Automating Processes with a Process Engine

Technical Preparations for Training Participants

Tiese Barrell

Version 1.1, 2022-10-17: update preparation descriptions and add experimental flag

Table of Contents

In this document, you'll find some preparation steps that ensure you can take part in the technical parts of the training.

Don't be put off by the length of the document; it's lengthy because there are instructions for various operating systems. You'll find the actual set up and check steps you need to do won't take all that long!

Technical Preparation for Assignments

To participate fully in the training, you'll need to do some technical preparations.

The training setup assumes you have access to a computer that you can use during the training. By *computer*, a desktop PC or laptop is meant; specifically, mobile phones and/or tablets are **not** sufficient for most parts of the training where a computer is used. The computer should be fit for software development purposes. The instructions that follow also assume that you have the permissions required to install and configure software on the computer and that you know your way around the computer and its software.

Operating System

It’s impossible to test on all operating systems, but most of the tools you’ll need are available for the most common operating systems. The training setup **should** work on computers that have one of the following operating systems:

- Windows
- macOS
- Linux

As usual, making sure that your operating system is relatively up to date with software and security patches will give the best results. The preparations assume you are familiar with your computer’s operating system and tools.

Developer Tools

The assignments where you need a computer make as few assumptions as possible of your background in software engineering and therefore use mostly general-purpose tools (specifically, not bound to a certain programming language) that most engineers are familiar with. Many of the tools used could be generally categorised as *developer tools*, for instance because they can be used from a command line prompt.

Web Browser

You'll need a modern web browser to access some information and applications. Make sure you have one of the following:

- Google Chrome or any other Chromium based browser, such as Microsoft Edge, Brave Browser, Vivaldi, etc.
- Mozilla Firefox or a derivative
- Apple Safari

Other web browsers such as Opera, QuteBrowser, etc. may work, but they are typically not officially supported by the tools we'll use, so your mileage may vary if they are the only option you have. In most cases, if you haven't been living in a cave for the past couple of years and have been browsing the modern web with your computer, you should have a web browser that will do just fine ;-) !

Text Editor and/or IDE

Make sure you have a tool handy that you can use to work with (mostly) text files, such as a text editor or IDE. The assignments don't require you to have many specific tools on top of basic text editing, so you can probably use the tools you are familiar with.

Here are some suggestions that are available for most operating systems.

