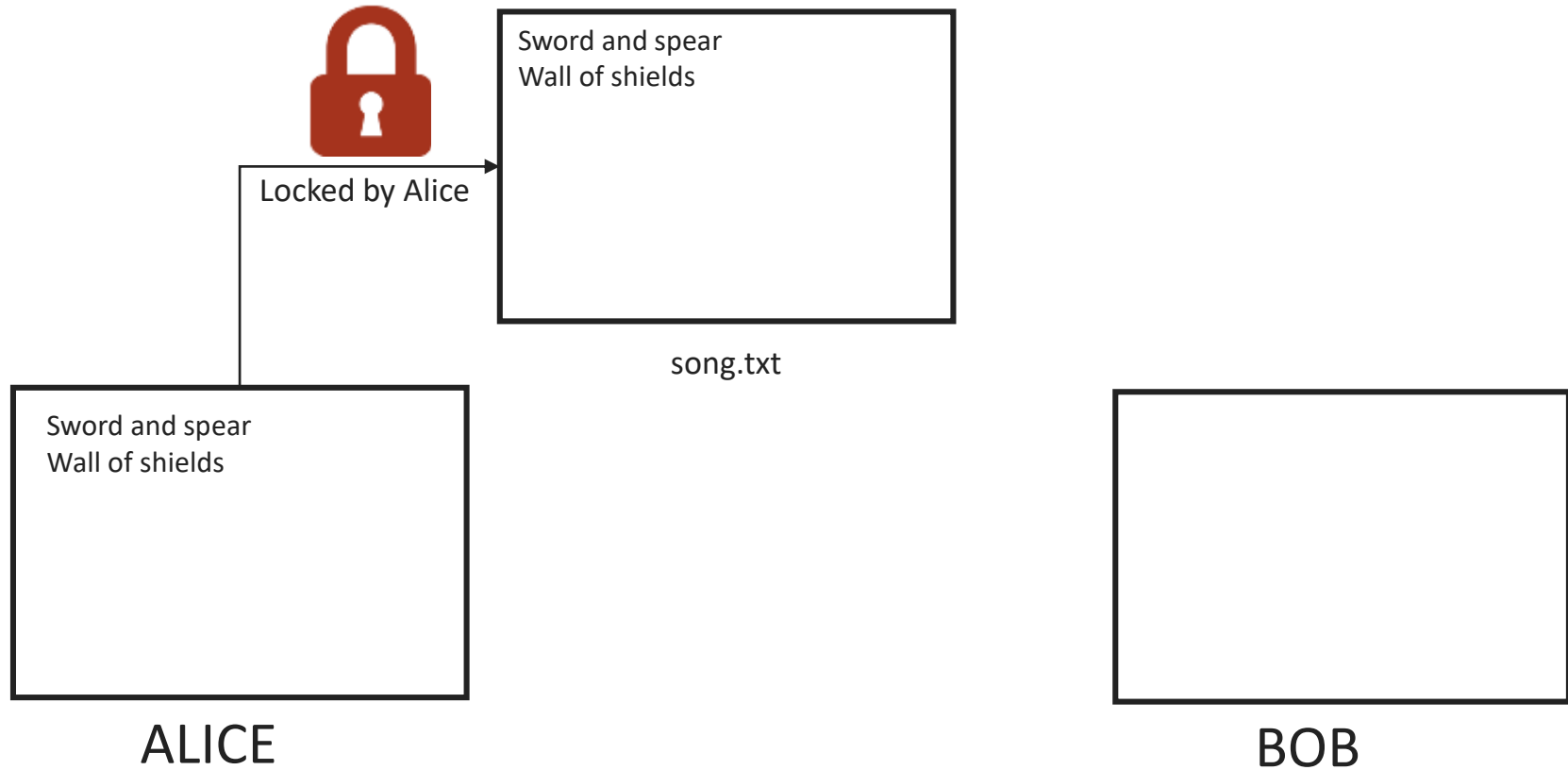
— <epam> —

# VCS types

---

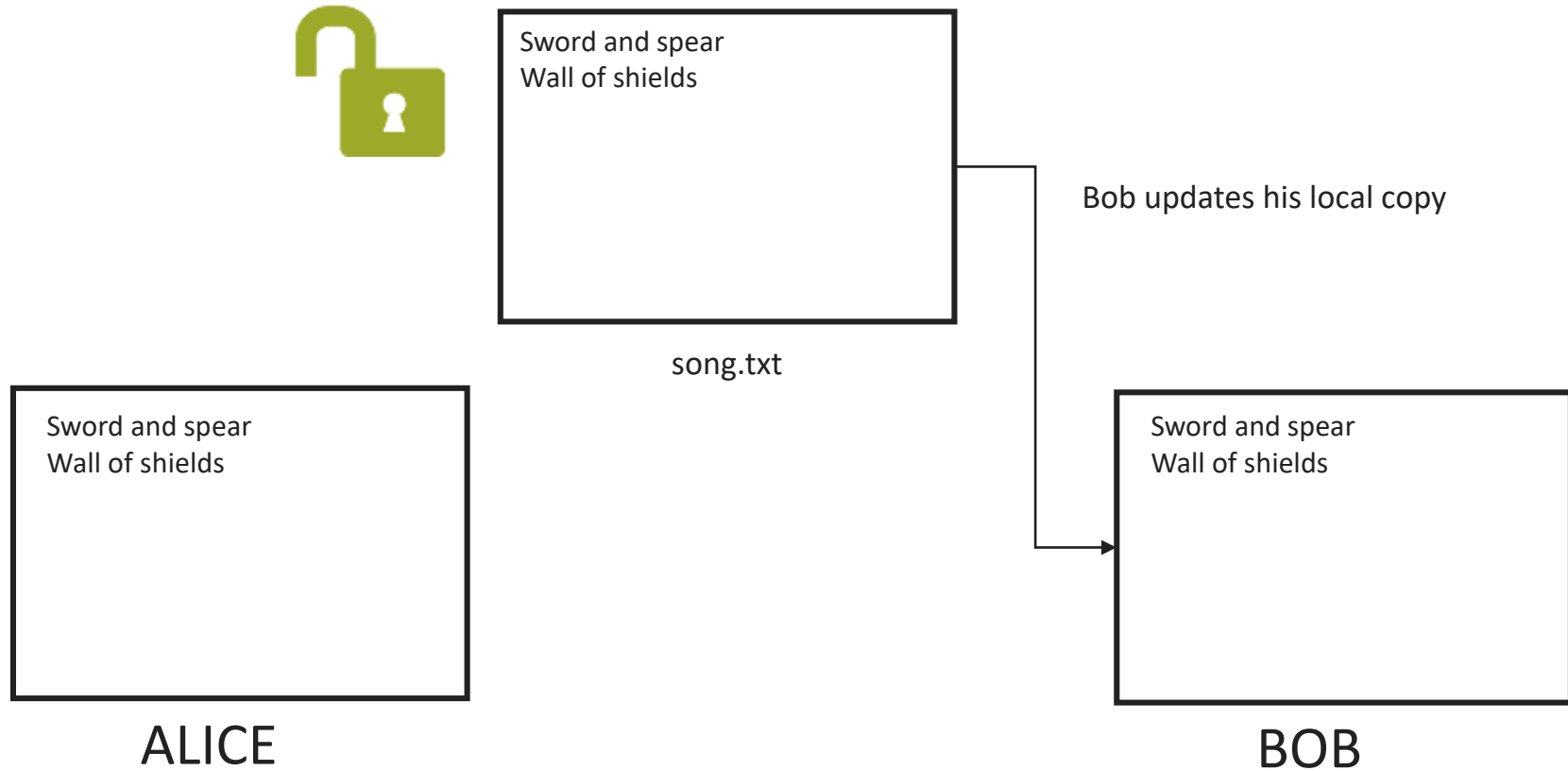


# Lock-modify-unlock strategy

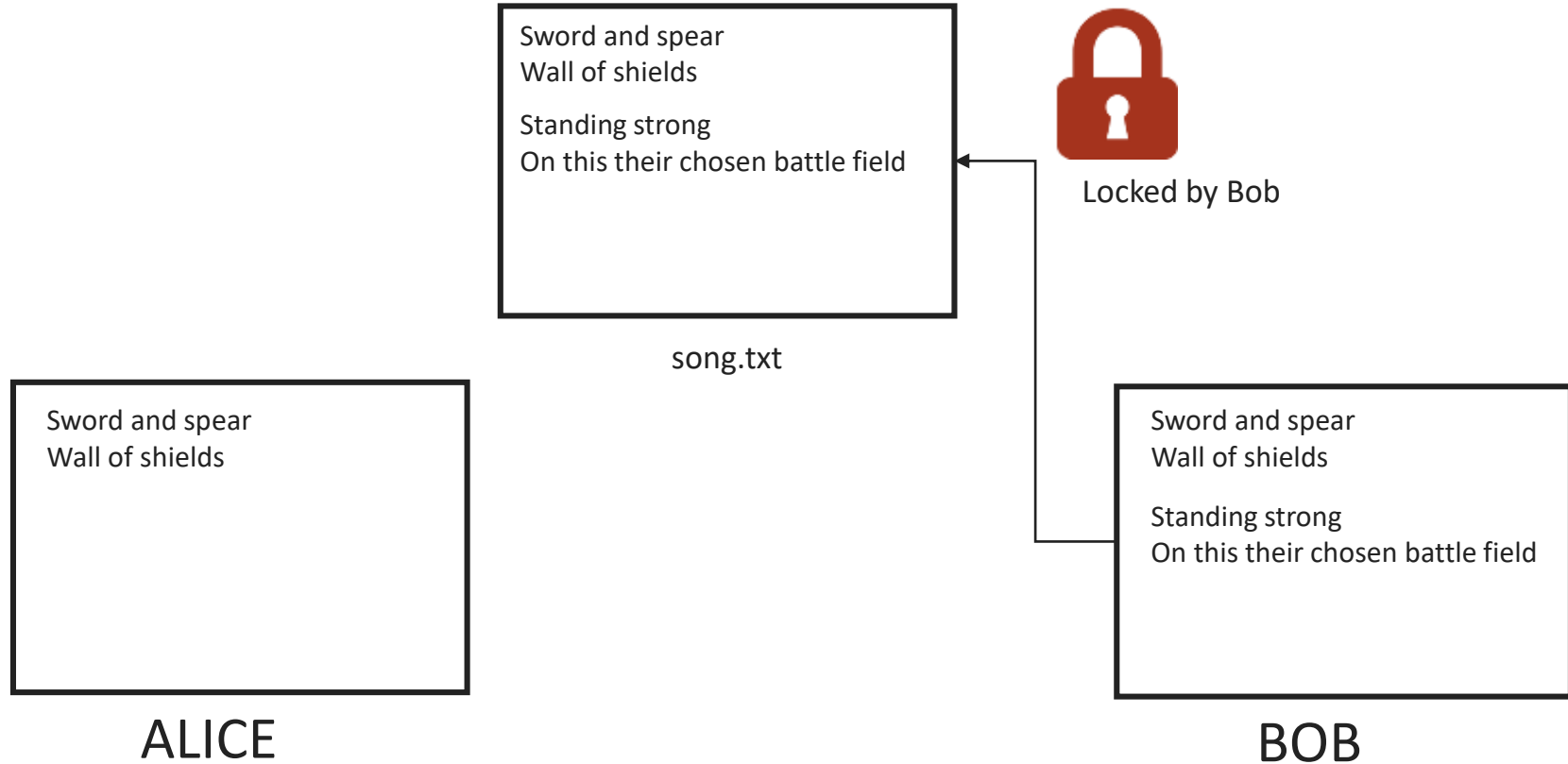




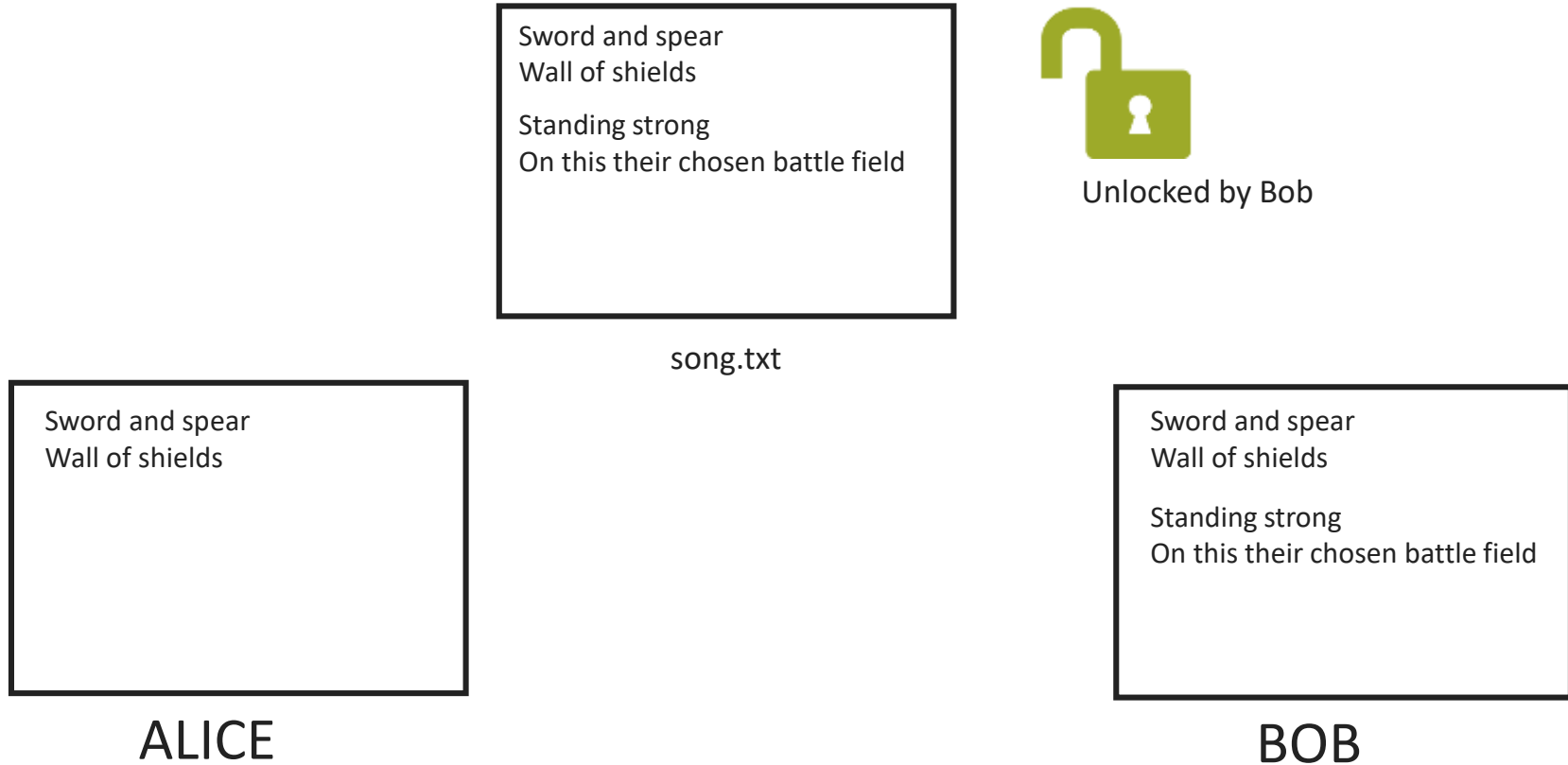
# Lock-modify-unlock strategy



# Lock-modify-unlock strategy

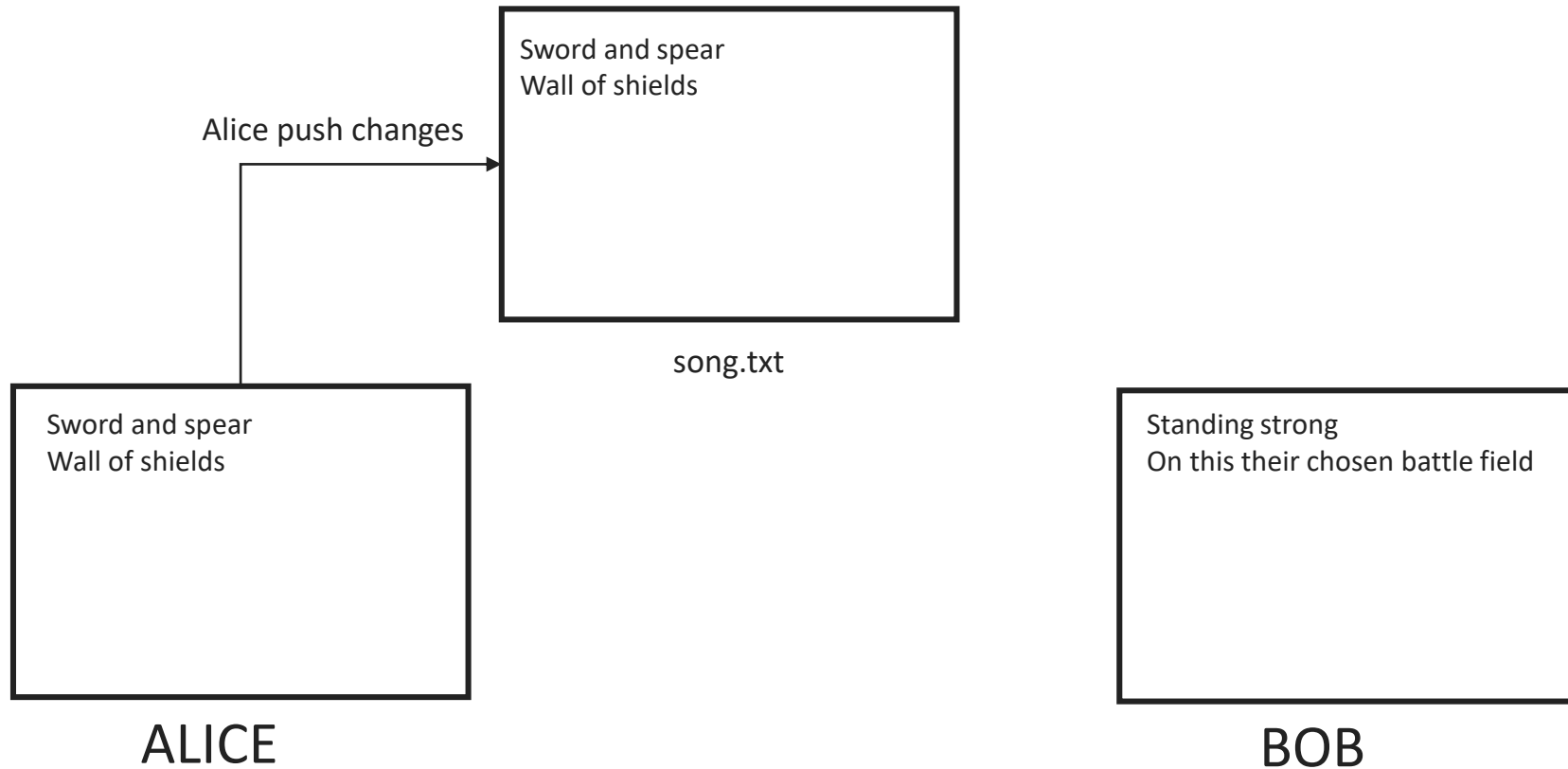


# Lock-modify-unlock strategy

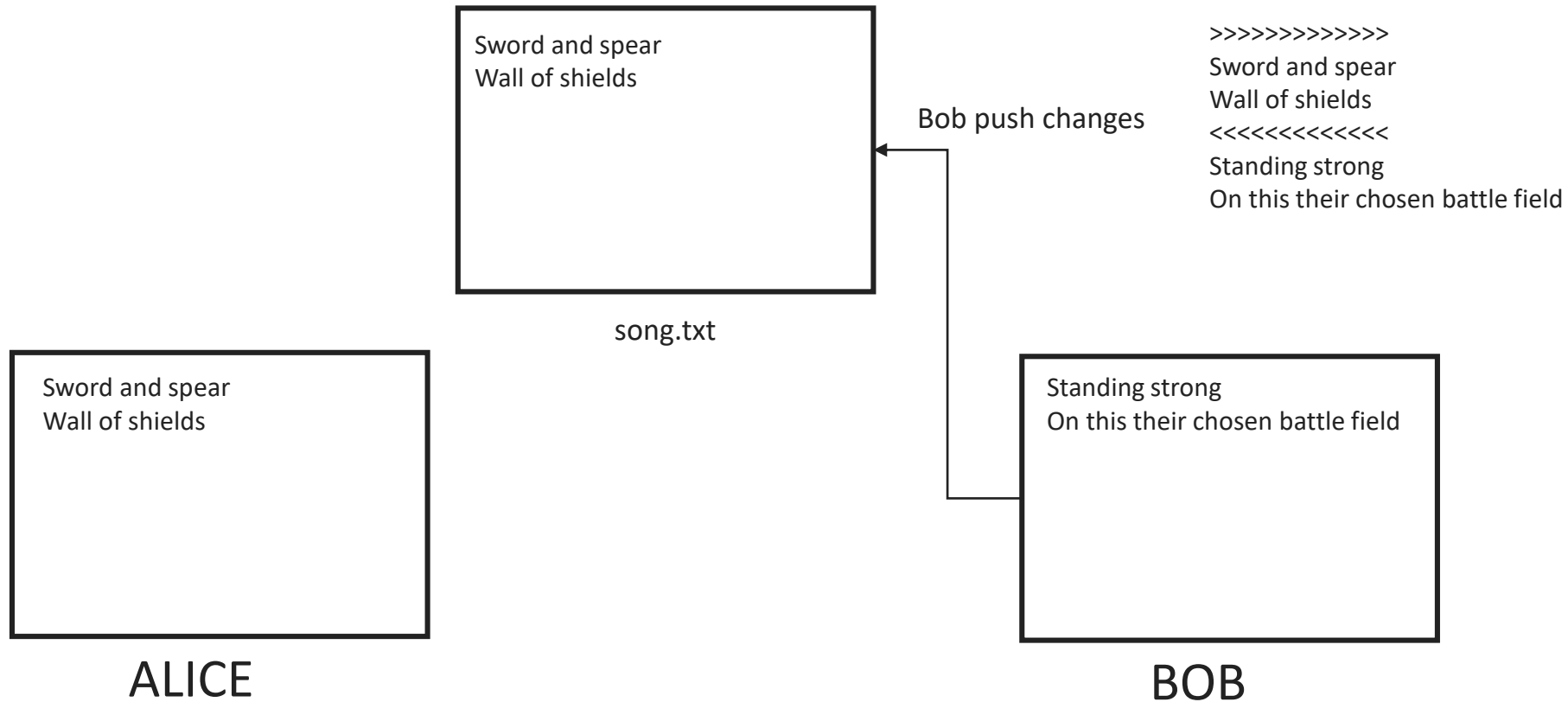


# Copy-modify-merge strategy

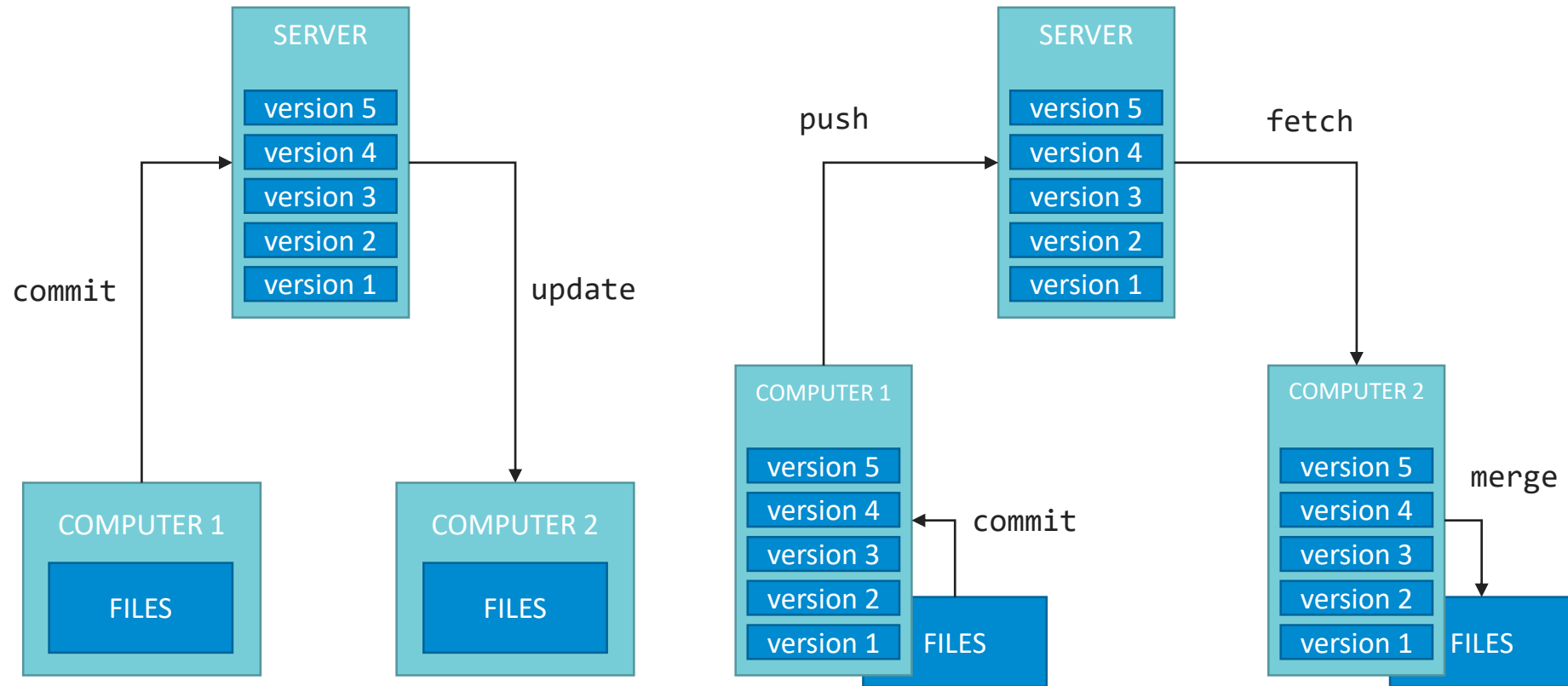
---



## Copy-modify-merge strategy



# Centralized vs Distributed



<epam>

# Why Git

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

# Why Git



- Git is released under the [GNU General Public License version 2.0](#), which is an [open source license](#). The Git project chose to use GPLv2 to guarantee your freedom to share and change free software - to make sure the software is free for all its users.

