ROS 2 and Gazebo Integration Best Practices

- Michael Carroll & Dharini Dutia

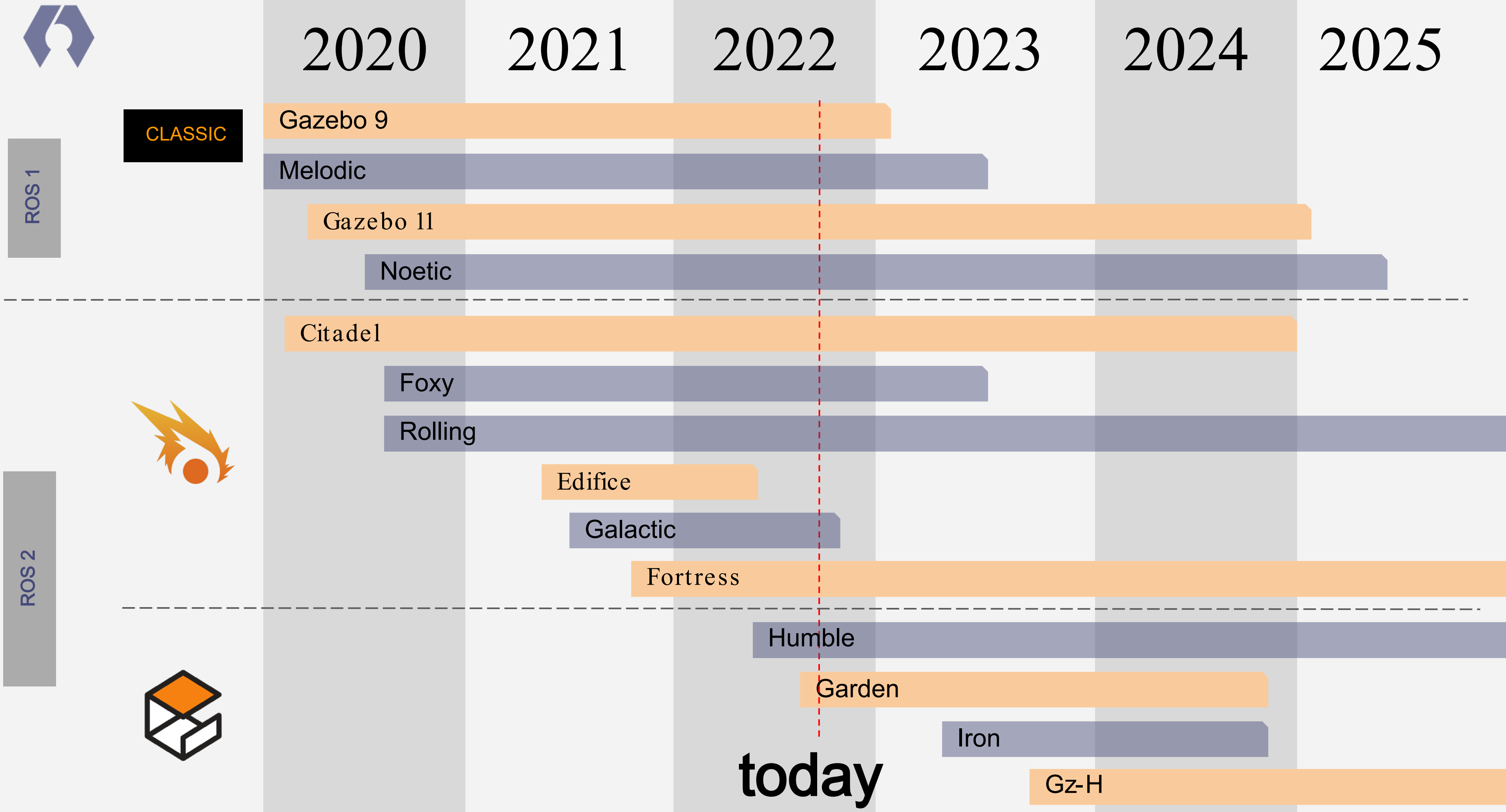
Outline

- ★ Choosing a ROS 2 and Gazebo Version
- ★ Structure your project
- ★ Creating Gazebo system plugins to control custom simulator behavior
- ★ Map ROS 2 and Gazebo topics/messages using a bridge
- ★ Creating or importing assets to be simulated
- ★ Running a simulation

Choosing a version

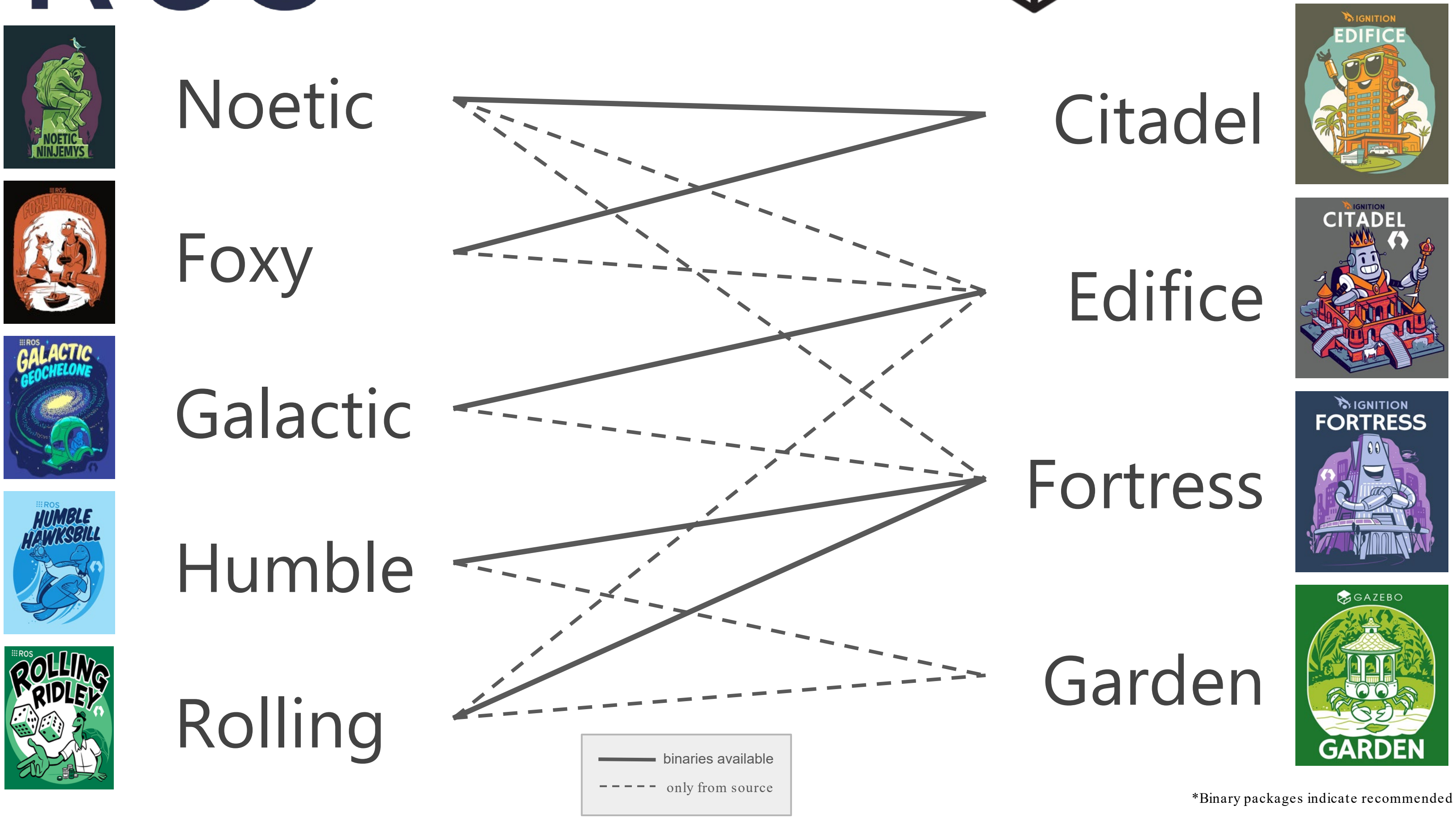
- Choose a version that matches any hard project constraints
  - e.g. OS version, ROS version, Gazebo version