IDEs

- IntelliJ IDEA
- Visual Studio Code
- Eclipse

Text Editors

- Notepad++
- Sublime Text
- GEdit

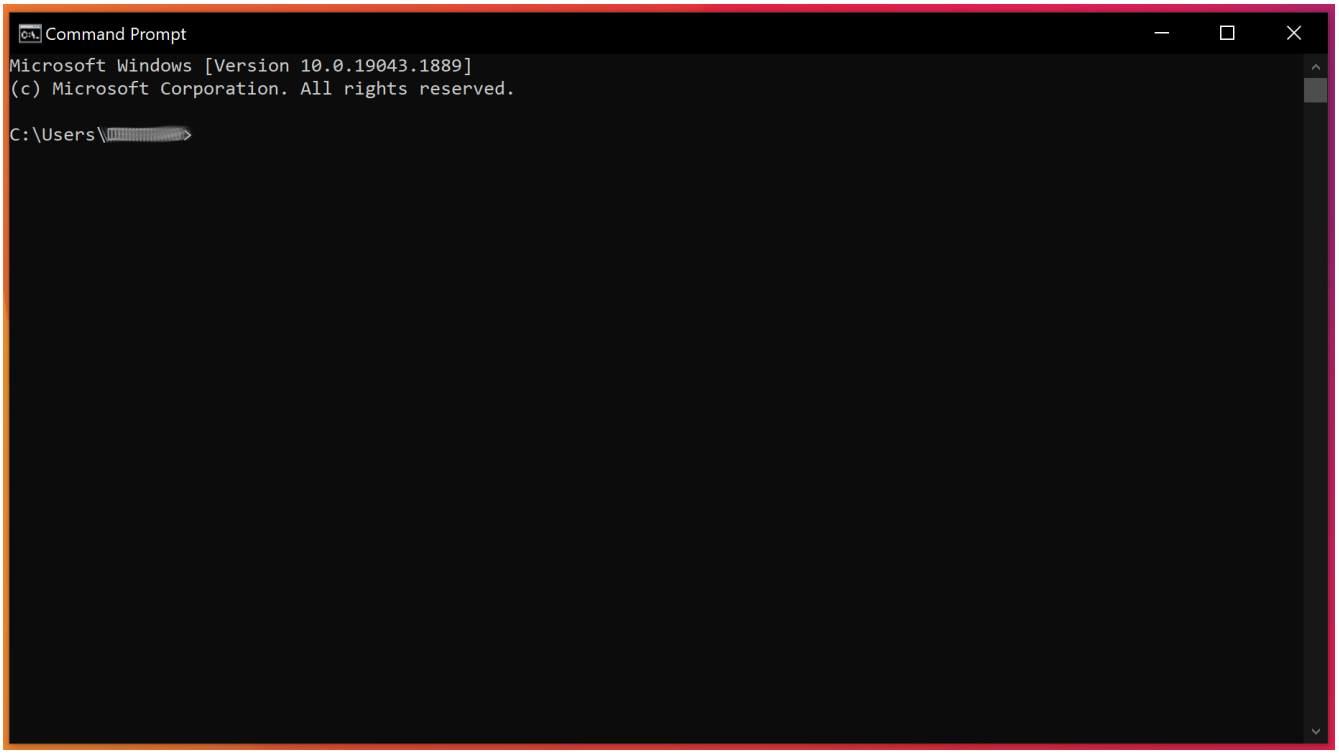
Terminal and Shell

As mentioned, many tools we'll use are first and foremost developer oriented tools. They typically have *command line interface*s (CLI) as the lowest common denominator for using them. Some have GUI tools on top of their CLI, with various offerings on different platforms.

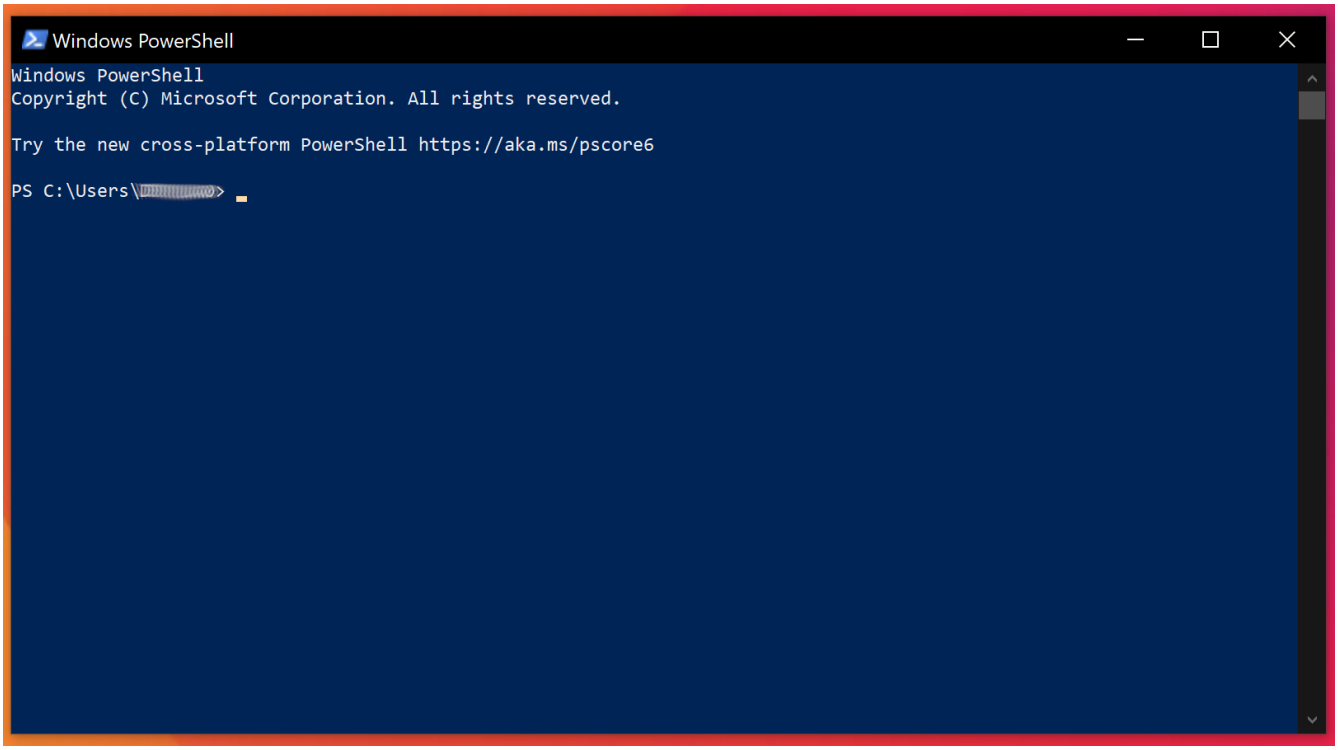
These tools can therefore be used from a *Terminal Emulator* program, which runs a *Shell* program, allowing you to enter (textual) commands and providing you with the output of the commands you enter. All operating systems come with a default terminal emulator program and a shell program. For what you need for this training, these default programs will suffice. You're also fine using a different one, if you prefer to do that.

