SDFormat: A robot description format in constant evolution

Alejandro Hernández Cordero
Addisu Taddese (Intrinsic)

ROSCON 2023

Table of content

SDFormat

- Simulation Description Format (SDFormat) is an XML format for describing objects and environments for robot simulators.
- Describes objects and environments for robot simulators, visualization, and control
- Designed for scientific robot applications
- Extensible format describing all aspects of robots, static and dynamic objects, lighting, terrain, and physics.

```xml
<?xml version="1.0" ?>
<sdf version="1.6">
  <world name="minimal_scene">

    <gui fullscreen="0">

      <!-- 3D scene -->
      <plugin filename="MinimalScene" name="3D View">
        <ignition-gui>
          <title>3D View</title>
          <property type="bool" key="showTitleBar">false</property>
          <property type="string" key="state">docked</property>
        </ignition-gui>

        <engine>ogre2</engine>
        <scene>scene</scene>
        <ambient_light>0.4 0.4 0.4</ambient_light>
        <background_color>0.8 0.8 0.8</background_color>
        <camera_pose>-6 0 6 0 0.5 0</camera_pose>
        <camera_clip>
          <near>0.25</near>
          <far>25000</far>
        </camera_clip>
      </plugin>

      <!-- Plugins that add functionality to the scene -->
      <plugin filename="EntityContextMenuPlugin" name="Entity context menu">
        <ignition-gui>
          <property key="state" type="string">floating</property>
          <property key="width" type="double">5</property>
          <property key="height" type="double">5</property>
          <property key="showTitleBar" type="bool">false</property>
        </ignition-gui>
      </plugin>
      <plugin filename="GzSceneManager" name="Scene Manager">
        <ignition-gui>
          <property key="resizable" type="bool">false</property>
          <property key="width" type="double">5</property>
          <property key="height" type="double">5</property>
          <property key="state" type="string">floating</property>
          <property key="showTitleBar" type="bool">false</property>
        </ignition-gui>
      </plugin>
      <plugin filename="InteractiveViewControl" name="Interactive view control"
```

Table of content

How you can improve SDFormat

- Make a proposal
- Request changes
  - Direct conversations, GitHub issues, ROS Discourse, Gazebo Community, etc
- Write a formal proposal!
  - Open a pull request on http://bit.ly/github_sdf_tutorials
  - Guidelines https://bit.ly/sdf_proposal_format
- Give feedback on new proposals

Table of content

SDFormat python bindings

- `pybind11`
- Programmatically create/edit models
- Compatible with `gz.math` and `gz.sim`
- Tutorial https://bit.ly/sdf_python_bindings

How to use it

```
import sdformat13 as sdf

root = sdf.Root()
try:
  root.load(input_file)
except sdf.SDFErrorsException as e:
  print(e, file=sys.stderr)

# Create a new element
world = root.world_by_index(0)
world.set_name('shapes')


…

with open('new_sdf.sdf', "w") as f:
  f.write(root.to_string())
```

Table of content

USD

- High-performance extensible software platform for animated 3D scenes
- Designed to meet the needs of large-scale film and visual effects production
- Expanding set of schemas, covering geometry, shading, lighting, and physics

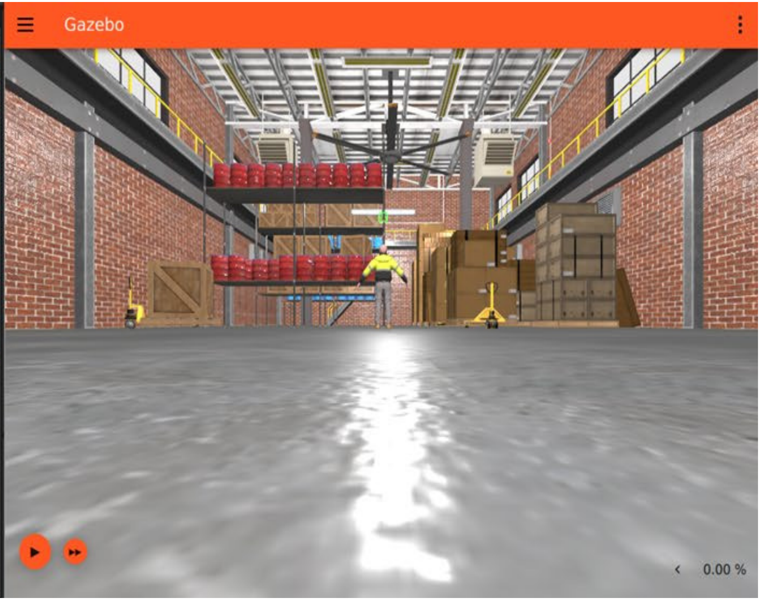
```
#usda 1.0
(
    endTimeCode = 100
    metersPerUnit = 1
    startTimeCode = 0
    timeCodesPerSecond = 24
    upAxis = "Z"
)

def "fuel"
{
    def PhysicsScene "physics"
    {
        vector3f physics:gravityDirection = (0, 0, -1)
        float physics:gravityMagnitude = 9.8
    }

    def Xform "panda" (
        prepend apiSchemas = ["PhysicsArticulationRootAPI"]
    )
    {
        float3 xformOp:rotateXYZ = (0, 0, 0)
        double3 xformOp:translate = (0, 0, 0)
        uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

        def Xform "panda_link0" (
            prepend apiSchemas = ["PhysicsRigidBodyAPI", "PhysicsMassAPI"]
        )
        {
            point3f physics:centerOfMass = (-0.025566, -0.0000287883, 0.057332)
            float3 physics:diagonalInertia = (0.0075390637, 0.010508018, 0.009864934)
            float physics:mass = 2.8142712
            float3 xformOp:rotateXYZ = (0, 0, 0)
            double3 xformOp:translate = (0, 0, 0)
            uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

            def Xform "panda_link0_visual"
            {
                float3 xformOp:rotateXYZ = (0, 0, 0)
                double3 xformOp:translate = (0, 0, 0)
                uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

                def "geometry" (
                    prepend apiSchemas = ["PhysicsCollisionAPI"]
                )
                {
                    rel material:binding = </Looks/Material_1>
```