## Version Control

All Respondents

Professional Developers



(ALL) respondents select all that apply

Google

facebook

Microsoft

twitter

LinkedIn

NETFLIX



PostgreSQL





# Git benefits



Fourth batch for 2.20 [a488ab5](#)

gitster committed a day ago

Merge branch 'sf/complete-stash-list' [47e1fb1](#)

gitster committed a day ago

Merge branch 'mw/doc-typofixes' [65f1b1b](#)

gitster committed a day ago

Merge branch 'js/mingw-wants-vista-or-above' [29cce95](#)

gitster committed a day ago

Merge branch 'rs/sequencer-oldset-insert-avoids-dups' [488e2e8](#)

gitster committed a day ago

<epam>

# Download, install, configure

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

# Download, install, configure

---

## DOWNLOAD & INSTALL

- Download binary from here: <http://git-scm.com/downloads>
- Follow the steps using the default options



## CONFIGURE

- Generate SSH key pair
- Send public key to repository owner or upload to your profile
- Configure username and email

```
ssh-keygen -t rsa -C "vitali_shulha@epam.com"
```

```
git config --global user.name "Vitali Shulha"
```

```
git config --global user.email "vitali_shulha@epam.com"
```



# Create a github repo and clone it

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Create github repository and clone it

- Simple as it can be
- Don't forget to select "Initialize this repository with a README"

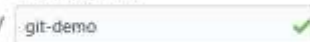
## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

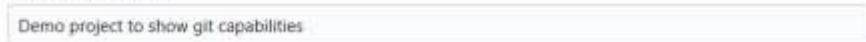


Repository name



Great repository names are short and memorable. Need inspiration? How about [urban-goggles](#).

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None**



Create repository

# Create github repository

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 vitallius / git-demo 

Great repository names are short and memorable. Need inspiration? How about [urban-goggles](#).

Description (optional)

Demo project to show git capabilities

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None** 

Create repository

vitallius / git-demo

Unwatch 1

Star 0

Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

Demo project to show git capabilities

Edit

Manage topics

1 commit

1 branch

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

vitallius initial commit

README.md

Initial commit

README.md

## git-demo

Demo project to show git capabilities

Clone with SSH

Use HTTPS

Use an SSH key and passphrase from account.

`git@github.com:vitallius/git-demo.git`

Open in Desktop

Download ZIP

# Cloning the repository

Select folder

Check content

Clone repo

Ensure it's done

Go to target dir

See master branch

See content

```
Vitali_Shulha@EPBYMINW2664 MINGW64 ~  
$ cd /c/data/temp/  
  
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp  
$ ls  
'!!!_Awesome Project - FINAL!!!!11'/' 'Awesome Project - Copy'/' 'Awesome Project - FINAL!!!!11'/' 'New folder'/'  
'!!!_Awesome Project - FINAL!!!!11.zip' 'Awesome Project - Copy (2)'/ 'Awesome Project - last version'/' 'New folder (2)'/  
'Awesome Project'/' 'Awesome Project - FINAL'/' 'Awesome Project - version 2'/'  
  
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp  
$ git clone git@github.com:vitaliuss/git-demo.git  
Cloning into 'git-demo'..  
Enter passphrase for key '/c/Users/Vitali_Shulha/.ssh/id_rsa':  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.  
  
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp  
$ ls  
'!!!_Awesome Project - FINAL!!!!11'/' 'Awesome Project - Copy'/' 'Awesome Project - FINAL!!!!11'/' git-demo/  
'!!!_Awesome Project - FINAL!!!!11.zip' 'Awesome Project - Copy (2)'/ 'Awesome Project - last version'/' 'New folder'/'  
'Awesome Project'/' 'Awesome Project - FINAL'/' 'Awesome Project - version 2'/' 'New folder (2)'/  
  
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp  
$ cd git-demo/  
  
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)  
$ ls  
README.md
```



# Commit and push

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER





# Making a first commit

Write line in text file

Write another line

Check the result

Add file to staging area

Commit the staged files

See the commit done

```
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ echo 'sword and spear' >> song.txt

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ echo 'wall of shields' >> song.txt

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ cat song.txt
sword and spear
wall of shields

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git add song.txt
warning: LF will be replaced by CRLF in song.txt.
The file will have its original line endings in your working directory.

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git commit -m "create my first song"
[master 52c1f68] create my first song
1 file changed, 2 insertions(+)
create mode 100644 song.txt

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git log
commit 52c1f6823084f558d69067857b0a9161c6c56f64
Author: Vitali Shulha <vitali_shulha@epam.com>
Date: Wed Oct 17 16:02:23 2018 +0300

    create my first song
```

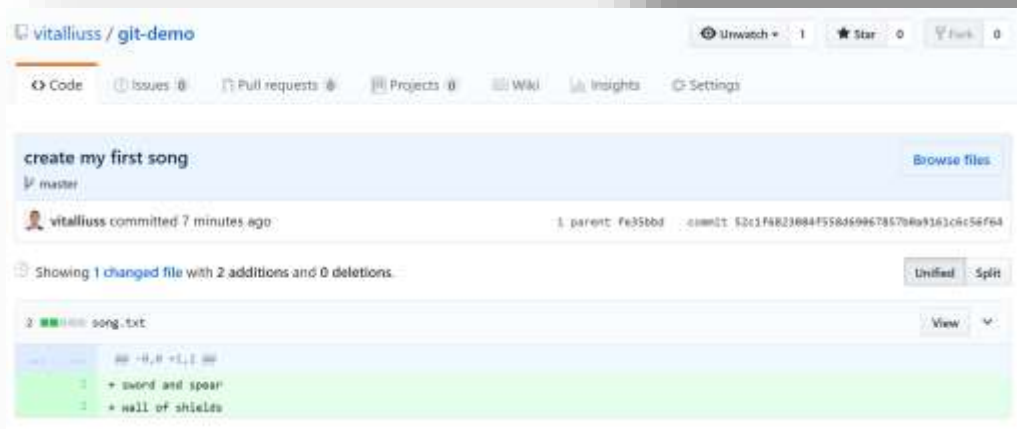
# Push in to github

Check link to github.com

Push the commits to remote

```
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git remote -v
origin  git@github.com:vitaliuss/git-demo.git (fetch)
origin  git@github.com:vitaliuss/git-demo.git (push)

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git push
Enter passphrase for key '/c/Users/Vitali_Shulha/.ssh/id_rsa':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:vitaliuss/git-demo.git
fe35bbd..52c1f68 master -> master
```





# Pull from remote

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Create commit in github and pull it

Edit file in web

Add new lines in the song

Commit changes

The screenshot shows the GitHub web interface for a repository named 'vitalliuss / git-demo'. The main content area displays the 'Edit file' page for 'song.txt' on the 'master' branch. The file content is:

```
1 sword and spear
2 wall of shields
```

Overlaid on this is the 'Commit changes' dialog box. It contains the following elements:

- A header with 'Commit changes' and a description: 'complete the first couplet with two more lines'.
- A text input field for an optional extended description.
- A dropdown menu showing the user 'vitalli\_shulha@epam.com'.
- Two radio buttons for commit options: 'Commit directly to the master branch' (selected) and 'Create a new branch for this commit and start a pull request'.
- Two buttons at the bottom: 'Commit changes' (green) and 'Cancel' (grey).

Lines from the text boxes on the left point to specific parts of the interface: 'Edit file in web' points to the 'Edit file' tab; 'Add new lines in the song' points to the file content area; and 'Commit changes' points to the 'Commit changes' button.

# Create commit in github and pull it

Ensure the commit done

Pull code from github

See the changes arrived

```
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ git pull
Enter passphrase for key '/c/Users/Vitali_Shulha/.ssh/id_rsa':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:vitaliuss/git-demo
   52c1f68..9258d31  master    -> origin/master
Updating 52c1f68..9258d31
Fast-forward
 song.txt | 2 ++
 1 file changed, 2 insertions(+)

Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ cat song.txt
sword and spear
wall of shields
standing strong
on this their chosen battle field
```

Branch: master ▾ git-demo / song.txt

 **vitaliuss** complete the first couplet with two more lines 9258d31 a minute ago

1 contributor

5 lines (4 sloc) 82 Bytes

Raw Blame History

```
1 sword and spear
2 wall of shields
3 standing strong
4 on this their chosen battle field
```



# Git GUI & gitk

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER





# Undoing changes

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Undoing changes

## Working directory

```
git checkout -- file.txt  
git checkout .  
git clean -xdf
```

## Staging area (Index)

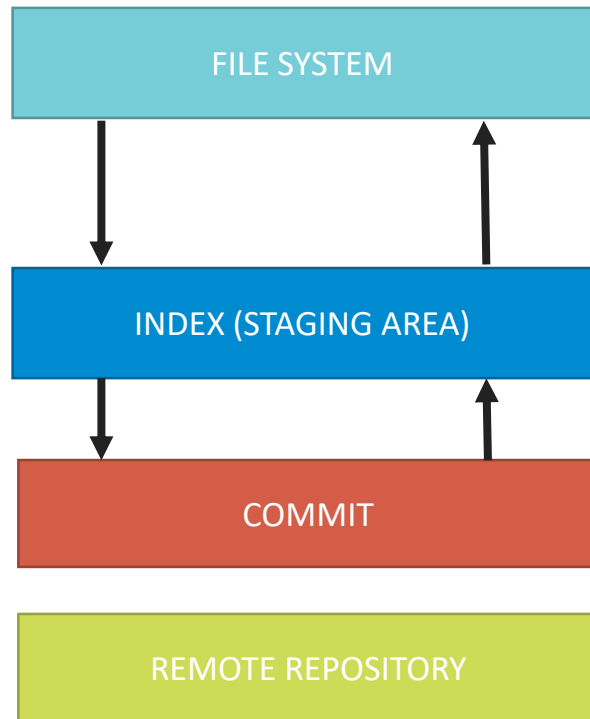
```
git reset -- file.txt
```

## Local branch

```
git reset HEAD^^ (HEAD~2)  
git commit --amend -m "commit message"
```

## Remote repository

```
git revert <sha1>
```







# Git revert

Version Control with Git. DevTestOps training.



**TRAINING**  
**CENTER**



# Undoing changes

## Working directory

```
git checkout -- file.txt  
git checkout .  
git clean -xdf
```

## Staging area (Index)

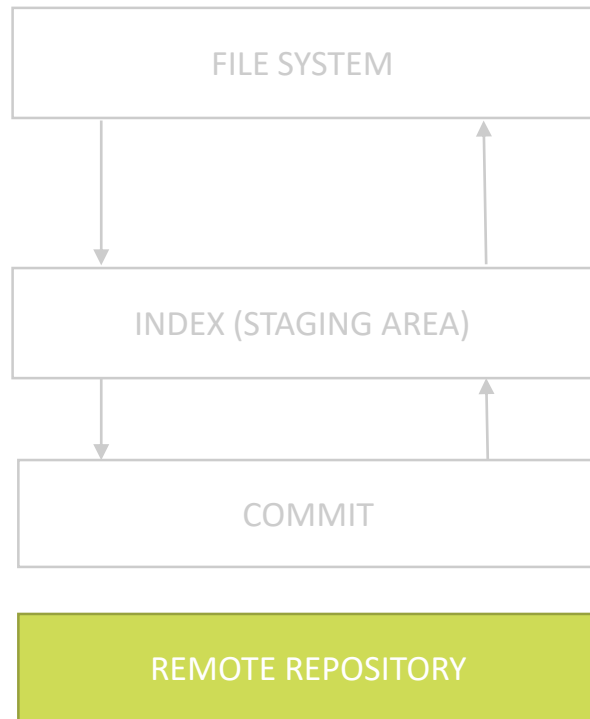
```
git reset -- file.txt
```

## Local branch

```
git reset HEAD^^ (HEAD~2)  
git commit --amend -m "commit message"
```

## Remote repository

```
git revert <sha1>
```





# Git reset

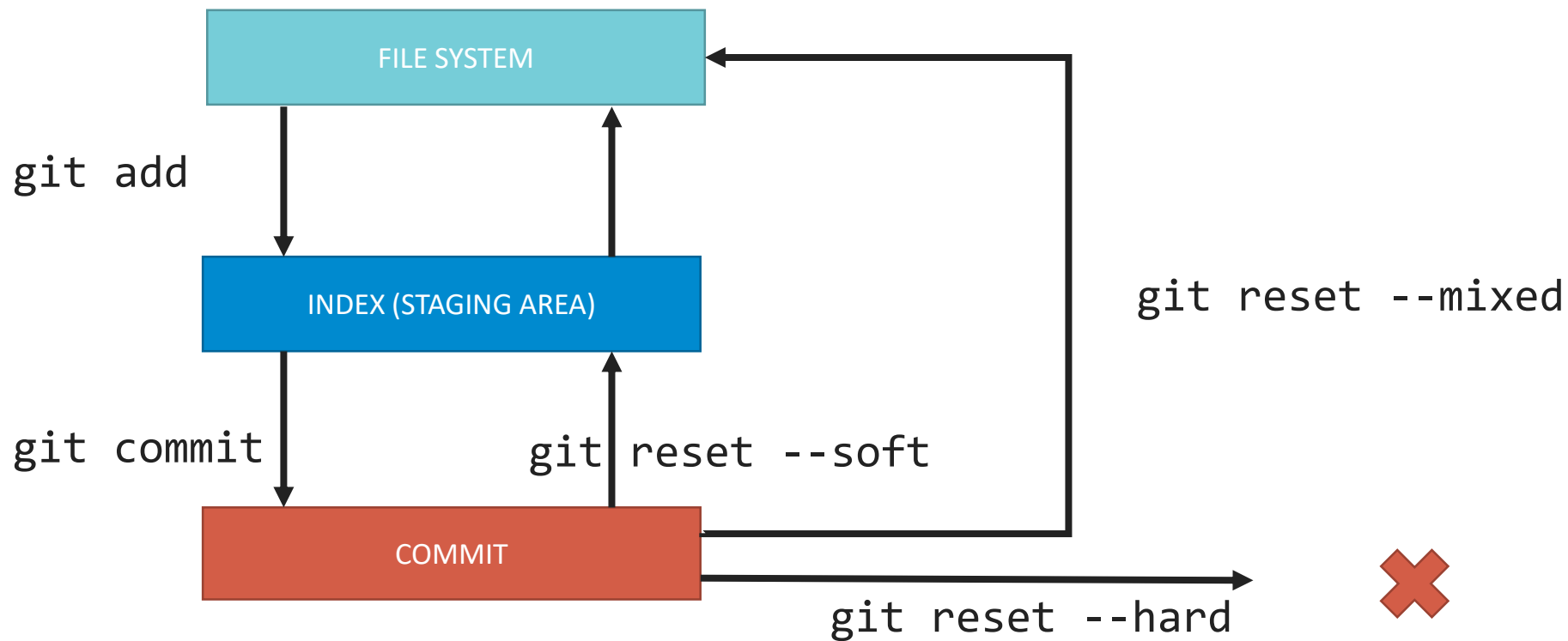
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Git reset



