

# 1 ROS introduction

---

## 1 ROS introduction

- 1.1 Main features of ROS
- 1.2 the overall architecture of ROS
  - 1.2.1 Computational graph level
  - 1.2.2 file system level
  - 1.2.3 Open source community level
- 1.3 Communication Mechanism
  - 1.3.1 Topic
  - 1.3.2 Service
  - 1.3.3 Action
- 1.4 common components
- 1.5 release version

ROS wiki: <http://wiki.ros.org/>

ROS Teaching: <http://wiki.ros.org/ROS/Tutorials>

ROS installation: <http://wiki.ros.org/melodic/Installation/Ubuntu>

ROS(Robot Operating System, referred to as "ROS") is an open source operating system for robots. It provides the services expected of an operating system, including hardware abstraction, low-level device control, implementation of commonly used functions, message passing between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run code across computers.

The main goal of ROS is to provide code reuse support for robotics research and development. ROS is a framework for distributed processes(aka "nodes") that are encapsulated in packages and functionality that can be easily shared and distributed. ROS also supports a federated system similar to code repositories, which also enables project collaboration and publishing. This design enables the development and implementation of a project to be completely independent from the file system to the user interface(not restricted by ROS). At the same time, all projects can be integrated by the basic tools of ROS.

## 1.1 Main features of ROS

- (1) Distributed architecture(each worker process is regarded as a node, which is managed uniformly by the node manager),
- (2) Multi-language support(such as C++, Python, etc.),
- (3) Good scalability(one node can be written, or many nodes can be organized into a larger project through roslaunch),
- (4) Open source code(ROS follows the BSD protocol and is completely free for individual and commercial applications and modifications).

## 1.2 the overall architecture of ROS

Open source community level: It mainly includes developer knowledge, code, and algorithm sharing.

File system level: used to describe the code and executable programs that can be found on the hard disk,

Computational graph level: It reflects the communication between processes and processes, and between processes and systems.

### 1.2.1 Computational graph level

- node

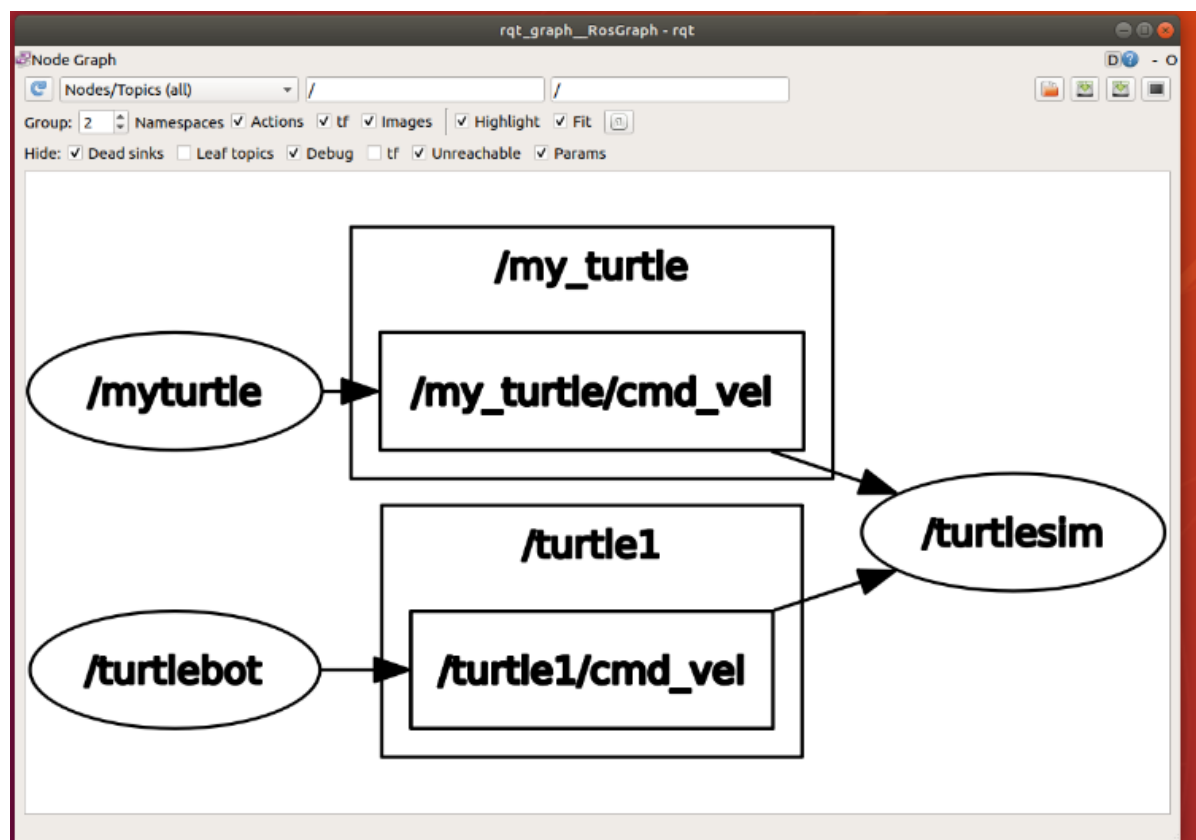
```
roscore  
roslaunch turtlesim turtlesim_node
```

