# Gitlab for contributors

## Overview

The Cloud Optimization Success (COS) Guidance team uses Gitlab to write, update, review, and version-control pre-production content.

We write in Markdown format, and we create one repository per whitepaper (which corresponds to how the XML is stored in Gitfarm). For more detail, see Well-Architected Gitlab methodology.

Follow these instructions based on your content type to use Gitlab for Well-Architected content creation, review, and publishing. These instructions differ slightly based on the type of guidance you are authoring.

# Framework and pillar guidance

Expand

# Lens whitepapers

Expand

# Guidance whitepapers

Hide
Each guidance whitepaper has its own individual project in the WA Guidance repository.

Guidance whitepapers differ in a few ways than lenses:

- Guidance whitepapers are not required to be organized by pillar
- Guidance whitepapers do not always have best practices (although they can have best practices if the authors desire it)

# Creating a new guidance whitepaper

To create a new guidance whitepaper, you need to fork the wa-guidance-template repository:

1. In the `wa-guidance-template` repository, in the top right-hand corner, choose **Fork**. **NOTE:** If you don't have access, slack @smatzek for access to the Lens guides Gitlab repository.
2. Fill out the following information:
    1. **Project name:** Enter the name of the guidance.
    2. **Project URL:** Select the correct namespace. For guidance, this is `cloud-optimization-success/well-architected-content/wa-guidance`.
    3. **Project slug:** Enter a URL slug for the guidance. This should just be the full name of the guidance whitepaper with dashes instead of spaces. For example, for a guidance whitepaper about generative AI, this might be something like generative-ai-guidance.
    4. **Project description:** (OPTIONAL) Enter a description for the guidance.
    5. **Branches to include:** All branches.
    6. **Visibility level:** Internal.
3. When all details are filled out, choose Fork project.
4. After the fork completes, you can see your new project created. This is the project where you will perform your work.
5. Feel free to delete the README.md file once your project is forked.

# Working in the guidance whitepaper

Now that you have your guidance project forked, you can begin updating the template files.

There are two primary ways to do this:

1. Update files in Gitlab
2. Update files locally, and push the changes to Gitlab using the CLI or Git application

If you are comfortable cloning a repository and making changes using the CLI, you can ignore these instructions.

Keep in mind the following:

- Guidance whitepapers only have primary template files:
    - **`guidance-body-template.md`**: Contains all required headers, and provides instructions for authoring guidance.
    - **`guidance-section-template.md`**: Duplicatable template file for any guidance-specific sections that do not correspond to any set subject. Provides instructions for authoring sections of the guidance that are unique to each whitepaper.
- Do not change any of the template header elements, including header level or title. You can add additional headers as needed based on your content.
- **For best practices:** (OPTIONAL) All best practices should be written in their own file and placed in the `best-practices` folder. You can duplicate the best practice template file contained in that folder for as many best practices as necessary.
- Each best practice title should be prefixed by a guidance-specific prefix.
    - For example, in a guidance about machine learning, the prefix could be ML. That means that

## Updating a file using the Gitlab UI

In guidance papers, because much of the body content is written in one single file, it's best to have authors work in separate branches so that their edits can be consolidated once they are ready for review.

Alternatively, you can have each author work in sequence and create a merge request when they are complete with their changes.

Keep this ideology in mind when you are creating branches.

1. Navigate to the file you wish to edit.
2. Open the file.
3. In the Gitlab display window for the file, choose the **Edit** dropdown, then choose **Edit single file**.
4. Make your edits in Markdown. It's helpful to keep a guide to Markdown formatting up when you're editing.
5. Once you are finished editing the file, in the top right-hand corner, choose **Commit changes**. You will see a **Commit changes** dialog pop up.
6. In the **Commit changes** dialog, enter a commit message that describes the changes you made.
7. Under **Branch**, choose **Commit to a new branch**.
8. Enter a branch name for your changes. Usually, this is something like `filename-authorname`.
9. Choose **Commit changes**.

## Create a merge request for review

Once all of the authors' changes are consolidated:

1. In the left-hand navigation, choose **Code**, then choose **Compare revisions**.
2. In the **Source** dropdown, choose **main**.
3. In the **Target** dropdown, choose the branch you wish to merge into main.
4. Choose **Compare**.
5. Choose **Create merge request**.
6. Fill out the merge request with the following required information:
    1. **Title:** Enter the best practice title as the title of the merge request.
    2. **Description:** Enter a brief description of the changes that were made to the best practice.
    3. **Assignee:** This can be left blank for now, but should eventually go to the reviewer for your best practice.
7. After you've filled these details out, choose **Create merge request**.

## Copying section template files in the Gitlab UI

There's no direct way to duplicate a file in the Gitlab UI. Instead:

1. Open the `guidance-section-template.md` file.
2. In the right-hand corner of the file dialog itself, choose the **Download** button.
3. Rename the file to the name of the new file. **NOTE:** Make sure you rename the file! Git doesn't like having two files with the same name.
4. Navigate one step back to the folder you were previously in.

6. In the **Commit changes** window:
    1. Choose the file to upload.
    2. Enter a commit message.
    3. Choose **Commit to main**.
7. Then, choose **Commit changes**.

## Requesting review for your updates

We use *merge requests* to review and tech edit updates to our documentation.

In merge requests, guidance leads, lens contributors, SMEs, and tech writers can add comments, suggest changes, and make edits directly to the changes you've made.

To request review for an update, you can simply assign the merge request to the correct person or @ mention them in the merge request (or both).

For more detail, see Documentation review.

Tags: