git-flow next CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows

INITIALIZATION

Initialize git-flow-next in a repository

```
$ git flow init
```

Use the GitHub Flow preset

```
$ git flow init --preset=github
```

Use the GitLab Flow preset

```
$ git flow init --preset=gitlab
```

CUSTOMIZATION

Launch interactive custom configuration

```
$ git flow init --custom
```

Specify the "main" branch name

```
$ git flow init --main <branch>
```

Specify the "develop" branch name

```
$ git flow init --develop <branch>
```

Specify the "feature" branch prefix

```
$ git flow init --feature <prefix>
```

Specify the "release" branch prefix

```
$ git flow init --release <prefix>
```

Specify the "hotfix" branch prefix

```
$ git flow init --hotfix <prefix>
```

CONFIGURATION

Manage git-flow configuration

```
$ git flow config
```

List the current configuration

```
$ git flow config list
```

Add a base branch

```
$ git flow config add base <name>
```

Add a topic branch type

```
$ git flow config add topic <name> <parent>
```

Edit a base branch configuration

```
$ git flow config edit base <name>
```

Rename a topic branch type

```
$ git flow config rename topic <old> <new>
```

Delete a topic branch type

```
$ git flow config delete topic <name>
```

TOPIC BRANCHES

Start a new topic branch

```
$ git flow <topic> start <name> [<base>]
```

List all topic branches of a certain type

```
$ git flow <topic> list
```

Update a topic branch with changes from its parent

```
$ git flow <topic> update [<name>]
```

Delete a topic branch

```
$ git flow <topic> delete <name>
```

Rename a topic branch

```
$ git flow <topic> rename <old-name> <new-name>
```

Switch to a topic branch

```
$ git flow <topic> checkout <name>
```

Finish a topic branch

```
$ git flow <topic> finish <name>
```

FINISHING BRANCHES - OPTIONS

Syntax example:

```
$ git flow feature finish login [OPTION]
```

Rebase the topic branch before merging

```
$ ... --rebase
```

Squash all commits into a single commit

```
$ ... --squash
```

Create a merge commit even for a fast-forward

```
$ ... --no-ff
```

Create a tag for the finished branch

```
$ ... --tag
```

Keep the branch after finishing

```
$ ... --keep
```

Force finish a non-standard branch

```
$ ... --force
```

Abort a finish operation

```
$ ... --abort
```

Continue a finish operation after resolving conflicts

```
$ ... --continue
```

GENERAL COMMANDS

Show current git-flow configuration

```
$ git flow overview
```

Show the version of git-flow-next

```
$ git flow version
```

TRADITIONAL GIT-FLOW

Base branches:
- **main**: Holds the production code.
- **develop**: Primary branch for all ongoing development.

Topic branches:
- **feature/***: Used for all new functionality.
- **bugfix/***: Reserved for bugfixes.
- **release/***: Used to prepare a new production release.
- **hotfix/***: Reserved for urgent, production-breaking fixes.
- **support/*** (optional): Used to manage and maintain older, currently supported versions of the software that are still in use.

GITHUB FLOW

Base branches:
- **main**: The single source of truth and the only base branch.

Topic branches:
- **feature/***: Covers all development work: new features, refactors, and bug fixes.

GITLAB FLOW

Base branches:
- **production**: The only branch that deploys to the production environment.
- **main**: The primary integration branch where all new features are initially merged.
- **staging**: The final gate. It is always updated from "main" before being promoted to "production".

Topic branches:
- **feature/***: Used for all new functionality.
- **hotfix/***: Reserved for urgent, production-breaking fixes.