

# MANDELROT APPS

## SYSTEM ADMIN GUIDE

### Contents

- Intro
- Admin features
- Compiling your own MApps version, and “installing”
- First run
- Suite administration
- ID control app (new in v2)
- Backups and migrations

### Intro

This document is for system admins who already know what's the MApps backend about (if not please go to see the “overview guide” available) and want to use it in their working environments. If this is the case, in a few minutes you will understand the basics and know all you need to get to work. This document seems long but it's not, there are many screencaptures and images because it's meant to be a “guide for dummies”. Follow along and you will see it by yourself. All the updated info, including the guides and some basic example apps to see how the system works, in the project GitHub repo:

[https://github.com/mandelrot/mapps\\_backend](https://github.com/mandelrot/mapps_backend)

This software has been made by Jose Alemán / Mandelrot. You can download the code and use it freely, including changing whatever you want. In case you want to know more or contact me:

My blog: <https://mandelrot.com> (spanish)

My professional webpage: <http://josealeman.info> (english)

## **Admin features**

The Mandelrot Apps software, once compiled (all about this later) and having the custom production package, will give you a 100% portable apps manager engine. This means you will just have to copy the package folder to the server, run the main executable, and that will be all. The webserver is already built-in, the databases are managed by the front app (no need to install anything)... As easy as it sounds.

You will be able to install-uninstall frontend apps just by copying a folder and pressing a button (without even restarting the system) and enable or disable apps just with two clicks.

An important concept to have in mind here is that the admin only takes care of the frontend apps “installation” and enabling. This is a pure abstract tool, meaning the backend’s job is only detecting the present frontend apps, allowing them being a part of the team, and routing the communications between them.

So this engine knows nothing about users, groups and so on. Each company/organization should create their own front apps and maybe (if they consider it necessary for their business logic) have an users manager module so the other apps can ask it for validations and so on. This backend will grant the communications between them and all the needed scaffold, how they use it is a developers’ task.

## **Compiling your own MApps version**

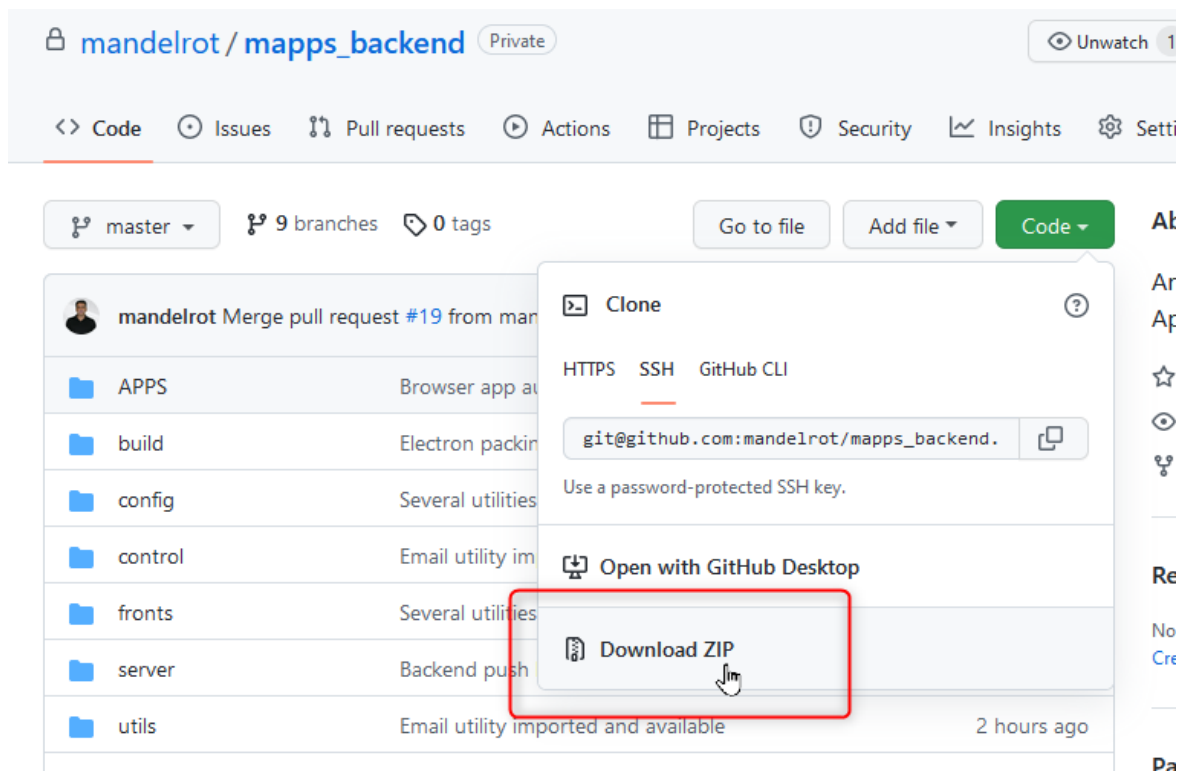
The good news: the same code generates you the final production package in Linux, Windows and Mac. The bad news: you will need to install some things to do this package generation. But I’ve got you covered, just follow the next steps and it won’t be difficult.

**Step 1:** use the same pc, server or environment where you will run the production version of the software. Being in Windows you can in theory generate a Linux version for example, but trust me on this and make it simple. The package generator will recognize your system and give you exactly the compiled version you need for it.

**Step 2:** go to the GitHub project page:

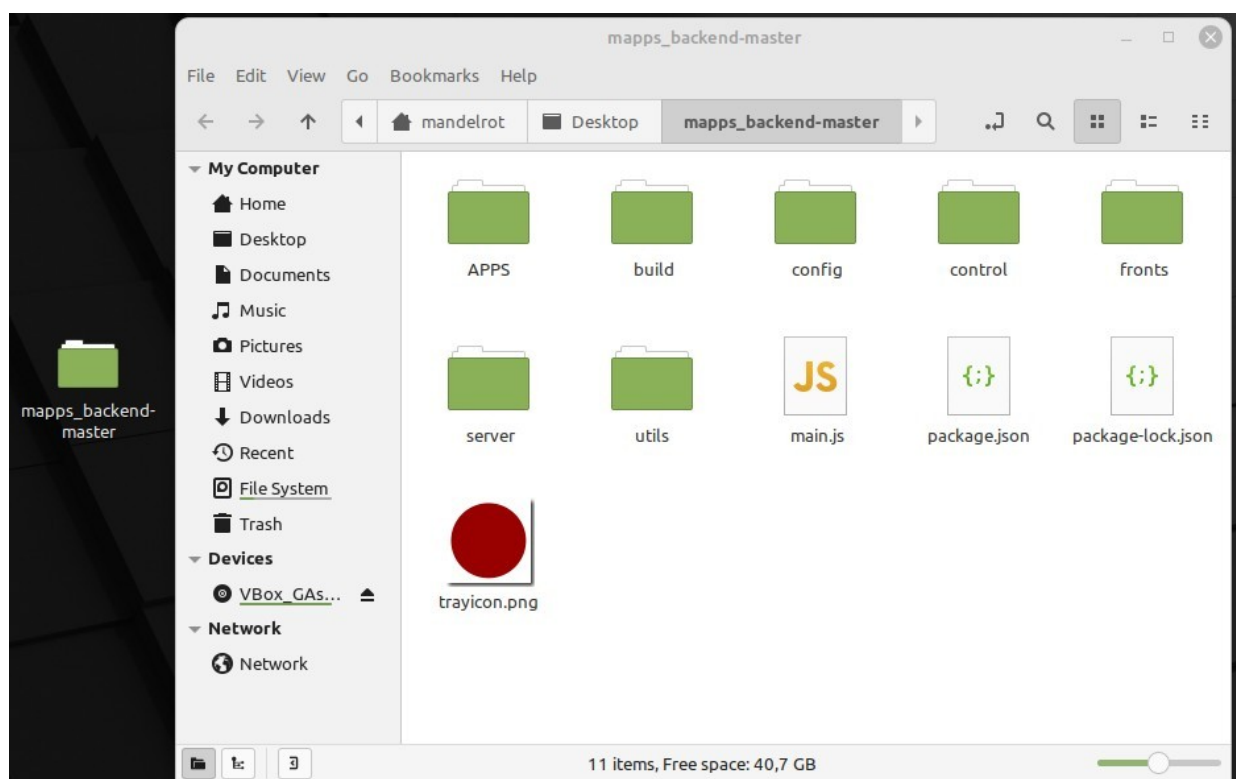
[https://github.com/mandelrot/mapps\\_backend](https://github.com/mandelrot/mapps_backend)

And once there download the code:



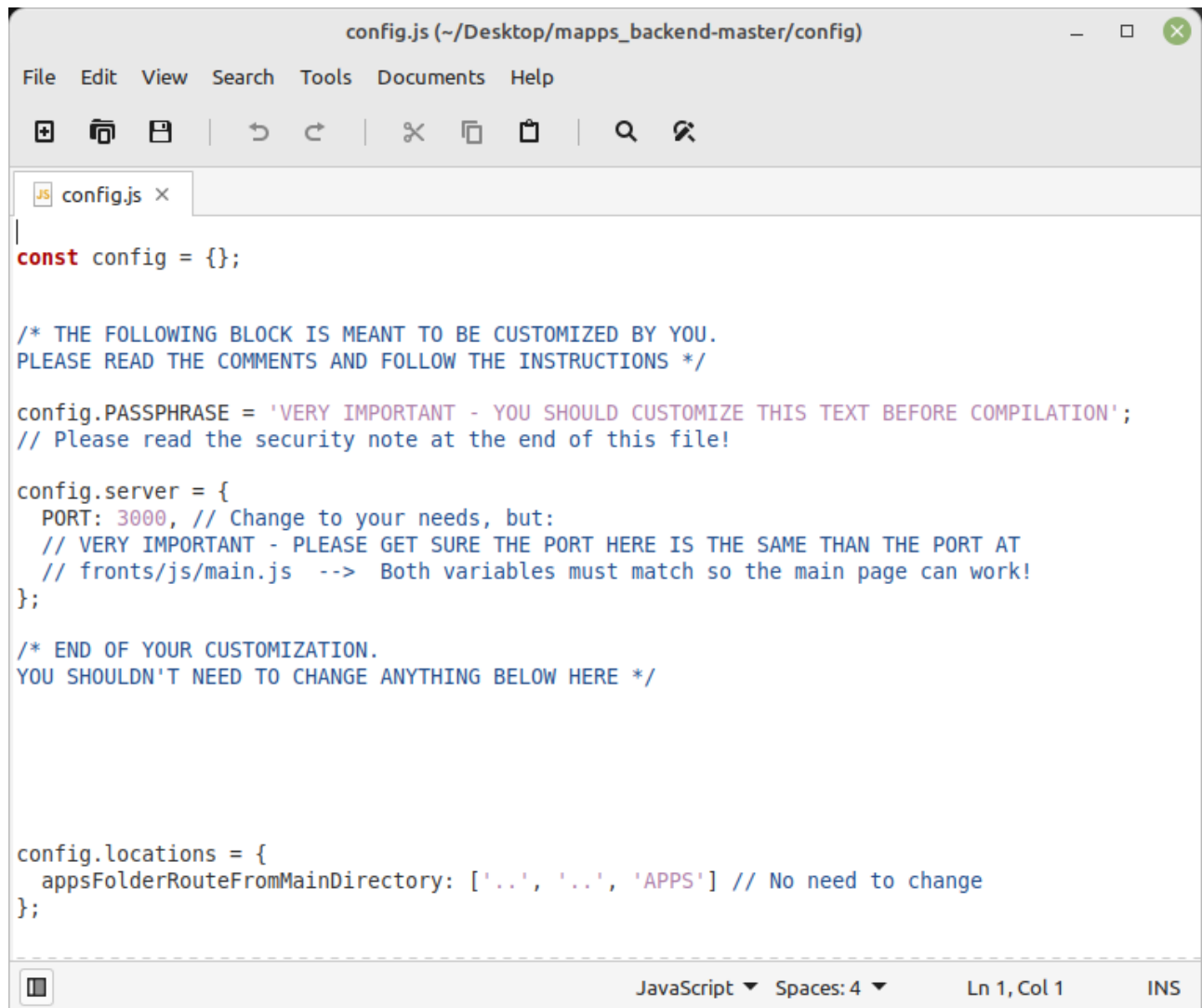
(From here we will continue mostly in a fresh Linux Mint desktop for demo purposes, but it's basically the same everywhere).

