

Bitbucket repository

EPAM Java Training

1. Create new account

- Go to <https://bitbucket.org/> and click “Get started”

ATLASSIAN
Bitbucket

Features Integrations Server Data Center Pricing

Log in **Get started**

For the code that drives us.

What will your code do?

Get started for free

Or host it yourself with Bitbucket Server

1. Create new account

- Enter your email address



[Features](#)

[Integrations](#)

[Server](#)

[Data Center](#)

[Pricing](#)

[Log in](#)

[Get started](#)

Create your account

Enter your email address

Continue

1. Create new account

- Complete your account details: Full Name and Password.
- Check “I’m not robot” and click “Continue”



[Features](#)

[Integrations](#)

[Server](#)

[Data Center](#)

[Pricing](#)

[Log in](#)

[Get started](#)

Check your inbox to verify your email

1. Create new account

- Create unique username for Bitbucket Cloud



Features

Integrations

Server

Data Center

Pricing

Log in

Get started

Almost done

Create a unique username for Bitbucket Cloud

bitbucket.org /

Continue

2. Create new repository

- Create a repository

Welcome to Bitbucket, Aleksandr



Flying solo?

Create your first repository and start bringing your ideas to life.

Create a repository



Plan to work with others?

Add an existing repository or create a new one, then start working better, together.

Create a team

New to Git? Check out our tutorials

2. Create new repository

- Repository name. Use your First Name and Surname. Like “aivanov”

Create a new repository [Import repository](#)

Repository name*

Access level This is a private repository

Repository type Git
 Mercurial

Advanced settings

Description

Forking

Project management Issue tracking
 Wiki

Language

Integrations Enable Hipchat notifications

2. Create new repository

- Bitbucket repository successfully was created.

The screenshot shows the Bitbucket repository settings page for a repository named 'java' owned by 'areshetnev'. The 'Settings' menu item is highlighted in the left sidebar. The 'User and group access' section is selected in the 'GENERAL' category. Under 'Users', a user named 'javaMog' is listed with 'Read' permissions. A dropdown menu is open for 'javaMog', showing a group named 'javamog' with 'owner' permissions. The 'Groups' section is currently empty, showing a 'Select a group' dropdown and 'Read' permissions.

Settings

GENERAL

- Repository details
- User and group access**
- Access keys
- Username aliases

WORKFLOW

- Branch permissions
- Default reviewers
- Webhooks
- Links

FEATURES

- Git LFS
- Wiki
- Issue tracker

User and group access

Here's where you grant users and [groups](#) access to this repository. For a list of all users with access to any of your private repositories, see which users count towards your bill on the [Users on plan](#) page. [Learn more](#)

Users

javaMog	Read	Add
javamog javamog	owner	

Groups

Select a group	Read	Add
----------------	------	-----

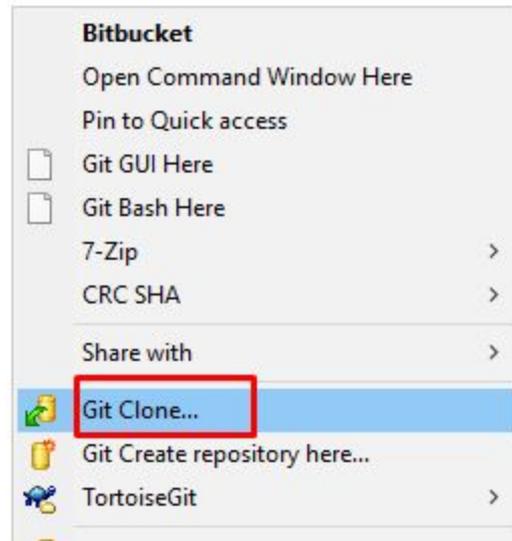
2. Create new repository

- Add default reviewers for new pull requests.

The screenshot shows the GitHub repository settings for user 'areshetnev'. The 'Settings' menu is open, and the 'Default reviewers' option is highlighted. The 'Default Reviewers' section contains a text box with the message: 'The following users will be added as reviewers on new pull requests by default.' Below this, there is an input field containing 'javamog' and an 'Add' button. A dropdown menu is open below the input field, showing a preview of the reviewer: 'epam javamog javamog'. Red boxes highlight the 'Settings' menu item, the 'Default reviewers' section, the 'Default Reviewers' title, the explanatory text box, the input field, the 'Add' button, and the dropdown menu.

3. Checkout repository

- Install TortoiseGit
- Right click and choose “Git Clone...”



