

## Система управления версиями Git. Gitlab, Github, Bitbucket

Git - это распределённая система управления версиями. Это стабильная и хорошо документированная система. Исчерпывающая документация доступна online:

- <https://git-scm.com/docs> - описание всех команд с примерами.
- <https://git-scm.com/book/en/v2> - книга о Git.
- <https://git-scm.com/book/ru/v2> - книга о Git на русском языке.
- <https://git-scm.com/videos> - видеоролики.
- <https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html> - список наиболее употребительных команд.
- <http://rogerdudler.github.io/git-guide/index.ru.html> - простое руководство по работе с git .

Кроме того, документация входит в состав git и доступна offline с помощью команды `git help`, например, `git help clone`.

### Важные моменты:

- все команды выполняются из командной строки (Git Bash) (зам. по желанию, можно использовать графические клиенты для работы с git);
- будьте внимательны при выполнении команд - не допускайте опечаток;
- на время всего курса у вас должен быть ОДИН репозиторий;
- поддерживайте свой репозиторий в «чистоте и порядке» - группируйте логически связанные файлы в папки, но не забывайте, что структура репозитория должна быть понятна не только вам. Если вы не уверены как правильно разместить папки и файлы, посоветуйтесь с преподавателем.
- файлы, которые вам больше не нужны, следует помещать в отдельные папки. Делать это следует всегда ОТДЕЛЬНОЙ командой `commit`;
- репозиторий всегда будет представлять собой последнюю версию вашего проекта. При этом вся история ВСЕГДА будет доступна для просмотра, а потому не создавайте лишних копий файлов, храните лишь рабочее состояние;
- не рекомендуется делать «большие» `commit`'ы, т.е. включать большое число файлов в один `commit`. Лучше создавать несколько `commit`'ов и добавлять осмысленные комментарии;
- используйте английский алфавит для название файлов и папок;
- не используйте транслит;
- никогда не делайте `commit` для:
  - файлов, которые вы не создавали (метаданные ОС, настройки проекта IDEA, временные файлы и т.п. в репозитории не нужны);
  - папки с бинарными данными, представляющими результаты сборки, т.к. они занимают много места и никому не нужны;
  - если вы видите такие файлы в `git status`, добавьте их в `.gitignore` (см. далее);

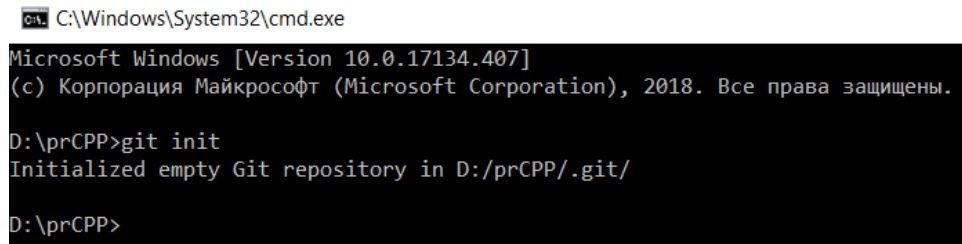
### Установка и настройка:

1. Зайти на сайт <http://git-scm.com> и следовать инструкциям по установке. Ниже описаны основные этапы.
2. Нажать “Downloads For Windows” (для ОС Windows) или скачать другую версию Git для вашей ОС.
3. Должна начаться загрузка установочного файла Git. Если загрузка не начинается, то нажать на ссылку для скачивания.
4. Запустить установочный файл. *Во время установки рекомендуется использовать все значения по умолчанию и убедиться, что выставлена опция “Use Git from the Windows Command Prompt” для Windows.*
5. Создать отдельную папку для проектов курса.

6. Перейти в режим работы с командной строкой, зайти в папку для проектов курса.

#### Создание нового локального репозитория и основы работы с ним:

1. Открыть командную строку (терминал) в папке, где есть права на запись. Предпочтительно завести отдельную папку в файловой системе для учебных проектов.
2. Ввести в командной строке: `git init` для создания нового локального репозитория.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.407]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

D:\prCPP>git init
Initialized empty Git repository in D:/prCPP/.git/

D:\prCPP>
```

3. Задать следующие данные о себе в командной строке:
  - a. `git config --global user.email "Ваш e-mail"`
  - b. `git config --global user.name "Ваше имя"`
4. Создать новый solution, используя Visual Studio в этой папке.  
Ваша директория с проектом должна выглядеть примерно так (включая скрытые файлы):  
<MyRepository>

```
.git ←this folder is usually hidden
├── lab01
│   ├── .vs
│   └── lab01.sln
└── ...
```
5. Проверить статус вашего репозитория с помощью команды `git status`.
6. Создать файл `.gitignore` в корне репозитория. Обратите внимание, что имя файла начинается с «.».
7. С помощью текстового редактора добавить туда файлы, которые Git должен игнорировать.

