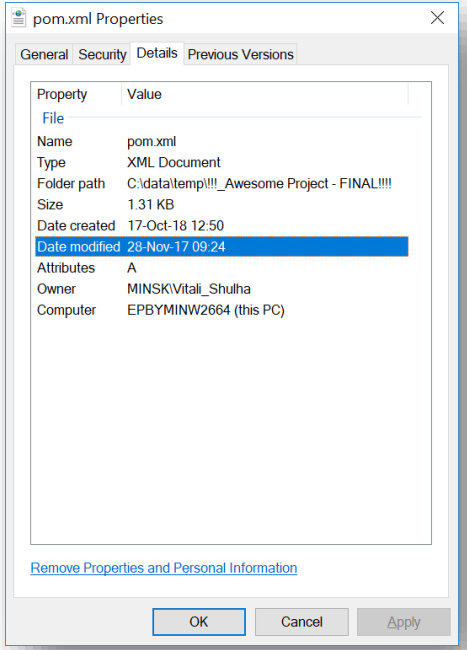
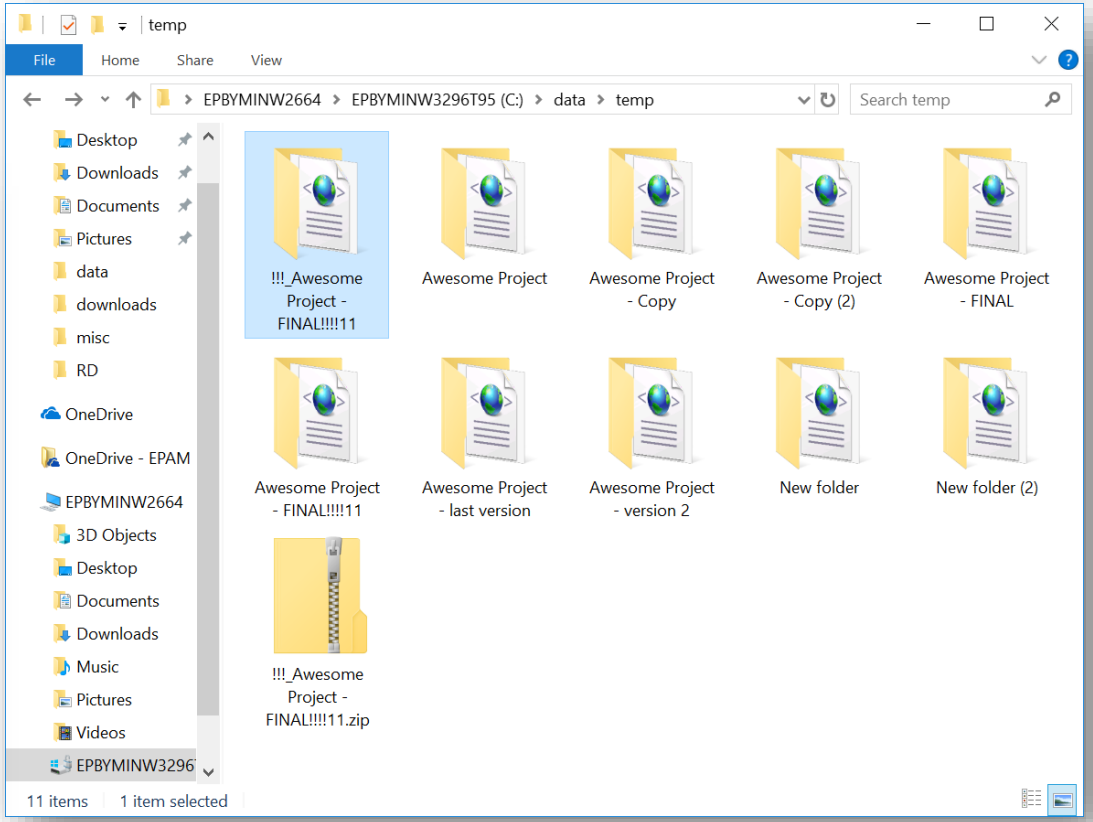
VCS concept

Version Control with Git. DevTestOps training.

Standalone work. Level 1 - beginner





Team work. Level 2 – network share

*\\epbyminw2664\Awesome Project\pom.xml - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