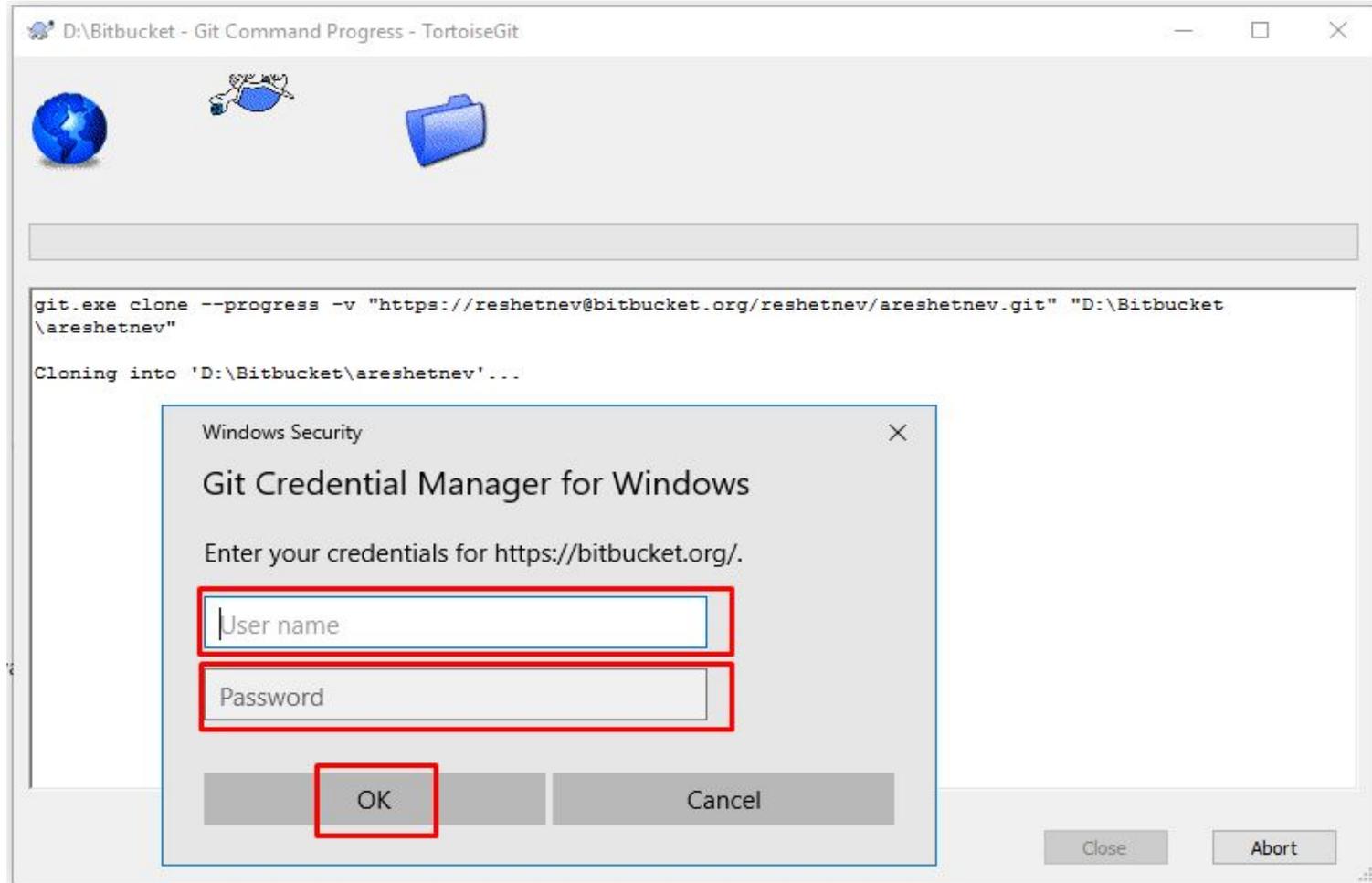
3. Checkout repository

- Choose correct checkout directory if needed

The image shows a Bitbucket repository overview for 'areshetnev' and a TortoiseGit 'Git clone' dialog box. In the Bitbucket interface, the 'Overview' tab is selected in the left sidebar, and the repository URL 'https://reshetnev@bitbucket.org/reshe...' is displayed in the top right. The TortoiseGit dialog is open, showing the 'Clone Existing Repository' section. The 'URL' field contains 'https://reshetnev@bitbucket.org/reshetnev/areshetnev.git' and the 'Directory' field contains 'D:\bitbucket\areshetnev'. The 'OK' button is highlighted in the bottom right corner of the dialog.

