

Keeping up with Documents in Ruby on Rails

[Starting from scratch?](#)

[Docker installation:](#)

[Docker Image and Container:](#)

[Github Actions:](#)

[Making a new Repo for your project:](#)

[Uploading your project to the new Repo:](#)

[Additional Details if needed for Github:](#)

Starting from scratch?

With Ruby, there has been a ton of information over a short period of time. This document is created to help in the future to get started with getting a foundation for developing your project. Feel free to skip around the steps needed to adjust where you are when creating your project.

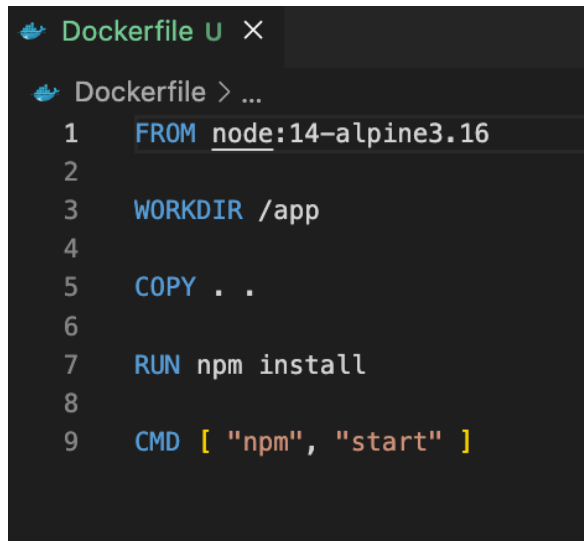
Docker installation:

To start from the very beginning, to have an easy and quick way to access your development environment, [Docker Desktop](#) is required. Docker Desktop will be used to manage your container to start work on your Ruby project.

Docker Image and Container:

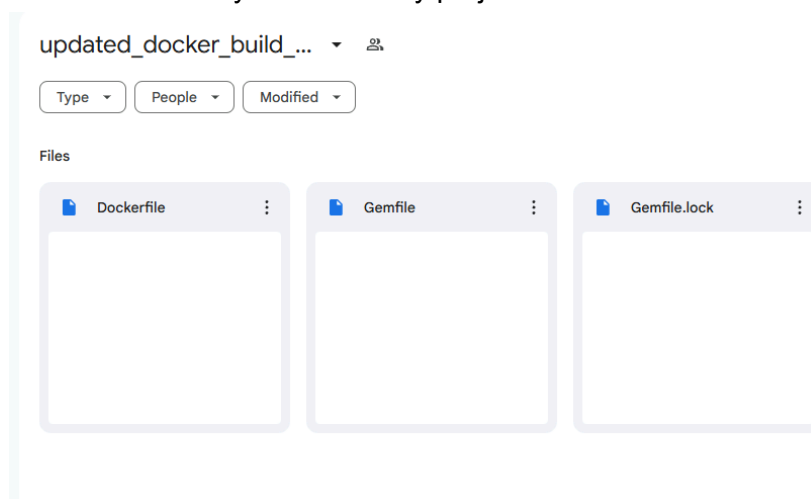
For the next step, a Dockerfile is required for the creation of an image. **Remember that a Dockerfile is essentially a text file that has the required commands to create an image.** In our case, our Ruby projects are referencing from this [Dockerfile](#).

From the [Medium.com](#), here is an example of what a Dockerfile will look like:

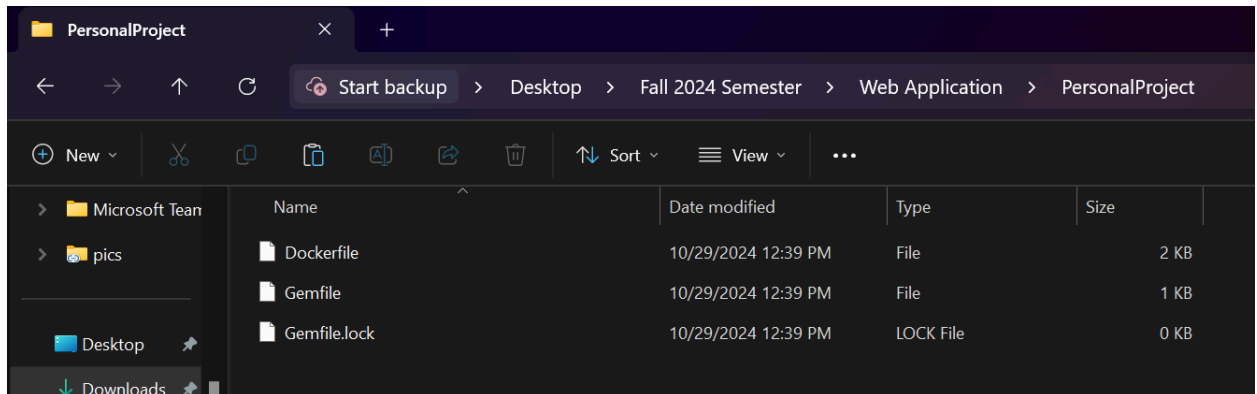


```
Dockerfile U X
Dockerfile > ...
1 FROM node:14-alpine3.16
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN npm install
8
9 CMD [ "npm", "start" ]
```

