Source installation

Prerequisites

- OS
  - Ubuntu 22.04
- ROS
  - ROS 2 Humble

  For ROS 2 system dependencies, refer to REP-2000.
- Git
  - Registering SSH keys to GitHub is preferable.

```
sudo apt-get -y update
sudo apt-get -y install git
```

> Note: If you wish to use ROS 2 Galactic on Ubuntu 20.04, refer to installation instruction from galactic branch, but be aware that Galactic version of Autoware might not have the latest features.

How to set up a development environment

1. Clone `autowarefoundation/autoware` and move to the directory.

   ```
   git clone https://github.com/autowarefoundation/autoware.git
   cd autoware
   ```

2. If you are installing Autoware for the first time, you can automatically install the dependencies by using the provided Ansible script.

   ```
   ./setup-dev-env.sh
   ```

   If you encounter any build issues, please consult the Troubleshooting section for assistance.

Info

Before installing NVIDIA libraries, please ensure that you have reviewed and agreed to the licenses.

- CUDA
- cuDNN
- TensorRT

Note

The following items will be automatically installed. If the ansible script doesn't work or if you already have different versions of dependent libraries installed, please install the following items manually.

- Install Ansible
- Install Build Tools
- Install Dev Tools
- Install gdown
- Install geographiclib
- Install pacmod
- Install the RMW Implementation
- Install ROS 2
- Install ROS 2 Dev Tools
- Install Nvidia CUDA
- Install Nvidia cuDNN and TensorRT
- Install the Autoware RViz Theme (only affects Autoware RViz)
- Download the Artifacts (for perception inference)

How to set up a workspace

Using Autoware Build GUI

If you prefer a graphical user interface (GUI) over the command line for launching and managing your simulations, refer to the `Using Autoware Build GUI` section at the end of this document for a step-by-step guide.

1. Create the `src` directory and clone repositories into it.

   Autoware uses vcstool to construct workspaces.

   ```
   cd autoware
   mkdir src
   vcs import src < autoware.repos
   ```

   If you are an active developer, you may also want to pull the nightly repositories, which contain the latest updates:

   ```
   vcs import src < autoware-nightly.repos
   ```

   > ⚠ Note: The nightly repositories are unstable and may contain bugs. Use them with caution.

   Optionally, you may also download the extra repositories that contain drivers for specific hardware, but they are not necessary for building and running Autoware:

   ```
   vcs import src < extra-packages.repos
   ```

2. Install dependent ROS packages.

   Autoware requires some ROS 2 packages in addition to the core components. The tool `rosdep` allows an automatic search and installation of such dependencies. You might need to run `rosdep update` before `rosdep install`.

   ```
   source /opt/ros/humble/setup.bash
   # Make sure all previously installed ros-$ROS_DISTRO-* packages are upgraded to their latest version
   sudo apt update && sudo apt upgrade
   rosdep update
   rosdep install -y --from-paths src --ignore-src --rosdistro $ROS_DISTRO
   ```

3. Install and set up ccache to speed up consecutive builds. *(optional but highly recommended)*

4. Build the workspace.

   Autoware uses colcon to build workspaces. For more advanced options, refer to the documentation.

   ```
   colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
   ```

   If there is any build issue, refer to Troubleshooting.

5. Follow the steps in Network Configuration before running Autoware.

6. Apply the settings recommended in Console settings for ROS 2 for a better development experience. *(optional)*

How to update a workspace

1. Update the `.repos` file.

   ```
   cd autoware
   git pull <remote> <your branch>
   ```

   `<remote>` is usually `git@github.com:autowarefoundation/autoware.git`

2. Update the repositories.

   ```
   vcs import src < autoware.repos
   ```

   > ⚠ If you are using nightly repositories, you can also update them.
   >
   > ```
   > vcs import src < autoware-nightly.repos
   > ```

   ```
   vcs pull src
   ```

   For Git users: - `vcs import` is similar to `git checkout`. - Note that it doesn't pull from the remote. - `vcs pull` is similar to `git pull`. - Note that it doesn't switch branches.

   For more information, refer to the official documentation.