open robotics

SDFormat -> USD

```
$ sdf2usd warehouse.sdf warehouse.usd
```

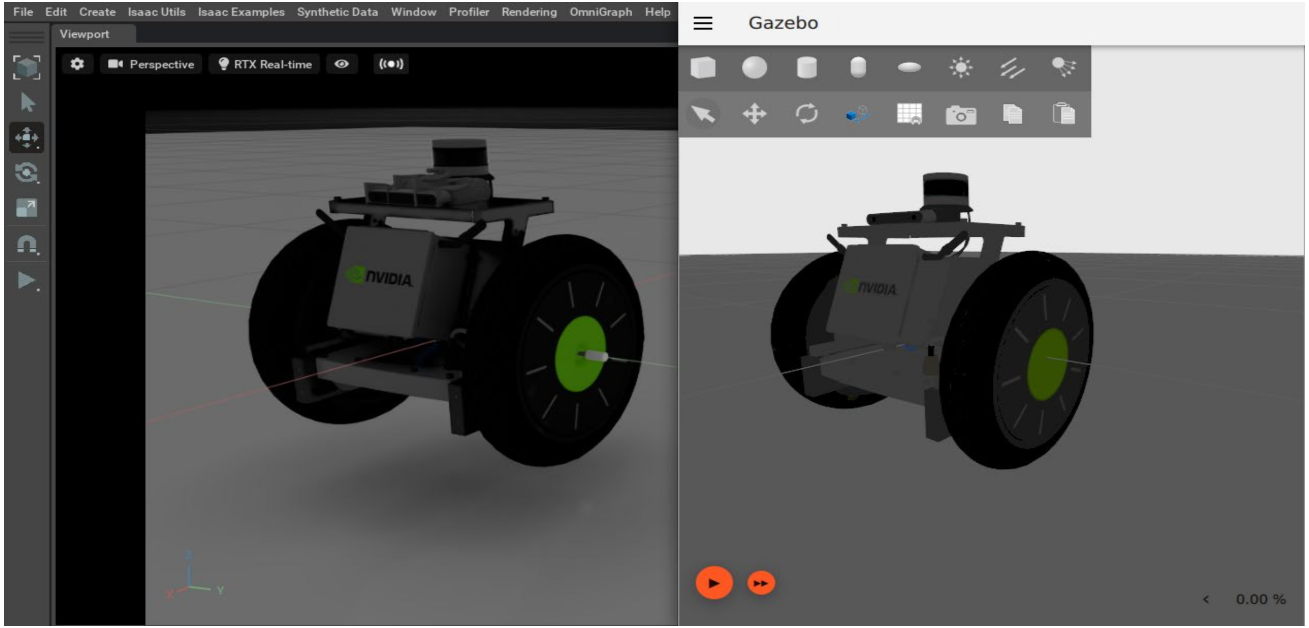
Isaac Sim



Gazebo

USD -> SDFormat

```
$ usd2sdf robot.usd robot.sdf
```



Isaac Sim Gazebo

MuJoCo

- XML model files
- MJCF models can represent complex dynamical systems with a wide range of features and model elements.
- Support joint, geom, site, camera and light.
  - Supported in the converter:
    - Bodies
    - Geoms
    - Sensors (Camera, Force torque, IMU)
    - Joints (Fixed, Free, Hinge, Slide)
    - Materials

```xml
<mujoco model="22 Humanoids">
  <option timestep="0.005"/>

  <size memory="100M"/>

  <asset>
    <texture type="skybox" builtin="gradient" rgb1=".3 .5 .7" rgb2="0 0 0" width="512" height="512"/>
    <texture name="body" type="cube" builtin="flat" mark="cross" width="127" height="1278"
             rgb1="0.8 0.6 0.4" rgb2="0.8 0.6 0.4" markrgb="1 1 1" random="0.01"/>
    <material name="body" texture="body" texuniform="true" rgba="0.8 0.6 .4 1"/>
    <texture name="grid" type="2d" builtin="checker" width="512" height="512" rgb1=".1 .2 .3" rgb2=".2 .3 .4"/>
    <material name="grid" texture="grid" texrepeat="1 1" texuniform="true" reflectance=".2"/>
  </asset>

  <default>
    <motor ctrlrange="-1 1" ctrllimited="true"/>
    <default class="body">
      <geom type="capsule" condim="1" friction=".7" solimp=".9 .99 .003" solref=".015 1" material="body"/>
      <joint type="hinge" damping=".2" stiffness="1" armature=".01" limited="true" solimplimit="0 .99 .01"/>
      <default class="big_joint">
        <joint damping="5" stiffness="10"/>
        <default class="big_stiff_joint">
          <joint stiffness="20"/>
        </default>
      </default>
    </default>
  </default>

  <visual>
    <map force="0.1" zfar="30"/>
    <rgba haze="0.15 0.25 0.35 1"/>
    <quality shadowsize="4096"/>
    <global offwidth="800" offheight="800"/>
  </visual>

  <worldbody>
    <geom size="10 10 .05" type="plane" material="grid" condim="3"/>
    <light dir=".2 1 -.4" diffuse=".8 .8 .8" specular="0.3 0.3 0.3" pos="-2 -10 4" cutoff="35"/>
    <light dir="-.2 1 -.4" diffuse=".8 .8 .8" specular="0.3 0.3 0.3" pos="2 -10 4" cutoff="35"/>

    <body name="1a_torso" pos="-1 0 1.5" childclass="body">
      <camera name="1a_back" pos="-3 0 1" xyaxes="0 -1 0 1 0 2" mode="trackcom"/>
      <camera name="1a_side" pos="0 -3 1" xyaxes="1 0 0 0 1 2" mode="trackcom"/>
      <freejoint name="1a_root"/>
      <geom name="1a_torso" fromto="0 -.07 0 0 .07 0" size=".07"/>
      <geom name="1a_upper_waist" fromto="-.01 -.06 -.12 -.01 .06 -.12" size=".06"/>
      <body name="1a_head" pos="0 0 .19">
        <geom name="1a_head" type="sphere" size=".09"/>
```

open robotics

SDFormat-MJCF: Installation

With `gz-garden` or higher version installed:

```
$ pip install sdformat-mjcf
```

```
❯ pip install sdformat-mjcf
Collecting sdformat-mjcf
  Using cached sdformat_mjcf-0.1.2-py3-none-any.whl (44 kB)
Collecting dm-control
  Using cached dm_control-1.0.7-py3-none-any.whl (38.6 MB)
Collecting mujoco>=2.2.2
  Downloading mujoco-2.2.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.8/3.8 MB 10.6 MB/s eta 0:00:00
Collecting dm-env
  Downloading dm_env-1.5-py3-none-any.whl (26 kB)
Collecting labmaze
  Downloading labmaze-1.0.5-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.whl (4.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 10.8 MB/s eta 0:00:00
Collecting pyparsing<3.0.0
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
```

SDFormat -> MJCF

```
$ sdf2mjcf input_file.sdf output_dir
```

Example: Pand Arm

MJCF -> SDFormat

```
$ mjcf2sdf input_file.xml output_dir
```

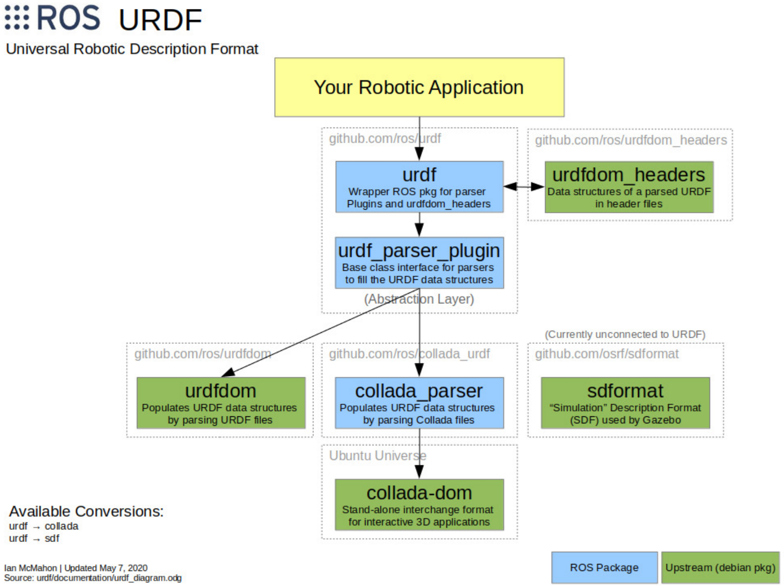
Example: Humanoid

Table of content

- What is SDFormat?
- How you can improve SDFormat
- Python bindings
- Converters for other robot description formats
- SDFormat in ROS 2
- New improvements in the SDFormat specification

SDFormat in ROS 2

- A `urdf_parser_plugin` for SDFormat
  - Parses SDFormat into URDF C++ data structures
  - SDFormat files can be loaded directly into `robot_describtion`



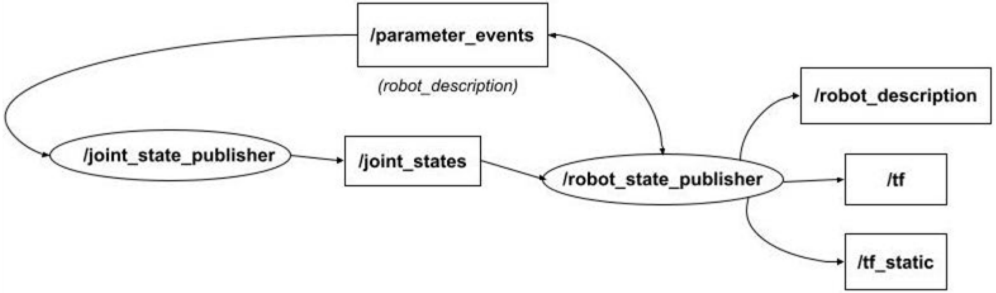


Table of content

Improved Composition

- Using the `<include>` tag for modularity
- Models are self-contained/standalone components similar to parts in CAD assembly
- Frames, links, and joints in a model should be considered the public "API" of the model
  - Frames as primary interface elements (eg. use frames to define mounting points)

