


# TIPS FOR DELIVERING AN EFFECTIVE LAB REPORT

Abdelbacet Mhamdi

Senior-lecturer, Dept. of EE

Institute of Technological Studies of Bizerte — Tunisia

abdelbacet.mhamdi@bizerte.r-iset.tn

 a-mhamdi

**Abstract** — We present some guidelines to follow in order to prepare well your labs. The main programming language you are going to use to implement some general purpose applications is Julia. The codes you will develop run on top of a Docker image, ensuring a consistent and reproducible environment. You will use one of these two interactive tools: Jupyter Lab or Pluto. It is preferable to write your lab reports in Typst, given the provided files.

**Index terms** — Julia, Typst, GitHub, Docker, Lab Report

## I. JULIA

The programming language we are going to learn through this module is Julia. Figure 1 shows its logo. It is chosen for:

- its high-performance computing capabilities,
- expressive syntax, and
- extensive ecosystem.

Julia is an ideal language for scientific computing, data analysis, and numerical simulations [1]–[3]. A quick reference guide for key concepts, syntax, and formulas is available at [cheatsheet.juliadocs.org](https://cheatsheet.juliadocs.org).



Figure 1: Julia logo

Insert your code in each lab report. The detailed explanation of the functions, along the packages you have used is a must. After each code snippet, add a screenshot that showcases your running work when applying the test provided in each exercise.

## II. TYPST

Consider using Typst to write your lab reports. The provided templates allow you to focus on the content and seamlessly create a professional-looking report.

Typst supports Markdown syntax, which provides a range of formatting options [4]. Here are some points to help you write your report:

### 1. Formatting Text:

- Surround a text with single asterisks (\*) to make it bold
- Use single underscores (‘\_’) around your text to emphasize it
- To create headings, use equal signs (=) followed by a space at the beginning of a line. The number of (=) symbols determines the heading level.

### 2. Inserting Objects:

- Use this syntax if you need to insert an image:

```
#figure(  
  image("IMAGE_NAME.EXT", width: 100%),  
  caption: [IMAGE_CAPTION],  
) <fig:LABEL>  
  
@fig:LABEL shows an image.
```

- Use this syntax if you need to draw a table:

```
#figure(  
  table(  
    columns: 4,  
    [Row 1], [a], [b], [c],  
    [Row 2], [1], [2], [3],  
  ),  
  caption: [Results],  
) <tab:LABEL>  
  
@tab:LABEL displays some results.
```

### 3. Creating lists:

- Unordered list: use a hyphen (-) followed by a space for each list item
- Ordered list: use a plus sign (+) followed by a space for each list item

### 4. Code snippets:

- Inline code: enclose the code within backticks (`)

- Block of code: use triple backticks followed by the word 'julia' to enable syntax highlighting

```
```julia
using Pkg
Pkg.update()
```
```

**REMINDER**

**IN EACH DOCUMENT, YOU HAVE TO INSERT WELL ANNOTATED SCREENSHOTS OF YOUR CODE AFTER BEING EXECUTED.**

You can leverage those features using the app's intuitive interface at the url [typst.app](https://typst.app), as shown in Figure 2. No installation is required, however, you may need to sign in in order to use the online editor. Keep an eye on your project size. Do not exceed 200MB. A fully fledged documentation on the usage of Typst is available at [typst.app/docs](https://typst.app/docs).

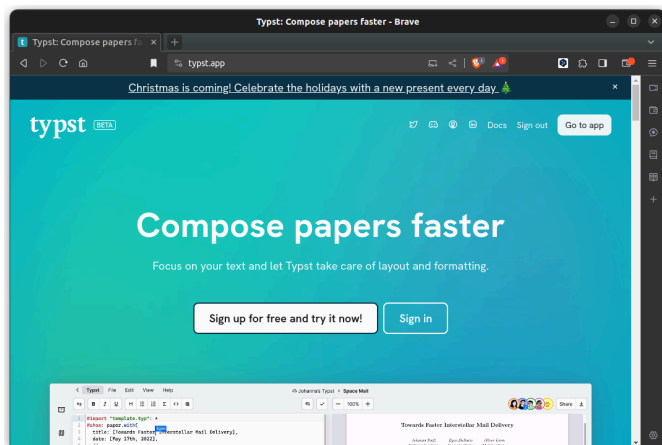


Figure 2: Typst app

### III. GITHUB

Share your code on GitHub. It's a fantastic way to foster a supportive coding community while gaining exposure to different coding styles and techniques [5]–[7].

