**General comments above the CLING documentation project:**

**THE CLING WEBPAGE:**
The first evident issue is that there is not one dedicated website/page, but rather a multitude of blogs/links/portal web/conference/YouTube videos of conferences, all providing precious information, but in an unorganized way, and written in different styles. As an example, the following links: https://root.cern/blog/cling-in-llvm/ and https://blog.llvm.org/posts/2020-11-30-interactive-cpp-with-cling/ provide a discoursive inshights about the scope of the CLING project and the development history. They also provide some practical info about how to use cling and pieces of code. In my opinion, nobody who actually wants to install and use CLING has the patience to extract practical info from a blog, but rather expects to find those information in the CLING web page, or on the CLING github's page. At this point, the most organized CLING webpage is the following: https://root.cern/cling/#download, which nevertheless lacks details and structure.

At first, we might think to work to extend and improve the https://root.cern/cling/#download page, creating a new *Overview* section where to insert links to the blog descriptive pages. Nevertheless, the final goal should be to eliminate those links so that the reader will find everything on the CLING webpage only. A brief description of the purpose of CLING should be included in the Overview, more details about CLING's functionality should be found in the *User's Guide*. A first draft can be provided by the Technical Writer (TW) by gathering all the blogs/videos existing, extracting the useful information, and rewriting them in a consistent, coherent form. The CLING developer's team should then provide feedback on this draft so that the TW can improve it and finalize it with updated information.
note: since one of the purposes of the CLING team is to make CLING independent from ROOT, perhaps a new dedicated webpage should be created *ex novo* rather than working on the root.cern/cling.

**AUDIENCE:**
The second issue I want to point out might to give insights about how to get from N to N+100 new users. What kind of audience could be interested in CLING? I do understand that CLING originates from ROOT, and was developed having the scientists from CERN as user target, who usually work with large amounts of data previously collected. In this case, compiling interactively might give the advantage of speeding up the process. Would CLING be useful for users whose code is dependent on real time data (eg. generated by hardware components?). Can we extract user case examples (i.e. pieces of code) for different audiences, so that people get a solid idea about why they should use CLING? Define a better user case, dividing examples for different areas (physicists from CERN/data science/general?). Defining the audience will also allow us to define the structure of the CLING webpage. For instance, an audience of scientists from CERN, who is familiar with ROOT, might be interested at a chronological history of CLING's development, since it would allow this audience to grasp the reason why CLING was developed and what advantages it has over the previous versions of C++ interpreters. It should look something like this (forgive me if the descriptions aren't correct, it's just a draft):

a. 19XX → ROOT, a software package based on XX was developed to help scientists from CERN to analyze large amounts of data.
b. 19XX → LLVM, a set of compilers and toolchain technologies, was developed by XX in order to XX
c. 19XX → CLANG, a compiler front end for C/C++/etc was developed by XX. CLANG allows users to XX.
d. 19XX→ CINT, a C++ interpreter for ROOT, was developed
e. 2014 → CLING, an interactive interpreter for C++ was developed by XX as part of the ROOT project, with the intention to use XX .
f. 20XX → we decided to move most of CLING onto the LLVM platform. This strategy has the following advantages: XX, XX, XX. Our vision is to further XXXX.

Nevertheless, this information would be irrelevant to an audience of pure software developers with no insights and no interest at the physics behind CLING development. For this audience, I would rather focus on the description of the reason why they should consider using CLING, and provide them with a multitude of practical examples (i.e. pieces of codes) that they can easily copy/paste into their terminal to get a real grasp of the software's potential.

**INSTALLING AND EXECUTING:**
Browsing through the project's issue page https://github.com/root-project/cling/issues?page=1&q=is%3Aissue+is%3Aopen I noticed that many potential users are having troubles at installing or executing CLING. A priority goal, and the first step to accomplish, should be the following: **how to get to the point that everybody who wants to install CLING can do it easily and in a short time.** The installation section should therefore always be updated, dependencies should be explicitly indicated, new releases/modifications should be included, and a step-by-step installation guide should be provided. Somebody from the team should be nominated as reference and should take care of maintaining this section updated.

Tutorials, use case examples should be in a form that the user can use as such and understand how to proceed in practice. I see tutorials scattered on many pages, such as:
https://root.cern/cling/#download,
https://notebooks.gesis.org/binder/jupyter/user/quantstack-xeus-cling-be48kuw4/notebooks/notebooks/xcpp.ipynb,
https://blog.llvm.org/posts/2020-12-21-interactive-cpp-for-data-science/,
https://blog.llvm.org/posts/2021-03-25-cling-beyond-just-interpreting-cpp/,
https://blog.llvm.org/posts/2020-11-30-interactive-cpp-with-cling/.
The tutorials should be collected in one document, reviewed by the team, and then re-arranged in the CLING webpage, making sure that code examples are organized from simple to complex cases. The code should be easy to copy-paste into the user's terminal. The process of rewriting them should follow a logic hierarchy: showing advanced features only when simple features

have been already described. A problem specific to the CLING project is that the software aims at different kinds of audiences who are therefore interested in different kinds of examples. Would it be useful to keep audiences separate into categories, providing practical examples separately, or it would be dispersive and generate chaos? This should be discussed with the team.