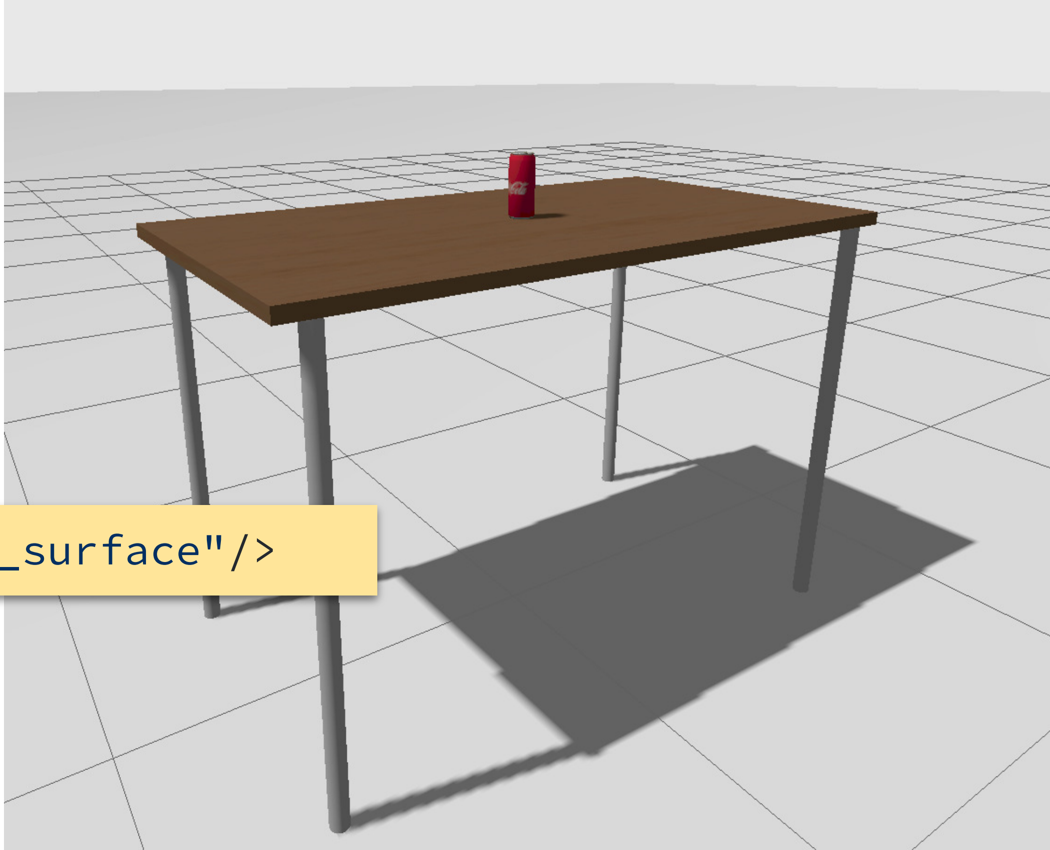
Model scope and the "::" syntax

- Only relative references are permitted
- References can access the current scope or child scopes

```
<model name="top_model">
  <link name="top_link"/>

  <frame name="top_to_bottom">
    <pose relative_to="mid_model::bottom_model::bottom_link"/>  <!-- VALID -->
  </frame>

  <model name="mid_model">
    <link name="mid_link">
      <pose relative_to="top_link"/> <!--Error -->
    </link>

    <model name="bottom_model">
      <link name="bottom_link">
    </model>
  </model>
</model>
```

Example: Frame Semantics with Nested References

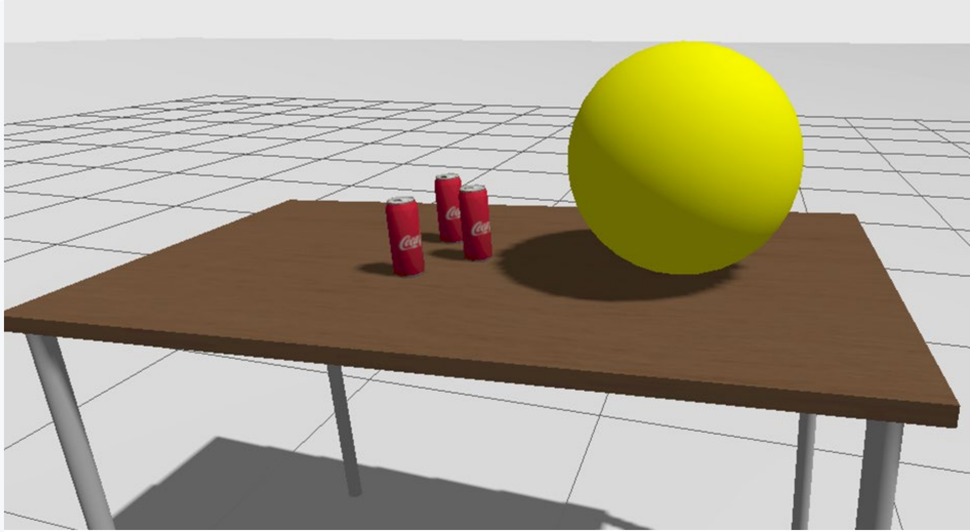
```
<sdf version="1.8">
  <world name="default">
    <include>
      <uri>models/table</uri>
      <name>table1</name>
    </include>

    <include>
      <uri>models/Coke</uri>
      <pose relative_to="table1::top_surface"/>
    </include>
  </world>
</sdf>
```



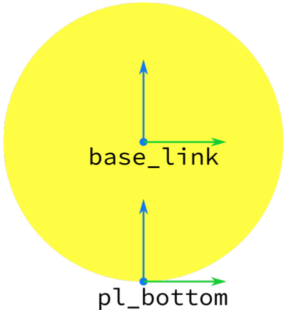
```
<pose relative_to="table1::top_surface"/>
```

Example: Placement Frame

```
<sdf version="1.8">
  <world name="default">
    <include>
      <uri>models/table</uri>
      <name>table1</name>
      <pose>0 0 0 0 -0.05 0</pose>
    </include>

    ...

    <include>
      <uri>models/big_sphere</uri>
      <placement_frame>pl_bottom</placement_frame>
      <pose relative_to="table1::top_surface">
         0.4 0 0 0 0 0
      </pose>
    </include>
  </world>
</sdf>
```

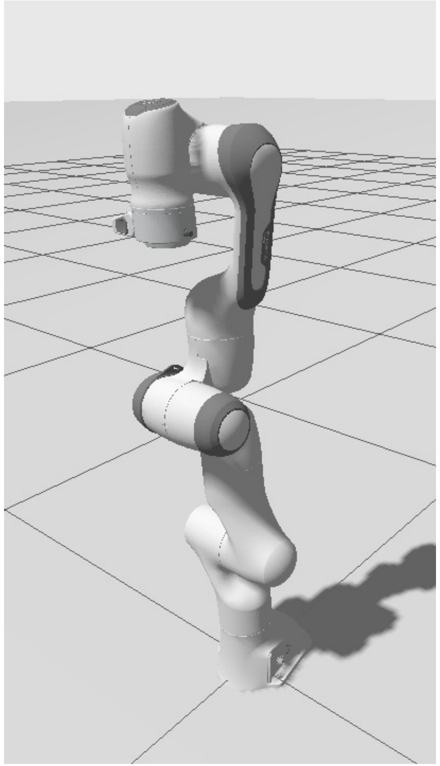


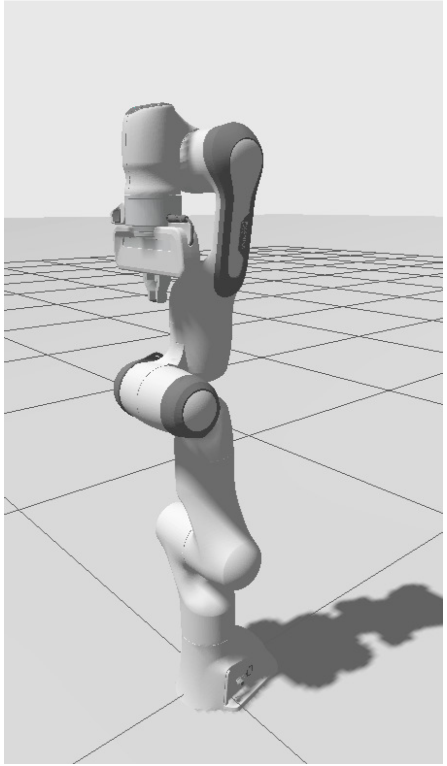
```
<placement_frame>pl_bottom</placement_frame>
```



Example: Robot arm assembly

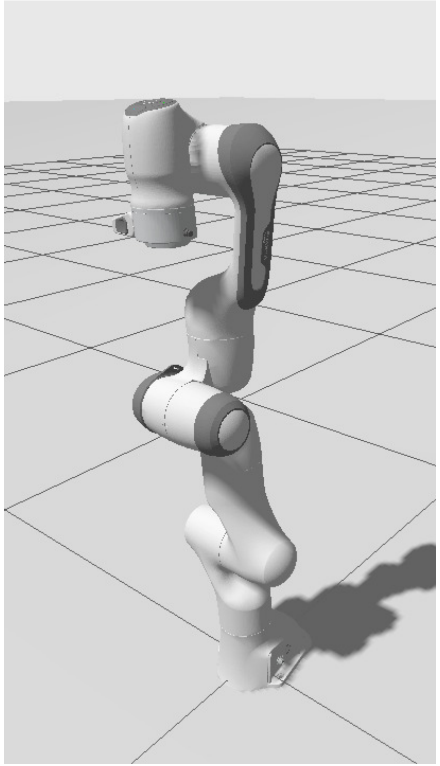
```xml
<model name="robot_arm_with_gripper">
  <joint name="fix_to_world" type="fixed">
    <parent>world</parent>
    <child>arm</child>
  </joint>
  <include>
    <uri>models/panda_arm</uri>
    <name>arm</name>
  </include>
  <joint name="attach_gripper" type="fixed">
    <parent>arm::gripper_mount</parent>
    <child>gripper</child>
  </joint>
  <include>
    <uri>models/panda_hand</uri>
    <name>gripper</name>
    <placement_frame>mount_point</placement_frame>
    <pose relative_to="arm::gripper_mount"/>
  </include>
</model>
```

