

VS Code Docker integration

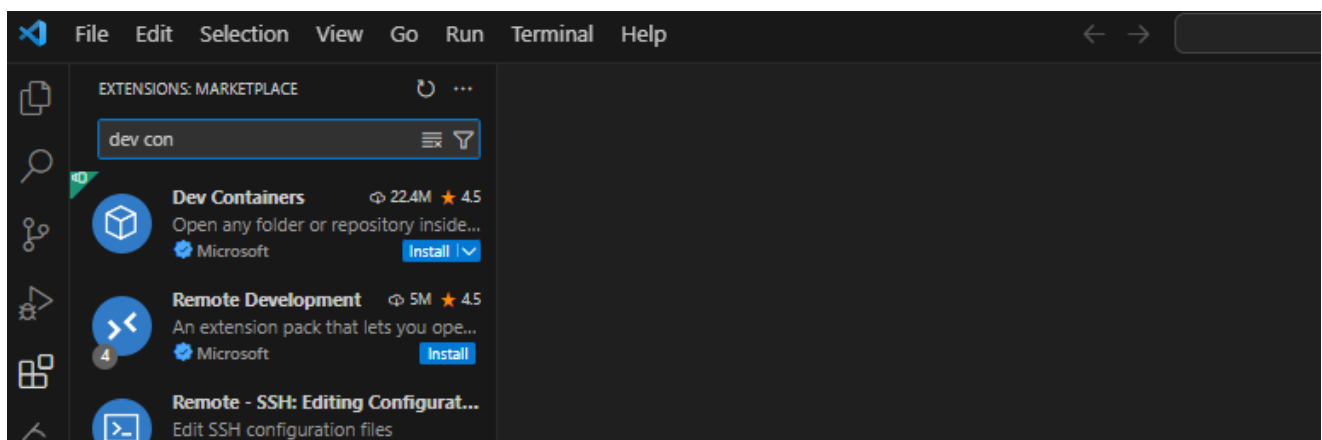
By JorgePRamos

In this guide we will show how to set up **Dev containers**, which allows us to develop in different OS with the full support of our IDE (linting, auto completion, plugins ...)

Important note: we need Docker to be installed in our machine.

Preliminary steps

Before we need to have both installed: VS Code and the "Dev Containers" extension.