<epam>

# .gitignore

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

# .gitignore

---

```
# 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

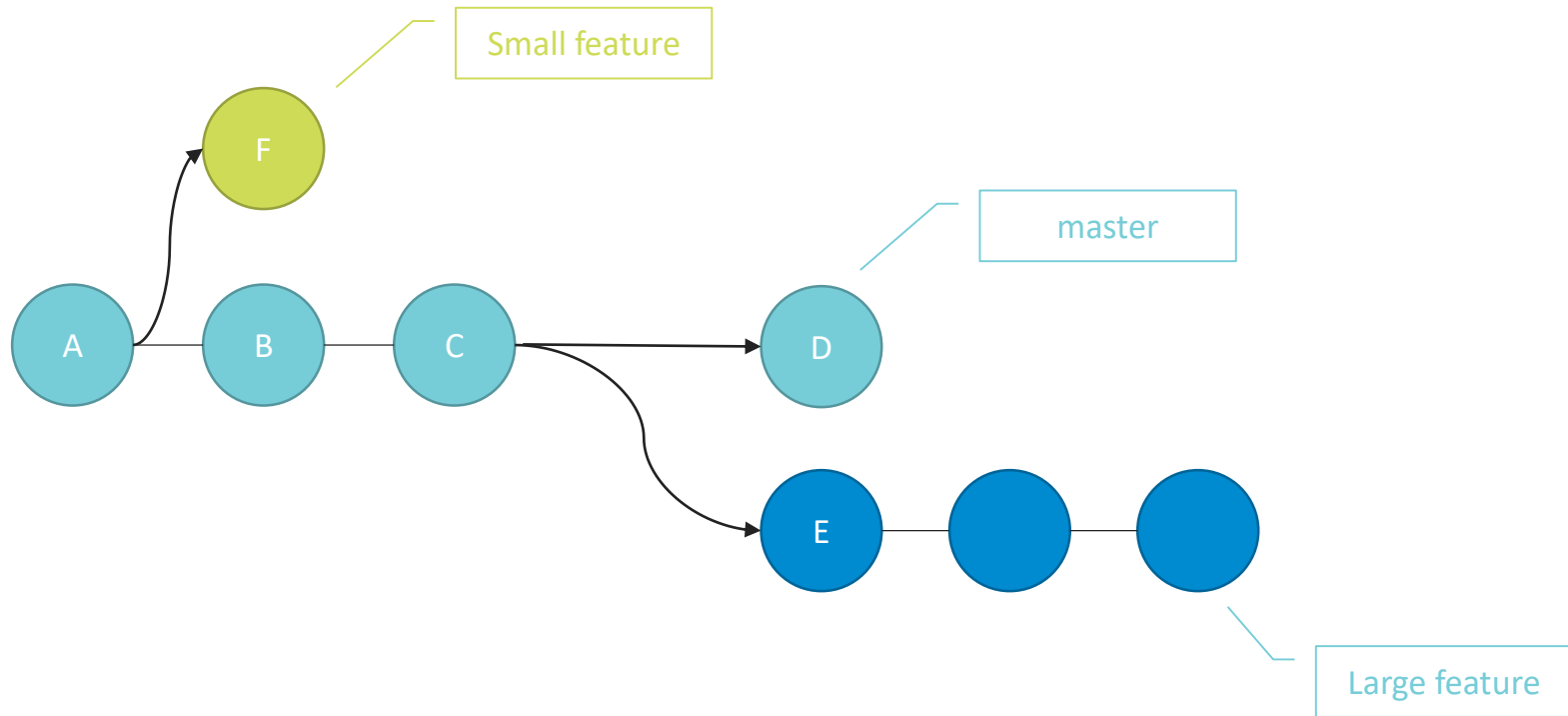
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

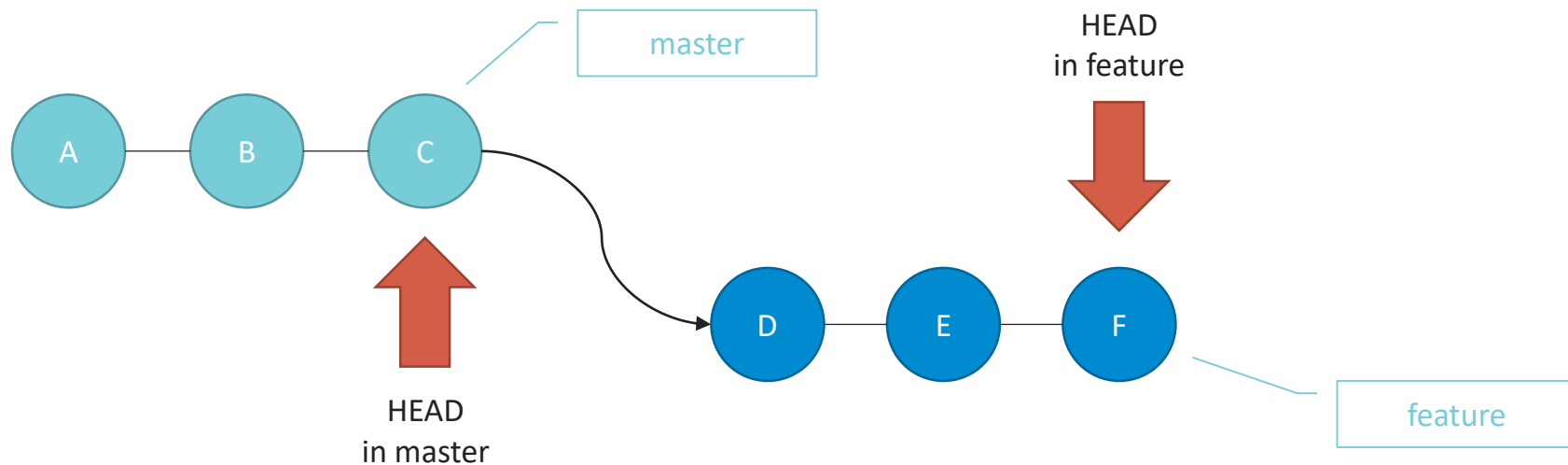


# Branch concept

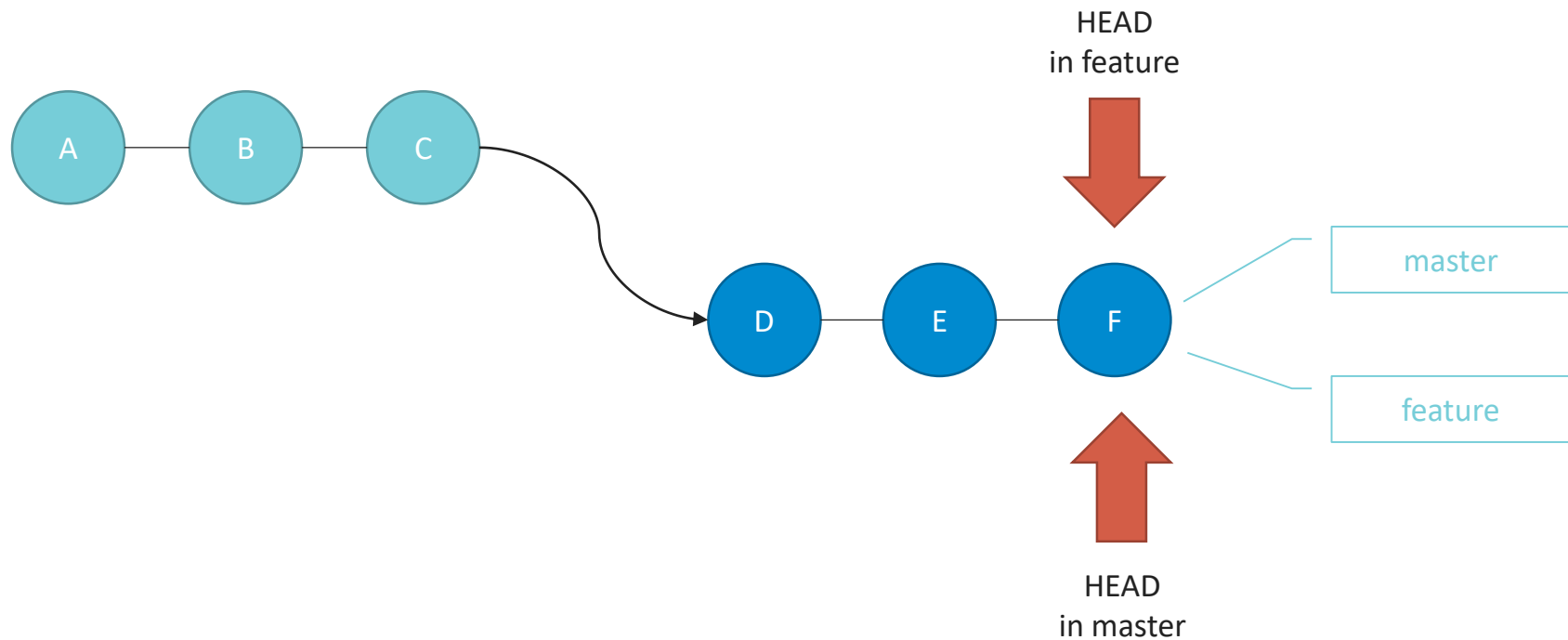




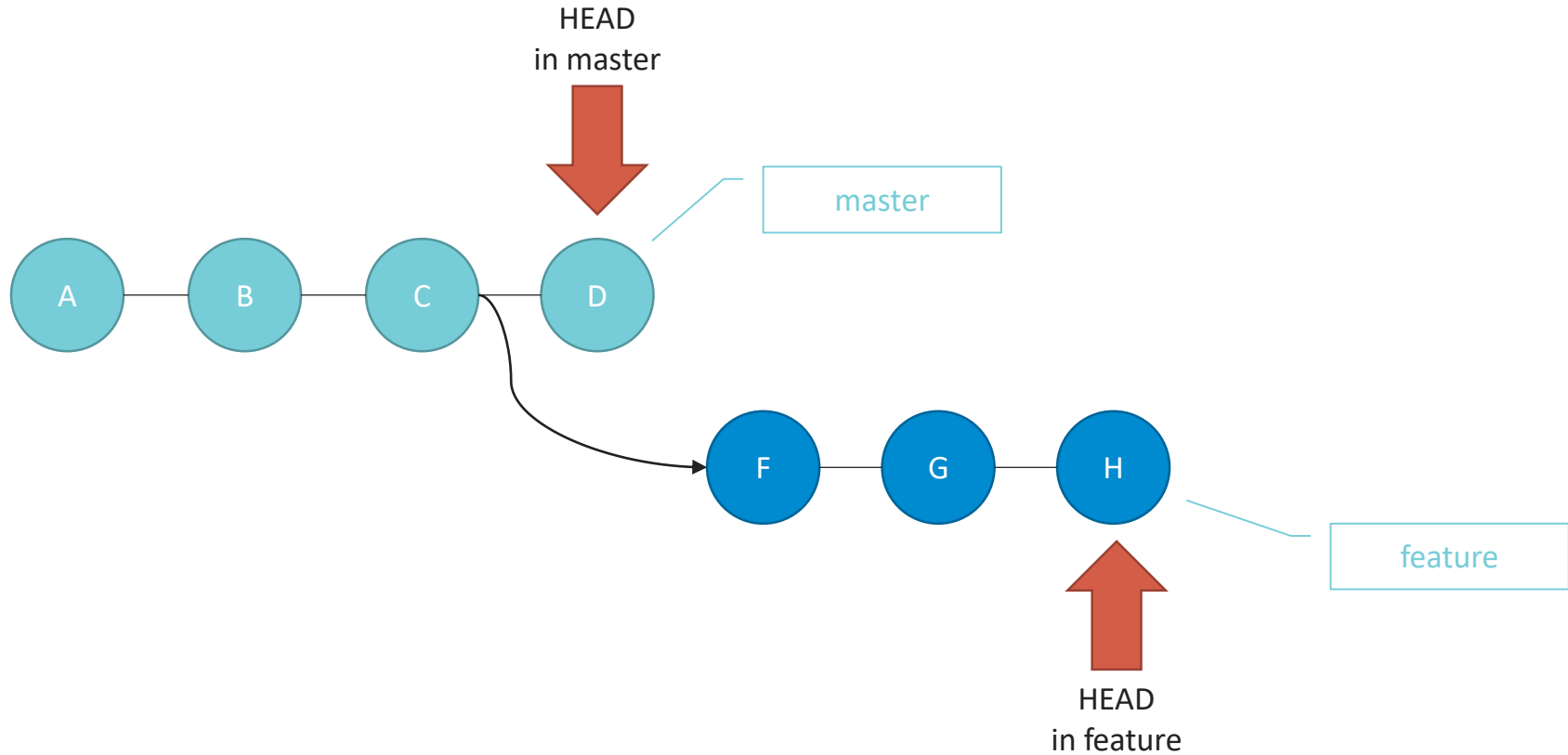
# Fast-forward merge



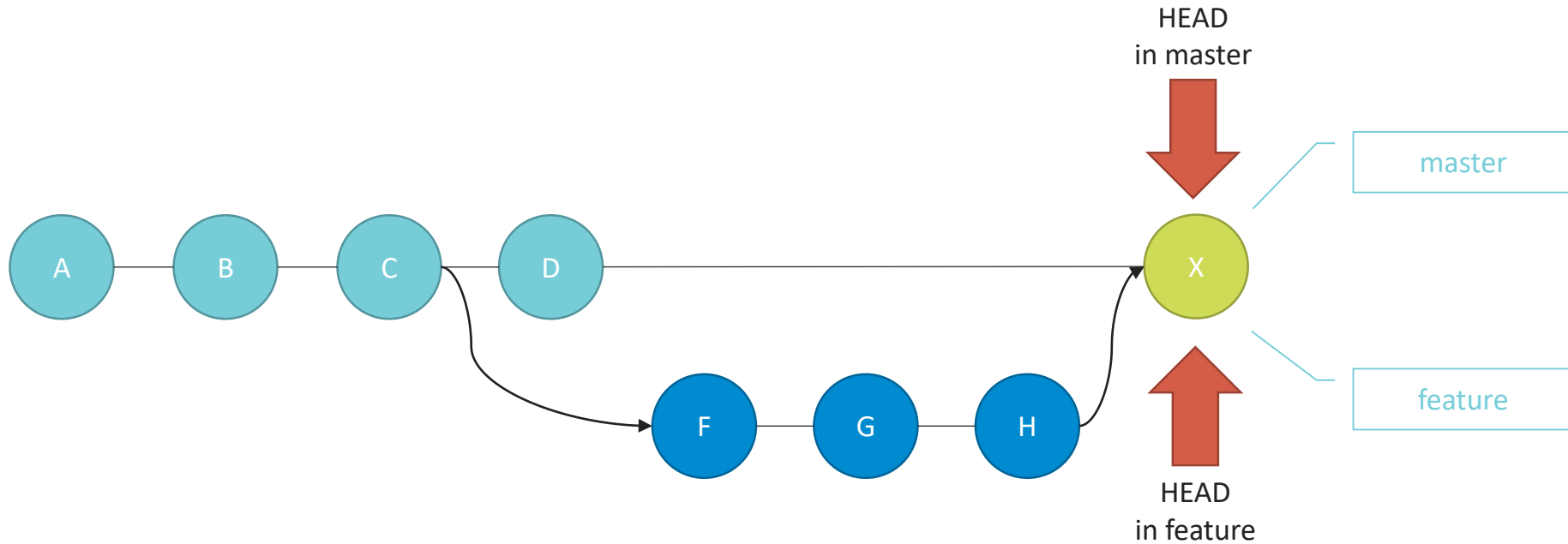
# Fast-forward merge



# Non fast-forward merge



# Non fast-forward merge





# Conflict solving

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Conflicts solving

---

## SOLVE CONFLICT

Abort merge