The following command is the entire line, after inputting a part, use the [Tab] key to complete it, and then modify the content

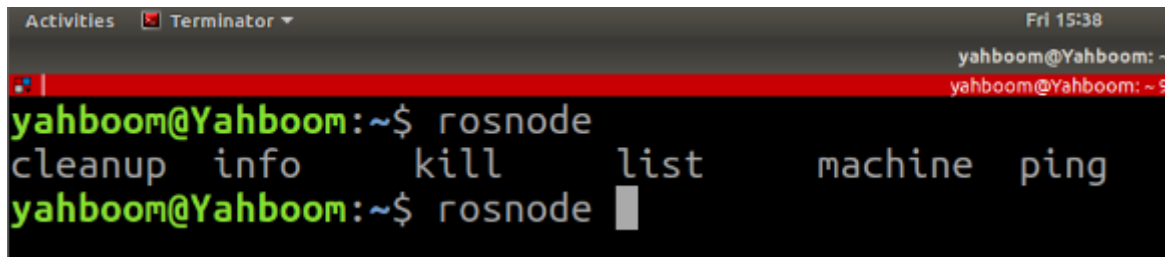
```
rosservice call /spawn "x: 3.0  
y: 3.0  
theta: 90.0  
name: 'my_turtle'"
```

Nodes are the main computational execution processes. ROS is composed of many nodes.

```
rqt_graph
```



When we enter [roscat] in the command line, then double-click `Tab` key, you will find these words appear below the command line

A terminal window titled 'Terminator' showing a user named 'yahboom@Yahboom' at a prompt '~\$'. The user has entered 'roscat' and pressed the Tab key twice, which has triggered a completion list. The list shows the following options: 'cleanup', 'info', 'kill', 'list', 'machine', and 'ping'. The cursor is currently positioned at the end of the 'roscat' command.

ROS command line tool `roscat` :

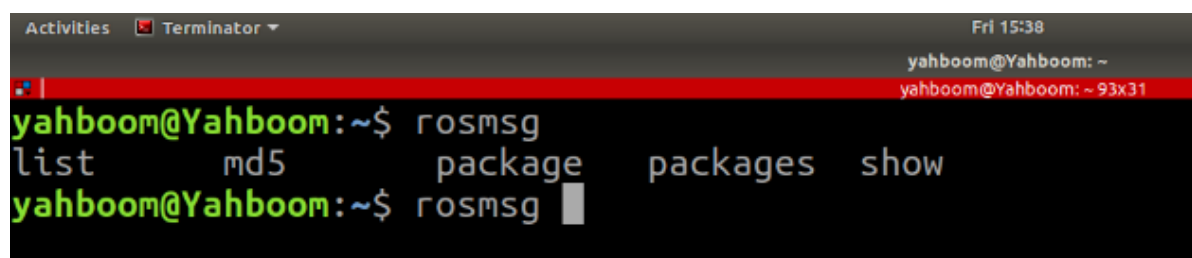
roscat command	effect
roscat list	Query all currently running nodes
roscat info node_name	Show details of the node
roscat kill node_name	end a node
roscat ping	Test if the node is alive
roscat machine	List nodes running on a specific machine or a list of machines
roscat cleanup	Clear the registration information of non-runnable nodes

When developing and debugging, you often need to view the current node and node information, so please remember these common commands. If you can't remember, you can also pass `roscat help` to view `roscat` usage of the command.