### IV. LINKS BUNDLE

You may find the following links useful:

- GitHub Repository (*Figure 3*)  
[github.com/a-mhamdi/infodev](https://github.com/a-mhamdi/infodev)

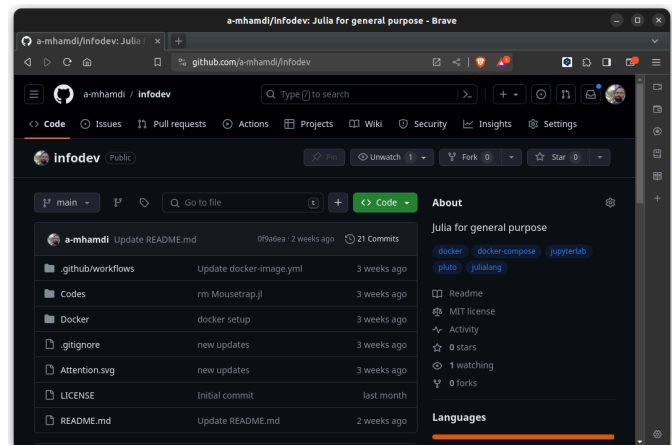


Figure 3: GitHub repository

- Docker Image (*Figure 4*)  
[hub.docker.com/repository/docker/abmhamdi/infodev](https://hub.docker.com/repository/docker/abmhamdi/infodev)

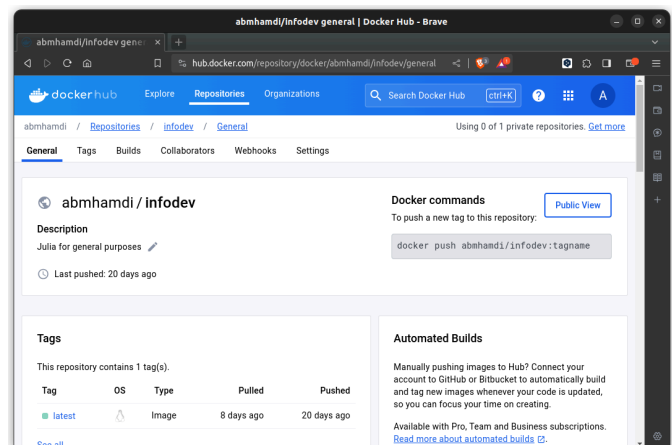





















Figure 4: Docker image

### V. TIMELINE

The following timeline is proposed to help you organize your work. It is not mandatory to follow it, but it is highly recommended to do so. The labs are designed to be completed in the order they are presented.

|                              | Feb.  |   |   |   |   | Mar.   |   |  |   |    | Apr.  |    |  |    |    |
|------------------------------|---|---|---|---|---|--|---|--|---|----|---|----|--|----|----|
|                              | W1  | W2  | W3  | W4  | W5  | W1   | W2  | W3   | W4  | W5 | W1  | W2 | W3   | W4 | W5 |
| <b>Lab #1</b>                |  |   |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Develop &amp; Code</i>    |  |   |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Review &amp; Update</i>   |   |   |  |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Finalize &amp; Submit</i> |   |   |   |  |   |  |   |  |   |    |   |    |  |    |    |
| <b>Lab #2</b>                |   |  |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Develop &amp; Code</i>    |   |  |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Review &amp; Update</i>   |   |   |   |  |   |  |   |  |   |    |   |    |  |    |    |
| <i>Finalize &amp; Submit</i> |   |   |   |   |  |  |   |  |   |    |   |    |  |    |    |
| <b>Lab #3</b>                |   |   |   |  |   |  |   |  |   |    |   |    |  |    |    |
| <i>Develop &amp; Code</i>    |   |   |   |  |   |  |   |  |   |    |   |    |  |    |    |
| <i>Review &amp; Update</i>   |   |   |   |   |   |   |   |  |   |    |   |    |  |    |    |
| <i>Finalize &amp; Submit</i> |   |   |   |   |   |  |  |  |   |    |   |    |  |    |    |
| <b>Lab #4</b>                |   |   |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Develop &amp; Code</i>    |   |   |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Review &amp; Update</i>   |   |   |   |   |   |  |   |  |   |    |   |    |  |    |    |
| <i>Finalize &amp; Submit</i> |   |   |   |   |   |  |   |  |  |    |   |    |  |    |    |
| <b>Exam</b>                  |   |   |   |   |   |  |   |  |   |    |  |    |  |    |    |
| <i>Review Session</i>        |   |   |   |   |   |  |   |  |   |    |  |    |  |    |    |
| <i>Evaluation</i>            |   |   |   |   |   |  |   |  |   |    |   |    |  |    |    |

## REFERENCES

- [1] S. Nagar, *Beginning Julia Programming*. in SpringerLink. Berkeley, CA: Apress, 2017.
- [2] A. SenGupta, *The little book of Julia algorithms*. 2020.
- [3] B. Lauwens, *Think Julia*. Beijing: O'Reilly, 2019.
- [4] T. Mailund, *Introducing Markdown and Pandoc*. Berkeley, CA: Apress L. P., 2019.
- [5] S. Guthals, *GitHub*, 2nd edition. in For dummies. Hoboken, NJ: John Wiley & Sons, Inc., 2023.
- [6] P. Bell and B. Beer, *Introducing GitHub*, 1. ed. Beijing: O'Reilly, 2015.
- [7] M. Tsitoara, *Beginning Git and GitHub*. Berkeley, CA: Apress, 2020.