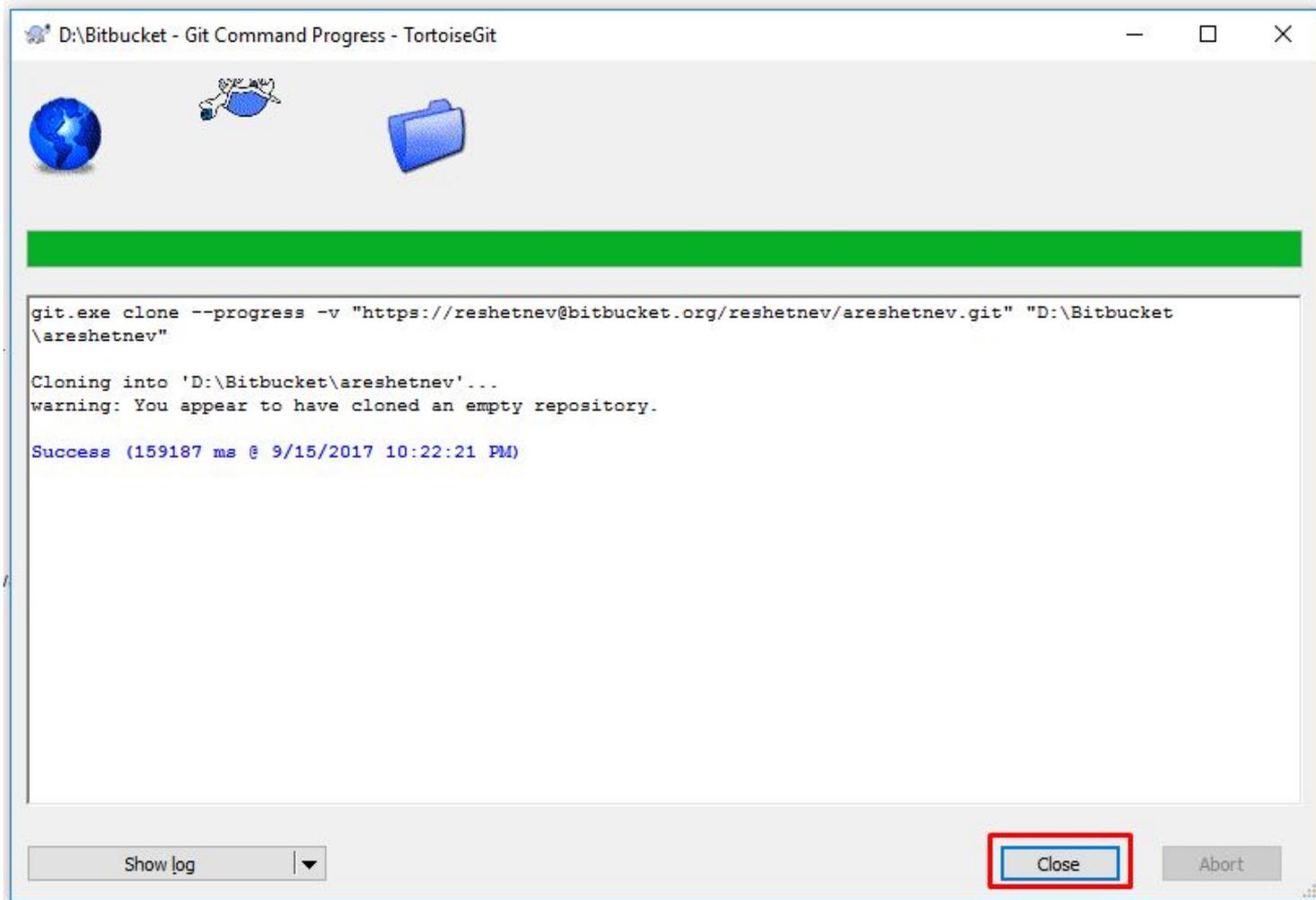
3. Checkout repository

- Enter credentials and click on save authentication checkbox



3. Checkout repository

- Checkout finished

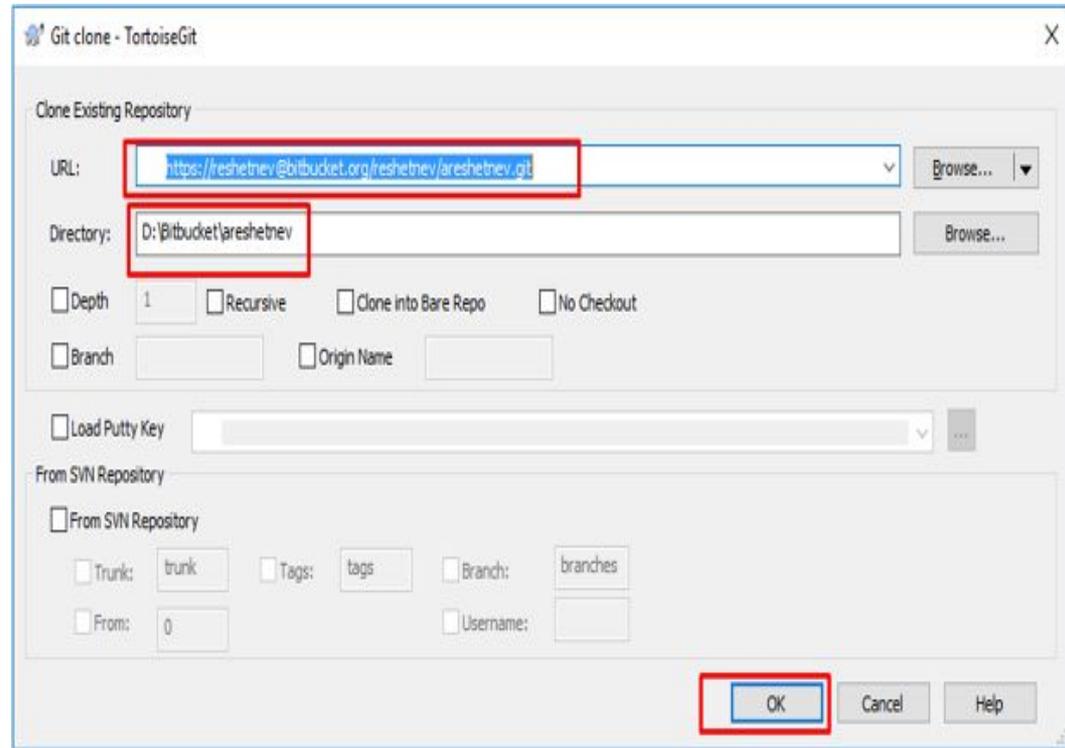
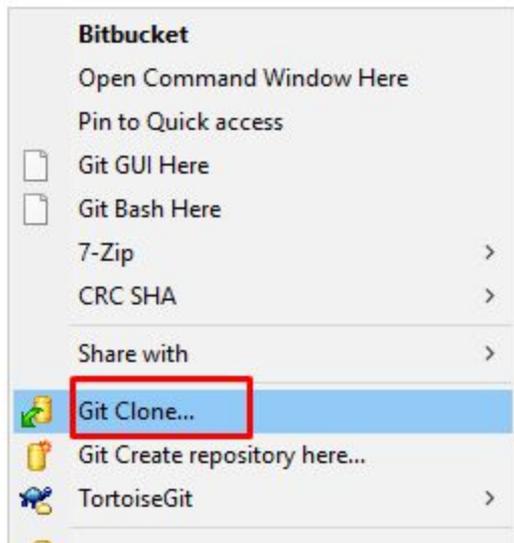


4. Pro Git

- Book “Pro Git”, Scott Chacon
- 1. <https://git-scm.com/book/ru/v1>
- 2. <https://git-scm.com/book/en/v2>
- 3. <https://github.com/progit/progit2>