- information

The nodes realize logical connection and data exchange with each other through messages.

When we enter [roscat] in the command line, then double-click `Tab` key, you will find these words appear below the command line

A terminal window titled 'Terminator' showing a user named 'yahboom@Yahboom' at a prompt '~\$'. The user has entered 'roscat' and pressed the Tab key twice, which has triggered a completion list. The list shows the following options: 'list', 'md5', 'package', 'packages', and 'show'. The cursor is currently positioned at the end of the 'roscat' command.

ROS command line tool `roscat` :

roscat command	effect
roscat show	Show a message field
roscat list	list all messages
roscat package	List all feature pack messages
roscat packages	List all feature packs with this message
roscat md5	Displays the MD5 check value of a message

- topic

Topics are a way of delivering messages(pub/sub). Each message is published to the corresponding topic, and each topic is strongly typed.

ROS topic messages can be transmitted using TCP/IP or UDP. The default transmission method used by ROS is TCP/IP. TCP-based transmission becomes TCPROS, which is a long connection method; UDP-based transmission becomes UDPROS, which is a low-latency and high-efficiency transmission method, but it is easy to lose data and is suitable for remote operation.

When we enter [rostopic] in the command line, then double-click `Tab` key, you will find these words appear below the command line

```

Activities Terminator FRI 15:38
yahboom@Yahboom: ~
yahboom@Yahboom: ~ 93x31
yahboom@Yahboom:~$ rostopic
bw    echo  find  hz    info  list  pub   type
yahboom@Yahboom:~$ rostopic

```

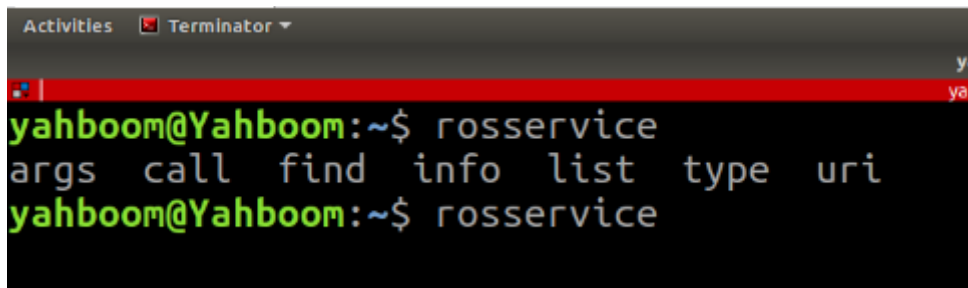
**ROS command line tool rostopic :**

rostopic command	effect
rostopic bw/topic	Displays the bandwidth used by the theme
rostopic echo/topic	Output the message corresponding to the topic to the screen
rostopic find message_type	Find topics by type
rostopic hz/topic	Shows how often a topic is published
rostopic info/topic	output information about the topic
rostopic list	Output list of active topics
rostopic pub /topic type args	Publish data to topic
rostopic type/topic	output topic type

- Services

Services are used in the request-response model and must also have a unique name. When a node provides a service, all nodes can communicate with it through code written using the ROS client.

When we enter [rosservice] in the command line, then double-click `Tab` key, you will find these words appear below the command line

A terminal window titled 'Terminator' showing the command 'rosservice' followed by its subcommands: 'args', 'call', 'find', 'info', 'list', 'type', and 'uri'. The prompt is 'yahboom@Yahboom:~\$'.