- Levels of support
  - Official binary supported combinations
  - Official source supported combinations
  - Unsupported combinations


2020
2021
2022
2023
2024
2025
ROS 1
CLASSIC
Gazebo 9
Melodic
Gazebo 11
Noetic
ROS 2
Citadel
Foxy
Rolling
Edifice
Galactic
Fortress
Humble
Garden
Iron
Gz-H
today

ROS

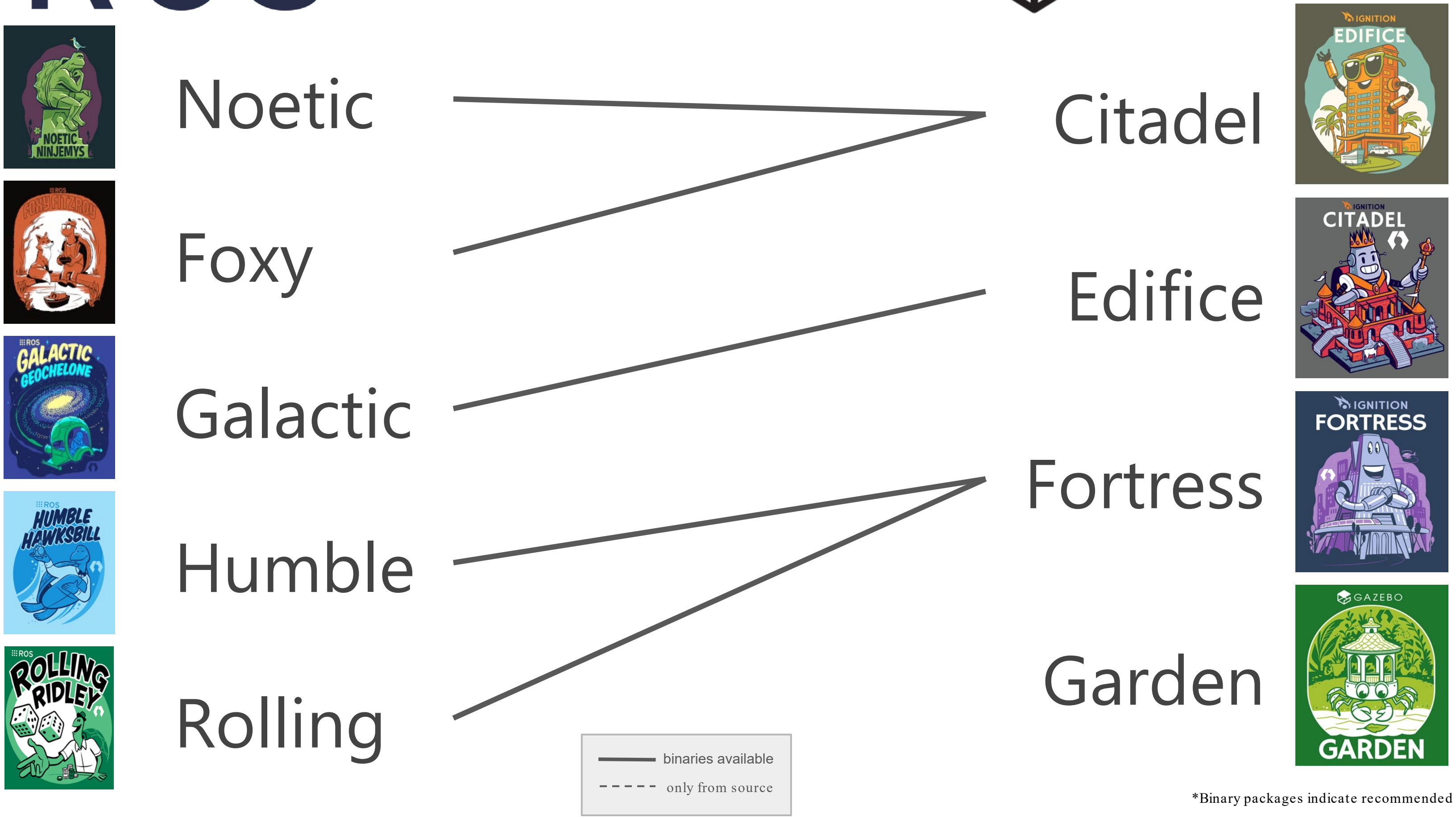
GAZEBO



*Binary packages indicate recommended combinations

ROS

GAZEBO



Noetic

Foxy

Galactic

Humble

Rolling

Citadel

Edifice

Fortress

Garden

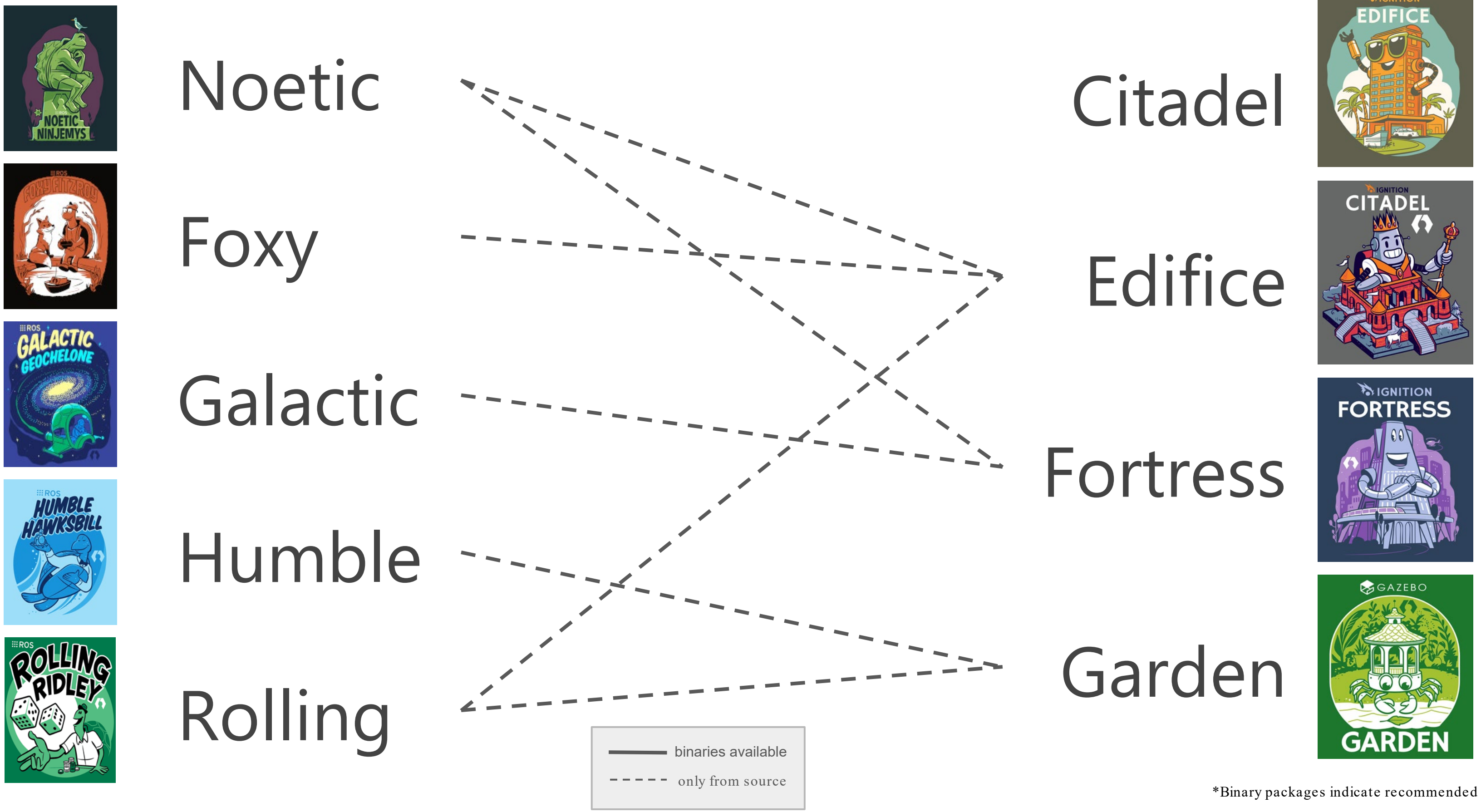
binaries available

only from source

*Binary packages indicate recommended combinations

ROS

GAZEBO



Noetic

Foxy

Galactic

Humble

Rolling

Citadel

Edifice

Fortress

Garden

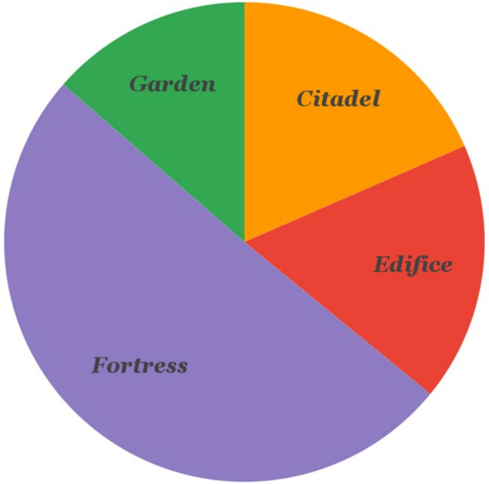
binaries available

only from source

*Binary packages indicate recommended combinations

ROS 2 and Gazebo Usage Survey Statistics*

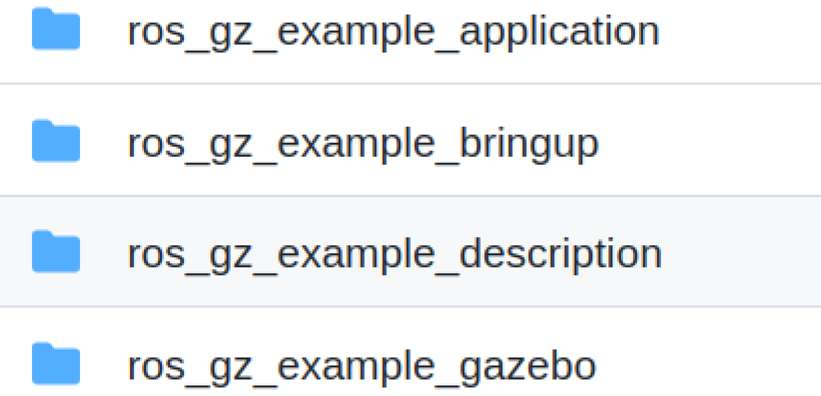
Though Gazebo Classic is widely used, *half* of surveyed ROS2 users have already worked with Gazebo Fortress