Once downloaded and zip-extracted you will end up with something like this:



**Step 3:** changing files. We will start with the **APPS** folder: you can safely delete it, this is not a part of the suite itself but some example frontend apps I made for the developers. For this tutorial I will keep it though, because I'll use it to show you how to manage the frontend apps once the engine is installed and running. So I will move the APPS folder to my desktop, you can do the same if you want so you can follow the demonstration along.

Now, you must DELETE the file **config/config\_dev.js** (very important, or the software won't work) and edit the **config/config.js**. There you will find this:



```
config.js (~/Desktop/mapps_backend-master/config)
File Edit View Search Tools Documents Help
JS config.js x
const config = {};

/* THE FOLLOWING BLOCK IS MEANT TO BE CUSTOMIZED BY YOU.
PLEASE READ THE COMMENTS AND FOLLOW THE INSTRUCTIONS */

config.PASSPHRASE = 'VERY IMPORTANT - YOU SHOULD CUSTOMIZE THIS TEXT BEFORE COMPILATION';
// Please read the security note at the end of this file!

config.server = {
  PORT: 3000, // Change to your needs, but:
  // VERY IMPORTANT - PLEASE GET SURE THE PORT HERE IS THE SAME THAN THE PORT AT
  // fronts/js/main.js --> Both variables must match so the main page can work!
};

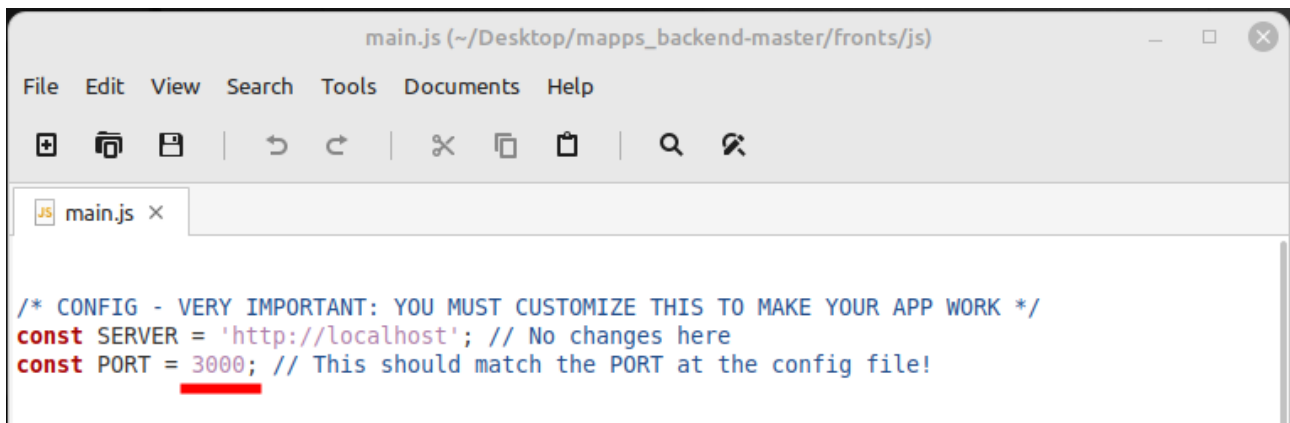
/* END OF YOUR CUSTOMIZATION.
YOU SHOULDN'T NEED TO CHANGE ANYTHING BELOW HERE */

config.locations = {
  appsFolderRouteFromMainDirectory: ['..', '..', 'APPS'] // No need to change
};

JavaScript Spaces: 4 Ln 1, Col 1 INS
```

The two things you need to change here are the PASSPHRASE and the server PORT. Please take a minute to read the final security note (it's more for developers, but good you to know it just in case).

As it says there, if you change the port you need to update it in another file too (it's not automatic for development reasons). Go to **fronts/js/main.js** and change the port there too, and that will be it.



```
main.js (~/Desktop/mapps_backend-master/fronts/js)
File Edit View Search Tools Documents Help
+ [Icons] | ↶ ↷ | ✂ [Icons] | 🔍 [Icon]
main.js x
/* CONFIG - VERY IMPORTANT: YOU MUST CUSTOMIZE THIS TO MAKE YOUR APP WORK */
const SERVER = 'http://localhost'; // No changes here
const PORT = 3000; // This should match the PORT at the config file!
```

**Step 4:** installing Node and NPM. Node and NPM may have different ways of being installed depending on your exact system, so here is the Node page and you won't have much trouble with it:

<https://nodejs.org>

In Windows you can download any available version you see in the main page. Any of those should get the job done and we will uninstall it after the compilation, so pick whatever you want.

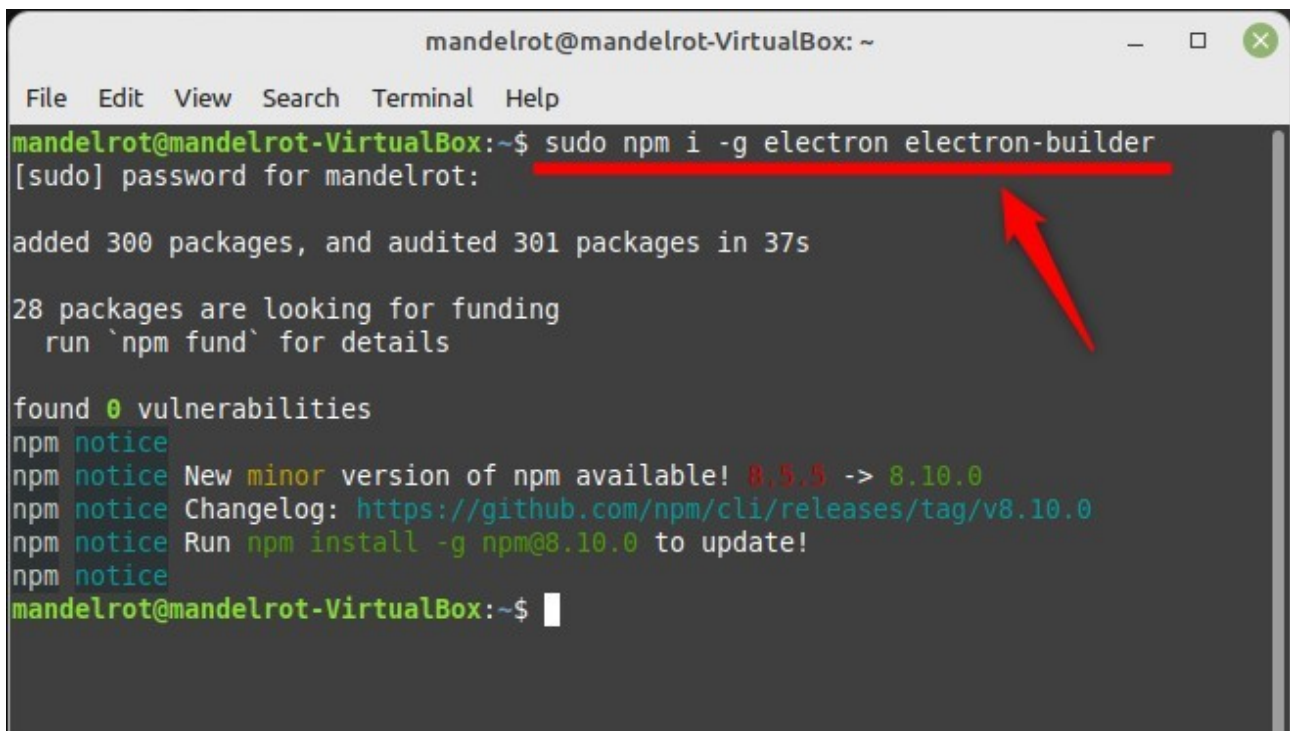
Anyway, the downloads page gives you several different options:

<https://nodejs.org/en/download>

Note for Linux: maybe when you are doing this the main page gives you an easier option, but when I have done it myself I've found the package manager options quite straightforward:

<https://nodejs.org/en/download/package-manager>

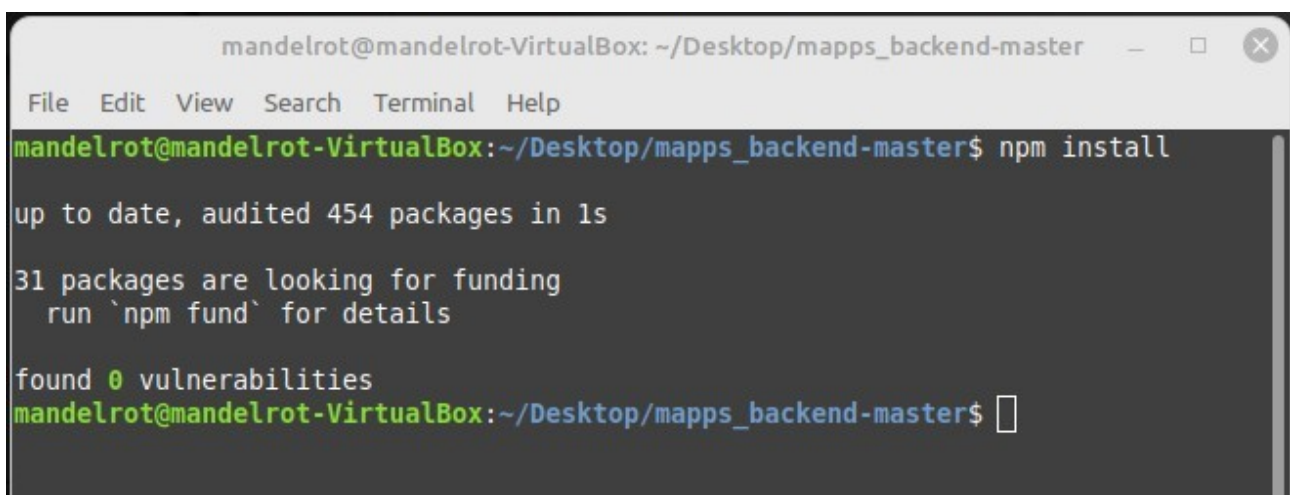
**Step 5:** installing **Electron / Electron builder** globally. NPM makes it easy, just do **npm i -g electron electron-builder** [enter] (if you are in Linux you may have to add the “sudo” to avoid writing permissions conflicts):



```
mandelrot@mandelrot-VirtualBox: ~  
File Edit View Search Terminal Help  
mandelrot@mandelrot-VirtualBox:~$ sudo npm i -g electron electron-builder  
[sudo] password for mandelrot:  
  
added 300 packages, and audited 301 packages in 37s  
  
28 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
npm notice  
npm notice New minor version of npm available! 8.5.5 -> 8.10.0  
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.10.0  
npm notice Run npm install -g npm@8.10.0 to update!  
npm notice  
mandelrot@mandelrot-VirtualBox:~$
```