   It might be the case that dependencies imported via `vcs import` have been moved/removed. VCStool does not currently handle those cases, so if builds fail after `vcs import`, cleaning and re-importing all dependencies may be necessary:

   ```
   rm -rf src/*
   vcs import src < autoware.repos
   ```

   > ⚠ If you are using nightly repositories, import them as well.
   >
   > ```
   > vcs import src < autoware-nightly.repos
   > ```

3. Install dependent ROS packages.

   ```
   source /opt/ros/humble/setup.bash
   # Make sure all previously installed ros-$ROS_DISTRO-* packages are upgraded
   ```

```
to their latest version
sudo apt update && sudo apt upgrade
rosdep update
rosdep install -y --from-paths src --ignore-src --rosdistro $ROS_DISTRO
```

4. Build the workspace.

```
colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release
```

Using Autoware Build GUI

In addition to the traditional command-line methods of building Autoware packages, developers and users can leverage the Autoware Build GUI for a more streamlined and user-friendly experience. This GUI application simplifies the process of building and managing Autoware packages.

Integration with Autoware source installation

When using the Autoware Build GUI in conjunction with the traditional source installation process:

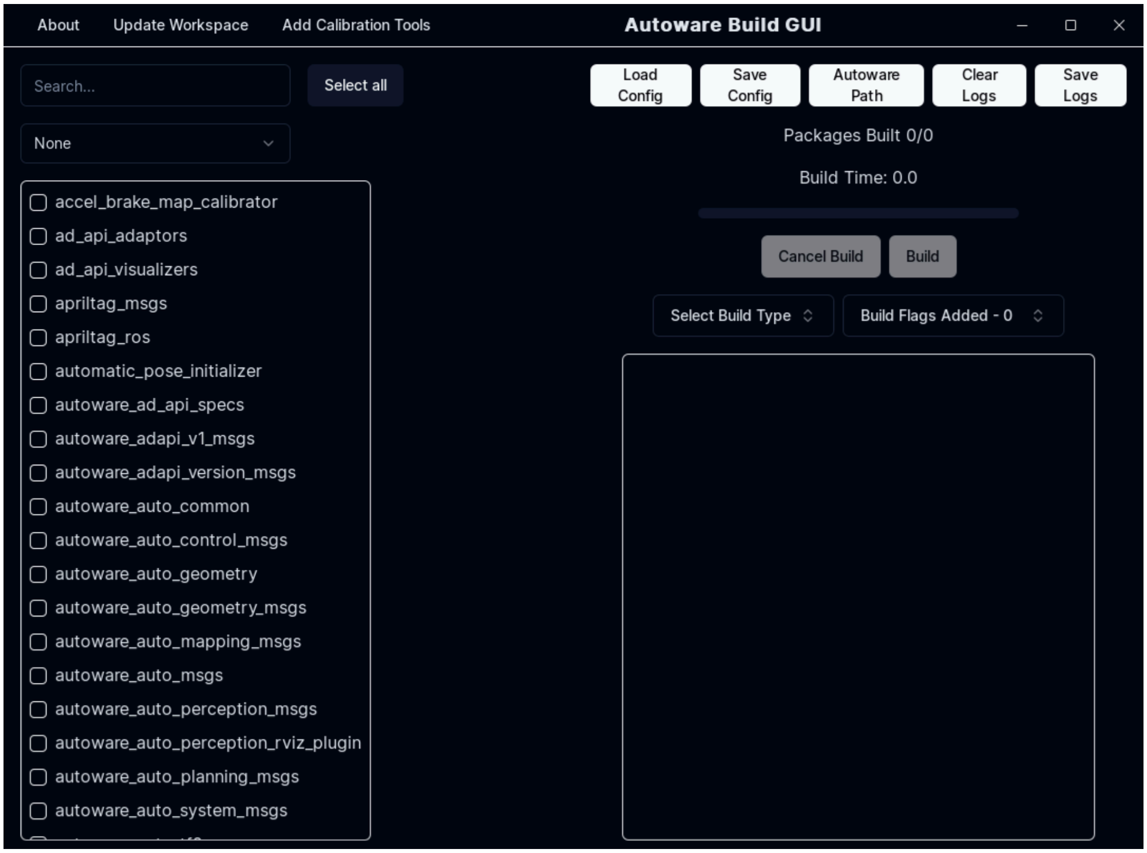
- **Initial Setup**: Follow the standard Autoware source installation guide to set up your environment and workspace.
- **Using the GUI**: Once the initial setup is complete, you can use the Autoware Build GUI to manage subsequent builds and package updates.