Still accepting responses

Structure your Project

To get started quickly, use the ros_gz_project_template!



ros_gz_example_application

ros_gz_example_bringup

ros_gz_example_description

ros_gz_example_gazebo

ROS 2 application libraries and nodes

ROS 2 launch files

SDFormat description of simulation assets

Gazebo specific system implementations

https://github.com/gazebosim/ros_gz_project_template

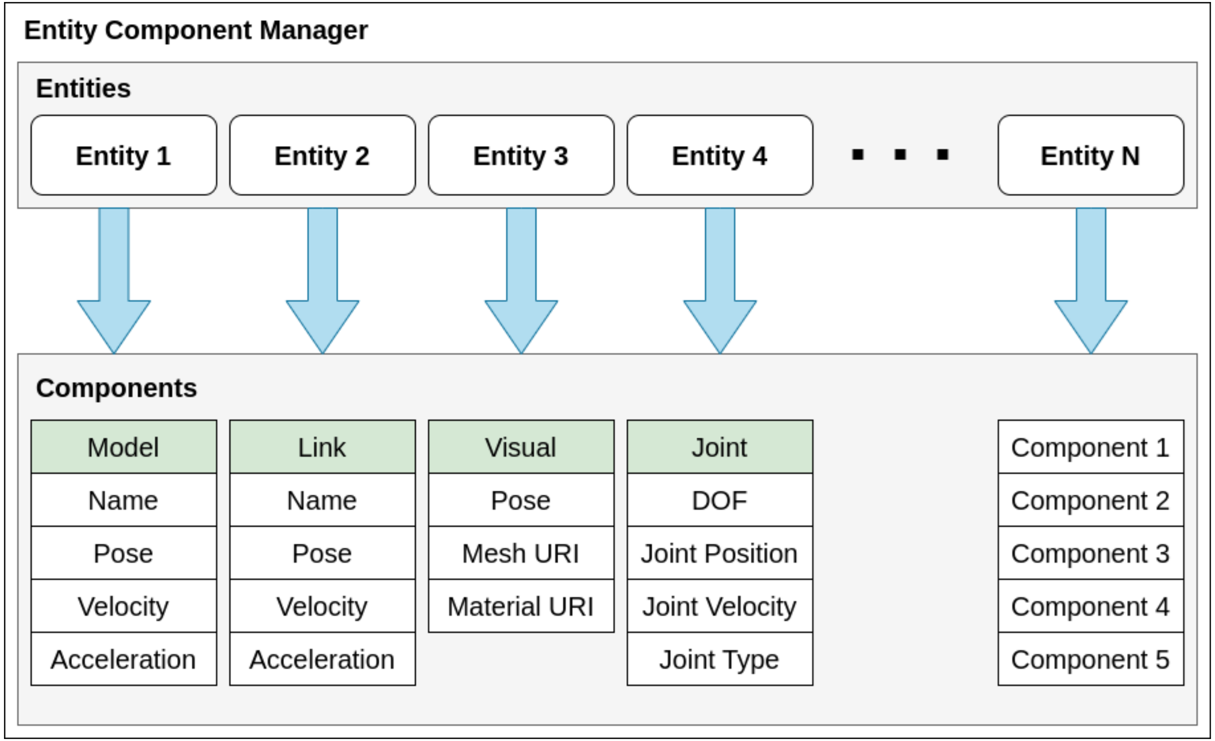
Writing Gazebo Systems

- Gazebo systems encapsulate all simulation-specific logic
- In contrast to Gazebo-classic, systems aren't specifically attached to models or worlds, but instead act upon entities and components
- Systems implement various interfaces to dictate behavior

TIP: Check to see what systems have been implemented before choosing to create one: https://github.com/gazebosim/gz-sim

The Entity Component Manager

- Every piece of simulation is an entity
- Each entity has one more more components attached to it