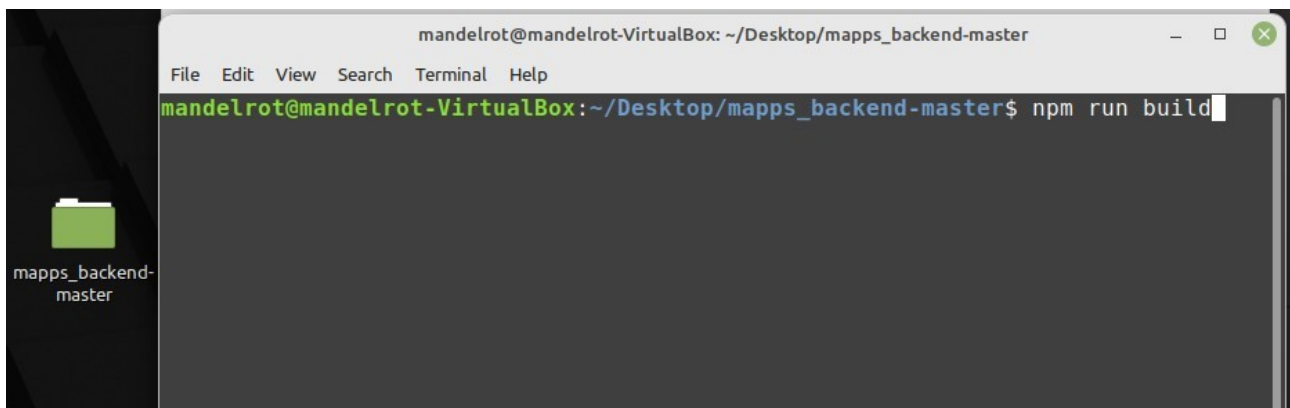
A red arrow points to the command line in the terminal window.

**Step 6:** (important: being in the downloaded root folder) you do **npm install** to import some final elements:



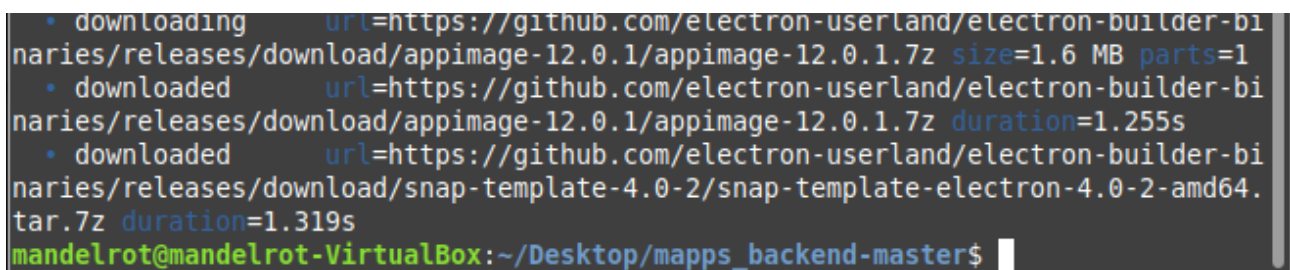
```
mandelrot@mandelrot-VirtualBox: ~/Desktop/maps_backend-master  
File Edit View Search Terminal Help  
mandelrot@mandelrot-VirtualBox:~/Desktop/maps_backend-master$ npm install  
  
up to date, audited 454 packages in 1s  
  
31 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
mandelrot@mandelrot-VirtualBox:~/Desktop/maps_backend-master$
```

**Step 7 - final:** once all the prerequisites are correctly installed, you will be able to do open a terminal (still in the downloaded root folder) and directly start the compilation process with `npm run build` [enter]:



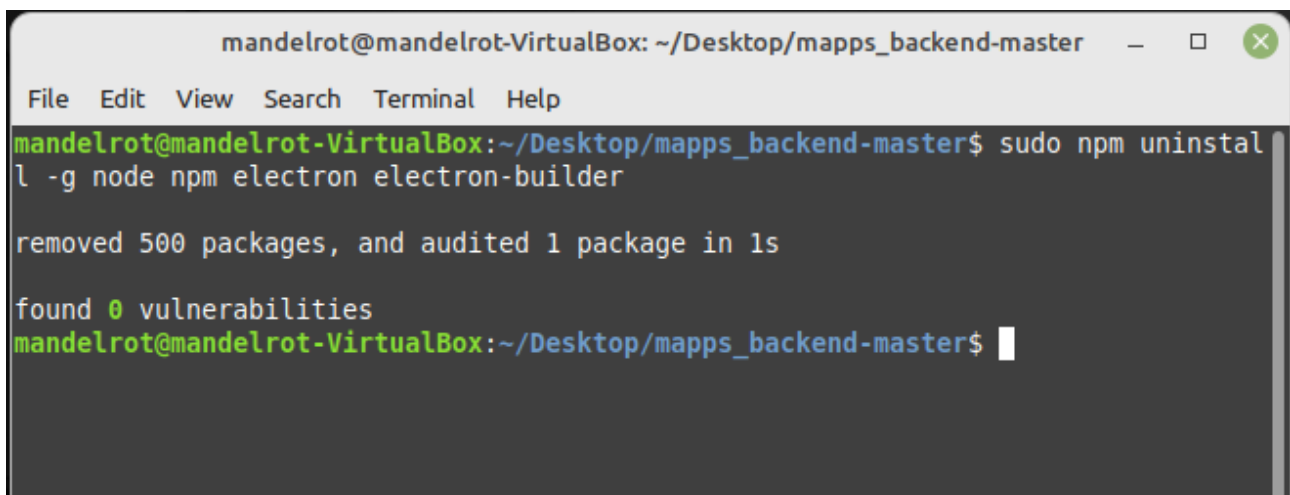
```
mandelrot@mandelrot-VirtualBox: ~/Desktop/mapps_backend-master
File Edit View Search Terminal Help
mandelrot@mandelrot-VirtualBox:~/Desktop/mapps_backend-master$ npm run build
```

The system will work for a while, just wait until it finishes.



```
• downloading    url=https://github.com/electron-userland/electron-builder-binaries/releases/download/appimage-12.0.1/appimage-12.0.1.7z size=1.6 MB parts=1
• downloaded     url=https://github.com/electron-userland/electron-builder-binaries/releases/download/appimage-12.0.1/appimage-12.0.1.7z duration=1.255s
• downloaded     url=https://github.com/electron-userland/electron-builder-binaries/releases/download/snap-template-4.0-2/snap-template-electron-4.0-2-amd64.tar.7z duration=1.319s
mandelrot@mandelrot-VirtualBox:~/Desktop/mapps_backend-master$
```

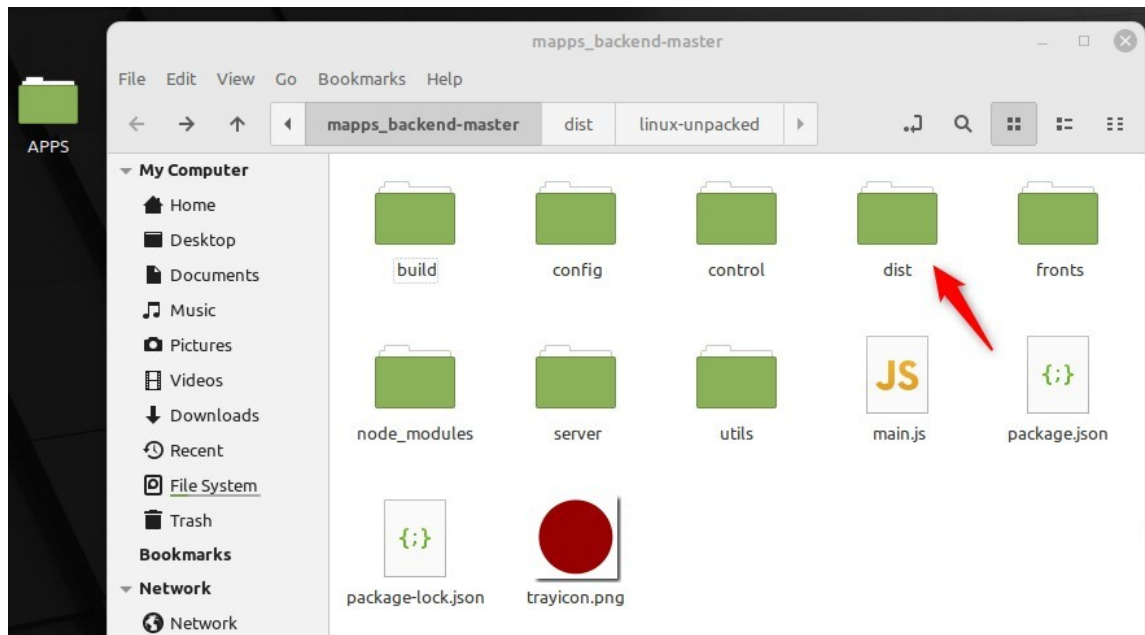
**Step 8 - optional:** you don't need Node, NPM, Electron or Electron Builder anymore; you may uninstall everything if you want. Use `npm uninstall -g node npm electron electron-builder` (with sudo if you are in Linux).



```
mandelrot@mandelrot-VirtualBox: ~/Desktop/mapps_backend-master
File Edit View Search Terminal Help
mandelrot@mandelrot-VirtualBox:~/Desktop/mapps_backend-master$ sudo npm uninstall -g node npm electron electron-builder
removed 500 packages, and audited 1 package in 1s
found 0 vulnerabilities
mandelrot@mandelrot-VirtualBox:~/Desktop/mapps_backend-master$
```

