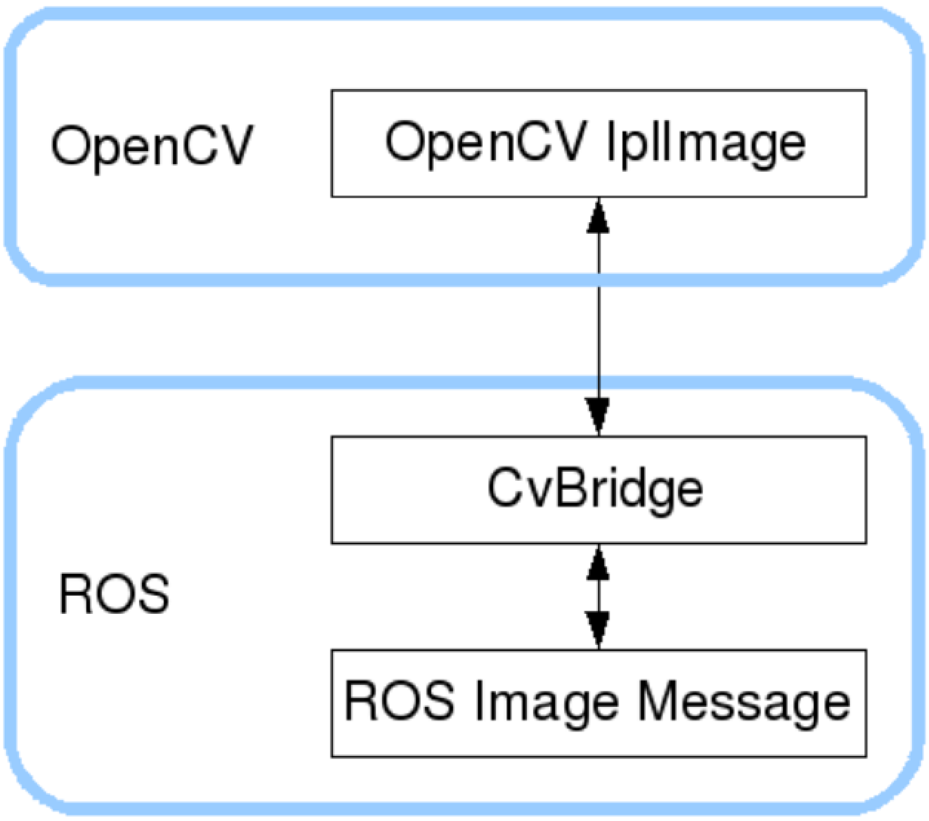
ROS+Opencv basics

1. Overview

ROS has integrated Opencv3.0 and above during the installation process, so there is almost no need to consider the installation configuration too much. ROS transmits images in its own sensor_msgs/Image message format and cannot directly process images, but the provided [CvBridge] can perfectly convert and be converted image data formats. [CvBridge] is a ROS library, which is equivalent to a bridge between ROS and Opencv.

The image data conversion between Opencv and ROS is shown in the figure below:



Although the installation configuration does not require too much consideration, the use environment still needs to be configured, mainly the two files [package.xml] and [CMakeLists.txt]. This function package not only uses [CvBridge], but also requires [Opencv] and [PCL], so it is configured together.

- package.xml

Add the following content,

```
<build_depend>sensor_msgs</build_depend>
<build_export_depend>sensor_msgs</build_export_depend>
<exec_depend>sensor_msgs</exec_depend>
<build_depend>std_msgs</build_depend>
<build_export_depend>std_msgs</build_export_depend>
<exec_depend>std_msgs</exec_depend>
<build_depend>cv_bridge</build_depend>
<build_export_depend>cv_bridge</build_export_depend>
<exec_depend>cv_bridge</exec_depend>
<exec_depend>image_transport</exec_depend>
```