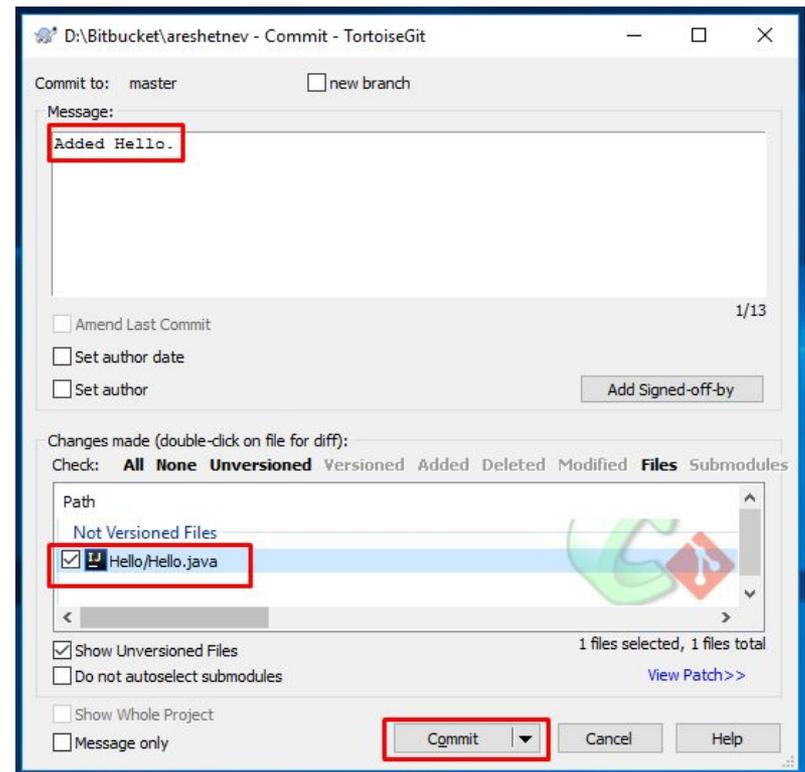
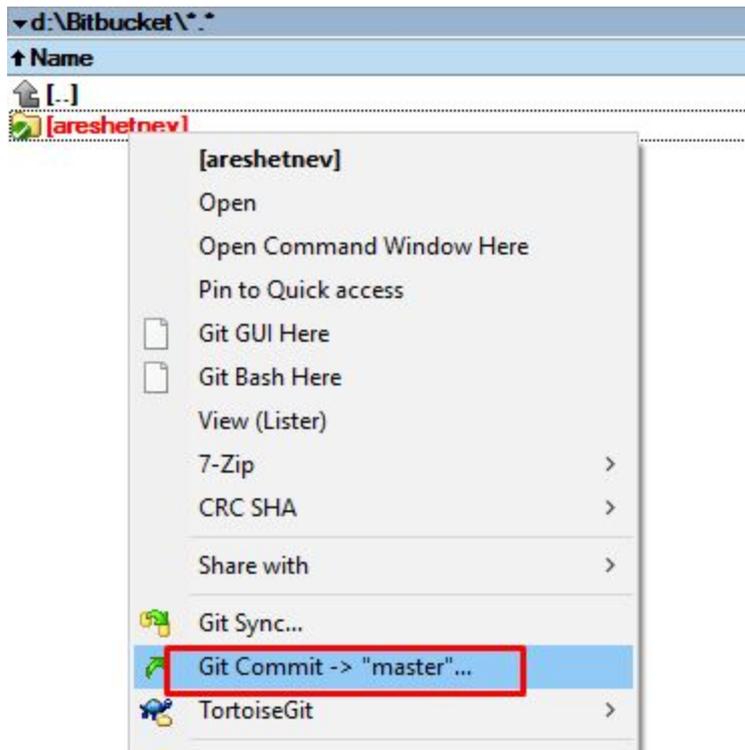
4. Pro Git

• Cloning an Existing Repository



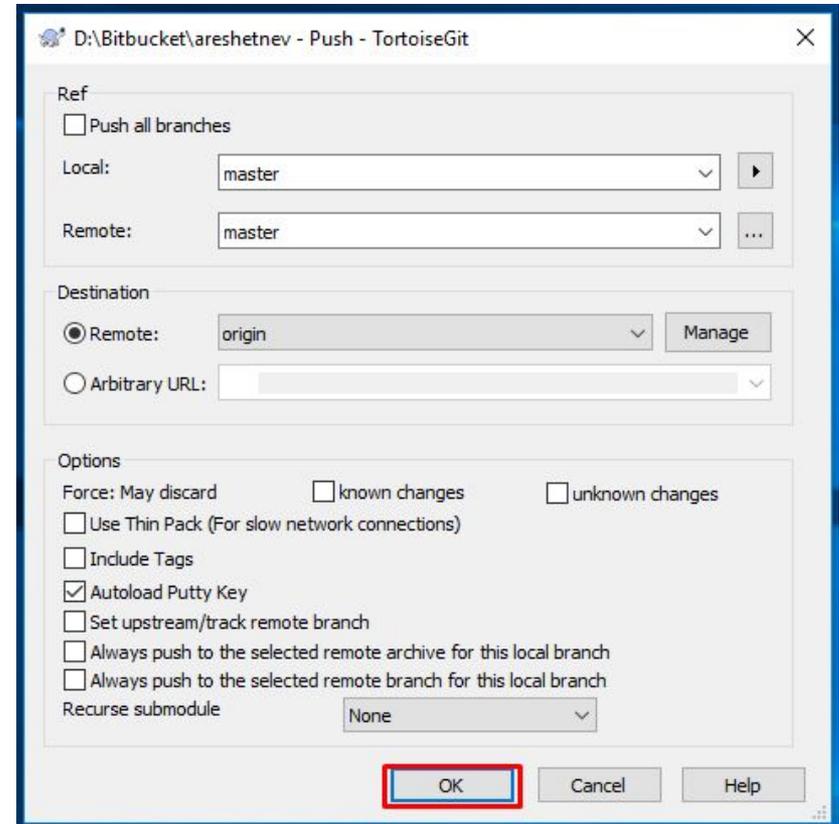
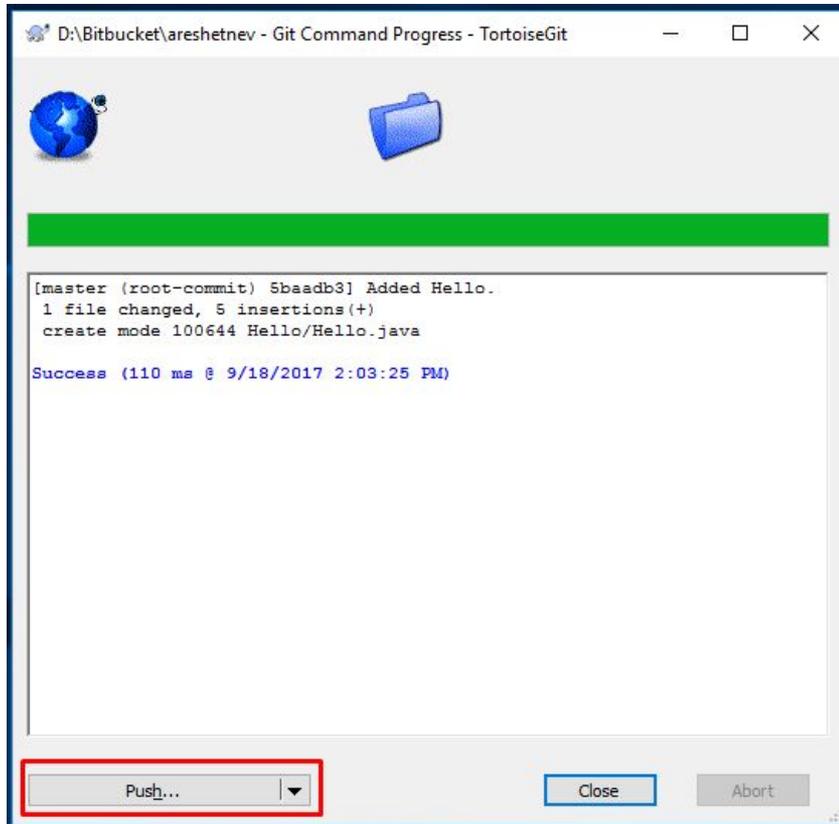
4. Pro Git

- Recording Changes to the local repository



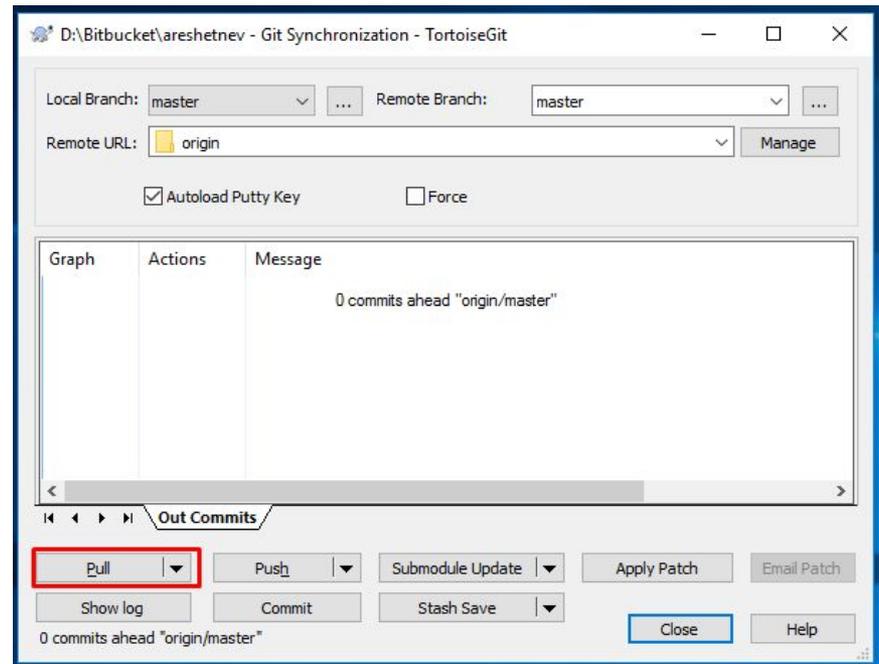
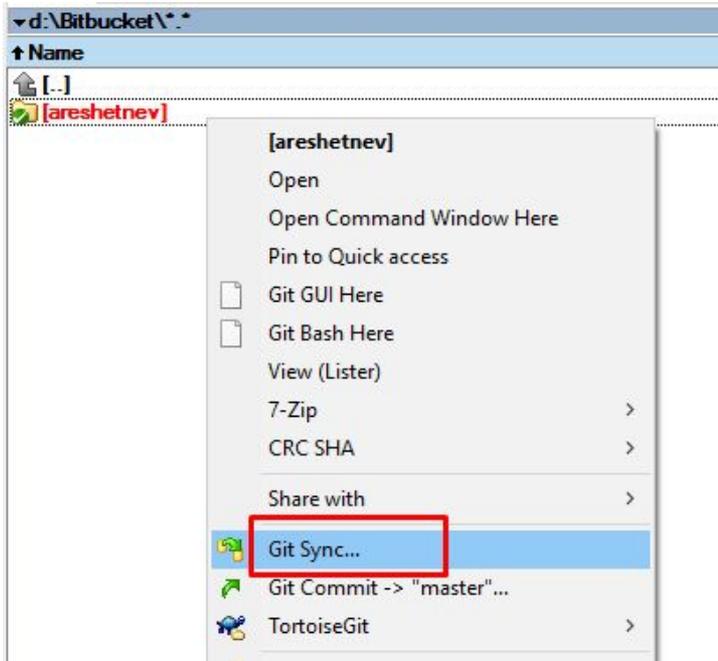
4. Pro Git

