Robotic Operating System (ROS)

Aula 2 –Configuração

Professores: André L.M. Marcato, Iago Z. Biundini, Milena F. Pinto

Universidade Federal de Juiz de Fora

Programa de Pós-Graduação em Engenharia Elétrica

Instalação

O primeiro passo para utilizar o ROS é a instalação do sistema operacional. O sistema ROS permite a instalação nas seguintes plataformas:

Instalação do ROS

A instalação do ROS pode ser encontrada em:
http://wiki.ros.org/noetic/Installation/Ubuntu

A própria empresa tem um tutorial para aplicação dos conhecimentos:

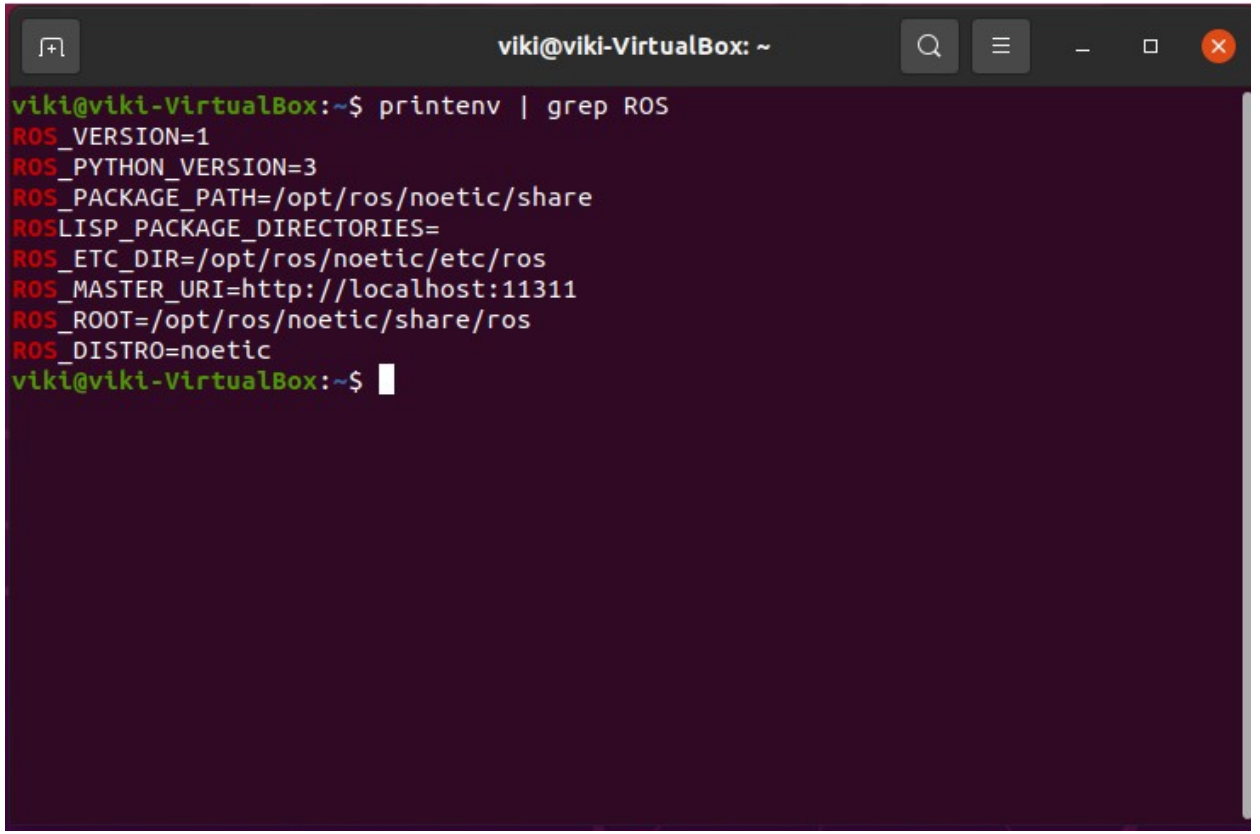
Em inglês: http://wiki.ros.org/ROS/Tutorials
Em Português: http://wiki.ros.org/pt_BR/ROS/Tutorials

Robotic Operating System - ROS

Gerenciando seu ambiente

- Se você estiver tendo problemas para encontrar ou usar seus pacotes ROS, certifique-se de ter seu ambiente configurado corretamente.
- Uma boa maneira de verificar é garantir que as variáveis de ambiente como ROS_ROOT e ROS_PACKAGE_PATH estejam definidas:

```
printenv | grep ROS
```



Gerenciando seu ambiente

- Se você acabou de instalar o ROS a partir do apt no Ubuntu, você terá os arquivos setup.*sh em “/opt/ros/distro/”, e você pode adicioná-los assim:

```
source /opt/ros/<distro>/setup.bash
```

Você precisará executar este comando em cada novo terminal que abrir para ter acesso aos comandos do ROS, a menos que adicione esta linha ao seu .bashrc.