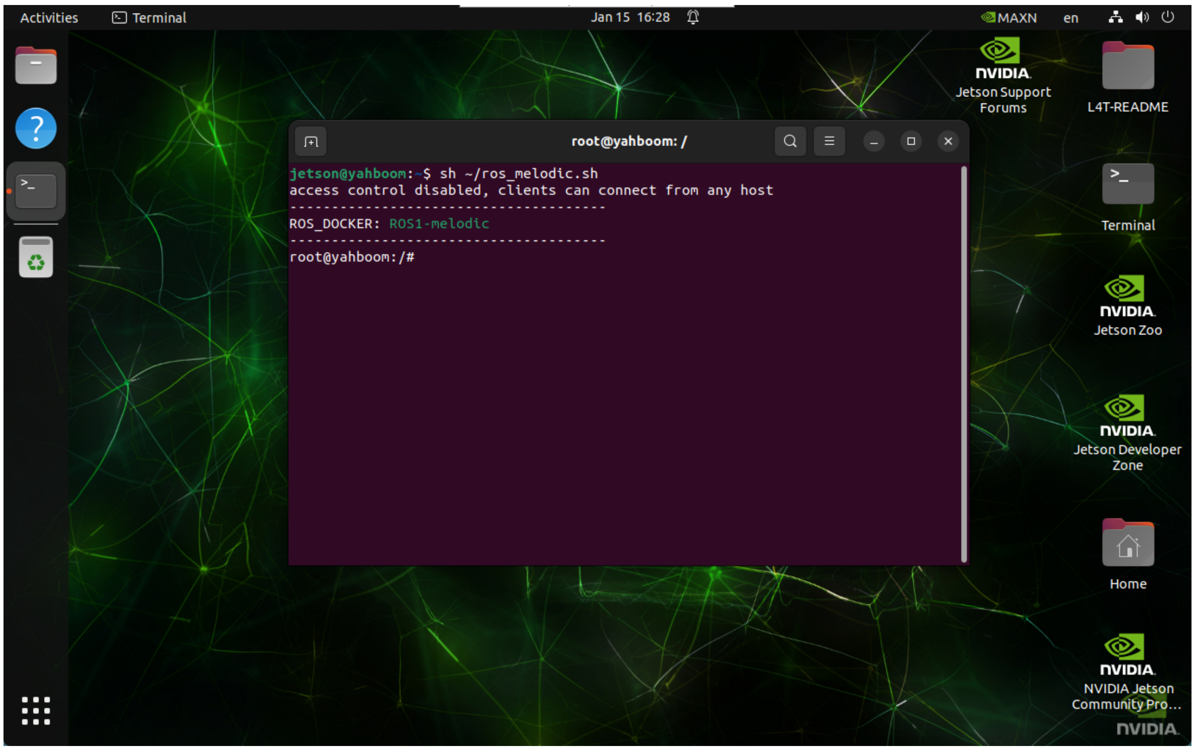
【cv_bridge】： Image conversion dependency package.

- CMakeLists.txt

This file has a lot of configuration content, please check the source file for specific content.

2. Enter Docker

```
sh ~/ros_melodic.sh
```


Activities
Terminal
Jan 15 16:28
MAXN
en
root@yahboom: /
NVIDIA Jetson Support Forums
L4T-README
Terminal
NVIDIA Jetson Zoo
NVIDIA Jetson Developer Zone
Home
NVIDIA NVIDIA Jetson Community Pro...

```
jetson@yahboom:~$ sh ~/ros_melodic.sh
access control disabled, clients can connect from any host
------------------------------------------
ROS_DOCKER: ROS1-melodic
------------------------------------------
root@yahboom:/#
```

