# Using `ros2doctor` to identify issues

**Goal:** Identify issues in your ROS 2 setup using the `ros2doctor` tool.

**Tutorial level:** Beginner

**Time:** 10 minutes

**Contents**



## Background

When your ROS 2 setup is not running as expected, you can check its settings with the `ros2doctor` tool.

`ros2doctor` checks all aspects of ROS 2, including platform, version, network, environment, running systems and more, and warns you about possible errors and reasons for issues.

## Prerequisites

`ros2doctor` is part of the `ros2cli` package. As long as you have `ros2cli` installed (which any normal install should have), you will be able to use `ros2doctor`.

This tutorial uses turtlesim to illustrate some of the examples.

# Tasks

## 1 Check your setup

Let's examine your general ROS 2 setup as a whole with `ros2doctor`. First, source ROS 2 in a new terminal, then enter the command:

```
ros2 doctor
```

This will conduct checks over all your setup modules and return warnings and errors.

If your ROS 2 setup is in perfect shape, you'll see a message similar to this:

```
All <n> checks passed
```

However, it's not unusual to have a few warnings returned. A `UserWarning` doesn't mean your setup is unusable; it's more likely just an indication that something is configured in a way that's not ideal.

If you do receive a warning, it will look something like this:

```
<path>: <line>: UserWarning: <message>
```

For example, `ros2doctor` will find this warning if you're using an unstable ROS 2 distribution:

```
UserWarning: Distribution <distro> is not fully supported or tested. To get more consistent
features, download a stable version at https://index.ros.org/doc/ros2/Installation/
```

If `ros2doctor` only finds warnings in your system, you will still receive the `All <n> checks passed` message.

Most checks are categorized as warnings as opposed to errors. It's mostly up to you, the user, to determine the importance of the feedback `ros2doctor` returns. If it does find a rare error in your setup, indicated by `UserWarning: ERROR:`, the check is considered failed.

You will see a message similar to the following list of issue feedback:

```
1/3 checks failed

Failed modules:  network
```

An error indicates the system is missing important settings or functions that are crucial to ROS 2. Errors should be addressed to ensure the system functions properly.

## 2 Check a system

You can also examine a running ROS 2 system to identify possible causes for issues. To see `ros2doctor` working on a running system, let's run turtlesim, which has nodes actively communicating with each other.

Start up the system by opening a new terminal, sourcing ROS 2, and entering the command:

```
ros2 run turtlesim turtlesim_node
```

Open another terminal and source ROS 2 to run the teleop controls:

```
ros2 run turtlesim turtle_teleop_key
```

Now run `ros2doctor` again in its own terminal. You will see the warnings and errors you had the last time you ran `ros2doctor` on your setup if you had any. Following those will be a couple new warnings relating to the system itself:

```
UserWarning: Publisher without subscriber detected on /turtle1/color_sensor.
UserWarning: Publisher without subscriber detected on /turtle1/pose.
```

It seems that the `/turtlesim` node publishes data to two topics that aren't being subscribed to, and `ros2doctor` thinks this could possibly lead to issues.

If you run commands to echo the `/color_sensor` and `/pose` topics, those warnings will disappear because the publishers will have subscribers.

You can try this by opening two new terminals while turtlesim is still running, sourcing ROS 2 in each, and running each of the following commands in their own terminal:

```
ros2 topic echo /turtle1/color_sensor
```

```
ros2 topic echo /turtle1/pose
```

Then run `ros2doctor` in its terminal again. The `publisher without subscriber` warnings will be gone. (Make sure to enter `Ctrl+C` in the terminals where you ran `echo`).

Now try exiting either the turtlesim window or quitting the teleop and running `ros2doctor` again. You'll see more warnings indicating `publisher without subscriber` or `subscriber without publisher` for different topics, now that one node in the system isn't available.

In a complex system with many nodes, `ros2doctor` would be invaluable for identifying possible reasons for communication issues.

## 3 Get a full report

While `ros2doctor` will let you know warnings about your network, system, etc., running it with the `--report` argument will give you much more detail to help you analyze issues.

You might want to use `--report` if you get a warning about your network setup and want to find out exactly what part of your configuration is causing the warning.

It's also very helpful when you need to open a support ticket to get help with ROS 2. You can copy and paste the relevant parts of your report into the ticket so the people helping you can better understand your environment and provide better assistance.

To get a full report, enter the following command in the terminal:

```
ros2 doctor --report
```

Which will return a list of information categorized into five groups:

```
NETWORK CONFIGURATION
...

PLATFORM INFORMATION
...

RMW MIDDLEWARE
...

ROS 2 INFORMATION
...

TOPIC LIST
...
```

You can crosscheck the information here against the warnings you get when running `ros2 doctor`. For example, if `ros2doctor` returned the warning (mentioned earlier) that your distribution is “not fully supported or tested”, you might take a look at the `ROS 2 INFORMATION` section of the report:

```
distribution name      : <distro>
distribution type      : ros2
distribution status    : prerelease
release platforms      : {'<platform>': ['<version>']}
```

Here you can see the `distribution status` is `prerelease`, which explains why it’s not fully supported.

## Summary

`ros2doctor` will inform you of problems in your ROS 2 setup and running systems. You can get a deeper look at information behind those warnings by using the `--report` argument.

Keep in mind, `ros2doctor` is not a debug tool; it won’t help with errors in your code or on the implementation side of your system.

## Related content

ros2doctor’s README will tell you more about different arguments. You might want to take a look around the `ros2doctor` repo as well, since it’s fairly beginner friendly and a great place to get started with contributing.

# Next steps

You've completed the beginner level tutorials!