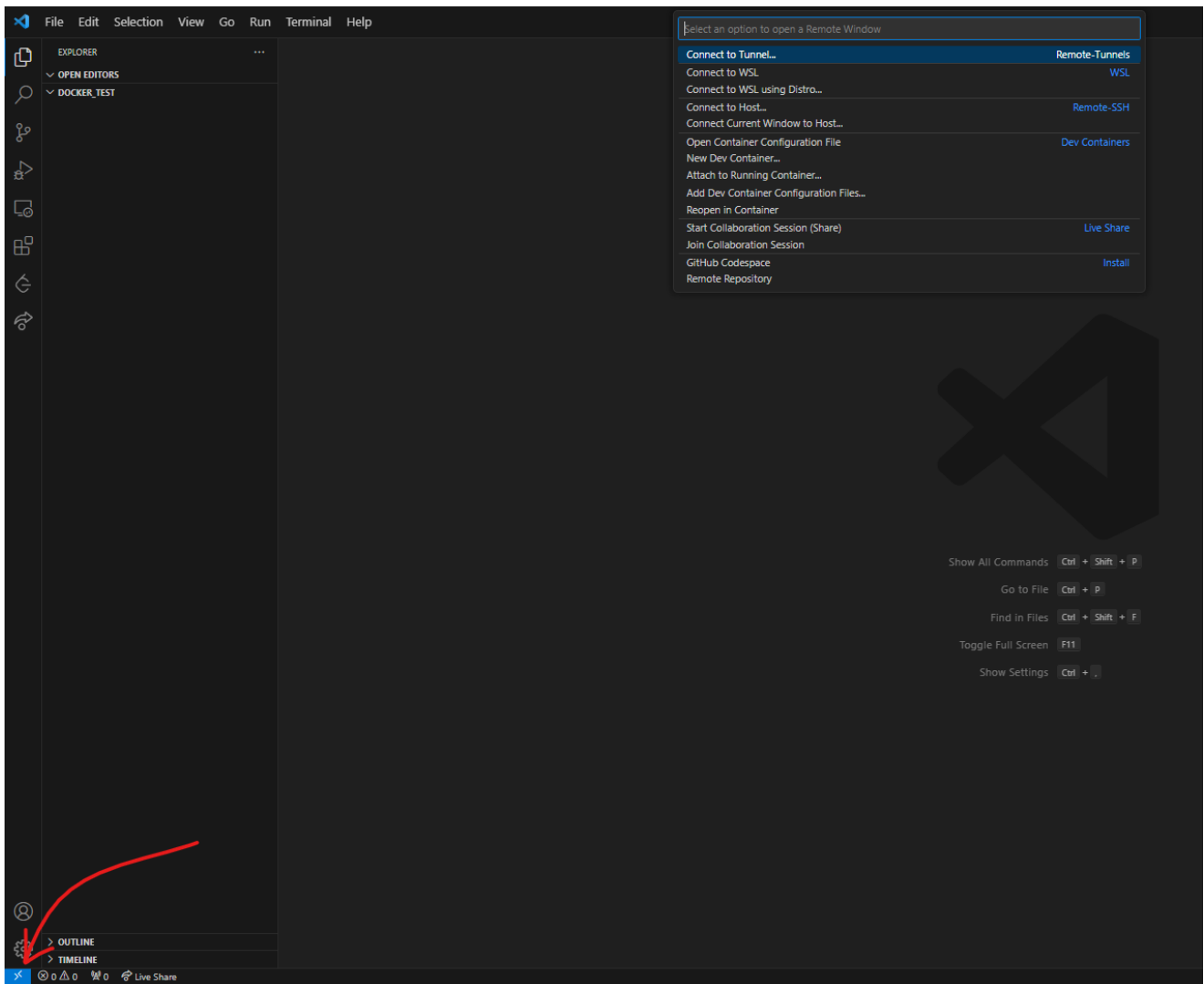
Note: A good practice is to install the "Remote Development" pack by Microsoft, which includes some additional extension that may be useful in our development as well as the "Dev Containers" extension.

Dev Containers use

After the quick installation of the necessary extensions we are ready to start developing.

Lets create a new folder which will act as our development root.

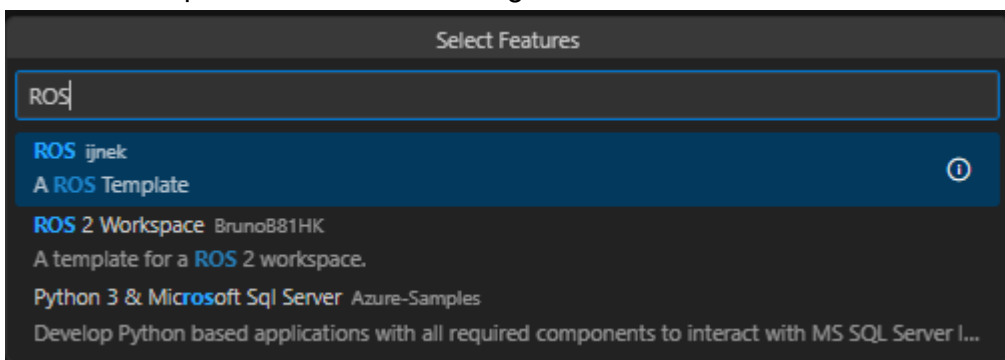
From here we need to click the blue icon on the bottom left corner (remote session button), which will bring up a context menu about remote options.



Here we will chose the option "Reopen in container" .