pom.xml

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchem
2      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_
3      <modelVersion>4.0.0</modelVersion>
4      <groupId>com.epam.ta</groupId>
5      <artifactId>GitHubAutomation</artifactId>
6      <packaging>jar</packaging>
7      <version>1.0</version>
8      <name>GitHubAutomation</name>
9      <url>http://maven.apache.org</url>
10
11     <dependencies>
12         <dependency>
13             <groupId>org.seleniumhq.selenium</gr
14             <artifactId>selenium-java</artifactI
15             <version>3.7.1</version>
16         </dependency>
17         <dependency>
18             <groupId>org.testng</groupId>
19             <artifactId>testng</artifactId>
20             <version>6.8.8</version>
21         </dependency>
```

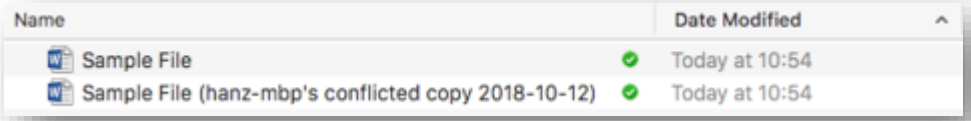
Reload

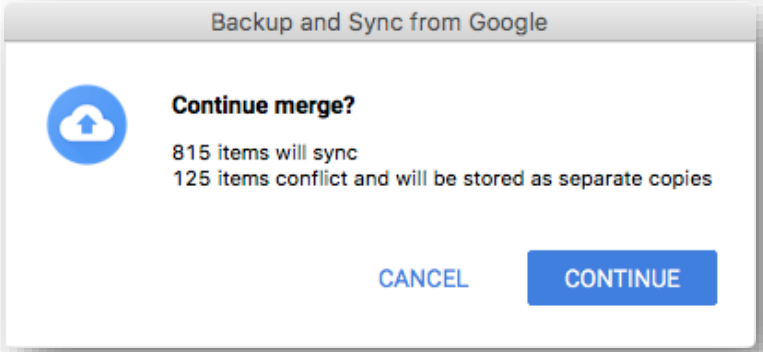
\\epbyminw2664\Awesome Project\pom.xml

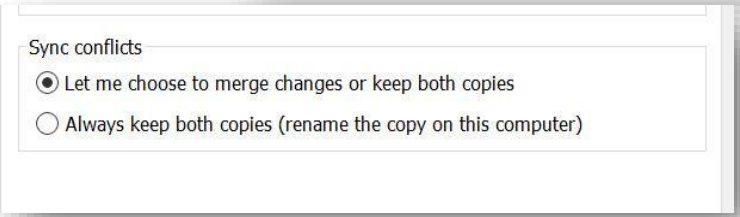
This file has been modified by another program.
Do you want to reload it and lose the changes made in Notepad++?

Yes No

Standalone/Team work. Level 3 - cloud







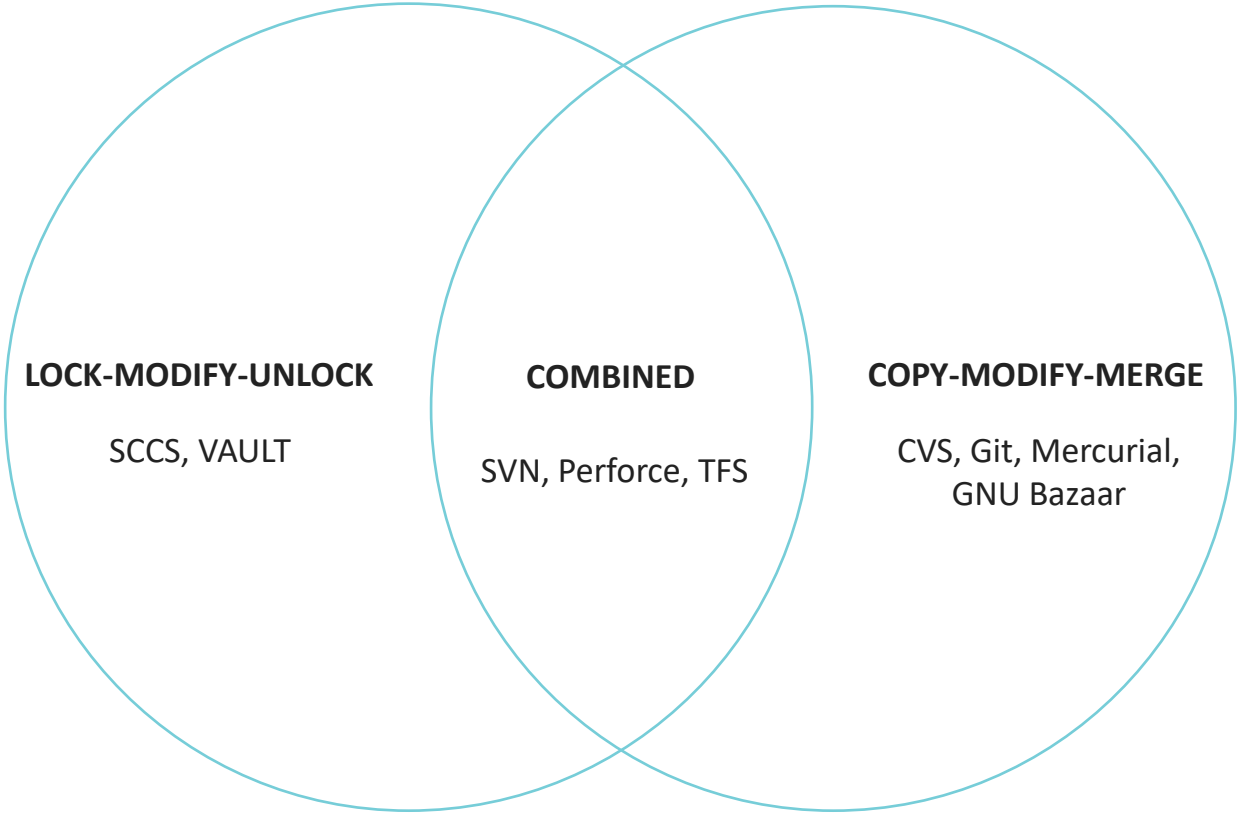
VCS goals

1. BACKUP AND RESTORE
2. SYNCHRONIZATION
3. UNDO
4. TRACK CHANGES AND OWNERSHIP
5. SANDBOXING
6. BRANCHING

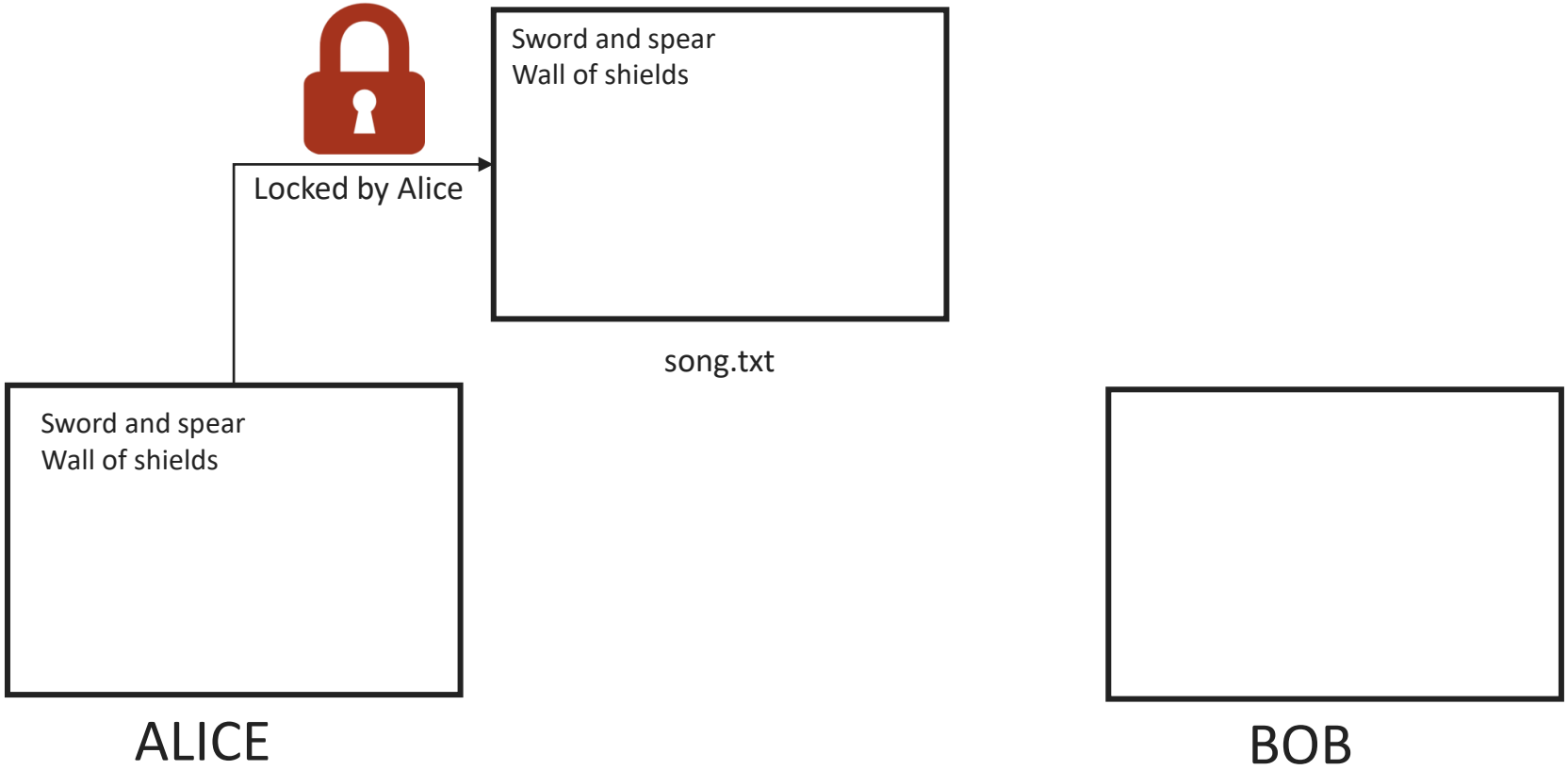
Version control types

Version Control with Git. DevTestOps training.

VCS types


LOCK-MODIFY-UNLOCK
SCCS, VAULT
COMBINED
SVN, Perforce, TFS
COPY-MODIFY-MERGE
CVS, Git, Mercurial, GNU Bazaar

Lock-modify-unlock strategy



Lock-modify-unlock strategy

Sword and spear
Wall of shields

song.txt

Bob updates his local copy

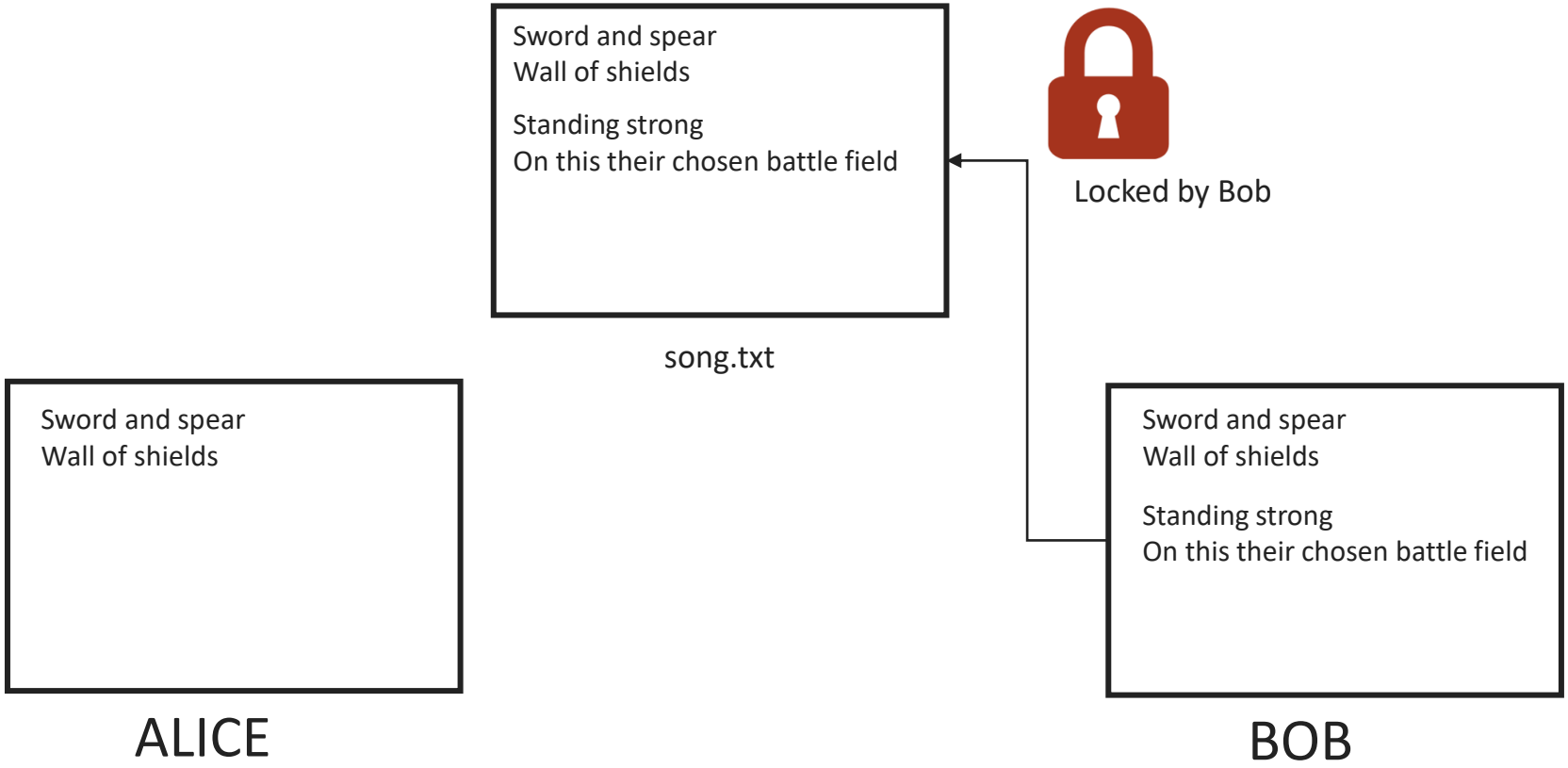
Sword and spear
Wall of shields

ALICE

Sword and spear
Wall of shields

BOB

Lock-modify-unlock strategy



Lock-modify-unlock strategy

Sword and spear
Wall of shields

Standing strong
On this their chosen battle field

song.txt

Unlocked by Bob

Sword and spear
Wall of shields

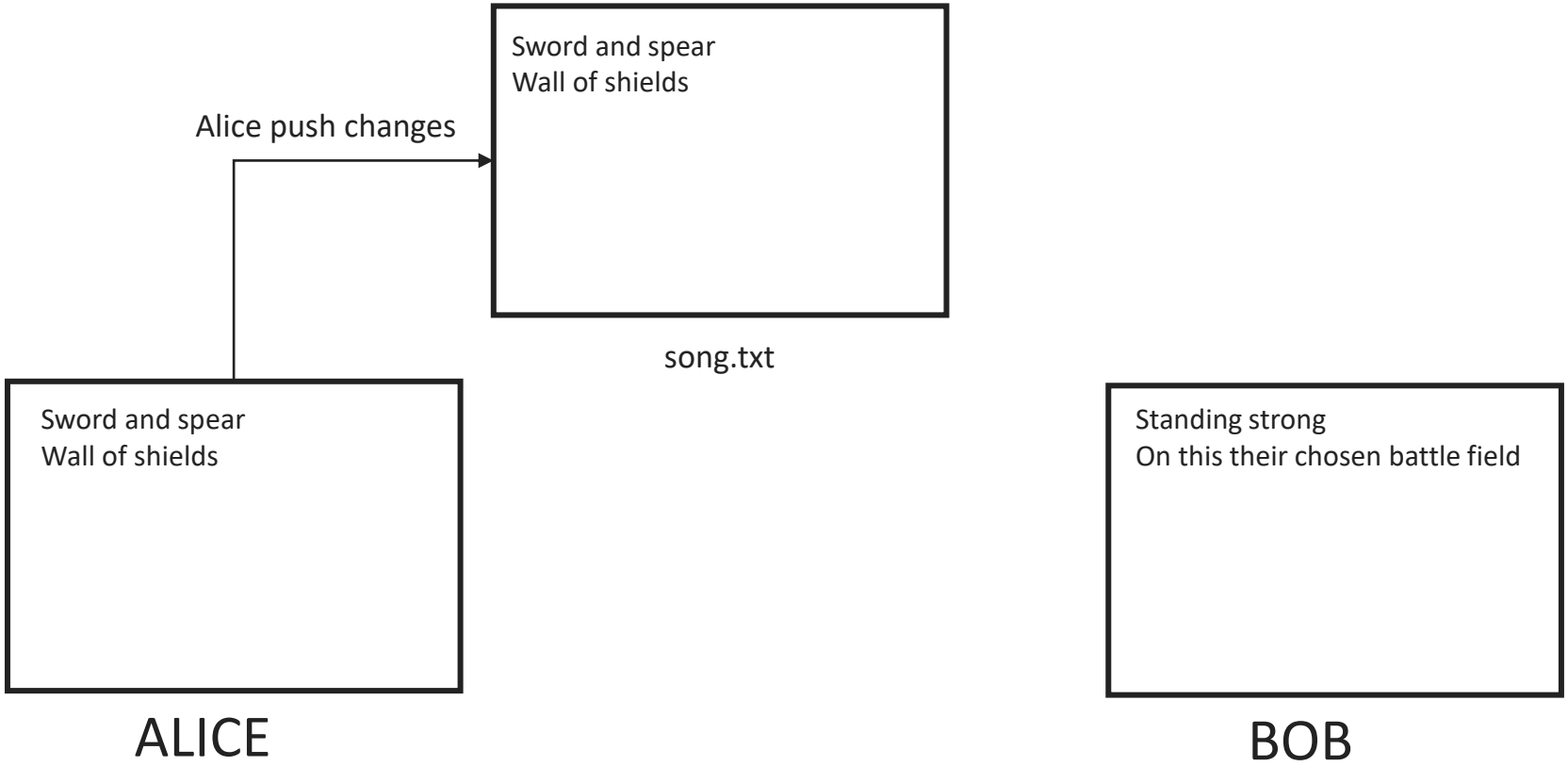
ALICE

Sword and spear
Wall of shields