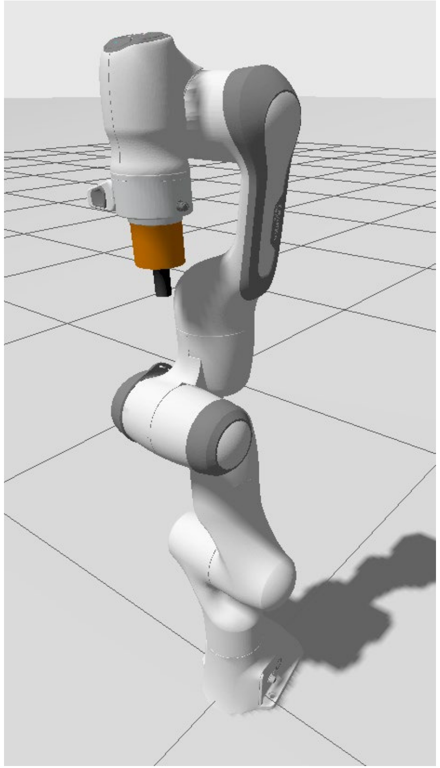




Example: Robot arm assembly

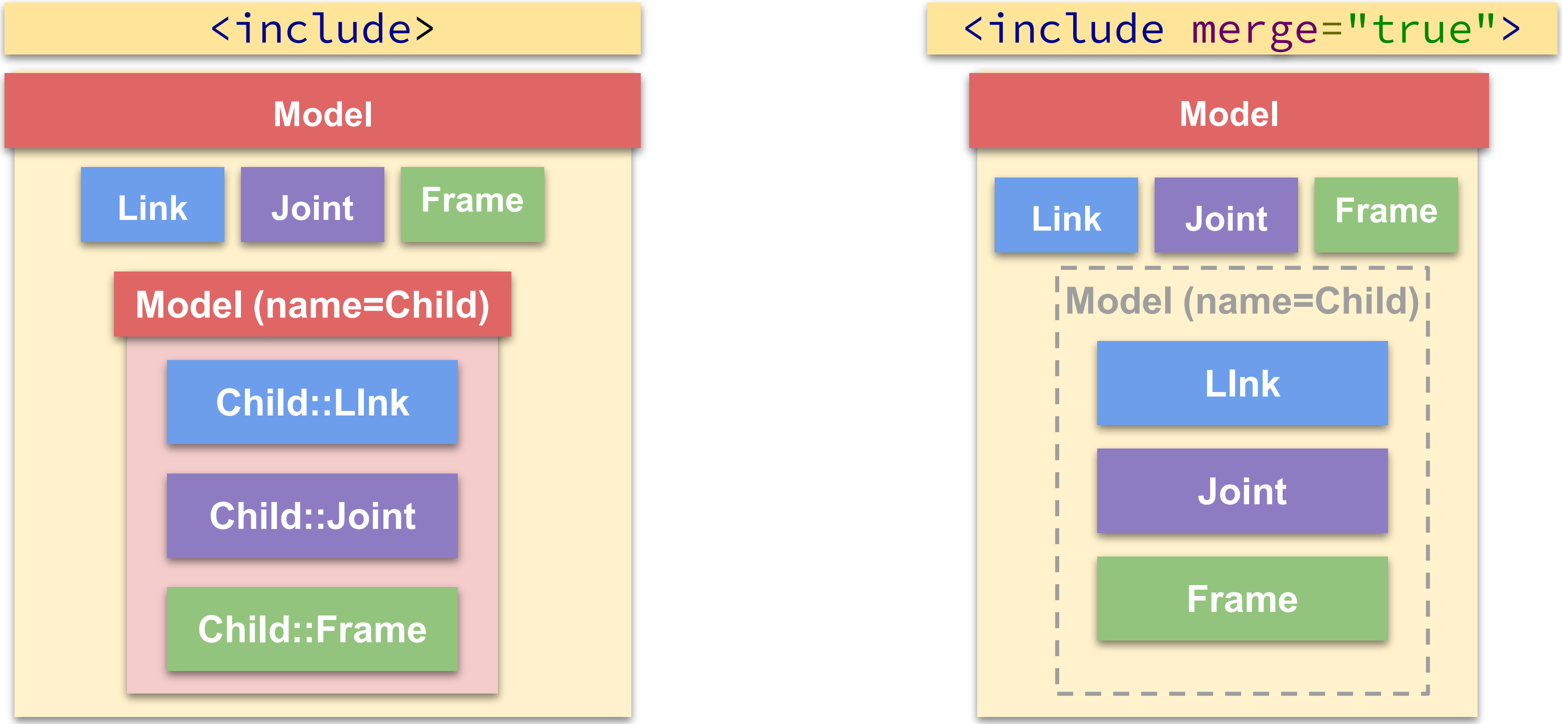
```
<model name="robot_arm_with_gripper">
  <joint name="fix_to_world" type="fixed">
    <parent>world</parent>
    <child>arm</child>
  </joint>
  <include>
    <uri>models/panda_arm</uri>
    <name>arm</name>
  </include>
  <joint name="attach_gripper" type="fixed">
    <parent>arm::gripper_mount</parent>
    <child>gripper</child>
  </joint>
  <include>
    <uri>models/simple_gripper</uri>     <!--changed -->
    <name>gripper</name>
    <placement_frame>mount_point</placement_frame>
    <pose relative_to="arm::gripper_mount"/>
  </include>
</model>
```





Merging for Composition

- Include other models without introducing a new scope



New ways of specifying pose

- Option to specify the rotation representation.
- Currently

```
<pose>{xyz}  {rpy_radians}</pose>
```

- New feature: Use Degrees

```
<pose degrees="true">{xyz}  {rpy_degrees}</pose>
```

- New feature: Rotation format

```
<pose rotation_format="euler_rpy">{xyz}  {rpy_radians}</pose>

<pose rotation_format="euler_rpy" degrees="true">{xyz}  {rpy_degrees}</pose>

<pose rotation_format="quat_xyzw">{xyz}  {quat_xyzw}</pose>
```

http://sdformat.org/tutorials?tut=better_pose_proposal

Fluid Added Mass

- The effective mass of a body increases when moving in a fluid
- Newton's second law: $$(\mathbf{M} + \mu)\ddot{\mathbf{x}} = \sum \mathbf{F}(\mathbf{x}, \mathbf{t})$$
  - where M is the body mass inertia matrix, μ is the fluid added mass matrix
- New `<fluid_added_mass>` element added under `//link/inertial/`
- Available in SDFormat 1.10

https://bit.ly/added_mass_proposal

Mimic Joint Actuation Constraint

- Gearbox joint type provides equivalent functionality but requires defining more joints and intermediate links
- Mimic constraint simplifies the definition

```
<link name="rack"/>
<link name="pinion"/>
<joint name="pinion_joint" type="revolute">
  <parent>world</parent>
  <child>pinion</child>
  ...
</joint>
```

```
<joint name="rack_joint" type="prismatic">
  <parent>world</parent>
  <child>rack</child>
  <axis>
    <xyz>1 0 0</xyz>
    <mimic joint="pinion_joint">
      <multiplier>0.02</multiplier>
      <offset>0.0</offset>
      <reference>0.0</reference>
    </mimic>
  </axis>
</joint>
```

http://sdformat.org/tutorials?tut=mimic_proposal

Automatic Moments of Inertia Calculations

- Bad Inertia values cause bad simulation
- Default Mass Matrix
  - mass = 1.0 Kg
  - Diagonal Elements = (1, 1, 1)
- 2 major workflows for computing inertial properties
  - Using CAD software
  - Mesh Processing Software, such as MeshLab

Automatic Moments of Inertia Calculations

- SDFormat 1.11 introduced `<inertial auto="true"/>`

```
<link name="robot_link">
  <inertial auto="true"/>
      <collision name="capsule_collision">
        <density>2710</density>
        <geometry>
          <capsule>
            <radius>0.5</radius>
            <length>0.7</length>
          </capsule>
        </geometry>
…
</link>
```

Summary

- Offline converters (USD and MJCF)
- Python API
- ROS 2 support
- SDFormat new features
  - SDFormat 1.7: Frame semantics, Parameter passing (experimental)
  - SDFormat 1.8: Composition (nested models)
  - SDFormat 1.9: New ways of specifying pose (angles in degrees, quaternions)
  - SDFormat 1.10: Merge-includes, `<joint>` in world, Fluid added mass
  - SDFormat 1.11: Automatic computation of moments of inertia

Feedback:

- We invite everyone to try it and test it! We are happy to receive your feedback

SONY

ROS / ROS 2 with Kubernetes and KubeEdge

Oct.19th.2023

ROSCon 2023 @ New Orleans, US

Agenda

- Who are we?
- Background
- Problems
- Goals / Requirements
- Kubernetes
- KubeEdge
- Sample Deployment
- What’s missing? Next-gen proposal
- Community

