Azure Repos Cheat Sheet

A quick reference guide to Azure Repos, covering essential commands and concepts for version control.

Getting Started with Azure Repos

Setting Up Your Environment

Creating a New Repository:

1. Navigate to your Azure DevOps project.
2. Select 'Repos' from the left-hand menu.
3. Click on the dropdown next to the current repository name and select 'New repository'.
4. Choose a name for your repository and click 'Create'.

Cloning an Existing Repository:

1. Navigate to the repository you want to clone in Azure Repos.
2. Click the 'Clone' button.
3. Copy the clone URL (HTTPS or SSH).
4. In your local terminal, use the command: `git clone <clone_url>`

Connecting with SSH:

1. Generate an SSH key pair if you don't already have one (`ssh-keygen -t rsa -b 4096`).
2. Add the public key to your Azure DevOps profile.
3. Use the SSH clone URL to clone and interact with the repository.

Basic Git Commands

`git init`	Initialize a new Git repository.
`git clone <url>`	Clone a repository from a remote URL.
`git add <file>`	Add a file to the staging area.
`git commit -m "<message>"`	Commit changes with a descriptive message.
`git push origin <branch>`	Push changes to a remote branch.
`git pull origin <branch>`	Pull changes from a remote branch.

Branching and Merging

Branch Management

`git branch <branch_name>`	Create a new branch.
`git checkout <branch_name>`	Switch to an existing branch.
`git branch -d <branch_name>`	Delete a branch locally (if merged).
`git push origin --delete <branch_name>`	Delete a branch remotely.
`git branch -a`	List all branches (local and remote).
`git checkout -b <new_branch> origin/<remote_branch>`	Create a new local branch and track a remote branch.

Merging Strategies

Basic Merge:

1. Checkout the target branch (e.g., `main`).
2. Run `git merge <feature_branch>` to merge the feature branch into the target branch.
3. Resolve any merge conflicts.
4. Commit the merge.

Merge with Pull Request (Recommended):

1. Create a pull request in Azure Repos from the feature branch to the target branch.
2. Review the changes and resolve any conflicts in the web interface.
3. Approve the pull request and complete the merge.

Resolving Merge Conflicts:

- Use `git status` to identify conflicting files.
- Open the conflicting files and manually resolve the conflicts, looking for `<<<<<<<`, `=======`, and `>>>>>>>` markers.
- After resolving, `git add` the files and `git commit` the changes.

Working with Remote Repositories

Managing Remotes

`git remote -v`	List configured remote connections.
`git remote add <name> <url>`	Add a new remote connection.
`git remote remove <name>`	Remove a remote connection.
`git remote rename <old_name> <new_name>`	Rename a remote connection.
`git fetch <remote>`	Fetch branches and/or tags (plus associated objects) from another repository.
`git remote update`	Fetch updates from all remotes.

Pull Requests in Azure Repos

Creating a Pull Request:

1. Push your branch to Azure Repos.
2. In Azure Repos, navigate to the 'Pull requests' section.
3. Click 'New pull request'.
4. Select the source branch and target branch.
5. Add a title and description for the pull request.
6. Assign reviewers and click 'Create'.

Reviewing a Pull Request:

1. Navigate to the 'Pull requests' section in Azure Repos.
2. Select the pull request you want to review.
3. Review the changes, add comments, and vote (Approve, Approve with suggestions, Wait, Reject).
4. Complete the pull request when all reviewers have approved the changes.

Completing a Pull Request:

- Once the required reviewers have approved the pull request, you can complete it.
- Options for completing include merging, squashing, and deleting the source branch.
- Azure Repos provides options to automatically complete pull requests based on branch policies.

Advanced Features

Stashing Changes

`git stash`	Stash your uncommitted changes.
`git stash save "<message>"`	Stash changes with a message.
`git stash list`	List all stashed changes.
`git stash apply`	Apply the latest stashed changes.
`git stash apply stash@{<n>}`	Apply a specific stashed change (e.g., `stash@{0}`).
`git stash drop stash@{<n>}`	Delete a specific stashed change.

Ignoring Files

Use a `.gitignore` file to specify intentionally untracked files that Git should ignore.

Example `.gitignore`:

```
*.log
/temp/
build/
```

Common `.gitignore` Patterns:

- `*.log`: Ignore all files with the `.log` extension.
- `/temp/`: Ignore the `temp` directory at the root of the repository.
- `build/`: Ignore the `build` directory (recursively).
- `config.ini`: Ignore a specific file named `config.ini`.

To ignore a file that has already been committed, you must first remove it from the index:

```
git rm --cached <file>
git commit -m "Remove file from index"
```