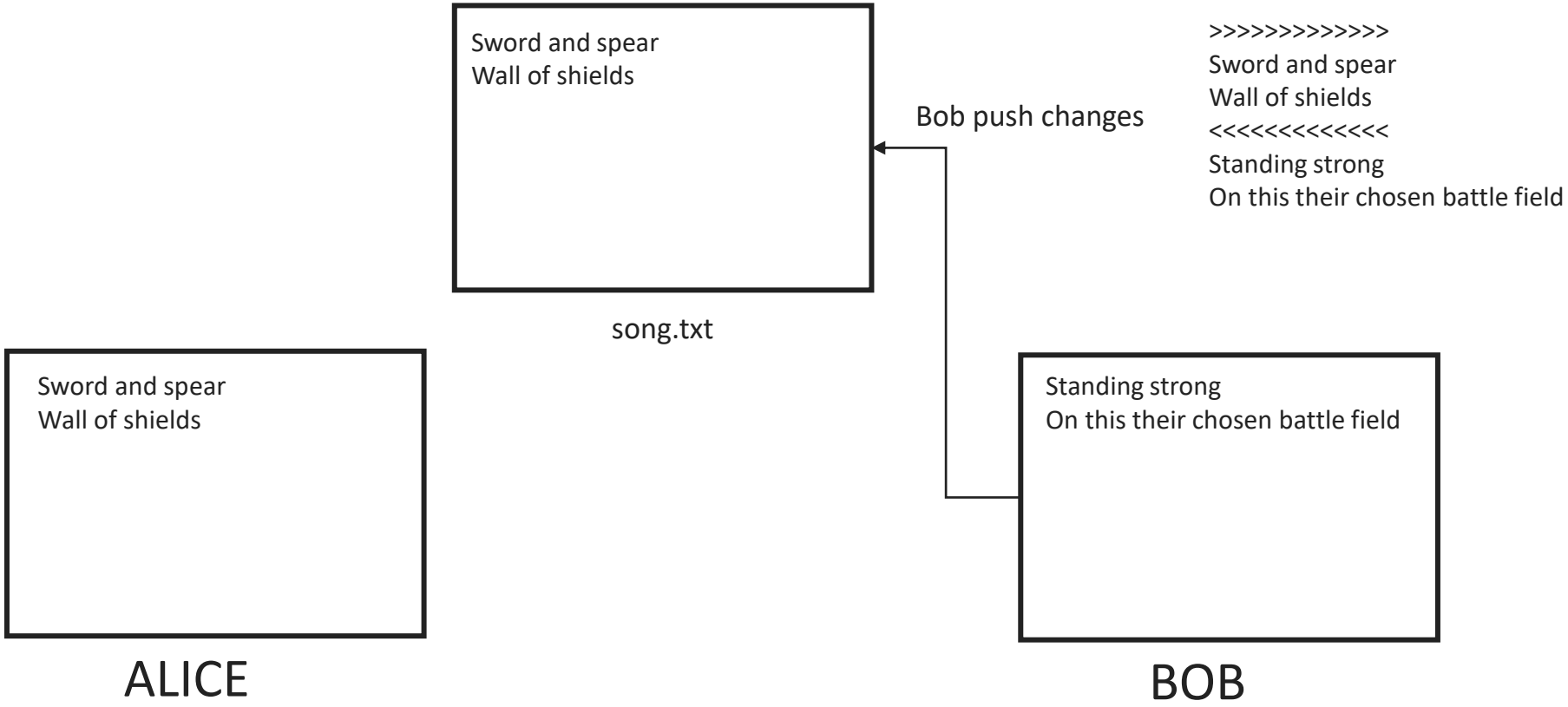
Standing strong
On this their chosen battle field

BOB

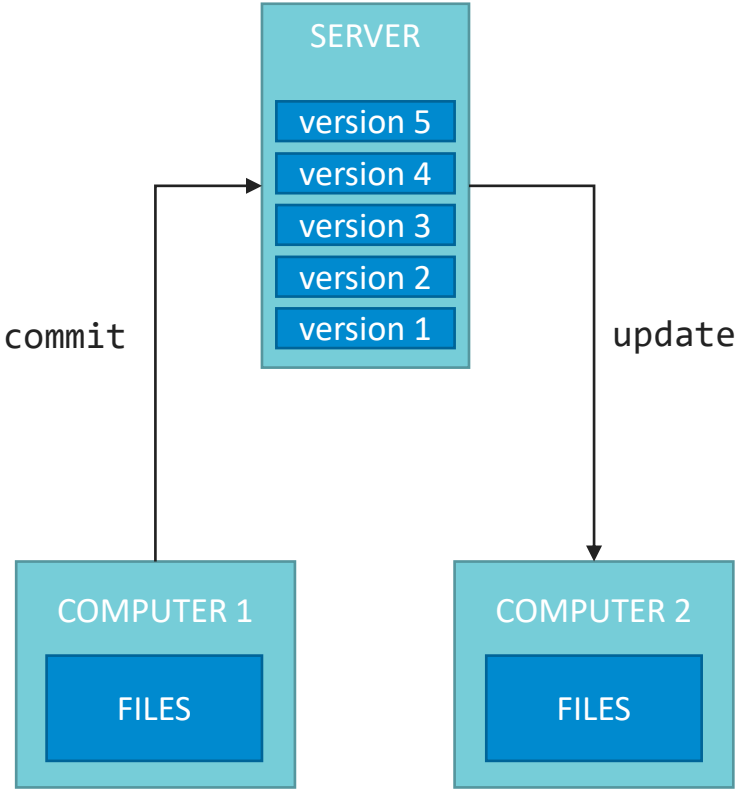
Copy-modify-merge strategy

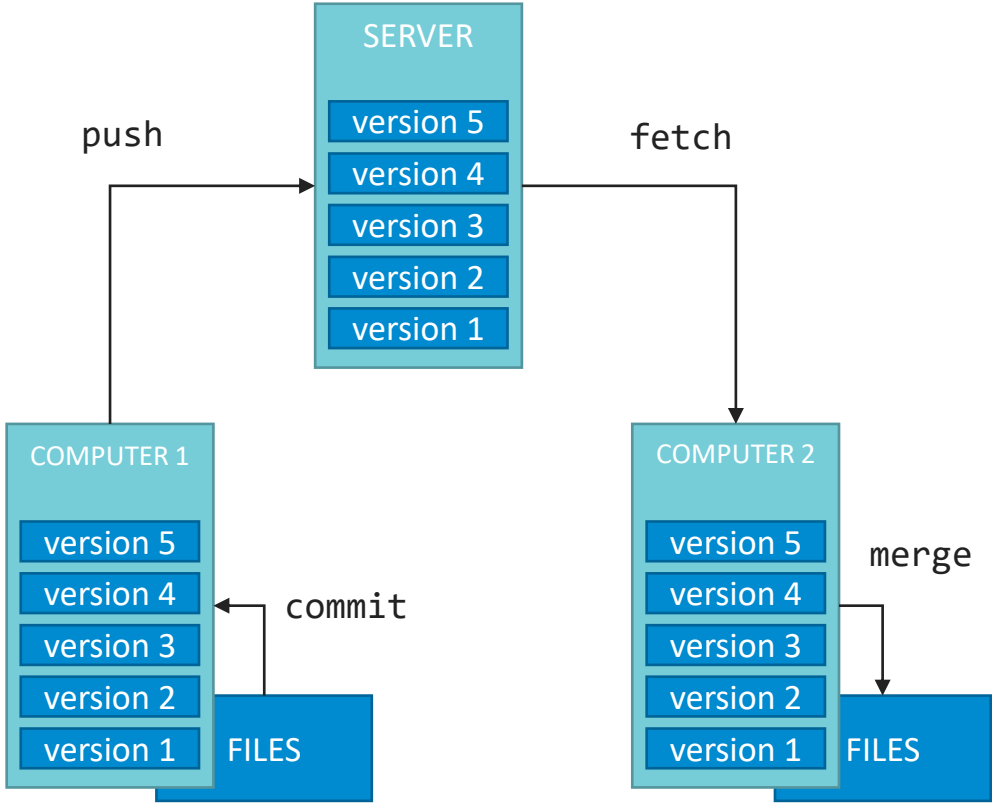


Copy-modify-merge strategy



Centralized vs Distributed



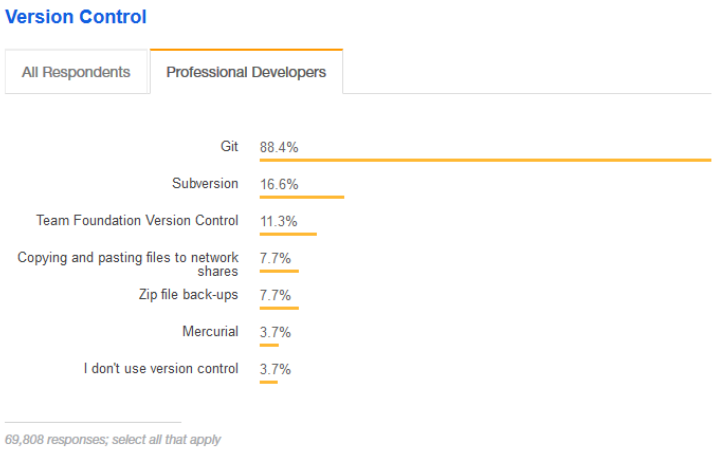


Why Git

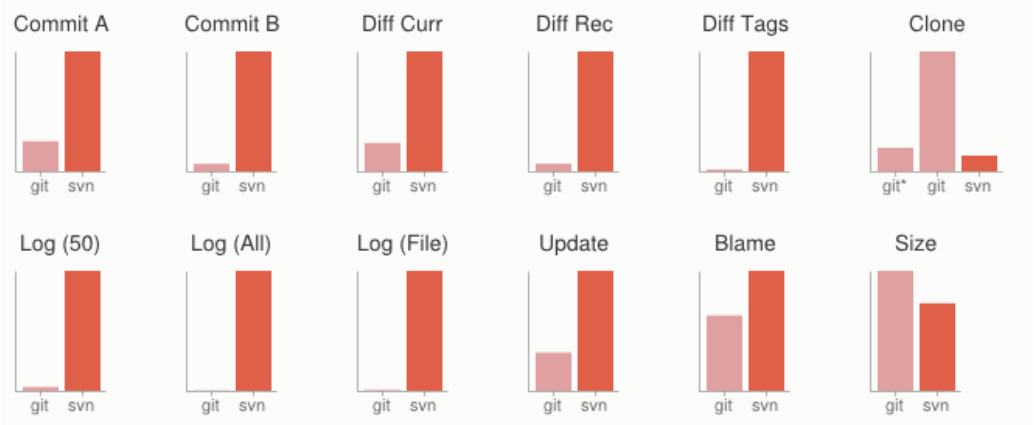
Version Control with Git. DevTestOps training.

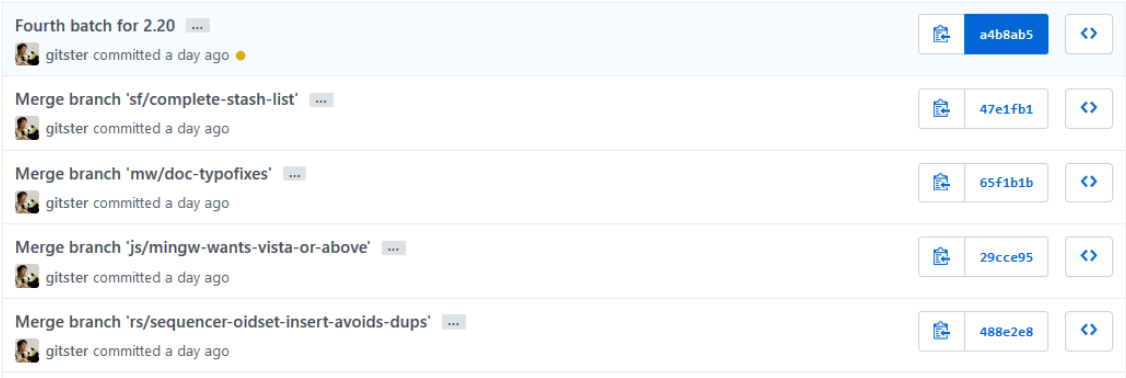
Why Git

- 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.



Git benefits





Download, install, configure

Version Control with Git. DevTestOps training.

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

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

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

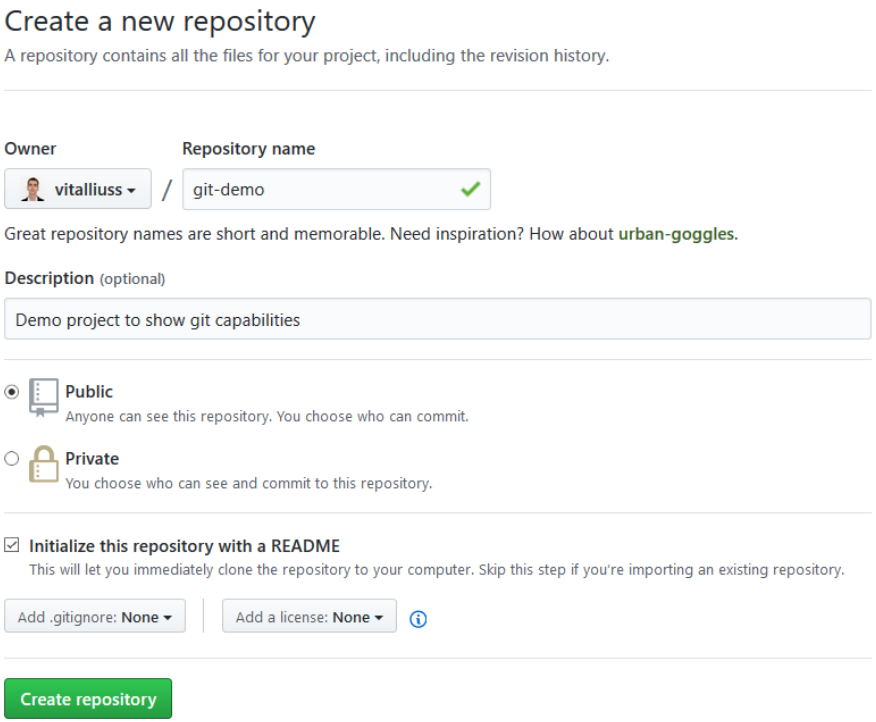
```
git config --global user.name “Vitali Shulha“
git config --global user.email “vitali_Shulha@epam.com"
```

Create a github repo and clone it

Version Control with Git. DevTestOps training.

Create github repository and clone it

- Simple as it can be
- Don’t forget to select “Initialize this repository with a README”



Create github repository

Create a new repository

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

Owner: vitalliuss / Repository name: 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 repos

Add .gitignore: None | Add a license: None

Create repository

vitalliuss / 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

vitalliuss Initial commit

README.md Initial commit

README.md

Clone with SSH — Use HTTPS

Use an SSH key and passphrase from account.

git@github.com:vitalliuss/git-demo.git

Open in Desktop | Download ZIP

git-demo

Demo project to show git capabilities

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:vitalliuss/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.

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

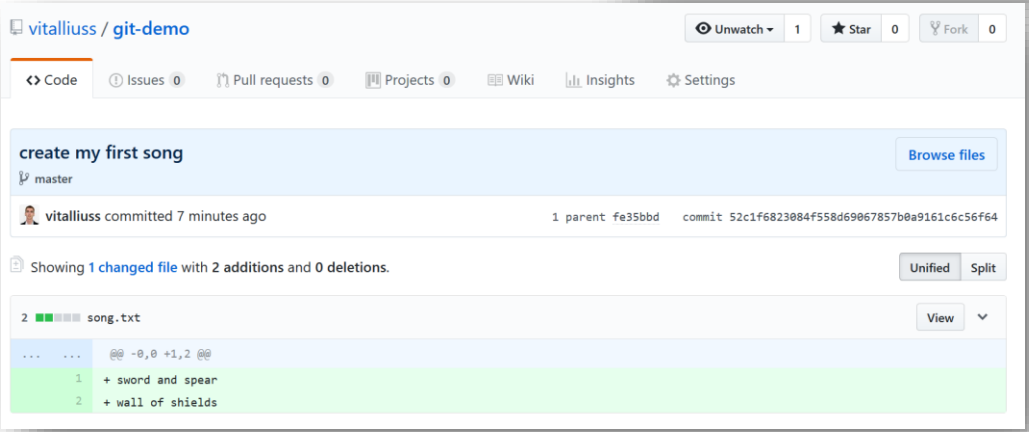
```
Vitali_Shulha@EPBYMINW2664 MINGW64 /c/data/temp/git-demo (master)
$ echo 'sword and spear' >> song.txt

