eclipse muto

an adaptive framework and a runtime platform for dynamically composable model-driven software stacks for ROS

Dr. Naci Dai
@nacidai

Deniz Memiş
@d8niz

eclipse muto

an adaptive framework and a runtime platform for dynamically composable model-driven software stacks for ROS

Muto Agent

Muto Composer

Muto Dashboard

F1Tenth Sandbox

Muto LiveUI

Challenges

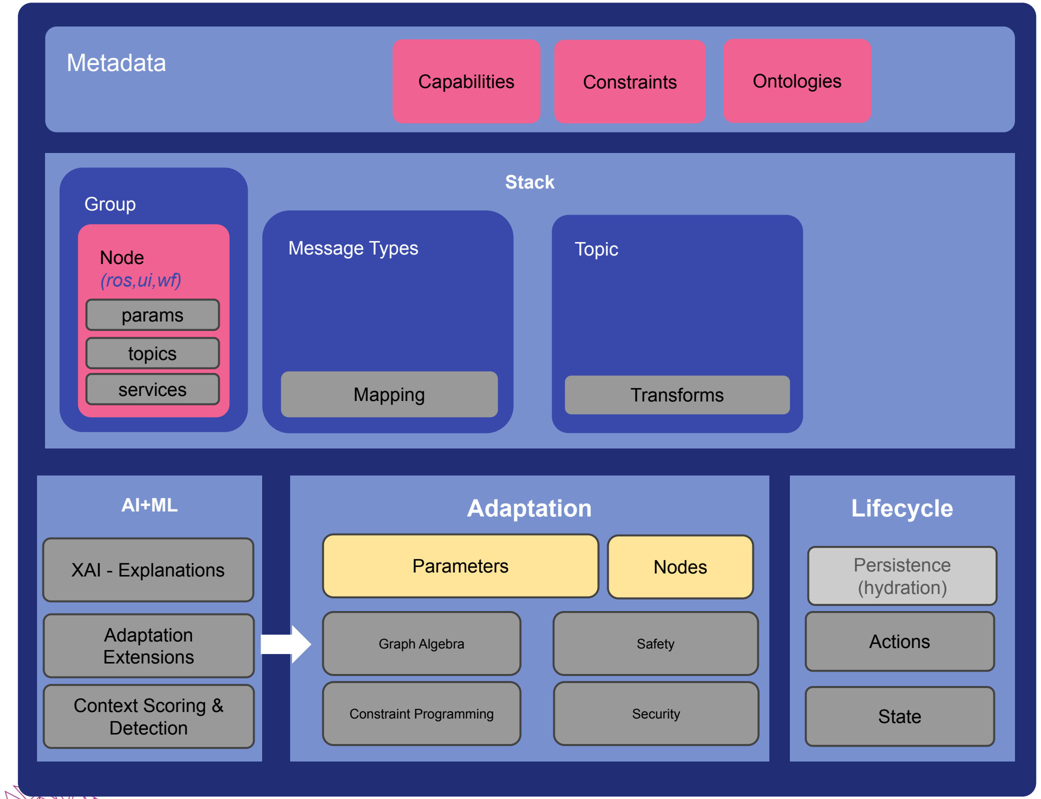
Bottom up
Closed & proprietary
Builtin biases
Hardwired & static
Built for a purpose

Response

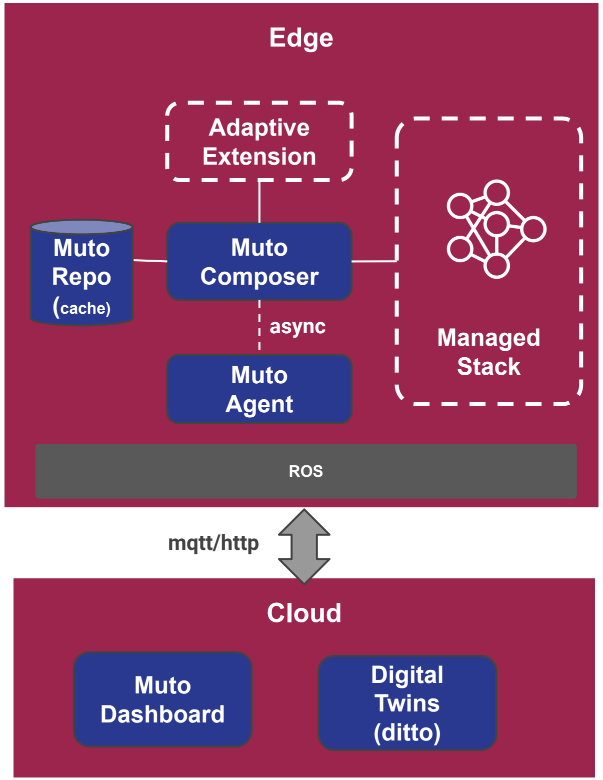
Model driven
Open
Contextual
Dynamic & Live
Adaptive & Extensible

