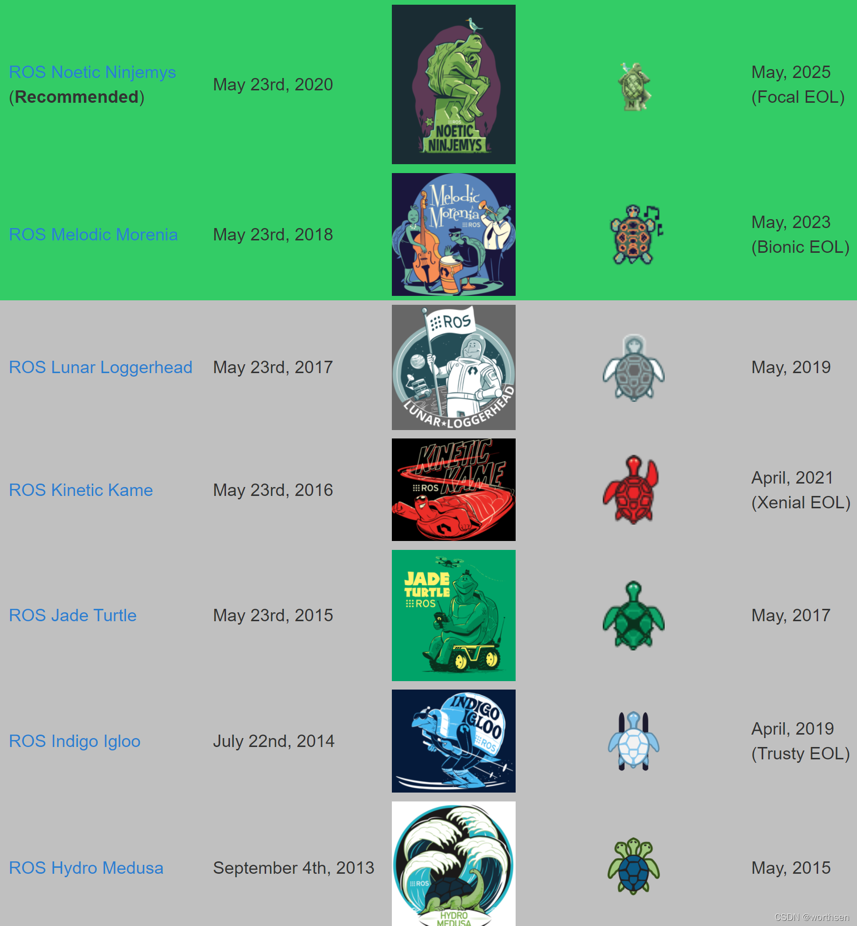
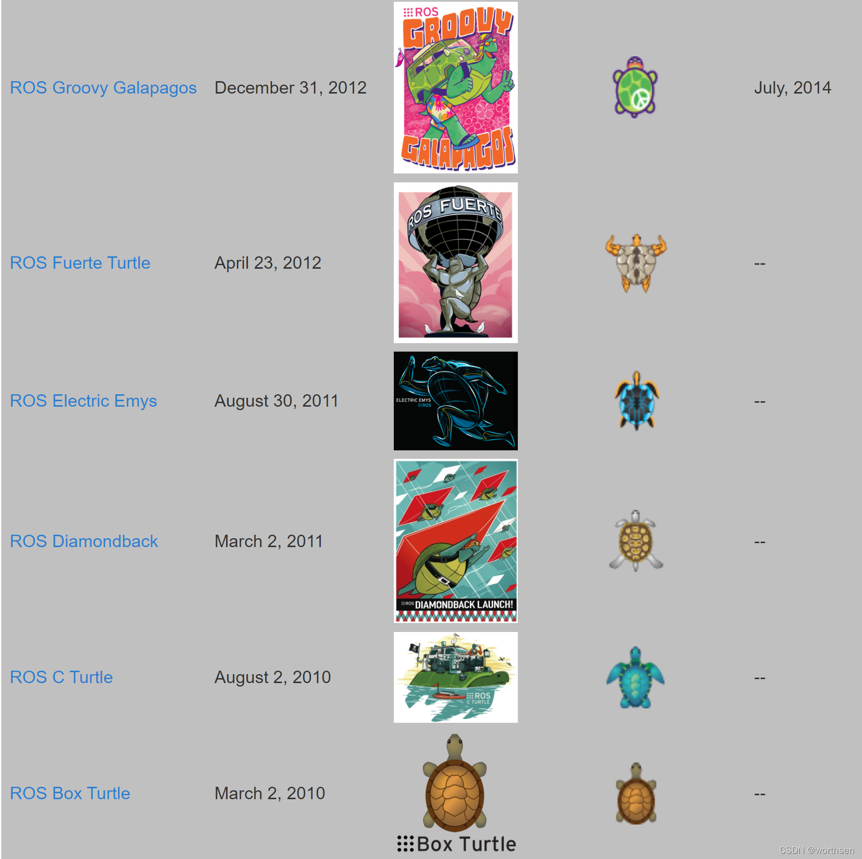
ROS