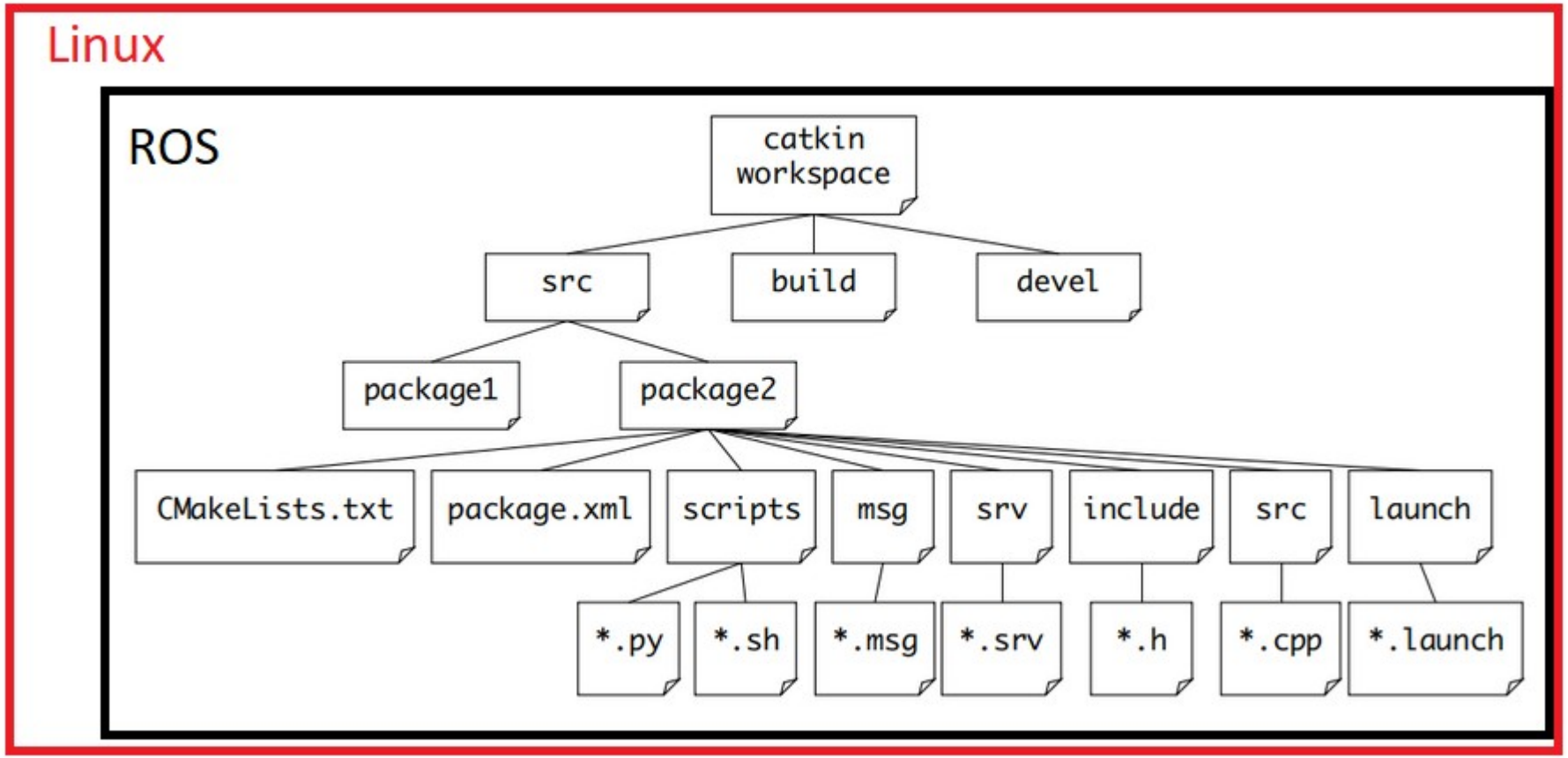
- No nosso caso estamos usando o ROS-Noetic, versão de 2020 e terá suporte até 2025, então o comando fica

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

Criar um espaço de trabalho ROS

Os pacotes catkin podem ser construídos como um projeto autônomo, da mesma forma que os projetos normais do cmake podem ser construídos, mas catkin também fornece o conceito de espaços de trabalho (workspace), onde você pode construir vários pacotes interdependentes juntos.