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:vitalliuss/git-demo.git (fetch)
origin  git@github.com:vitalliuss/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:vitalliuss/git-demo.git
   fe35bbd..52c1f68  master -> master
```



Pull from remote

Version Control with Git. DevTestOps training.

Create commit in github and pull it

Edit file in web

Add new lines in the song

Commit changes

Edit file | Preview changes | Spaces

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

Commit changes

complete the first couplet with two more lines

Add an optional extended description...

vitali_shulha@epam.com

Commit directly to the `master` branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit changes | Cancel

vitalliuss / **git-demo**

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki

Branch: master | git-demo / song.txt

vitalliuss create my first song

1 contributor

3 lines (2 sloc) | 32 Bytes

Raw | Blame | History | Edit this file

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

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:vitalliuss/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 Find file Copy path

vitalliuss 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.

Undoing changes

Version Control with Git. DevTestOps training.

Undoing changes

Working directory

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

Staging area (Index)

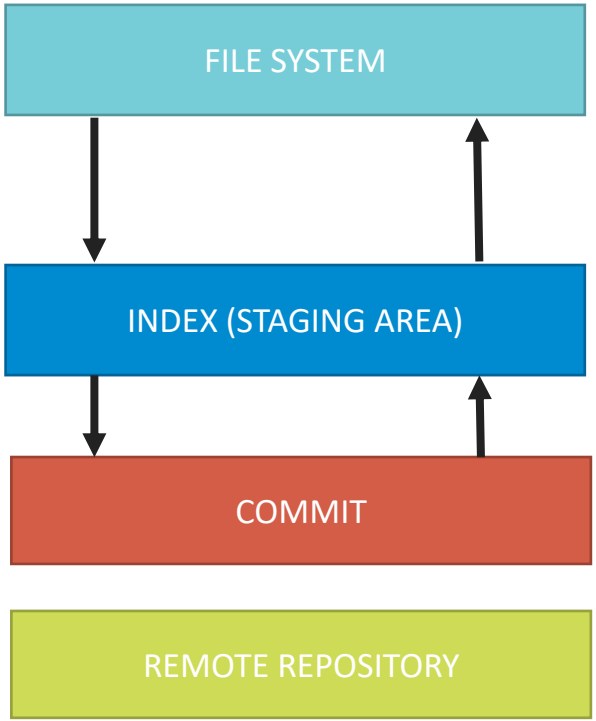
```
git reset -- file.txt
```

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.

Undoing changes

Working directory

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

Staging area (Index)

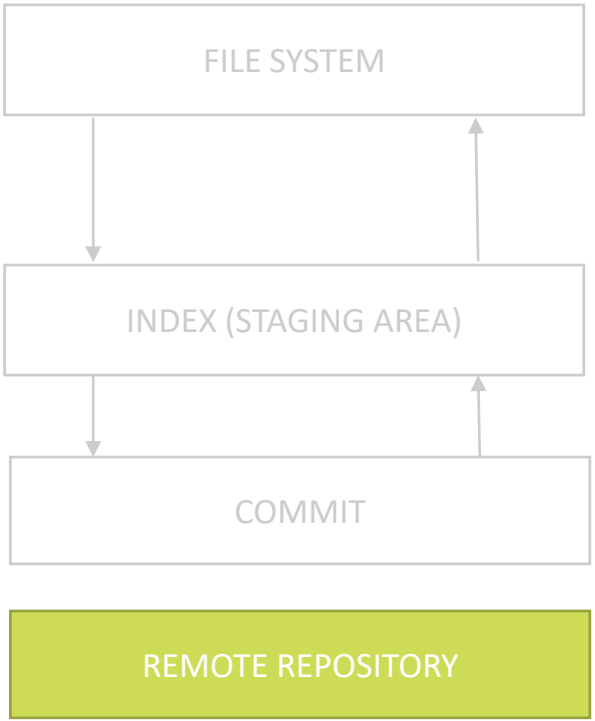
```
git reset -- file.txt
```

Local branch

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

Remote repository

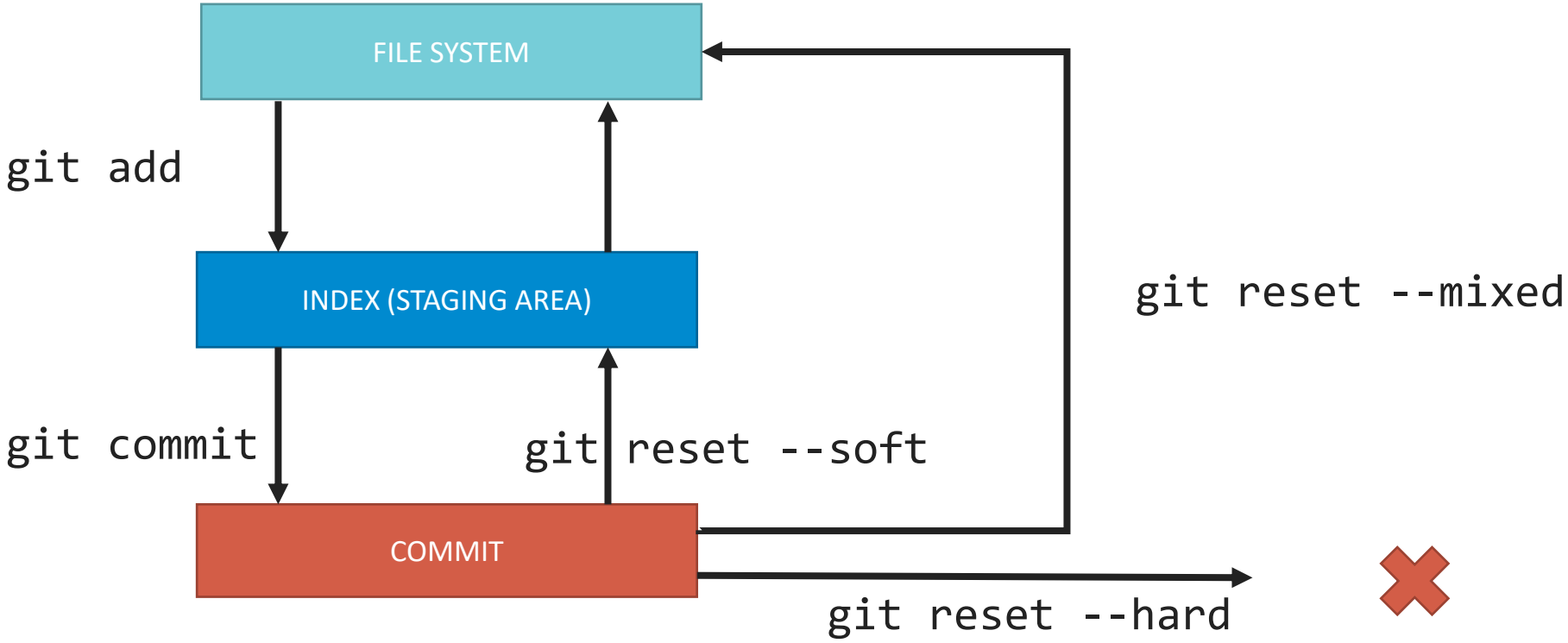
```
git revert <sha1>
```



Git reset

Version Control with Git. DevTestOps training.

Git reset


FILE SYSTEM
git add
INDEX (STAGING AREA)
git reset --mixed
git commit
git reset --soft
COMMIT
git reset --hard

.gitignore

Version Control with Git. DevTestOps training.

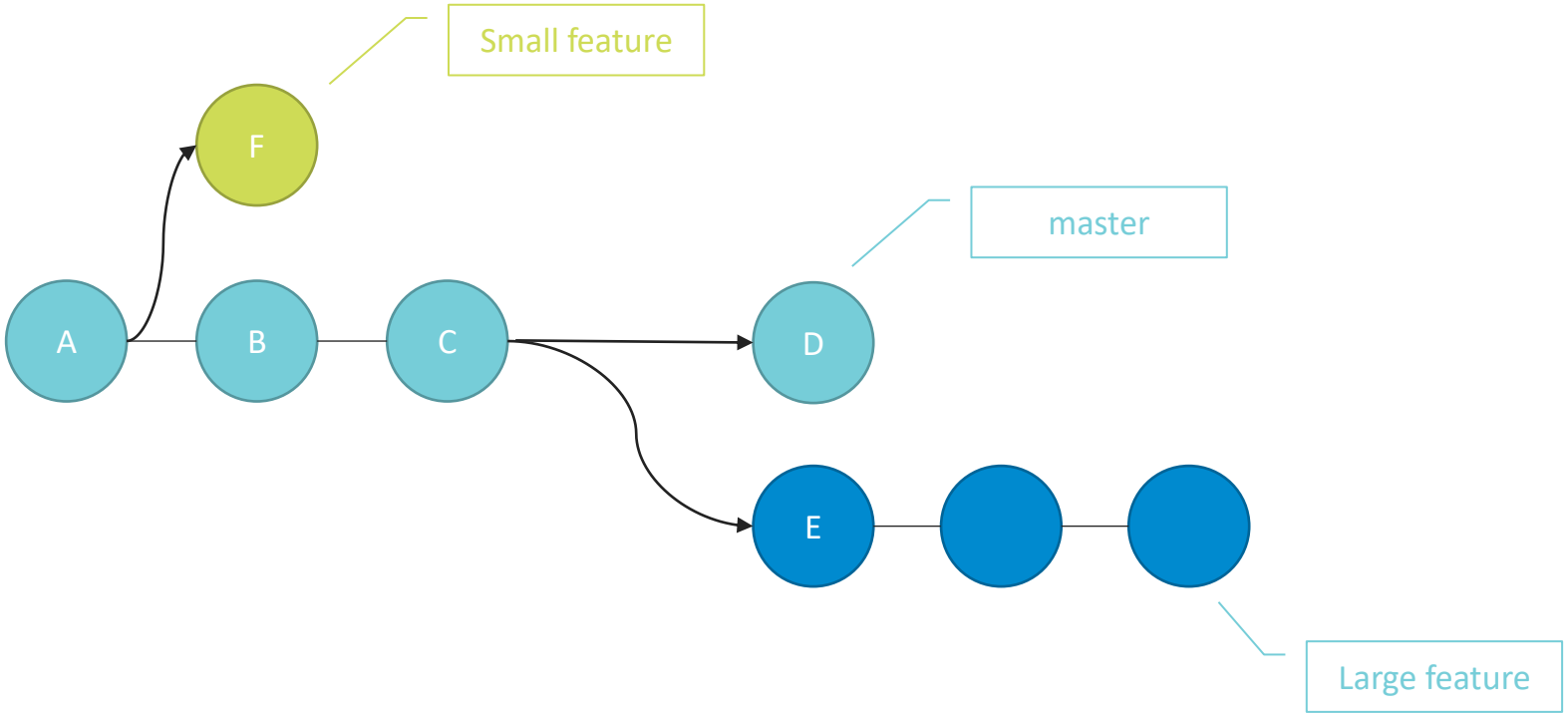
.gitignore