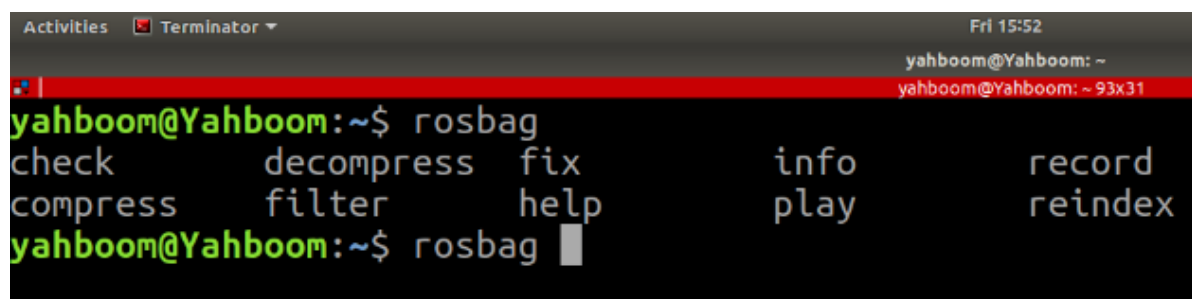
ROS command line tool rosservice :

rosservice command	effect
rosservice args /service	show service parameters
rosservice call /service	Request service with input parameters
rosservice find /service	Find topics by type
rosservice info /service	Display information about the specified service
rosservice list	Display active service information
rosservice uri /service	show ROSRPC URI service
rosservice type /service	Show service type

- message logging package

A message record bag is a file format for saving and replaying ROS message data, saved in a .bag file. is an important mechanism for storing data.

When we enter [rosvbag] in the command line, then double-click `Tab` key, you will find these words appear below the command line

A terminal window titled 'Terminator' showing the command 'rosvbag' followed by its subcommands: 'check', 'decompress', 'fix', 'info', 'record', 'compress', 'filter', 'help', 'play', and 'reindex'. The prompt is 'yahboom@Yahboom:~\$'.

ROS command line tool rosvbag :

<b>rosvbag command</b>	<b>effect</b>
check	Determine if a package can be made in the current system, or if it can be migrated.
decompress	Compress one or more package files.
filter	Extract one or more package files.
fix	Fix message in package file to play on current system.
help	Get help information on related command guidelines.
info	Summarize the contents of one or more package files.
play	Play back the contents of one or more package files in a time-synchronized manner.
record	Record a package file with the contents of the specified topic.
reindex	Reindex one or more package files.

- parameter server

The parameter server is a shared multivariable dictionary accessible over the network, stored on the node manager by key.

When we enter [rosvparam] in the command line, then double-click Tab key, you will find these words appear below the command line

```

Activities Terminator
Fri 16:04
yahboom@Yahboom: ~
yahboom@Yahboom: ~ 93x5
yahboom@Yahboom:~$ rosvparam
delete dump get list load set
yahboom@Yahboom:~$ rosvparam

