

2.3	Using the git Command	43
2.3.1	Setting the Name and Email Address (git config)	44
2.3.2	Downloading a Repository (git clone)	44
2.3.3	Adding Files (git add)	46
2.3.4	Saving an Intermediate State (git commit)	46
2.3.5	Adding and Changing Files, More Commits	47
2.3.6	Status (git status)	47
2.3.7	Excluding Files from Git Management (.gitignore file)	48
2.3.8	Transferring the Repository to a Remote Server (git push)	49
2.3.9	Updating the Local Repository (git pull)	50
2.3.10	Uploading a Local Repository to GitHub/GitLab	51
2.3.11	Branches (git checkout and git merge)	52
2.3.12	Logging (git log)	55
2.3.13	More Git Commands, Options, Special Cases, and Basics	56
2.4	Authentication	56
2.4.1	Windows Credential Manager	57
2.4.2	macOS Keychain	59
2.4.3	libsecret (Linux)	60
2.4.4	SSH instead of HTTPS	61
2.4.5	Different SSH Keys for Multiple GitHub/GitLab Accounts	64
2.4.6	If It Doesn't Work	65
2.5	Learning Git in a Playful Way (Githug)	66
2.5.1	Requirements	66
2.5.2	Game Structure	67
2.6	IDEs and Editors	68
2.6.1	Git GUI	68
2.6.2	GitHub Desktop	69
2.6.3	IntelliJ IDEA	71
2.6.4	TortoiseGit	73
2.6.5	Visual Studio Code	73
2.6.6	Xcode	74
2.7	Contributing to a Third-Party GitHub Project	76
2.7.1	Forks	76
2.7.2	Pull Requests	76
2.8	Synchronization and Backups	78
2.8.1	Git Issues	78
2.8.2	Conclusion	78

3 Basic Principles of Git 81

3.1 Terminology	81
3.1.1 About Commits	82
3.1.2 Log and Logging	84
3.1.3 Local and Remote Repositories	84
3.1.4 Hooks, Submodules, and Subtrees	85
3.2 The Git Database	85
3.2.1 The .git Directory	85
3.2.2 Git Object Types: Commits, BLOBs, Trees, and Tags	87
3.2.3 References	88
3.3 Commits	89
3.3.1 The Staging Area	89
3.3.2 The Commit	91
3.3.3 More Details	92
3.3.4 Renaming, Moving, or Deleting Files from the Repository	95
3.4 Commit-Undo	96
3.4.1 Not Saving a Change Permanently after All (git reset)	96
3.4.2 Restoring Changes Made since the Last Commit (git restore)	96
3.4.3 Viewing a File in an Old Version (git show)	97
3.4.4 Viewing Changes Compared to an Old Version (git diff)	98
3.4.5 Restoring a File to an Old Version (git restore)	98
3.4.6 Reverting the Last Commits (git revert)	99
3.4.7 Reverting the Last Commits (git reset)	100
3.4.8 Switching Temporarily to an Older Commit (git checkout)	101
3.4.9 Changing the Commit Message	103
3.5 Branches	104
3.5.1 Using Branches	105
3.5.2 Problems Switching between Branches (git checkout)	107
3.5.3 Determining “main” as the Default Name for New Repositories	108
3.5.4 Renaming “master” to “main”	108
3.5.5 Internal Details	109
3.6 Merging	110
3.6.1 Merging Branches (git merge)	110
3.6.2 Main Merge or Feature Merge?	113
3.6.3 Fast-Forward Merges	114
3.6.4 Octopus Merges	115
3.6.5 Merge Process	115
3.6.6 Cherry-Picking	115