icons made by bqlqn from flaticon.com
Icons made by freepik from flaticon.com

Overview



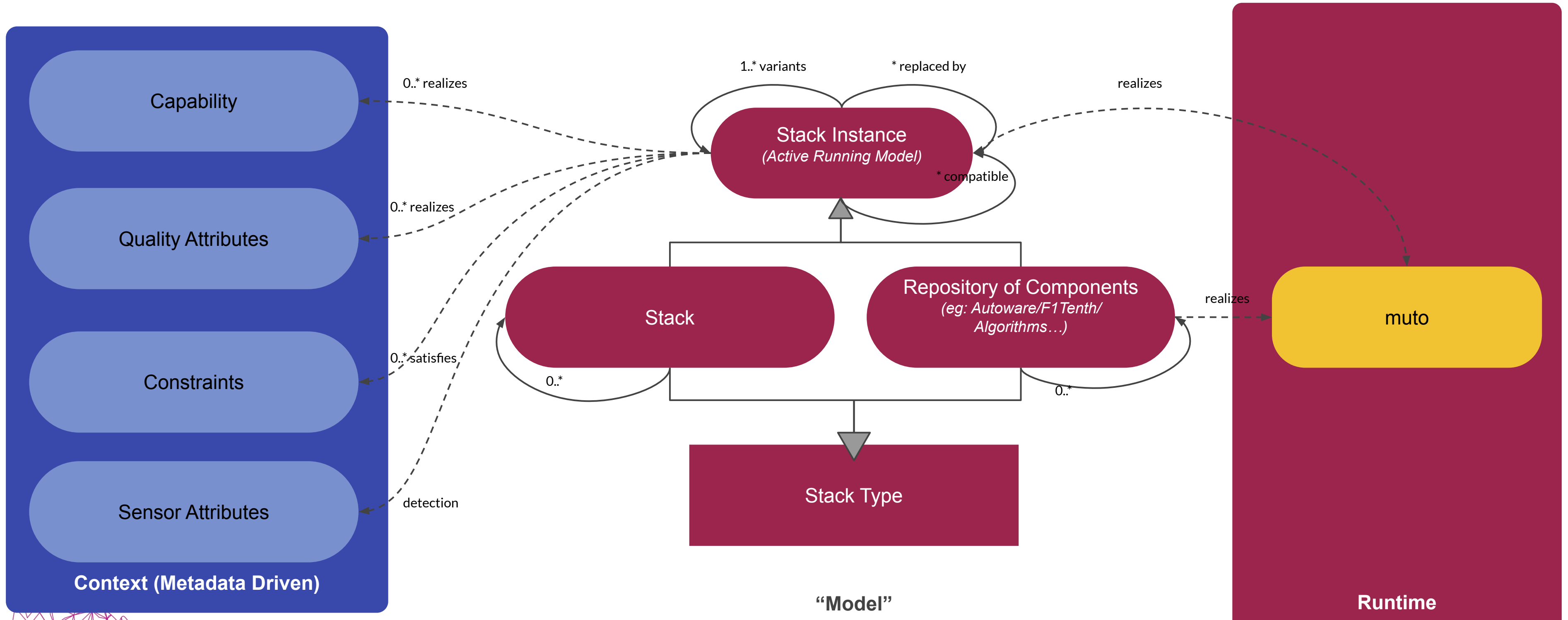
muto runtimes



muto

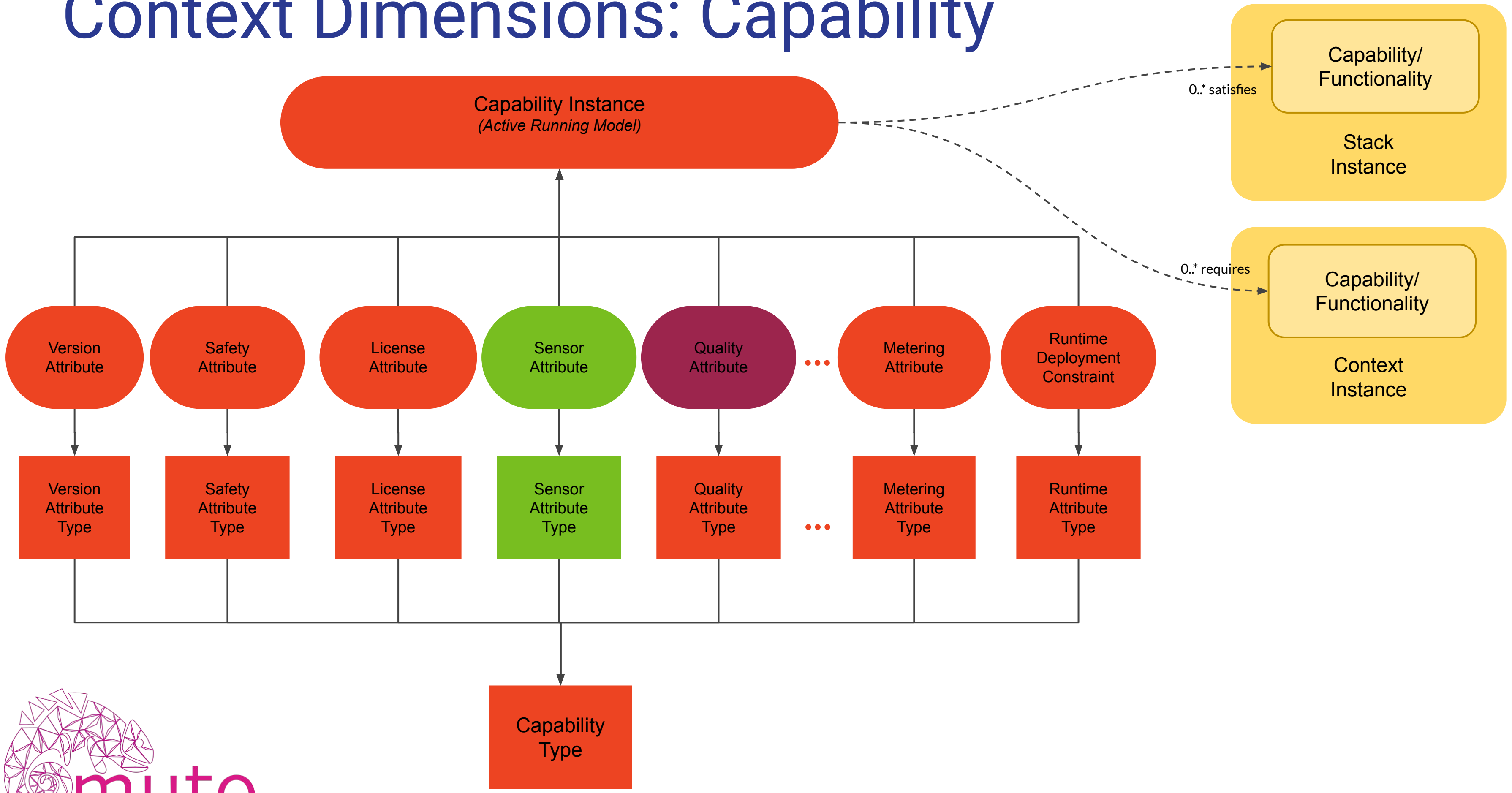
icons made by bqlqn from flaticon.com
Icons made by freepik from flaticon.com