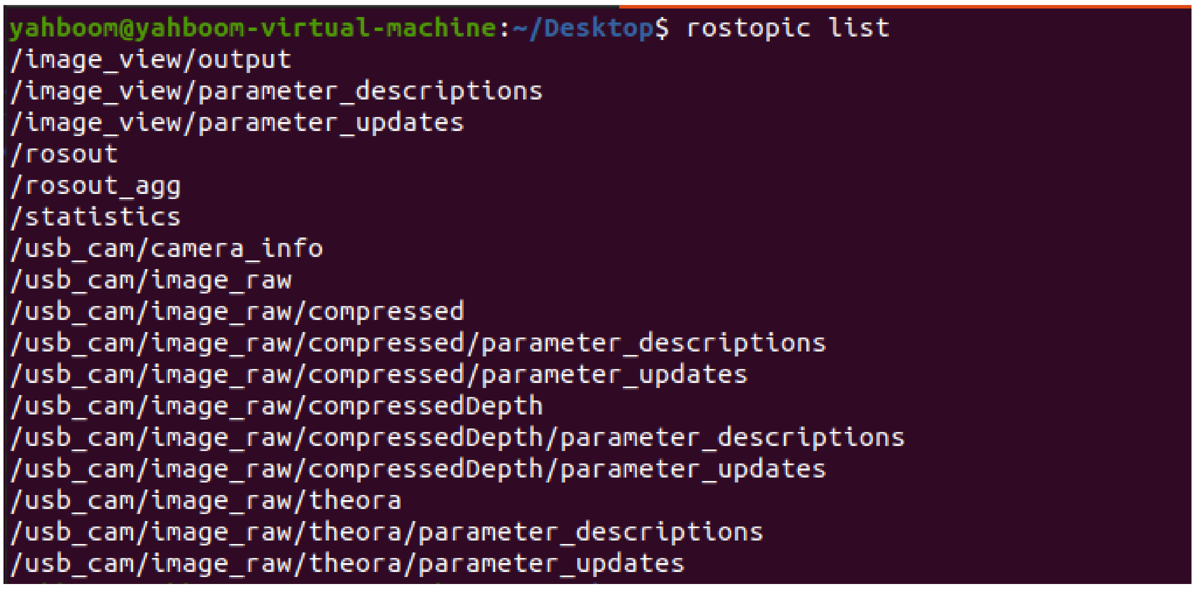
3. Start the USB camera

Terminal input,

```
roslaunch usb_cam usb_cam-test.launch
```

View topics

```
rostopic list
```

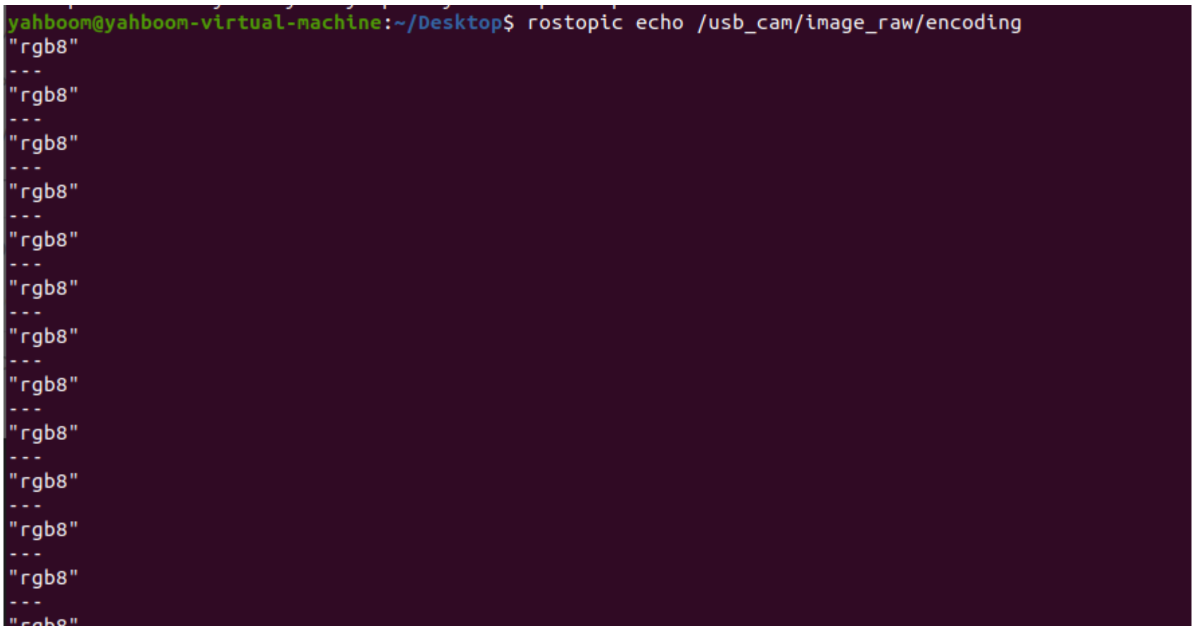


```
yahboom@yahboom-virtual-machine:~/Desktop$ rostopic list
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
/statistics
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptions
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
```