- Recording Changes to the remote repository



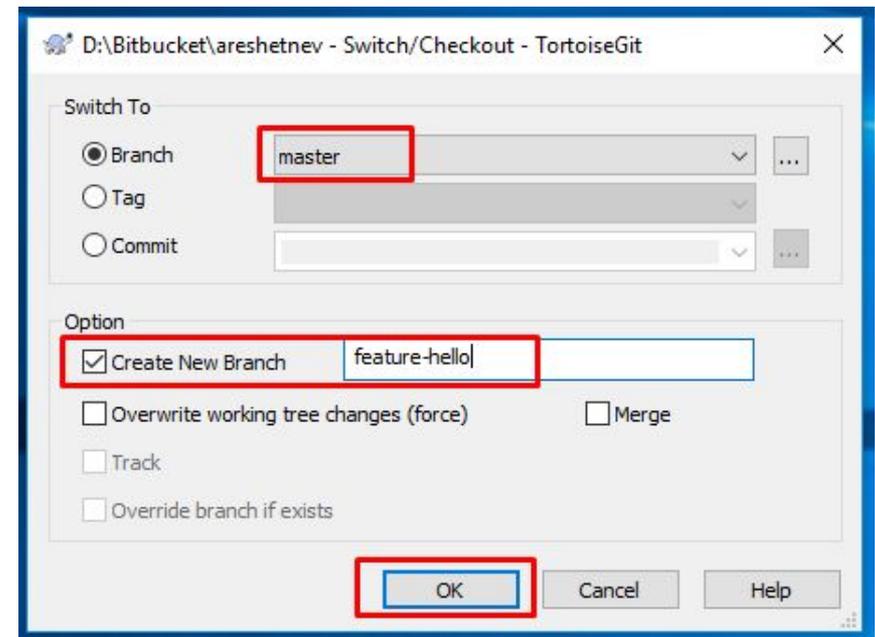
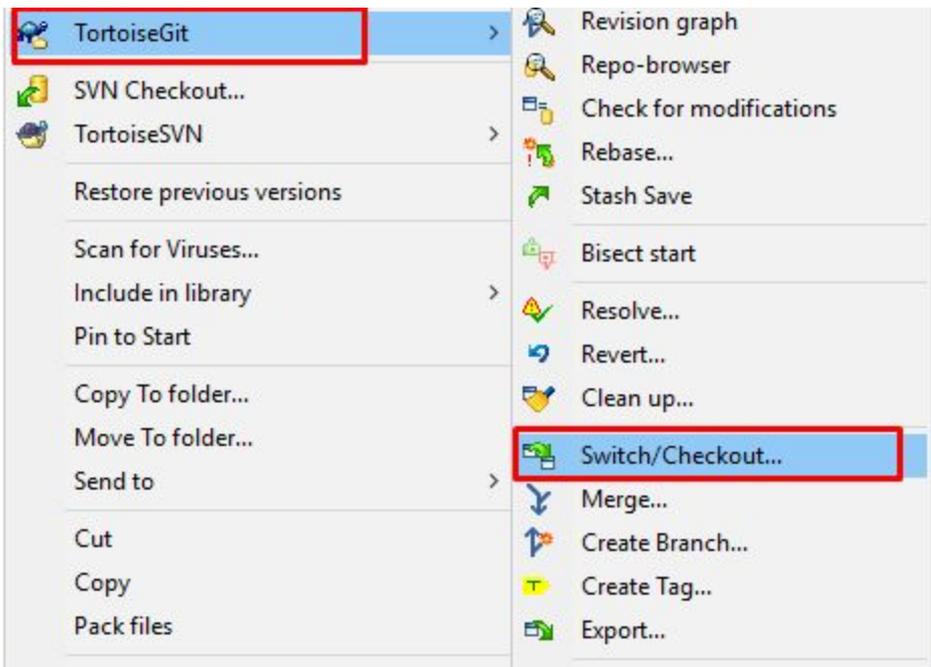
4. Pro Git

- Getting updates from remote repository



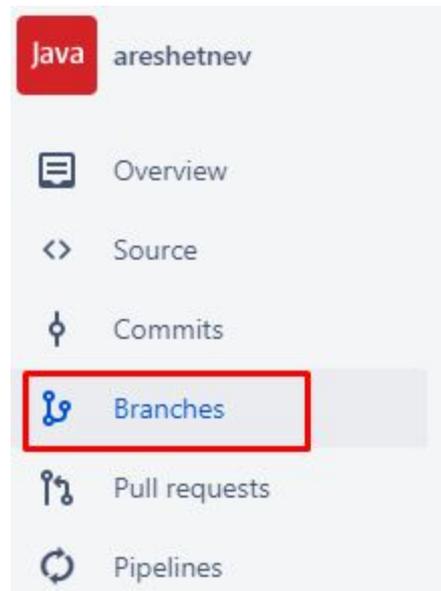
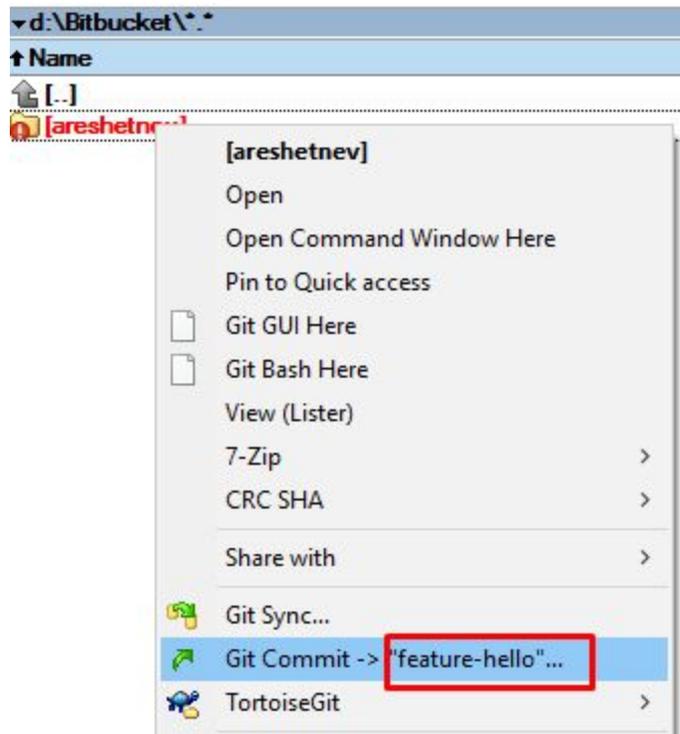
4. Pro Git

- Creating branch



4. Pro Git

- Make sure that you commit in the created branch.
- After commit & push (check-in), Create pull request.



Aleksandr Reshetnev / areshetnev

Branches

FILTERS: **Active** Merged

Branch

master **MAIN BRANCH**

feature-hello

4. Pro Git

- Creating pull request

The screenshot shows a Git web interface for a repository named 'areshetnev'. The left sidebar contains navigation links: Overview, Source, Commits, Branches (selected), Pull requests, Pipelines, Downloads, Boards, and Settings. The main content area shows the 'feature-hello' branch selected. A diagram illustrates the branch structure: a 'master' branch with a commit labeled 'change destination' and a 'feature-hello' branch branching off from it. Action buttons include 'Check out', 'View source', 'Merge', and 'Create pull request' (highlighted with a red box). Below the diagram, there are tabs for 'Diff', 'Commits', and 'Merged pull requests'. The 'Commits' tab is active, displaying a table of commit history.

