**Documentation Preparation steps:**

* Define the document type.
* Collect and analyse input files from SMEs.

SME:

* Who - Software developers, QA Testing, Product specialist.
* What – Provide input, specifications, safety notice.
* How – Word, chm, images, videos.

**Steps:**

* Receive task request.
* Collect and analyse all the available input information.
* Involve SMEs and stakeholders.
* Question and answer session with SME for any clarification.
* Identify the document type.
* Define front matter and metadata for the document type.
* Initiate document creation.

**Git Hub:** (Cloud server)

Timeline

Description automatically generated

**Git & GitHub** are tied closely together to make working with them both a seamless experience. While Git takes care of the underlying version control, GitHub is the collaboration platform built on top of it. GitHub is the place for pull requests, comments, reviews, integrated tests, and so much more. Most developers work locally to develop and use GitHub for collaboration. That ranges from using GitHub to host the shared remote repository, to working with colleagues and capitalizing on features like protected branches, code review, GitHub Actions, and more.

**Steps:**

* Create a GitHub account.
  + Onetime process.
* Create a standalone repository.
  + A standalone repository must be created for each document.
* Define standard rules for repo and branches.

**Git:** (using Command line)

* Git is distributed version control software. Version control is a way to save changes over time without overwriting previous versions. Being distributed means that every developer working with a Git repository has a copy of that entire repository - every commit, every branch, every file.

Learning & Mastering Git Commands:

If you're getting started with Git, a great place to start is the Git Cheat sheet. It's translated into many languages, open source as a part of the github/training-kit repository, and a great starting place for the fundamentals on the command line.

Some of the most important and most used commands that you'll find there are:

***git clone [url]:*** Clone (download) a repository that already exists on GitHub, including all the files, branches, and commits.

***git status:*** Always a good idea, this command shows you what branch you're on, what files are in the working or staging directory, and any other important information.

***git branch:*** This shows the existing branches in your local repository. You can also use git branch [branch-name] to create a branch from your current location, or git branch --all to see all branches, both the local ones on your machine, and the remote tracking branches stored from the last git pull or git fetch from the remote.

***git checkout [branch-name]:*** Switches to the specified branch and updates the working directory.

***git add [file]:***Snapshots the file in preparation for versioning, adding it to the staging area.

***git commit -m "descriptive message":*** Records file snapshots permanently in version history.

***git pull:*** Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. git pull is a combination of git fetch and git merge.

***git push:*** Uploads all local branch commits to the remote.

***git log:*** Browse and inspect the evolution of project files.

***git remote -v:*** Show the associated remote repositories and their stored name, like origin.

**Steps:**

* Create input and output folders and organize the input files in compliance with folder structure.
* Initiate Git (.git folder).
* Clone the repository. (git clone)
* Establish remote connection of the repository with .git folder and sync all the available files to GitHub cloud.
  + Type the following in the cmd prompt and press enter for each code of line.
    - git remote add origin <repo link>
    - git push -u origin main
    - git push –set –upstream origin main
* To sync, use Git Push
  + - Git commit
    - Git add . (To add all files to the staging area)

**Hugo:**

Hugo is a general-purpose website framework. Technically speaking, Hugo is a static site generator. Unlike systems that dynamically build a page with each visitor request, Hugo builds pages when you create or update your content. Since websites are viewed far more often than they are edited, Hugo is designed to provide an optimal viewing experience for your website’s end users and an ideal writing experience for website authors.

Getting started: <https://gohugo.io/getting-started/quick-start>

**Steps:**

* Create a site.
  + *hugo new site <sitename>*
  + Download the hugo theme ‘Relearn’ and place it inside the themes folder. <https://mcshelby.github.io/hugo-theme-relearn/index.html>
  + Edit the config file to add the theme name.
* Add content.
  + Create pages: 1. Home page, 2. Chapter & 3. Default page.
    - *hugo new --kind home \_index.md*
    - *hugo new --kind chapter <name>/\_index.md*
    - *hugo new <chapter>/<name>/\_index.md*
  + Introduce markdown syntax. Refer <https://mcshelby.github.io/hugo-theme-relearn/cont/markdown/index.html>
* Review content and configure the site.
* Publish the site locally and do a final review.