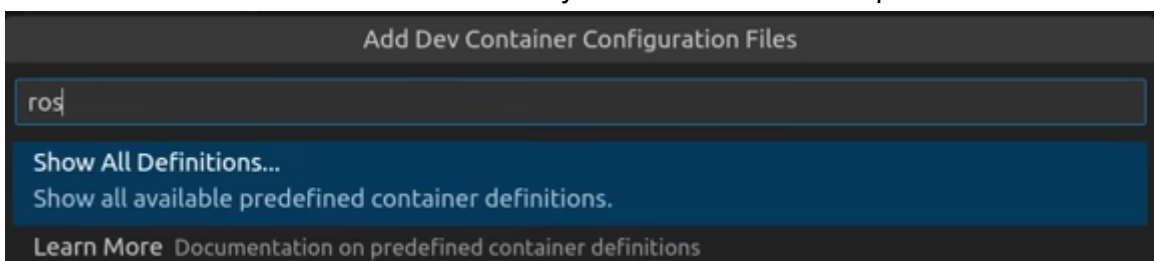
Now is we are presented with an extensive list of docker images.

For our example lets use a ROS2 image.

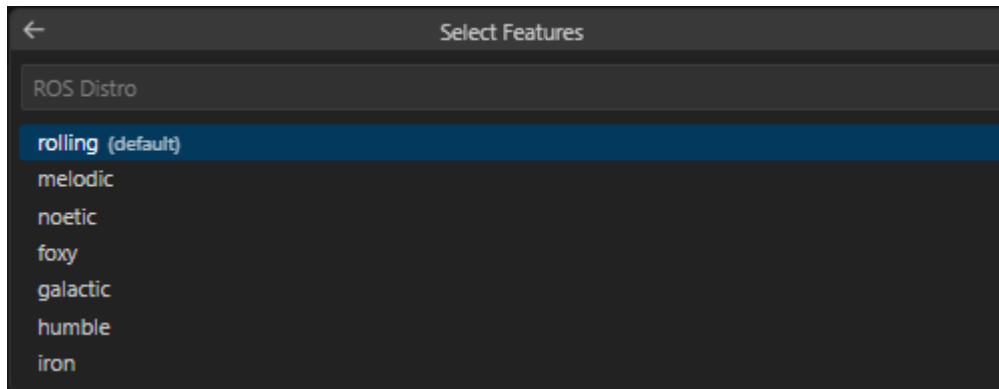


In our case we will chose the image provided by "ijnek" real name "Kenji Brameld" which is a renown ROS2 contributor.

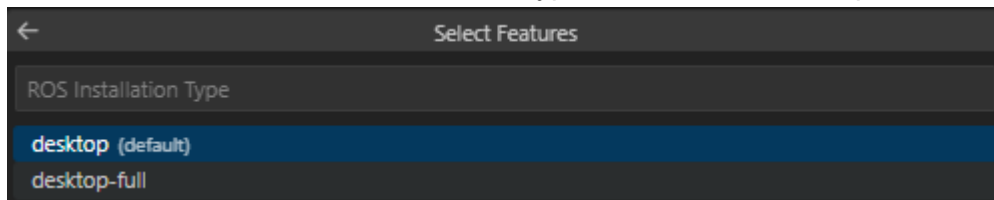
Note: If our search is unsuccessful we may want to to check the option "Show All Definitions".