```
# 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
```

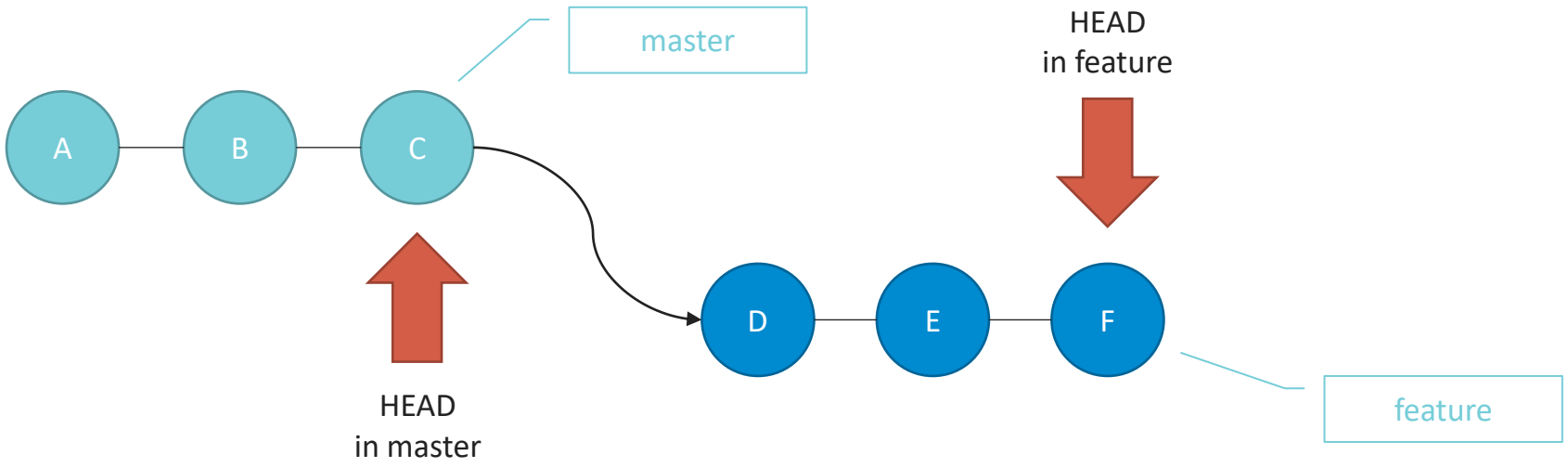
Branching and merge

Version Control with Git. DevTestOps training.

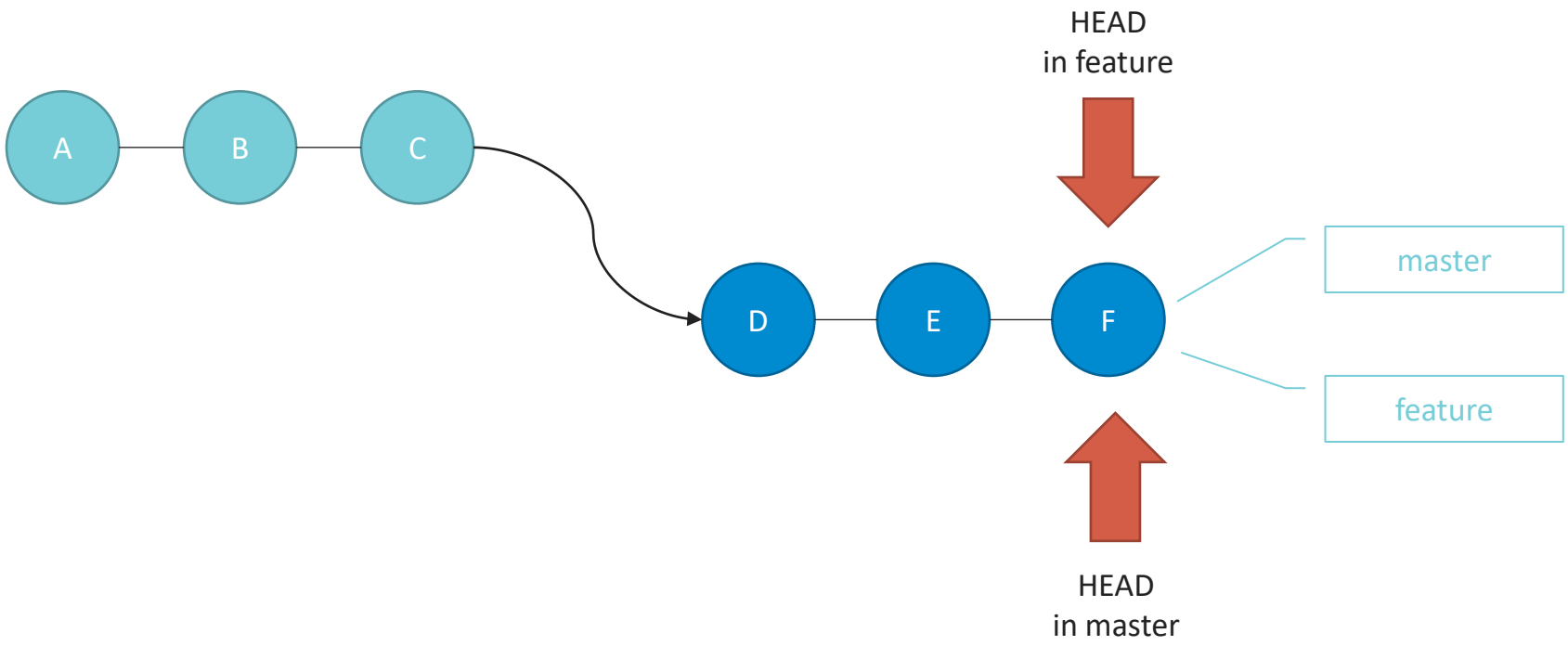
Branch concept



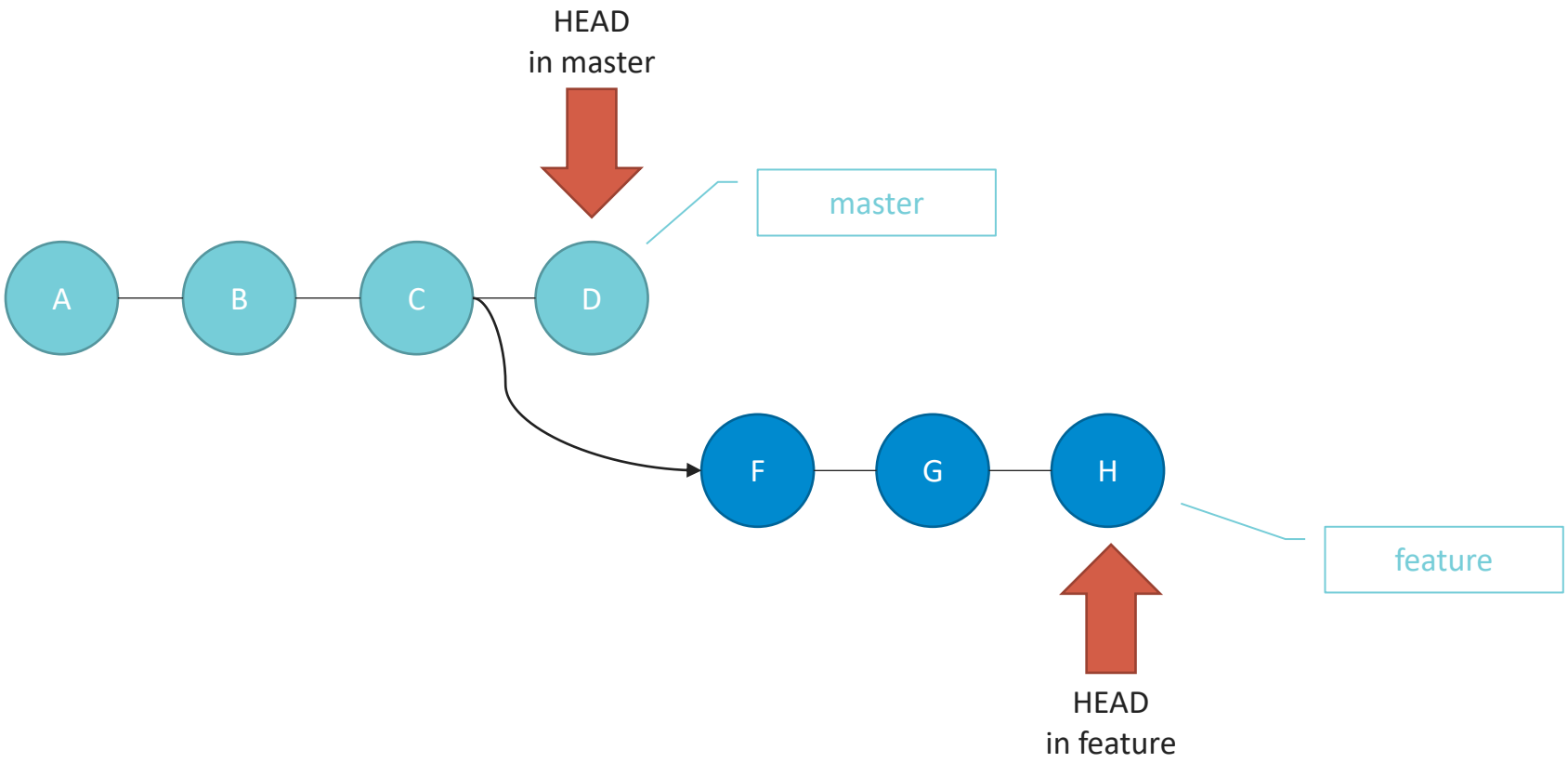
Fast-forward merge



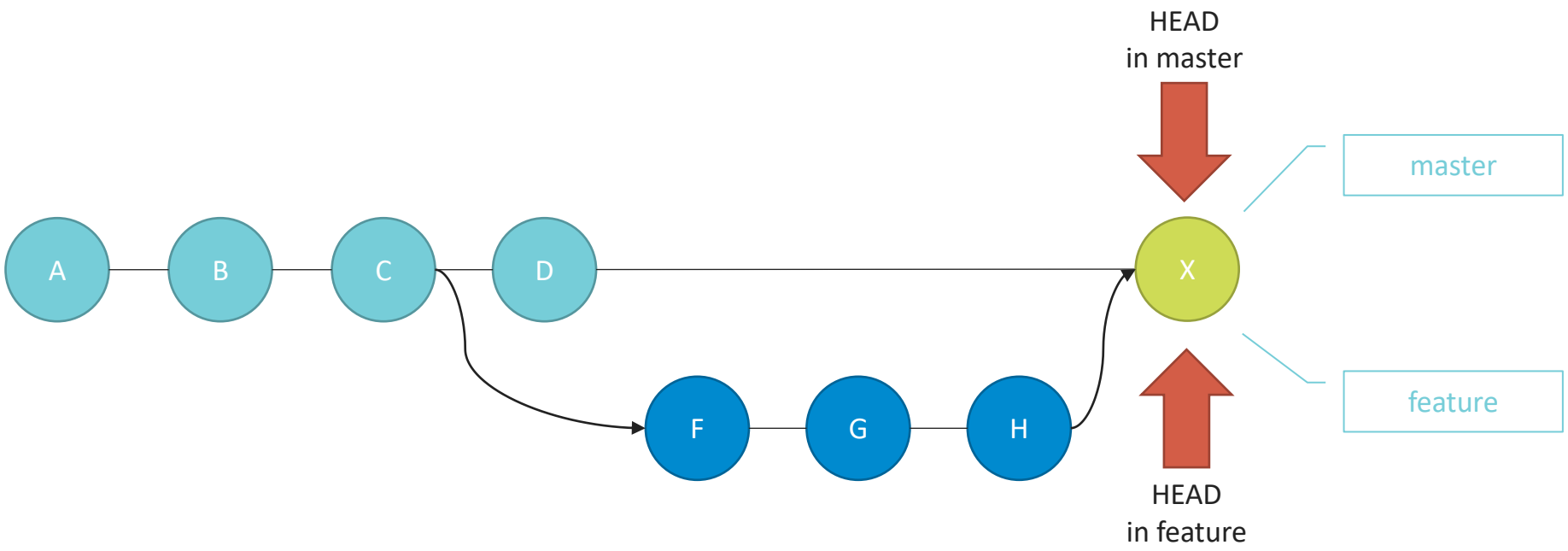
Fast-forward merge



Non fast-forward merge



Non fast-forward merge



Conflict solving

Version Control with Git. DevTestOps training.

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

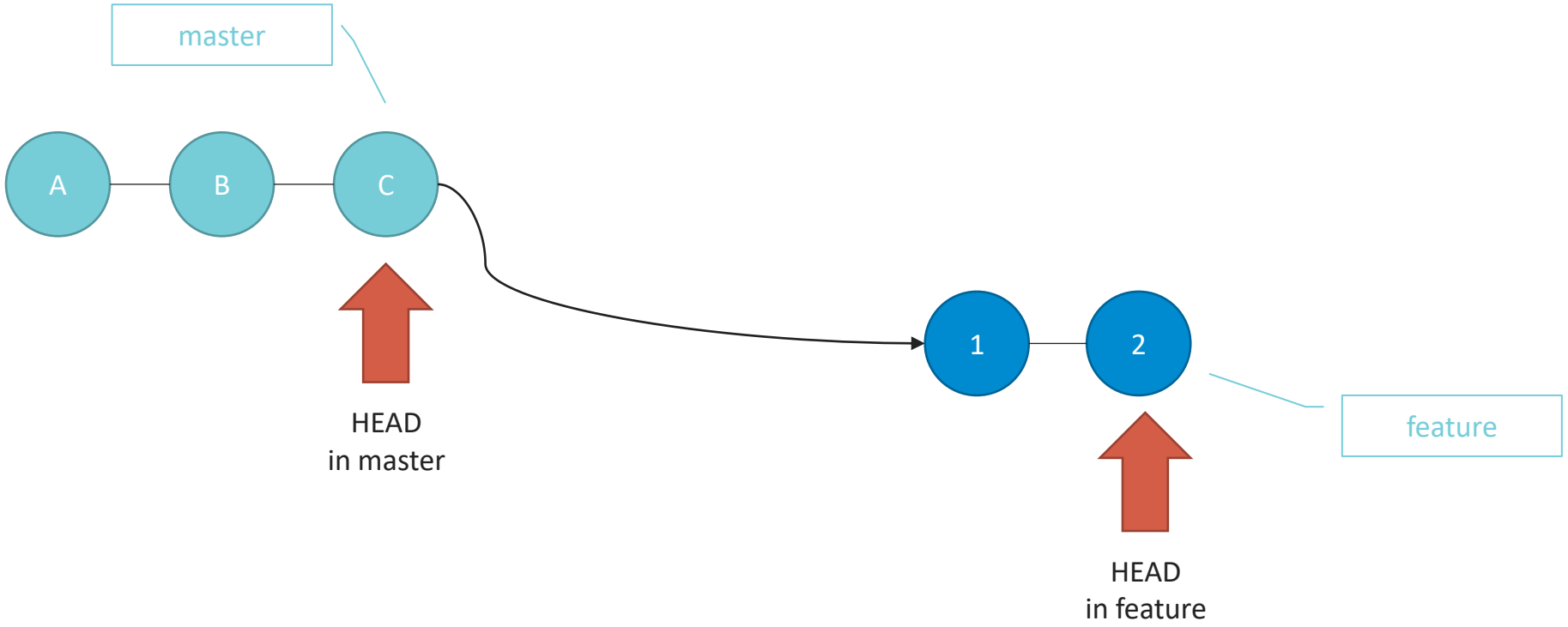
AVOID CONFLICT