Entity Component Manager

Entities

Entity 1 | Entity 2 | Entity 3 | Entity 4 | ■ ■ ■ | Entity N

Components

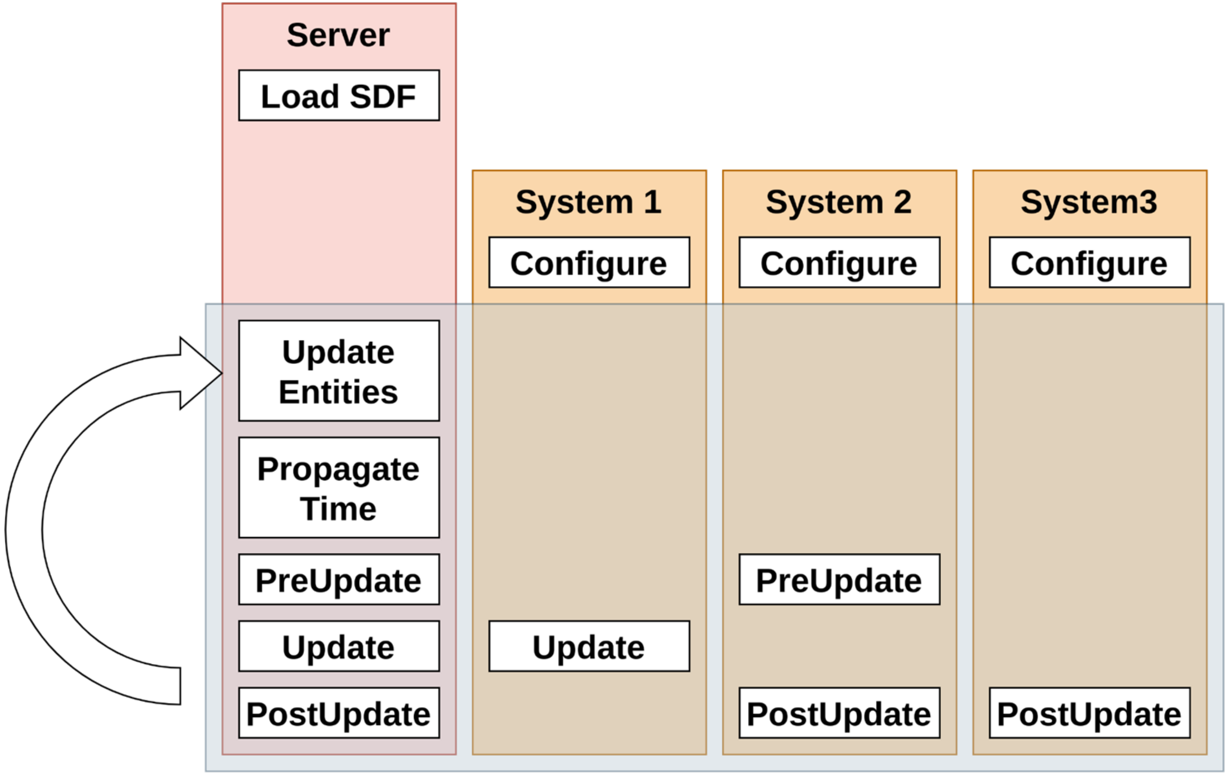
Model	Link	Visual	Joint	Component 1
Name	Name	Pose	DOF	Component 2
Pose	Pose	Mesh URI	Joint Position	Component 3
Velocity	Velocity	Material URI	Joint Velocity	Component 4
Acceleration	Acceleration		Joint Type	Component 5

Writing Gazebo Systems

APIs system developers can implement:

- Configure
  - Called when plugin loaded, provided ECM and SDF attributes
- PreUpdate
  - Can mutate entities and components to set forces, torques, velocities
- Update
  - Physics update, generally should not be implemented by any other systems
- PostUpdate
  - Cannot mutate, but can read components and publish/ send events
- Reset
  - Can be used to add reset-specific behavior

The Simulation Loop



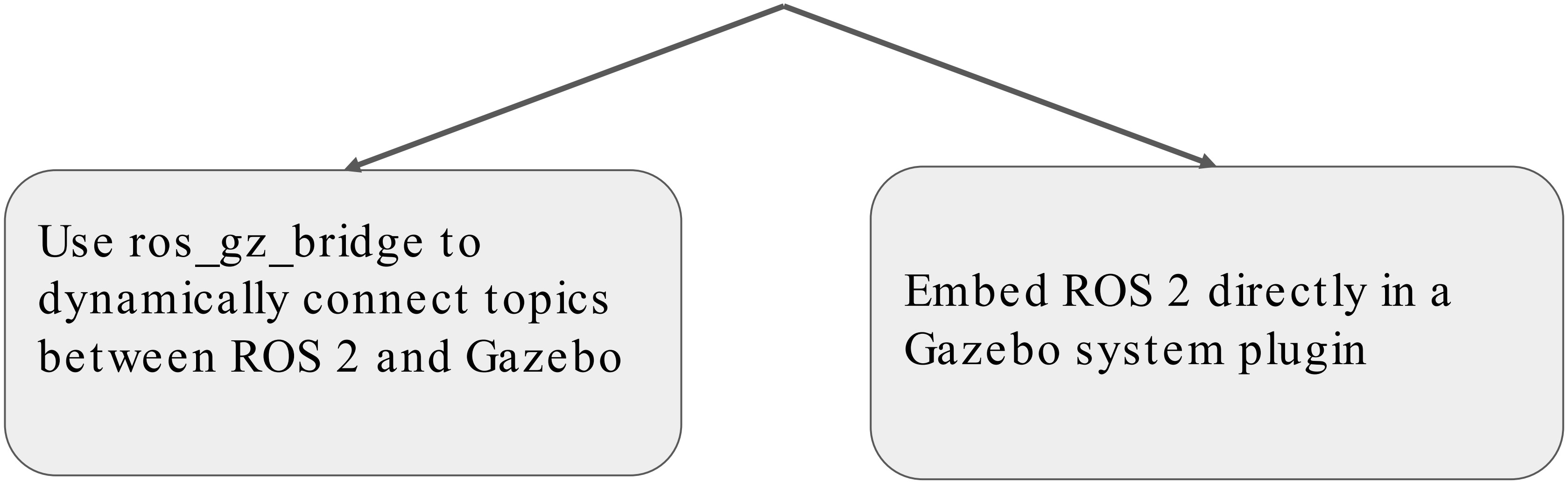
Building a Gazebo system in ROS 2

- Gazebo systems are shared libraries located via environment variables
- With ROS 2, use ament_hooks to install and locate Gazebo systems

```
# CMakeLists.txt
add_library(RosGzExampleSystem SHARED src/RosGzExampleSystem.cc)
install(TARGETS RosGzExampleSystem DESTINATION lib/${PROJECT_NAME})
ament_environment_hooks("${CMAKE_CURRENT_SOURCE_DIR}/hooks/${PROJECT_NAME}.dsv.in")

# Hooks file (.dsv format)
prepend-non-duplicate;GZ_SIM_RESOURCE_PATH;@CMAKE_INSTALL_PREFIX@/share/@PROJECT_NAME@/worlds
prepend-non-duplicate;GZ_SIM_SYSTEM_PLUGIN_PATH;lib/@PROJECT_NAME@/
```

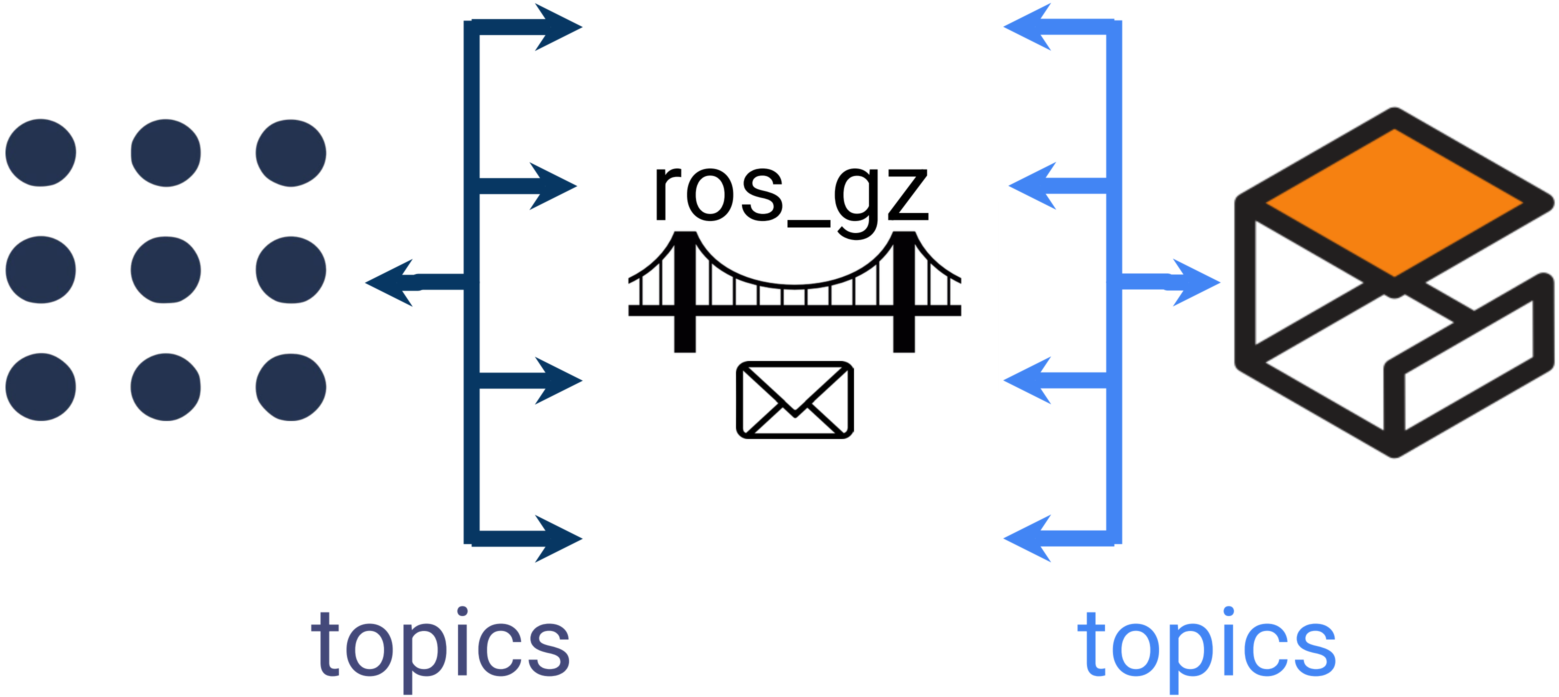
Connecting Gazebo and ROS 2

Two primary mechanisms depending on your application:



Use ros_gz_bridge to dynamically connect topics between ROS 2 and Gazebo

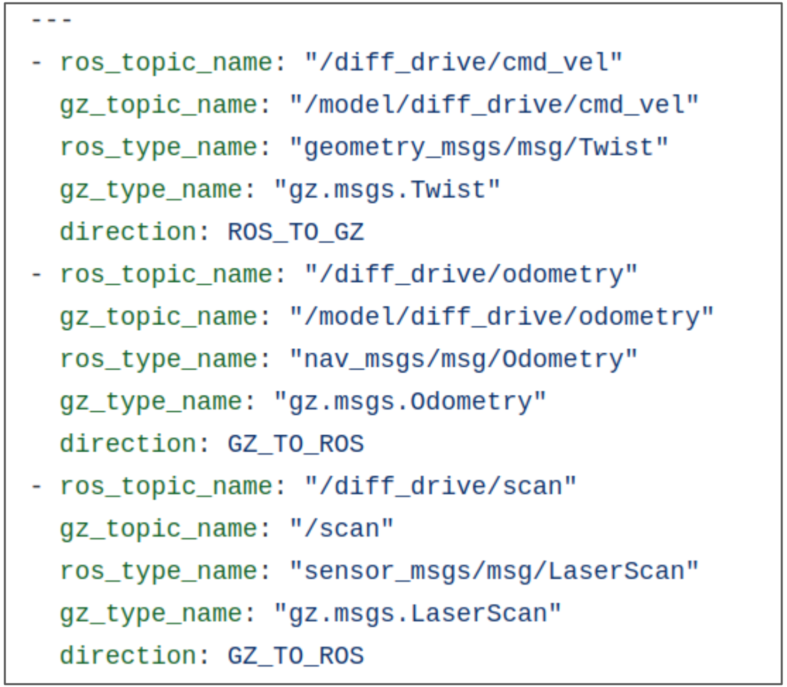
Embed ROS 2 directly in a Gazebo system plugin



The bridge isolates Gazebo transport topics and ROS 2 topics.

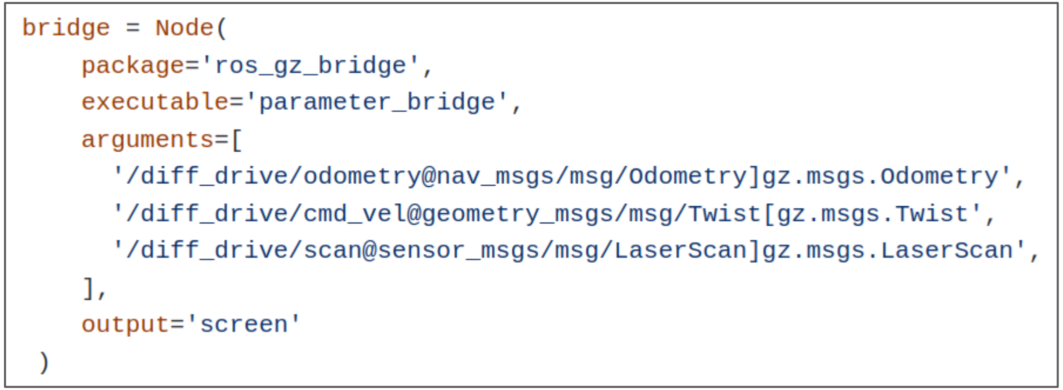
Each topic can be connected in one direction or bidirectionally.

Configuring ros_gz_bridge



```yaml
---
- ros_topic_name: "/diff_drive/cmd_vel"
  gz_topic_name: "/model/diff_drive/cmd_vel"
  ros_type_name: "geometry_msgs/msg/Twist"
  gz_type_name: "gz.msgs.Twist"
  direction: ROS_TO_GZ
- ros_topic_name: "/diff_drive/odometry"
  gz_topic_name: "/model/diff_drive/odometry"
  ros_type_name: "nav_msgs/msg/Odometry"
  gz_type_name: "gz.msgs.Odometry"
  direction: GZ_TO_ROS
- ros_topic_name: "/diff_drive/scan"
  gz_topic_name: "/scan"
  ros_type_name: "sensor_msgs/msg/LaserScan"
  gz_type_name: "gz.msgs.LaserScan"
  direction: GZ_TO_ROS
```

YAML



```python
bridge = Node(
    package='ros_gz_bridge',
    executable='parameter_bridge',
    arguments=[
      '/diff_drive/odometry@nav_msgs/msg/Odometry]gz.msgs.Odometry',
      '/diff_drive/cmd_vel@geometry_msgs/msg/Twist[gz.msgs.Twist',
      '/diff_drive/scan@sensor_msgs/msg/LaserScan]gz.msgs.LaserScan',
    ],
    output='screen'
 )
```

