GIT

اصول اولیه

در حین نوشتن کد (یا هر متن دیگری) ممکن است بخواهیم تغییرات را به چند دلیل پیگیری کنیم:

- نگاهی به تغییر کد هایی که در گذشته توسط شما یا برنامه نویسان دیگر نوشته شده است
- پیدا کردن خطاهای احتمالی و اینکه چه زمانی و توسط چه شخصی ایجاد شده است
- لغو تغییراتی که در گذشته ایجاد شده است
- نمایش میزان کارکرد هر برنامه نویس

Install git

برای نصب git وارد سایت زیر شوید و آنرا روی سیستم خود نصب کنید.

https://git-scm.com/downloads/

پس از دانلود و نصب برای نمایش نسخه نصب شده روی سیستم عامل خودتان، می توانید دستور زیر را در ترمینال وارد کنید

```
git --version
```

خروجی به شکل زیر خواهد بود

```
git version xx.xx.xx.windows.1
```

نمایش تنظیمات

```
git config –list

diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
```

لیست تنظیمات

ادامه نمایش تنظیمات --س

```
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.fsmonitor=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=
user.name=
color.ui=auto
```

لیست تنظیمات

تنظیمات diff

diff.astextplain.textconv=astextplain

این تنظیم مشخص می‌کند که Git برای تفاوت‌ها (diff) از مبدل astextplain استفاده کند. این مبدل معمولاً برای تبدیل فایل‌های باینری به فرمت متنی استفاده می‌شود تا بتوان تفاوت‌ها را به شکل متنی نمایش داد.

تنظیمات filter

filter.lfs.clean=git-lfs clean -- %f

این تنظیم برای استفاده از Git LFS (Large File Storage) است. دستور clean برای Git LFS برای آماده‌سازی فایل‌ها قبل از اضافه شدن به ریپازیتوری است. این دستور فایل‌ها را به فرمت مناسب برای ذخیره در LFS تبدیل می‌کند.

filter.lfs.smudge=git-lfs smudge -- %f

این تنظیم نیز برای Git LFS است. دستور smudge برای Git LFS برای بازگردانی فایل‌ها به حالت اصلی‌شان پس از خارج کردن از LFS است.

filter.lfs.process=git-lfs filter-process

این تنظیم از فرآیند فیلتر Git LFS برای مدیریت فایل‌های بزرگ استفاده می‌کند. این فرآیند به طور خودکار فایل‌ها را هنگام افزودن یا گرفتن از ریپازیتوری مدیریت می‌کند.

filter.lfs.required=true

این تنظیم مشخص می‌کند که استفاده از Git LFS برای این ریپازیتوری ضروری است و بدون آن عملیات انجام نمی‌شود.

لیست تنظیمات

تنظیمات http

http.sslbackend=openssl

این تنظیم مشخص می‌کند که Gitاز opensslبه عنوان بک‌اند SSLبرای اتصالات HTTP/HTTPSاستفاده کند. opensslیک کتابخانه محبوب برای مدیریت اتصالات امن است.

http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt

این تنظیم مسیر فایل گواهینامه‌های SSL CA (Certificate Authority)را مشخص می‌کند که برای اعتبارسنجی اتصالات HTTPSاستفاده می‌شود.

تنظیمات core

core.autocrlf=true

این تنظیم برای مدیریت انتهای خطوط فایل‌ها است. اگر روی ویندوز هستید، تنظیم trueباعث می‌شود که Gitخطوط پایان یونیکس (\n) را به خطوط پایان ویندوز (\r\n) تبدیل کند و برعکس.

core.fscache=true

این تنظیم فعال‌سازی کش سیستم فایل برای افزایش سرعت عملیات Gitدر سیستم فایل را مشخص می‌کند.

core.symlinks=false

این تنظیم مشخص می‌کند که Gitنباید لینک‌های نمادین (symlinks) را به عنوان فایل‌های خاص مدیریت کند. این تنظیم معمولاً روی سیستم‌های ویندوز استفاده می‌شود که پشتیبانی ضعیفی از symlinksدارند.

core.fsmonitor=true

این تنظیم فعال‌سازی مانیتور فایل سیستم برای بهبود عملکرد عملیات Gitرا مشخص می‌کند. این کار باعث می‌شود که Gitتغییرات در فایل‌ها را سریع‌تر تشخیص دهد.