Commonly used are /usb_cam/image_raw and /usb_cam/image_raw/compressed. The former is a normal image, and the latter is a compressed image.

Check the encoding format of the topic: rostopic echo +【topic】+encoding, for example,

```
rostopic echo /usb_cam/image_raw/encoding
```



```
yahboom@yahboom-virtual-machine:~/Desktop$ rostopic echo /usb_cam/image_raw/encoding
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
```

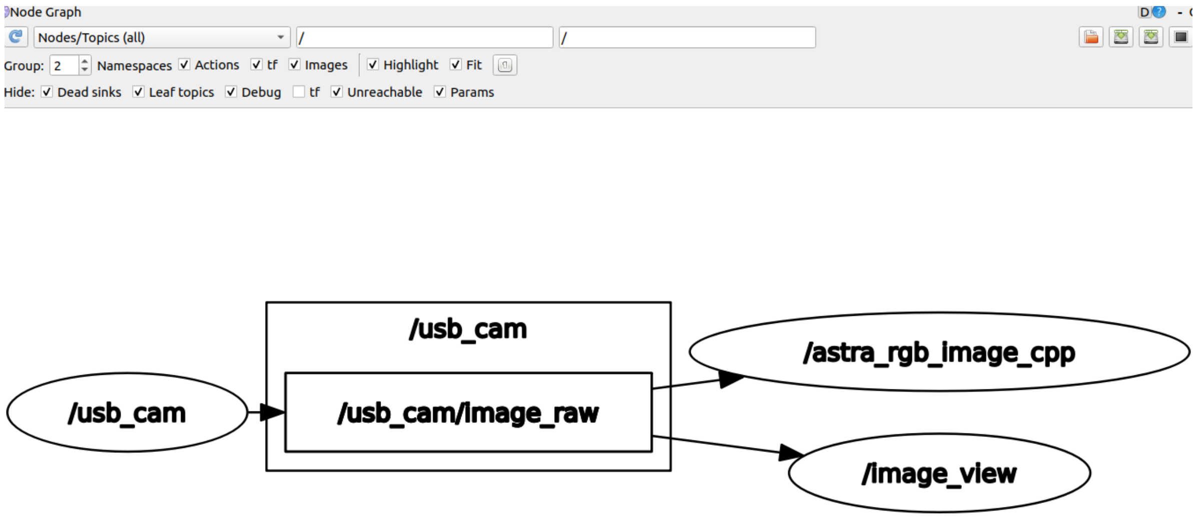
3.1, Start the color image subscription node

```
roslaunch usb_cam usb_cam-test.launch
rosrun yahboomcar_visual usb_cam_image.py
```

3.2, View the node graph

Terminal input,

```
rqt_graph
```


Node Graph

Nodes/Topics (all) | / | /

Group: 2 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit

Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug ☐ tf ✓ Unreachable ✓ Params