```
git merge --abort
```

Resolve by selecting version

```
git checkout --Xours --Xtheirs
```

Resolve manually

```
git diff
```

Undo merge

```
git revert 09fe472
```

User merge tool

## AVOID CONFLICT

- Short commits
- No edits to whitespaces
- Merge often



# Rebase

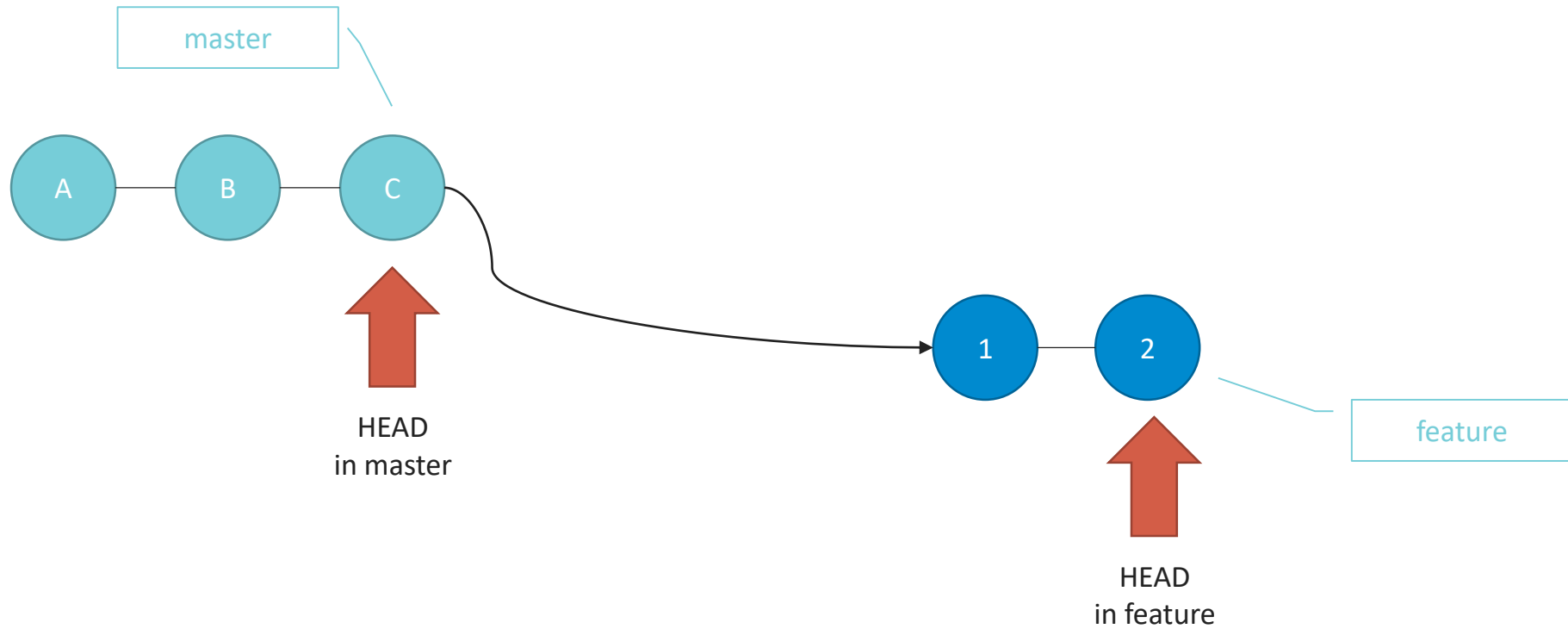
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

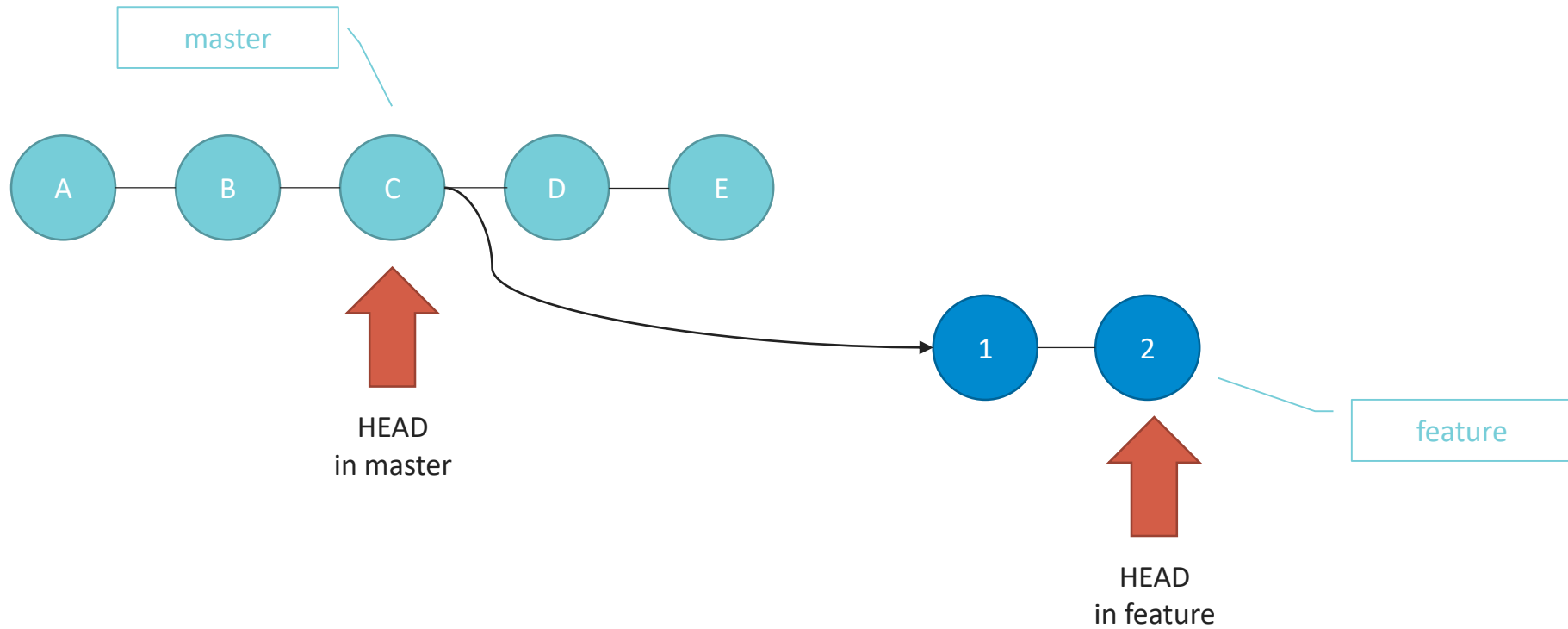


# Rebase

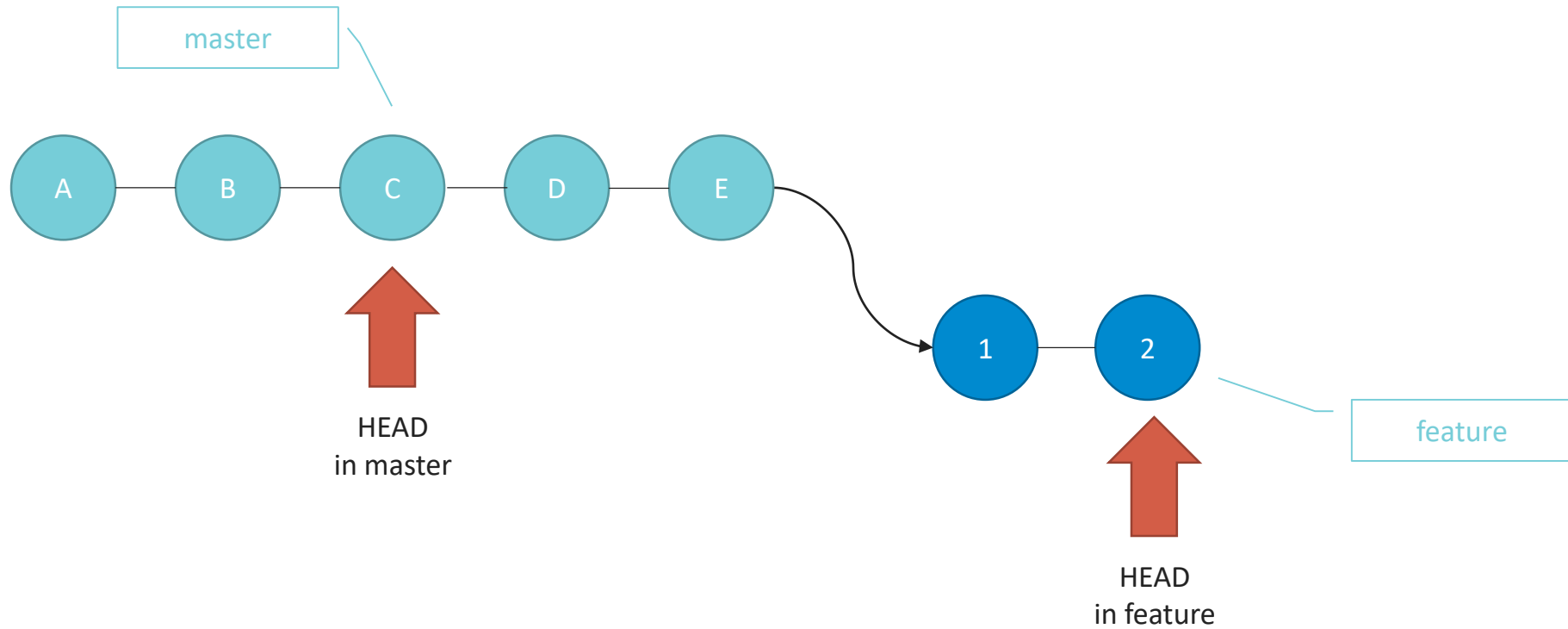




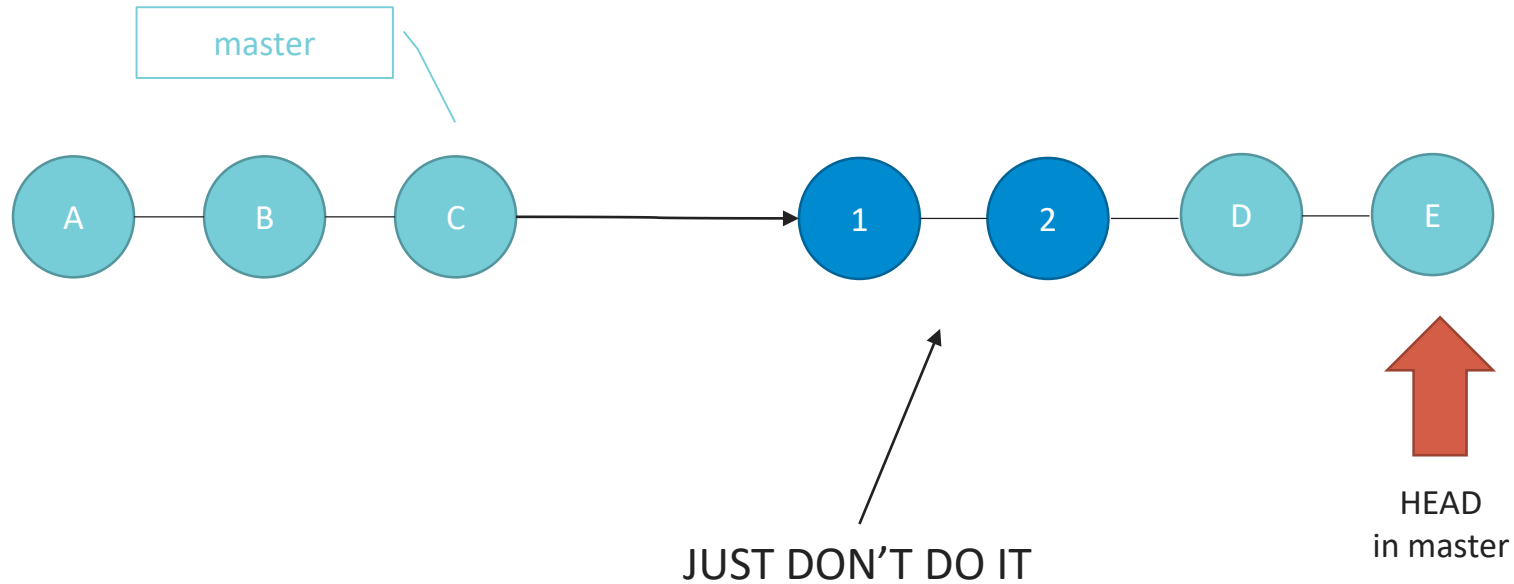
# Rebase



# Rebase



# Golden rule of Rebase





# Cherry-pick

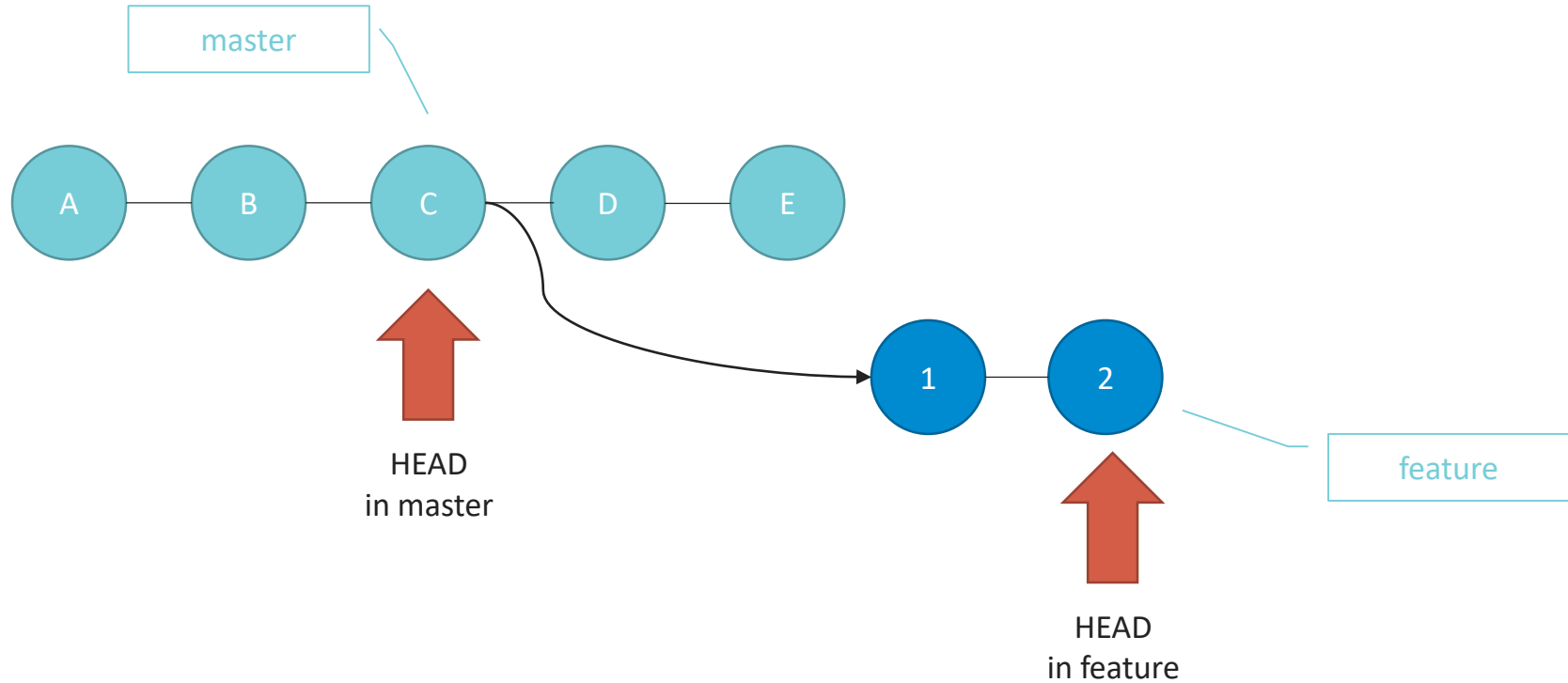
Version Control with Git. DevTestOps training.



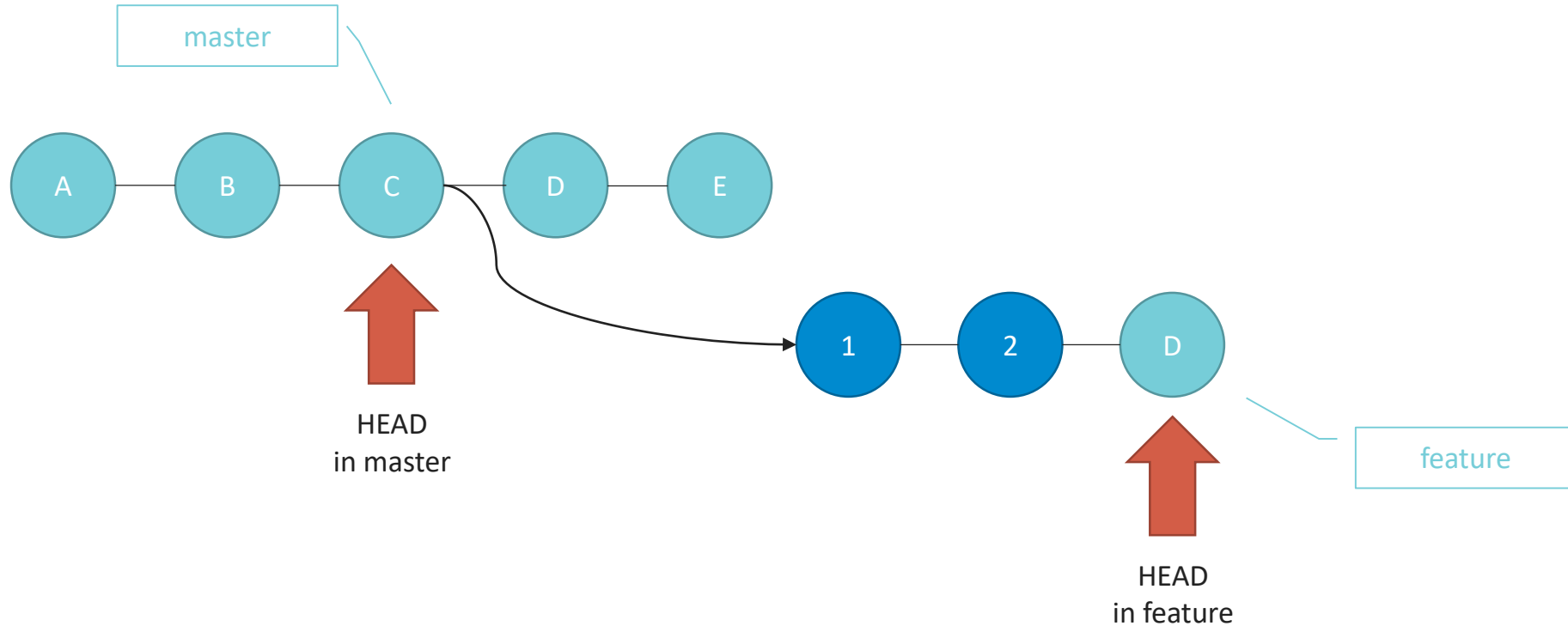
**TRAINING**  
CENTER



# Cherry pick



# Cherry-pick



<epam>

# Tags

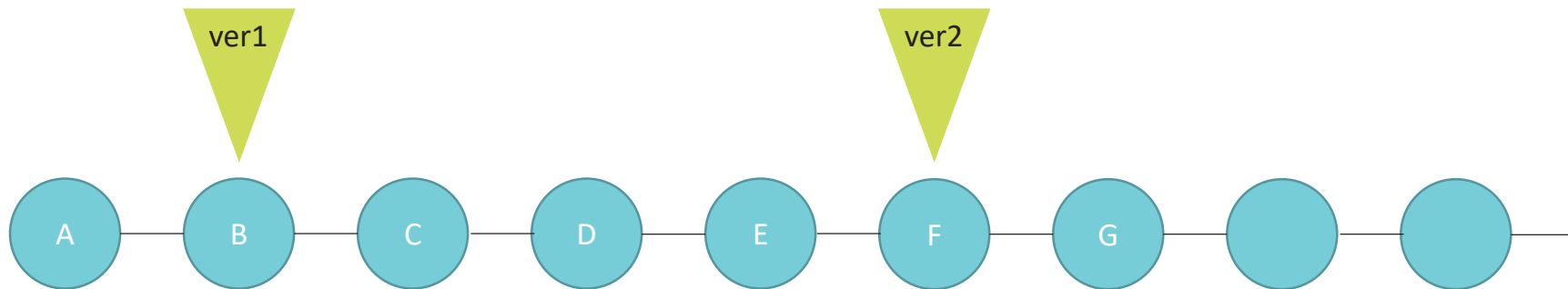
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

# Tags



Mark commit with tag