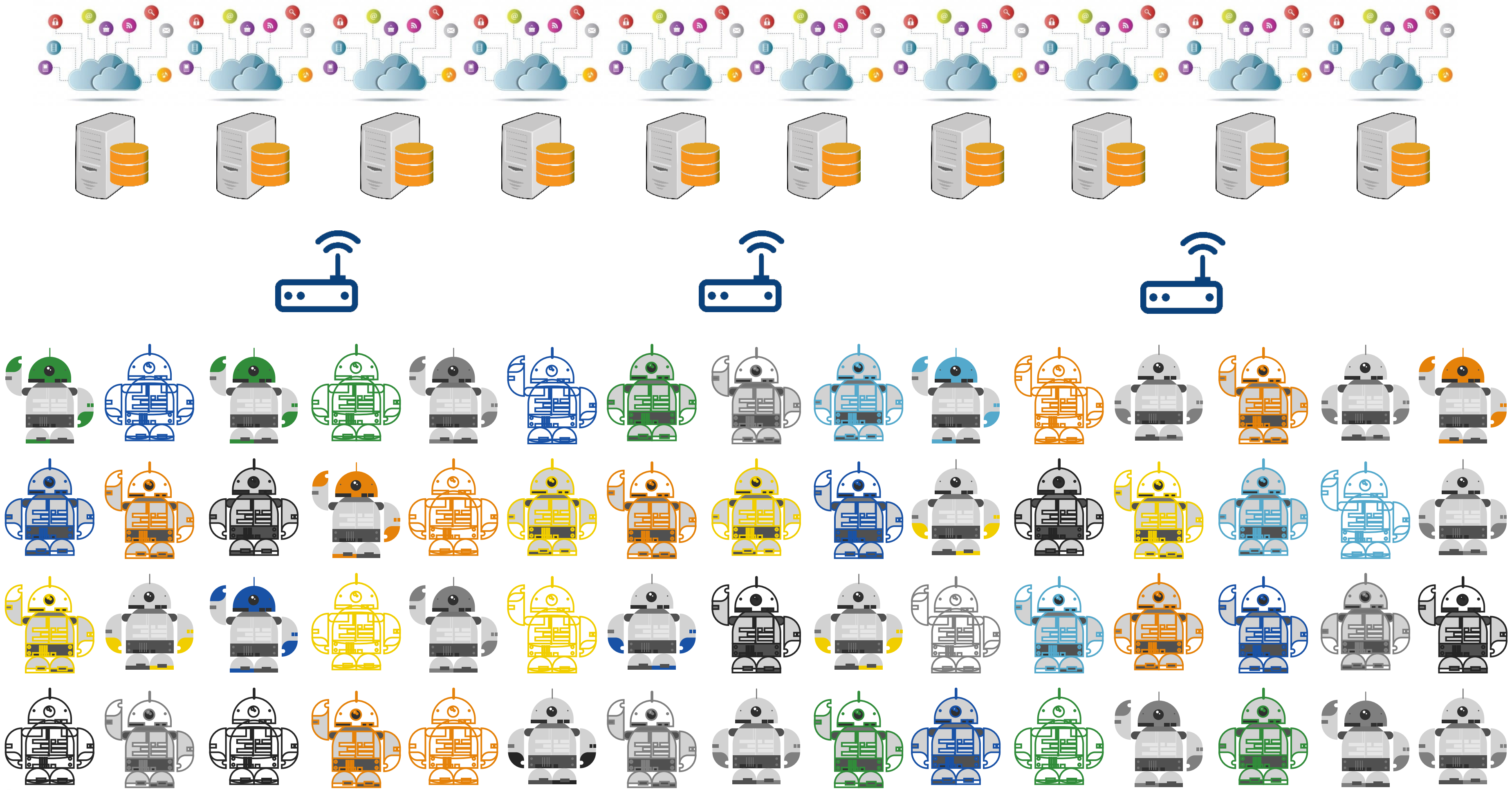
Who are we?

- Tomoya Fujita (Presenter)
  - Software Engineer, Sony R&D US Laboratory
  - ROS TSC (Technical Steering Committee)
  - KubeEdge SIG Robotics Chair
  - fujitatomoya@github , tomoyafujita@linkedin

- Co-Authors
  - Yin Ding (Engineering Manager, Google)
    - KubeEdge TSC, Co-Founder of KubeEdge Project
    - Leading the Kubernetes Hardening team
  - Kevin Wang (Lead of Cloud Native Open Source Team, Huawei)
    - KubeEdge TSC, Co-Founder of KubeEdge Project
    - CNCF Ambassador, TOC contributor
  - Fei Xu (Senior Engineer, Huawei)
    - KubeEdge TSC, Maintainer

Background

- Broad use cases.
- Distributed and Connected System.
- Collaborative and Orchestrated Application.
- Circulatory Functioning System and Development
- Specific Hardware Acceleration.
- Security. (Device, Data, Network)

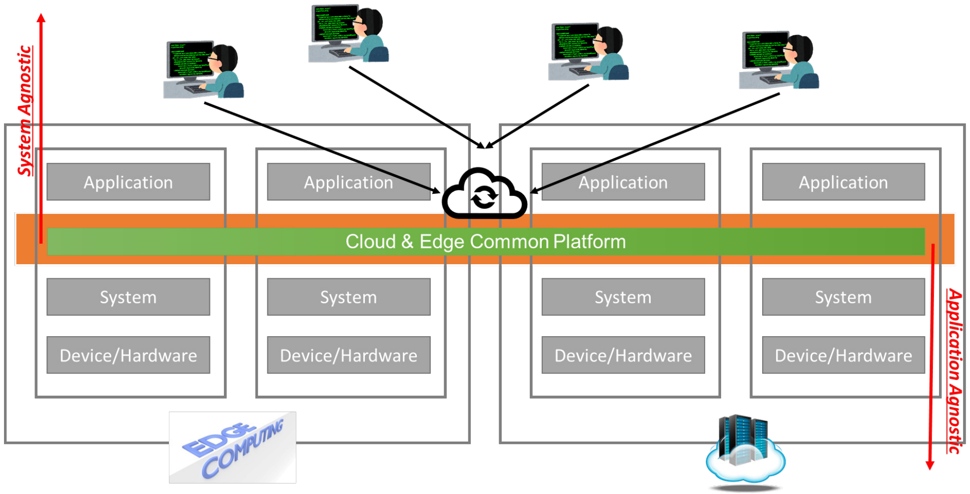


What is the pain?

- Platform Dependencies.
- Proprietary hardware support.
- Application Modularity.
- System and Security Integration.
- Application Specific Network Bridge.
- Application Developer Friendly.

Goal / Requirements

- Flexible Application Deployment.
- Zero Trust Security Support.
- Application Agnostic Network Configuration.
- Extend Device Capability.
- System Global Observability.
- Platform Agnostic Device Abstraction.



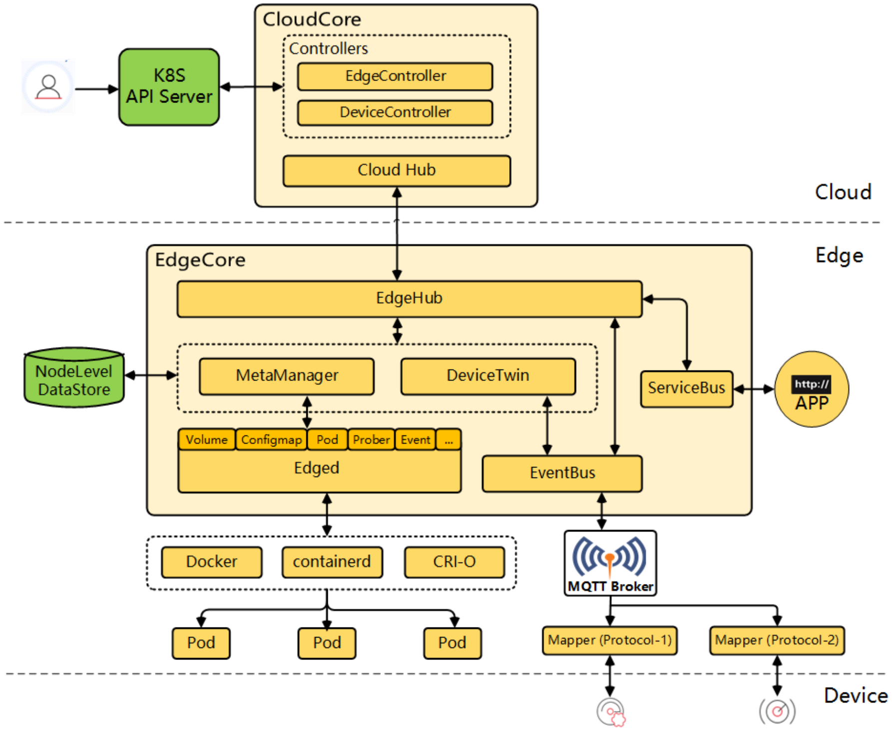
Kubernetes (Service Mesh)

- **Application Deployment and Orchestration.**
- **Device Capability and Label Control.**
- Custom Resource Extension.
- **Auto- Scaling and Healing.**
- **Roll Up/Down, Canary Test.**
- **Role Based Access Control.**
- **Device-Plugin / Container Device Interfaces.**
- **Container Network Interfaces.**
- Traffic Management.
- Observability.
- Security Policy.

KubeEdge

is built upon Kubernetes and provides core infrastructure support for networking, application deployment and metadata synchronization between cloud and edge.