Esse ambiente de trabalho é isolado do Linux, permitindo instalação de pacotes sem interferir nos arquivos originais do sistema operacional



Criar um espaço de trabalho ROS

- A primeira etapa é a criação de uma pasta para o workspace:

```
mkdir -p ~/<nome_do_workspace>/src
```

- Um nome bastante comum é: catkin_ws

```
mkdir -p ~/catkin_ws/src
```

- O próximo passo é entrar na pasta "catkin_ws":

```
cd ~/catkin_ws/
```

- O comando catkin_make é uma ferramenta conveniente para trabalhar com áreas de trabalho catkin. Executando-o pela primeira vez em seu espaço de trabalho, ele criará um link CMakeLists.txt

```
catkin_make
```

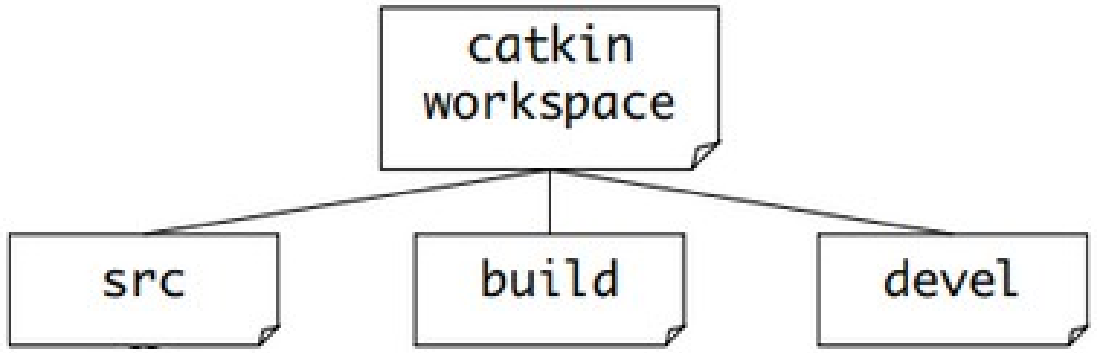
Robotic Operating System - ROS

Criar um espaço de trabalho ROS

```
#### Running command: "cmake /home/viki/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/home/viki/catkin_ws/dev
el -DCMAKE_INSTALL_PREFIX=/home/viki/catkin_ws/install -G Unix Makefiles" in "/home/viki/catkin_ws/b
uild"
####
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Using CATKIN_DEVEL_PREFIX: /home/viki/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/noetic
-- This workspace overlays: /opt/ros/noetic
-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.10", minimum required is "3")
-- Using PYTHON_EXECUTABLE: /usr/bin/python3
-- Using Debian Python package layout
-- Found PY_em: /usr/lib/python3/dist-packages/em.py
-- Using empy: /usr/lib/python3/dist-packages/em.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/viki/catkin_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/googletest': gtests will be built
-- Found gmock sources under '/usr/src/googletest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Found Threads: TRUE
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/viki/catkin_ws/build
####
#### Running command: "make -j6 -l6" in "/home/viki/catkin_ws/build"
####
```

Robotic Operating System - ROS

Criar um espaço de trabalho ROS



Se necessário, as pastas build e devel podem ser apagadas inteiramente com o comando:

```
catkin_clean
```

src	build	devel
Trabalhe aqui	Não mexa	Não mexa
A pasta src ou source contém o código-fonte. É aqui que você pode clonar, criar e editar o código-fonte dos pacotes que deseja construir.	A pasta de construção (build) é onde o CMake é chamado para criar os pacotes na pasta de origem. Informações de cache e outros arquivos intermediários são mantidos aqui.	A pasta de desenvolvimento (devel) é onde os pacotes construídos são colocados (antes de serem instalados).

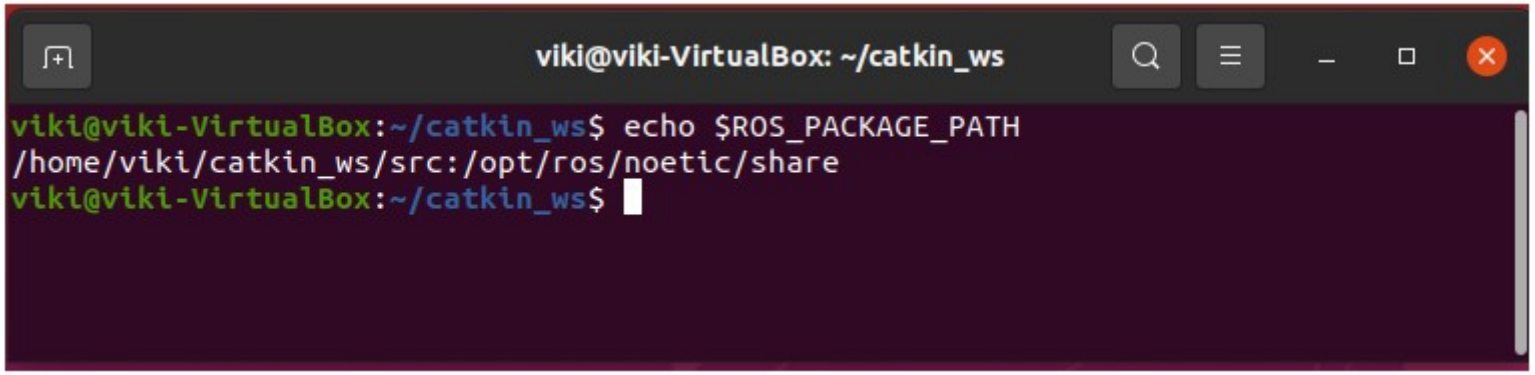
Criar um espaço de trabalho ROS

- Antes de continuar, vamos abrir a fonte de seu novo arquivo setup.*sh:

```
source devel/setup.bash
```

