Bitbucket Cheat Sheet

A quick reference guide to Bitbucket, covering essential Git commands and Bitbucket-specific features for effective version control and collaboration.

Basic Git Commands

Configuration

`git config --global user.name "Your Name"`	Sets the name you want attached to your commit transactions.
`git config --global user.email "your_email@example.com"`	Sets the email you want attached to your commit transactions.
`git config --list`	Lists all git configurations.

Starting a New Repository

`git init`	Initializes a new Git repository in the current directory.
`git clone <repository_url>`	Clones an existing repository from a URL (e.g., Bitbucket).

Basic Workflow

`git add <file>`	Adds a file to the staging area.
`git add .`	Adds all modified files to the staging area.
`git commit -m "Commit message"`	Commits staged changes with a descriptive message.
`git push origin <branch_name>`	Pushes local commits to a remote repository (e.g., Bitbucket).
`git pull origin <branch_name>`	Pulls changes from a remote repository to your local branch.
`git status`	Shows the status of the working directory and staging area.

Branching and Merging

Branch Management

`git branch`	Lists all local branches. The current branch is highlighted.
`git branch <new_branch_name>`	Creates a new branch.
`git checkout <branch_name>`	Switches to the specified branch.
`git checkout -b <new_branch_name>`	Creates a new branch and switches to it.
`git branch -d <branch_name>`	Deletes a branch (if it's already merged).
`git branch -D <branch_name>`	Force deletes a branch (even if it's not merged).

Merging Branches

`git merge <branch_name>`	Merges the specified branch into the current branch.
`git mergetool`	Launches a merge tool to resolve conflicts during a merge.
`git commit`	After resolving conflicts, commit the merged changes.

Remote Branches

`git push origin --delete <branch_name>`	Deletes a branch on the remote repository.
`git fetch`	Fetches the latest changes from the remote repository without merging.
`git remote -v`	Lists all remote connections.
`git branch -r`	Lists remote branches

Bitbucket Specific Features

Pull Requests

Pull requests are a core feature for code review and collaboration in Bitbucket.

1. Create a new branch for your changes.
2. Commit your changes to the branch.
3. Push the branch to Bitbucket.
4. Create a pull request from your branch to the target branch (e.g., `main`).
5. Assign reviewers to the pull request.
6. Address feedback and make further commits if necessary.
7. Once approved, merge the pull request.

Bitbucket Pipelines

Bitbucket Pipelines allows you to automate your build, test, and deployment workflows directly within Bitbucket.

- Configuration is done via a `bitbucket-pipelines.yml` file in the repository root.
- Define steps, services, and variables in the YAML file.
- Pipelines are triggered automatically on commits, pull requests, or manually.

Example `bitbucket-pipelines.yml`:

```yaml
pipelines:
  default:
    - step:
        script:
          - echo "Running tests..."
          - ./run_tests.sh
```

Bitbucket Cloud

Private Repositories	Bitbucket offers free private repositories for small teams.
Integrations	Integrates with Jira, Trello, and other Atlassian products.
REST API	Provides a REST API for automating tasks and integrating with other systems.

Advanced Git Techniques

Stashing

`git stash`	Temporarily saves changes that you don't want to commit immediately.
`git stash pop`	Applies the most recent stashed changes and removes it from the stash.
`git stash list`	Lists all stashed changes.
`git stash apply stash@{n}`	Applies a specific stashed change without removing it from the stash.
`git stash drop stash@{n}`	Removes a specific stashed change.

Rebasing

Rebasing is an alternative to merging that integrates changes from one branch into another by rewriting the commit history.

`git rebase <branch_name>` - Reapplies commits from the current branch onto the specified branch. Use with caution, especially on shared branches, as it modifies the commit history.

Ignoring Files

`.gitignore` file	Create a `.gitignore` file in the repository root to specify intentionally untracked files that Git should ignore.
	
	Example:
	
	`*.log`
	`/temp/`
	`config.ini`