Contextual Adaptation



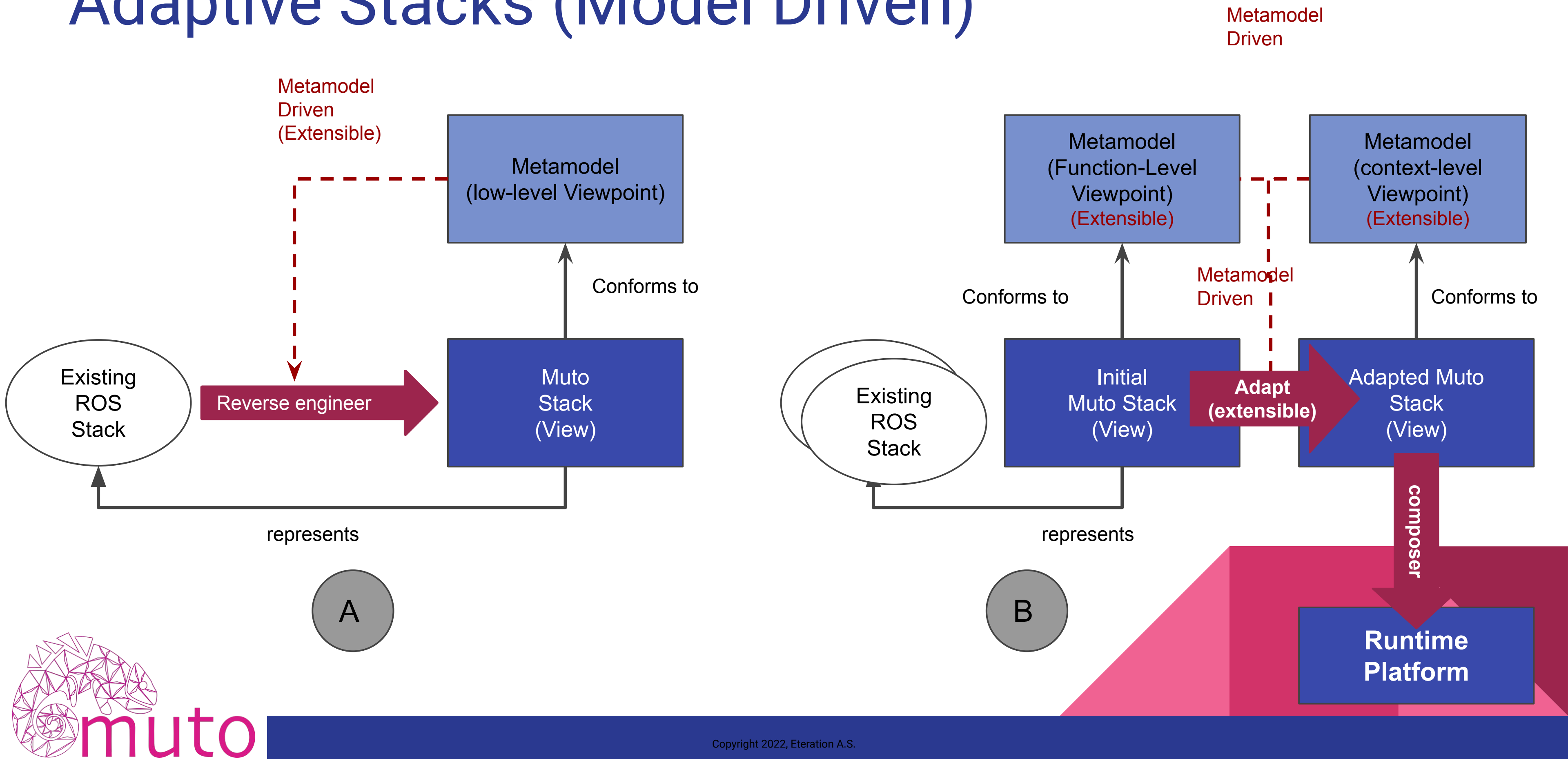
muto

Context Dimensions: Capability

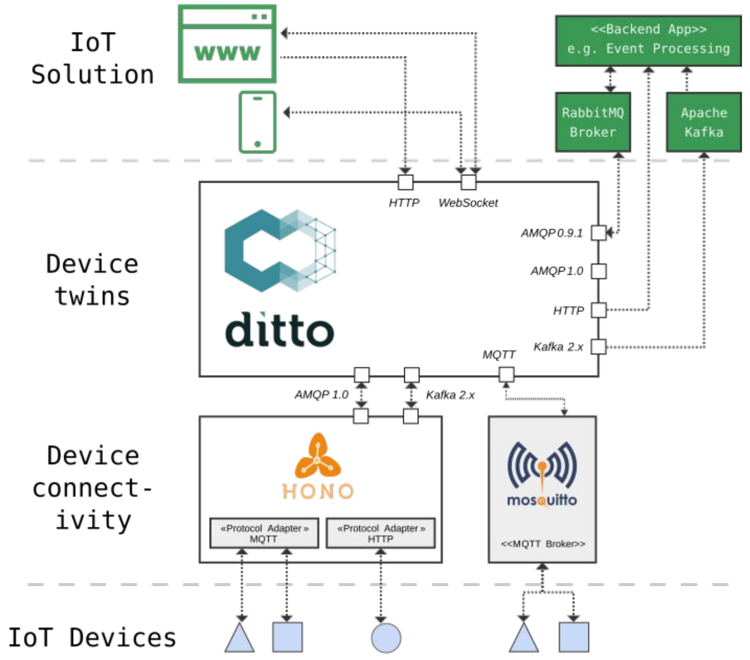


muto

Adaptive Stacks (Model Driven)