Windows

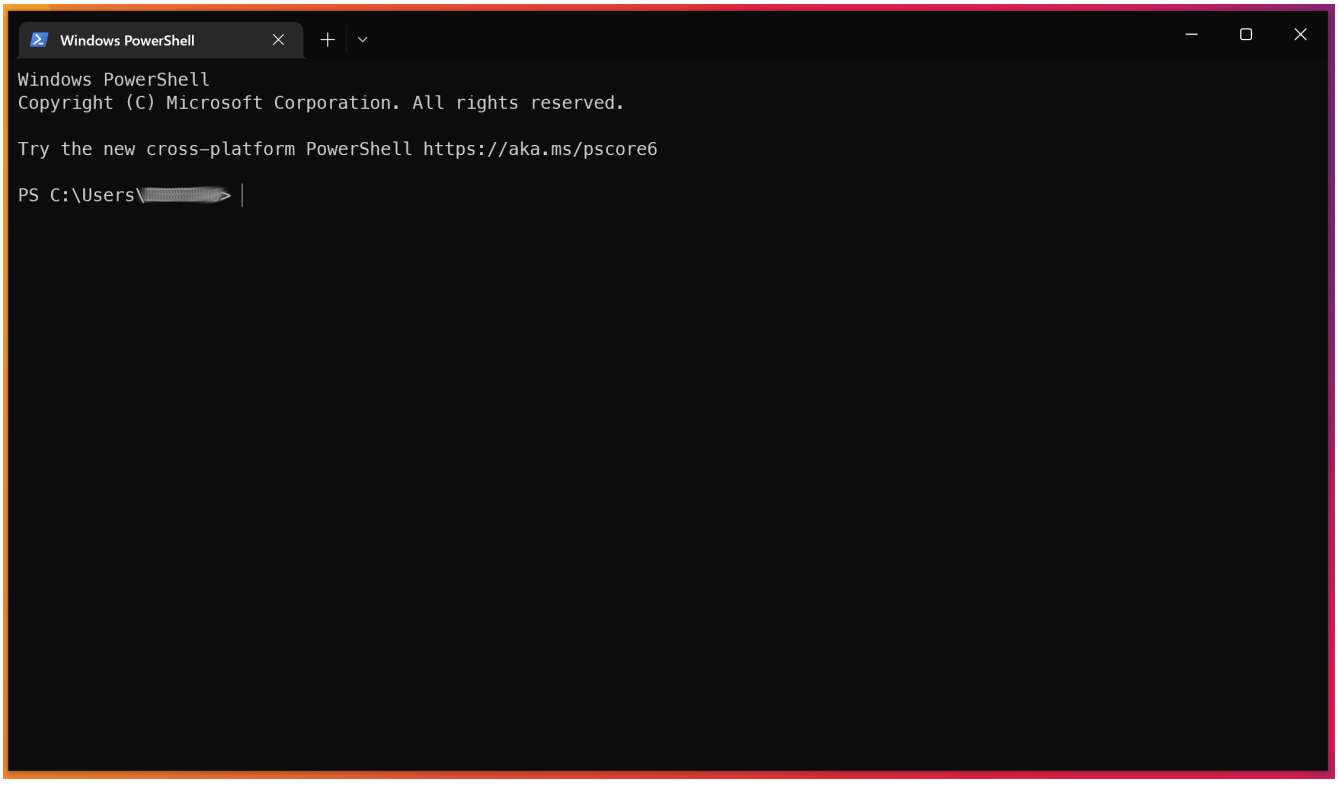
Recent versions of Windows come with at least the default *Command Prompt* program, which you can launch from the start menu by going to **Start › Search › Command** and pressing `Enter`.