**Render:**

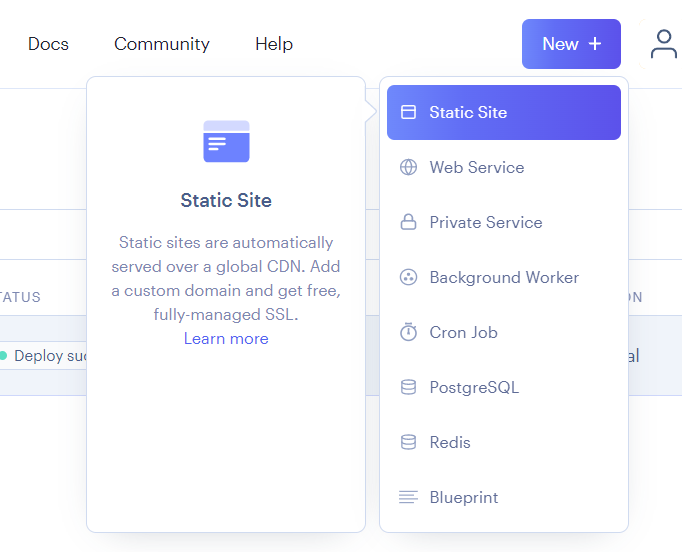
Static Sites

Deploying static sites on Render is incredibly easy. Just link your GitHub or GitLab repo — we build your site every time you push to your repo and serve it over a global CDN with fully managed TLS certificates.

Static sites on Render are free, with no cost at all to you unless you go above 100 GB of bandwidth per month.

**Steps:**

* Sign in into Render using GitHub account. (One time process)
* Click New to create a Static site.



* Connect a GitHub repository that you wish to deploy.

Graphical user interface, text, application, email

Description automatically generated

* Fill in the fields in the following format:
  + Name = <Name of the site>
  + Branch = Main
  + Root directory = <Project folder location>
  + Build command = hugo --gc –minify
  + Publish directory = Public (public folder inside hugo project folder)

Graphical user interface, application, Teams

Description automatically generated

* Click Create Static site.

**Description:**

The scope of this document is to briefly describe the processes and steps involved in Technical Documentation at Metamation.

*Boundaries:* Document request from customer – Document published to customer.

**Prerequisites:**

* Basic knowledge using command prompt.
* Git - GitHub
* Software and access.
* Documenting experience.

**Input:**

Inputs to start creating a document comes always from SME. A Subject Matter Expert can be a software developer, software tester, product specialist, and product manager.

File format includes:

* Doc
* Pdf
* Chm
* Markdown
* Images/videos
* Specification sheet
* Drawings
* Hazard statements

**Output:**

Static site website.

**Technical Documentation:**

Following are the steps involved in technical documentation process:

Step 1: Collect & analyze input.

Step 2: Identify document type.

Step 3: Create/organize repository.

Step 4: Add/Edit content.

Step 5: Language & Editorial review.

Step 6: Approval.

Step 7: Publishing and translation.

Step 8: Collect feedback.

**Roles & Responsibilities:**

Technical Writer

Technical Editor

Technical Documentation Specialist/Manager

**Document types:**

* In terms of data classification system.
* In terms of usage.
* Established documentation standards.

Software installation: Contact IT.

Folder structure: Storing input files - .md files – images – videos. Git upload/download.

Decide document front matter:

Document structure & Working instruction:

Hazard statements:

Language review: (Opt)

Team communication & notification:

Extras……..

Git:

* Create a project folder inside Hugo.
* Inside the project folder, open command prompt and initiate Git.
  + Type ‘git init’. hidden folder is created.
* Create input and output folders.
* Place the input files such as content file, images, videos, and diagrams in the input folder.
* Establish remote connection of the repository with .git folder.
* Type the following in the cmd prompt and press enter for each code of line.
  + git remote add origin <repo link>
  + git push -u origin main
  + git push –set –upstream origin main

Hugo:

* Create a project folder inside Hugo.
* Create a new site using command prompt.
  + Hugo new site <site name>
* Download the hugo theme ‘Relearn’ and place it inside the themes folder.
* Open the Config.toml file and add the theme as ‘relearn’.
  + **Note:** Name of the theme inside Config should exactly match the theme’s folder name.
* Add homepage and fill the content in Markdown format.
  + hugo new --kind home \_index.md
* Add chapter page and fill it with content.
  + hugo new --kind chapter <name>/\_index.md
* Add default page and fill it with content.
  + hugo new <chapter>/<name>.md

You can always check the output as a preview by typing:

Hugo server

Github:

<https://docs.github.com/en/get-started/quickstart/hello-world>

* Create a Github account. (One time process)
* Create a new repository.
  + Whenever we are working on a new task/document, it is mandatory to create a new repository.
  + When working on revision we can direct jump to existing project folder and repositories.
* Add stakeholders to this repository as collaborators.