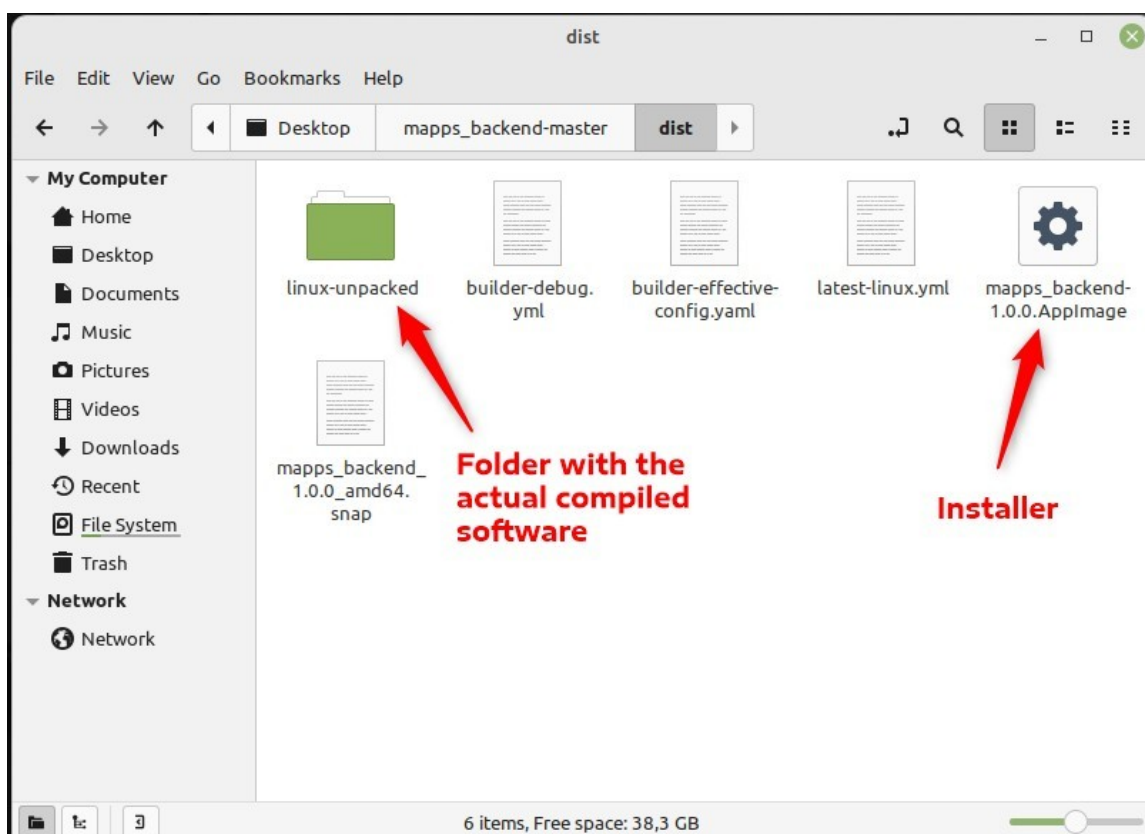


The compilation is complete. Now you have a production package, in a new generated **dist** folder.



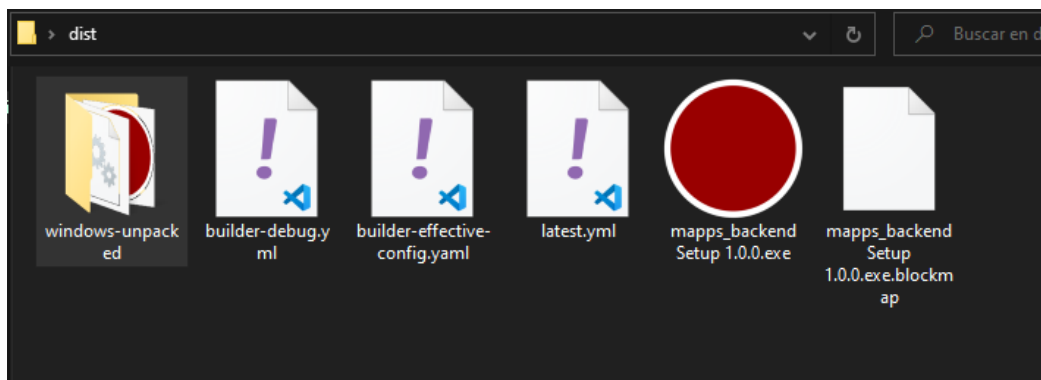
Regardless the system you use, in the “dist” folder you will always find two things: the actual software folder, and the installer.

Linux:





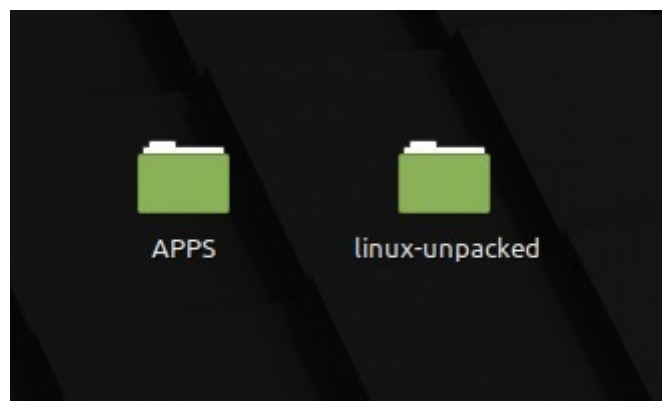
Windows:



The “unpacked” folder is what you need (the portable version). Why not installing it? You could, but since you will work with frontend apps (and they need to be in a relative path so the backend can find them) next you will see that the portable option makes everything straightforward and much easier.

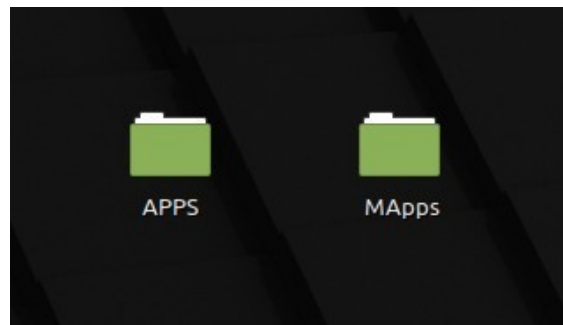
## First run

Now we are inside the portable software folder. In my case, just to get rid of things we will not need anymore, I will copy the folder to the desktop and delete everything else. This is what we have now in Linux, in Windows it would be the same (with the “windows-unpacked” folder):



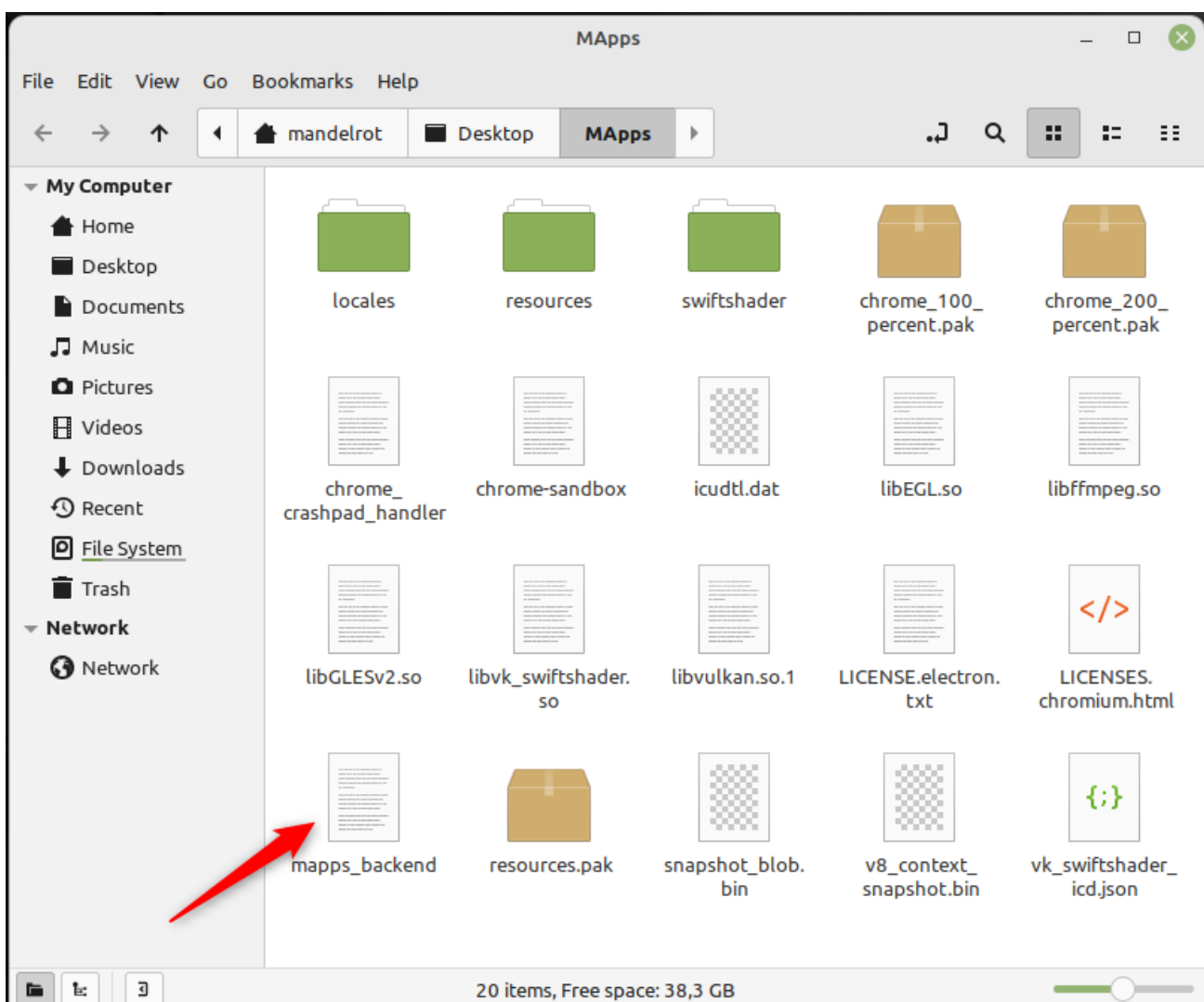
Remember we don't really need the “APPS” folder, it's just for demo purposes but in real life you could delete it as well.

You don't need to do this if you don't want to, but I will rename the main folder name so it makes more sense in production to work with it in the future knowing what you have there.