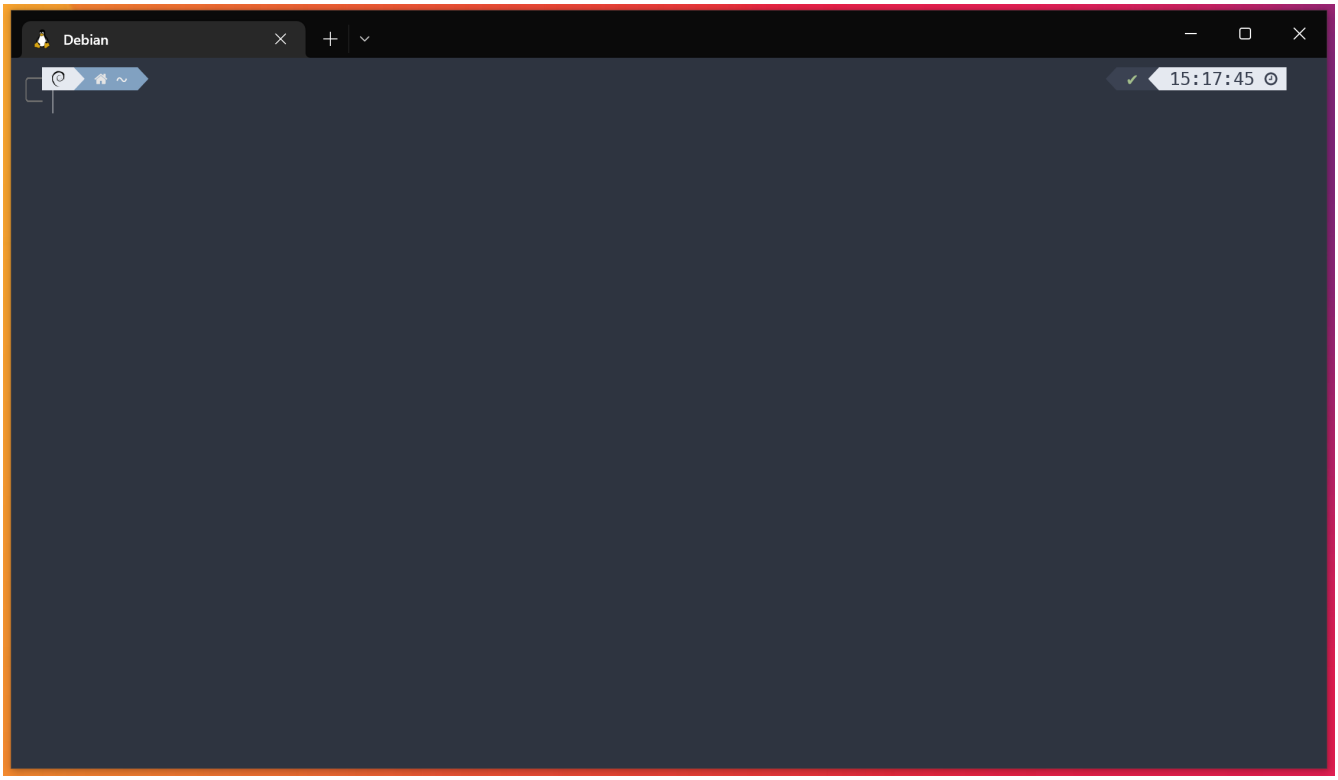
In addition, in most cases *Windows Powershell* is also installed, which is a more capable terminal plus shell combination. You can start it in the same way: **Start › Search › Powershell**, select from the list and press `Enter`.



You can also install the *Windows Terminal* from the Windows Store. Run it with **Start › Search › Terminal**, select from the list and press `Enter`.



Some developers prefer the experience of Linux command line tools when working on a Windows operating system. This can be achieved by using the *Windows Subsystem for Linux* (WSL). You can start one of the terminal applications mentioned above and then perform a command to enter the Linux operating system in the shell. The example below shows a Debian Linux system running inside Windows Terminal with WSL.