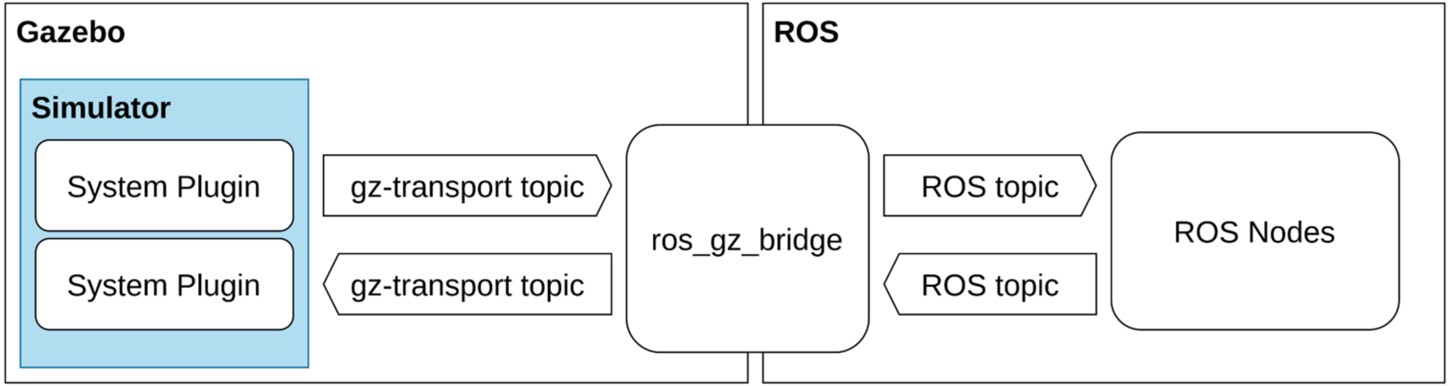
Command Line Arguments

Embedding ROS 2 in Gazebo

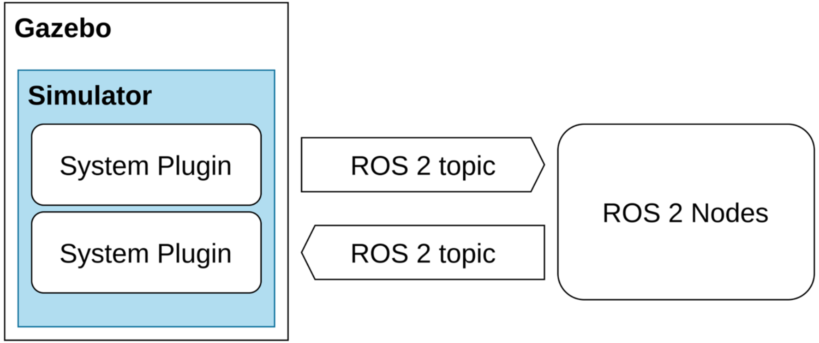
```
void RosSystem::Configure(
  const gz::sim::Entity & entity,
  const std::shared_ptr<const sdf::Element> & element,
  gz::sim::EntityComponentManager & ecm,
  gz::sim::EventManager & eventManager)
{
  // Ensure that ROS is setup
  if (!rclcpp::ok()) {
    rclcpp::init(0, nullptr);
  }

  // Read configuration from SDF file
  auto node_name = element->Get<std::string>("node_name", "RosSystem").first;
  auto talker_topic = element->Get<std::string>("talker_topic", "talker").first;
  auto listener_topic = element->Get<std::string>("listener_topic", "listener").first;

  node_ = rclcpp::Node::make_shared(node_name);
  listener_sub_ = node_->create_subscription<std_msgs::msg::String>(listener_topic,
    1, std::bind(&RosSystem::OnStringMessage, this, std::placeholders::_1));
  talker_pub_ = node_->create_publisher<std_msgs::msg::String>(talker_topic, 1);
}
```

Embedding ROS 2 in Gazebo

Bridge



Direct Embedding



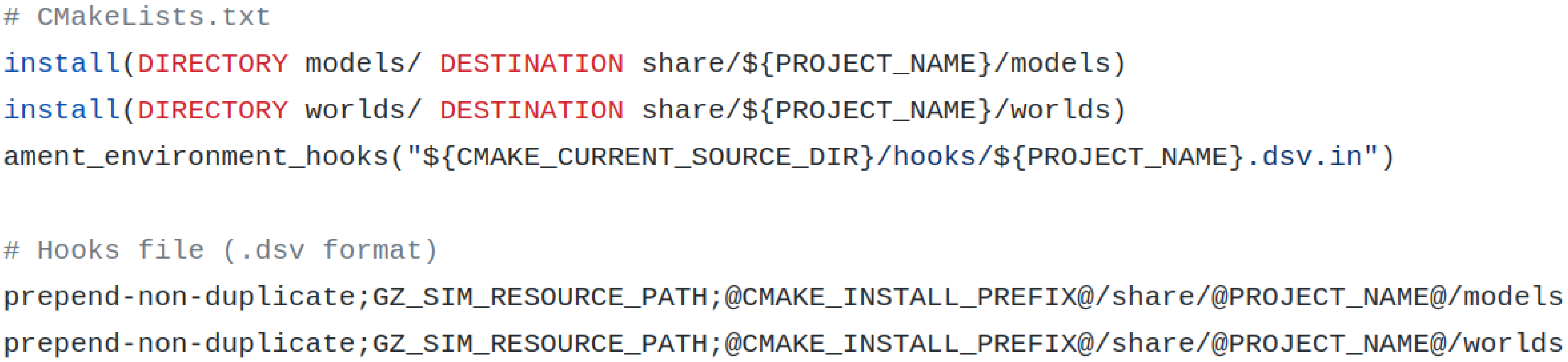
Bridge vs Embedding

ros_gz_bridge	**Embedded Node**
- Limited to topics and services	+ More access to ROS primitives
+ Isolates Gazebo and ROS versions	- Couples Gazebo and ROS versions
+ Isolates Gazebo and ROS runtime	- Couples Gazebo and ROS runtime
- Access to simulator state only through exposed transport topics	+ Direct access to simulator state

Bonus: In ROS 2, no roscore makes embedding easier than ever!

Simulation Assets

- Assets = models (URDF, SDF, etc), meshes and materials, world SDFs
- Can be installed as part of ROS 2 packages and exported as model:// or package://



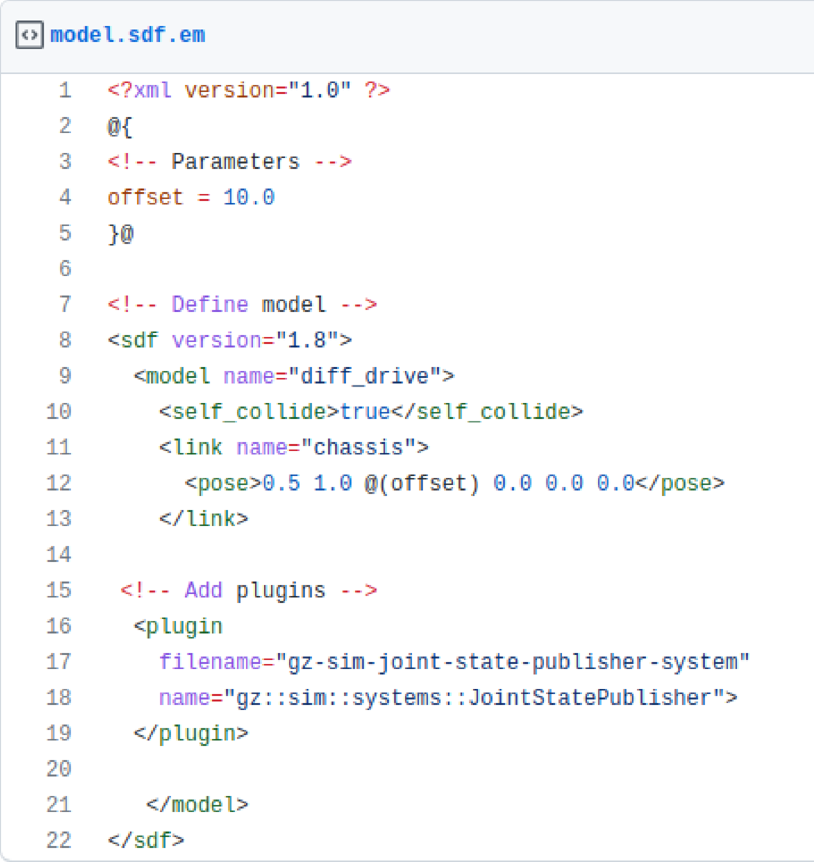
```
# CMakeLists.txt
install(DIRECTORY models/ DESTINATION share/${PROJECT_NAME}/models)
install(DIRECTORY worlds/ DESTINATION share/${PROJECT_NAME}/worlds)
ament_environment_hooks("${CMAKE_CURRENT_SOURCE_DIR}/hooks/${PROJECT_NAME}.dsv.in")

# Hooks file (.dsv format)
prepend-non-duplicate;GZ_SIM_RESOURCE_PATH;@CMAKE_INSTALL_PREFIX@/share/@PROJECT_NAME@/models
prepend-non-duplicate;GZ_SIM_RESOURCE_PATH;@CMAKE_INSTALL_PREFIX@/share/@PROJECT_NAME@/worlds
```

Creating Assets

- Simulated models and worlds are defined by SDFormat description files in Gazebo

- SDF files can be static (loaded from disk) or dynamically generated on the fly:
    - Using template languages like ERB
    - Using pySDF

- Gazebo systems are attached via the sdf plugin tag



model.sdf.em

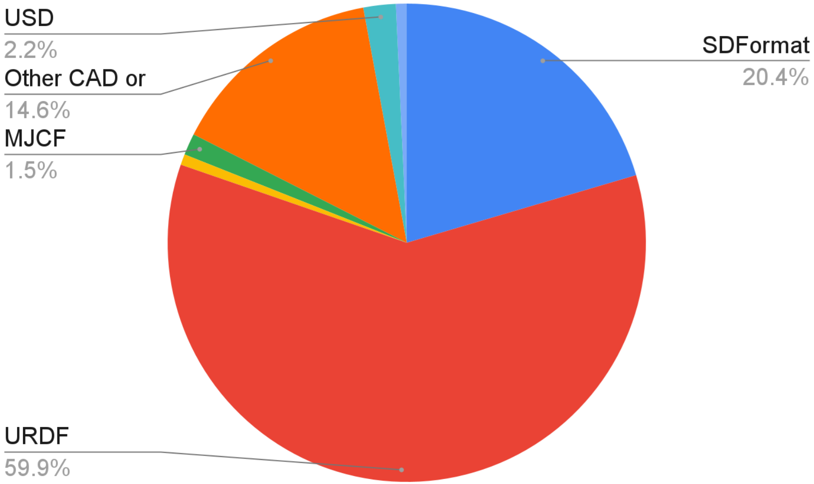
```
<?xml version="1.0" ?>
@{
<!-- Parameters -->
offset = 10.0
}@

<!-- Define model -->
<sdf version="1.8">
  <model name="diff_drive">
    <self_collide>true</self_collide>
    <link name="chassis">
      <pose>0.5 1.0 @(offset) 0.0 0.0 0.0</pose>
    </link>

 <!-- Add plugins -->
  <plugin
    filename="gz-sim-joint-state-publisher-system"
    name="gz::sim::systems::JointStatePublisher">
  </plugin>

   </model>
</sdf>
```

Importing Existing Assets

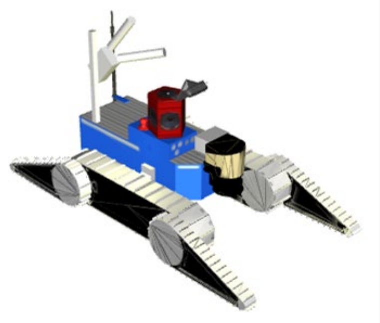
- There is a library of SDF simulation assets on Fuel

- Support for other file formats:
  - USD
  - Mujoco
  - URDF

- Assimp loader for other mesh formats
  - Collada, Blender, glTF



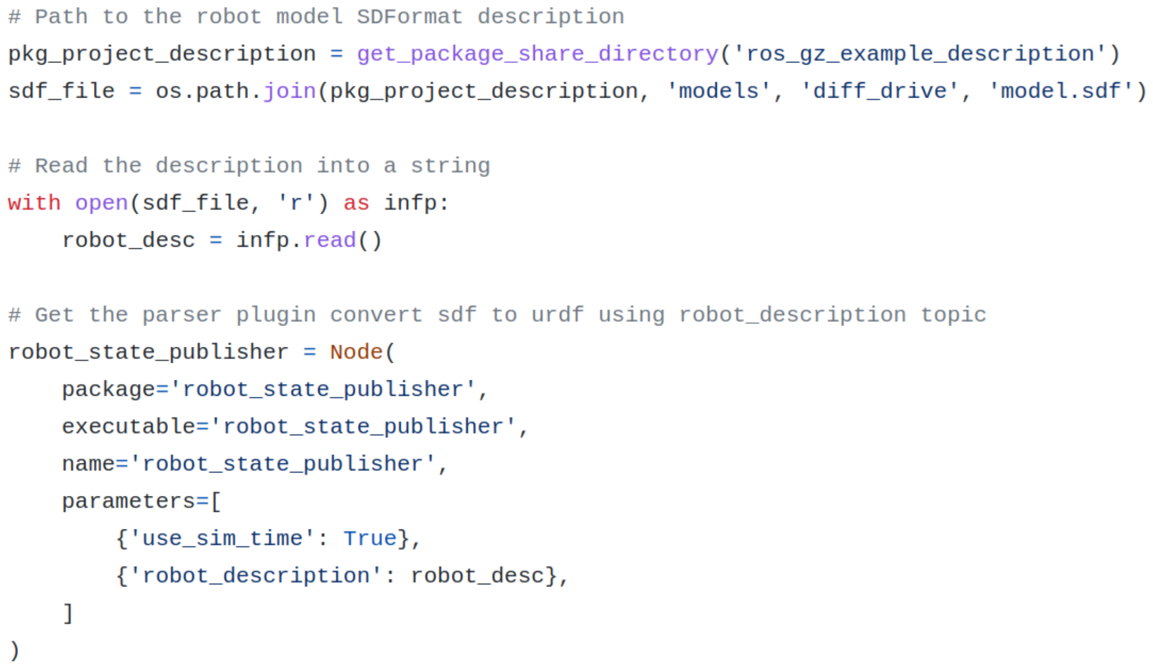
GAZEBO FUEL



app.gazebosim.org/dashboard

Using Assets in ROS

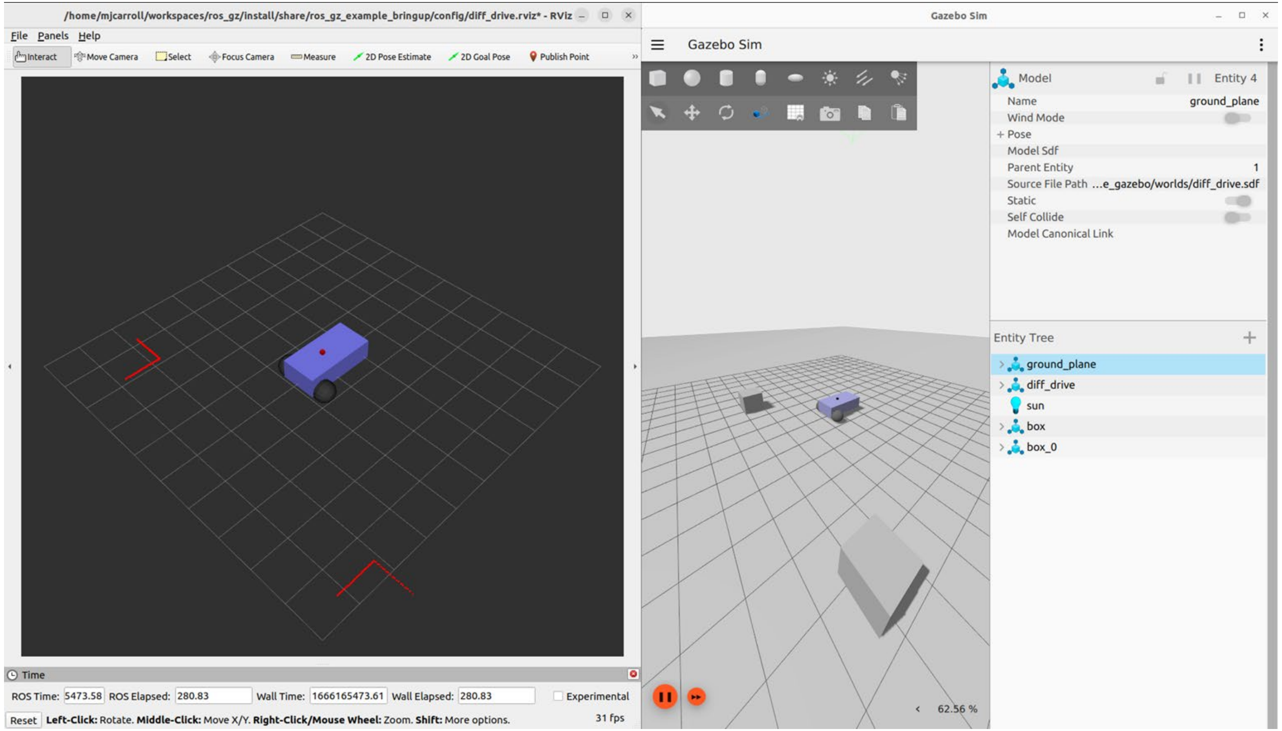
Use sdformat_urdf to share common assets:



```
# Path to the robot model SDFormat description
pkg_project_description = get_package_share_directory('ros_gz_example_description')
sdf_file = os.path.join(pkg_project_description, 'models', 'diff_drive', 'model.sdf')

# Read the description into a string
with open(sdf_file, 'r') as infp:
    robot_desc = infp.read()

# Get the parser plugin convert sdf to urdf using robot_description topic
robot_state_publisher = Node(
    package='robot_state_publisher',
    executable='robot_state_publisher',
    name='robot_state_publisher',
    parameters=[
        {'use_sim_time': True},
        {'robot_description': robot_desc},
    ]
)
```

Using Assets in ROS

Use sdformat_urdf to share common assets:



/home/mjcarroll/workspaces/ros_gz/install/share/ros_gz_example_bringup/config/diff_drive.rviz* - RViz

ROS Time: 5473.58 ROS Elapsed: 280.83 Wall Time: 1666165473.61 Wall Elapsed: 280.83 □ Experimental

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 31 fps

Model Entity 4

Name	ground_plane
Wind Mode	
+ Pose	
Model Sdf	
Parent Entity	1
Source File Path	...e_gazebo/worlds/diff_drive.sdf
Static	
Self Collide	
Model Canonical Link	

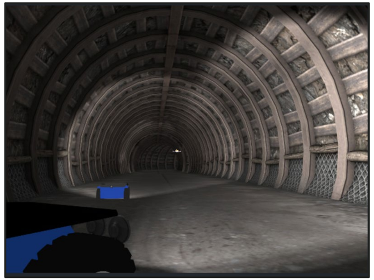
Entity Tree

- ground_plane
- diff_drive
- sun
- box
- box_0

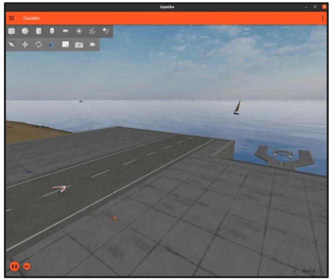
62.56 %

Running a simulation

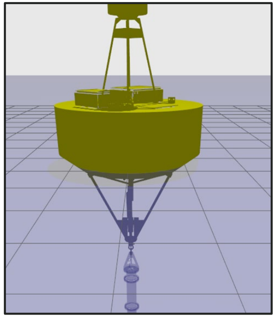
Examples of successful integrations



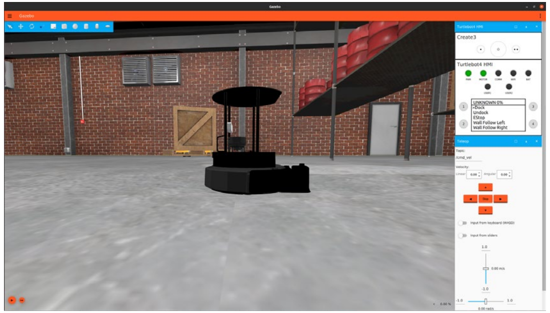
DARPA SubT Challenge



MBZIRC UAV and USV Challenge



MBARI Wave Energy Converter

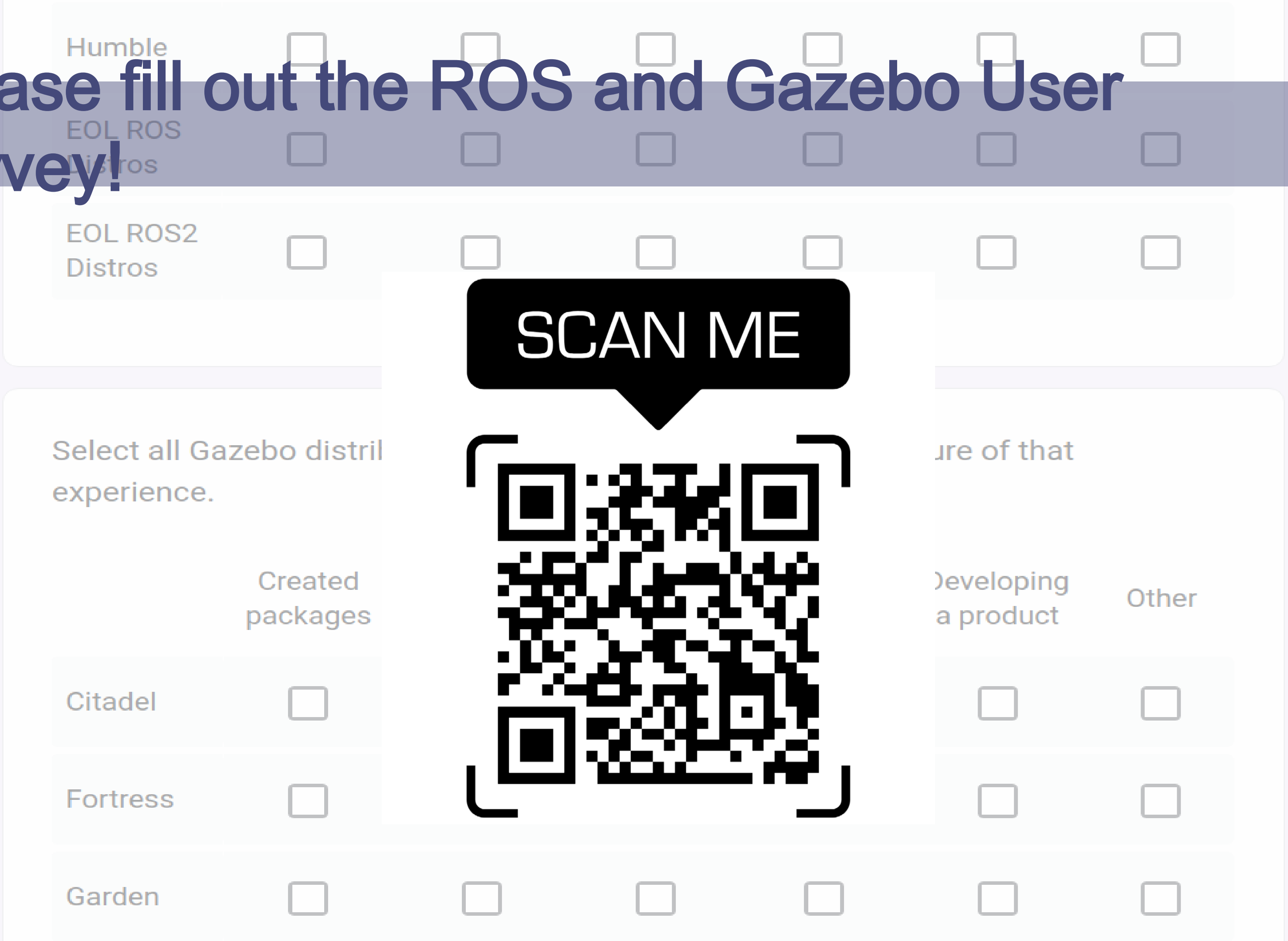


Topic: /cmd_vel

Velocity: Linear 0.00 Angular 0.00

TurtleBot 4 Simulator

Please fill out the ROS and Gazebo User Survey!



Humble	□	□	□	□	□	□
EOL ROS Distros	□	□	□	□	□	□
EOL ROS2 Distros	□	□	□	□	□	□

Select all Gazebo distril ure of that experience.

	Created packages				Developing a product	Other
Citadel	□				□	□
Fortress	□				□	□
Garden	□	□	□	□	□	□

SCAN ME

Thank you!
Any
questions?

https://github.com/gazebosim/ros_gz
https://github.com/gazebosim/ros_gz_project_template

Migration Notes for Gazebo -Classic to Gazebo Sim

```
// OLD
class GAZEBO_VISIBLE ArduPilotPlugin:
    public ModelPlugin
```

```
// NEW
class GZ_SIM_VISIBLE ArduPilotPlugin:
    public gz::sim::System,
    public gz::sim::ISystemConfigure,
    public gz::sim::ISystemPostUpdate,
    public gz::sim::ISystemPreUpdate
```

Migration Notes for Gazebo -Classic to Gazebo Sim

```
// OLD
virtual void Load(
  physics::ModelPtr _model,
  sdf::ElementPtr _sdf);
```

```
// NEW
void Configure(const gz::sim::Entity &_entity,
    const std::shared_ptr<const sdf::Element> &_sdf,
    gz::sim::EntityComponentManager &_ecm,
    gz::sim::EventManager &_eventMgr);
```

Migration Notes for Gazebo -Classic to Gazebo Sim

```
// OLD
void OnUpdate()
```

```
// NEW
void PreUpdate(const gz::sim::UpdateInfo &_info,
    gz::sim::EntityComponentManager &_ecm);

void PostUpdate(const gz::sim::UpdateInfo &_info,
    const gz::sim::EntityComponentManager &_ecm);
```