- Para certificar-se de que seu espaço de trabalho seja sobreposto corretamente pelo script de configuração, certifique-se de que a variável de ambiente ROS_PACKAGE_PATH:

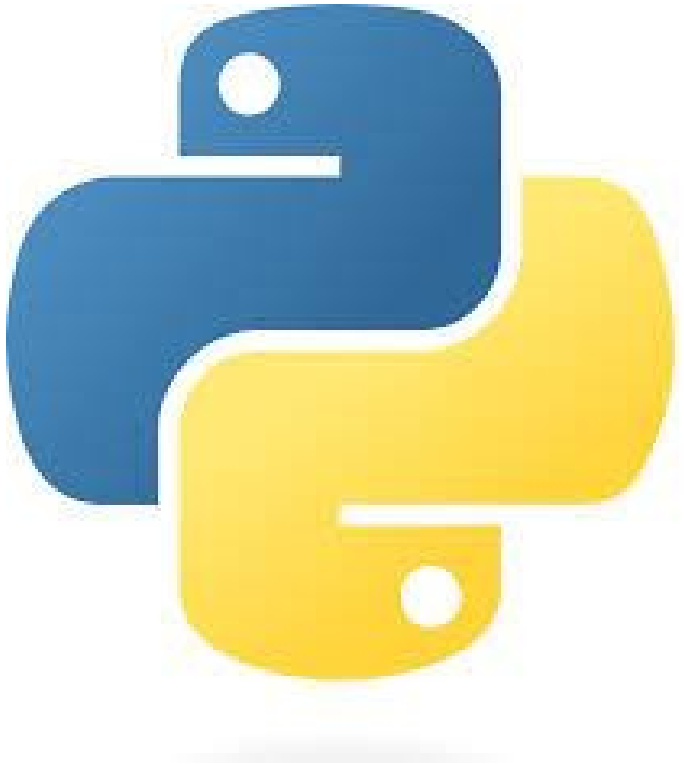
```
echo $ROS_PACKAGE_PATH
```



Instalando e Configurando o Pycharm

ROS

Linguagem de Programação



Linguagem de Programação

Fácil aprendizado	**Curva de aprendizado mais lenta**
Fácil acesso a bibliotecas	Bibliotecas com menor compartilhamento pela comunidade
Mais lenta	Mais rápida
Interpretada	Pré compilada (executáveis na máquina)