```
-----
# содержимое файла .gitignore для VS
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.
##
## Get latest from
https://github.com/github/gitignore/blob/master/VisualStudio.gitignore

# User-specific files
*.rsuser
*.suo
*.user
*.userosscache
*.sln.docstates

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
[Oo]bj/
[Ll]og/
```

```
# Visual Studio 2015/2017 cache/options directory
.vs/
# Uncomment if you have tasks that create the project's static files in
wwwroot
#wwwroot/

# Visual Studio 2017 auto generated files
Generated\ Files/

# MSTest test Results
[Tt]est[Rr]esult*/
[Bb]uild[Ll]og.*

# NUNIT
*.VisualState.xml
TestResult.xml

# Build Results of an ATL Project
[Dd]ebugPS/
[Rr]eleasePS/
dlldata.c

# Benchmark Results
BenchmarkDotNet.Artifacts/

# .NET Core
project.lock.json
project.fragment.lock.json
artifacts/

# StyleCop
StyleCopReport.xml

# Files built by Visual Studio
*_i.c
*_p.c
*_h.h
*.ilk
*.meta
*.obj
*.iobj
*.pch
*.pdb
*.ipdb
*.pgc
*.pgd
*.rsp
*.sbr
*.tlb
*.tli
*.tlh
*.tmp
*.tmp_proj
*_wpftmp.csproj
*.log
*.vspcc
*.vsscc
.builds
*.pidb
*.svclog
*.scc

# Chutzpah Test files
_Chutzpah*
```

```
# Visual C++ cache files
ipch/
*.aps
*.ncb
*.opendb
*.opensdf
*.sdf
*.cachefile
*.VC.db
*.VC.VC.opendb

# Visual Studio profiler
*.psess
*.vsp
*.vspx
*.sap

# Visual Studio Trace Files
*.e2e

# TFS 2012 Local Workspace
$tf/

# Guidance Automation Toolkit
*.gpState

# ReSharper is a .NET coding add-in
_ReSharper*/
*.[Rr]e[Ss]harper
*.DotSettings.user

# JustCode is a .NET coding add-in
.JustCode

# TeamCity is a build add-in
_TeamCity*

# DotCover is a Code Coverage Tool
*.dotCover

# AxoCover is a Code Coverage Tool
.axoCover/*
!.axoCover/settings.json

# Visual Studio code coverage results
*.coverage
*.coveragexml

# NCrunch
_NCrunch_*
.*crunch*.local.xml
nCrunchTemp_*

# MightyMoose
*.mm.*
AutoTest.Net/

# Web workbench (sass)
.sass-cache/

# Installshield output folder
[Ee]xpress/
```

```

# DocProject is a documentation generator add-in
DocProject/buildhelp/
DocProject/Help/*.HxT
DocProject/Help/*.HxC
DocProject/Help/*.hhc
DocProject/Help/*.hhk
DocProject/Help/*.hhp
DocProject/Help/Html2
DocProject/Help/html

# Click-Once directory
publish/

# Publish Web Output
*.[Pp]ublish.xml
*.azurePubxml
# Note: Comment the next line if you want to checkin your web deploy
settings,
# but database connection strings (with potential passwords) will be
unencrypted
*.pubxml
*.publishproj

# Microsoft Azure Web App publish settings. Comment the next line if you want
to
# checkin your Azure Web App publish settings, but sensitive information
contained
# in these scripts will be unencrypted
PublishScripts/

# NuGet Packages
*.nupkg
# The packages folder can be ignored because of Package Restore
**/[Pp]ackages/*
# except build/, which is used as an MSBuild target.
!*/[Pp]ackages/build/
# Uncomment if necessary however generally it will be regenerated when needed
#!*/[Pp]ackages/repositories.config
# NuGet v3's project.json files produces more ignorable files
*.nuget.props
*.nuget.targets