This integration offers a more accessible approach to building and managing Autoware packages, catering to both new users and experienced developers.

Getting started with Autoware Build GUI

1. **Installation:** Ensure you have installed the Autoware Build GUI. Installation instructions.

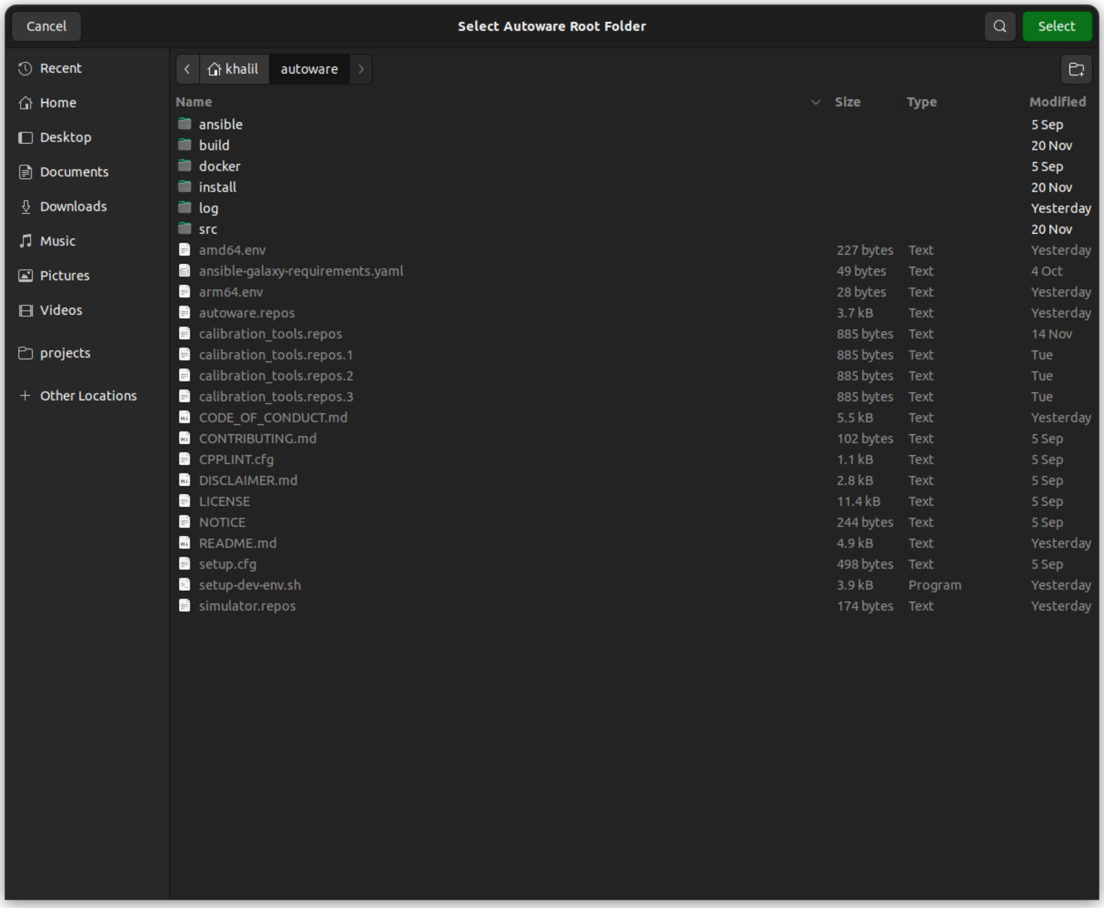
2. **Launching the App**: Once installed, launch the Autoware Build GUI.



Packages Built 0/0

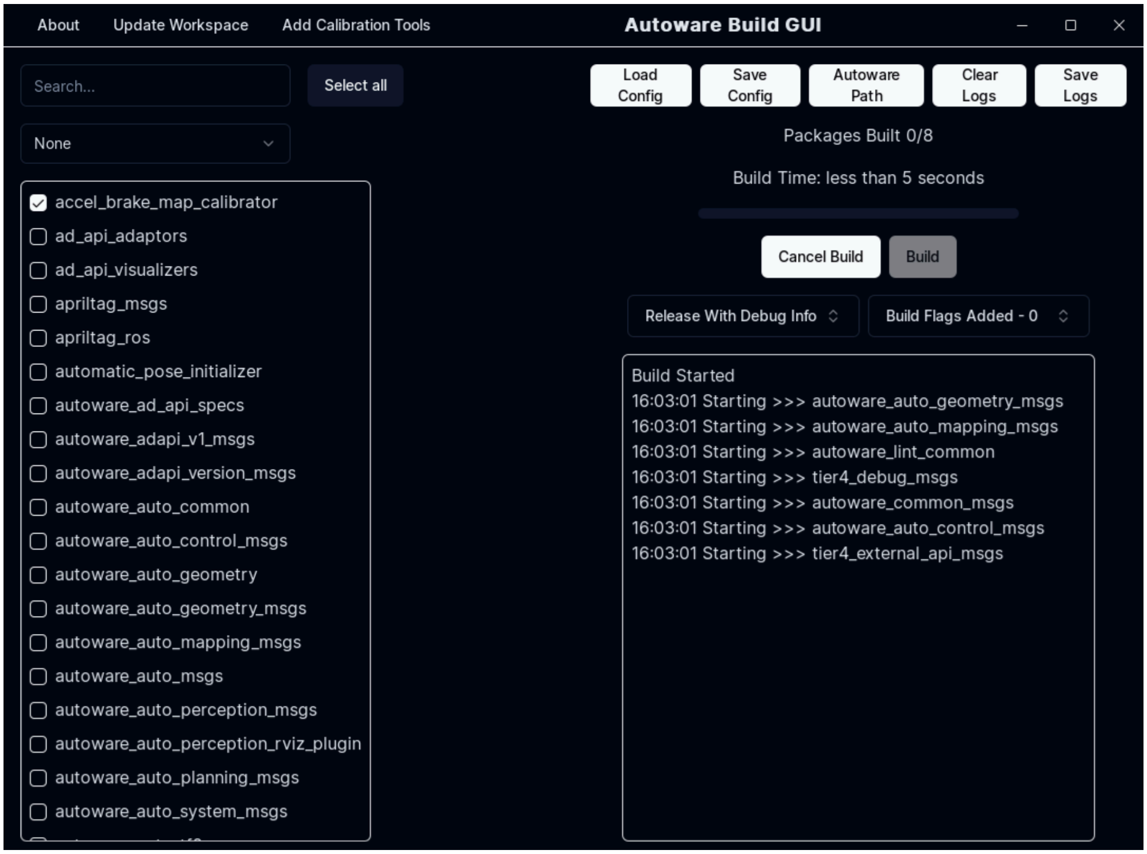
Build Time: 0.0

3. **Setting Up**: Set the path to your Autoware folder within the GUI.



Name	Size	Type	Modified
ansible			5 Sep
build			20 Nov
docker			5 Sep
install			20 Nov
log			Yesterday
src			20 Nov
amd64.env	227 bytes	Text	Yesterday
ansible-galaxy-requirements.yaml	49 bytes	Text	4 Oct
arm64.env	28 bytes	Text	Yesterday
autoware.repos	3.7 kB	Text	Yesterday
calibration_tools.repos	885 bytes	Text	14 Nov
calibration_tools.repos.1	885 bytes	Text	Tue
calibration_tools.repos.2	885 bytes	Text	Tue
calibration_tools.repos.3	885 bytes	Text	Tue
CODE_OF_CONDUCT.md	5.5 kB	Text	Yesterday
CONTRIBUTING.md	102 bytes	Text	5 Sep
CPPLINT.cfg	1.1 kB	Text	5 Sep
DISCLAIMER.md	2.8 kB	Text	5 Sep
LICENSE	11.4 kB	Text	5 Sep
NOTICE	244 bytes	Text	5 Sep
README.md	4.9 kB	Text	Yesterday
setup.cfg	498 bytes	Text	5 Sep
setup-dev-env.sh	3.9 kB	Program	Yesterday
simulator.repos	174 bytes	Text	Yesterday

4. **Building Packages**: Select the Autoware packages you wish to build and manage the build process through the GUI.



None

☑ accel_brake_map_calibrator
□ ad_api_adaptors
□ ad_api_visualizers
□ apriltag_msgs
□ apriltag_ros
□ automatic_pose_initializer
□ autoware_ad_api_specs
□ autoware_adapi_v1_msgs
□ autoware_adapi_version_msgs
□ autoware_auto_common
□ autoware_auto_control_msgs
□ autoware_auto_geometry
□ autoware_auto_geometry_msgs
□ autoware_auto_mapping_msgs
□ autoware_auto_msgs
□ autoware_auto_perception_msgs
□ autoware_auto_perception_rviz_plugin
□ autoware_auto_planning_msgs
□ autoware_auto_system_msgs

Packages Built 0/8

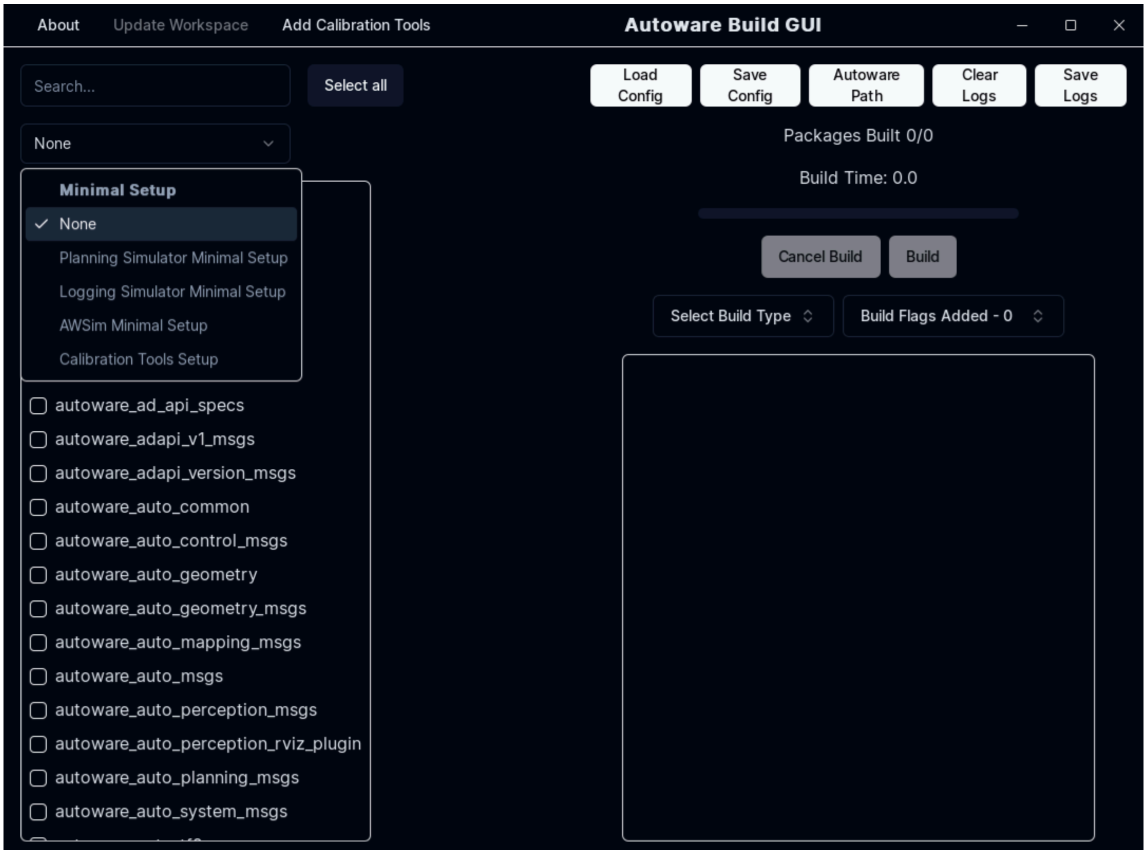
Build Time: less than 5 seconds

Release With Debug Info

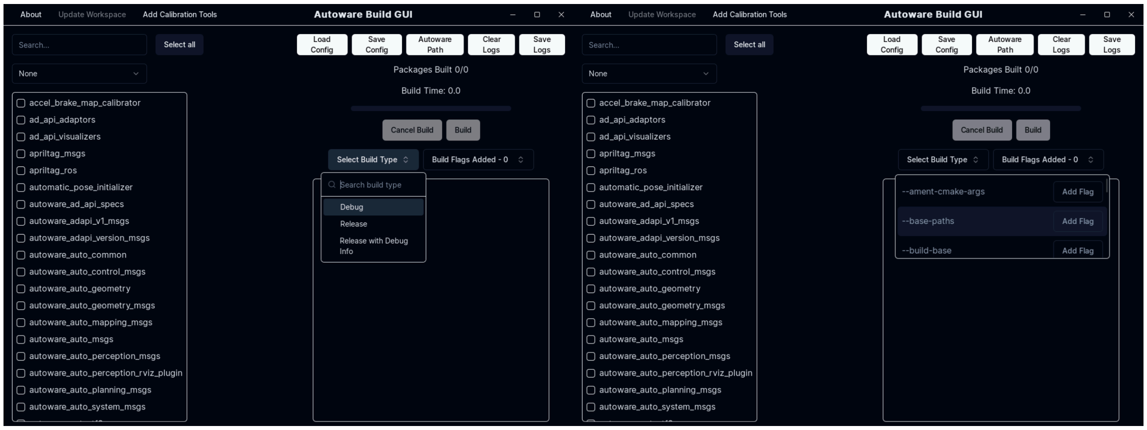
Build Flags Added - 0

```
Build Started
16:03:01 Starting >>> autoware_auto_geometry_msgs
16:03:01 Starting >>> autoware_auto_mapping_msgs
16:03:01 Starting >>> autoware_lint_common
16:03:01 Starting >>> tier4_debug_msgs
16:03:01 Starting >>> autoware_common_msgs
16:03:01 Starting >>> autoware_auto_control_msgs
16:03:01 Starting >>> tier4_external_api_msgs
```

4.1. **Build Configuration**: Choose from a list of default build configurations, or select the packages you wish to build manually.

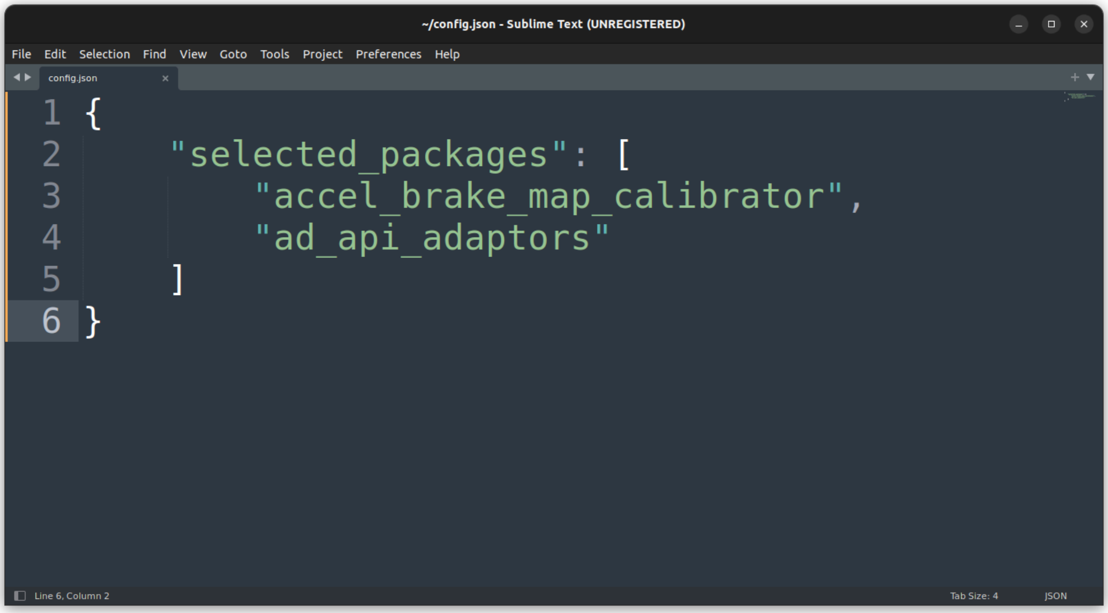


4.2. **Build Options**: Choose which build type you wish to use, with ability to specify additional build options.



5. **Save and Load**: Save your build configurations for future use, or load a previously saved configuration if you don't wish to build all packages or use one of the default configurations

provided.


~/config.json - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
config.json
Line 6, Column 2
Tab Size: 4
JSON

```json
{
	"selected_packages": [
		"accel_brake_map_calibrator",
		"ad_api_adaptors"
	]
}
```

6. **Updating Workspace**: Update your Autoware workspace's packages to the latest version using the GUI or adding Calibration tools to the workspace.