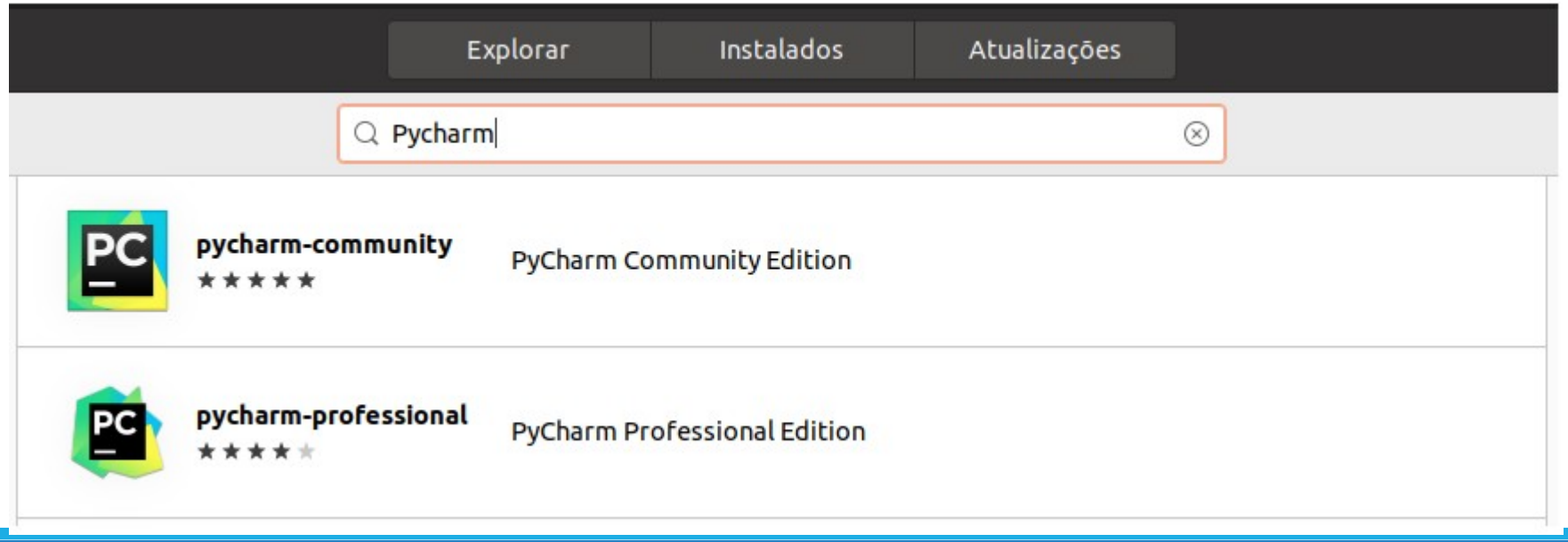
Linguagem de Programação

- Orientada a objetos
- De tipagem dinâmica - As variáveis no Python podem armazenar qualquer tipo de dados, independente do seu valor atual;
- Multiplataforma e open source;
- IA, Machine Learning e Big Data;
- Não utiliza ponto e vírgula (;)para finalizar uma instrução;
- Utiliza indentação por espaços;
- Não há chaves ({}) para delimitar o início e final de um bloco de código

Instalando e Configurando o Pycharm

Sobre Pycharm

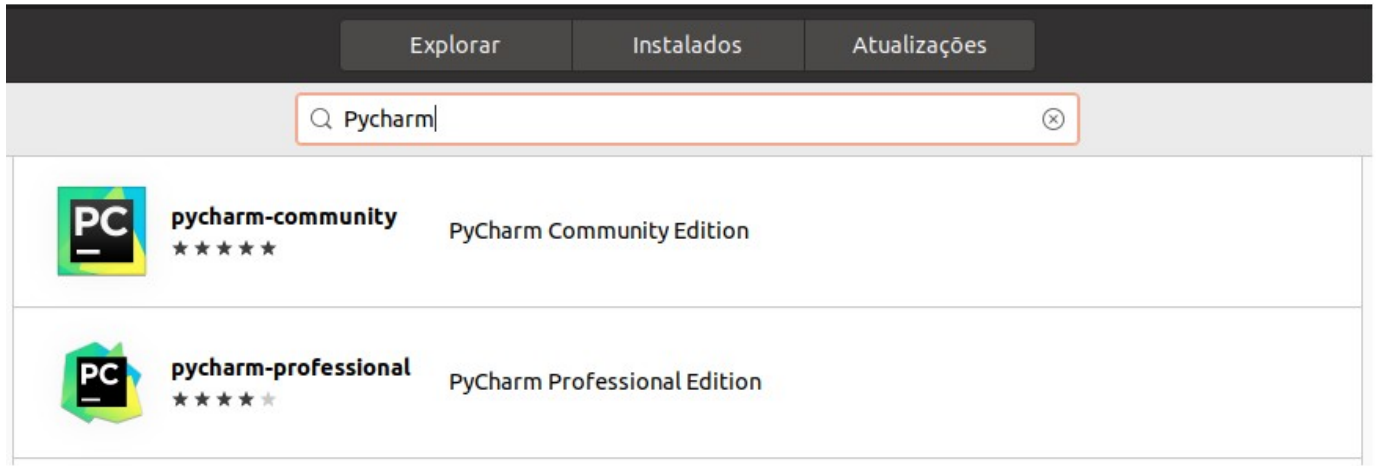
Pycharm é um ambiente de desenvolvimento integrado usado em programação de computadores, especificamente para a linguagem de programação Python. É desenvolvido pela empresa checa JetBrains.



Instalando e Configurando o Pycharm

Sobre Pycharm

- Pycharm é um ambiente de desenvolvimento integrado usado em programação de computadores, especificamente para a linguagem de programação Python. É desenvolvido pela empresa checa JetBrains.



- No link da JetBrains-student é possível utilizar o seu e-mail do @engenharia.ufjf.br para uma licença de estudante que possibilita utilizar o Pycharm-Professional.

Instalando e Configurando o Pycharm

Criação do ambiente Virtual

- Dentro da sua pasta do “ /catkin_ws/src” iremos criar um “virtualenv”.
- Um ambiente virtual é um ambiente Python, de modo que o interpretador, as bibliotecas e os scripts Python instalados nele são isolados daqueles instalados em outros ambientes virtuais e (por padrão) quaisquer bibliotecas instaladas no sistema operacional.

```
sudo apt install python3-virtualenv

cd ~/catkin_ws/src/

virtualenv venv --system-site-
packages
```