تنظیمات pull

pull.rebase=false

این تنظیم مشخص می‌کند که به صورت پیش‌فرض از mergeبه جای rebaseبرای عملیات git pullاستفاده شود. یعنی تغییرات را به صورت یکپارچه ادغام می‌کند نه اینکه تاریخچه را تغییر دهد.

لیست تنظیمات

تنظیمات credential

credential.helper=manager

این تنظیم مشخص می‌کند که از Git Credential Managerبرای مدیریت اطلاعات احراز هویت استفاده شود. این ابزار اطلاعات ورود به سیستم را ذخیره و مدیریت می‌کند.

credential.https://dev.azure.com.usehttppath=true

این تنظیم مشخص می‌کند که Git Credential Managerاز مسیرهای کامل URLبرای ذخیره و مدیریت اعتبارهای مربوط به https://dev.azure.com استفاده کند.

تنظیمات init

init.defaultbranch=master

این تنظیم شاخه پیش‌فرض برای ریپازیتوری‌های جدید را مشخص می‌کند. در اینجا، شاخه پیش‌فرض masterاست.

تنظیمات user

user.email=xxxxx

این تنظیم ایمیل کاربر را برای کامیت‌ها مشخص می‌کند.

user.name=xxxxxx

این تنظیم نام کاربر را برای کامیت‌ها مشخص می‌کند.

تنظیمات color

color.ui=auto

این تنظیم مشخص می‌کند که Gitبه صورت خودکار از رنگ‌بندی برای خروجی استفاده کند تا خوانایی بهتری داشته باشد.

Git commands

دستورات پیکر بندی

git config --global user.name "your name"

git config --global user.email "your-email@example.com"

git config --list

New origin – Remote Control Github

برای کار با Remote Control باید پکیج CLI رو از Github نصب کنید.

در ابتدا از لینک زیر CLI را نصب کنید (برای ویندوز)

https://cli.github.com/

برای مک

brew install gh

برای لینوکس

sudo apt install gh

Git commands

ایجاد و مدیریت ریپازیتوری لوکال

کپی و حذف یک ریپازیتوری لوکال

git init

git clone <repository_url>

rm –rf [clone folder oder git folder]

آماده سازی ریپازیتوری برای ارسال به یک مخزن داخلی و خارجی

git remote add origin <repository_url>

git remote -v

git remote remove origin

Git commands

افزودن و حذف فایل‌ها

```
git add <file>
git add .
git rm <file>
git rm --cached <file>
git mv <old_filename> <new_filename>
```

کامیتها

```
git commit -m "commit message"
git commit --amend -m "new commit message"
```