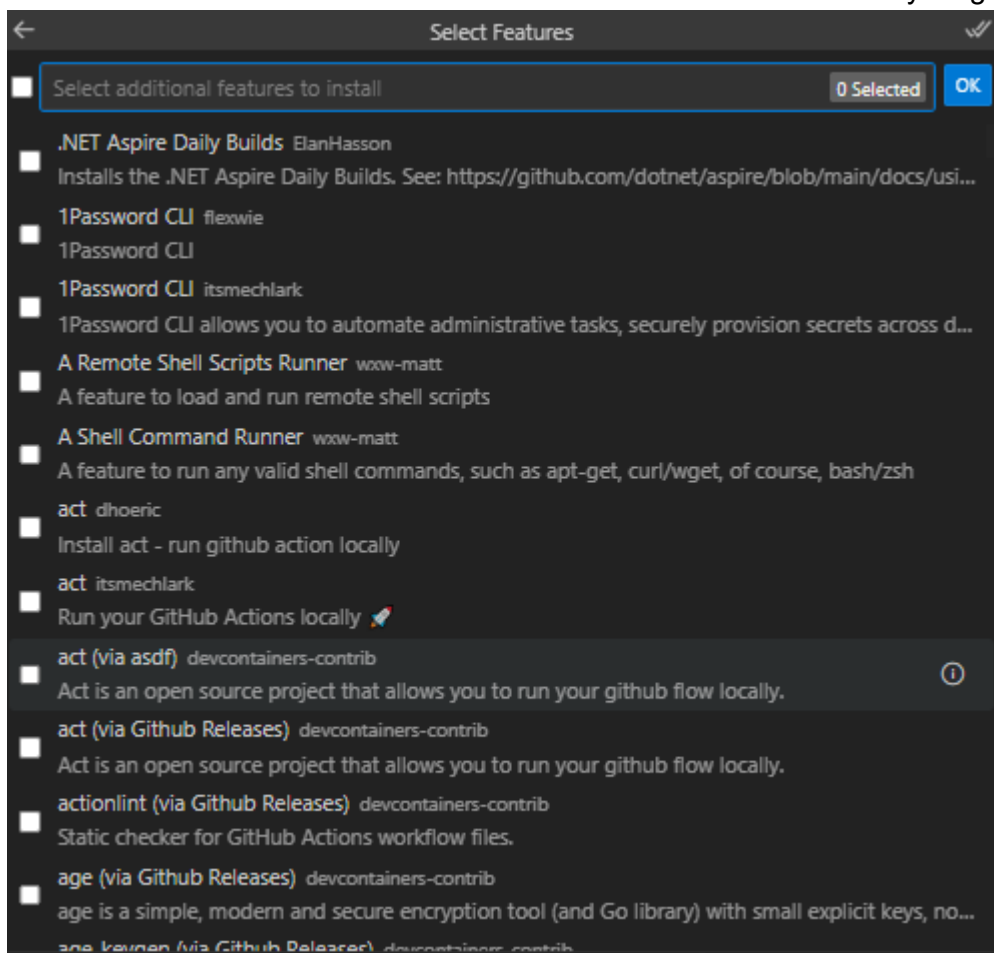
Then we can choose our distribution in our case "humble"



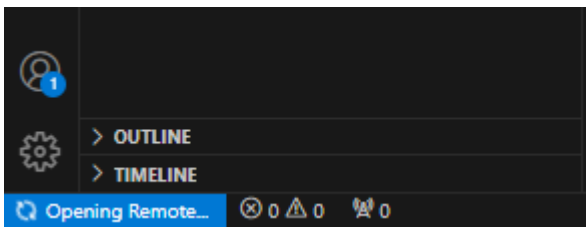
Then we need to select our installation type. in our case "desktop"



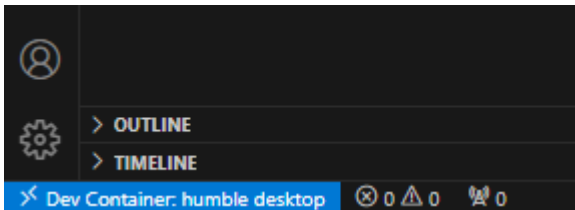
And lastly we will be prompted to install additional features by default which can be still be added after the creation of our container. In our case we will not choose anything.



After this our container will be created and we can wait in the meantime and maybe grab a coffee or a tea, in our case coffee (the superior option) ;)



We will know that the build has finished when the bottom left icon changes to our containers name.