- Cloud-Edge Coordination
- Edge Computing
- Edge Autonomy
- Simplified Deployment
- Kubernetes-native Support
- Resource Efficient



https://kubeedge.io/docs/

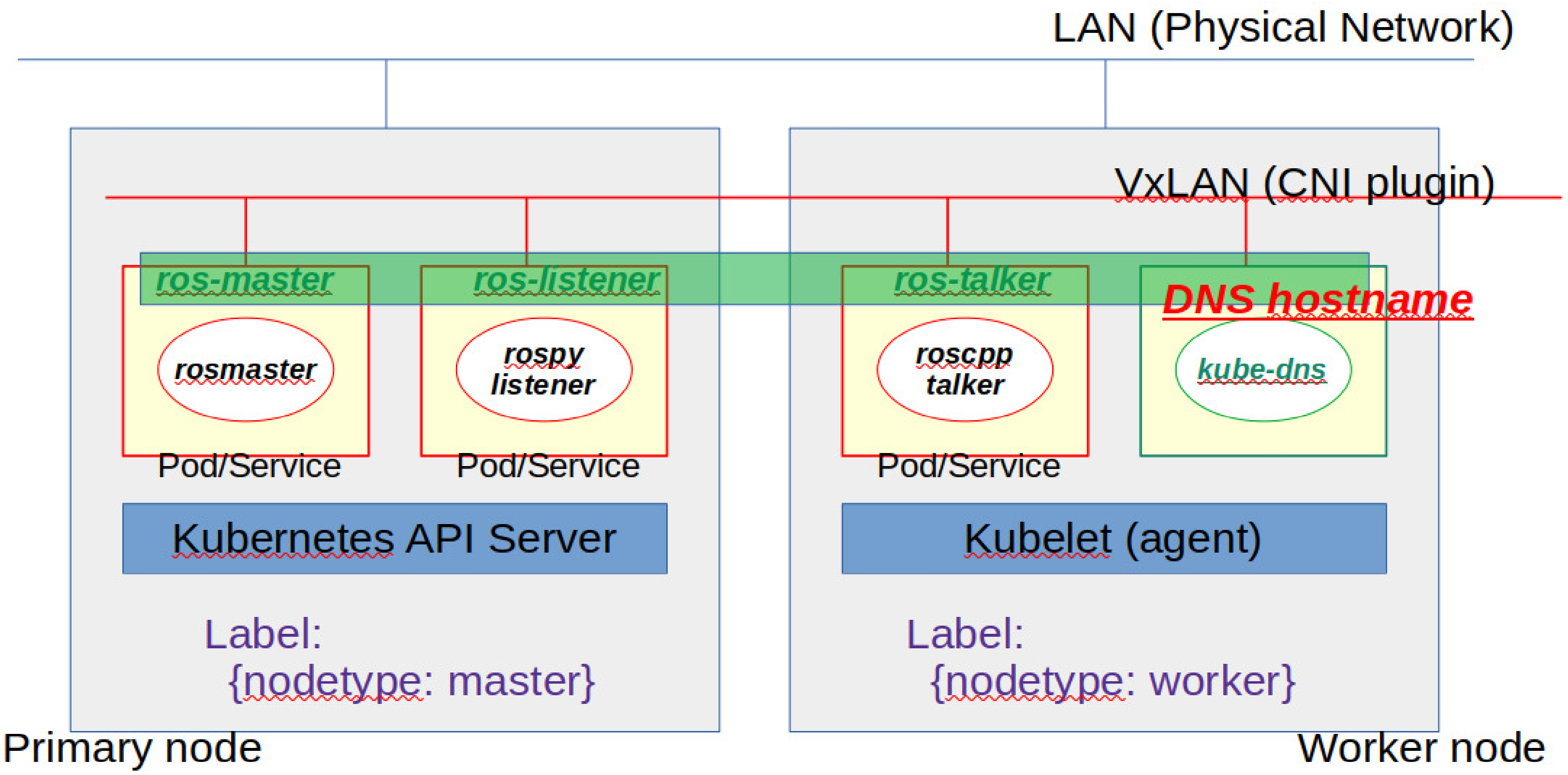
Sample Deployment

Theory is good, but please see how it works in the flesh!

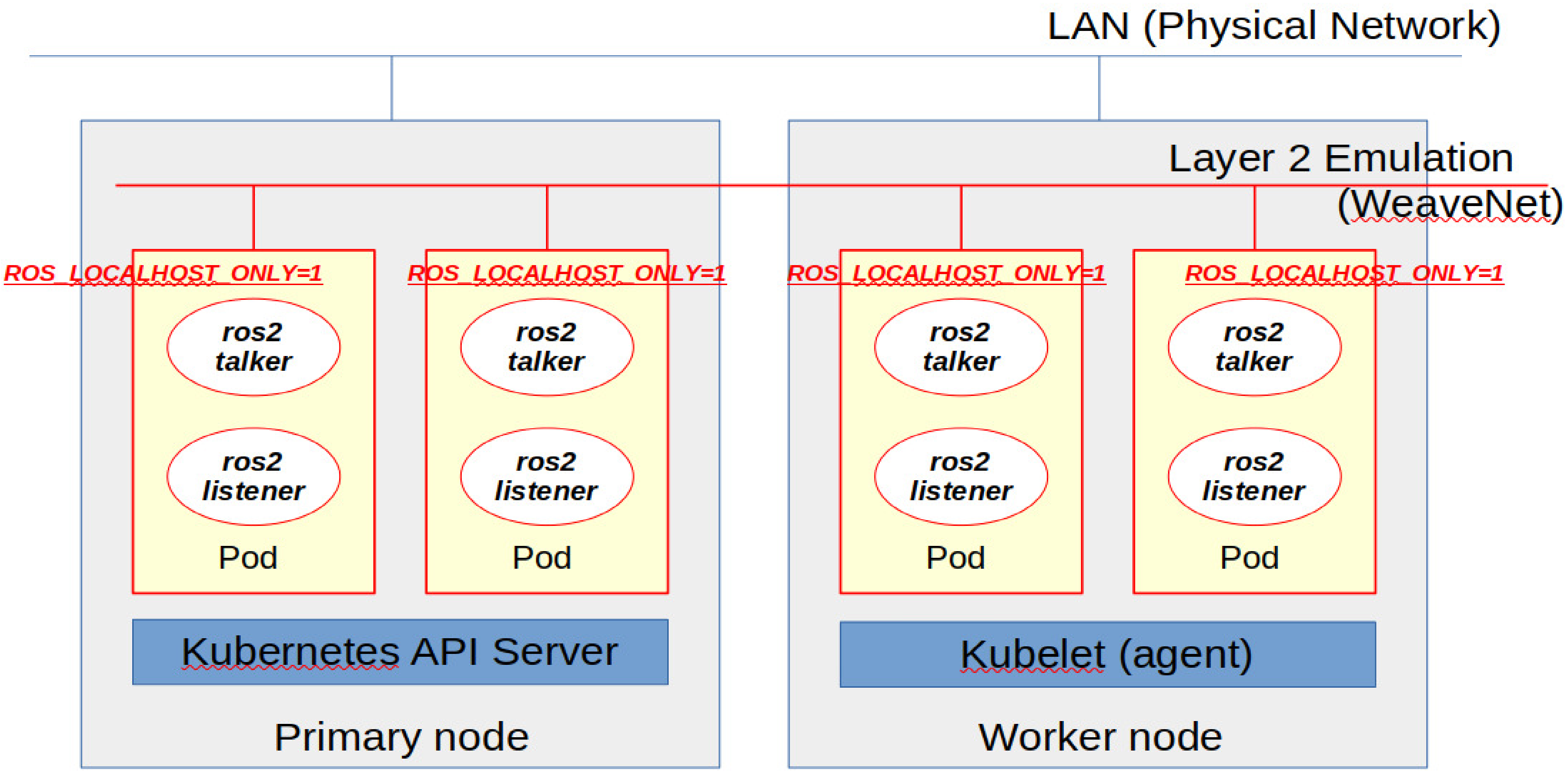
ROS Kubernetes Tutorials

Contribution(Issues/PRs) always welcome!