Git status

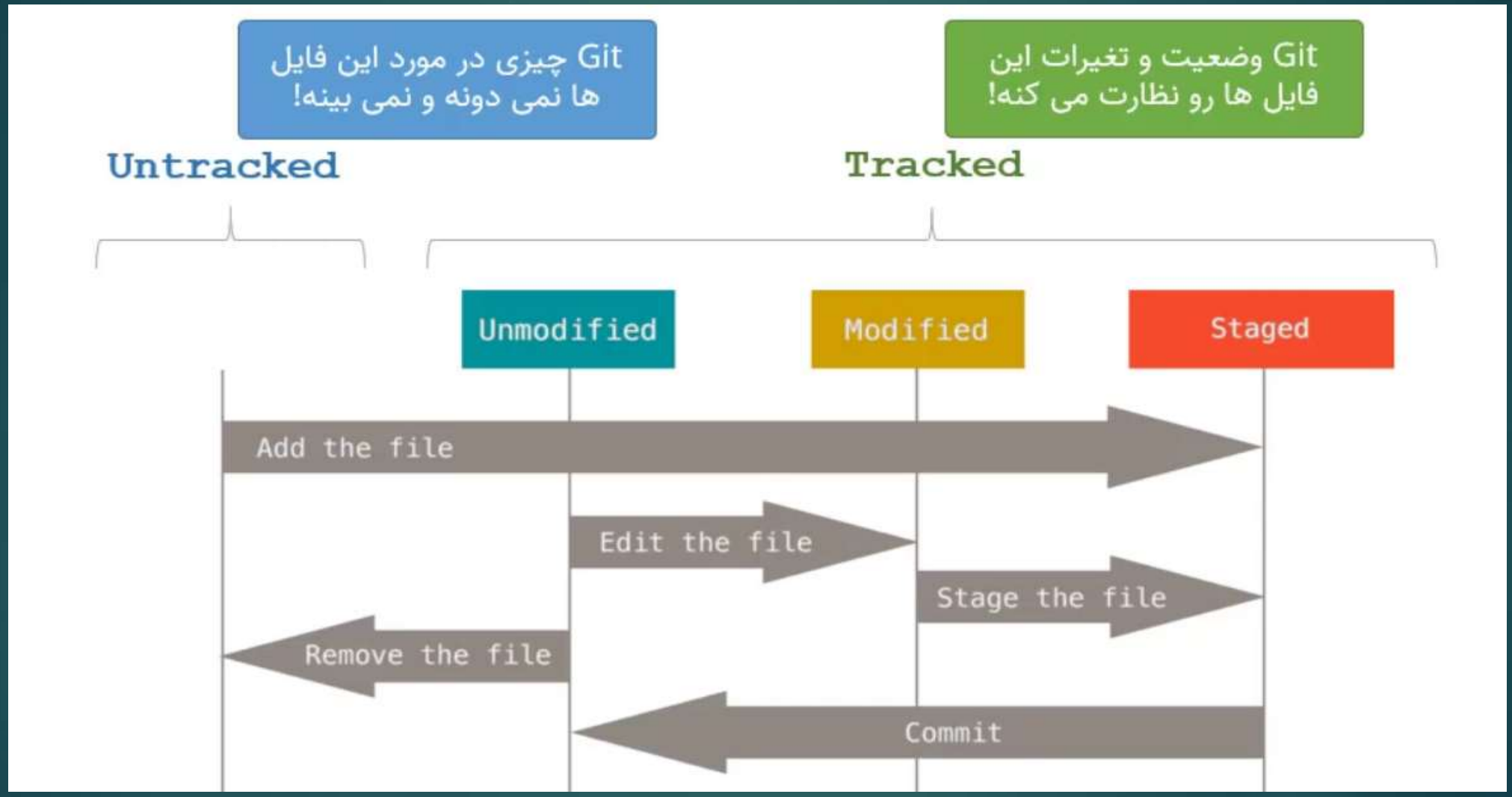
UNTRACKED

توسط GIT هندل نمی شود

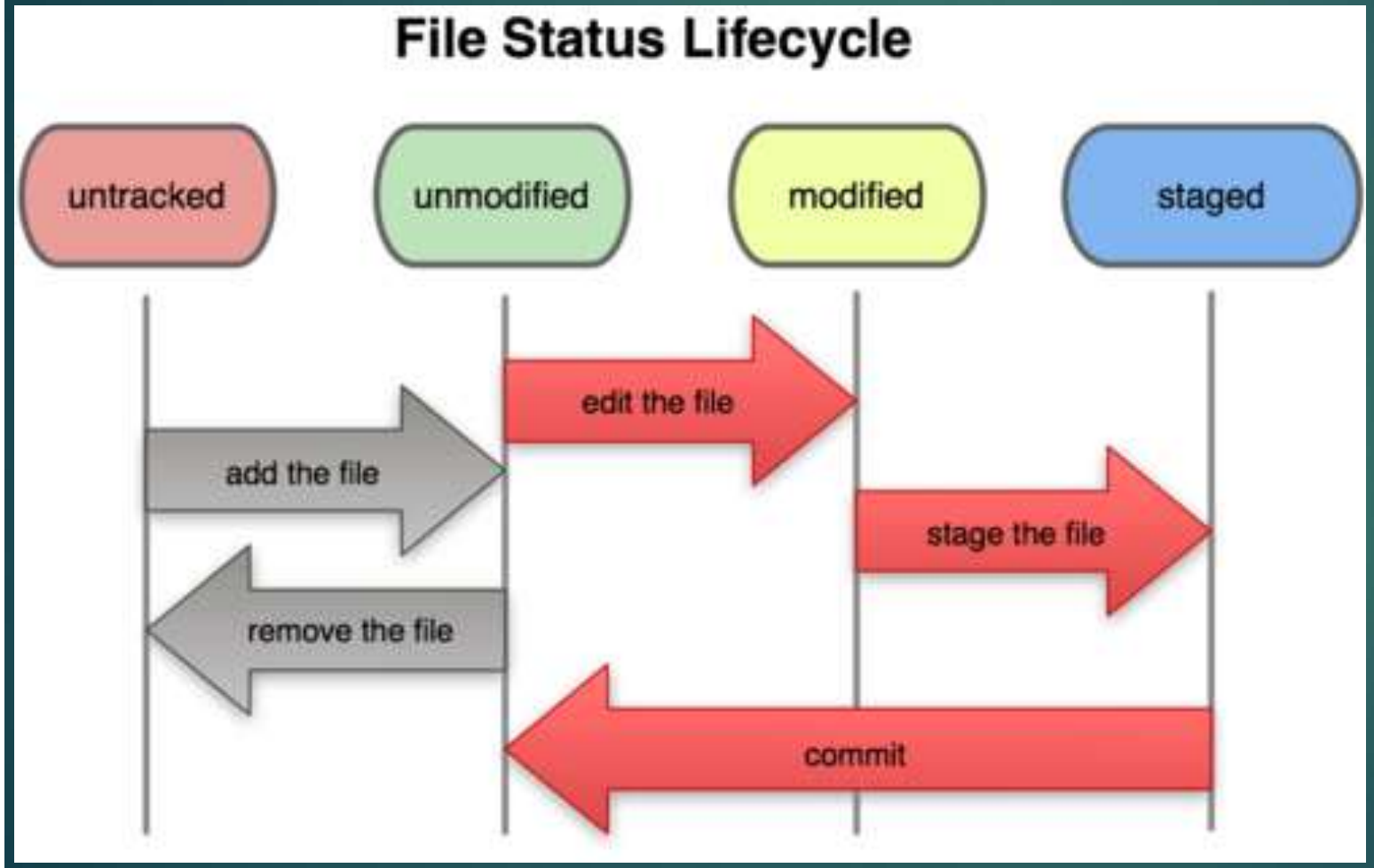
TRACKED

توسط GIT هندل می شود

Git status


Git چیزی در مورد این فایل ها نمی دونه و نمی بینه!
Git وضعیت و تغیرات این فایل ها رو نظارت می کنه!
Untracked
Tracked
Unmodified
Modified
Staged
Add the file
Edit the file
Stage the file
Remove the file
Commit

Git file lifecycle



GIT STATUS: نمایش وضعیت پروژه

UNTRACKED: فایل به GIT اضافه نشده است که با دستور GIT ADD اضافه می شود.