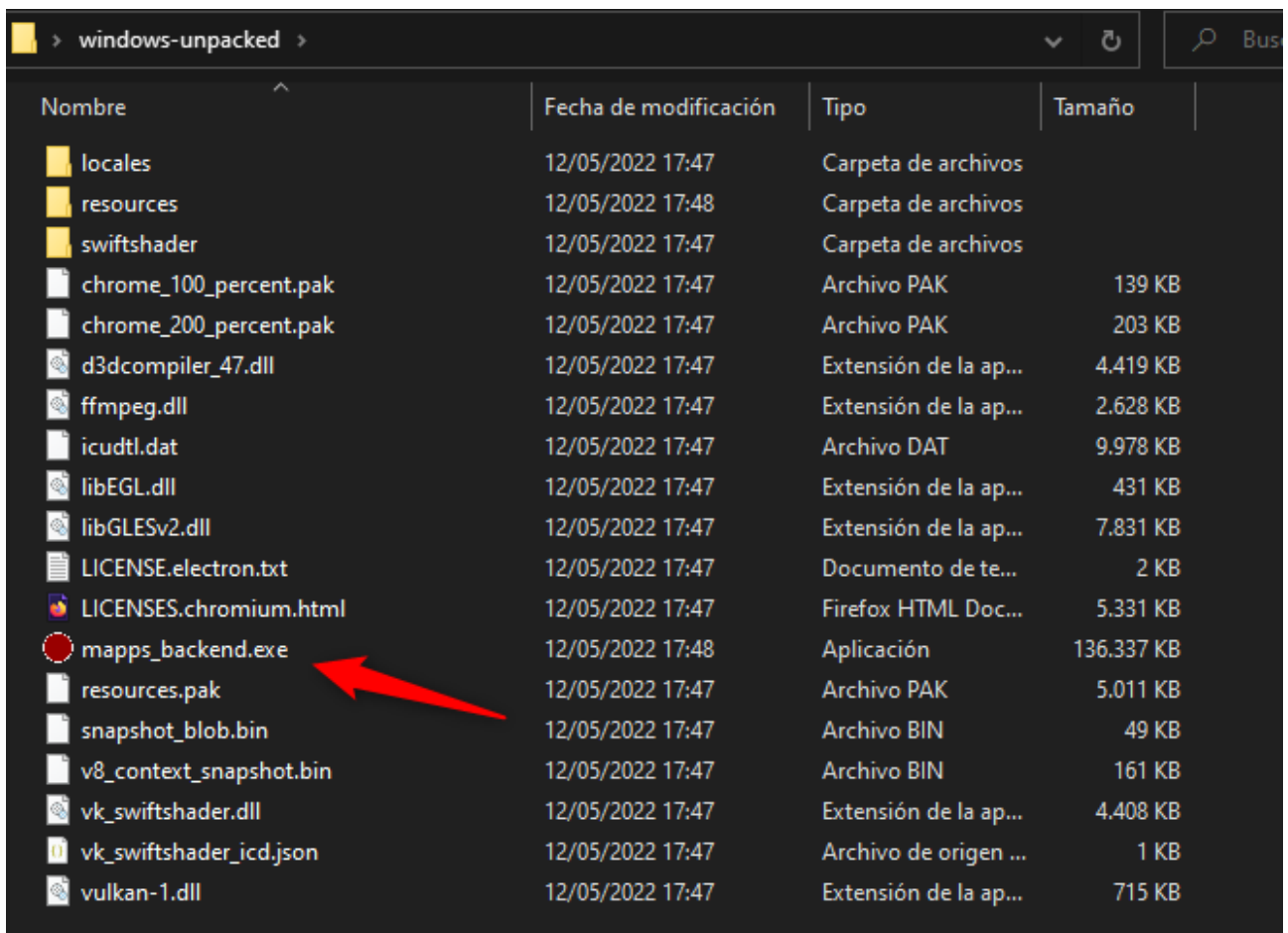


So we have the actual **MApps** main folder and, just for this example, the other one ("APPS") that contains some demo frontend apps.

Inside the main folder we will find the executable file to run the engine. Linux:



Windows:



The screenshot shows a Windows File Explorer window with the address bar set to 'windows-unpacked'. The window displays a list of files and folders. A red arrow points to the file 'mapps\_backend.exe'.

Nombre	Fecha de modificación	Tipo	Tamaño
locales	12/05/2022 17:47	Carpeta de archivos	
resources	12/05/2022 17:48	Carpeta de archivos	
swiftshader	12/05/2022 17:47	Carpeta de archivos	
chrome_100_percent.pak	12/05/2022 17:47	Archivo PAK	139 KB
chrome_200_percent.pak	12/05/2022 17:47	Archivo PAK	203 KB
d3dcompiler_47.dll	12/05/2022 17:47	Extensión de la ap...	4.419 KB
ffmpeg.dll	12/05/2022 17:47	Extensión de la ap...	2.628 KB
icudtl.dat	12/05/2022 17:47	Archivo DAT	9.978 KB
libEGL.dll	12/05/2022 17:47	Extensión de la ap...	431 KB
libGLSv2.dll	12/05/2022 17:47	Extensión de la ap...	7.831 KB
LICENSE.electron.txt	12/05/2022 17:47	Documento de te...	2 KB
LICENSES.chromium.html	12/05/2022 17:47	Firefox HTML Doc...	5.331 KB
mapps_backend.exe	12/05/2022 17:48	Aplicación	136.337 KB
resources.pak	12/05/2022 17:47	Archivo PAK	5.011 KB
snapshot_blob.bin	12/05/2022 17:47	Archivo BIN	49 KB
v8_context_snapshot.bin	12/05/2022 17:47	Archivo BIN	161 KB
vk_swiftshader.dll	12/05/2022 17:47	Extensión de la ap...	4.408 KB
vk_swiftshader_icd.json	12/05/2022 17:47	Archivo de origen ...	1 KB
vulkan-1.dll	12/05/2022 17:47	Extensión de la ap...	715 KB

In the moment you run it the software will work straight away.

Please be aware of this: when you run the engine a new APPS folder will be generated here (containing no frontend app yet). The system checks whether the folder exists and, if it's not there, then generates it. More on this in a moment.

So run the executable file, and you will see a window (maximized by default) containing this:

# MANDELROT APPS: ADMIN

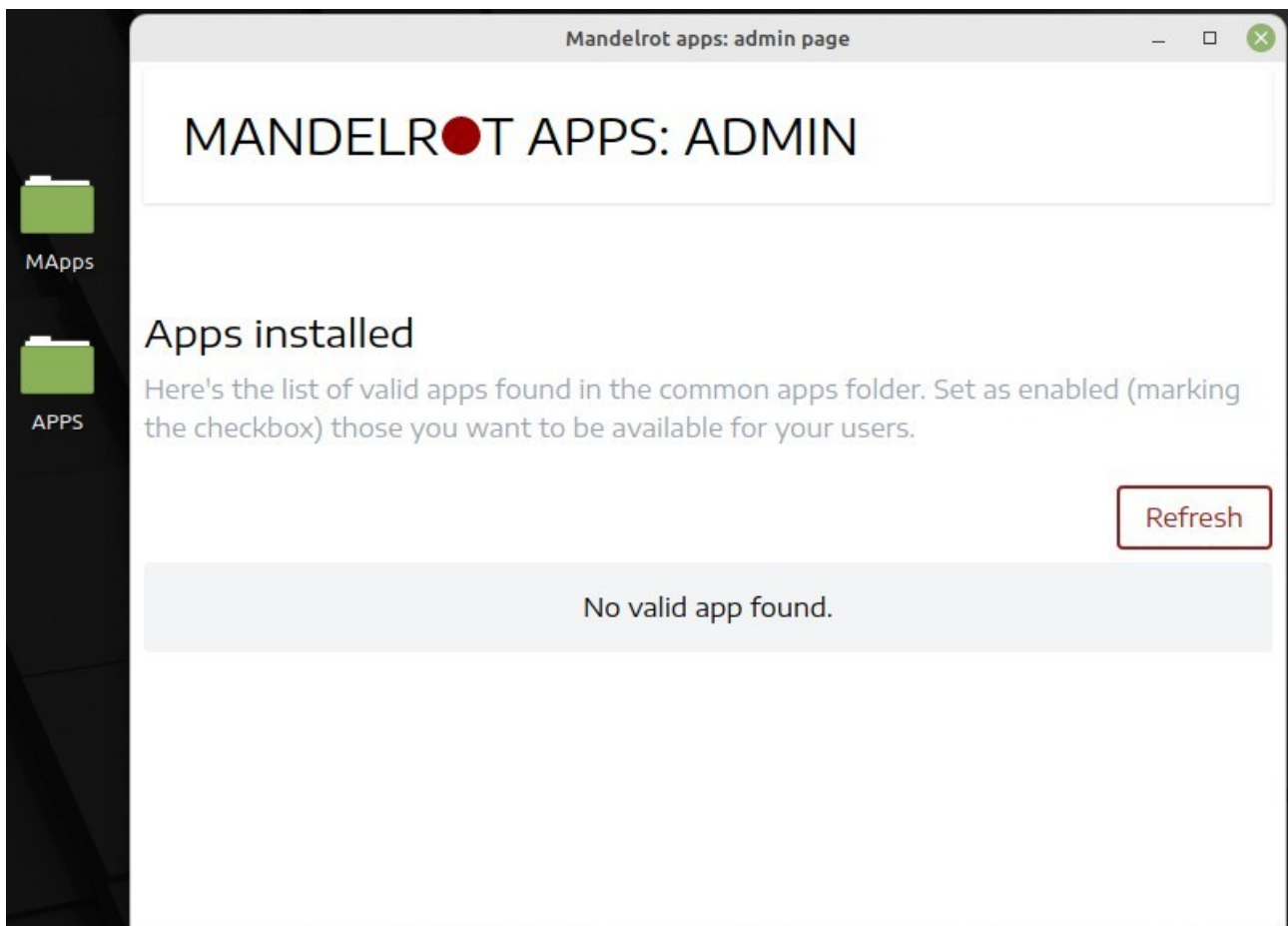
## Apps installed

Here's the list of valid apps found in the common apps folder. Set as enabled (marking the checkbox) those you want to be available for your users.

Refresh

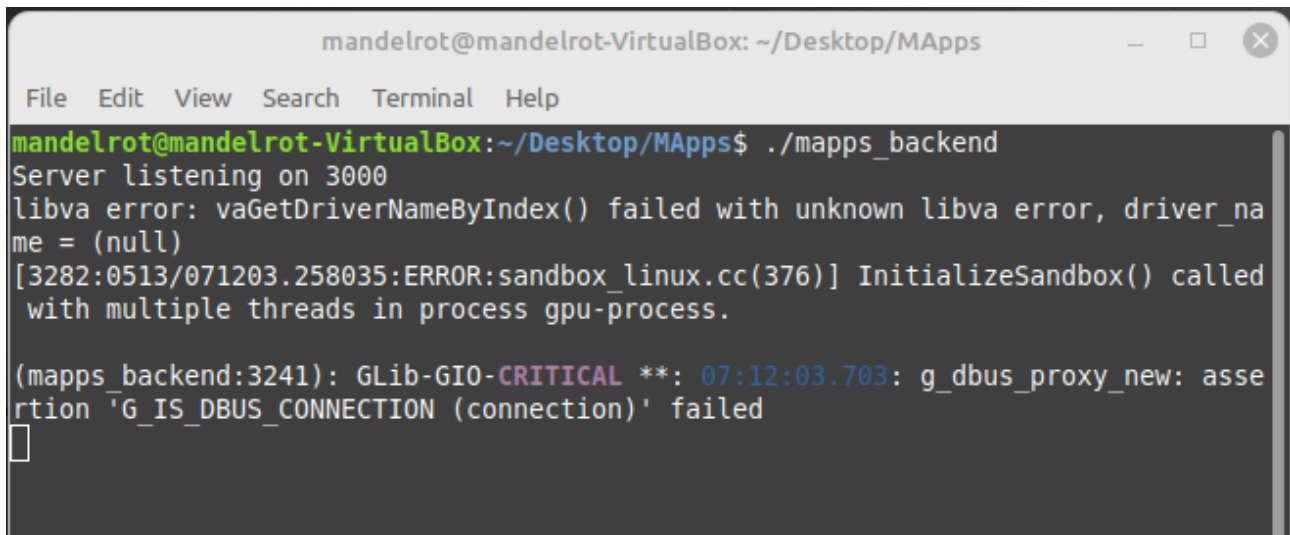
No valid app found.

Double-click on the title bar to un-maximize it, and we have our next workspace.



There is no valid app found because the new generated APPS folder (inside the main MApps folder) has no valid app yet. But you have successfully run the engine and everything is ready and working.