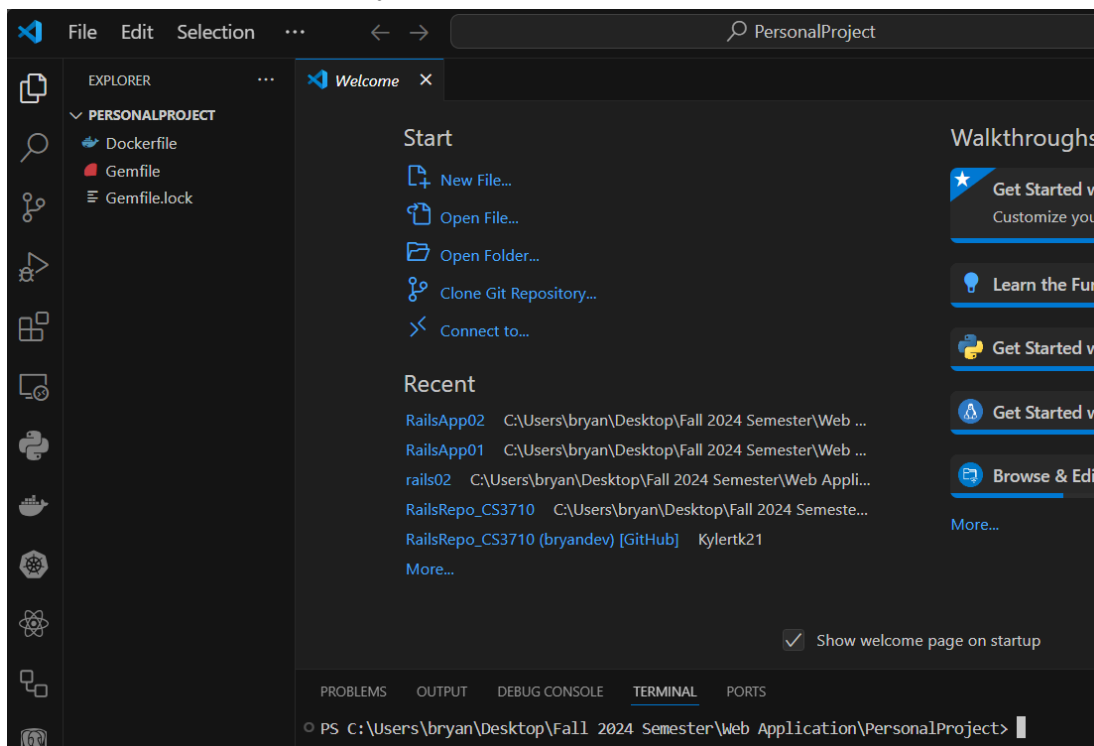
1. ***(Ignore this first point if there are no other files besides the Dockerfile.)*** After downloading Dockerfile, notice that there are a couple of other files listed within the folder listed as “Gemfile” and “Gemfile.lock”. These files are needed for Ruby, but not needed to create your new Ruby project. You can leave these together for the moment.