```
git tag ver1
```

View tags

```
git tag -list
```

Push

```
git push --tags
```

Check it out

```
git checkout ver1
```



<epam>

# Stashing

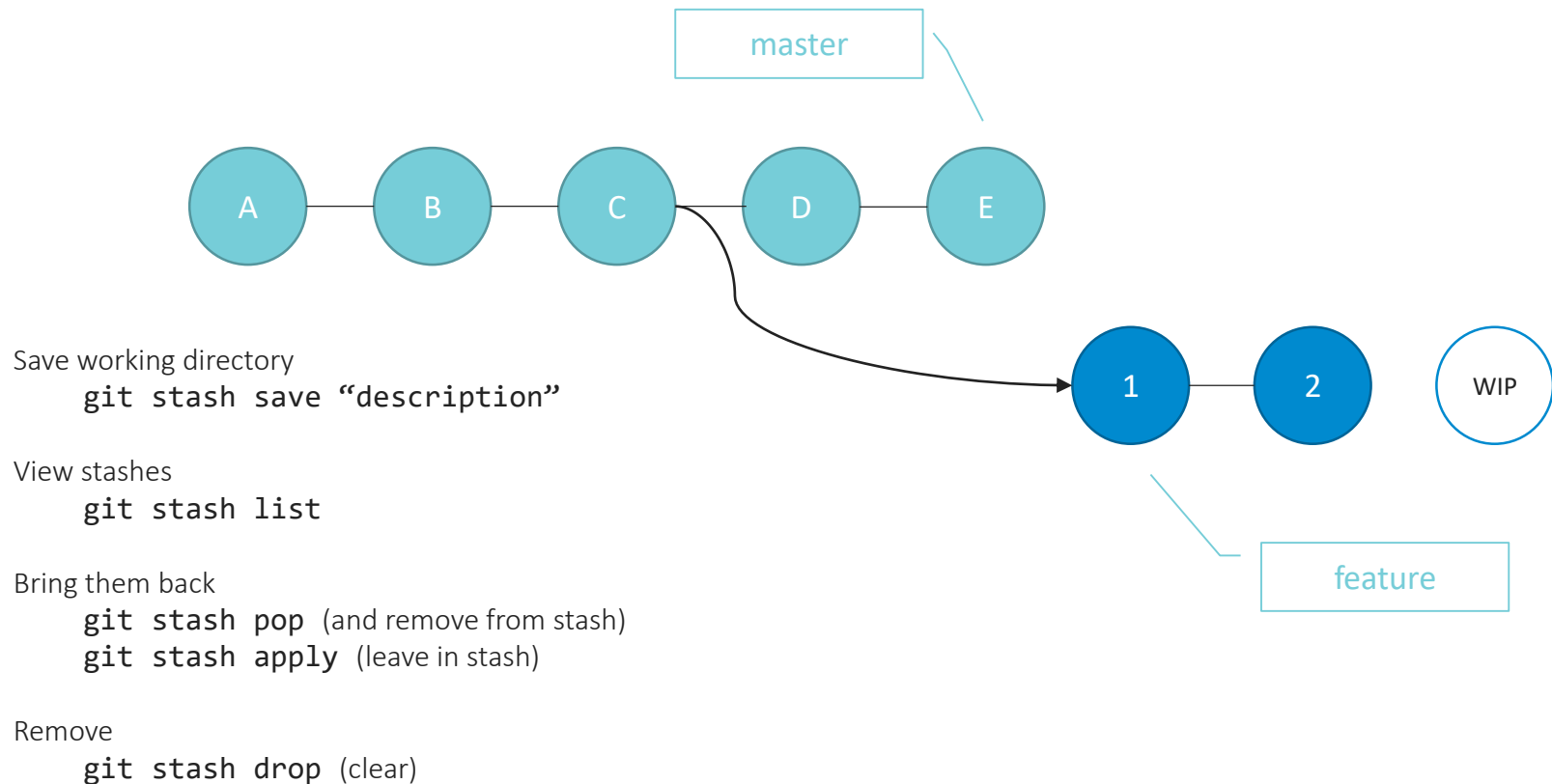
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

# Stash





# Remotes

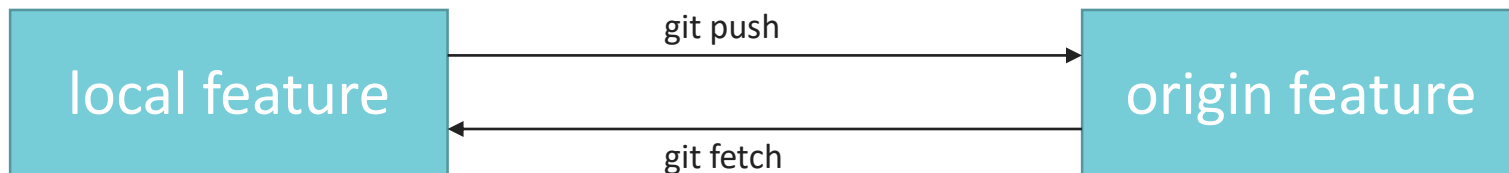
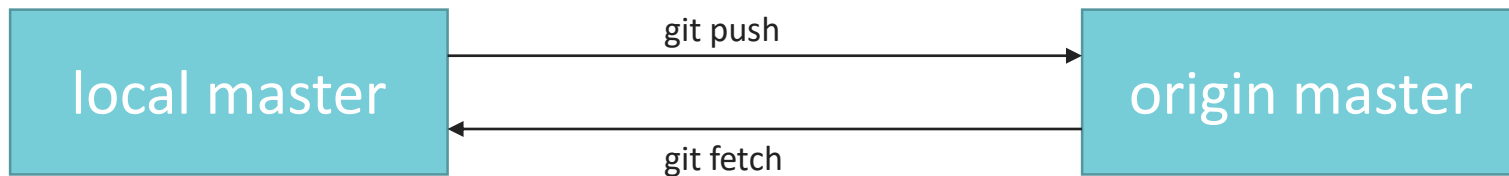
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER



# Remotes



Add

```
git remote add <name> <url>  
git remote add origin git@github.com:user/repo.git
```

View

```
git remote -v  
git remote show <name>
```



# Branching strategies

Version Control with Git. DevTestOps training.

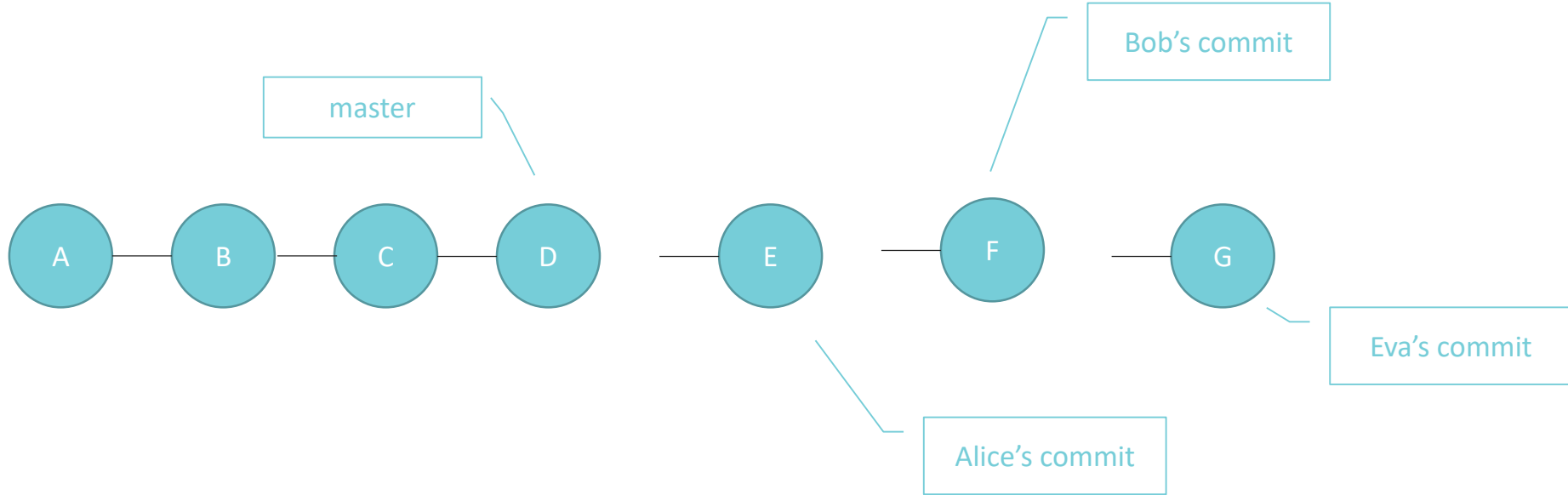


**TRAINING**  
**CENTER**

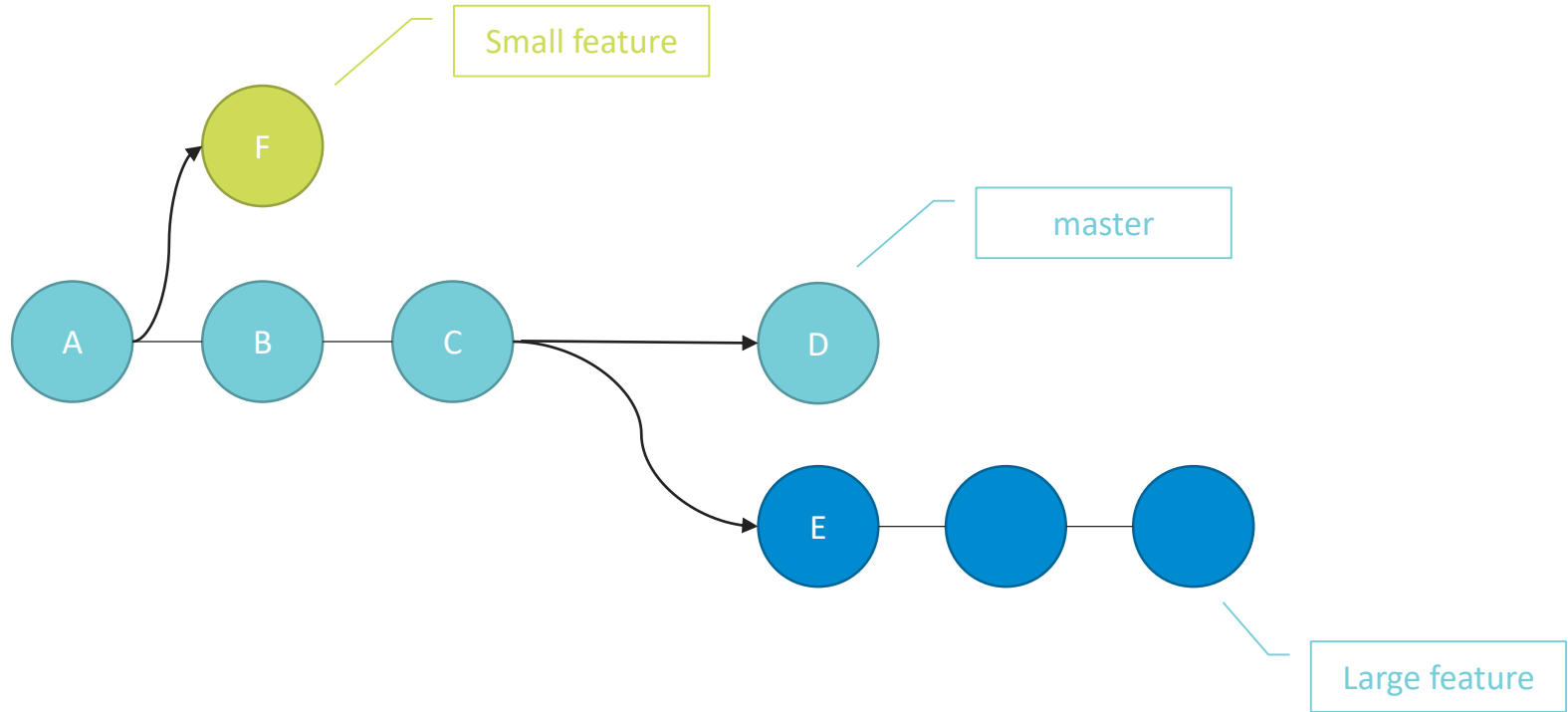


# Centralized strategy

---



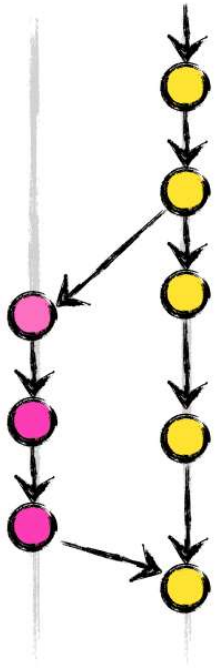
# Feature-branch workflow



# Gitflow

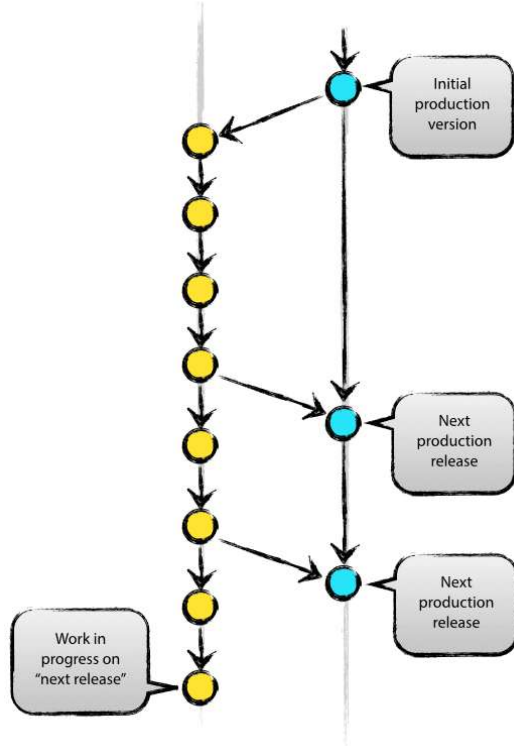
feature  
branches

develop



develop

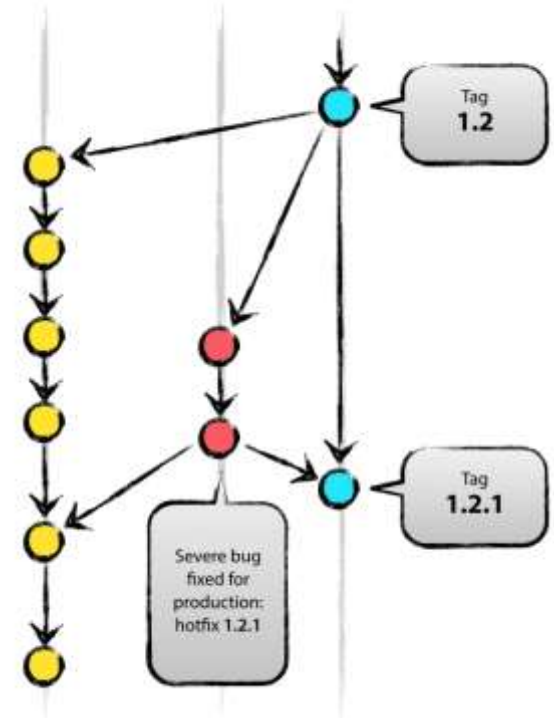
master



develop

hotfixes

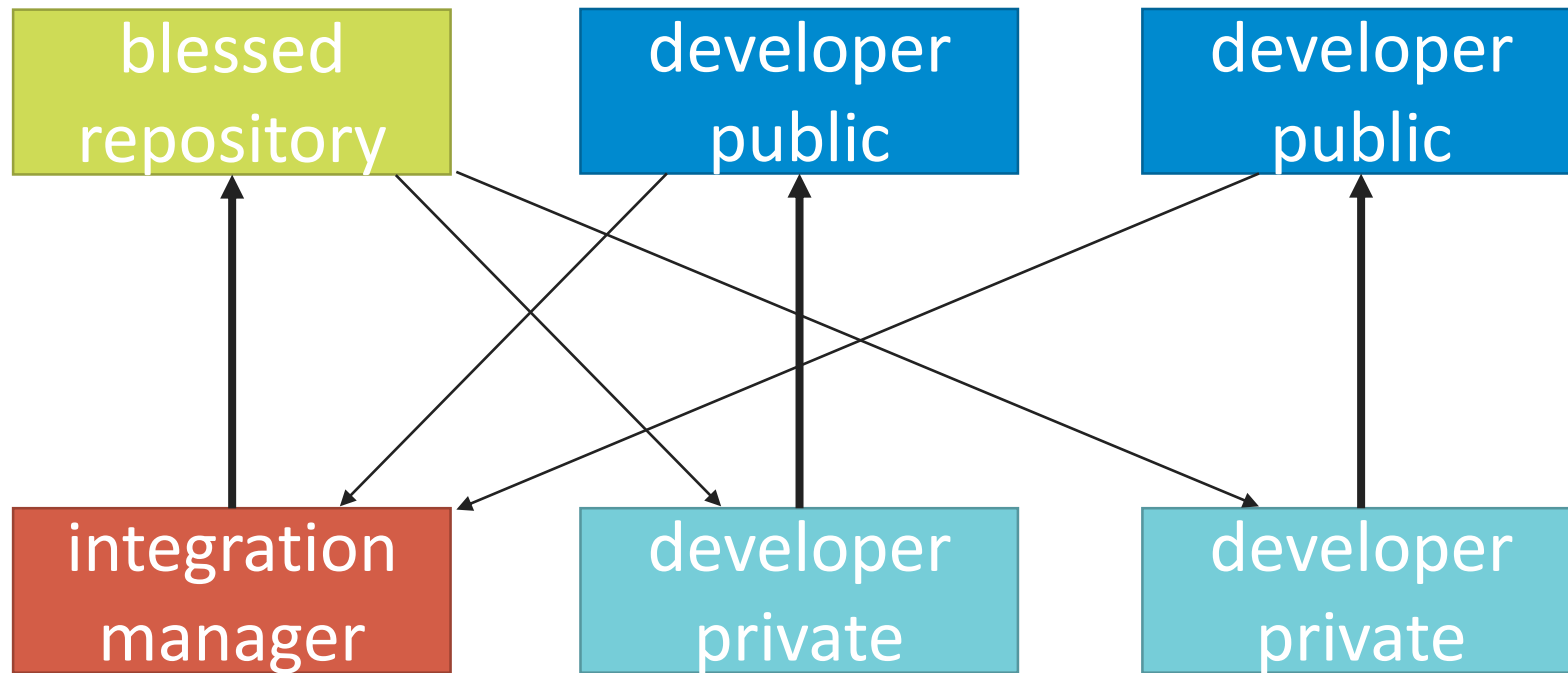
master



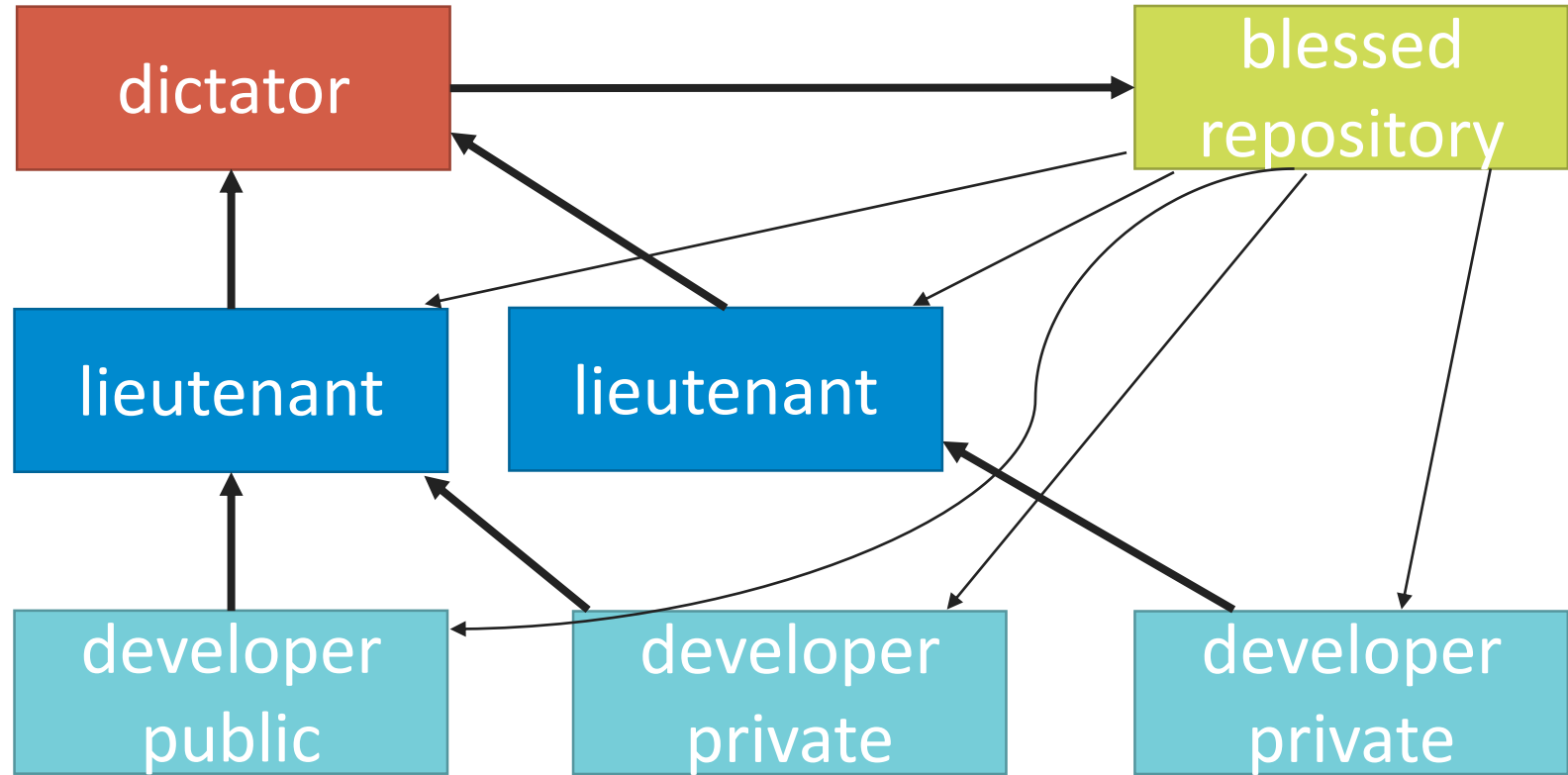
Original post by Vincent Driessen: <https://nvie.com/posts/a-successful-git-branching-model/>



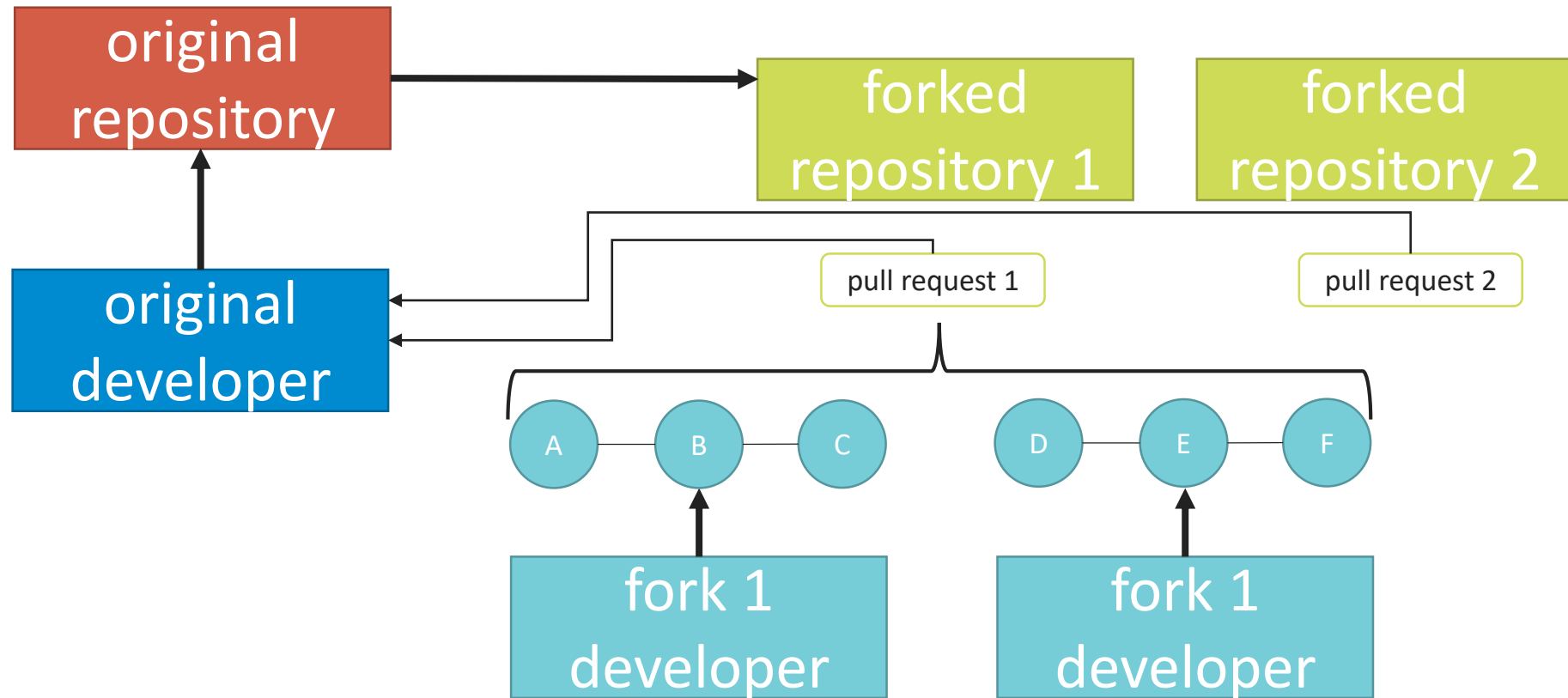
## Integration manager workflow



## Dictator and Lieutenants workflow



## Forking workflow



<epam>

# Inside .git folder

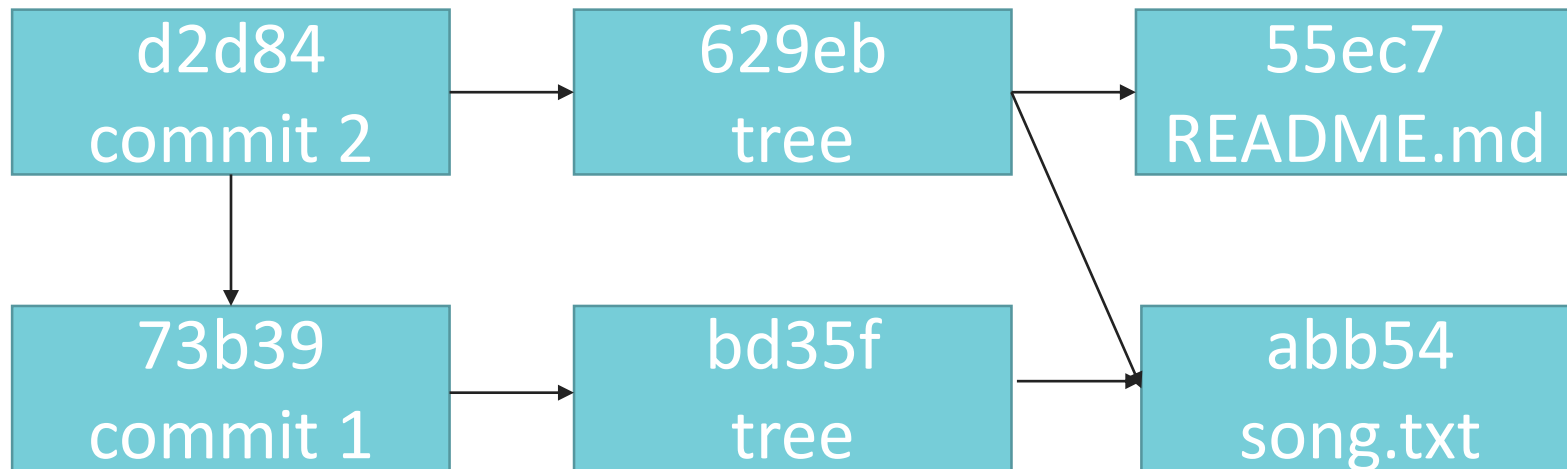
Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

## Commit, tree, blob



- `git show -s --pretty=raw d2d84`
- `git ls-tree 629eb`
- `git show abb54`

<epam>

# Extras

Version Control with Git. DevTestOps training.



**TRAINING**  
CENTER

— <epam> —

## EXTRAS

- `git config -- global user.name "Vitali Shulha"`
- `git config -- global user.email "vitali_shulha@epam.com"`
- `git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe'"`
- `git blame`
- `git bisect`
- `git log --pretty=oneline`
- `git log --pretty=format:"%h %s" -graph`
- `git config --global alias.last 'log -1 HEAD'`
- `git last`
- `git log master..experiment`
- `git filter-branch --tree-filter 'rm -f passwords.txt' HEAD`
- `git rerere`
- `git submodule`

## READ MORE

- **Pro Git** by Scott Chacon and Ben Straub
- **Version Control with Git** by Jon Loeliger, Matthew McCullough