# Microsoft Azure Build Output
csx/
*.build.csdef

# Microsoft Azure Emulator
ecf/
rcf/

# Windows Store app package directories and files
AppPackages/
BundleArtifacts/
Package.StoreAssociation.xml
_pkginfo.txt
*.appx

# Visual Studio cache files
# files ending in .cache can be ignored
*.[Cc]ache
# but keep track of directories ending in .cache
!*. [Cc]ache/

# Others

```

```

ClientBin/
~$*
*~
*.dbmdl
*.dbproj.schemaview
*.jfm
*.pfx
*.publishsettings
orleans.codegen.cs

# Including strong name files can present a security risk
# (https://github.com/github/gitignore/pull/2483#issue-259490424)
#*.snk

# Since there are multiple workflows, uncomment next line to ignore
bower_components
# (https://github.com/github/gitignore/pull/1529#issuecomment-104372622)
#bower_components/

# RIA/Silverlight projects
Generated_Code/

# Backup & report files from converting an old project file
# to a newer Visual Studio version. Backup files are not needed,
# because we have git ;-)
_UpgradeReport_Files/
Backup*/
UpgradeLog*.XML
UpgradeLog*.htm
ServiceFabricBackup/
*.rptproj.bak

# SQL Server files
*.mdf
*.ldf
*.ndf

# Business Intelligence projects
*.rdl.data
*.bim.layout
*.bim_*.settings
*.rptproj.rsuser

# Microsoft Fakes
FakesAssemblies/

# GhostDoc plugin setting file
*.GhostDoc.xml

# Node.js Tools for Visual Studio
.ntvs_analysis.dat
node_modules/

# Visual Studio 6 build log
*.plg

# Visual Studio 6 workspace options file
*.opt

# Visual Studio 6 auto-generated workspace file (contains which files were
open etc.)
*.vbw

# Visual Studio LightSwitch build output

```

```

**/*.HTMLClient/GeneratedArtifacts
**/*.DesktopClient/GeneratedArtifacts
**/*.DesktopClient/ModelManifest.xml
**/*.Server/GeneratedArtifacts
**/*.Server/ModelManifest.xml
_Pvt_Extensions

