Lab 2.1: Install Git

Objective: Install the Git package

- Install the `git` package

Steps:

- Use the package manager to install the git package

Lab 2.2: Install Git Bash Integration

Objective: Install the Git Bash Integration

- Install the `bash-completion` package
- Modify your prompt to highlight the git state

Steps:

- Use the package manager to install the `bash-completion` package
- Fetch the `git-prompt.sh` script from https://raw.githubusercontent.com/git/git/master/contrib/completion/git-prompt.sh
- Customize your prompt
- Persist changes in your .bashrc file

Lab 3.1: Configure your username and email address

Objective: Configure your username and email address

- Configure your username and email address using Git CLI commands

Steps:

- Use `git config --global user.name "Your Name"`
- Use `git config --global user.email "name@domain.com"`
- Verify the changes with `git config --global --list`

Lab 4.1: Clone an existing Git repository

Objective: Clone an existing Git repository

- Clone an existing Git repository

Steps:

- Navigate to https://github.com/icinga/icinga2
- Copy the clone URL
- Navigate into your home directory
- Use `git clone` to clone the remote Git repository

Lab 4.2: Initialize a local Git repository

Objective: Initialize Git repository

- Initialize Git repository

Steps:

- Create a new directory called `training` in your home directory
- Change into it
- Run `git init`

Lab 4.3: Add a new README.md file

Objective: Add a new README.md file to the current change index

- Add a new README.md file

Steps:

- Change into `$HOME/training`
- Create README.md and add `# GitLab Training Notes` as first line
- Use `git add` to add the file to the current change index
- Verify the change with `git status`

Lab 4.4: Reset File from Staging Index

Objective: Reset File from Staging Index

- Reset file from staging index

Steps:

- Change into `$HOME/training`
- Remove the previously added `README.md` file from the staging index with `git reset README.md`
- Verify it with `git status` and explain what happened.
- Re-add the `README.md` and examine again with `git status`.

Lab 4.5: Examine current changes

Objective: Examine current changes

- Examine current changes

Steps:

- Change into `$HOME/training`
- Edit README.md
- Use `git status` to see unstaged changes
- Add the changed files to the staging area
- Use `git status` again

Lab 4.6: Use Git Diff

Objective: Play with Git Diff

- Use `git diff`

Steps:

- Change into `$HOME/training`
- Edit README.md
- Use `git diff` to compare unstaged changes
- Add the changed file to the staging area
- Use `git diff` again
- Explain what `git diff --staged` does

Lab 4.7: Add .gitignore file and exclude files/directories

Objective: Examine current changes

- Add .gitignore file and exclude files/directories

Steps:

- Change into `$HOME/training`
- Create a file `generated.tmp`
- Create a directory `debug` with the file `.timestamp`
- Examine the state with `git status`
- Exclude them in a .gitignore file
- Examine the state with `git status`

Lab 5.1: Commit Changes

Objective: Commit Changes

- Modify files and commit your changes

Steps:

- Change into `$HOME/training`
- Modify the `README.md` file and add more docs
- Add the change to the staging index
- Commit the change to your Git history with `git commit -v README.md`

Next steps:

- Use `git log` to verify the history

Lab 5.2: Examine the Commit History

Objective: Examine the Commit History

- Examine the commit history

Steps:

- Change into `$HOME/training`
- Add and commit remaining changes e.g. `.gitignore`
- Use `git log` to print the current history
- Use `git show` to show specific commits (defaults to the latest)
- Use `git diff` to compare changes between specific revisions
- Use `git blame .gitignore` to see the authors for the file

Lab 5.3: Learn more about tig

Objective: Install and use tig

- Learn more about tig

Steps:

- Install the tig package
- Run tig in `$HOME/training`
- Clone a different repository and run tig there e.g. `$HOME/icinga2`

Next Steps:

- Select a line and press `Enter`
- `q` quits the detail view and the application

Lab 5.4: Amend changes to commits

Objective: Use git amend

- Use git amend

Steps:

- Change into `$HOME/training`
- Modify `README.md` and add docs about amend
- Add `README.md` to the staging index and commit the change
- Edit `README.md` again and add it to staging
- Use `git commit --amend README.md` to add the change to the previous commit