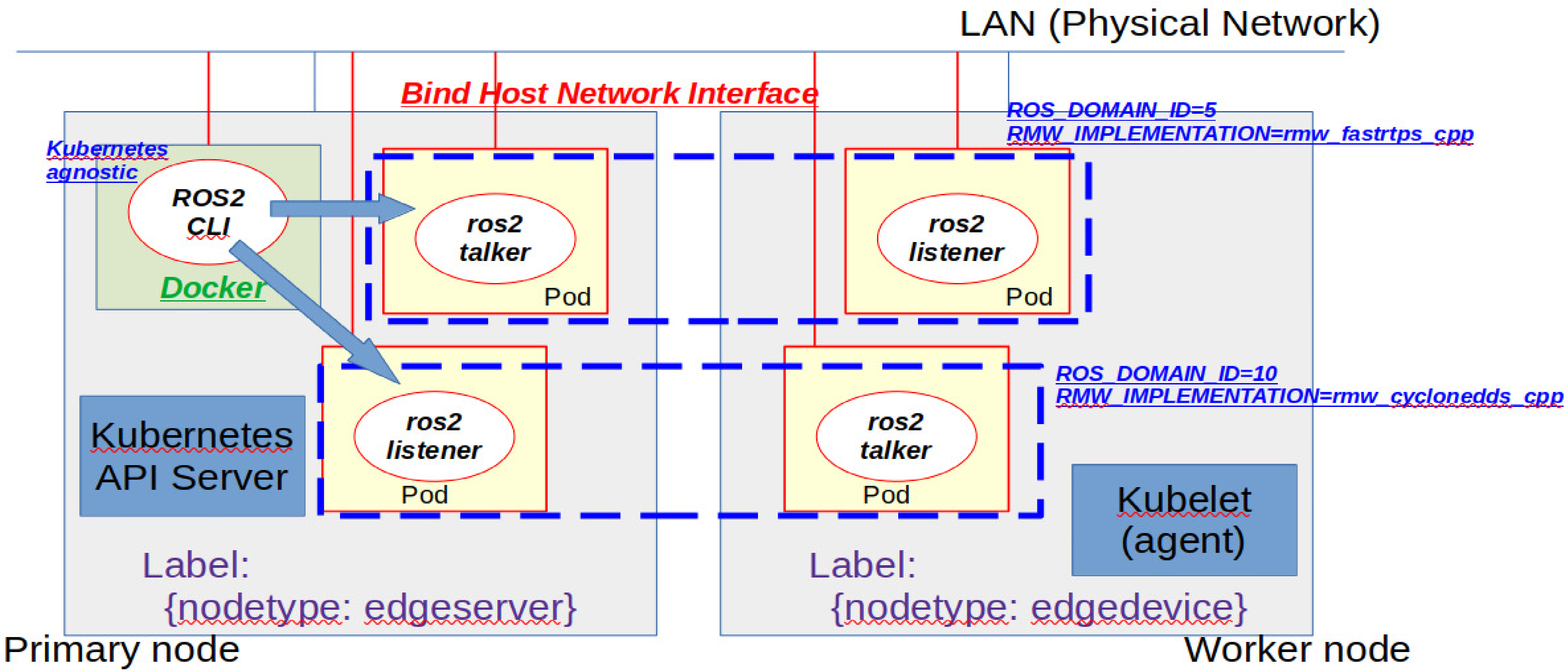
ROS Multi-Node Deployment



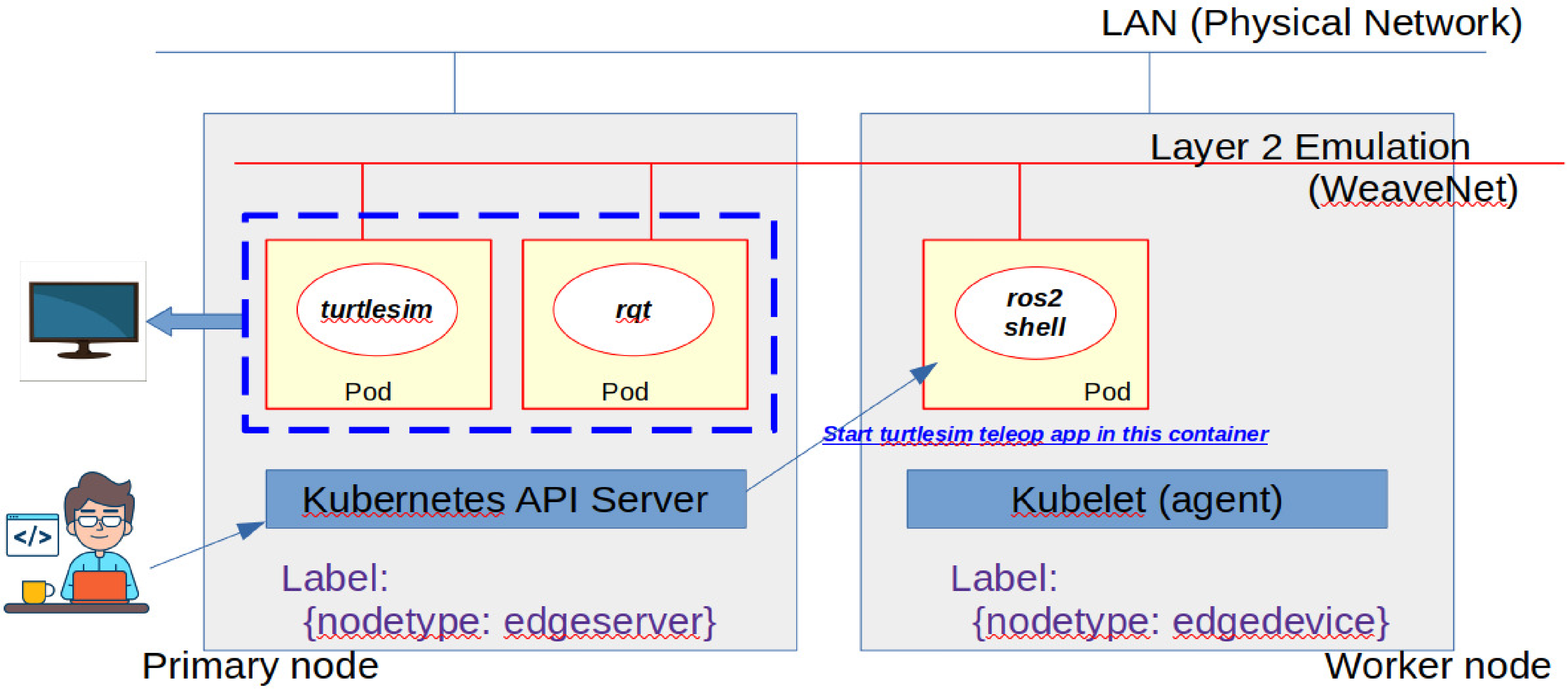
ROS 2 Localhost Only



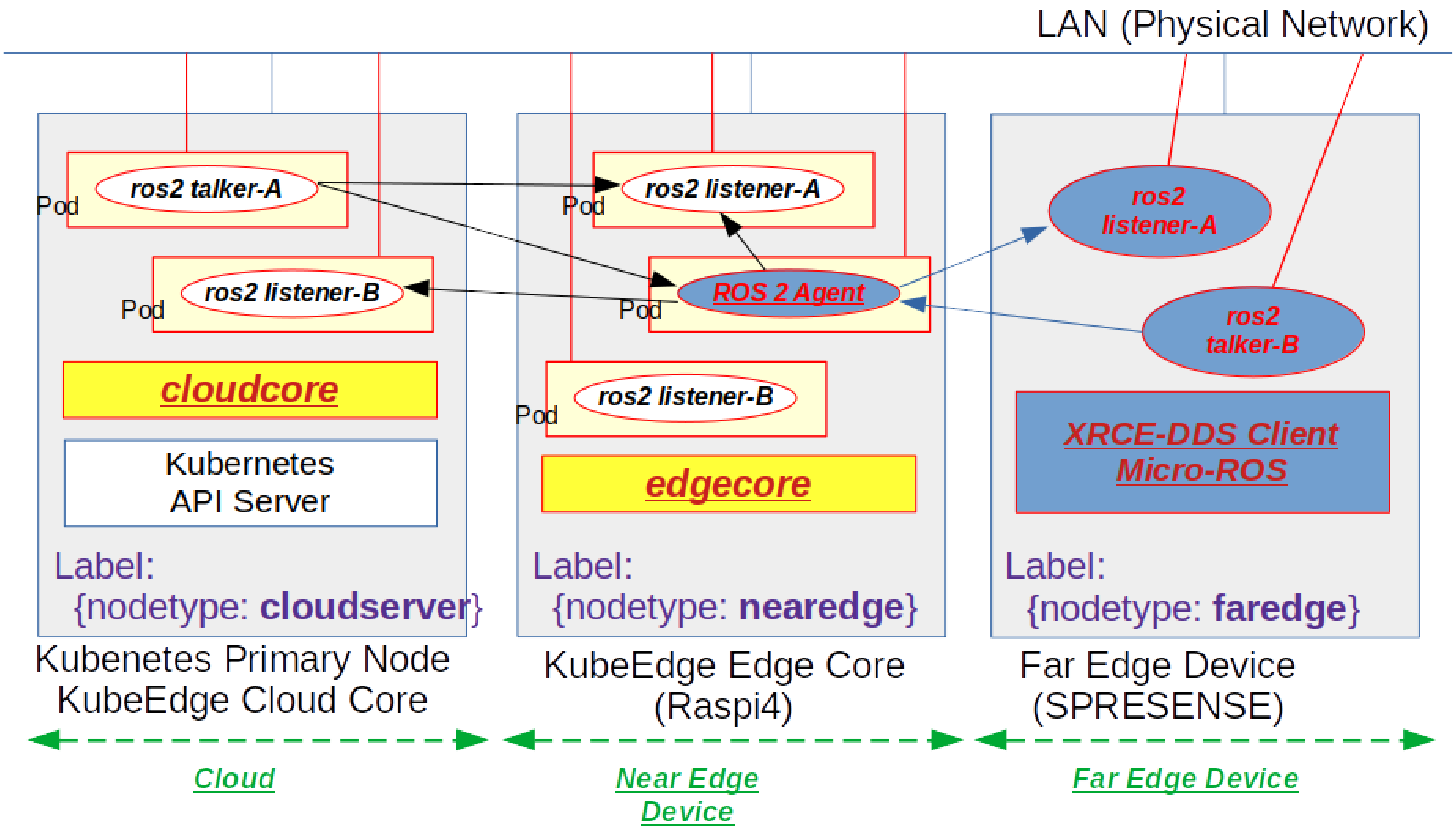
ROS 2 Logical Partition / Multiple RMW Implementation