Instalação do pacote virtualenv

Ir para a pasta ‘src’ do catkin_ws

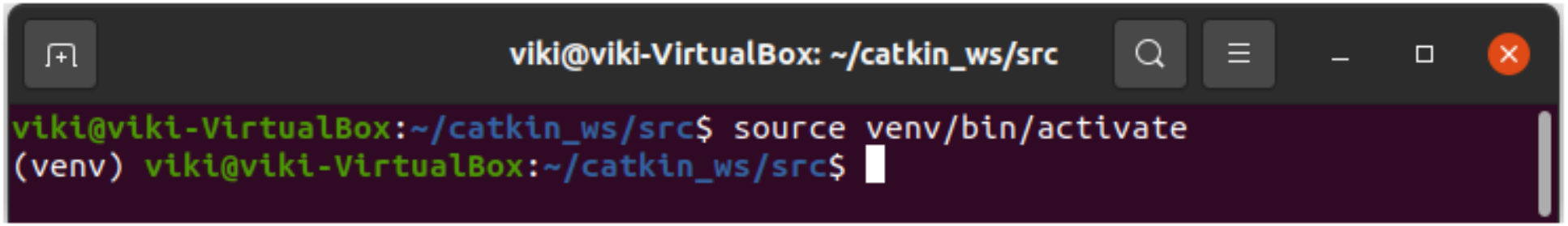
Criação do ambiente virtual com nome venv

Instalando e Configurando o Pycharm

Criação do ambiente Virtual

- Para ativar o virtualenv só fazer:

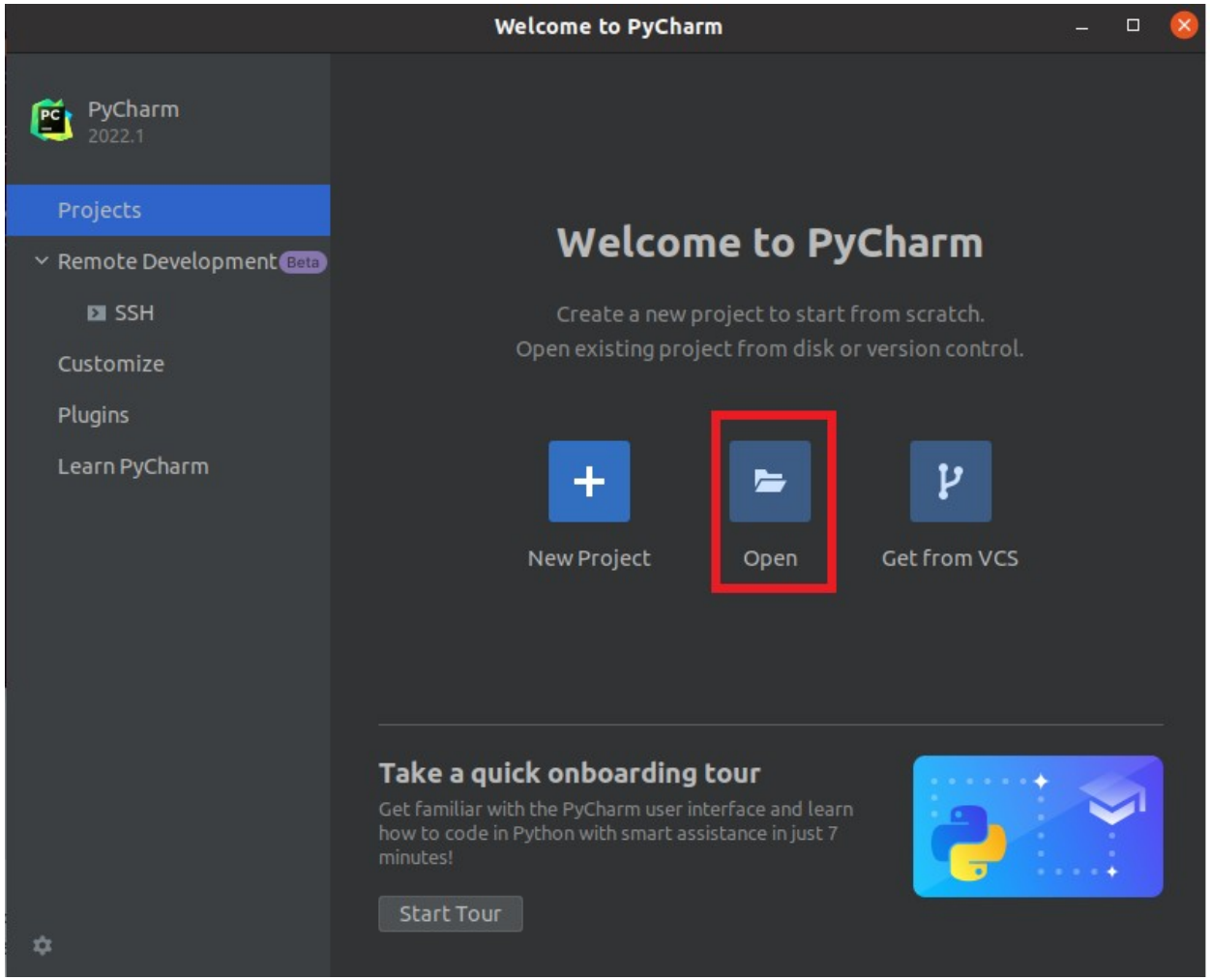
```
source venv/bin/activate
```



O termo (venv) no inicio do terminal mostra que você está usando o ambiente virtual.