Author	Commit	Message	Date	Builds
Aleksandr Resh...	af94b5f	Refactored Hello	11 minutes ago	

4. Pro Git

The screenshot displays the GitHub 'Create a pull request' interface. On the left is a sidebar with navigation options: Overview, Source, Commits, Branches, Pull requests (highlighted), Pipelines, Downloads, Boards, and Settings. The main content area shows the user 'Aleksandr Reshetnev / areshetnev' and the 'Pull requests' section. The 'Create a pull request' form includes:

- Repository: reshetnev / areshetnev
- Source branch: feature-hello
- Target branch: master
- Title: Refactored Hello
- Description: Homework for Lecture 1
- Reviewer: javamog
- Close branch: Close **feature-hello** after the pull request is merged
- Button: Create pull request

At the bottom, there are tabs for 'Diff' and 'Commits'. The 'Commits' tab is active, showing a table with the following data:

Author	Commit	Message
Aleksandr Resh...	af94b5f	Refactored Hello

4. Pro Git

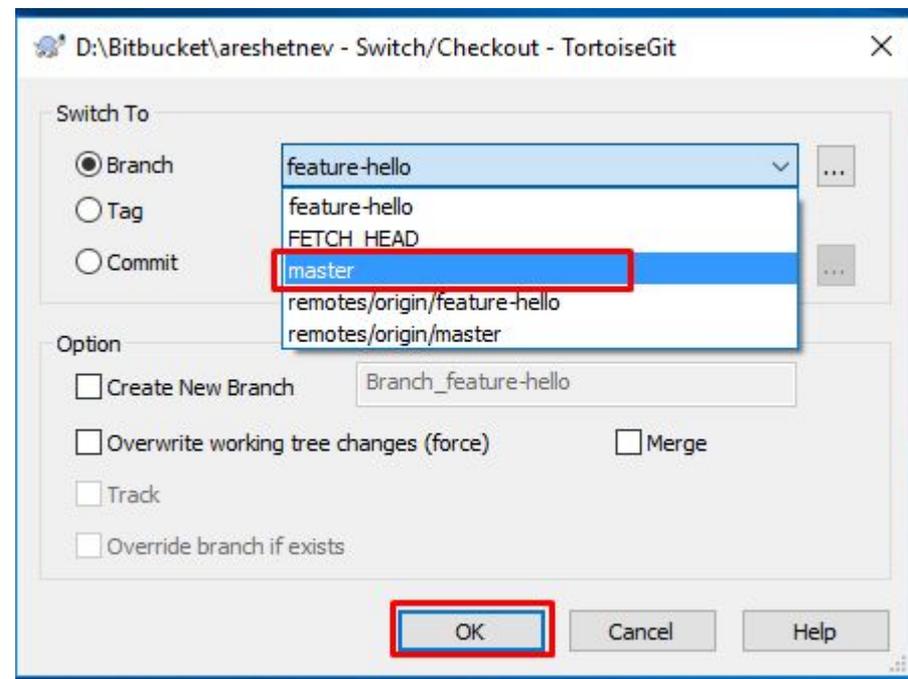
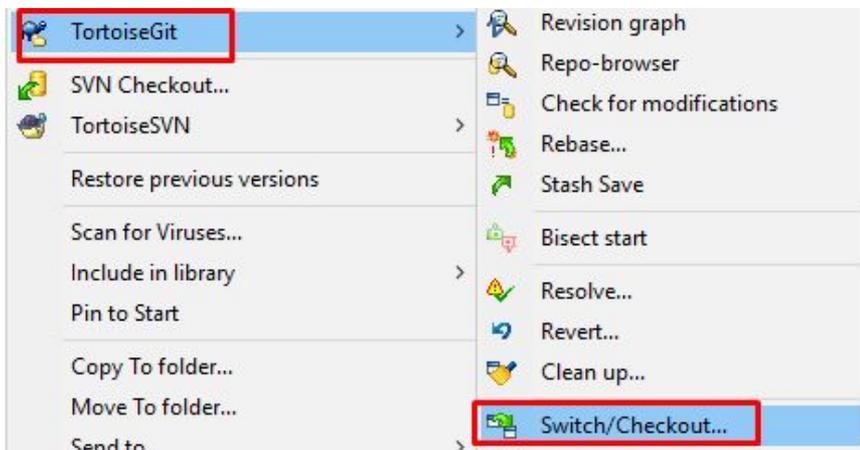
- Merge *only approved* pull request

The screenshot shows a GitHub pull request page for the repository 'areshetnev / areshetnev'. The pull request is titled 'Refactored Hello' and is currently 'OPEN'. It shows a diff for 'Hello/Hello.java' with 3 lines added and 3 lines removed. The author is Aleksandr Reshetnev. A reviewer, 'javamog', has approved the pull request 39 seconds ago. The 'Merge' button is highlighted with a red box. The 'Description' field contains the text 'Homework for Lecture 1'. The 'Files changed' section shows 'Hello/Hello.java' with a diff view. The diff shows the following code changes:

```
1 1 public class Hello {
2 -     public static void main(String[] args) {
3 -         System.out.println("Hello, World!");
4 -     }
2 +     public static void main(String[] args) {
3 +         System.out.println("Hello, World!");
4 +     }
5 5 }
```