2. Next after downloading the files, create a new folder and move the new downloaded files into the folder.



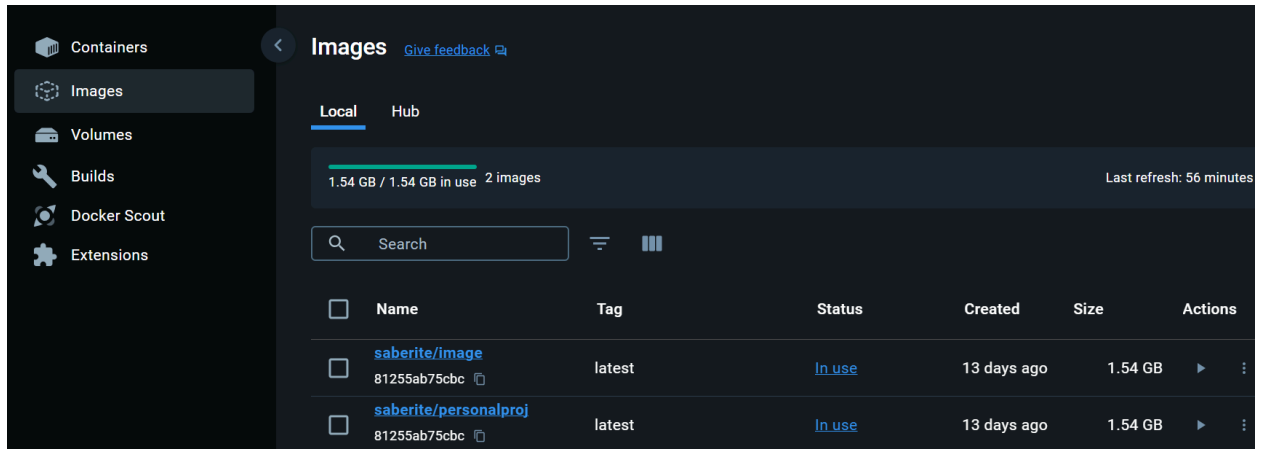
3. Following suit, load up your VSCode Studios and you want to open the folder with the files inside. Then make sure your terminal is opened



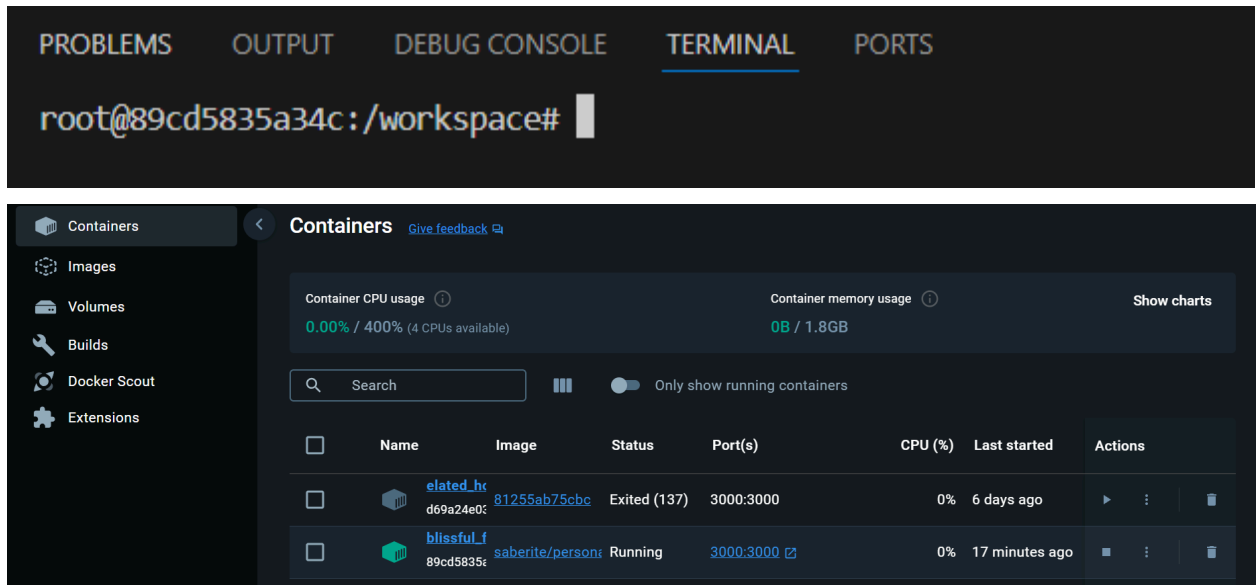
4. **IMPORTANT STEP:** When inside the terminal, the following command will use the Dockerfile to create an image for your usage. The following command requires that you have a **Docker Username**. The command is:
 - a. **Docker buildx build -t [dockerUserName]/[Image_name] .**
 - b. Or use **docker build -t [dockerUserName]/[Image_name] .**
 - i. The [Image_name] can be anything you want to name the image (image of creation of image)
 - ii. **Do not forget the “.” in the command!**
 - iii. Note that the image creation can take some time....

```
\Web Application\PersonalProject> docker buildx build -t saberite/personalproj .
```

