[Git & Gitlab] (CheatSheet)

Git Commands

- **Initialize a Repository**: `git init`
- **Clone a Repository**: `git clone [URL]`
- **Add File(s) to Staging**: `git add [file]`
- **Commit Changes**: `git commit -m "[commit message]"`
- **Push Changes**: `git push [remote] [branch]`
- **Pull Changes**: `git pull [remote]`
- **Create a Branch**: `git branch [branch_name]`
- **Switch Branches**: `git checkout [branch_name]`
- **Merge Branch**: `git merge [branch_name]`
- **Fetch Remote Repository**: `git fetch [remote]`
- **View Status**: `git status`
- **View Commit History**: `git log`
- **Stash Changes**: `git stash`
- **Apply Stashed Changes**: `git stash apply`
- **Reset to Commit/HEAD**: `git reset --hard [commit]`
- **Rebase Branch**: `git rebase [branch_name]`
- **Delete Branch**: `git branch -d [branch_name]`
- **View Remote Repositories**: `git remote -v`
- **Tag a Commit**: `git tag [tag_name] [commit]`
- **Show Changes/Differences**: `git diff`

GitLab-Specific Commands

- **Create a New Project**: `gitlab project create [name]`
- **List Projects**: `gitlab project list`
- **Create a Merge Request**: `gitlab merge-request create`
- **List Merge Requests**: `gitlab merge-request list`
- **Create an Issue**: `gitlab issue create`
- **List Issues**: `gitlab issue list`
- **Upload a File to Repository**: `gitlab project-file upload`
- **Download a File from Repository**: `gitlab project-file download`

By: Waleed Mousa

GitLab CI/CD

`.gitlab-ci.yml` Configuration Basics

```
stages:
  - build
  - test
  - deploy
```

- **Define Stages:Define a Job:**

```
build_job:
  stage: build
  script:
    - echo "Building the project."
```

- **Setting up Test Job:**

```
test_job:
  stage: test
  script:
    - echo "Running tests."
```

- **Setting up Deploy Job:**

```
deploy_job:
  stage: deploy
  script:
    - echo "Deploying application."
```

- **Defining Artifacts:**

```
artifacts:
  paths:
    - build/
```

By: Waleed Mousa

- **Only/Except Policies**:

```
only:
  - master
```

- **Cache Dependencies**:

```
cache:
  paths:
    - .maven/
```

- **Using Docker Images**:

```
image: maven:latest
```

- **Defining Variables**:

```
variables:
  MAVEN_CLI_OPTS: "-s .m2/settings.xml"
```

- **Using Services (e.g., Databases)**:

```
services:
  - postgres:latest
```

- **Manual Job Trigger**:

```
deploy_prod:
  stage: deploy
  script:
    - echo "Deploying to production."
  when: manual
```

- **Scheduled Pipelines**:
    - Configured through the GitLab UI under CI/CD > Schedules.

GitLab Runner

- **Install GitLab Runner**: Follow official installation guide
- **Register a Runner**: `gitlab-runner register`
- **Start GitLab Runner**: `gitlab-runner start`
- **Stop GitLab Runner**: `gitlab-runner stop`
- **Unregister a Runner**: `gitlab-runner unregister --url [CI_SERVER_URL] --token [REGISTRATION_TOKEN]`

Advanced CI/CD

- **Multi-Project Pipelines**:
  - Use `trigger` in `.gitlab-ci.yml` to trigger pipelines in other projects.
- **Environment and Deployment**:

```
deploy_to_prod:
  stage: deploy
  environment:
    name: production
  script:
    - echo "Deploying to production."
```

Review Apps:

- Set up dynamic environments for merge requests.
- **Include External CI/CD Configuration**:

```
include:
  - project: 'other/project'
    ref: master
    file: '/path/to/template.yml'
```

- **Using Anchors for Template Reuse**:

```
.template: &template
  script:
    - echo "This is a script template."
```

By: Waleed Mousa

```
use_template:
  <<: *template
```

Advanced Git Operations

- **Cherry-pick a Commit**: git cherry-pick [commit_hash]
- **Revert a Commit**: git revert [commit_hash]
- **Interactive Rebase**: git rebase -i [base_commit]
- **Squash Commits**: git rebase -i [base_commit] *(then use 'squash')*
- **Amend a Commit**: git commit --amend
- **Stash Specific Files**: git stash push [file_name]
- **Clean Untracked Files**: git clean -f
- **Reflog to Recover Lost Commits**: git reflog
- **Check Out a Remote Branch**: git checkout -b [branch_name] [remote_name]/[branch_name]
- **List Tags**: git tag