3.7	Stashing	117
3.7.1	Caching and Restoring Changes	117
3.7.2	Stashing in Practice	118
3.7.3	Managing Multiple Changes	118
3.8	Remote Repositories	118
3.8.1	Initialization Work	119
3.8.2	Push and Pull	120
3.8.3	Remote Branches	122
3.8.4	Internal Details	125
3.8.5	Multiple Remote Repositories	126
3.8.6	Workflows	128
3.8.7	Configuring Your Own Git Server	128
3.9	Resolving Merge Conflicts	129
3.9.1	Collisions in the Code	129
3.9.2	Merge Tools	131
3.9.3	Binary File Conflicts	132
3.9.4	Merge Abort and Undo	133
3.9.5	Content-Related Merge Conflicts	134
3.9.6	MERGE Files	135
3.10	Rebasing	135
3.10.1	Example	136
3.10.2	Concept	137
3.10.3	Merge Conflicts during Rebasing	137
3.10.4	Side Effects	138
3.10.5	Pull with Rebasing	138
3.10.6	Special Rebasing Cases and Undo	139
3.10.7	Squashing	140
3.11	Tags	141
3.11.1	Listing Tags	141
3.11.2	Simple Tags versus Annotated Tags	142
3.11.3	Synchronizing Tags	143
3.11.4	Setting Tags Subsequently	143
3.11.5	Deleting Tags	144
3.11.6	Modifying or Correcting Tags (Retagging)	144
3.11.7	Signed Tags	145
3.12	References to Commits	145
3.12.1	Reference Names	146
3.12.2	refname@{date} and refname@{n}	147
3.12.3	Accessing Previous Versions	148
3.12.4	Examples	149
3.12.5	References to Files	150

3.13	Internal Details of Git	150
3.13.1	Object Packages	150
3.13.2	SHA-1 Hash Codes	151
3.13.3	The .git/index File	151
3.13.4	Commands for Managing the Git Database	152

4 Data Analysis in the Git Repository 153

4.1	Searching Commits (git log)	153
4.1.1	Clear Logging	155
4.1.2	Custom Formatting (Pretty Syntax)	156
4.1.3	Searching Commit Messages	157
4.1.4	Searching Commits That Modify Specific Files	158
4.1.5	Searching Commits of a Specific Developer	158
4.1.6	Restricting the Commit Range (Range Syntax)	159
4.1.7	Limiting Commits in Time	160
4.1.8	Sorting Commits	160
4.1.9	Tagged Commits (git tag)	162
4.1.10	Reference Log (git reflog)	162
4.2	Searching Files	163
4.2.1	Viewing Old Versions of a File (git show)	163
4.2.2	Viewing Differences between Files (git diff)	164
4.2.3	Viewing Differences between Commits	165
4.2.4	Searching Files (git grep)	166
4.2.5	Determining the Authorship of Code (git blame)	168
4.3	Searching for Errors (git bisect)	169
4.4	Statistics and Visualization	170
4.4.1	Simple Number Games (git shortlog)	170
4.4.2	Statistical Tools and Scripts	171
4.4.3	Visualizing Branches	172
4.4.4	GitGraph.js	173

5 GitHub 175

5.1	Pull Requests	176
5.1.1	Pull Requests on a Team	177
5.1.2	Pull Requests in Public Projects	179

5.2	Actions	180
5.2.1	YAML Syntax	181
5.2.2	Notification to Slack	182
5.2.3	The Continuous Integration Pipeline	184
5.3	Package Manager (GitHub Packages)	188
5.3.1	Example	189
5.4	Automatic Security Scans	193
5.4.1	Node.js Security	193
5.5	Other GitHub Features	197
5.5.1	Collaboration	197
5.5.2	Issues	197
5.5.3	Discussions and Teams	198
5.5.4	Wiki	199
5.5.5	Gists	200
5.5.6	GitHub Pages	201
5.6	GitHub Command-Line Interface	202
5.6.1	Installation	203
5.6.2	Examples of Use	204
5.7	Codespaces	206

6 GitLab 209

6.1	On-Premise versus Cloud	210
6.2	Installation	211
6.2.1	Installing GitLab Runner	214
6.2.2	Backup	216
6.3	The First Project	218
6.4	Pipelines	220
6.4.1	Auto DevOps	220
6.4.2	Manual Pipelines	223
6.4.3	Test Stage in the Manual Pipeline	225
6.4.4	Release Stage in the Manual Pipeline	226
6.4.5	Debugging Pipelines	227
6.5	Merge Requests	229
6.6	Web IDE	232
6.7	Gitpod	233

7 Azure DevOps, Bitbucket, Gitea, and Gitolite 237

7.1 Azure DevOps	237
7.1.1 Trying Out Azure DevOps	238
7.1.2 Test Plans	241
7.1.3 Conclusion	242
7.2 Bitbucket	242
7.3 Gitea	244
7.3.1 Trying Out Gitea	244
7.3.2 Server Installation with Docker	245
7.3.3 Server Installation on Ubuntu 20.04	247
7.3.4 A First Example with Gitea	250
7.4 Gitolite	255
7.4.1 Installation	255
7.4.2 Application	256