Metamodel Driven
Metamodel Driven (Extensible)
Metamodel (low-level Viewpoint)
Conforms to
Existing ROS Stack
Reverse engineer
Muto Stack (View)
represents
A
Metamodel (Function-Level Viewpoint) (Extensible)
Metamodel (context-level Viewpoint) (Extensible)
Conforms to
Metamodel Driven
Conforms to
Existing ROS Stack
Initial Muto Stack (View)
Adapt (extensible)
Adapted Muto Stack (View)
composer
represents
B
Runtime Platform
muto

muto uses ditto for stack and vehicle digital twins*



This diagram is reproduced from https://www.eclipse.org/ditto

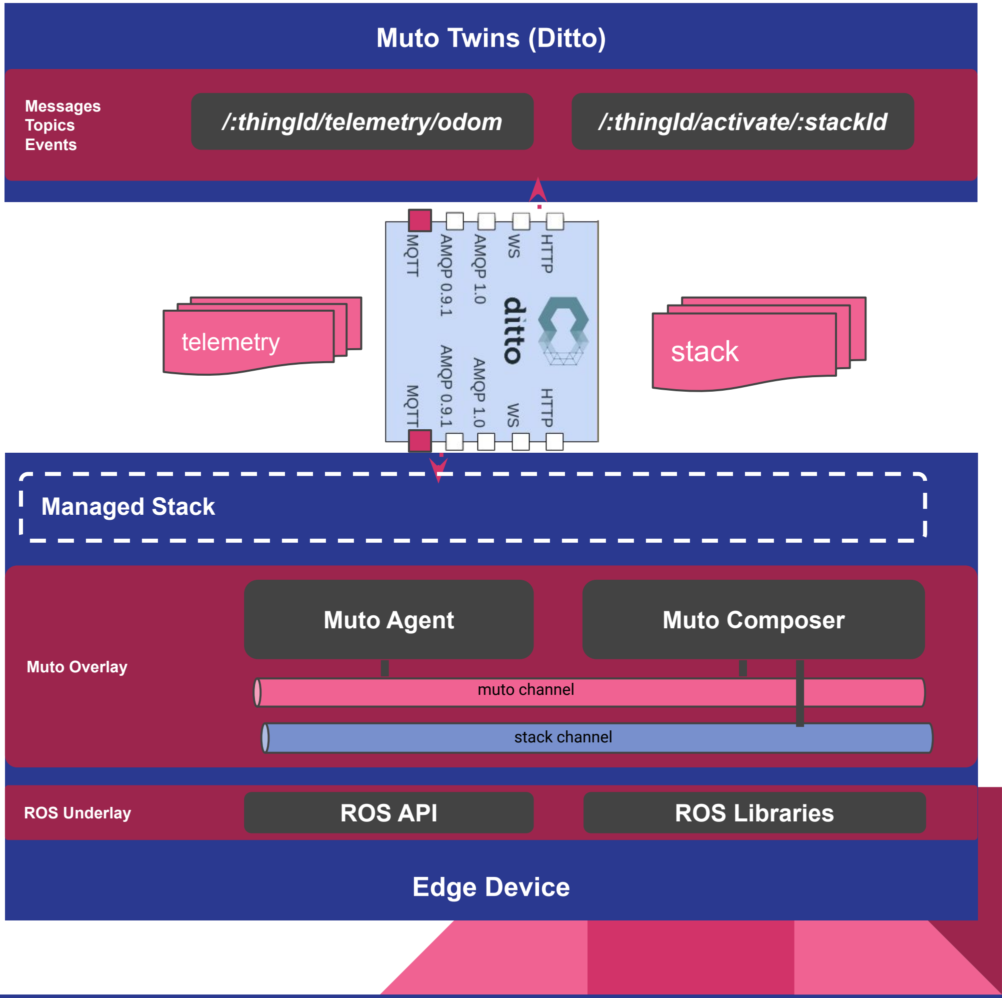
stack

vehicle

*A digital twin is a virtual representation that serves as the real-time digital counterpart of a physical object or process.

Muto Agent

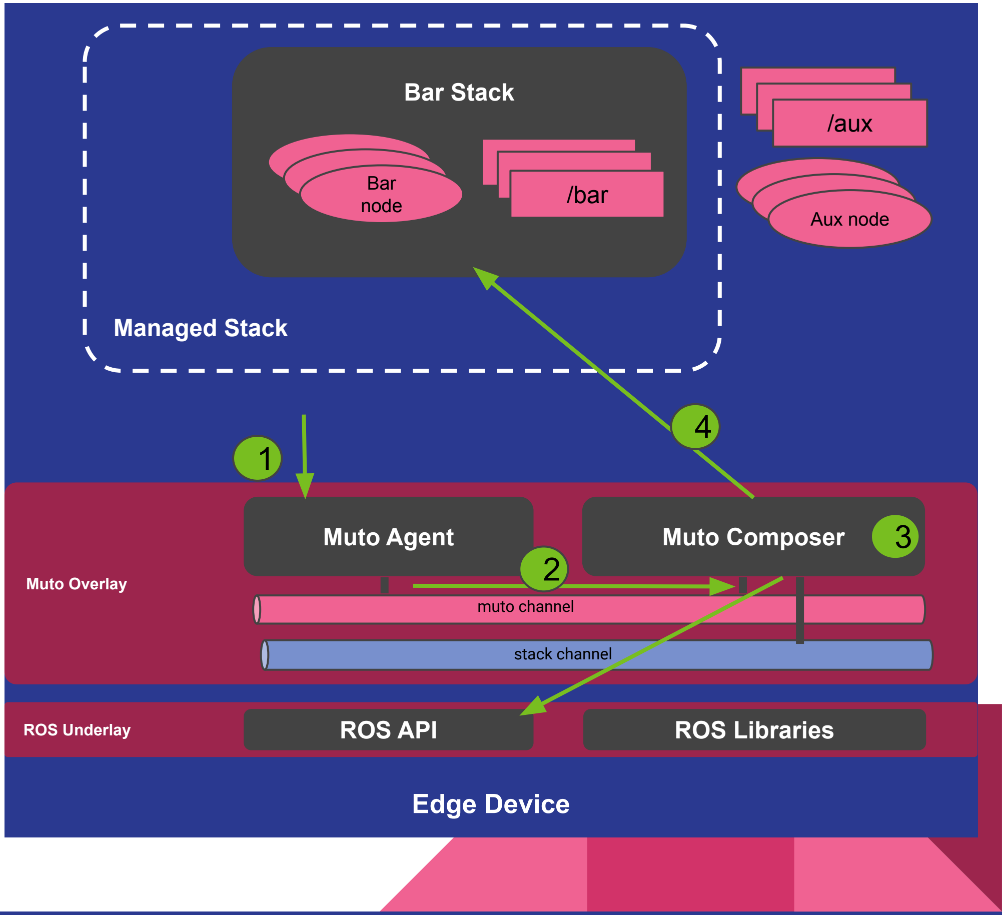
- A runtime ROS component
- Acts as a gateway for remote management capabilities
    - i.e. eclipse ditto twins
- Bidirectional (cloud <-> edge)
    - Relays messages to *Composer* for stack lifecycle management
    - Streams edge device information
- Asynchronous