**This is, in my opinion, a draft of the essential feature that the CLING webpage/GitHub page should provide:**
note: the reference documentation is now empty, but this same draft, or an improved version of it, might be used later in order to organize the existing documentation into logical classes.

| NAME OF THE SECTION | CONTENT OF THE SECTION | REFERENCE DOCUMENTATION | COMMENTS | PRIORITY (from 1-most important to 3-least important) |
|---|---|---|---|---|
| **Overview** | 1. Overview of CLING's functionality It should begin with something like: *CLING is an interactive C++ interpreter. CLING enables you to …* 2. History of CLING development (if we decide to insert it) | https://blog.llvm.org/posts/2020-11-30-interactive-cpp-with-cling/ <br><br> https://blog.llvm.org/posts/2021-03-25-cling-beyond-just-interpreting-cpp/ | Use visuals rather than descriptive language. Videos or screenshots. When visuals are used: text on the left and graphic on the right, both explaining the same concept | 2 |
| **Release notes** | It very briefly describes the last release updates, specific changes included in CLING. It should link to a changelog section for more detailed info | | A member from the team should be in charge of maintaining it constantly updated | 1 |
| **Installation** | **1.Introduction**→describes the purpose of this document <br><br> **2.Pre-installation and dependencies**→ explains what hardware and software are needed for your product to function properly. Example: *CLING can be installed on CONDA or on Docker Hub. If you use CONDA, you can install with:* → there should be one line command that can be easily copied/pasted. <br><br> 3.Step-by-step **installation procedure** <br><br> **4. Troubleshooting**→ describes how to fix common issues encountered during installation <br><br> 5.**Uninstalling procedures** | | Style should be structured writing: heading, small blocks, breaking down complex instructions into simple, one-concept/component instructions organized into numbered lists. It helps the reader to stay more concentrated on the procedure | 1 |
| **Starting CLING** | After installation, what do I need to execute to actually start cling (e.g. write cling in terminal and maybe a few simple code examples) | | It should be kept as simple as possible | 1 |

| | | | | |
|---|---|---|---|---|
| **User's Guide** | Explains what is CLING, contains examples and tutorials of different use cases | | Organized logically from simple (Hello World) to complex. Eventually, organized by category (CERN related, data science, etc). All of the codes should be easy to copy/paste, a quality check should be made to make sure that they are functioning. All of them should be introduced by a description of the code purpose, generated output, eventually with the aid of visuals if useful. Better to start with a few, working examples rather than having many chaoticones. | **2** |
| **Reporting an issue** | Gives the user possibility to report problems with the software (bugs, installation problems) | | The TW will discuss the following with the team: where should the user report an issue? GitHub? Is there any specific template to report an issue? https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/configuring-issue-templates-for-your-repository and https://marker.io/blog/bug-report-template could be used as a refererence | **1** |
| **Developer Guide** | 1.Code style (example Google style code, etc)<br>2.Code of conduct<br>3.Special dependencies<br>4.Pull request template | | Having a developer guide it's in my opinion essential in order to get experienced users to contribute to the project. The code of conduct should be detailed and well defined. Nevertheless, it is less priority than the User's Guide | **3** |
| **CLING Changelog** | A list of chronological order of all the changes in the software (es. added/improved/removed features) | | One team's member in charge of this section | **2** |

**Timeline:**
**note: TW=Technical Writer // Team=the team of CLING's developers**

| | FROM-TO | COMMENTS |
|---|---|---|
| **Before starting:**<br>Discussion of this draft with the team, the TW will receive comments from the team so that the plan can be improved. A new, updated plan will be then generated by the TW. | May 3-9 (this week) | The team should help the TW to define the priorities. |
| **"State of the art"**:<br>Reading of all existing material, extraction of information from all the sources and classification of those into logical classes such as: info about the | May 9-20 | This part involves a lot of solitary reading from the technical writer, but also dialogue with the team when some parts are not clear. In particular, the team should help the TW to |

| | | |
|---|---|---|
| history of cling, cling purpose, tutorials, installation guide, user guide. | | understand when an information is out of date and therefore should be eliminated, and where new material should be generated. |
| **Plan and analyze**:<br>1.Assessment of Audience<br>3. Define Team and roles | May 21-23 | Roles should be defined now as much as possible (who, from the team, will be in charge of improving the installation manual, who will work on the tutorials, etc) |
| **Sketch of CLING manual:**<br>1.Decide the outline<br>2. Organize rationally<br>3.Offer info about the SW | May 23-June 30 | This segment is the main part and must provide detailed and concrete information on how to use CLING. It should begin with the basic setup required, then move further to numbered and logical steps that define how the app should be utilized. |
| **Write a readable manual** | July 1O - October 15 | Priority should be given to existing users (i.e. by generating an installation and how-to-start guide that are reliable and updated).<br>Following this, tutorials should be included starting from easy to more advanced.<br>The writing standard should aim at:<br>1.Explain with graphics and visuals together with short text<br>2.Use numeric lists<br>3.Provide instances:code examples<br>4.Write in active voice and simple vocabulary<br>5.Keep it brief |
| **Test the new CLING webpage** | October 15 - October 30 | Assigning to a group of testers who are not acquainted with the SW might be a strategy: can they install CLING easily? Do they understand CLING's purpose? Can they code using CLING easily? Perhaps, a first test should be done after the *Installation* and *Starting with CLING* sections are written. |
| **Keep CLING updated** | From October on | This should be the goal of the CLING team after the SoD ends. The team should define who is in charge of upgrading info about SW modification, answering to reported issues, etc. |

**Summarize of the identified steps:**

1) Review of the existing documentation, identifying what should be moved in each section
2) Writing of the *Release Notes*, *Installation*, S*tarting CLING*, *Reporting and Issue* sections. This way, we make sure that everybody who already knows CLING but had issue at installing it can now install and execute CLING
3) Writing of the *Overview*, *User's Guide*, *Changelog* section
4) Writing of the *Developer's Guide* (optional, I am not sure that there will be enough time but it can be at least drafted)