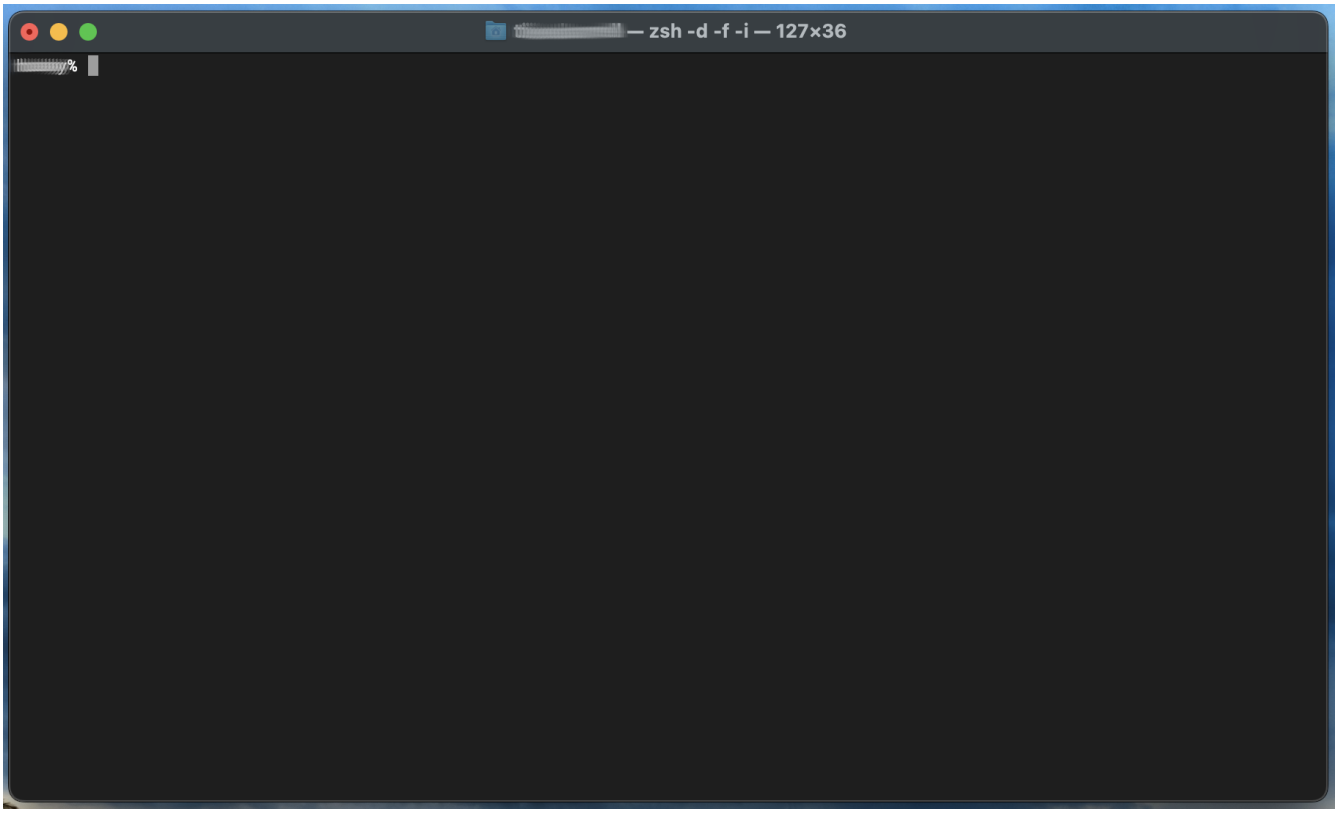


Any of these options should work fine for the assignments. Take note that if you wish to use WSL, you should install the other tools inside the Linux operating system, so the commands are also available there.

macOS

macOS comes equipped with the *Terminal* application out of the box. The *Z Shell* or `zsh` is the default shell used with Terminal. The Bourne Again Shell (`bash`) is also installed by default.

Open the Terminal application from the **Applications › Terminal** menu, or use *Spotlight* to start it by using `Cmd` + `Space` and typing *Terminal*, then selecting from the list and pressing `Enter`.



You can install many other terminal emulators on macOS, such as *Alacritty*, *iTerm* and *kitty*.

You can use any combination of Terminal and Shell you like to work with the command line tools for the assignments.

Linux

All Linux distributions come with a Shell program and all *desktop environments* ship a Terminal Emulator. Even without a desktop environment, you can install a terminal and shell programs and launch the terminal to get started. You'll find that the three popular shells *Bourne Again Shell* (`bash`), *Z Shell* (`zsh`) and *Fish Shell* (`fish`) are available from your distribution's repositories.

If you're on Linux, chances are you are already quite familiar with using command line programs, a terminal and shell.

You can use any combination of Terminal and Shell you like to work with the command line tools for the assignments. The commands were **not** tested with the `fish` shell.

Curl

The `curl` project provides two tools, the `libcurl` library and the `curl` command line tool.

The first is a client-side URL transfer library that supports a whole host of protocols. The `curl` tool can be used for getting and sending data with a URL syntax and uses `libcurl` under the hood.

When you install `curl`, you get both and can use the `curl` command to perform requests and read the results. We will use `curl` in some places to perform HTTP(s) calls, mostly to perform REST requests. You will be provided with the basic commands you can copy and paste to use `curl`.

GUI Alternatives

If you prefer a GUI tool over a command line tool, *Postman* is one of the best-known ones. You can also install an extension to most browsers that will offer similar functionality - these are normally called REST Client or something similar. Yet Another REST Client is an example of an extension for Chromium based browsers. Most IDEs also have some functionality to perform requests and inspect the results.

Windows

The `curl` program is pre-installed for Windows 10 and 11. Test whether you can use `curl` by going to your terminal application and run the following command.

```
$ curl -V
```

Note that the `-V` flag is a capital letter `V`. You should see the version information of the `curl` program being printed.

If you get a message that the `curl` program cannot be found, install it first from the website.

Check that you can access a URL with another `curl` command.

```
$ curl https://www.example.com
```

The result will be a bunch of HTML for the website, if the command is successful.

That's it for `curl`!

macOS

The `curl` program is pre-installed macOS. Test whether you can use `curl` by going to your terminal application and run the following command.

```
$ curl -V
```

Note that the `-V` flag is a capital letter `V`. You should see the version information of the `curl` program being printed.

If you get a message that the `curl` program cannot be found, install it first from the website or use a package manager like Homebrew to install it.

Check that you can access a URL with another `curl` command.

```
$ curl https://www.example.com
```

The result will be a bunch of HTML for the website, if the command is successful.

That's it for `curl`!

Linux

The `curl` program is pre-installed on most distributions. Test whether you can use `curl` by going to your terminal application and run the following command.

```
$ curl -V
```

Note that the `-V` flag is a capital letter `V`. You should see the version information of the `curl` program being printed.

If you get a message that the `curl` program cannot be found, install it first from the website or use your distribution's package to install it from the repositories.