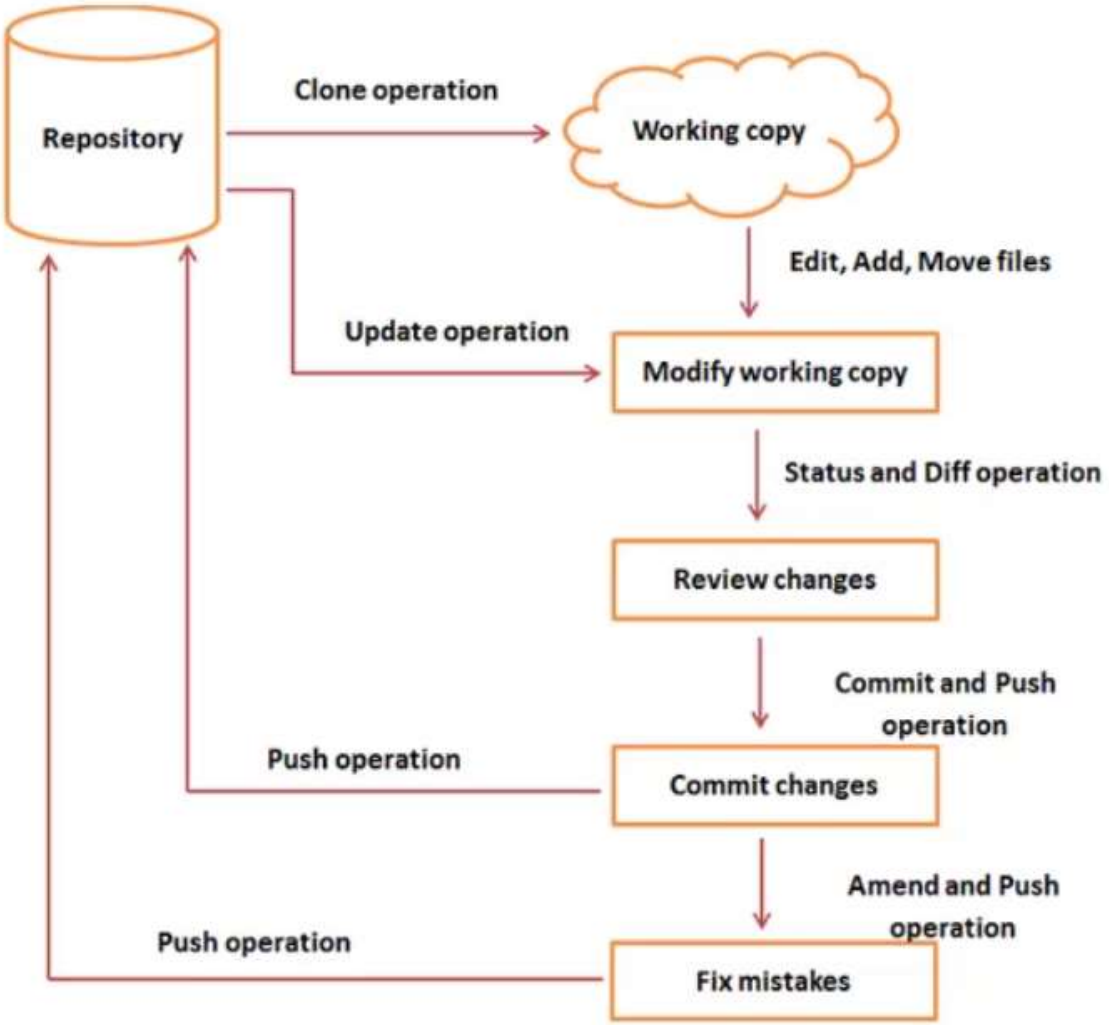
UNMODIFIED: تغییر در فایل ایجاد شده است اما در GIT ذخیره نشده است.

MODIFIED: تغییرات اعمال شده

GIT LIFECYCLE

General workflow is as follows

- You clone the Git repository as a working copy.
- You modify the working copy by adding/editing files.
- If necessary, you also update the working copy by taking other developer's changes.
- You review the changes before commit.
- You commit changes. If everything is fine, then you push the changes to the repository.
- After committing, if you realize something is wrong, then you correct the last commit, fix conflicts and push the changes to the repository.