Advanced GitLab Features

- **Set Up Protected Branches**: GitLab UI > Repository > Protected Branches
- **Configure Project Access**: GitLab UI > Project > Settings > General
- **Add Group and Project Members**: GitLab UI > Project/Group > Members
- **Set Up Webhooks for Integration**: GitLab UI > Project > Settings > Webhooks
- **Set Up Issue Boards for Agile Management**: GitLab UI > Project > Issue Board
- **Create and Manage Labels**: GitLab UI > Project > Labels
- **Configure Repository Mirroring**: GitLab UI > Repository > Settings > Repository Mirroring
- **Use GitLab Snippets for Code Sharing**: GitLab UI > Project > Snippets
- **Configuring Project Wikis for Documentation**: GitLab UI > Project > Wiki
- **Using GitLab's Container Registry**: docker push registry.gitlab.com/[username]/[project]

By: Waleed Mousa

GitLab CI/CD Advanced Usage

- **Using Multi-Stage Pipelines**:
  - Define multiple stages in `.gitlab-ci.yml` and specify the stage for each job.
- **Deploy to Multiple Environments**:

```
deploy_staging:
  stage: deploy
  script: echo "Deploying to staging"
  environment: staging
```

- **Use of Artifacts and Dependencies**:

```
build:
  stage: build
  script: make build
  artifacts:
    paths:
      - build/

test:
  stage: test
  script: make test
  dependencies:
    - build
```

- **Docker-in-Docker for Building Containers**:

```
image: docker:19.03.12
services:
  - docker:19.03.12-dind
```

- **Multi-Project Pipeline Triggers**:

```
trigger:
  project: my/other-project
  branch: master
```

- **Using Variables and Secrets**:

```
variables:
  DATABASE_URL: "postgres://..."
```

- **Scheduled Pipeline Execution**:
- Configure in GitLab UI under CI/CD > Schedules.
- **Optimizing with Caching**:

```
cache:
  paths:
    - .maven/
```

- **Setting Up Review Apps**:
    - Use dynamic environments to preview changes.
- **Using Auto DevOps in GitLab**:
    - Enable Auto DevOps in project settings.

Advanced CI/CD Techniques

- **Parent-Child Pipelines for Modular CI/CD**:

```
build:
  stage: build
  trigger:
    include: 'path/to/child-pipeline.yml'
```

- **Using GitLab's CI Lint Tool**:
    - Validate `.gitlab-ci.yml` using the CI Lint tool in GitLab UI.
- **Coverage Reports and Code Quality**:
    - Integrate testing and code quality tools and report coverage in the pipeline.
- **Accessibility Testing within Pipelines**:
    - Integrate accessibility testing tools like `pa11y` or `axe-core`.

GitLab Runner Advanced Configurations

- **Register Multiple Runners**:
  - Register different runners for specific jobs or tags.
- **Configure Runner Executors**:
  - Choose between `shell`, `docker`, `virtualbox`, etc., when registering runners.
- **Using Tags to Control Job Execution**:

```
job1:
  tags:
    - linux
    - docker
```

- **Runner Caching to Speed Up Builds**:
  - Configure cache in runner's `config.toml`.
- **Configuring Parallel Job Execution in Runner**:
  - Set `concurrent` in the runner's `config.toml`.

Git Tips for Efficient Workflow

- **Alias Common Commands**: `git config --global alias.co checkout`
- **Interactive Staging**: `git add -i`
- **Patch Mode for Partial Commits**: `git add -p`
- **Git Bisect to Find Bugs**: `git bisect start; git bisect bad; git bisect good [good_commit]`
- **Search in Git Logs**: `git log --grep="pattern"`

GitLab Tips for Better Project Management

- **Quick Actions in Issues/MRs**: Use `/close`, `/label`, `/assign` in comments.
- **Preview Merge Conflict Resolution**:
  - Use the conflict resolution UI in GitLab merge requests.
- **Use GitLab's Built-in CI/CD Templates**:
  - Utilize existing `.gitlab-ci.yml` templates for common languages and frameworks.
- **Set Up GitLab Pages for Project Documentation**:

By: Waleed Mousa

    - Host static websites directly from a GitLab repository.
- **Monitor Pipeline Performance**:
    - Use the Pipelines page to monitor build times and optimize them.

GitLab CI/CD Security Practices

- **Scan for Vulnerabilities**: * Integrate SAST (Static Application Security Testing) and DAST (Dynamic Application Security Testing) in pipelines.
- **Dependency Scanning**: * Add dependency scanning jobs to the CI/CD pipeline.
- **Container Scanning for Docker Images**: * Scan Docker images for vulnerabilities as part of the CI/CD pipeline.