5. Take notice after the image is created, it will be listed under “Images” in Docker Desktop



6. Next head back to your terminal, you need to start the image from your terminal to create a container. **Keep in mind what terminal you are using.** The commands are as follows:
- Windows Powershell:
 - docker run -it -p 3000:3000 -v \${PWD}:/workspace [Image Name]**
 - Windows WSL 2, Mac/Linux Terminal or Git Bash
 - docker run -it -p 3000:3000 -v \$(pwd):/workspace [Image Name]**
 - Command Line(CMD)
 - docker run -it -p 3000:3000 -v %cd%:/workspace [Image Name]**
7. Remember that the [Image Name] is the name you gave the image when you created the image.
8. You will know you done the correct steps when your terminal looks like this with “#” before your inputs and congrats you have a container listed in Docker Desktop now for your image:



9. Important to also remember to when connecting back to a container there are a few steps:

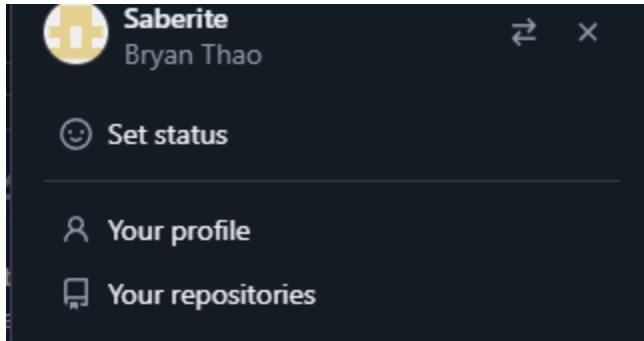
- Ensure you start the container in Docker Desktop
- Run this command in your terminal: `docker exec -it [container_name] /bin/sh`

Github Actions:

Making a new Repo for your project:

If you already have a development repo, refer to [here](#).

Go to your Github account and click on your profile picture. We want to navigate to "Your Repository"



Next, select “New” for a new repository



Fill out the fields and make sure to leave these options:

- “.gitignore” as **NONE**
- **Set to “public” for anyone to view the repo**
- **Do not add a README file. You will be using a project board for the most part**

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * Saberite ▾ / **Repository name *** TestName

✔ TestName is available.

Great repository names are short and memorable. Need inspiration? How about **stunning-lamp** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Uploading your project to the new Repo:

This is the last important step to ensure you are set to develop on your new rails project. This section is designed to make sure your work is correctly getting uploaded to your Github Repo!

(Image from M03 L05 From Professor Deb)

```
git status
git add .
git commit -m 'test branch change'
git push --set-upstream origin branchTest
```

The image above is the short-answer to how to push your project to your github repo, Here are the steps and why:

1. **Git status** : Gets the current directory you are working. Important to know what you are pushing!
2. **Git add .** : Place all files in the directory for uploading.
3. **Git commit -m "[Message here]"** : This is allow you note what you added to the changed/new files uploaded to the Github repo
4. **Git push --set-upstream origin [branch]** : **THIS STEP IS HIGHLY IMPORTANT TO DO**. This command allows you to connect the project you created to the github repo
 - a. As a side note, ensure you signed in with Github in VSCode or your coding environment as **it's required for uploading**.

Additional Details if needed for Github:

A good standard is to have two different branches. Below is a continuation of the last image to help with creating another branch to house a baseline for your project.

```
git checkout main
git status
git merge branchTest ← Integrate branch
                        changes with main
git status
git push
```

To explain:

1. After using **Git push --set-upstream origin [branch]**, you can create another branch or set to the "Main" branch. This can be done by using "**Git checkout main**"
 - a. If you want a new branch to merge to, use these commands and then continue:
 - i. **Git branch [name_of_branch]**
 - ii. **Git checkout [name_of_branch]**
2. You can use **Git status** to ensure you are in the correct directory
3. When Merging with the command **Git merge [branch]**, the branch you are on currently is replicating everything in **[branch]** you want.

- a. For example:
 - i. If on the main branch...
 - 1. Using git merge [branch]
 - a. Main will copy everything in [branch]

And now you are all set for development!