ROS 2 Deployment Intermediate

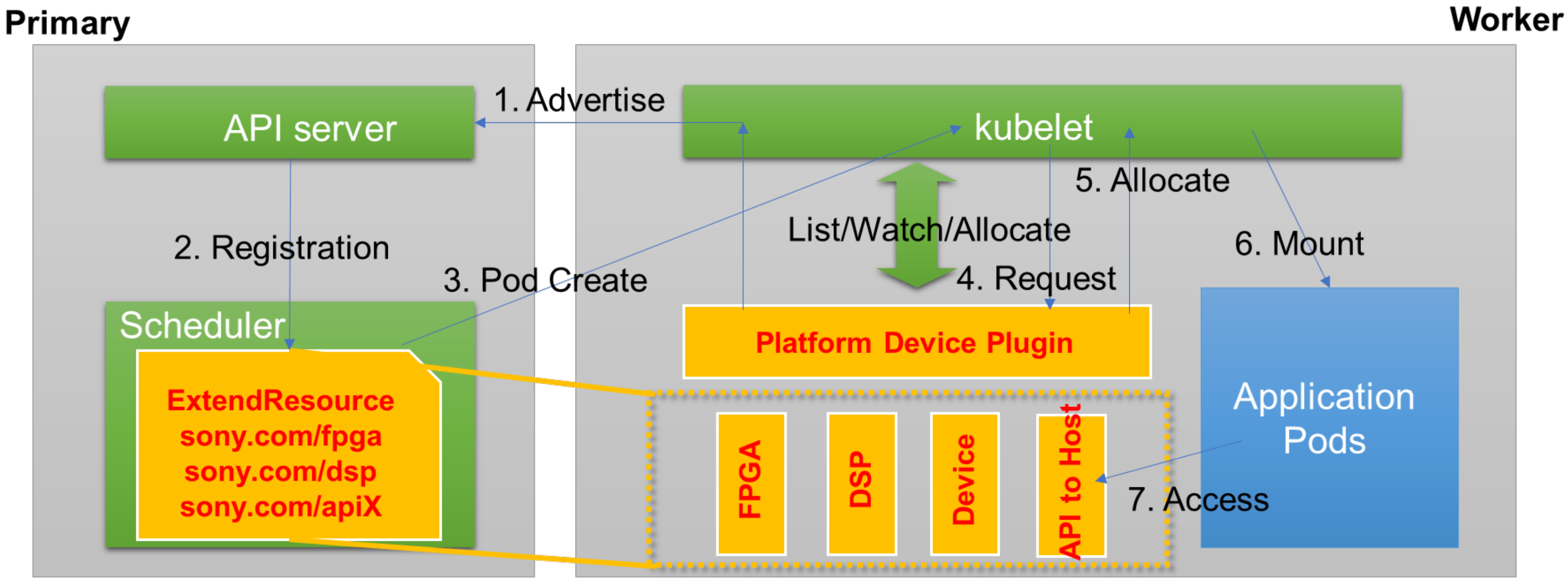


ROS 2 / Micro-ROS with KubeEdge (W.I.P)

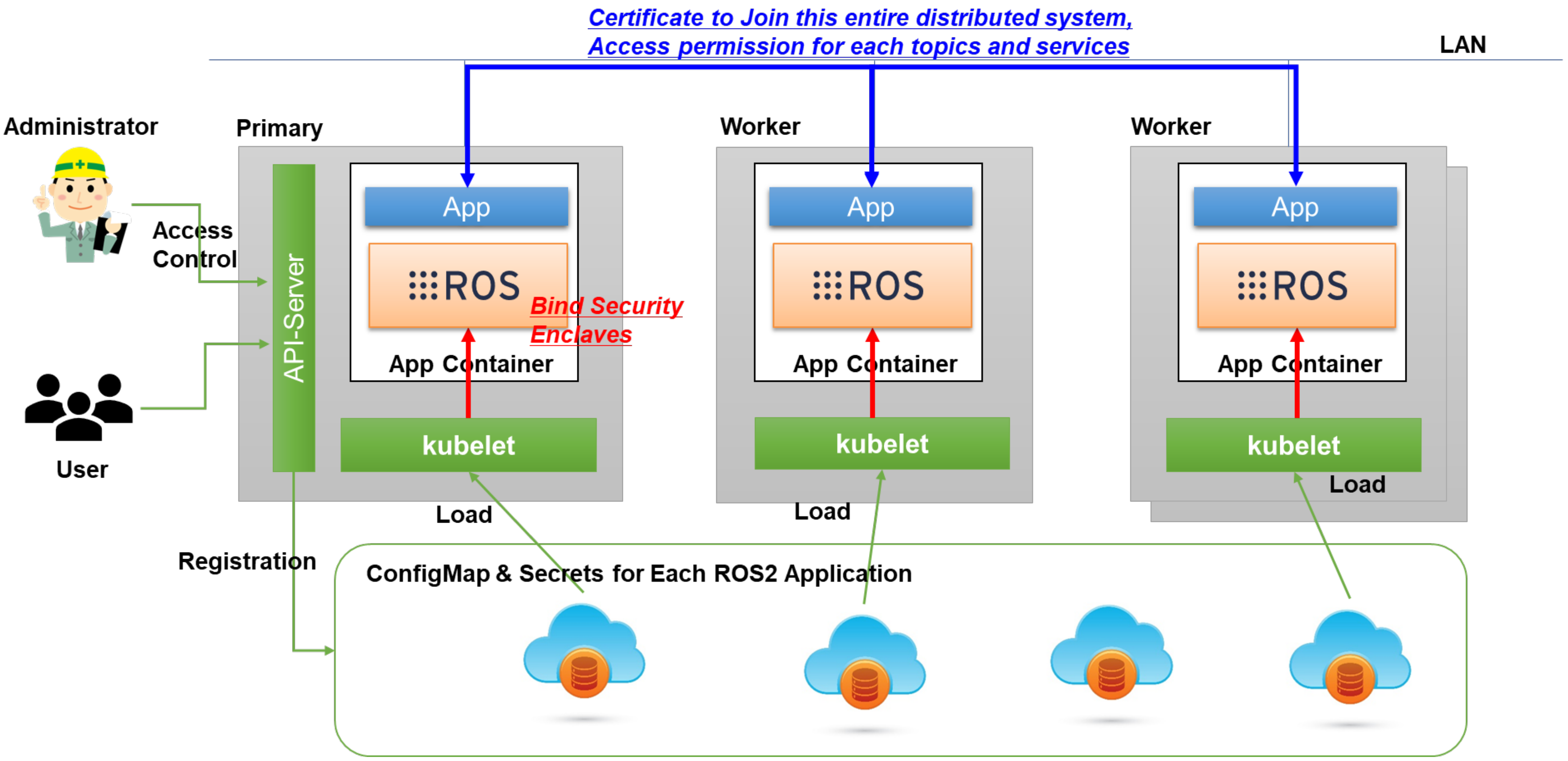


Device-Abstraction (Device-Plugin, Container Device Interface)

- Kubernetes Custom Resource Definition, that allows us to plugin vendor specific hardware and device to the containers.
- After advertising the custom resource to Kubernetes, Kubernetes controls those resources with workload based on application requirements.



Support SROS 2 security enclaves via ConfigMap



What's missing? Proposals?

- ~~Device Abstraction Enhancement~~
    - ~~KEP-3162: Add Deallocate and PostStopContainer to Device Manager API~~
    - ~~Add CDI devices to device plugin API resolved this issue.~~
- KubeEdge CNI support (e.g edgemesh, Cilium)
- More Edge Optimization / Configurable Options for Resource Constrained Device Support

- **Cloud-Native Robotics Management Solution**
    - `RoboDevOps` through Edge-Cloud Synergy
    - Cloud-Native Digital Twin for testing and data generation training
    - Robotics App Development Friendly
    - Cloud Robotics Custom Resource Definition and Operator Proposal
    - Edgemesh: adaptive cross-edge and edge-cloud data plane support
    - VSLAM algorithm with KubeEdge
    - Building a Robot-Oriented Intelligent Monitoring System

Community

Kubernetes
IoT Edge WG

KubeEdge
SIG Robotics

ROS Kubernetes
Tutorials

SONY