Check that you can access a URL with another `curl` command.

```
$ curl https://www.example.com
```

The result will be a bunch of HTML for the website, if the command is successful.

That's it for `curl`!

Git

Git is used to work with the assignment project. The project is stored in a Git version control repository that you can *clone* and work with on your computer.

If you're not familiar with Git and its commands and operations, don't worry: the steps you need to take for the assignments will all be listed, so you can just follow along with the instructions.

If your editor/IDE has a built-in GUI for working with `git`, you can of course use that too. The same goes for any separate GUI application you normally use for Git, such as SourceTree, Git Tower or Gitkraken. The instructions will leverage shell commands, to remain agnostic of the specifics of certain tools.

Windows

Go to your terminal application and perform the following command to check that `git` is installed.

```
$ git -v
```

The version of the `git` program should be returned. If it's not, install Git first by going to the website and following the instructions there.

Once you have `git` available as a command, change into a temporary directory somewhere in your terminal application and test whether you can *clone* a remote repository.

```
$ git clone https://github.com/curl/curl
```

This will create a folder named `curl` in your temporary directory and copy all of the contents of the online repository into it and set up the most recent state of the project's source code as a workspace. You don't need the `curl` project's sources, so if you succeeded with this step, you can delete the directory and its contents.

For Windows, you can also install *Git Bash*, a GUI tool that incorporates some command line tools, such as the *Bourne Again Shell* (`bash`) and `git` itself.

macOS

Go to your terminal application and perform the following command to check that `git` is installed.

```
$ git -v
```

The version of the `git` program should be returned. If it's not, install Git first by going to the website and following the instructions there.

Once you have `git` available as a command, change into a temporary directory somewhere in your terminal application and test whether you can *clone* a remote repository.

```
$ git clone https://github.com/curl/curl
```

This will create a folder named `curl` in your temporary directory and copy all of the contents of the online repository into it and set up the most recent state of the project's source code as a workspace. You don't need the `curl` project's sources, so if you succeeded with this step, you can delete the directory and its contents.

Linux

Go to your terminal application and perform the following command to check that `git` is installed.

```
$ git -v
```

The version of the `git` program should be returned. If it's not, install Git first by going to the website and following the instructions there.

Once you have `git` available as a command, change into a temporary directory somewhere in your terminal application and test whether you can *clone* a remote repository.

```
$ git clone https://github.com/curl/curl
```

This will create a folder named `curl` in your temporary directory and copy all of the contents of the online repository into it and set up the most recent state of the project's source code as a workspace. You don't need the `curl` project's sources, so if you succeeded with this step, you can delete the directory and its contents.

Node

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. By installing Node on you computer, you can use source files written in Javascript and run them as programs with Node.

As with the other tools, don't worry about the specifics of Node and programming in Javascript; you'll only need to follow instructions to use it.

Make sure you have a recent version of Node installed on your computer.

Windows

Check the website for the current versions.

Run the following command in your terminal application to make sure your version is up to date.

```
$ node -v
```

The output should show the version of Node. If the program cannot be found, follow the instructions on the website to install Node on your computer.

With Node installed, create a new file in a temporary directory to test Node out. Copy the following content into the file and save it as `server.js`.

Example Node.js Program

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

This program (which was taken from the Node.js getting started guide) runs a web server, that you can interact with. You can now start the program from your terminal application, by changing to the directory of the `server.js` file and running it with the `node` command.

Running the server.js Program

```
$ node server.js
```

The output should be:

Output of the server.js program

```
Server running at http://127.0.0.1:3000/
```

The server is now running until you press `Ctrl` + `c` in your terminal application. Start another instance of your terminal application and use `curl` to test the server.

Testing the server

```
$ curl http://127.0.0.1:3000/
```

The response from the server should read `Hello World`.

Node is now working fine!

macOS

Check the website for the current versions.

Run the following command in your terminal application to make sure your version is up to date.

```
$ node -v
```

The output should show the version of Node. If the program cannot be found, follow the instructions on the website to install Node on your computer.

With Node installed, create a new file in a temporary directory to test Node out. Copy the following content into the file and save it as `server.js`.

Example Node.js Program

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
```

```
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

This program (which was taken from the Node.js getting started guide) runs a web server, that you can interact with. You can now start the program from your terminal application, by changing to the directory of the `server.js` file and running it with the `node` command.

Running the server.js Program

```
$ node server.js
```

The output should be:

Output of the server.js program

```
Server running at http://127.0.0.1:3000/
```

The server is now running until you press `Ctrl` + `c` in your terminal application. Start another instance of your terminal application and use `curl` to test the server.

Testing the server

```
$ curl http://127.0.0.1:3000/
```

The response from the server should read `Hello World`.

Node is now working fine!

Linux

Check the website for the current versions.