Instalando e Configurando o Pycharm

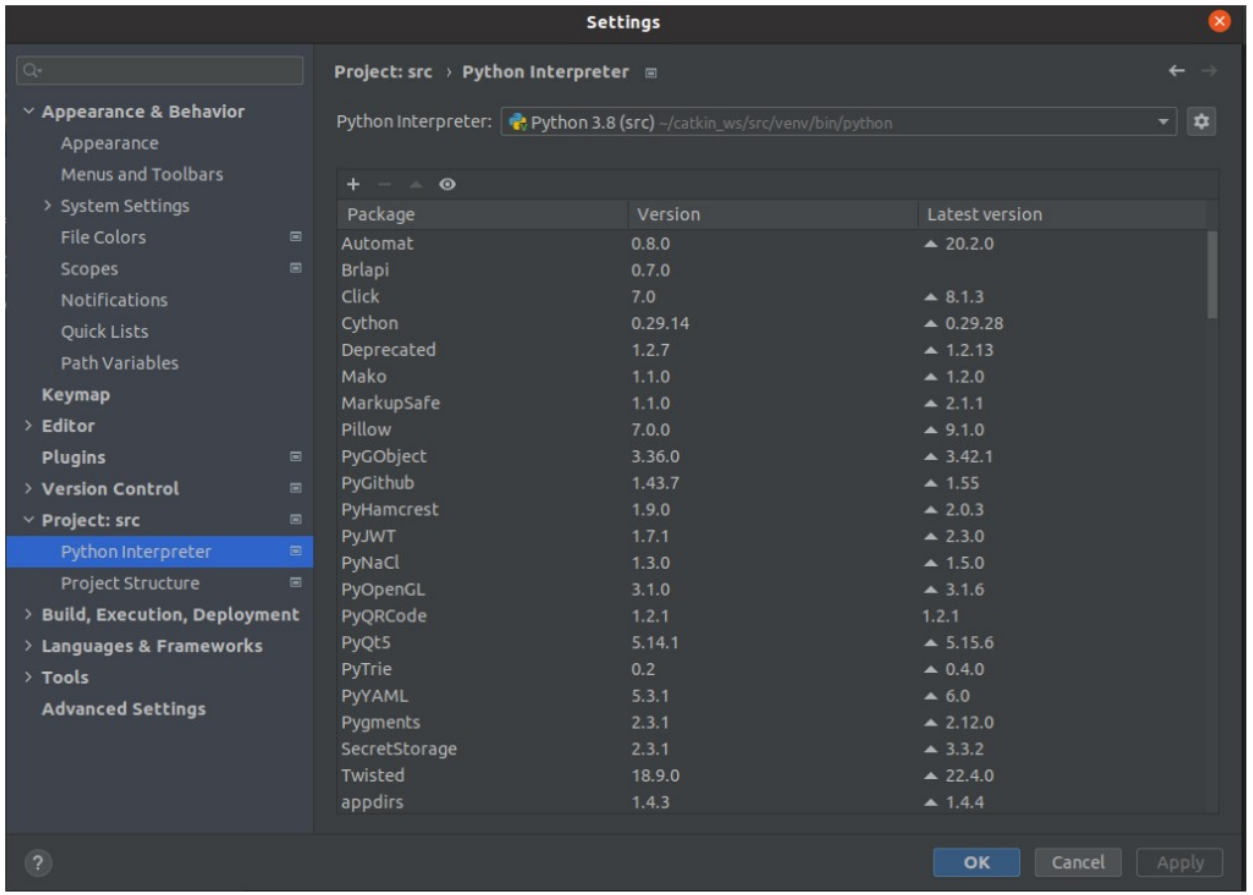
Configurando o Pycharm para o ambiente ROS



- Inicialmente vamos abrir um projeto:
- O projeto aberto será o “ /catkin_ws/src”.