```

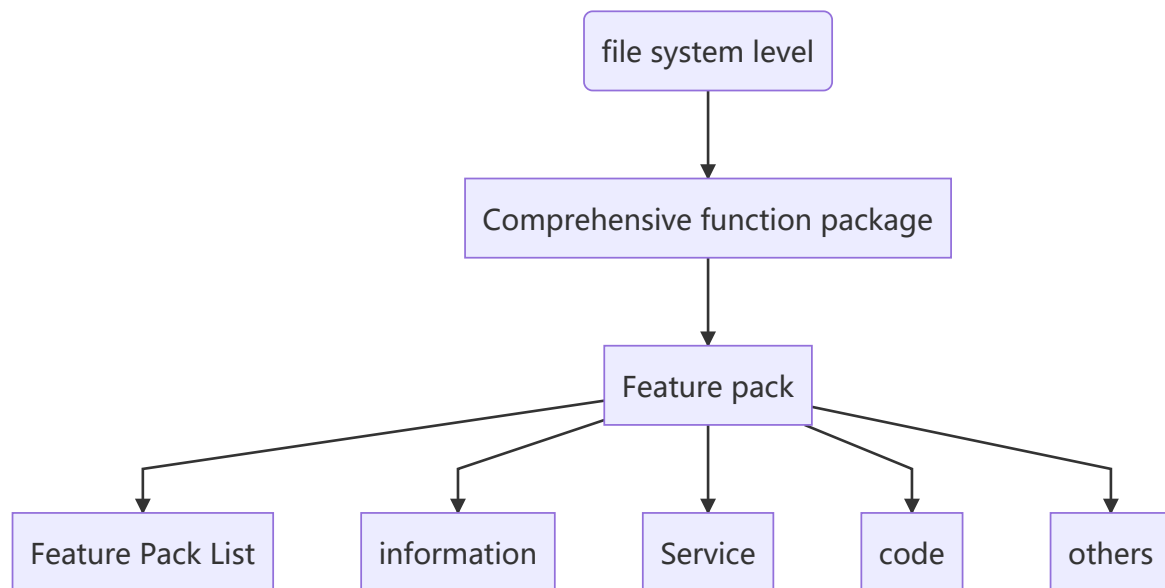
**ROS command line tool rosvparam :**

<b>rosvparam command</b>	<b>effect</b>
rosvparam delete parameter	delete parameter
rosvparam dump file	Write the parameters from the parameter server to a file
rosvparam get parameter	get parameter value
rosvparam list	List parameters in parameter server
rosvparam load file	Load parameters from file to parameter server
rosvparam set parameter value	Setting parameters

- Node Manager(Master)

The node manager is used for registration and lookup of topics, service names, etc. In the entire ROS system, if there is no node manager, there will be no communication between nodes.

## 1.2.2 file system level



Dependencies can be configured between feature packages. If function package A depends on function package B, then when the ROS build system is used, B must be built earlier than A, and A can use the header files and library files in B.

The concepts at the file system level are as follows:

- Feature pack list:

This list is to indicate the dependencies of the function package, source file compilation flag information, etc. The package.xml file in the function package is a list of function packages.

- Feature pack:

The function package is the basic form of software organization in the ROS system, including running nodes and configuration files.

### ROS package related commands

rospack command	effect
rospack help	show usage of rospack
rospack list	List all packages on this machine
rospack depends [package]	Show package dependencies
rospack find [package]	locate a package
rospack profile	Refresh location records for all packages

- Comprehensive function package

By grouping several feature packages together, a comprehensive feature package can be formed.

- message type

When sending messages between nodes in ROS, message descriptions need to be made in advance. Standard types of messages are provided in ROS, or you can define them yourself. The description of the message type is stored in the msg file under the function package.

- Service type

Defines the data structures for service requests and responses provided by each process in the ROS system.

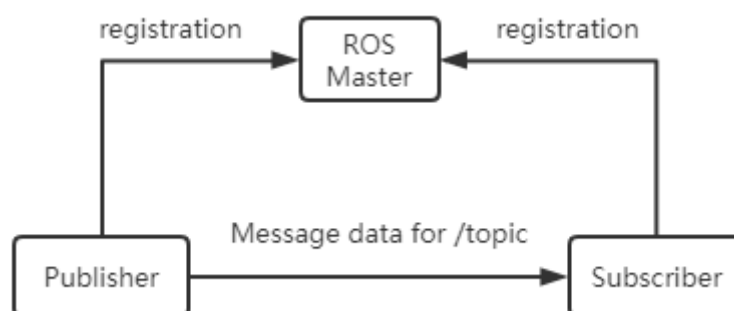
### 1.2.3 Open source community level

- Distribution: A ROS distribution is a series of comprehensive function packages with version numbers that can be installed independently. ROS distributions play a similar role as Linux distributions. This makes ROS software installation easier and maintains a consistent version through a software collection.
- Repository: ROS relies on a website or hosting service that shares open source code and software repositories, where different organizations can publish and share their own robotics software and programs.
- ROS Wiki: The ROS Wiki is the main forum for documenting information about the ROS system. Anyone can sign up for an account, contribute their own files, provide corrections or updates, write tutorials, and more.
- Bug Ticket System: If you find a problem or want to propose a new feature, ROS provides this resource to do it.
- Mailing list: The ROS user mailing list is the main communication channel about ROS, which can exchange various questions or information from ROS software updates to the use of ROS software like a forum.
- ROS Answer: Users can use this resource to ask questions