- Short commits
- No edits to whitespaces
- Merge often

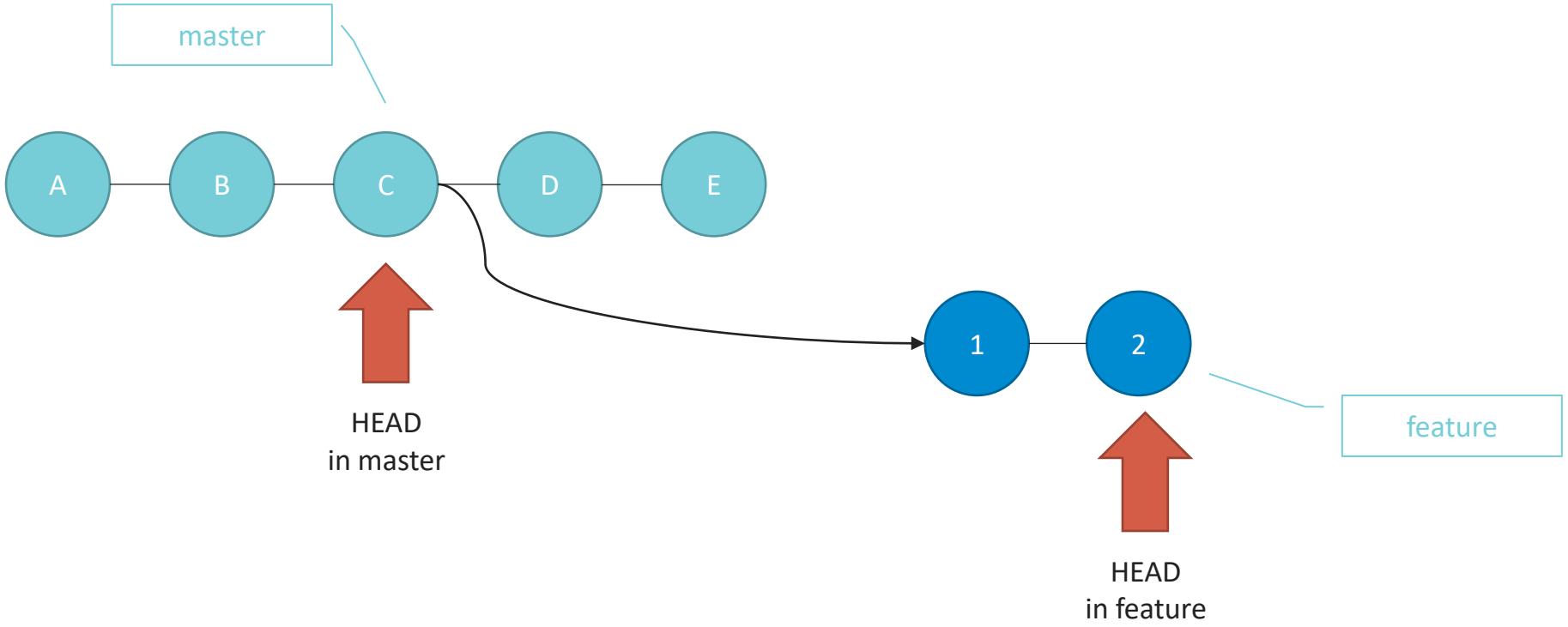
Rebase

Version Control with Git. DevTestOps training.

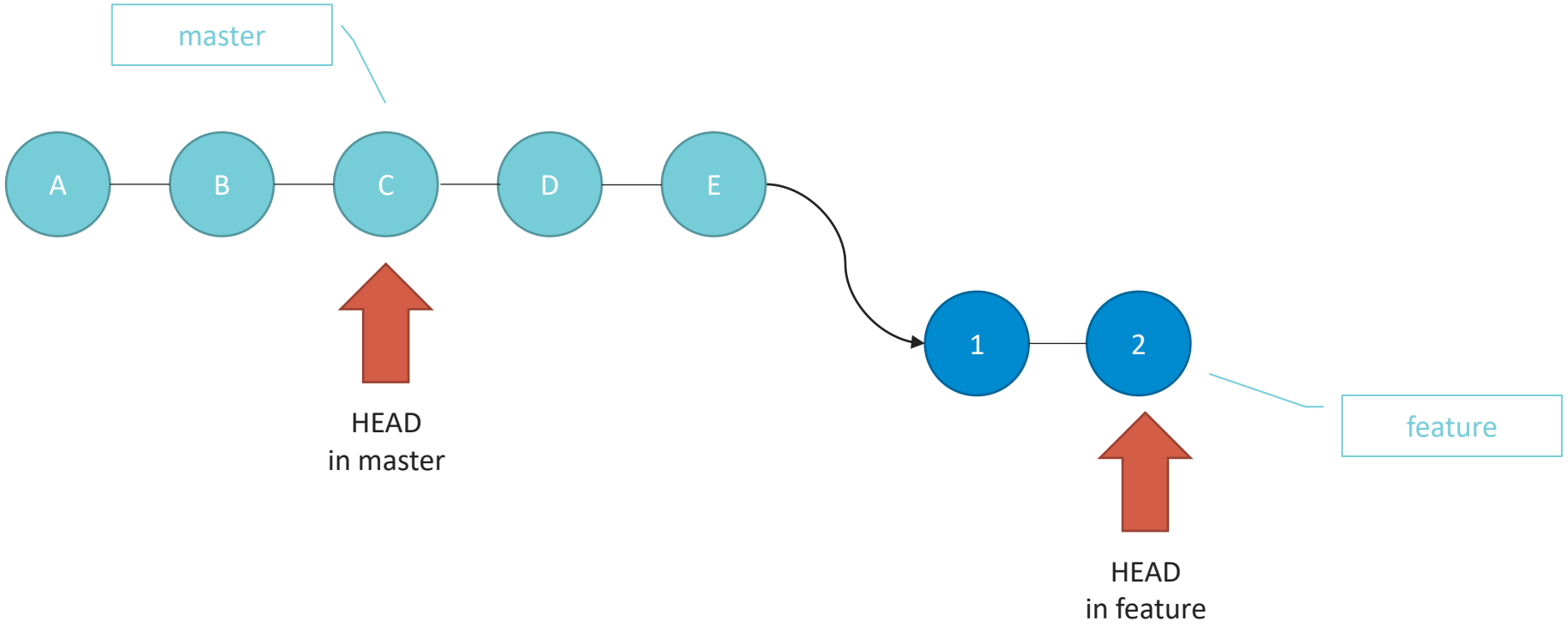
Rebase



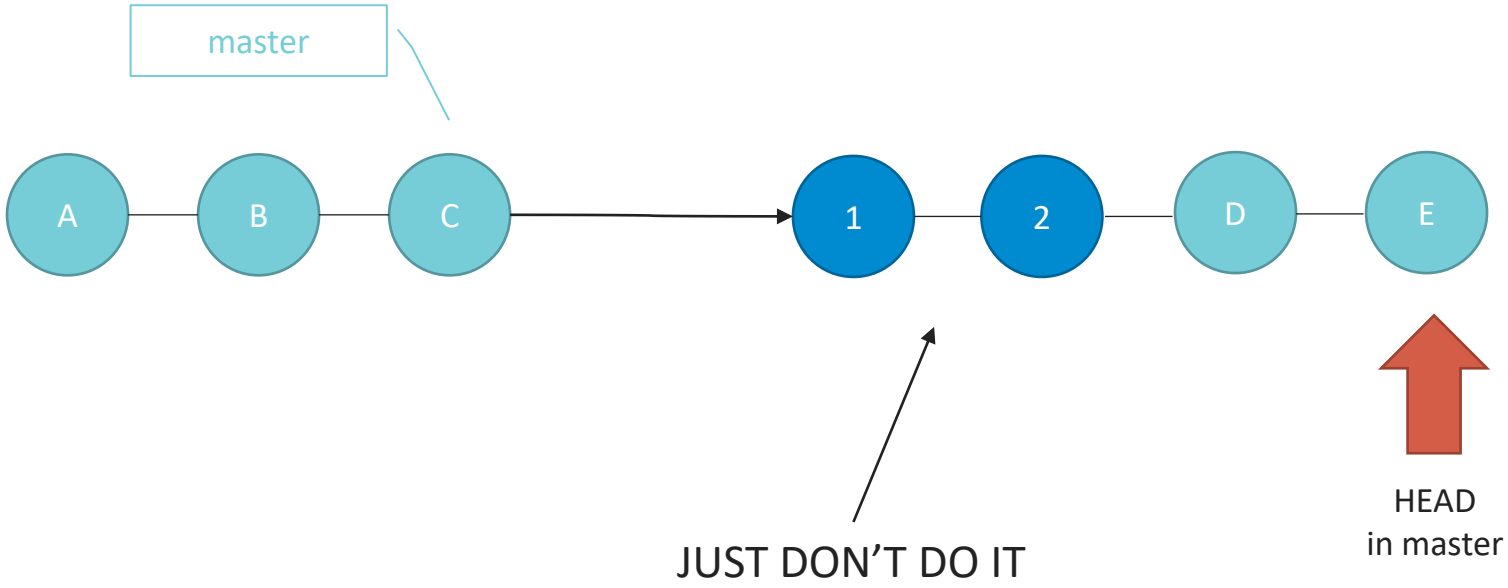
Rebase



Rebase



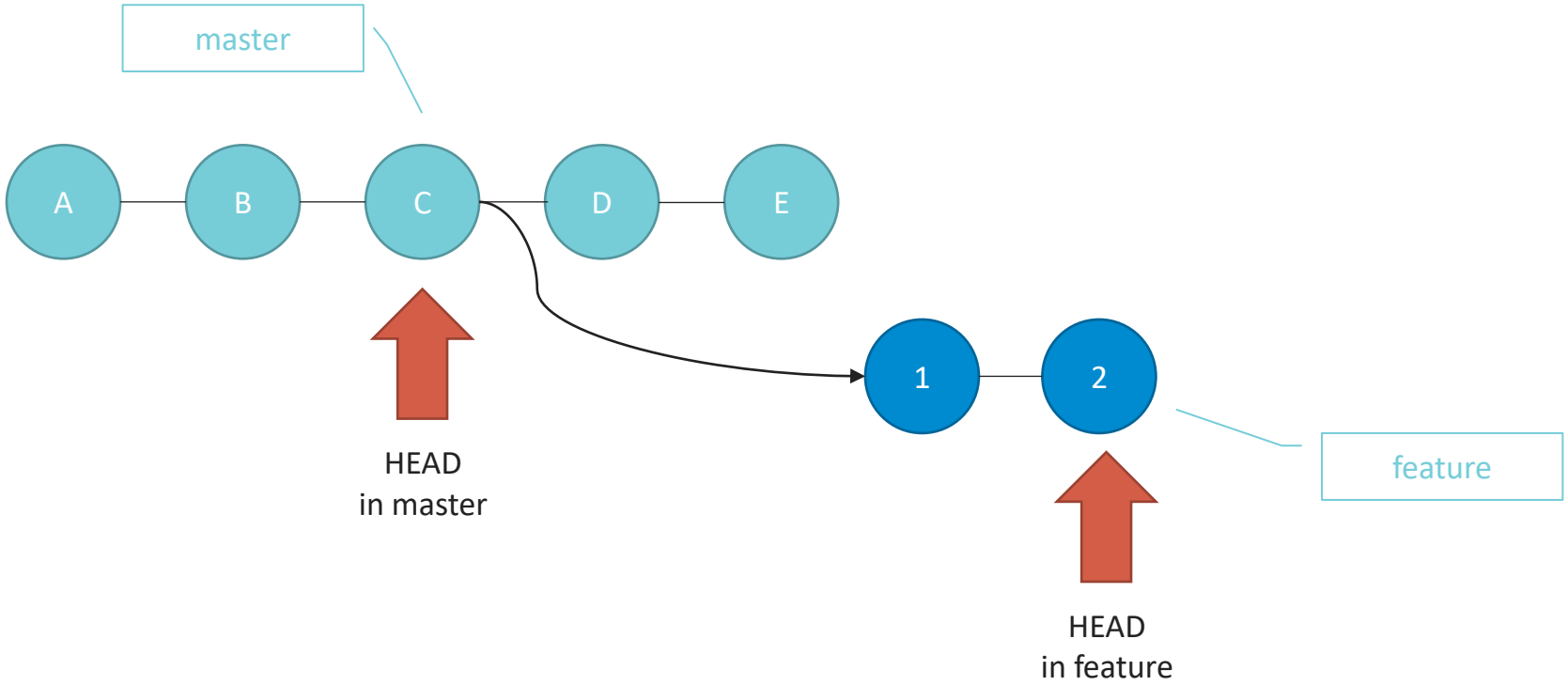
Golden rule of Rebase



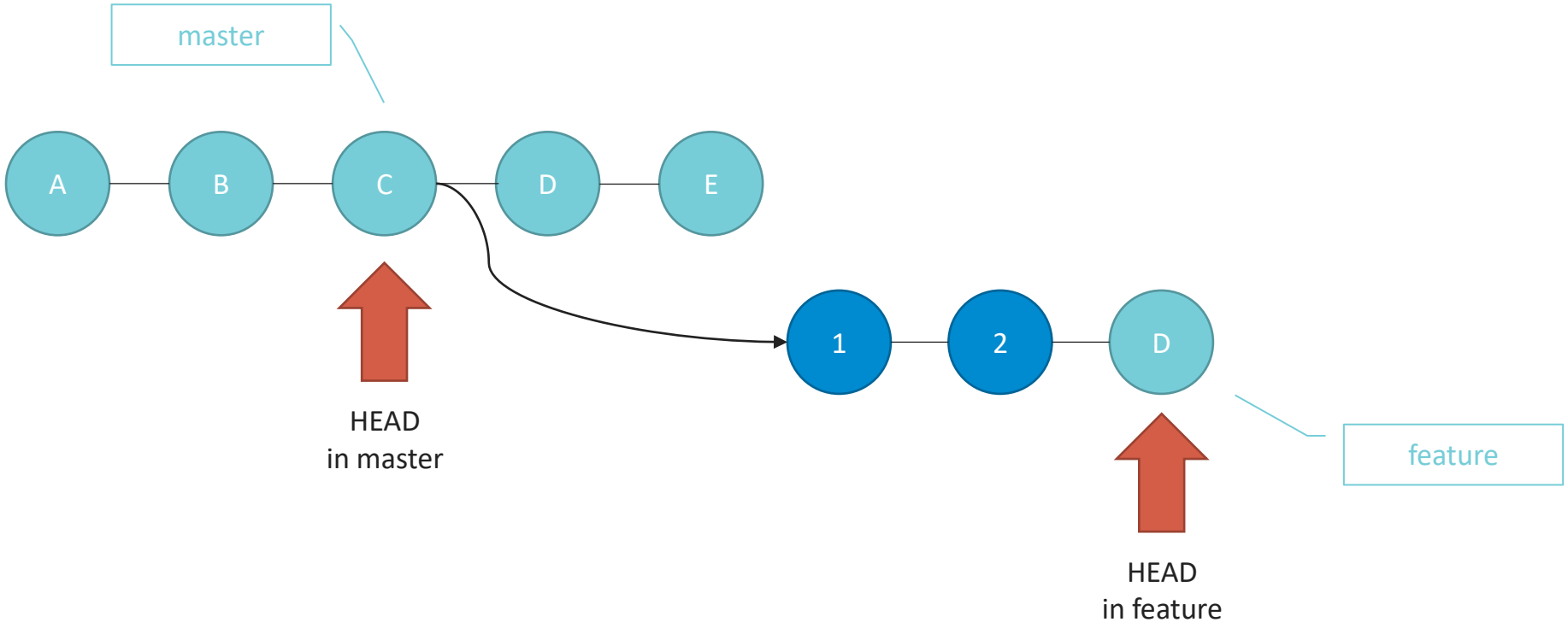
Cherry-pick

Version Control with Git. DevTestOps training.

Cherry pick



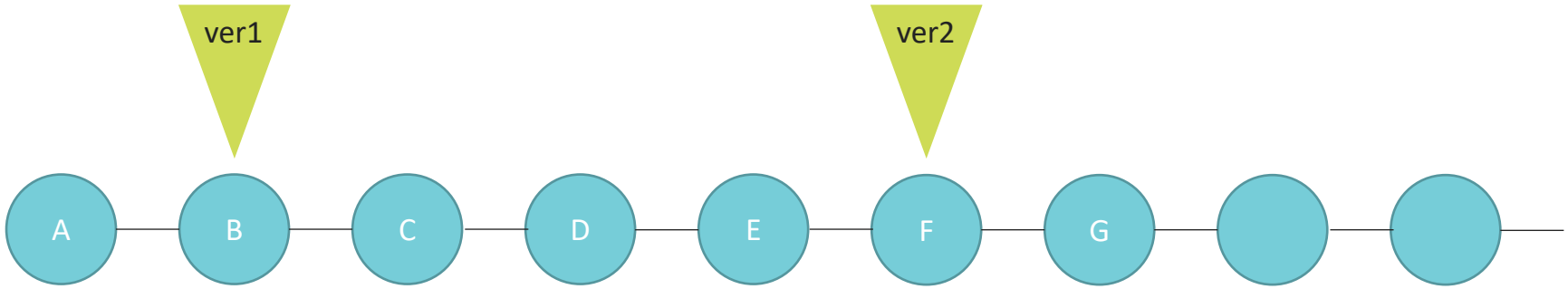
Cherry-pick



Tags

Version Control with Git. DevTestOps training.

Tags



Mark commit with tag