Git commands

وضعیت و تاریخچه

git status

git log

git log --oneline

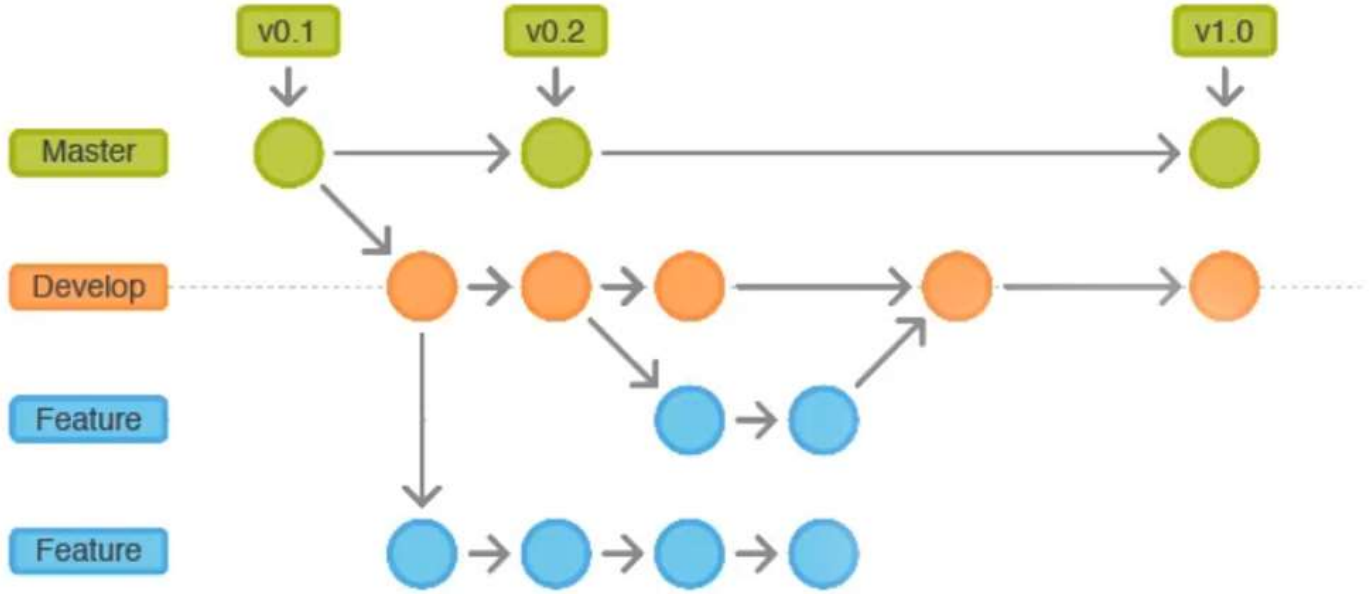
git log --graph

git show <commit_hash>

git diff

git diff <commit_hash>

Git branch



Git commands

branches شاخه‌ها

git branch نمایش شاخه ها

git branch <branch_name> ایجاد شاخه

git branch -d <branch_name> حذف شاخه

Git branch –D [نام شاخه] برای حذف شاخه های ادغام نشده و در صورتی که فایل هایی در این شاخه وجود داشته باشد که در شاخه مستر وجود نداشته باشد

git checkout /switch <branch_name> سوئیچ به شاخه

git checkout -b <branch_name> ایجاد شاخه و سوئیچ به آن

git merge <branch_name> ادغام شاخه جاری با شاخه دیگر به همراه کامیت و نگهداری تاریخچه هر دو شاخه

بازیابی شاخه

git reflog

git checkout –b [branch name] [commit sha-1]

unmerge شاخه

git reset [--hard –soft --mixed]