Now any terminal we open in Vs Code will be automatically inside of our container.

```
TERMINAL  PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE

vscode@docker-desktop:/Docker_Test$ ros2 -h
usage: ros2 [-h] [--use-python-default-buffering] Call `ros2 <command> -h` for more details

ros2 is an extensible command-line tool for ROS 2.

options:
  -h, --help            show this help message and exit
  --use-python-default-buffering
                        Do not force line buffering in stdout and instead use the default buffering

Commands:
  action              Various action related sub-commands
  bag                 Various rosbag related sub-commands
  component            Various component related sub-commands
  daemon              Various daemon related sub-commands
  doctor              Check ROS setup and other potential issues
  interface            Show information about ROS interfaces
```

Talker - Listener example

Lets run a couple of ROS2 nodes to celebrate. Neat!

```
TERMINAL  PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE

vscode@docker-desktop:/Docker_Test$ source /opt/ros/humble/setup.bash
vscode@docker-desktop:/Docker_Test$ ros2 run demo_nodes_cpp talker
[INFO] [1707485512.661675507] [talker]: Publishing: 'Hello World: 1'
[INFO] [1707485513.661673312] [talker]: Publishing: 'Hello World: 2'
[INFO] [1707485514.661673443] [talker]: Publishing: 'Hello World: 3'
[INFO] [1707485515.661677048] [talker]: Publishing: 'Hello World: 4'
[INFO] [1707485516.661673370] [talker]: Publishing: 'Hello World: 5'
[INFO] [1707485517.664462415] [talker]: Publishing: 'Hello World: 6'
[INFO] [1707485518.668749542] [talker]: Publishing: 'Hello World: 7'
[INFO] [1707485519.668767088] [talker]: Publishing: 'Hello World: 8'
[INFO] [1707485520.668781051] [talker]: Publishing: 'Hello World: 9'
[INFO] [1707485521.668781205] [talker]: Publishing: 'Hello World: 10'
[INFO] [1707485522.668776080] [talker]: Publishing: 'Hello World: 11'
[INFO] [1707485523.668769221] [talker]: Publishing: 'Hello World: 12'
[INFO] [1707485524.668779497] [talker]: Publishing: 'Hello World: 13'
[INFO] [1707485525.668777301] [talker]: Publishing: 'Hello World: 14'
[INFO] [1707485526.668752113] [talker]: Publishing: 'Hello World: 15'
[INFO] [1707485527.668704677] [talker]: Publishing: 'Hello World: 16'
[INFO] [1707485528.668655229] [talker]: Publishing: 'Hello World: 17'
[INFO] [1707485529.668599475] [talker]: Publishing: 'Hello World: 18'
[INFO] [1707485530.668540789] [talker]: Publishing: 'Hello World: 19'

vscode@docker-desktop:/Docker_Test$ source /opt/ros/humble/setup.bash
vscode@docker-desktop:/Docker_Test$ ros2 run demo_nodes_py listener
[INFO] [1707485517.677138162] [listener]: I heard: [Hello World: 6]
[INFO] [1707485518.669529442] [listener]: I heard: [Hello World: 7]
[INFO] [1707485519.669488468] [listener]: I heard: [Hello World: 8]
[INFO] [1707485520.669675164] [listener]: I heard: [Hello World: 9]
[INFO] [1707485521.669539705] [listener]: I heard: [Hello World: 10]
[INFO] [1707485522.669512846] [listener]: I heard: [Hello World: 11]
[INFO] [1707485523.669453853] [listener]: I heard: [Hello World: 12]
[INFO] [1707485524.669492582] [listener]: I heard: [Hello World: 13]
[INFO] [1707485525.669537260] [listener]: I heard: [Hello World: 14]
[INFO] [1707485526.669528362] [listener]: I heard: [Hello World: 15]
[INFO] [1707485527.669515449] [listener]: I heard: [Hello World: 16]
[INFO] [1707485528.669327026] [listener]: I heard: [Hello World: 17]
[INFO] [1707485529.669341392] [listener]: I heard: [Hello World: 18]
[INFO] [1707485530.669347303] [listener]: I heard: [Hello World: 19]
[INFO] [1707485531.669296141] [listener]: I heard: [Hello World: 20]
[INFO] [1707485532.669083877] [listener]: I heard: [Hello World: 21]
[INFO] [1707485533.669276298] [listener]: I heard: [Hello World: 22]
[INFO] [1707485534.668918440] [listener]: I heard: [Hello World: 23]
[INFO] [1707485535.668450143] [listener]: I heard: [Hello World: 24]
```