muto

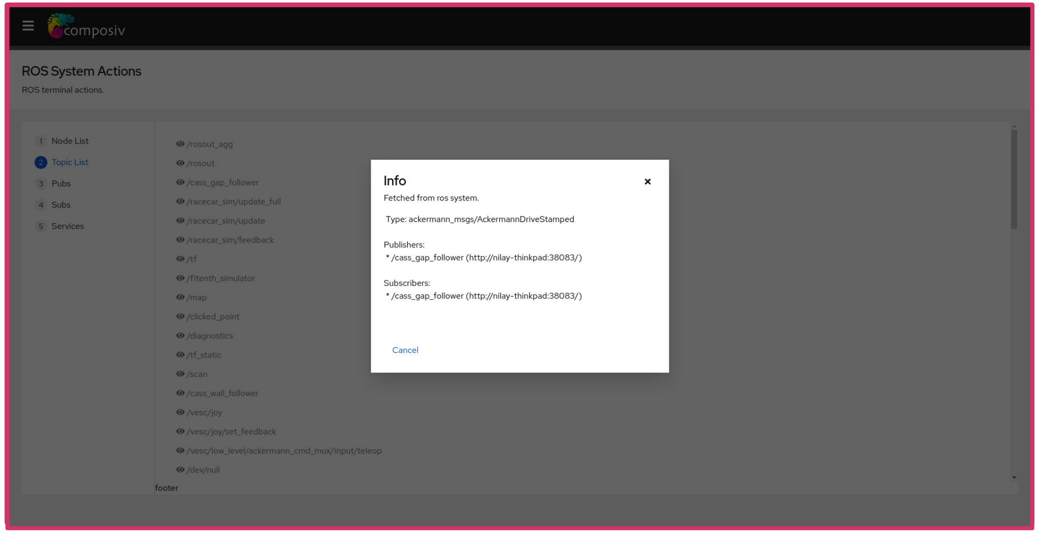
Muto Composer

- A runtime ROS component
- Composes and manages life cycle of ROS nodes defined by the stack model
- Node graph algebra
  - Stack introspection
  - Stack diff
  - Node Lifecycle actions
  - Interact with Param server



Muto Dashboard

- Centralized management
  - *'A dashboard to rule them all'*
- Extensible and modular
  - Composed of pluggable µFrontends
  - (Muto LiveUI)
- Example modules
  - ROS Actions
  - Vehicles
  - Stack
    - Remote control
  - ROS Graph



ROS System Actions

ROS terminal actions.

/rosout_agg
/rosout
/cass_gap_follower
/racecar_sim/update_full
/racecar_sim/update
/racecar_sim/feedback
/tf
/f1tenth_simulator
/map
/clicked_point
/diagnostics
/tf_static
/scan
/cass_wall_follower
/vesc/joy
/vesc/joy/set_feedback
/vesc/low_level/ackermann_cmd_mux/input/teleop
/dev/null

Info

Fetched from ros system.

Type: ackermann_msgs/AckermannDriveStamped

Publishers:
* /cass_gap_follower (http://nilay-thinkpad:38083/)

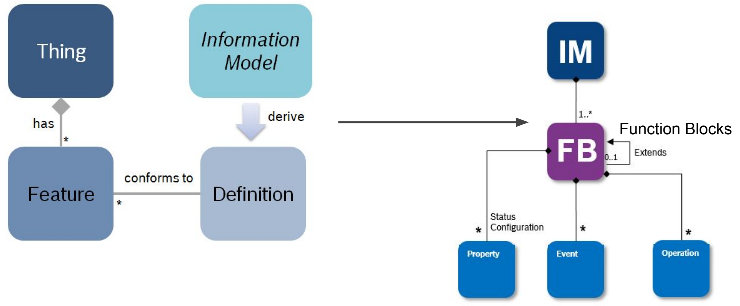
Subscribers:
* /cass_gap_follower (http://nilay-thinkpad:38083/)

Vehicle

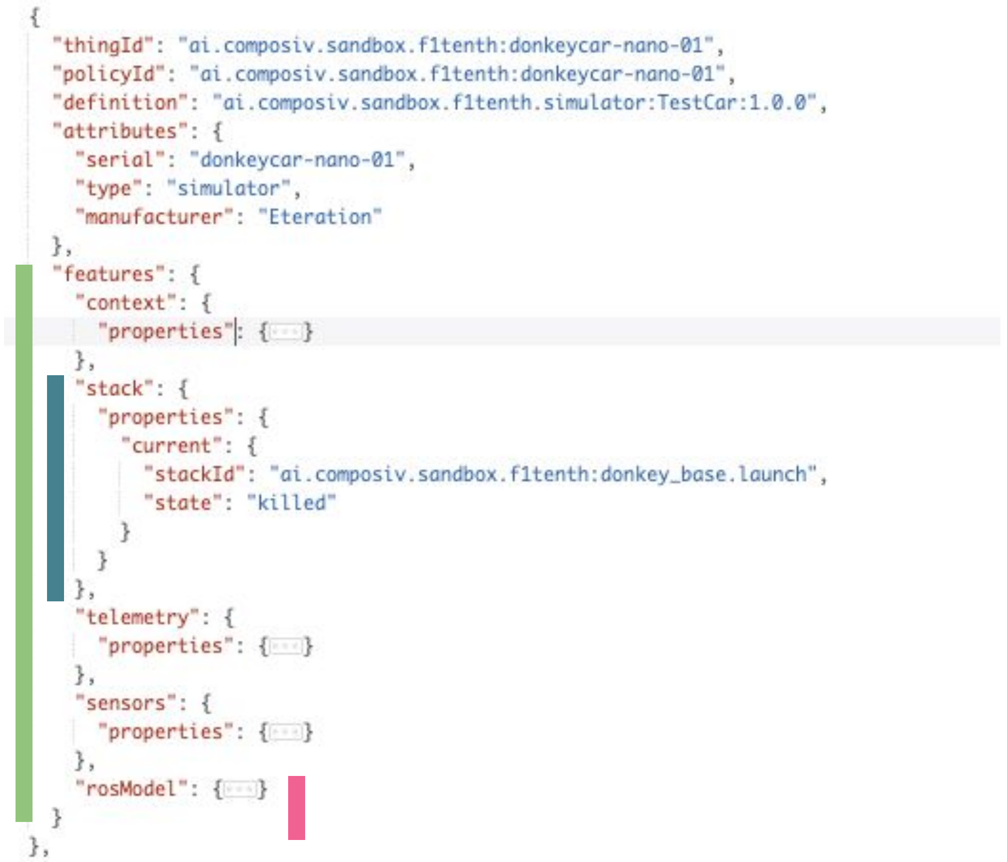
Associate a stack with a vehicle and manage its lifecycle