Note for Linux: in some distros, if you run the software using the terminal you may see some log errors like these:

A screenshot of a terminal window titled 'mandelrot@mandelrot-VirtualBox: ~/Desktop/MApps'. The terminal shows the command './mapps\_backend' being executed. The output includes 'Server listening on 3000', a 'libva error: vaGetDriverNameByIndex() failed with unknown libva error, driver\_name = (null)', and a message from 'sandbox\_linux.cc(376)' about 'InitializeSandbox() called with multiple threads in process gpu-process.'. A critical Glib-GIO error is also shown: '(mapps\_backend:3241): GLib-GIO-CRITICAL \*\*: 07:12:03.703: g\_dbus\_proxy\_new: assertion 'G\_IS\_DBUS\_CONNECTION (connection)' failed'.

```
mandelrot@mandelrot-VirtualBox: ~/Desktop/MApps
File Edit View Search Terminal Help
mandelrot@mandelrot-VirtualBox:~/Desktop/MApps$ ./mapps_backend
Server listening on 3000
libva error: vaGetDriverNameByIndex() failed with unknown libva error, driver_name = (null)
[3282:0513/071203.258035:ERROR:sandbox_linux.cc(376)] InitializeSandbox() called with multiple threads in process gpu-process.

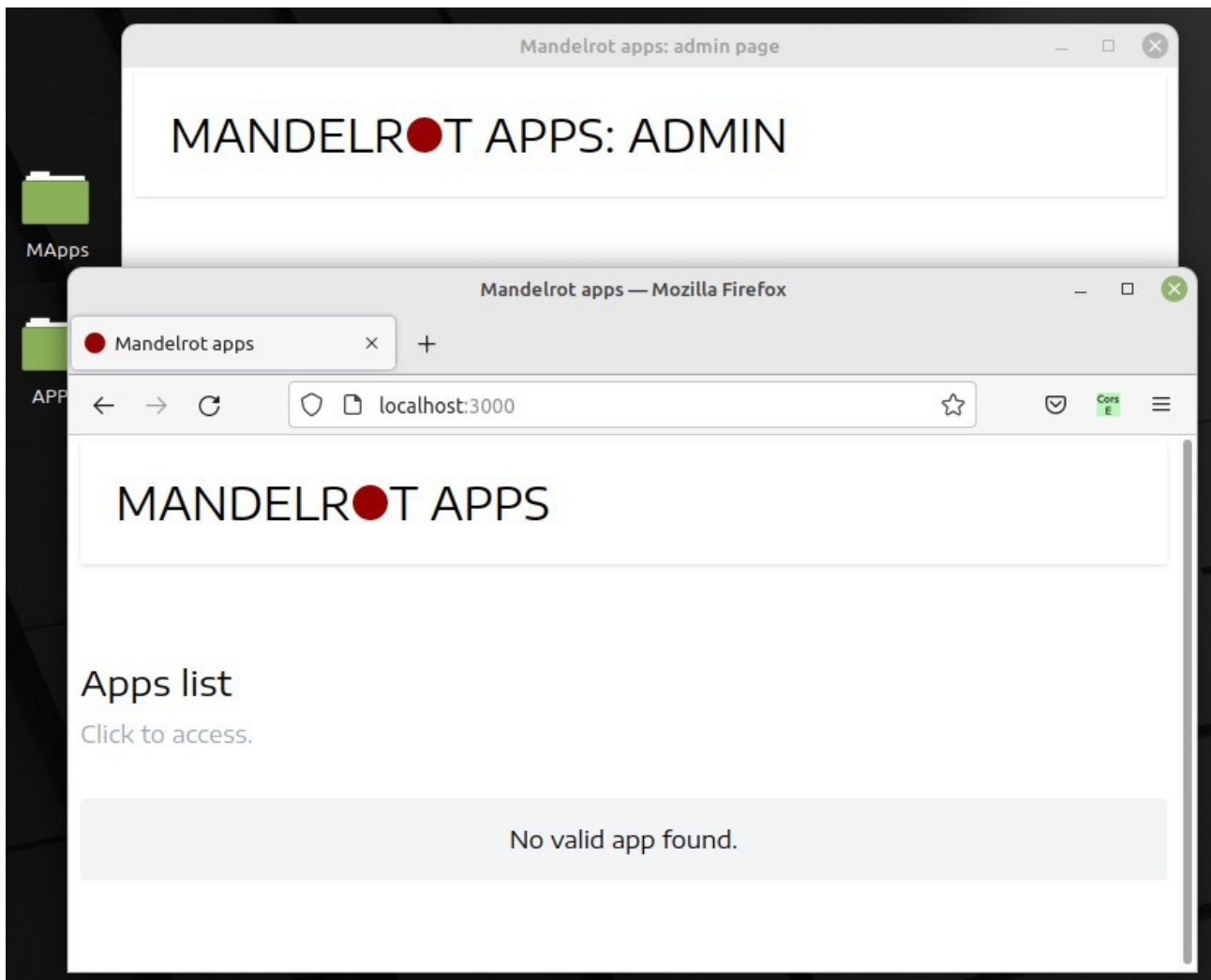
(mapps_backend:3241): GLib-GIO-CRITICAL **: 07:12:03.703: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
```

You can ignore them. It shouldn't affect to anything, and it seems to be an SO weird behavior (if you are interested [here](#) you have something about it) but with no consequence besides these log messages. Maybe other distros I haven't tried will show other errors (this is why many developers don't like systems), but the software should work just fine.

## Suite administration

Everything is ready to install new frontend apps. In the real production environment you are supposed to get the apps from your developers, in this case we will use some of the example apps I made.

Quick note: this engine has not only opened your admin window, but also the webserver your users will connect to. We'll enter into this in a moment, but just to demonstrate how things are working here is what we have:

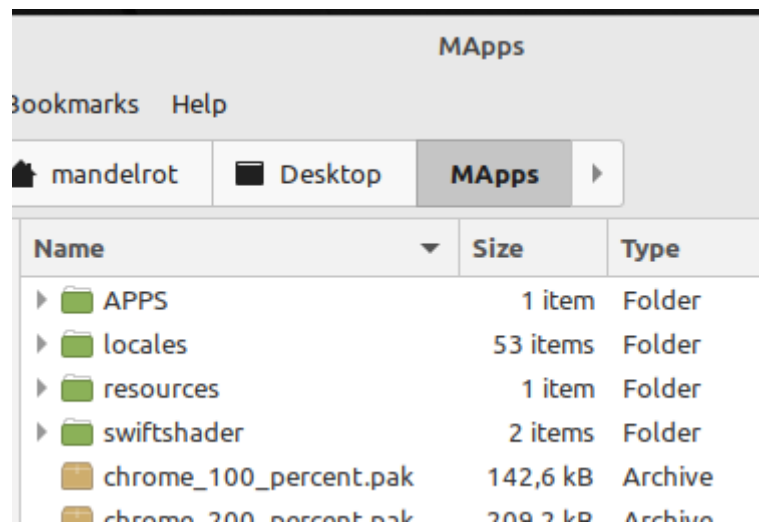


(The url port is, of course, the port you set in the `config.js` file)

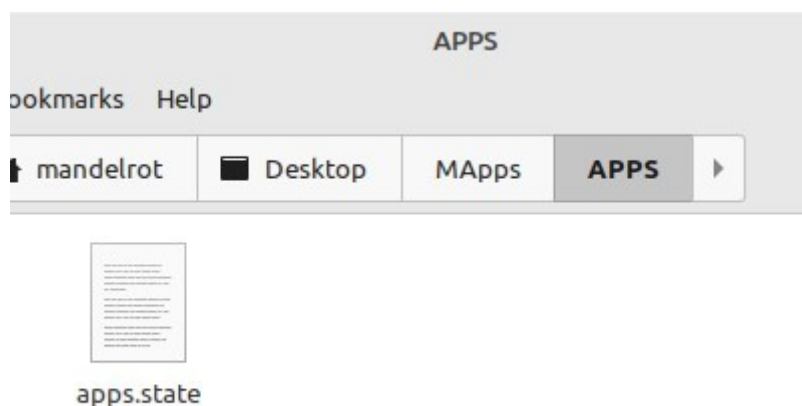
This Firefox window you see here is the users main page, where they will access the frontend apps. But let's ignore this for now, I just wanted to point out that you do not need to install anything or do anything else to have everything running. You have your main folder with an executable file to run, and that's it.

So let's close the users browser window and get back to install new frontend apps.

So now we see that, inside the engine main folder (MApps), a new APPS folder has been created.



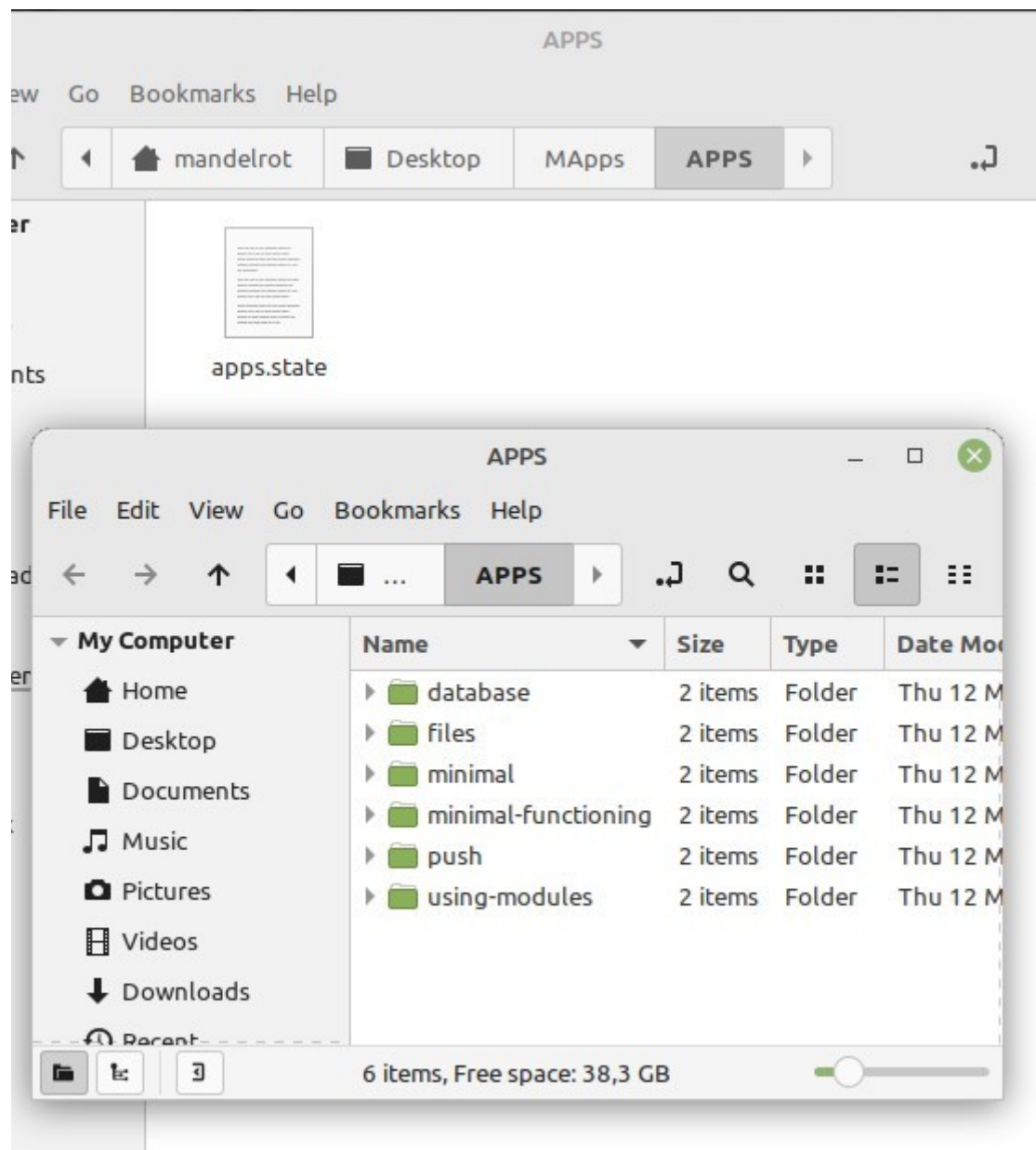
Inside it there's only a file, no frontend app yet. This file is an internal registry file and you should ignore it; anyway, in case you delete it by accident, it will be self-generated again when needed.