Working with the Dev Container

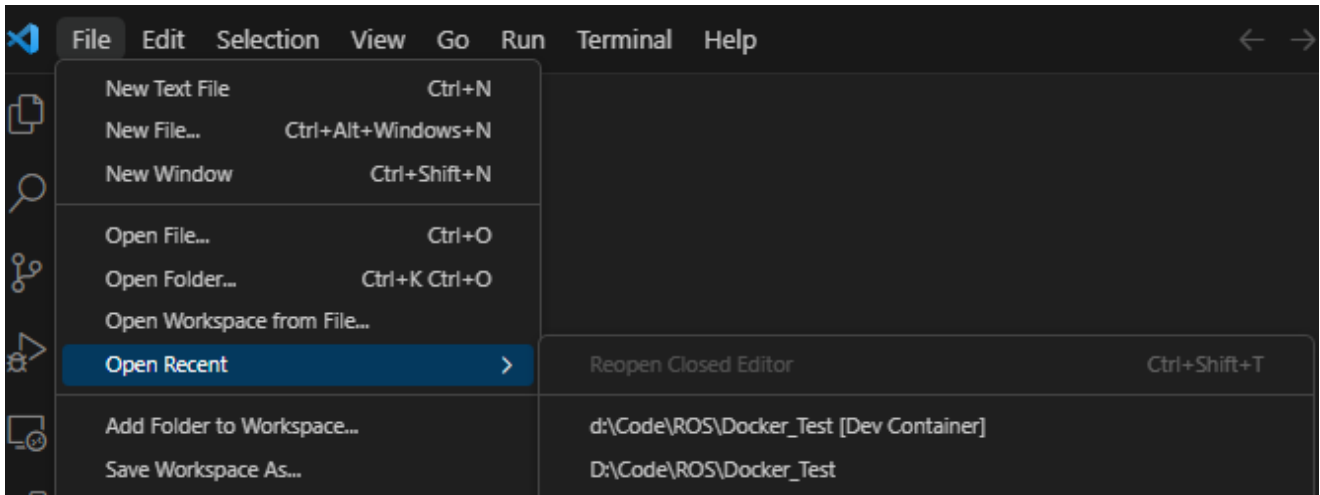
Extensions

Extensions work different when working inside of container and we have to understand the we will have local extension, being the ones you already have installed and remote extension which will reside specifically on our container and will aid us in development.

Reopening an existing container

We can can open back where we left selecting our container by:

File --> Open Recent --> [Container name]



GUI for our containers

In order to have access to GUI applications that we may run in our container like RViz or Gazebo, we must edit our container `.json` .

We will be adding the desktop lite feature

And forwarding a couple ports from our container

```
"features": {  
  "ghcr.io/devcontainers/features/desktop-lite:1": {},  
},  
"forwardPorts": [6080, 5901],  
"portsAttributes": {  
  "6080": {  
    "label": "Desktop-Web"  
  },  
  "5901": {
```

```
"label": "Desktop-VNC"
```

```
}
```

```
}````
```

****Important note:**** Do not forget to separate with commas in order to follow the JSON syntax.

Last step is to delete the argument from `runArgs` named `"--network=host"` which can prevent the launch of desktop lite.

Once the changes are done we will either get prompted by Vs Code to rebuild our container or do ourselves.