Bonus:

- Adopt the commit message using `git commit --amend`

Lab 6.1: Show the current branch

Objective: Show the current branch

- Show the current branch

Steps:

- Change into `$HOME/training`
- Use `git branch` to highlight the current branch

Lab 6.2: Create and checkout a new branch

Objective: Create and checkout a new branch

- Create and checkout a new branch

Steps:

- Change into `$HOME/training`
- Create a new branch `feature/docs` based off `master` with `git branch feature/docs master`
- List the branches with `git branch`
- Checkout the new branch with `git checkout feature/docs`

Bonus:

- Verify how `git checkout -b feature/docs2` works
- Explain how it helps here

Lab 6.3: Delete the branch

Objective: Delete the branch

- Delete the previously created branch

Steps:

- Change into `$HOME/training`
- Switch to the master branch
- Use `git branch -d feature/docs2` to delete the selected branch

Bonus:

- Try to delete the branch you are currently on

Lab 6.4: Show the second commit

Objective: Show the second commit

- Show the second commit

Steps:

- Change into `$HOME/training`
- Use `HEAD` and only show the second latest commit.

Lab 6.5: Show history of different branch

Objective: Delete the branch

- Use `git log` from the master branch on another branch

Steps:

- Create a new branch aside from master, if not existing: `git checkout -b feature/docs`
- Switch to the master branch
- Use `git log feature/docs`

Bonus:

- Modify and commit changes
- Diff current HEAD against `feature/docs` branch

Lab 7.1: Create GitLab Project

Objective: Create a new GitLab app in NWS

- Create a new GitLab app in NWS

Steps:

- Navigate to https://nws.netways.de and register a trial account if not existing
- Choose Apps > GitLab CE > Basic
- Deploy the app
- Choose Access and Live View and set a secure password for the `root` user.
- Login

Lab 7.2: Create GitLab Project

Objective: Create a new GitLab project for the current user

- Create a new GitLab project for the current user

Steps:

- Click the `+` icon next to the search field
- Choose `New Project`
- Add the name `training`
- Leave it as `Private`
- Create the project

Note:

- Learn about the project view and the HTTPS clone URL

Lab 7.3: Add the repository as remote origin

Objective: Add the repository as remote origin

- Add the repository as remote origin

Steps:

- Open the project in GitLab and extract the `HTTPS` clone URL
- Navigate into your local repository
- Use `git remote add origin <remoteurl>`
- Push your local branch
- Use `--set-upstream` (short: `-u`) to enable the local branch tracking the remote repository

Bonus:

- Configure the default push method to `simple`
- Explain what `git push -u origin --all` does suggested by GitLab

Lab 7.4: Add a credential cache

Objective: Add the credentail cache to the configuration

- Add the credentail cache to the configuration

Steps:

- Go to your terminal
- Use `git config credential.helper 'cache --timeout=99999'`

Lab 7.5: Explore Project History

Objective: Learn more about the project history

- Learn more about GitLab and the project's history
- Compare the local history to the remote project's history

Steps:

- Click on `History` in the project view and examine the Git commits
- Run `git log` or `tig` on your shell and compare them to GitLab

Bonus:

- Use `Repository > Graph` in GitLab

Lab 8.1: Learn more about git push

Objective: Learn more about git push

- Learn more about git push

Steps:

- Change into `$HOME/training`
- Edit `README.md` and add a note on `git push`
- Add and commit the changes
- Push the changes

Bonus:

- List all remote branches with `git branch -r`

Lab 8.2: Learn more about git fetch and git pull

Objective: Learn more about git fetch and git pull

- Learn more about git fetch and git pull

Steps:

- Go to your project repository in GitLab
- Edit the `README.md` in your browser and commit the change to master
- Run `git fetch` and explain `git diff master origin/master`
- Run `git pull`
- Explain the difference

Bonus:

- Repeat push and pull multiple times

Lab 8.3: Add Git Tag

Objective: Add git tag

- Add git tag

Steps:

- Use `git tag` and add the `v0.1` tag
- Verify the added tag with `git tag -l`
- Push tags to remote origin with `git push --tags`
- Open GitLab and navigate into `Repository > Tags`

Bonus:

- Add a tag description with `git tag -m "Release v0.1" v0.1`

Lab 8.4: Learn more about git stash

Objective: Learn more about git stash

- Learn more about git stash

Steps:

- Change into `$HOME/training`
- Edit `README.md`
- Examine the status with git status
- Stash your current changes to the working directory
- Run git status again
- Examine the stash with `git stash list` and `git stash show -p`
- Fetch the previously stashed changes with `git stash pop`

Lab 8.5: Learn more about git cherry-pick

Objective: Learn more about git cherry-pick

- Learn more about git cherry-pick

Steps:

- Create and checkout the `feature/docs-hotfix` branch
- Edit `README.md` and commit the change
- Use `git log -1` to examine the Git commit
- Checkout the master branch
- Use `git cherry-pick -x <id>`
- Verify the commit with `git show`

Lab 9.1: Collaborate in a central repository

Objective: Create conflicting history tree

- Create conflicting history tree

Steps:

- Open the GitLab project `training`
- Edit `README.md`, add `This change is from my colleague.`.
- Stage & commit the change to master

Local CLI Steps:

- Change into `training` directory
- Edit `README.md`, add `This is my local change.`.
- Commit and try to push, explain the error message

Lab 9.2: Resolve conflicts in a central repository

Objective: Rebase your local history with the remote repository

- Rebase your local history with the remote repository

Steps:

- Fetch remote with `git fetch`
- Compare changes with `git diff origin/master`
- Rebase with `git rebase origin/master`
- Resolve possible merge conflicts, add them
- Continue with `git rebase --continue`, push rebased history

Lab 9.3: Use Feature Branches

Objective: Create a new feature branch

- Create a new feature branch

Steps:

- Change into `$HOME/training`
- Use `git checkout -b feature/docs-workflows` to create a new feature branch based on the master
- Add and commit changes
- Push the branch to your central repository

Lab 9.4: Merge Feature Branches

Objective: Merge Feature Branches

- Update master branch and merge feature branch

Steps:

- Change into `$HOME/training`
- Checkout the feature branch `feature/docs-workflows`
- Edit `README.md`, add and commit the changes
- Diff the feature branch to the current master with `git diff master`
- Checkout the `master` branch
- Merge the feature branch as non-fast-forward with `--no-ff`
- Show the history tree with `tig` or inside GitLab

Bonus:

- Explain why the forced merge commit with `--no-ff` is important

Lab 9.5: Create Milestone and First Issue

Objective: Create Milestone and First Issue

- Create Milestone and First Issue

Steps:

- Navigate into `Issues > Milestones`
- Select `New Milestone` and use `v0.1` as title
- Navigate to `Issues` and select `New issue`
- Use `Update documentation` as title, add a description
- Assign the `v0.1` milestone

Lab 9.6: Create Merge Request

Objective: Create Merge Request

- Create Merge Request

Steps:

- Create/checkout the branch `feature/docs-merge-request`
- Edit `README.md`, add, commit and push the changes
- Open the proposed GitLab URL in your browser
- Fill in the merge request and add `fixes #1` as description
- Merge the MR and tick `delect source branch`
- Analyse the history in GitLab/tig and open issue #1

Bonus:

- Run `git fetch --prune` and `git branch -d feature/docs-merge-request`

Lab 9.7: Rebase and squash commits

Objective: Rebase and squash commits

- Rebase and squash commits

Steps:

- Add 3 commits to the `master` branch and push them.
- Use `git rebase -i HEAD~3` to start the interactive mode. `HEAD~3` takes the last 3 commits compared to current HEAD.
- Use `pick` for the top commit`
- Replace `pick` with `squash` for the other commits
- Save and edit the final commit message
- Use `git log` to verify the history

Bonus:

- Push the changed commit history using `git push -f` and explain what happens

Lab 9.8: Force Push and Protected Branches

Objective: Force Push and Protected Branches

- Try to force push and learn about protected branches in GitLab

Steps:

- Run `git push -f` in the master branch
- Explain the error
- Navigate into `GitLab > Project > Settings > Repository`
- Temporarily unprotected the `master` branch
- Run `git push -f` again
- Protect the `master` branch again and discuss with the trainer

Lab 9.9: Delete remote branch

Objective: Delete remote branch

- Delete remote branch

Steps:

- Change into `$HOME/training`
- Create or identify a remote branch `feature/docs-wrong-name`
- Delete the remote branch

Lab 11.1: Inspect CI Runner settings

Get CI Runner Token

- Use the GitLab Admin UI to inspect CI runners

Steps:

- Navigate to Admin > Overview > Runners
- Inspect the token
- Check existing runners

Lab 11.2: Create .gitlab-ci.yml

Create CI configuration

- Create CI configuration file .gitlab-ci.yml

Steps:

- Create the `.gitlab-ci.yml` file in the `training` directory (vim, nano, etc.)
- Add `image: alpine:latest` to specify base image
- Add job `all_tests` with `script` as array element, which itself runs `exit 1`

Lab 11.3: Push to GitLab

Push CI config and trigger GitLab job

- Add .gitlab-ci.yml to Git and push to GitLab

Steps:

- Use `git add .gitlab-ci.yml` and commit the change
- Push the commit into the remote repository
- Navigate to the project into `CI / CD` and verify the running job

Bonus:

- Modify the exit code to `0`, add, commit, push and verify again

Lab 11.4: Practical Example for CI Runners: Preparations

Prepare container to convert Markdown to HTML

- Prepare container to convert Markdown to HTML

Steps:

- Modify `.gitlab-ci.yml` and add a `before_script` section after the `image` section
- Update `apk` package manager and install `python3` and `py-pip` packages
- Use `pip` to install the `markdown` and `Pygments` libraries
- Commit and push the changes

Example:

```
before_script:
  - apk update && apk add python3 py-pip
  - pip install markdown Pygments
```

Lab 11.5: Practical Example for CI Runners: Create Docs

Create HTML docs from Markdown

- Create HTML docs from Markdown

Steps:

- Add a new job `markdown` after the `all_tests` job
- Add `script` and convert `README.md` to `README.html` using Python
- Add `artifacts` with `paths` pointing to `README.html`. Expires in `1 week`.
- Commit and push the changes
- Download and view the `README.html` file in your browser

Lab 11.6: Practical Example for CI Runners: Update Docs

Update docs

- Add what you have learned so far into README.md and generate docs

Steps:

- Edit `README.md`
- Commit and push changes
- Download and view the `README.html` file in your browser

Lab 11.7: CI: Pipelines

Create HTML docs from Markdown

- Build a job pipeline with stages

Steps:

- Edit .gitlab-ci.yml and add stages
- Add jobs to stages
- Commit and push the changes
- Check the GitLab Job Pipelines

Lab 12.1: Use the Issue Board

Use the Issue Board

- Use the Issue Board

Steps:

- Navigate to `Issues` and create a new issue `Learn about the issue board`
- Navigate to `Issues > Board`
- Choose to create `To Do/Doing` labels
- Drag the issue from `Open` to `To Do` to `Doing` to `Closed`
- Explain the changed labels and workflow

Lab 12.2: Update README.md with the Web IDE

Update README.md with the Web IDE

- Use the Web IDE to write documentation and verify CI results

Steps:

- Navigate to `Repository > Files` and click `Web IDE`
- Select `README.md` from the tree
- Edit the file, use the live preview
- Commit the changes, select `Commit to master branch`
- Select the rocket icon on the right, check the pipeline status

Lab 13.1: Use Git Blame

Objective: Use Git Blame

- Use `git blame`

Steps:

- Pick a file from your local git repository
- Use `git blame filename`
- Explain the line prefix and its meaning

Lab 13.2: Add an alias for git diff

Objective: Add an alias for git diff

- Add an alias for the `git diff` command

Steps:

- Edit the `$HOME/.gitconfig` file
- Add a new `[alias]` section if not existing
- Add `d` as an alias for `diff`