```
git tag ver1
```

View tags

```
git tag –list
```

Push

```
git push --tags
```

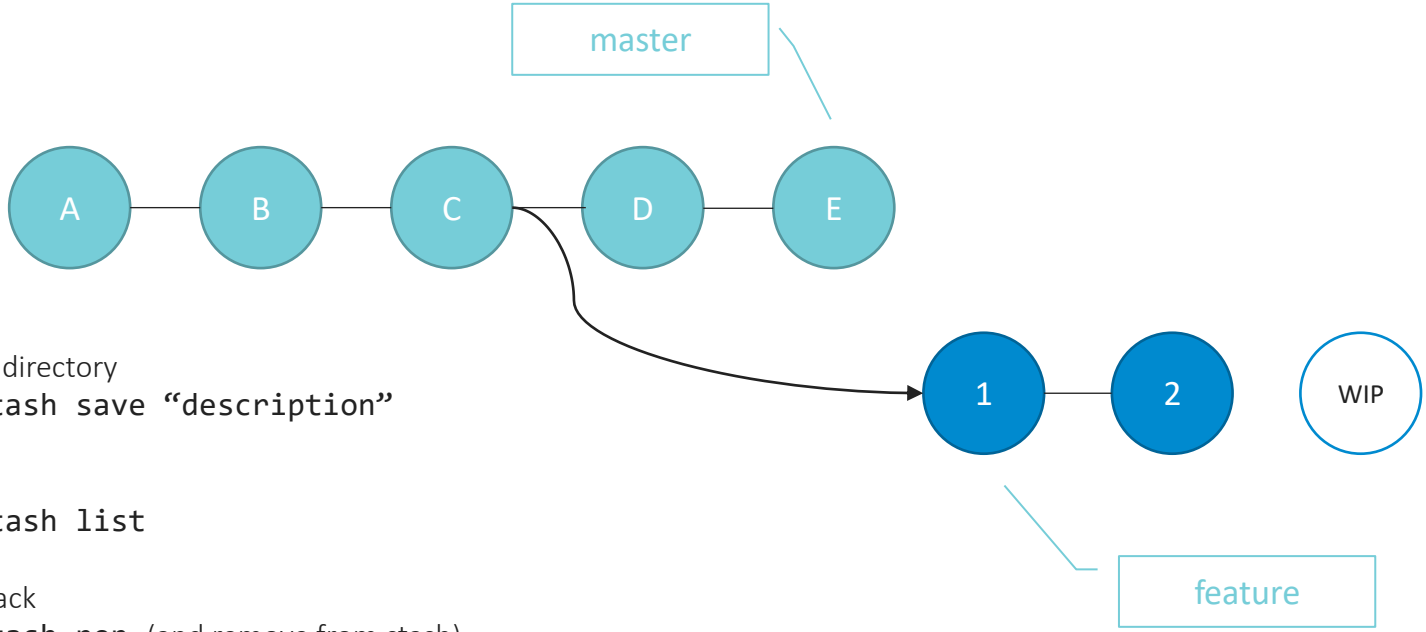
Check it out

```
git checkout ver1
```

Stashing

Version Control with Git. DevTestOps training.

Stash



Save working directory

```
git stash save “description”
```

View stashes

```
git stash list
```

Bring them back

```
git stash pop  (and remove from stash)
git stash apply  (leave in stash)
```

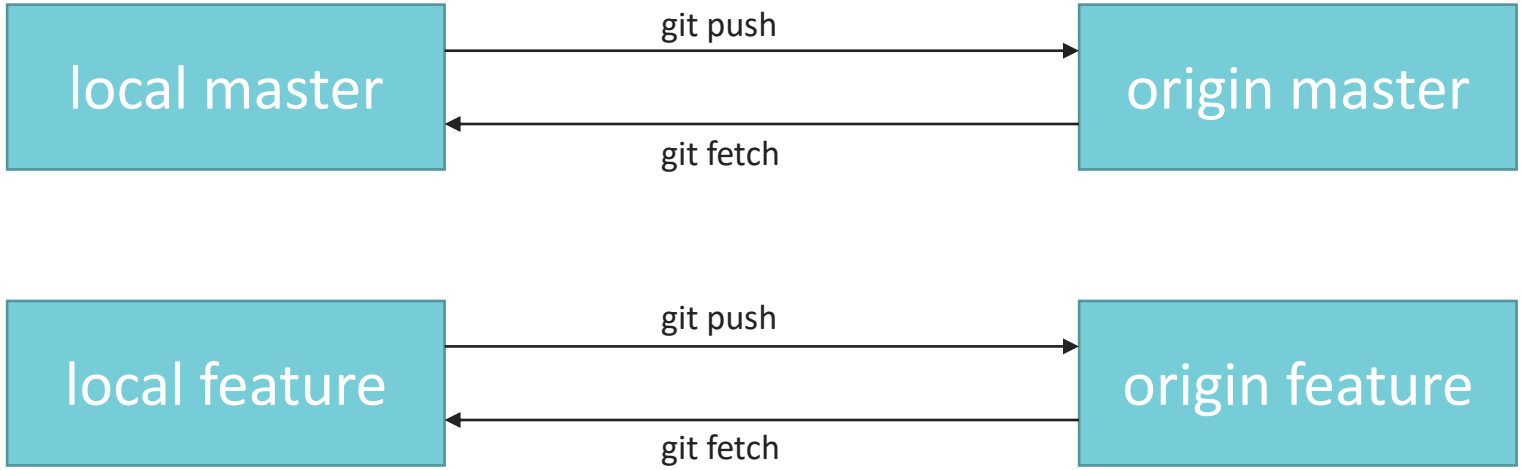
Remove