8 Workflows 259

8.1 Instructions for the Team	259
8.2 Solo Development	260
8.2.1 Conclusion	261
8.3 Feature Branches for Teams	262
8.3.1 New Function, New Branch	262
8.3.2 Example	262
8.3.3 Code Review	266
8.3.4 Merge	266
8.3.5 Rebasing	267
8.3.6 Conclusion	268
8.4 Merge/Pull Requests	269
8.4.1 Forks	271
8.4.2 Conclusion	272
8.5 Long-Running Branches: Gitflow	272
8.5.1 Main, Develop, Feature	273
8.5.2 Hot Bugfixes	275
8.5.3 Bugfixes in the develop Branch	276
8.5.4 Another New Function	276
8.5.5 Conclusion	277

11.1.8	Your Local Changes Would Be Overwritten (git checkout, git switch)	345
11.1.9	Your Branch Is Ahead of a Remote/Branch by n Commits (git pull, git status)	346
11.1.10	You're in a Detached HEAD State (git checkout)	346
11.1.11	Pathspec Did Not Match Any Files Known to Git (git checkout)	346
11.1.12	Please Enter a Commit Message to Explain Why This Merge Is Necessary (git pull)	347
11.1.13	Pulling without Specifying How to Reconcile Divergent Branches Is Discouraged (git pull)	347
11.1.14	Cannot Pull with Rebase: You Have Unstaged/Uncommitted Changes (git pull)	347
11.1.15	There Is No Tracking Information for the Current Branch (git pull)	348
11.1.16	Your Local Changes Would Be Overwritten (git merge, git pull)	348
11.1.17	Failed to Push Some Refs to <somerepo.git> (git push)	348
11.1.18	The Current Branch <name> Has No Upstream Branch (git push)	349
11.1.19	Merge Failed, Merge Conflict in <file> (git merge, etc.)	349
11.2	Saving Empty Directories	350
11.3	Merge for a Single File	350
11.3.1	git merge-file	350
11.3.2	git checkout	351
11.4	Deleting Files Permanently from Git	351
11.4.1	Local Changes Only, without Push (git rm)	352
11.4.2	Previously Uploaded Changes, with Push (git filter-branch)	353
11.4.3	Previously Uploaded Changes, after Push (git filter-repo)	356
11.4.4	Previously Uploaded Changes, after Push (BFG Repo Cleaner)	357
11.5	Splitting a Project	359
11.6	Moving Commits to a Different Branch	359
11.6.1	git reset	360
11.6.2	git cherry-pick	361

12 Command Reference 363

12.1	The git Command	363
12.1.1	Porcelain versus Plumbing	365
12.1.2	General Options	366
12.1.3	git add	366
12.1.4	git bisect	367

12.1.5	git blame	367
12.1.6	git branch	368
12.1.7	git checkout	369
12.1.8	git cherry-pick	370
12.1.9	git clean	371
12.1.10	git clone	371
12.1.11	git commit	372
12.1.12	git config	373
12.1.13	git diff	374
12.1.14	git fetch	375
12.1.15	git gc	375
12.1.16	git gui	376
12.1.17	git grep	376
12.1.18	git init	377
12.1.19	git log	377
12.1.20	git ls-files	379
12.1.21	git merge	380
12.1.22	git merge-base	381
12.1.23	git merge-file	381
12.1.24	git mergetool	381
12.1.25	git mv	381
12.1.26	git pull	382
12.1.27	git push	383
12.1.28	git rebase	384
12.1.29	git reflog	384
12.1.30	git remote	384
12.1.31	git reset	385
12.1.32	git restore	386
12.1.33	git rev-list	387
12.1.34	git revert	387
12.1.35	git rm	388
12.1.36	git shortlog	388
12.1.37	git show	388
12.1.38	git stage	389
12.1.39	git status	389
12.1.40	git submodule	389
12.1.41	git subtree	390
12.1.42	git switch	390
12.1.43	git tag	391
12.2	Revision Syntax	392
12.2.1	Commit Ranges (rev1..rev2 versus rev1...rev2)	392

12.3	git Configuration	393
12.3.1	Configuration File .git/config	393
12.3.2	Basic Settings	394
12.3.3	Configuration File .gitignore	396
12.3.4	Configuration File .gitmodules	396
12.3.5	Configuration File .gitattributes	397
	The Authors	399
	Index	401