# Paket dependency manager
.paket/paket.exe
paket-files/

# FAKE - F# Make
.fake/

# JetBrains Rider
.idea/
*.sln.iml

# CodeRush personal settings
.cr/personal

# Python Tools for Visual Studio (PTVS)
__pycache__/*
*.pyc

# Cake - Uncomment if you are using it
# tools/**
# !tools/packages.config

# Tabs Studio
*.tss

# Telerik's JustMock configuration file
*.jmconfig

# BizTalk build output
*.btp.cs
*.btm.cs
*.odx.cs
*.xsd.cs

# OpenCover UI analysis results
OpenCover/

# Azure Stream Analytics local run output
ASALocalRun/

# MSBuild Binary and Structured Log
*.binlog

# NVidia Nsight GPU debugger configuration file
*.nvuser

# MFractors (Xamarin productivity tool) working folder
.mfractor/

# Local History for Visual Studio
.localhistory/
-----

```

9. В файле не должно быть лишних символов (включая пробелы).
10. С помощью команды `git add .gitignore` подготовьте файл к коммиту.
11. С помощью команды `git commit -m "add .gitignore"` зафиксируйте изменения в локальном репозитории.

12. Если вы хотите изменить что-то в зафиксированных файлах (код вдруг перестал работать, хотя раньше работал или просто хотите вернуть то, что было ранее), используйте следующие методы исправления истории:

- `git commit --amend "комментарий"` перезапись последнего коммита;
  - `git revert <commit> "комментарий"` добавления коммита, который обращает `<commit>`, где `<commit>` - это идентификатор коммита, например, `c9b93f7`. Данная команда отменяет изменения, сделанные данным коммитом, не обязательно предшествующим. Важно отметить, что:
    - старые изменения все равно будут видны в истории и при необходимости их можно снова обратить, как отрицание отрицания;
    - если “отменяемый” коммит был сделан давно (и накопилось много изменений), то возможно потребуется разрешить исправить конфликты изменений.
  - `git reset --soft <commit> "комментарий"` удалить из истории все, что после `<commit>`, но сохранить состояние рабочей директории.
4. Проверить историю локального репозитория с помощью команды `git log`.

Создание профиля на GitLab, GitHub или Bitbucket и удаленного репозитория:

1. Зайти на сайт <https://gitlab.com> (или <https://github.com>, или <https://bitbucket.org/>).
2. Зарегистрироваться, войти на сайт.
3. Создать репозиторий/проект.

Project name

java0

Project URL

Project slug

Project slug

java0

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

java for beginners

Visibility Level

☒ Private  
Project access must be granted explicitly to each user.

☐ Internal  
The project can be accessed by any logged in user.

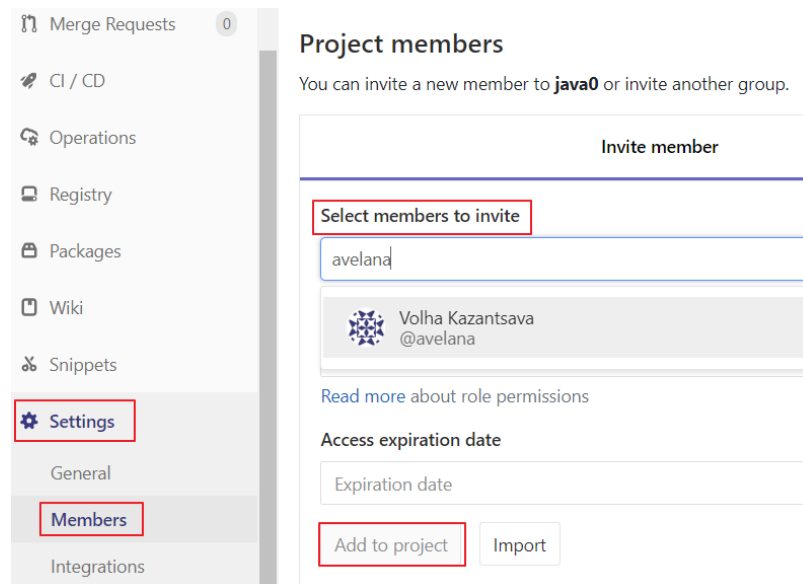
☐ Public  
The project can be accessed without any authentication.

☐ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project

4. В настройках репозитория/проекта добавить пользователей, имеющих доступ к вашему репозиторию.





5. Сохранить ссылку на ваш репозиторий и передать ссылку преподавателю.

#### Синхронизация локального и удаленного репозитория:

1. Вернуться в консоль Git.
2. Привязать удаленный репозиторий к локальному при помощи команды `git remote add origin <url>` где `<url>` это ссылка на ваш репозиторий.
3. Отправляем изменения, зафиксированные в локальном репозитории (ветка `master`), на удаленный сервер  
`git push origin master`.
4. Если вы хотите упростить себе работу, сделайте `origin master` параметрами по умолчанию. Для этого вместо `git push origin master` при первом запуске команды введите `git push -u origin master`. После этого вы можете пользоваться командой `git push` без параметров для отправки истории коммитов на удаленный сервер.
5. (!) Git спросит у вас логин и пароль от вашей учетной записи на GitLab/GitHub/Bitbucket.

6. Обновите в браузере страницу <https://gitlab.com> (или <https://github.com>, или <https://bitbucket.org/>) и убедитесь, что папка вашего проекта, а также файл `.gitignore` появился в каталоге вашего репозитория.

#### Копирование существующего удаленного репозитория и работа с ним:

1. Клонировать наш репозиторий. Для этого перейдем в любую другую папку (после выполнения задания ее можно будет удалить).
2. Затем вводим команду `git clone <url>`, где `<url>` - ссылка на наш репозиторий.
3. В файловом менеджере вы увидите, что появилась папка с названием вашего репозитория. Вы можете сравнить состояние ваших репозитория с помощью команды `git log` - оба репозитория должны показывать одинаковую историю коммитов.
4. **ВАЖНО:** в клонированном репозитории не будет файлов `*.iml`, папки `.idea` и всего того, что вы добавили в `.gitignore`.
5. Добавьте файл `README.txt` с описанием вашего репозитория в корневую папку репозитория. Зафиксируйте изменения и отправьте их на удаленный сервер при помощи последовательности команд `git add`, `git commit` и `git push`.

#### Проверка результатов работы с удаленным репозиторием:

1. Теперь вернемся в наш исходный репозиторий (где все начиналось).

2. Заберем изменения с удаленного репозитория с помощью команды `git pull` (или `git pull origin master`, если вы не использовали команду `git push -u origin master`).
3. Проверим наличие файла `README.txt` и его содержимое, чтобы убедиться что изменения были сделаны.