Run the following command in your terminal application to make sure your version is up to date.

```
$ node -v
```

The output should show the version of Node. If the program cannot be found, follow the instructions on the website to install Node on your computer.

With Node installed, create a new file in a temporary directory to test Node out. Copy the following content into the file and save it as `server.js`.

Example Node.js Program

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

This program (which was taken from the Node.js getting started guide) runs a web server, that you can interact with. You can now start the program from your terminal application, by changing to the directory of the `server.js` file and running it with the `node` command.

Running the server.js Program

```
  $ node server.js
```

The output should be:

Output of the server.js program

```
Server running at http://127.0.0.1:3000/
```

The server is now running until you press `Ctrl` + `C` in your terminal application. Start another instance of your terminal application and use `curl` to test the server.

Testing the server

```
  $ curl http://127.0.0.1:3000/
```

The response from the server should read `Hello World`.

Node is now working fine!

Docker

The Docker project offers tooling and a runtime for running containerised applications on a host system. We will use Docker to run programs that are prepared for us, without having to install the entire toolchain required to compile, package, install and run the programs.

Installing Docker and getting it to work can be difficult in some cases, but mostly following the general steps from the installation guide should be enough. Docker also provides GUI tooling for some operating systems, which allows you to inspect and modify the state of Docker itself and the containers running on your system.

Windows

If you have Docker on your computer, the following command in your terminal application should show you the version of Docker that is installed.

```
$ docker -v
```

If the program cannot be found on your computer, install it using the instructions from the website. You may also get an error that the Docker daemon isn't running. In that case, Docker is installed, but not ready to execute commands. Start the Docker Desktop app or on Linux, the Docker service and retry the command.

Docker can download (*pull*) container images to your computer and run them for you. You can also create a definition of a container image using a special file, `Dockerfile`, which is first built into an image using Docker, then run as a container. We'll use that second option later, but let's start by checking the first option, which uses a publicly available image.

Perform the following command in your terminal application.

```
$ docker run -d -p 80:80 docker/getting-started
```

This will pull the *getting-started* image from the public Docker registry and create a copy on your computer. Then this image will be started as a container and make a server available on port 80.

Test the server with `curl`.

```
$ curl -I http://localhost:80
```

The `-I` flag is used to just show the headers of the response instead of the body. The output will look something like this:

```
HTTP/1.1 200 OK
Server: nginx/1.21.6
Date: Mon, 11 Apr 2022 15:49:23 GMT
Content-Type: text/html
Content-Length: 8697
Last-Modified: Mon, 11 Apr 2022 15:23:07 GMT
Connection: keep-alive
ETag: "625447db-21f9"
Accept-Ranges: bytes
```

You can stop the running container by executing the following commands.

```
$ docker ps
```

This will show the running containers, one of which is based on the image `docker/getting-started`. Copy the value of that container that is in the `CONTAINER ID` field. In the example output below, that value is `212b964ec3f2`.

Example Output for `docker ps`

```
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS             PORTS                                NAMES
212b964ec3f2   docker/getting-started   "/docker-entrypoint.…"   About an hour ago
Up About an hour   0.0.0.0:80->80/tcp, :::80->80/tcp   inspiring_villani
```

Now you can use the container's ID to stop it.

Stopping the Container

```
$ docker stop 212b964ec3f2
```

Running another `docker ps` command should show the container is no longer running.

As a final step, you can pull another image to your computer with Docker. This image will be used during the training. Downloading it beforehand will save the download time during the training and help to prevent the network to slow down if all participants try it around the same time emoji:wink[].

```
$ docker pull camunda/camunda-bpm-platform:run-latest
```

macOS

If you have Docker on your computer, the following command in your terminal application should show you the version of Docker that is installed.

```
$ docker -v
```

If the program cannot be found on your computer, install it using the instructions from the website. You may also get an error that the Docker daemon isn't running. In that case, Docker is installed, but not ready to execute commands. Start the Docker Desktop app or on Linux, the Docker service and retry the command.

Docker can download (*pull*) container images to your computer and run them for you. You can also create a definition of a container image using a special file, `Dockerfile`, which is first built into an image using Docker, then run as a container. We'll use that second option later, but let's start by checking the first option, which uses a publicly available image.

Perform the following command in your terminal application.

```
$ docker run -d -p 80:80 docker/getting-started
```

This will pull the *getting-started* image from the public Docker registry and create a copy on your computer. Then this image will be started as a container and make a server available on port 80.

Test the server with `curl`.

```
$ curl -I http://localhost:80
```

The `-I` flag is used to just show the headers of the response instead of the body. The output will look something like this:

```
HTTP/1.1 200 OK
Server: nginx/1.21.6
Date: Mon, 11 Apr 2022 15:49:23 GMT
Content-Type: text/html
Content-Length: 8697
Last-Modified: Mon, 11 Apr 2022 15:23:07 GMT
Connection: keep-alive
ETag: "625447db-21f9"
Accept-Ranges: bytes
```

You can stop the running container by executing the following commands.

```
$ docker ps
```

This will show the running containers, one of which is based on the image `docker/getting-started`. Copy the value of that container that is in the `CONTAINER ID` field. In the example output below, that value is `212b964ec3f2`.