ROS State (nodes/topics/params)

Telemetry and sensor metadata



Class diagram for things from
https://www.eclipse.org/ditto

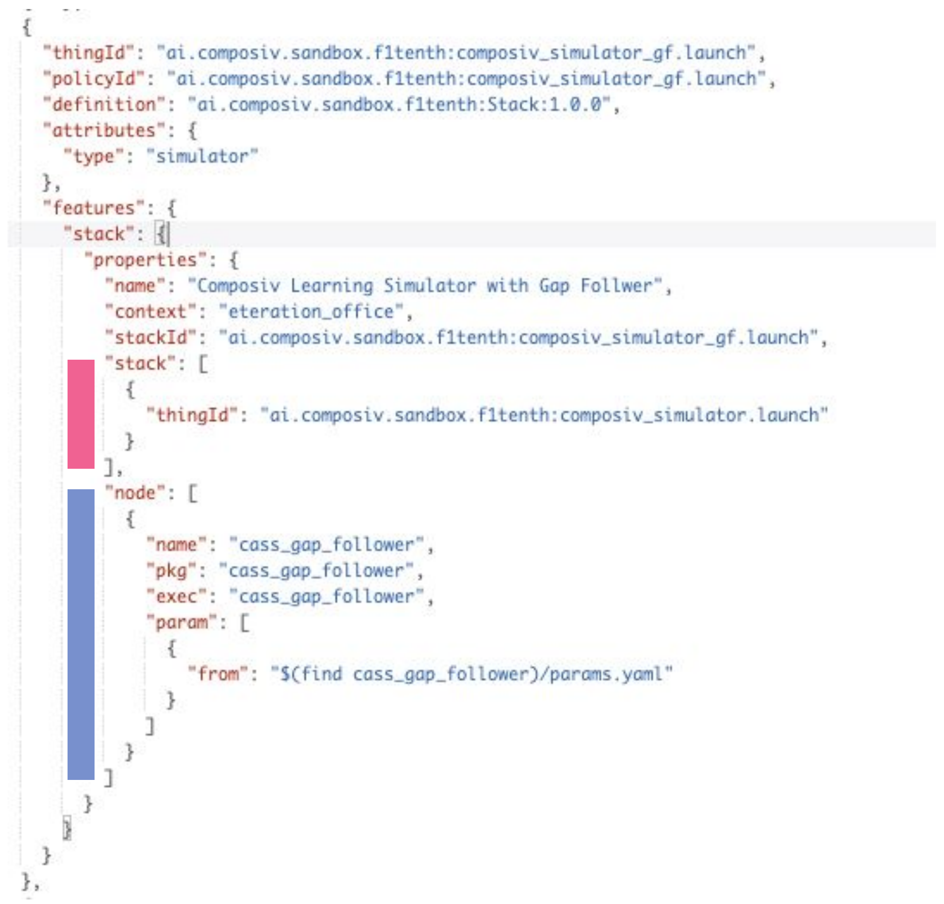


```
{
  "thingId": "ai.composiv.sandbox.f1tenth:donkeycar-nano-01",
  "policyId": "ai.composiv.sandbox.f1tenth:donkeycar-nano-01",
  "definition": "ai.composiv.sandbox.f1tenth.simulator:TestCar:1.0.0",
  "attributes": {
    "serial": "donkeycar-nano-01",
    "type": "simulator",
    "manufacturer": "Eteration"
  },
  "features": {
    "context": {
      "properties": {…}
    },
    "stack": {
      "properties": {
        "current": {
          "stackId": "ai.composiv.sandbox.f1tenth:donkey_base.launch",
          "state": "killed"
        }
      }
    },
    "telemetry": {
      "properties": {…}
    },
    "sensors": {
      "properties": {…}
    },
    "rosModel": {…}
  }
},
```

Stack

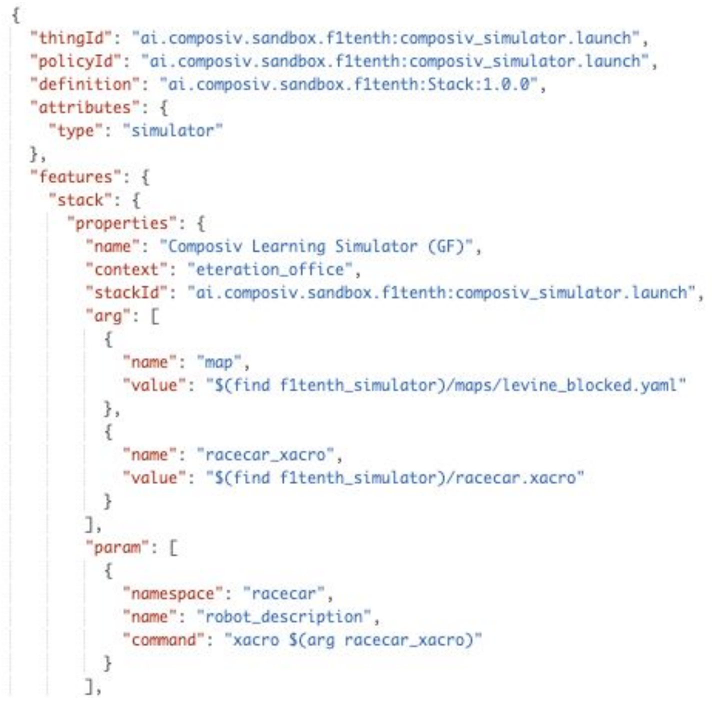
Stack Model

- Follows ROS constructs
    - args/params/nodes/topics/…
- Modular
    - reference other stack “things”