Instalando e Configurando o Pycharm

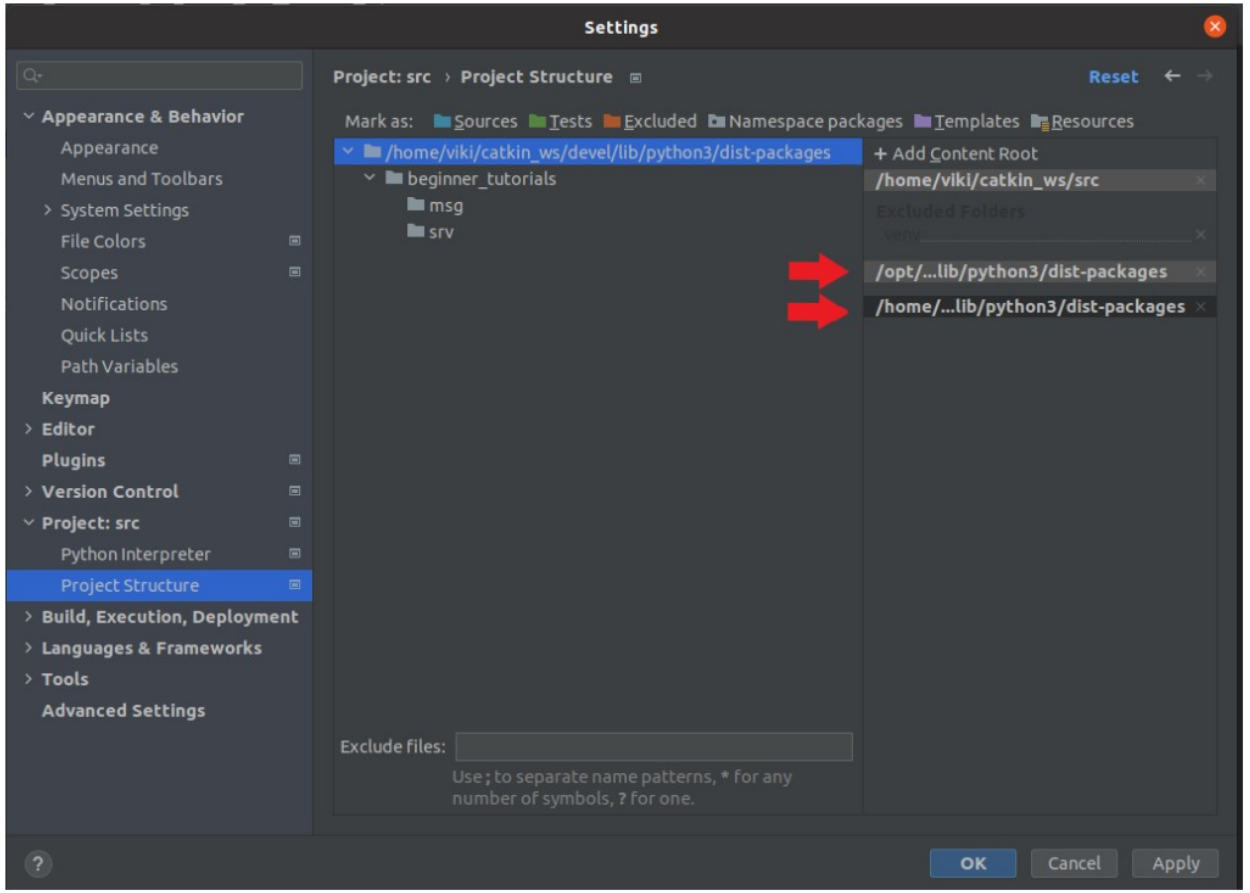
Configurando o Pycharm para o ambiente ROS



- No canto superior esquerdo, em file > setting > Project: src > Python Interpreter teremos a aba

Instalando e Configurando o Pycharm

Configurando o Pycharm para o ambiente ROS



- No canto superior esquerdo, em file > setting > Project: src > Python Structure teremos a aba

- Na opção Add Content Root adicione os seguintes diretórios:
  - /home/user/catkin_ws/devel/lib/python3/dist-packages
  - /opt/ros/distro/lib/python3/dist-packages

A pasta “/home/user/catkin_ws/devel/lib/python3/dist-packages” contem os arquivos de distribuição do Python de seus programas dentro do src realizados pelo catkin_make. Se nenhum pacote tiver arquivos em python, essa pasta não existirá.

Alias

ROS

Alias

O que é Alias?

- “Alias” para comandos no terminal do Linux são uma espécie de atalho ou apelido que você da para comandos maiores afim de agilizar a utilização do terminal. Isso pode ser muito útil para aqueles que trabalham com servidores ou que costumam digitar comandos grandes sem uma interface gráfica.

- Para criar um alias temporário, simplesmente digite no terminal:

```
alias NOME_DO_ATALHO='COMANDO_ORIGINAL'
```

- Exemplo:

```
alias atualiza='sudo apt-get update && sudo apt-get upgrade'
```

Alias

O que é Alias?

```
alias atualiza='sudo apt-get update && sudo apt-get upgrade'
```

- Agora toda vez que você digitar a palavra “atualiza”, o comando descrito acima será executado e os pacotes necessários serão atualizados. Um problema com o alias acima é que ele não será permanente, assim que você fechar o terminal, o mesmo deixará de existir.
- Para criar um alias permanente, podemos editar o arquivo “.bashrc” presente em “home” do sistema Linux. Toda vez que você abrir o terminal o arquivo “.bashrc” é executado. Assim todos os alias ficam registrados.

Alias

Nosso Alias no .bashrc

```
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
source ~/catkin_ws/src/venv/bin/activate

export ROS_HOSTNAME=localhost
export ROS_MASTER_URI=http://${ROS_HOSTNAME}:11311

alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
alias st='cd ~/catkin_ws && source devel/setup.bash'
```

- Importa o .bash do ROS raiz
- Importa o .bash do catkin_ws
- Carrega o venv do ambiente virtual
- IP do Computador que está executando o roscore
- Atalho pro catkin_ws
- Atalho pro src
- Atalho pro catkin_make
- Atalho pra carregar o setup.bash do catkin_ws