Inside the frontend apps folder we moved to the desktop before, we have some example apps we will use for this demo. Your developer should give you a folder like one of these ones, everything needed should be contained in it.

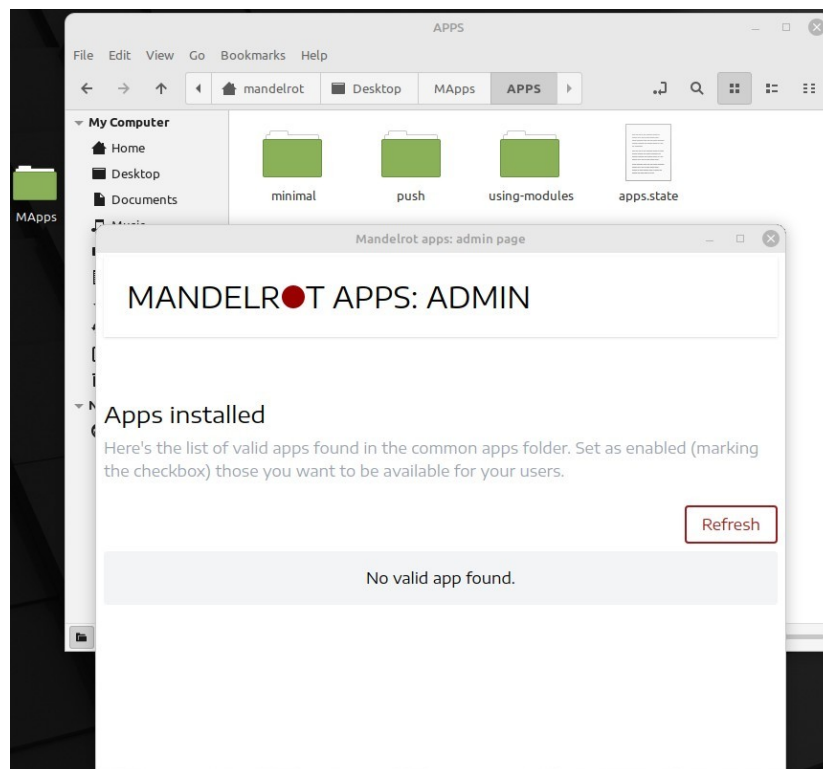
Please note: at this point the engine is running. You don't need to restart the server or do anything when installing-uninstalling frontend apps.



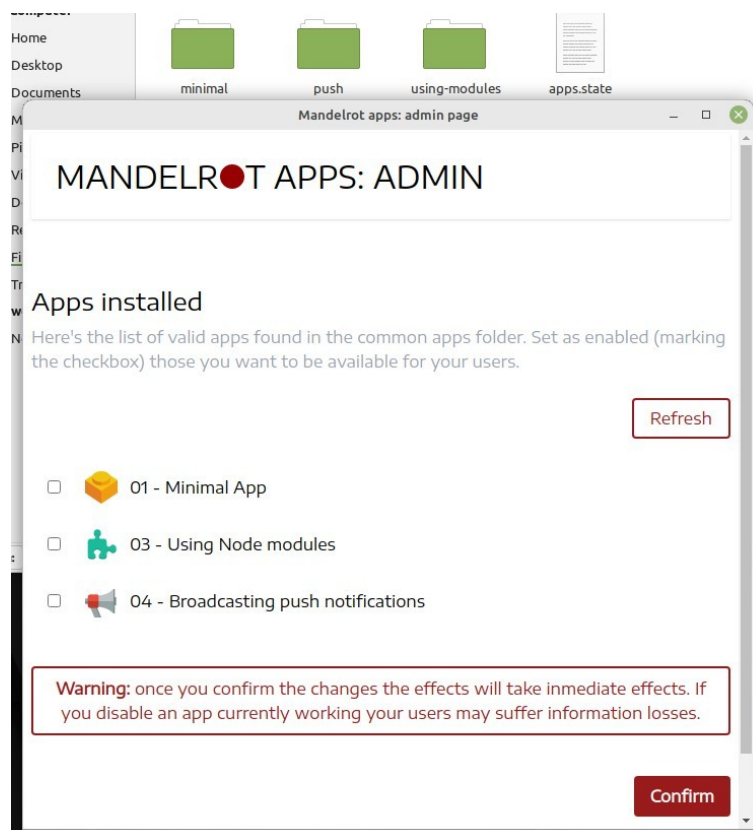


For this example I will copy three of these apps in the suite: “minimal”, “push” and “using-modules”. Once done, I will delete the APPS folder since we don’t need it anymore.

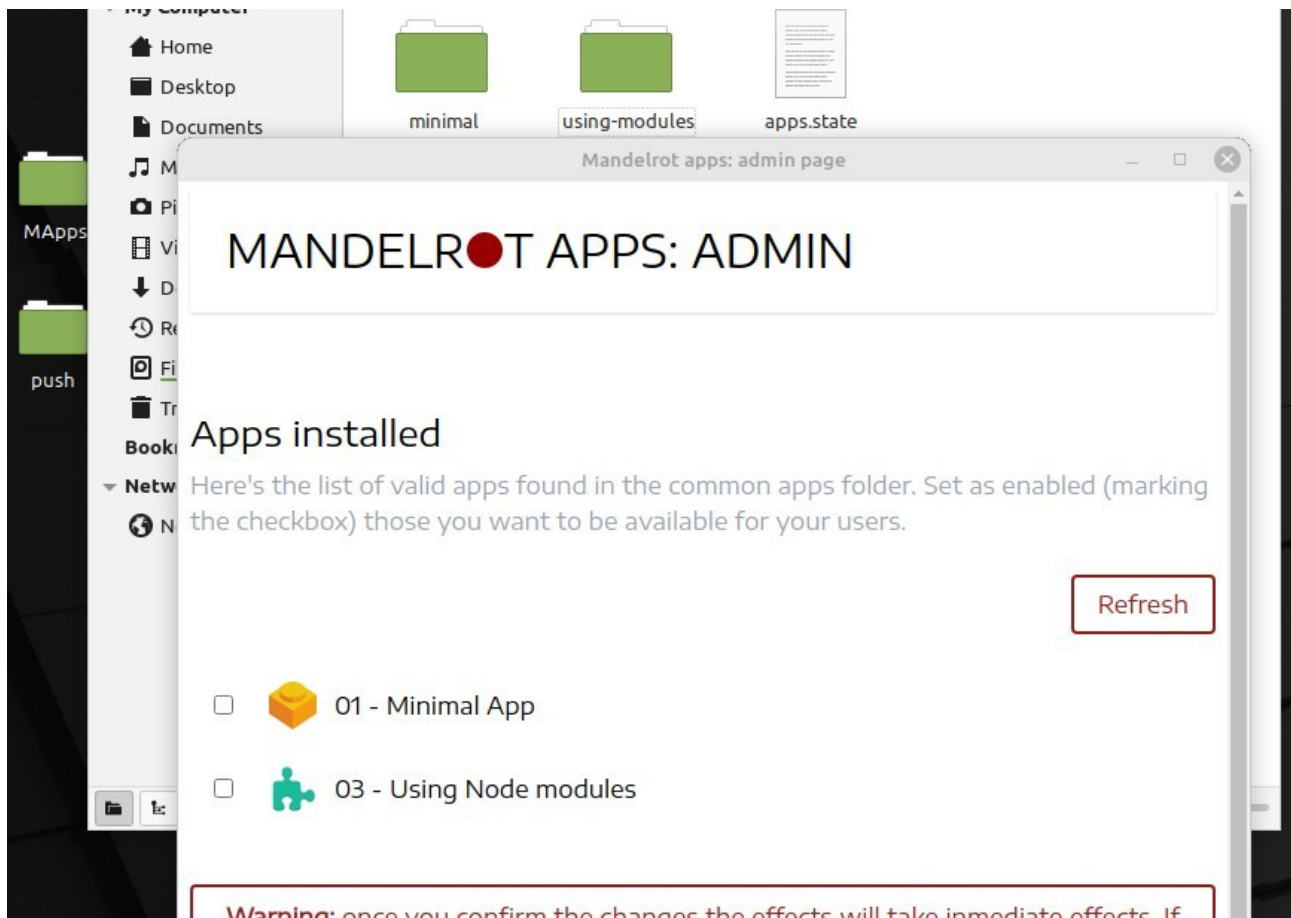
So this is what we have:



And now, just by clicking the “Refresh” button...



Still with the admin window open, I will move the “push” folder to the desktop (outside the APPS folder) and click Refresh again. This is what happens:



So you see. The admin should never need to restart anything or do anything special to admin the suite. Installing or uninstalling frontend apps in the suite is as easy as moving the folder into the APPS folder and refreshing the admin button.