```
git stash drop  (clear)
```

Remotes

Version Control with Git. DevTestOps training.

Remotes



Add

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

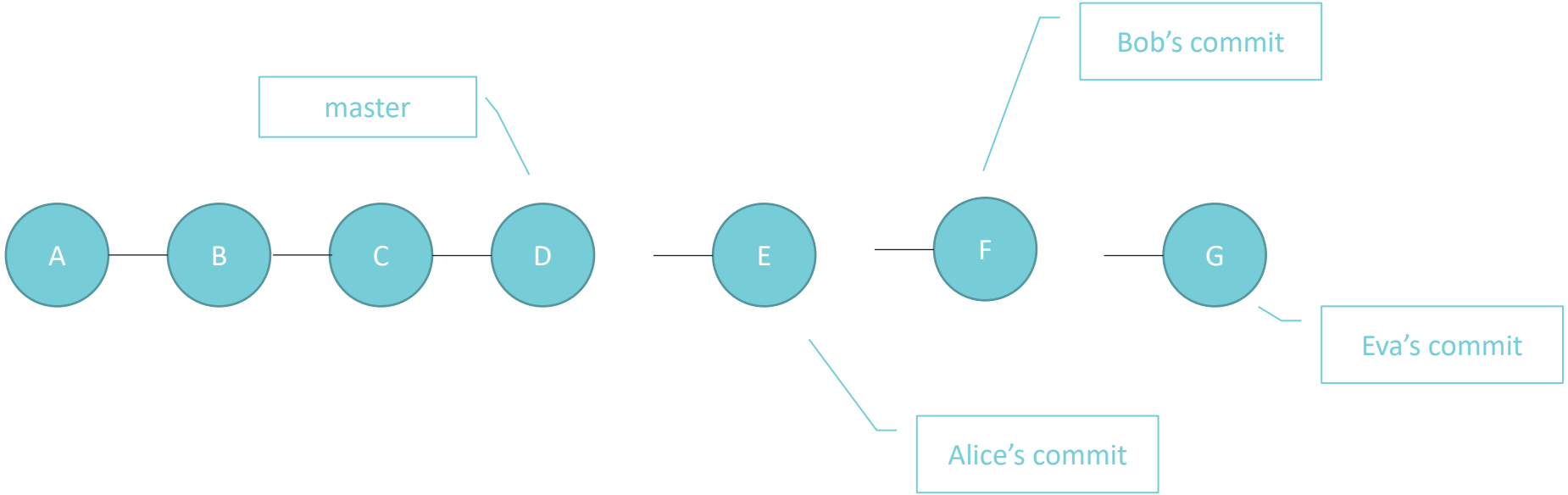
View

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

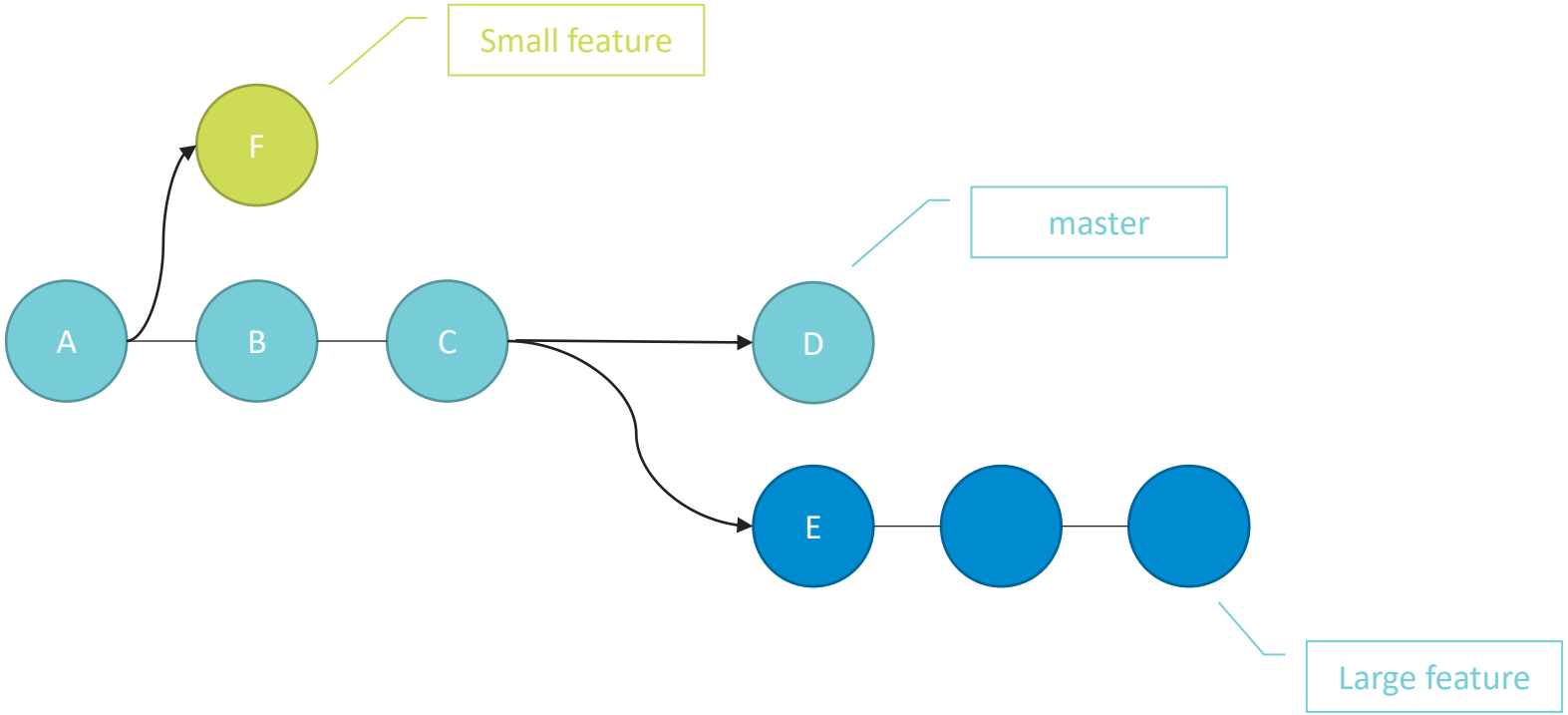
Branching strategies

Version Control with Git. DevTestOps training.

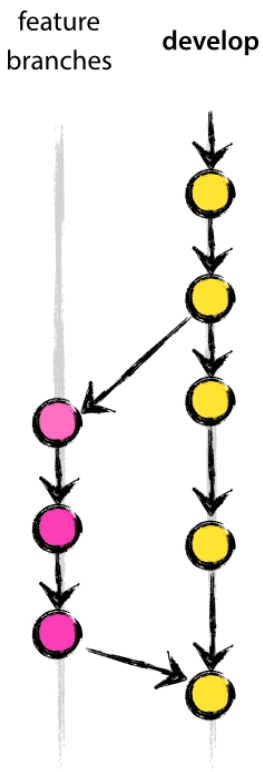
Centralized strategy

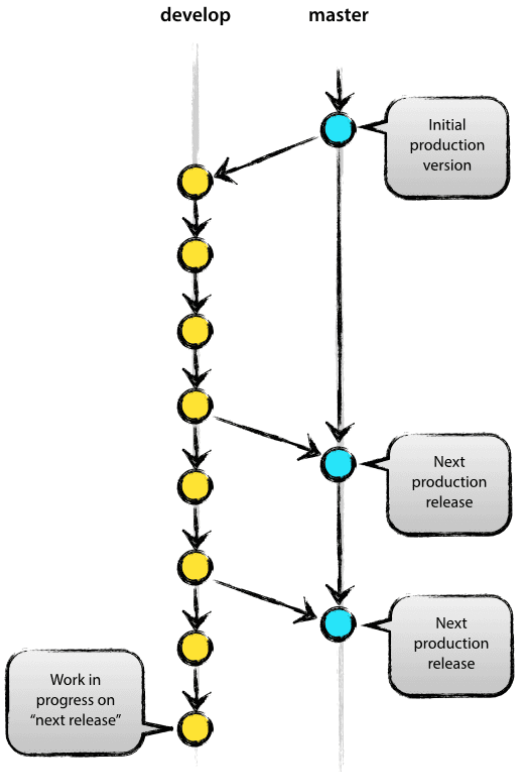


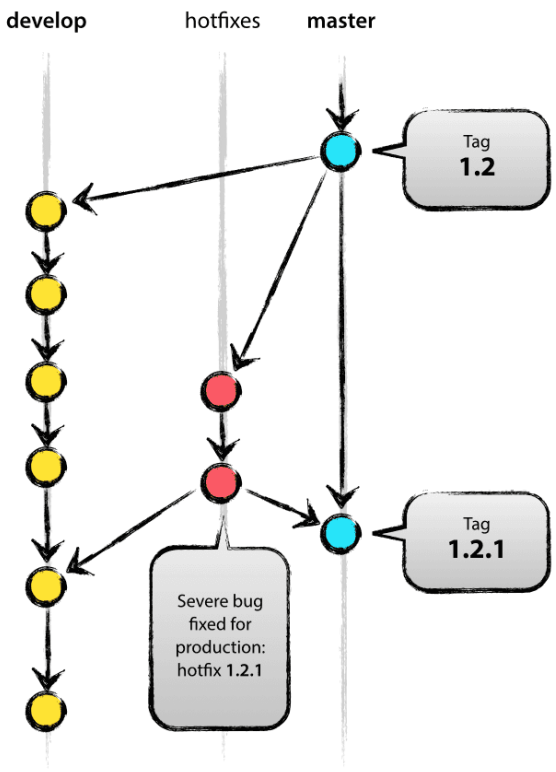
Feature-branch workflow



Gitflow

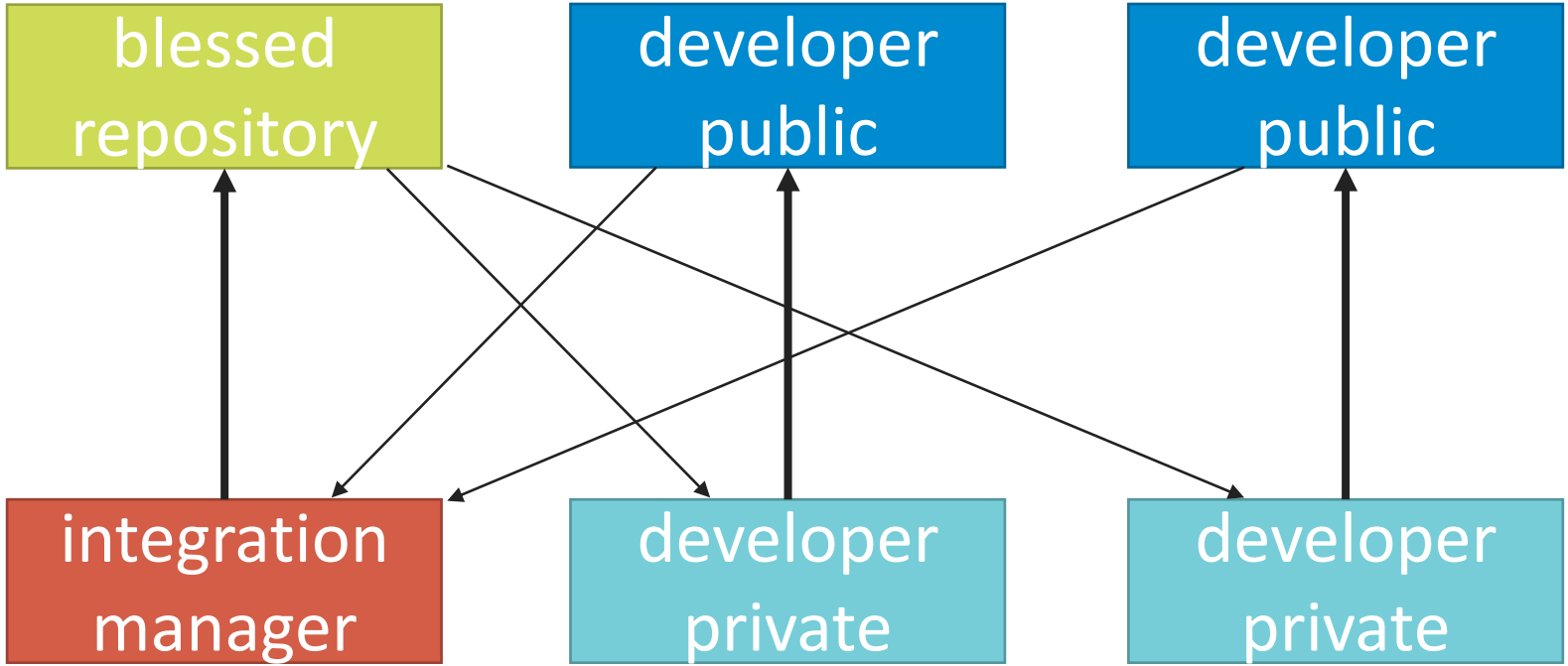




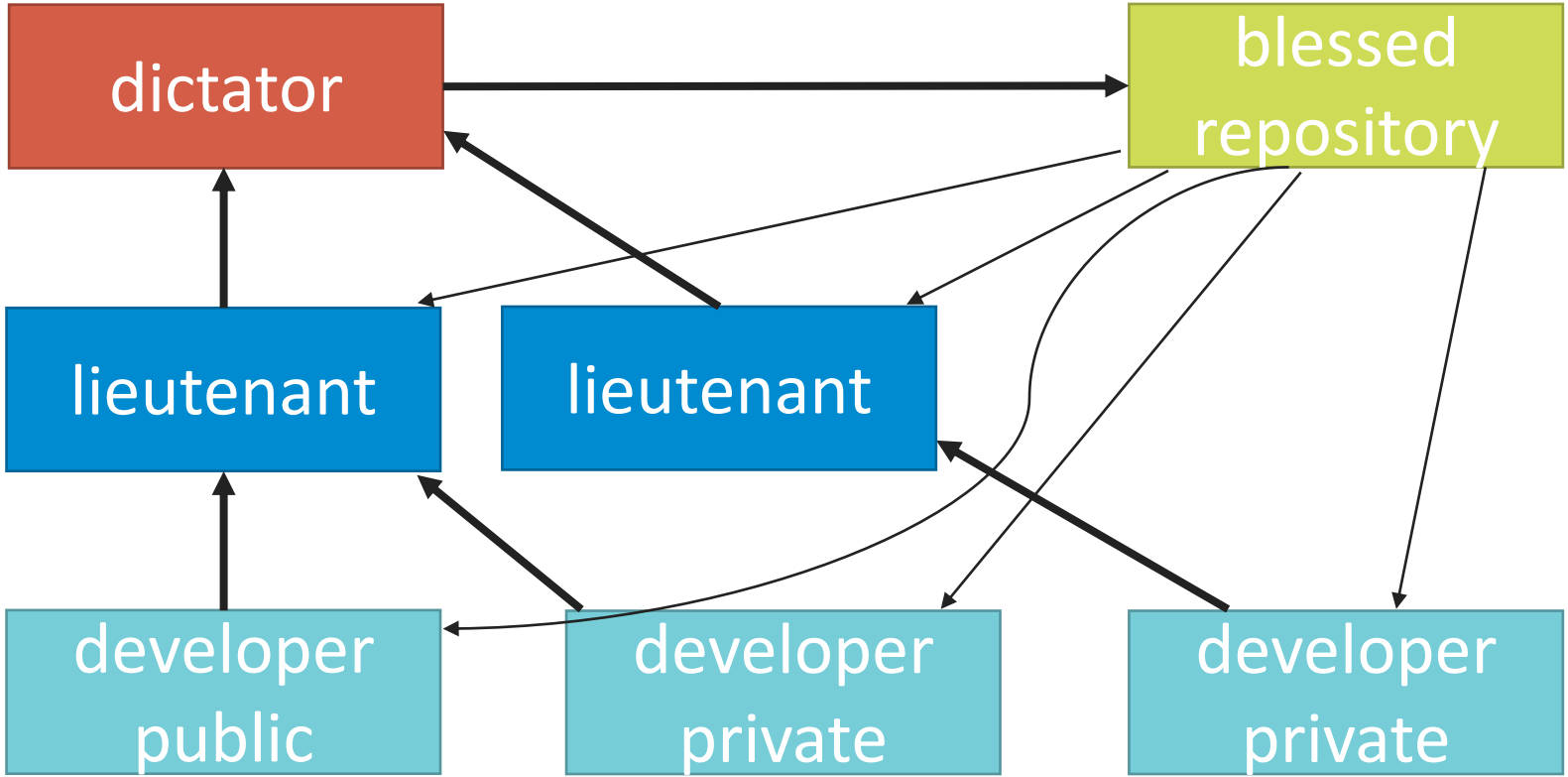


Original post by Vincent Driessen: https://nvie.com/posts/a-successful-git-branching-model/

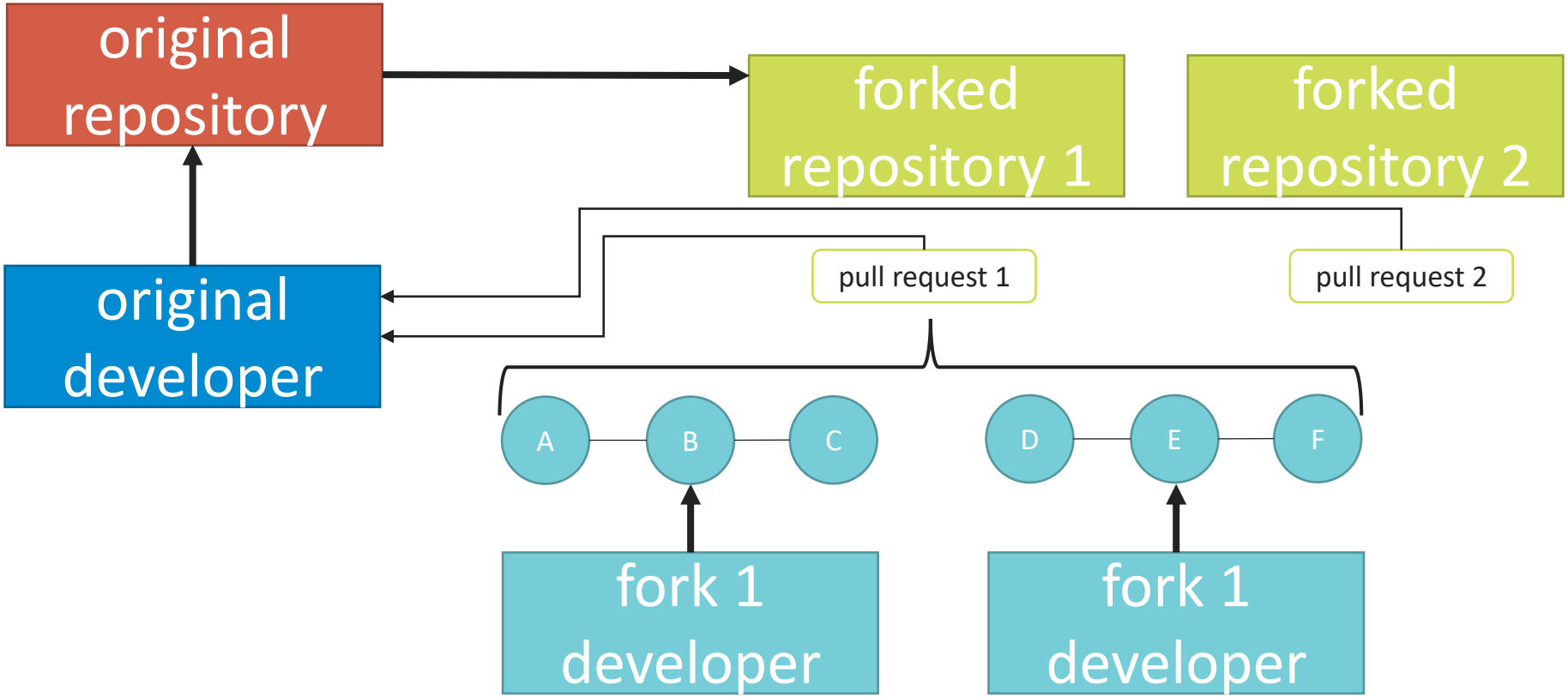
Integration manager workflow



Dictator and Lieutenants workflow



Forking workflow



Inside .git folder

Version Control with Git. DevTestOps training.

Commit, tree, blob

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

Extras

Version Control with Git. DevTestOps training.

Extras

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