ros1版本





ros2版本

| **Distro** | **Release date** | **Logo** | **EOL date** |
| --- | --- | --- | --- |
| [Humble Hawksbill](https://docs.ros.org/en/rolling/Releases/Release-Humble-Hawksbill.html) | May 23rd, 2022 | Humble logo | May 2027 |
| [Galactic Geochelone](https://docs.ros.org/en/rolling/Releases/Release-Galactic-Geochelone.html) | May 23rd, 2021 | Galactic logo | November 2022 |
| [Foxy Fitzroy](https://docs.ros.org/en/rolling/Releases/Release-Foxy-Fitzroy.html) | June 5th, 2020 | Foxy logo | May 2023 |
| [Eloquent Elusor](https://docs.ros.org/en/rolling/Releases/Release-Eloquent-Elusor.html) | November 22nd, 2019 | Eloquent logo | November 2020 |
| [Dashing Diademata](https://docs.ros.org/en/rolling/Releases/Release-Dashing-Diademata.html) | May 31st, 2019 | Dashing logo | May 2021 |
| [Crystal Clemmys](https://docs.ros.org/en/rolling/Releases/Release-Crystal-Clemmys.html) | December 14th, 2018 | Crystal logo | December 2019 |
| [Bouncy Bolson](https://docs.ros.org/en/rolling/Releases/Release-Bouncy-Bolson.html) | July 2nd, 2018 | Bouncy logo | July 2019 |
| [Ardent Apalone](https://docs.ros.org/en/rolling/Releases/Release-Ardent-Apalone.html) | December 8th, 2017 | Ardent logo | December 2018 |
| [beta3](https://docs.ros.org/en/rolling/Releases/Beta3-Overview.html) | September 13th, 2017 |  | December 2017 |
| [beta2](https://docs.ros.org/en/rolling/Releases/Beta2-Overview.html) | July 5th, 2017 |  | September 2017 |
| [beta1](https://docs.ros.org/en/rolling/Releases/Beta1-Overview.html) | December 19th, 2016 |  | Jul 2017 |
| [alpha1 - alpha8](https://docs.ros.org/en/rolling/Releases/Alpha-Overview.html) | August 31th, 2015 |  | December 2016 |

ROS1与ROS2区别

4.1 更稳定、应用更广

ROS1最初由Willow Garage于2007年创建，在开源机器人社区中已经变得巨大。

ROS1背后的团队已经了解到 - 凭借这些年的经验，缺少哪些重要功能以及可以改进的内容。不幸的是，将所有这些修改添加到ROS1中将需要许多重大更改，并使ROS1非常不稳定。因此，ROS2是从头开始开发的，并且是一个全新的ROS。

至于现在ROS在业界还不是很受欢迎，并且缺乏一些最重要的要求，比如实时、安全、认证、安保。ROS2 的目标之一是使其与工业应用兼容。

4.2 接口更统一

在 ROS1 中，对于 Cpp，你使用 roscpp，而对于 Python，你使用 rospy。这两个库都是完全独立的，从头开始构建。有些功能接口不统一。

ROS2 具有更多层。只有一个名为 rcl 的基本库，它是用 C 语言实现的。这是包含所有 ROS2 核心功能的基础。在此基础上构建Cpp和Python接口，底层接口统一。

4.3 使用 OOP 写入节点

在 ROS1 中，没有特定的结构告诉您应如何编写节点功能。您可以决定在程序中的任何位置添加回调函数，或者根据需要使用OOP，但每个函数的实现都可以是唯一的。

在 ROS2 中，情况有所不同。有一个关于如何编写节点的约定。您必须创建一个继承自 Node 对象的类。这将为每个人节省大量时间。您已经拥有了一个良好的模块化结构来编写节点。这将使您的程序更干净，开发人员在不同项目上的合作将更容易。

4.4 集中式到分布式

ROS1 中，在运行节点之前，需要启动 ROS 主节点。ROS1 主节点将充当节点的 DNS 服务器，以便它们可以相互检索。

ROS2 中，不再有 ROS 主节点！这不再是一个集中式系统。每个节点都具有发现其他节点的能力。可以简单地启动一个节点，而不必担心是否有主节点正在运行。可以创建完全分布式系统。

4.5 服务由同步到异步

在 ROS1 中，服务是同步的。当服务客户端向服务器请求请求时，该请求将一直卡住，直到服务器响应（或失败）。

在 ROS2 中，服务是异步的。调用服务时，可以添加一个回调函数，该函数将在服务器响应时触发。同时，您的主线程不会卡住。

参考

1、http://wiki.ros.org/Distributions

2、https://docs.ros.org/en/rolling/Releases.html

3、https://roboticsbackend.com/ros1-vs-ros2-practical-overview/

4、ros代码仓库