Lets open the command pallet and look for the "Rebuild Container" option.

![[Pasted image 20240209170951.png]]

After its finished building we will notice 2 ports opened in Vs Code

![[Pasted image 20240209171536.png]]

If we open our browser of choice and enter: `localhost:6080` will be greeted with the following screen.

![[Pasted image 20240209171651.png]]

Click "Connect" and type `vscode` as the default password.

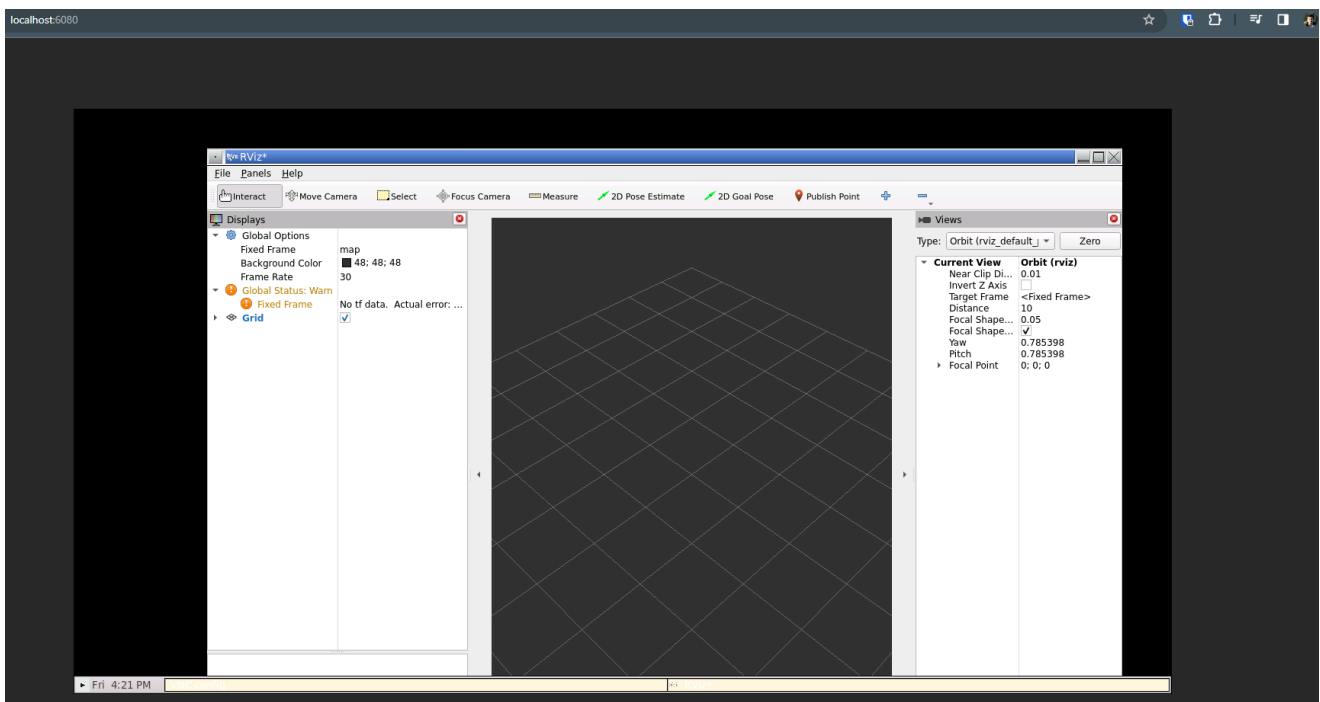
****Lo and behold our Linux desktop.****

Lets try starting `rviz` by running the following command in our terminal.

```
`` bash
```

```
source /opt/ros/humble/setup.bash
```

```
rviz2
```



From now on we can use our browser as our "virtual monitor" leaving behind those dark days of having 2 keyboards, two mouses and 2 screen to work directly on our ROS machine.