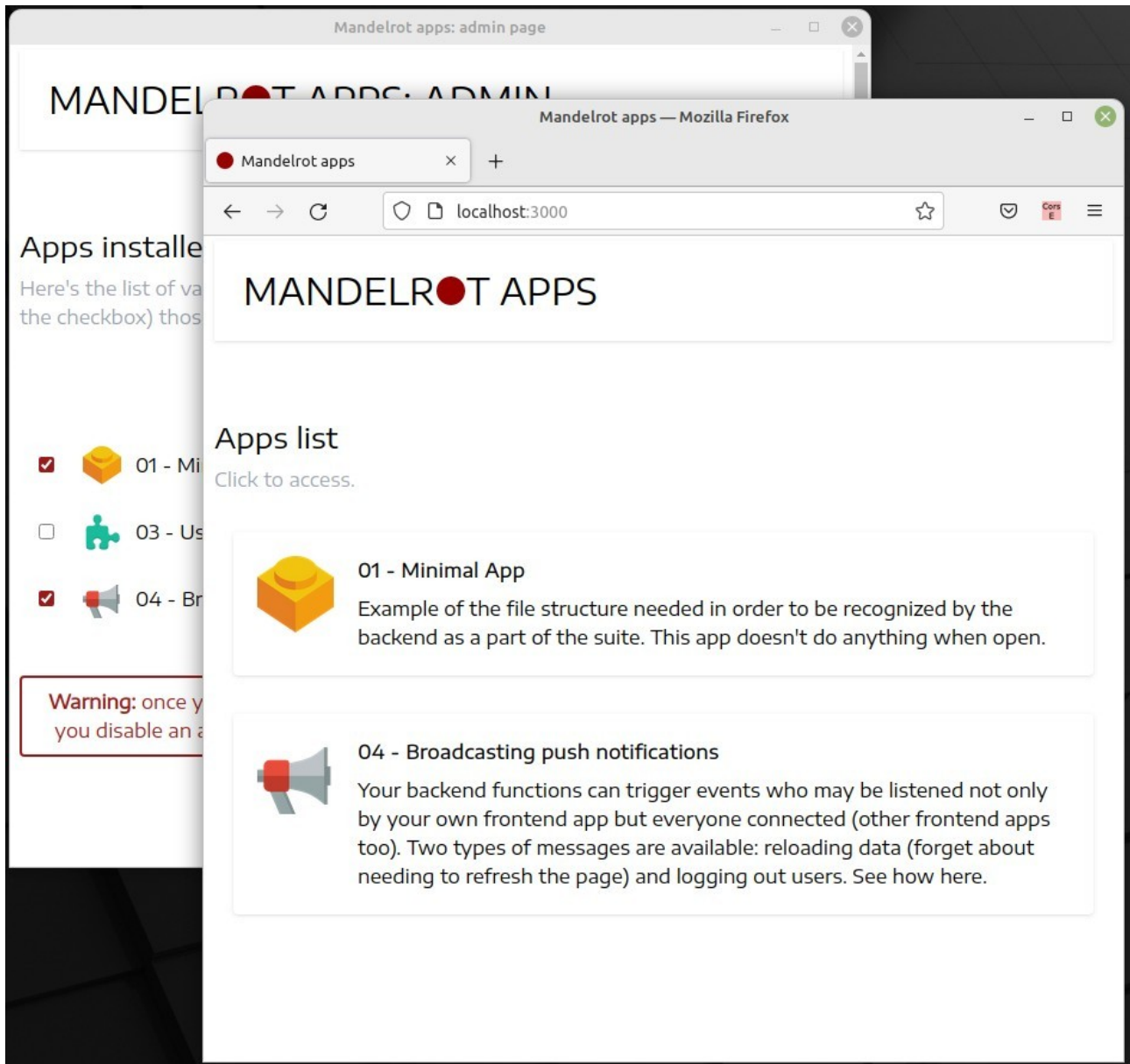
I will restore the “push” app to the APPS folder and refresh the admin window to continue with our three example apps.

Important: only the folders with a valid internal app structure will be accepted by the engine and included in the installed apps list. If you move any element into the APPS folder and the engine doesn't recognize it (while recognize others), you can be 100% sure it does not have a valid file structure. For more info please see the frontend developer guide available.

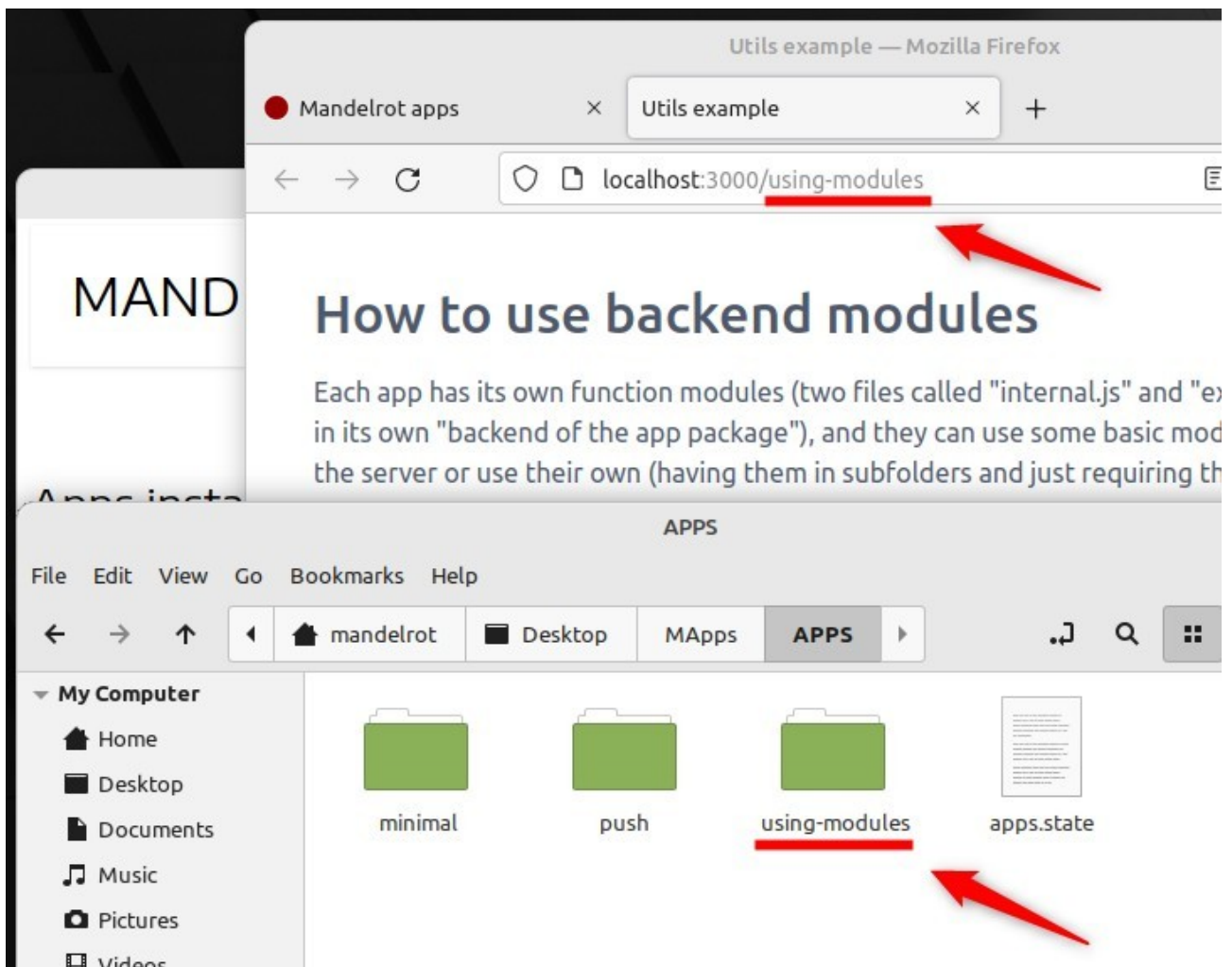
More important: the admin should NEVER change the app folder name, because it's used internally as an unique ID. The developer should have this in mind when creating the frontend app (see frontend developer guide).

Now we will enable/disable apps so our users will see them in the browser. Note: the browser window will self-refresh, any change you make in the admin window will be instantly updated for them.

So I will open a browser window again pointing at localhost:your-port. In the admin window select two of the apps and press the red “Confirm” button at the bottom right.



And then your users will access each app just by clicking any of the list items (a new tab will open). Notice that the url of the frontend app is an extension of the main page, plus the name of the app folder.



Important: the admin can check from the server that the app is installed (that's their job), but the app internal logic should not work from here because of the CORS security policy the modern browsers have.

This has nothing to do with this software: nowadays all the major browsers have restrictions to NOT allow local access to resources that should be remote. Your users (connected from their pc's and remotely accessing the server) won't have any problem, but the internal functionality of the frontend app shouldn't be available directly within the server at "localhost".

In the moment of writing this guide Firefox has an extension called "CORS everywhere" that bypasses this restriction, and in fact I have used it for testing. But remember: this is a bad security practice and the system admin should never use it. You check the app works, your users will remotely access and use it.

## ID control app (new in v2)

Normally when you have a software that involves users (not the only usecase, but the most common one) you need to identify who wants to do something or access to some information in order to determine which information the backend will provide. Example: you want to see your emails, and the backend needs to know who you are in order to show you only your messages.

You could do this in the version 1 of this software but, since the engine was intended to be as abstract as possible, it was more complicated for the programmers. Basically, when an app asked something to the backend, AFTER receiving the request the targeted app needed to see if any other app had information about this user to grant the permissions needed to do that thing. The system admins had nothing to do here, everything was managed by the frontends.

Since the version 2 the sysadmins have a new (optional) task: if they want, they can set an “ID control app” that will intercept and identify the requests BEFORE letting them pass to the targeted apps. This makes the system much faster and easier to manage, and the sysadmin will only need to say which app will do that by default.









### Set ID control app

If you have an app capable of doing users validation, you can tell the system to check the identity of who is calling (via token, see docs) before executing any required task. The dropdown button list below will display which apps currently installed can do that so you can select what you want.

**Important:** you are allowed to set a ID control app that is not active in the apps list. If you do so, your app will be used only for this checking purpose and nothing else.

No ID control app set ▼

If you don't set any app the system will work same as in v1: the frontend apps will have all the work. However now the system will check your apps list to see whether there are some of them that can do the ID control and (if any) you will see them displayed in the dropdown list.

- ☐  01 - Minimal App
- ☐  02 - Minimal functioning App
- ☐  03 - Using Node modules
- ☐  04 - Broadcasting push notifications
- ☐  05 - Database example
- ☐  06 - Files management
- ☐  07 - ID control - App with token
- ☐  08 - ID control - Validator


## Set ID control app

If you have an app capable of doing users validation, you can tell the system to check the identity of who is calling (via token, see docs) before executing any required task. The dropdown button list below will display which apps currently installed can do that so you can select what you want.

**Important:** you are allowed to set a ID control app that is not active in the apps list. If you do so, your app will be used only for this checking purpose and nothing else.

No ID control app set ▾

-- No ID control app --

 08 - ID control - Validator

**Warning:** once you confirm the changes the effects will take immediate effects. If you disable an app currently working your users may suffer information losses.

As you see in this example, we have several apps installed and only one of them is able to do the ID control. Its name here is just an example, normally it should be an “Users” app (or similar) that should store all the users data and login information, and then let the others app know who is calling.











If the admin designates an app to be the system default ID control app, but they want to leave some frontend apps out of this control (because they don't need identification or by any other reason) it can be easily done by marking these frontend apps as exceptions. The exceptions list will not be displayed until an app is designated to do the ID control:

08 - ID control - Validator ▾

### Exceptions

(In case you want some of your apps to be used without token checking)

- ☐  01 - Minimal App
- ☐  02 - Minimal functioning App
- ☐  03 - Using Node modules
- ☐  04 - Broadcasting push notifications
- ☐  05 - Database example
- ☐  06 - Files management
- ☐  07 - ID control - App with token
- ☐  08 - ID control - Validator

Usually the developers will document all the frontend app features and the admins will know how the apps work (so they will understand the implications of setting an ID control app). Anyway, this feature is optional so it can be just not used.

## **Backups and migrations**

When you have a 100% portable system the migration and backup explanations are short. You copy the MApps folder wherever you want, run the executable file and that's it. Everything works again right away.

If you compile your own engine (with your own PASSPHRASE) you definitely should keep a copy of the engine main folder somewhere else. This pass will be used in an internal encryption module available for the frontend apps too, meaning if you forget the passphrase and the developer has used it to encrypt something in their frontend app, if you have to compile the engine again but you don't use the same passphrase in the config, the front apps will NOT be able to decrypt the information encrypted with the lost pass. Watch out!

Partial frontend backups: all the frontend apps information is stored within the frontend app folder itself, and each developer will decide where exactly they will locate it. So if you want to backup specific items (databases and so on) you will need to ask your developer in which sub-folder they placed it. The abstraction of this concept implies that the developer actively decides what's optimal for their app. The good news is that all is based in the filesystem, so it will just a matter of knowing which sub-elements to copy... In case you don't want to backup the whole app folder to make it even simpler.