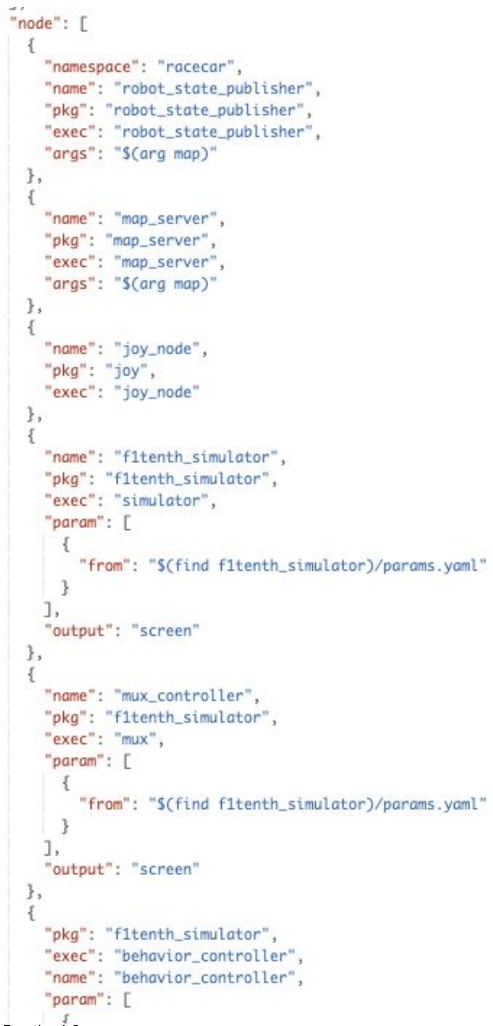


```
{
  "thingId": "ai.composiv.sandbox.f1tenth:composiv_simulator_gf.launch",
  "policyId": "ai.composiv.sandbox.f1tenth:composiv_simulator_gf.launch",
  "definition": "ai.composiv.sandbox.f1tenth:Stack:1.0.0",
  "attributes": {
    "type": "simulator"
  },
  "features": {
    "stack": {
      "properties": {
        "name": "Composiv Learning Simulator with Gap Follwer",
        "context": "eteration_office",
        "stackId": "ai.composiv.sandbox.f1tenth:composiv_simulator_gf.launch",
        "stack": [
          {
            "thingId": "ai.composiv.sandbox.f1tenth:composiv_simulator.launch"
          }
        ],
        "node": [
          {
            "name": "cass_gap_follower",
            "pkg": "cass_gap_follower",
            "exec": "cass_gap_follower",
            "param": [
              {
                "from": "$(find cass_gap_follower)/params.yaml"
              }
            ]
          }
        ]
      }
    }
  }
},
```

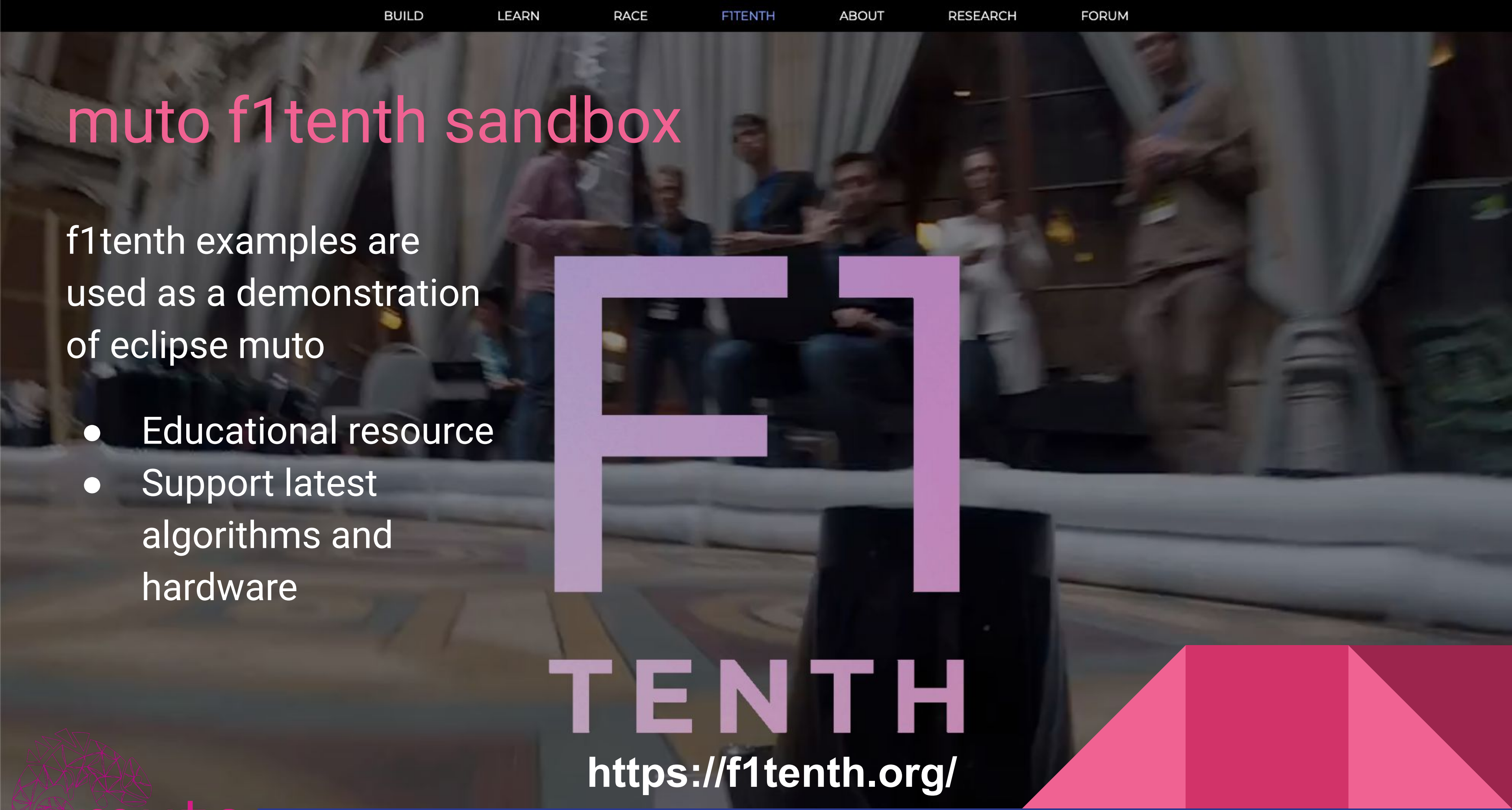
More Complex Stack



```
{
  "thingId": "ai.composiv.sandbox.f1tenth:composiv_simulator.launch",
  "policyId": "ai.composiv.sandbox.f1tenth:composiv_simulator.launch",
  "definition": "ai.composiv.sandbox.f1tenth:Stack:1.0.0",
  "attributes": {
    "type": "simulator"
  },
  "features": {
    "stack": {
      "properties": {
        "name": "Composiv Learning Simulator (GF)",
        "context": "eteration_office",
        "stackId": "ai.composiv.sandbox.f1tenth:composiv_simulator.launch",
        "arg": [
          {
            "name": "map",
            "value": "$(find f1tenth_simulator)/maps/levine_blocked.yaml"
          },
          {
            "name": "racecar_xacro",
            "value": "$(find f1tenth_simulator)/racecar.xacro"
          }
        ],
        "param": [
          {
            "namespace": "racecar",
            "name": "robot_description",
            "command": "xacro $(arg racecar_xacro)"
          }
        ],
```



```
"node": [
  {
    "namespace": "racecar",
    "name": "robot_state_publisher",
    "pkg": "robot_state_publisher",
    "exec": "robot_state_publisher",
    "args": "$(arg map)"
  },
  {
    "name": "map_server",
    "pkg": "map_server",
    "exec": "map_server",
    "args": "$(arg map)"
  },
  {
    "name": "joy_node",
    "pkg": "joy",
    "exec": "joy_node"
  },
  {
    "name": "f1tenth_simulator",
    "pkg": "f1tenth_simulator",
    "exec": "simulator",
    "param": [
      {
        "from": "$(find f1tenth_simulator)/params.yaml"
      }
    ],
    "output": "screen"
  },
  {
    "name": "mux_controller",
    "pkg": "f1tenth_simulator",
    "exec": "mux",
    "param": [
      {
        "from": "$(find f1tenth_simulator)/params.yaml"
      }
    ],
    "output": "screen"
  },
  {
    "pkg": "f1tenth_simulator",
    "exec": "behavior_controller",
    "name": "behavior_controller",
    "param": [
```