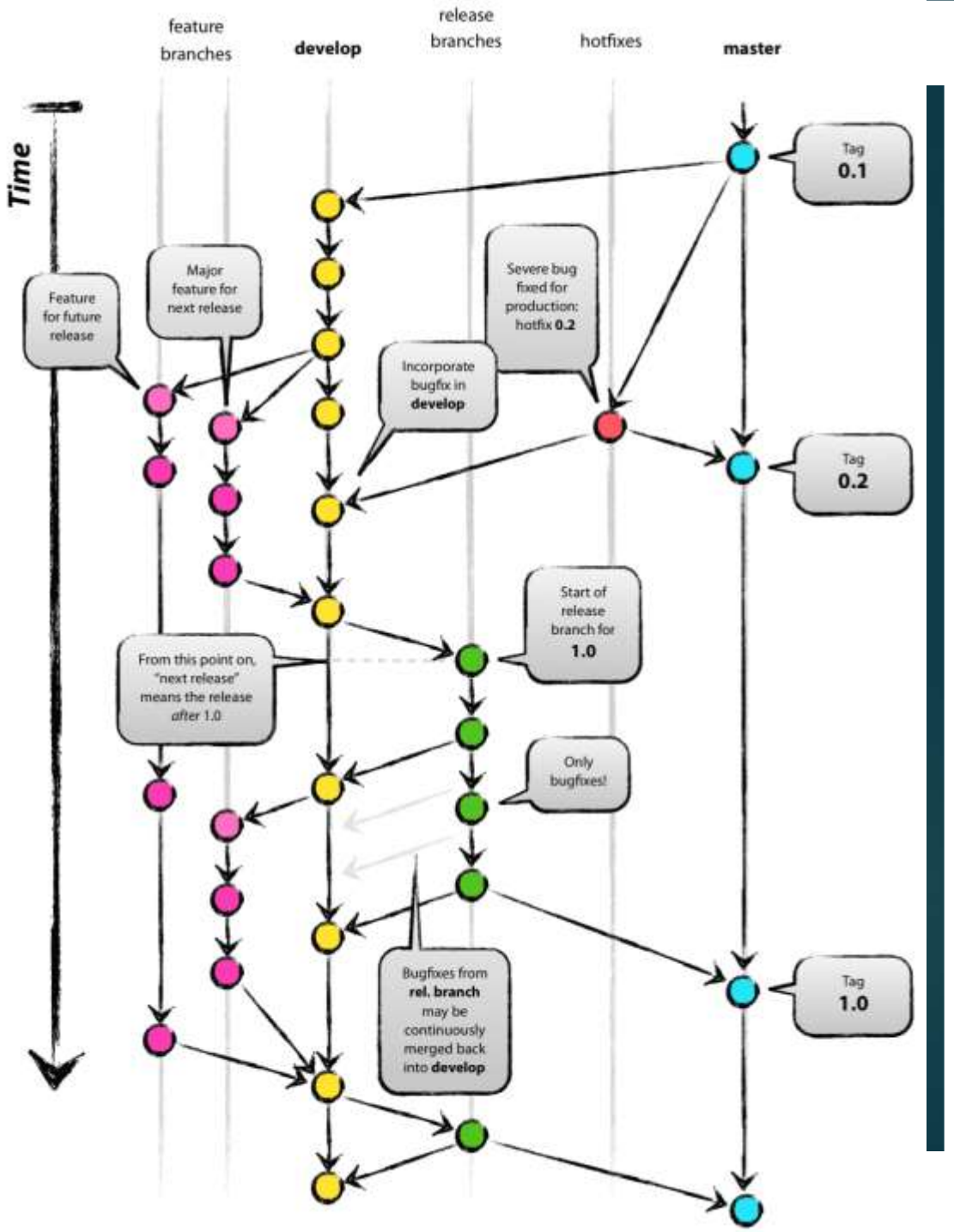
Git commands

--hard بازنشانی شاخص و پوشه کاری به حالت commit مشخص‌شده، حذف تمام تغییرات غیرمتعهد.

-- soft فقط بازنشانی HEAD به commit مشخص‌شده، نگه‌داشتن تغییرات در شاخص و پوشه کاری.

--mixed پیش‌فرض بازنشانی شاخص به commit مشخص‌شده، نگه‌داشتن تغییرات در پوشه کاری.

Git branch


feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.1
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

CLI-Git Commands

فعال سازی محل ذخیره سازی توکن

```
git config --global credential.helper store
```

ابتدا یک توکن از گیت هاب بسازید و کنترل های مد نظر را فعال کنید.

برای ورود به نام کاربری از دستور

```
gh auth login (github.com –> https –> y –> code auth gen)
```

بررسی لاگین بودن:

```
git config --get-regexp user\.name
```

بررسی توکن فعال بودن:

```
git config --get-regexp token
```

بررسی تنظیمات گلوبال برای لاگین به گیت هاب

```
git login –global --list
```

CLI-Git Commands

دستورات مدیریت ریپازیتوری گیت هاب با cli:

gh repo list [<owner>] [flags]

gh repo create [<name>] [--public --private]

gh repo delete [repo-name]

gh push –u [origin name] [branch name]

Git command

تگ‌ها ((tags در اصل همان ورژن ها هستند

git tag نمایش تگ ها

git tag –a v1.0.0 –m “version 1.0.0” ایجاد یک تگ

git tag -d <tag_name> حذف تگ محلی

git push origin <tag_name> ارسال تگ موجود به گیت هاب

git push origin --delete <tag_name> حذف تگ موجود از گیت هاب

استش ((stash (در نسخه جدید گیت نیازی به استفاده نیست و صرفا به دلیل اطلاع از عملکرد این دستور در این بخش آورده شده است)

git stash

git stash save "stash message"

git stash list

git stash apply

git stash drop

git stash pop

git stash clear

Git command

دریافت پروژه از ریپازیتوری گیت هاب

موارد مهم قابت توجه:

- پوشه مورد نظر همان نام در ریپازیتوری باشد
- دستور git init را برای فعال شدن گیت اجرا کنید
- سپس دستور زیر را اجرا کنید

- git pull [repo-url]

Git command

تحقیقات در مورد موارد زیر:

GIT FETCH

GIT REMOTE UPDATE

GIT CHERRY-PICK

GIT REVERT

GIT ARCHIVE