LINC Git Repository Guidelines

This repository will provide basic guidelines on how to properly create and manage Software Repositories that will be hosted under LINC's Git Organization.

More specifically the topics covered in this guideline are listed below:

Naming Conventions

It is important to use proper naming when creating a repository which will be hosted under LINC. A repository name should be clear, easy to read and by reading it someone should get an idea of what the repository is about.

The following list can be used as a checklist in order to properly chose a name for your repository:

1. **Select a descriptive name for your repository.** When thinking about naming your repository try to avoid selecting names that are too generic. Be specific! The name of your repository should reflect to the capabilities of your project and inform everyone that is interested about your project what is it about.

   > Example: You are a Bsc student that is doing his Bsc Thesis in LINC. Your Thesis topic demands from you to develop an Eclipse CHE plugin that orchestrates cloud application deployments to Amazon Web Service

   Bad repository naming: Based on the example above names such:

   - ADE
   - Thesis
   - Diplomatiki
   - .....

   The names above are too generic and do not provide any useful information about the capabilities and nature of your work.

   Good repository naming: On the other hand as good repository names based on the example above can be considered:

   - eclipse-che-aws-orchestration-plugin
   - EclipseCheAWSOrchestrator
2. **Follow one of the naming conventions below**

   - **Use lower case letters and dashes** to seperate different words. This is your best option and the easiest one to read (e.g., eclipse-che-aws-orchestrator-plugin)

    - **Use the camel case** (e.g., EclipseCheAWSOrchestrator) Although this conventions is ok it is a little bit more difficult to read especially for longer repository names.
3. **Be consistent when naming your repositories**

Providing sufficient repository descriptions

Every repository that is hosted under LINC should have at least ***one*** `README.md` file inside the repository root folder(like the one you are reading now) that contains cobrehensive description of the content of the repository.

Inside the `README.md` one should include:

1. Short description of the problem that the software is trying to solve.
2. What was the approach used to solve the problem and the basic features of the software.
3. Complete and clear step-by-step description of the development and execution environments needed to run the software (e.g., how to resolve dependencies, versions of programming language etc.)
4. Complete and clear step-by-step description of how to use and run the software.
5. Explanation of the API/Classes/Functions (with examples if possible)
6. State what kind of licence is your software under.
7. List the contributors and contact information.

`README.md` is written in Markdown language. It is advised to use .md files within subfolders of your projects whenever you feel there is a need to explain something specific in detail.

Repository Good Practices

Adding .gitignore

A `.gitignore` file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected.Each line in a `.gitignore` file specifies a pattern.

Usually when performing commits to your github repository you would like to ignore IDE automatically generated files and folders which usually contain meta-data regarding the structure of your project and folders(e.g., the `.idea\` folder automatically created by InteliJ or `.project\` folder created by Eclipse).

- How to create .gitignore

Filetypes in a git repositories

You should keep in mind that github is a software repository and VCS and you have to treat it as such. For this reason you ***should not commit*** other type of files except of source codes.

Especially you should avoid uploading to your repository any kind of **executables** `.exe .bat .jar ...` etc. and other binary type of files.

Protecting sensitive information

In many occasions you will have to write software that it needs to communicate with other systems (such as Databases etc.) that may require authentication or other sensitive information or some local paths on the development machine. Usually this information is used within some kind of configuration file or a source code file itself.

When it is time to commit changes to the repo make sure that your files do not contain sensitive information such as **passwords, local paths, private keys etc.**

It is advised that whenever such occasion occurs add the original files in `.gitignore` and create exact copies of those files with the sensitive information excluded for the commit.

Visibility of repositories

In Git there are two levels of repository visibility:

1. **Public Repository**: Repositories set to Public are accessible to everyone and appear to search results
2. **Private Repository**: Repositories set to Private are only accessible by the contributors of the repository, the owners and members of the organization that this repository belongs to.

It is advised that when creating a repository to LINC's Git your repository should be set to **Public**.

Private Repositories at LINC can only be created upon prior communication with Prof. M. Dikaiakos and Prof. G. Pallis.

- How to set the visibility of a repository

Example LINC Repositories

Here are some good LINC repositories that follow tha majority of the guidelines provided in this document

- https://github.com/UCY-LINC-LAB/CloudServiceDeployer
- https://github.com/UCY-LINC-LAB/NBAStatsCollector
- https://github.com/UCY-LINC-LAB/MovieDB

Organization of Repositories

Organizations are great for creating distinct groups of users, such as divisions or groups working on similar projects. Public repositories that belong to an organization are accessible to users in other organizations, while private repositories are inaccessible to anyone but members of the organization.

Teams

LINC uses teams to create groups of members and control access to repositories. Team members can be granted read, write, or admin permissions to specific repositories.

An example of LINC Teams is the team ADE2017 in which the Bsc. Thesis of 2017 students are stored.

- Reporisory Permission Levels

Licensing

Public repositories on GitHub are often used to share open source software. For your repository to truly be open source, you'll need to license it so that others are free to use, change, and distribute the software.

You're under no obligation to choose a license. However, without a license, the default copyright laws apply, meaning that you retain all rights to your source code and no one may reproduce, distribute, or create derivative works from your work. If you're creating an open source project, we strongly encourage you to include an open source license.

Most people place their license text in a file named `LICENSE.txt` (or `LICENSE.md`) in the root of the repository

Some projects include information about their license in their README. For example, a project's README may include a note saying "This project is licensed under the terms of the MIT license."

As a best practice, we encourage you to include the license file with your project.

The choice of the type of license should be decided with Prof. M. Dikaiakos and Prof. G. Pallis

- How to add a license to a Github repository
- Which open-source license to pick

Helpful Material

- Github Help Page
- Pro Git book

**Updated on: 8/6/2017*