## 1.3 Communication Mechanism

### 1.3.1 Topic

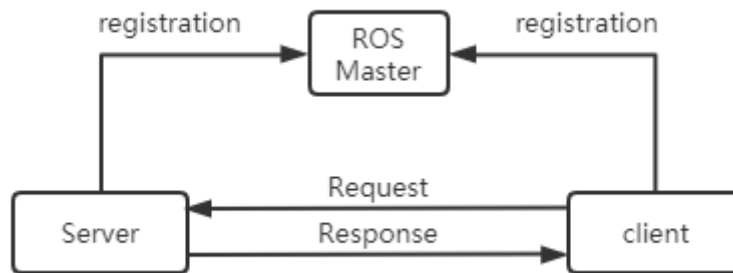
The asynchronous publish-subscribe communication mode is widely used in ros. Topic is generally used for one-way, message flow communication. Topic generally has a strong type definition: a type of topic can only receive/send messages of a specific data type. Publisher is not required to have type consistency, but subscriber will check the md5 of the type and report an error when accepting it.



### 1.3.2 Service

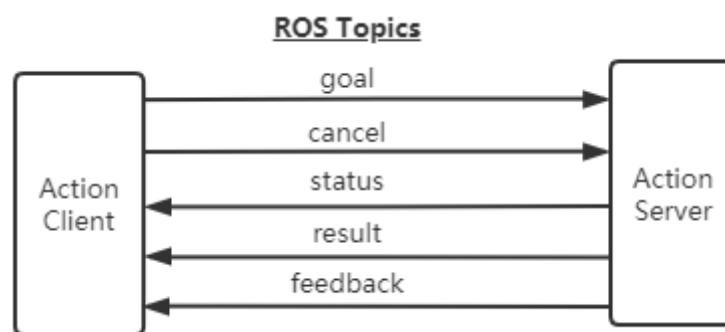
service is used to handle synchronous communication in ros communication, using server/client semantics. Each service type has two parts: request and response. For the server in the service, ros will not check the name conflict. Only the last registered server will take effect and establish a connection with the client.





### 1.3.3 Action

Action is composed of multiple topics to define tasks. Task definitions include goals(Goal), task execution process status feedback(Feedback), and results(Result). Compiling action will automatically generate seven structures: Action, ActionGoal, ActionFeedback, ActionResult, Goal, Feedback, and Result structures.



Features of Action:

- A question and answer communication mechanism
- with continuous feedback
- Can be terminated during a task
- ROS-based message mechanism implementation

Action interface:

- goal: publish the task goal
- cancel: request to cancel the task
- status: notify the client of the current status
- feedback: Periodic feedback task running monitoring data
- result: Send the execution result of the task to the client, only published once.

Communication Mode Features Comparison

Features	Topic	Service	Action
response mechanism	none	result response	progress response, result response
synchronicity	asynchronous	Synchronize	asynchronous
communication model	Publisher, Subscriber	Client, Server	Client, Server
Node correspondence	many-to-many	Many(Client) to one(Server)	Many(Client) to one(Server)

## 1.4 common components

launch startup file; TF coordinate transformation; Rviz; Gazebo; QT toolbox; Navigation; Moveit!

launch: Launch File is a way to start multiple nodes at the same time in ROS. It can also automatically start the ROS Master node manager, and can realize various configurations of each node, providing the operation of multiple nodes. Great convenience.

TF coordinate transformation: There are often a large number of component elements in the robot body and the working environment of the robot. The positions and attitudes of different components are involved in robot design and robot applications. TF is a function package that allows users to track multiple coordinate systems over time., which uses a tree data structure, buffers and maintains the coordinate transformation relationship between multiple coordinate systems according to time, and can help developers to complete the transformation of coordinates such as points and vectors between coordinate systems at any time.