Example Output for `docker ps`

```
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS             PORTS                                  NAMES
212b964ec3f2   docker/getting-started   "/docker-entrypoint.…"   About an hour ago
Up About an hour   0.0.0.0:80->80/tcp, :::80->80/tcp   inspiring_villani
```

Now you can use the container's ID to stop it.

Stopping the Container

```
$ docker stop 212b964ec3f2
```

Running another `docker ps` command should show the container is no longer running.

As a final step, you can pull another image to your computer with Docker. This image will be used during the training. Downloading it beforehand will save the download time during the training and help to prevent the network to slow down if all participants try it around the same time emoji:wink[].

```
$ docker pull camunda/camunda-bpm-platform:run-latest
```

Linux

If you have Docker on your computer, the following command in your terminal application should show you the version of Docker that is installed.

```
$ docker -v
```

If the program cannot be found on your computer, install it using the instructions from the website. You may also get an error that the Docker daemon isn't running. In that case, Docker is installed, but not ready to execute commands. Start the Docker Desktop app or on Linux, the Docker service and retry the command.

Docker can download (*pull*) container images to your computer and run them for you. You can also create a definition of a container image using a special file, `Dockerfile`, which is first built into an image using Docker, then run as a container. We'll use that second option later, but let's start by checking the first option, which uses a publicly available image.

Perform the following command in your terminal application.

```
$ docker run -d -p 80:80 docker/getting-started
```

This will pull the *getting-started* image from the public Docker registry and create a copy on your computer. Then this image will be started as a container and make a server available on port 80.

Test the server with `curl`.

```
$ curl -I http://localhost:80
```

The `-I` flag is used to just show the headers of the response instead of the body. The output will look something like this:

```
HTTP/1.1 200 OK
Server: nginx/1.21.6
Date: Mon, 11 Apr 2022 15:49:23 GMT
Content-Type: text/html
Content-Length: 8697
Last-Modified: Mon, 11 Apr 2022 15:23:07 GMT
Connection: keep-alive
ETag: "625447db-21f9"
Accept-Ranges: bytes
```

You can stop the running container by executing the following commands.

```
$ docker ps
```

This will show the running containers, one of which is based on the image `docker/getting-started`. Copy the value of that container that is in the `CONTAINER ID` field. In the example output below, that value is `212b964ec3f2`.

Example Output for `docker ps`

```
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS             PORTS                                NAMES
212b964ec3f2   docker/getting-started   "/docker-entrypoint.…"   About an hour ago
Up About an hour   0.0.0.0:80->80/tcp, :::80->80/tcp   inspiring_villani
```

Now you can use the container's ID to stop it.

Stopping the Container

```
$ docker stop 212b964ec3f2
```

Running another `docker ps` command should show the container is no longer running.

As a final step, you can pull another image to your computer with Docker. This image will be used during the training. Downloading it beforehand will save the download time during the training and help to prevent the network to slow down if all participants try it around the same time emoji:wink[].

```
$ docker pull camunda/camunda-bpm-platform:run-latest
```

Camunda Modeler

The graphical tool *Camunda Modeler* is a freely available tool to create graphical models for business processes and decisions. The technical configuration to execute the models with software can also be added with Camunda Modeler. It adds a number of vendor specific extensions and features to the base modelling capabilities of the open source bpmn.io modeller project, from the same vendor, Camunda.

This training may not be about Camunda specifically, but we will use the *Camunda Modeler* during the assignments.

Windows

Install Camunda Modeler by downloading it from the website.

Although Modeler can be installed as an application native to your operating system, simply extracting the application to a folder somewhere is enough to be able to run and use it.

In the extracted folder or your applications, there should be a *Camunda Modeler* application. Run or double-click it.

The application should start up. That's enough for now, we'll look at Camunda Modeler during the training's assignments.

macOS

Install Camunda Modeler by downloading it from the website.

Although Modeler can be installed as an application native to your operating system, simply extracting the application to a folder somewhere is enough to be able to run and use it.

In the extracted folder or your applications, there should be a *Camunda Modeler* application. Run or double-click it.

The application should start up. That's enough for now, we'll look at Camunda Modeler during the training's assignments.

Linux

Install Camunda Modeler by downloading it from the website.

Although Modeler can be installed as an application native to your operating system, simply extracting the application to a folder somewhere is enough to be able to run and use it.

In the extracted folder or your applications, there should be a *Camunda Modeler* application. Run or double-click it.

The application should start up. That's enough for now, we'll look at Camunda Modeler during the training's assignments.