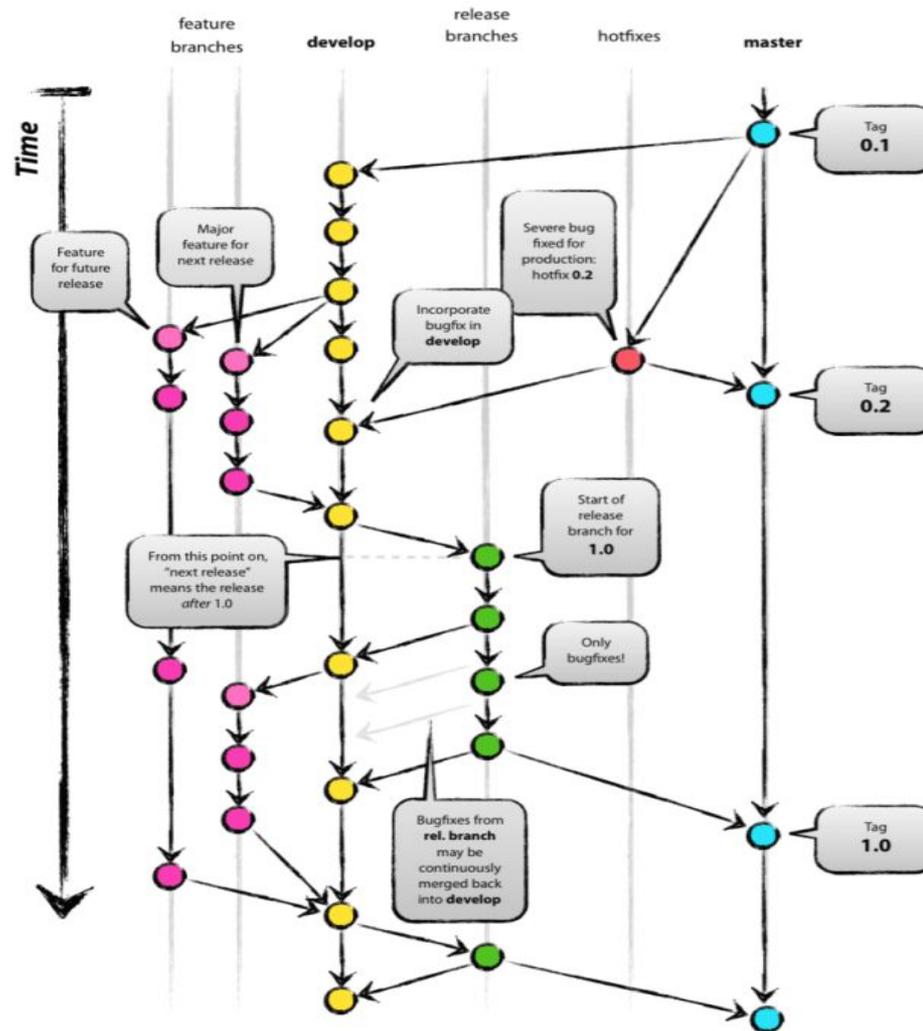
4. Pro Git

- Switch to *master* branch and update local repository

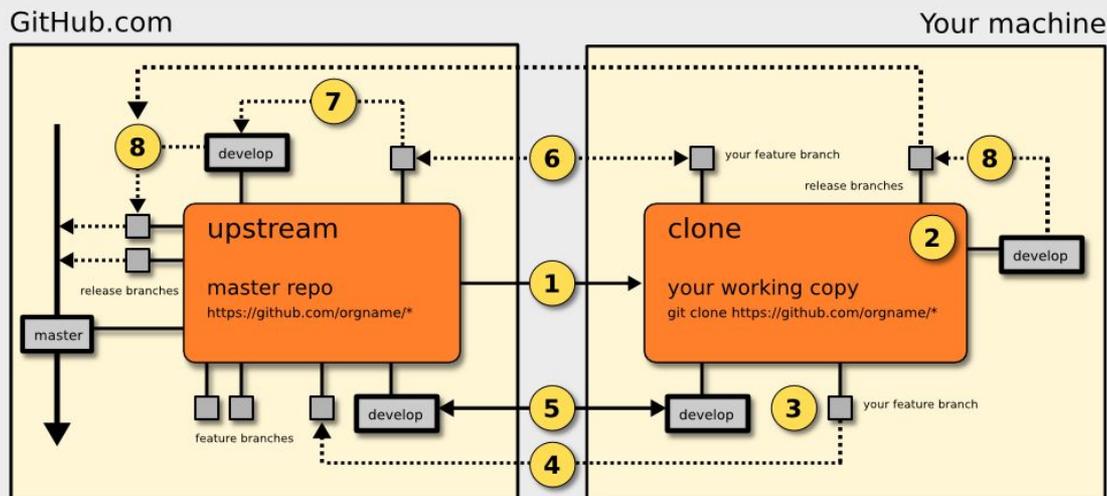


5. GitFlow

- <http://nvie.com/posts/a-successful-git-branching-model/>



6. HubFlow: GitFlow For GitHub



1 Clone the repo
`$ git clone git@github.com:orgname/##repo##`

2 Initialise the HubFlow tools
`$ git hf init`

3 Create a feature branch
`$ git hf feature start ##feature-name##`
- or checkout an existing feature branch
`$ git hf feature checkout ##feature-name##`

4 Push your feature branch back to GitHub as you make progress on your changes
`$ git hf push`

5 Keep up to date with completed features on GitHub on a regular basis
`$ git hf update`

6 Push your changes back to GitHub when you are ready to share them with collaborators.
`$ git hf push`
If you are collaborating with others, remember to fetch their changes back too!
`$ git hf pull`

7 When your feature is ready for review, it's time to merge your changes into origin/develop. First, push your feature:
`$ git hf push`
Use the GitHub website to issue a pull request to origin/develop from origin/feature/##feature-name##.
Ask a colleague to review your pull-request; don't accept it yourself unless you have to. Once accepted, close your feature:
`$ git hf feature finish (##feature-name##)`

8 When you have enough completed features, create a release branch, test and fix it, and then merge it into origin/master
`$ git hf release start ##release-version##`
<test + fix cycle>
`$ git hf release finish ##release-version##`

7. Useful Links

- <https://tortoisegit.org/>
- <https://datasift.github.io/gitflow/index.html>
- <https://datasift.github.io/gitflow/IntroducingGitFlow.html>
- <https://datasift.github.io/gitflow/GitFlowForGitHub.html>