QT Toolbox: In order to facilitate visual debugging and display, ROS provides a background graphics tool suite of the Qt architecture - `rqt_common_plugins`, which contains many utilities: log output tool(`rqt_console`), computational graph visualization tool(`rqt_graph`), data drawing tool(`rqt_plot`), parameter dynamic configuration tool(`rqt_reconfigure`)

Rviz: rviz is a 3D visualization tool that is well compatible with various robot platforms based on the ROS software framework. In rviz, you can use XML to describe the size, mass, position, material, joints and other attributes of any physical object such as robots and surrounding objects, and present them in the interface. At the same time, rviz can also display the information of the robot's sensors, the motion state of the robot, and the changes of the surrounding environment in real time through a graphical method.

Gazebo: Gazebo is a powerful 3D physics simulation platform with a powerful physics engine, high-quality graphics rendering, convenient programming and graphics interfaces, and most importantly, it is open source and free. Although the robot model in Gazebo is the same as that used by rviz, the physical properties of the robot and surrounding environment, such as mass, friction coefficient, elasticity coefficient, etc., need to be added to the model. The sensor information of the robot can also be added to the simulation environment in the form of plug-ins and displayed in a visual way.



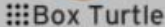
Navigation: Navigation is a two-dimensional navigation function package of ROS. In simple terms, it is based on the input information flow of sensors such as odometer and the global position of the robot, through the navigation algorithm, calculates the safe and reliable robot speed control command.

Moveit: Moveit!Feature packages are the most commonly used toolkits and are mainly used for trajectory planning. Moveit!The configuration assistant is used to configure some files that need to be used in planning, which is very important.

## 1.5 release version

Reference link: <http://wiki.ros.org/Distributions>

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015
ROS Groovy Galapagos	December 31, 2012			July, 2014
ROS Fuerte Turtle	April 23, 2012			--
ROS Electric Emys	August 30, 2011			--
ROS Diamondback	March 2, 2011			--

ROS C Turtle	August 2, 2010			--
ROS Box Turtle	March 2, 2010			--
				

ROS distribution(ROS distribution) refers to the version of the ROS package, which [Linux](#) distribution(such as [Ubuntu](#)) is similar in concept to a The purpose of a ROS distribution is to allow developers to use a relatively stable codebase until they are ready to version everything. Therefore, after each release, ROS developers usually only fix bugs for that release, while providing a small number of improvements to the core package. As of October 2019, the version name, release time and version life cycle of the major releases of ROS are shown in the following table:

version name	release date	Version life cycle	Operating system platform
ROS Noetic Ninjemys	May 2020	May 2025	Ubuntu 20.04
ROS Melodic Morenia	May 23, 2018	May 2023	Ubuntu 17.10, Ubuntu 18.04, Debian 9, Windows 10
ROS Lunar Loggerhead	May 23, 2017	May 2019	Ubuntu 16.04, Ubuntu 16.10, Ubuntu 17.04, Debian 9
ROS Kinetic Kame	May 23, 2016	April 2021	Ubuntu 15.10, Ubuntu 16.04, Debian 8
ROS Jade Turtle	May 23, 2015	May 2017	Ubuntu 14.04, Ubuntu 14.10, Ubuntu 15.04
ROS Indigo Igloo	July 22, 2014	April 2019	Ubuntu 13.04, Ubuntu 14.04
ROS Hydro Medusa	September 4, 2013	May 2015	Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04
ROS Groovy Galapagos	December 31, 2012	July 2014	Ubuntu 11.10, Ubuntu 12.04, Ubuntu 12.10
ROS Strong Turtle	April 23, 2012	--	Ubuntu 10.04, Ubuntu 11.10, Ubuntu 12.04
ROS Electric Emys	August 30, 2011	--	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04, Ubuntu 11.10
ROS Diamondback	March 2, 2011	--	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04
ROS C Turtle	August 2, 2010	--	Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04, Ubuntu 10.10
ROS Box Turtle	March 2, 2010	--	Ubuntu 8.04, Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04