muto f1tenth sandbox

f1tenth examples are used as a demonstration of eclipse muto

- Educational resource
- Support latest algorithms and hardware

https://f1tenth.org/

muto

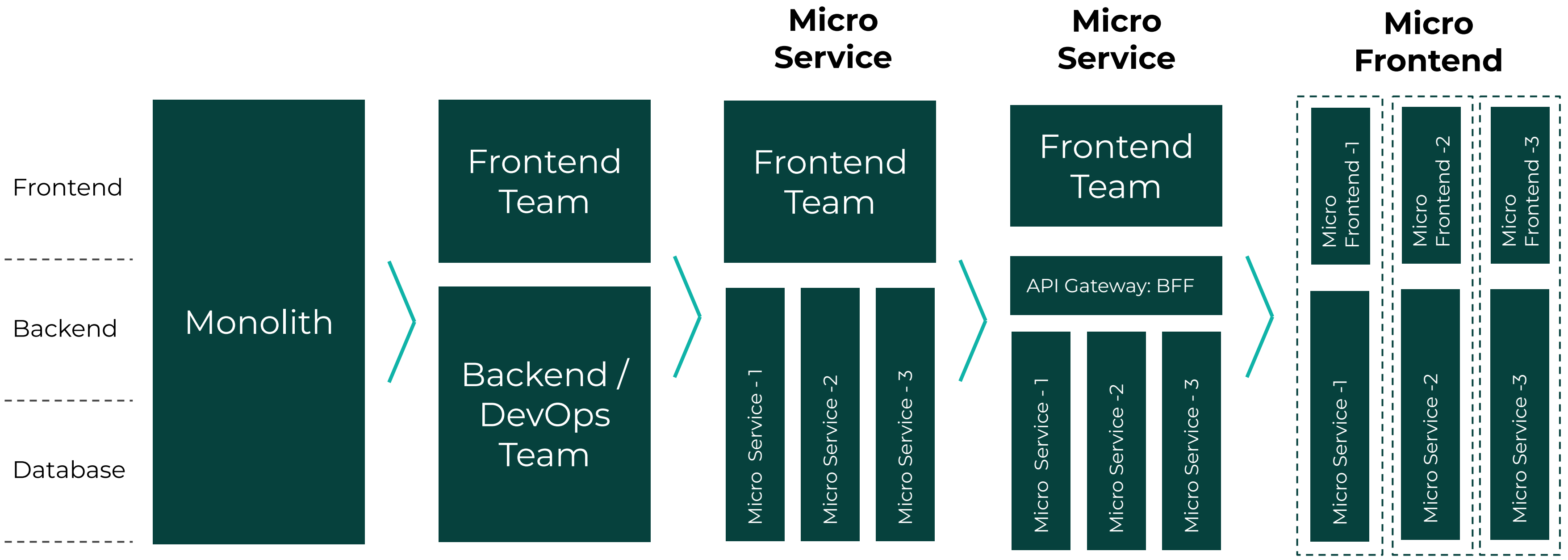
muto liveUI

LiveUI helps you divide a monolithic frontend into smaller, more manageable micro frontends.

There is no magic, LiveUI allows you to split and manage your codebase, teams, release processes and runtimes independently

Composable at Runtime! LiveUI has the ability to change UI while its running

Micro Frontend ?



Independently Deployed

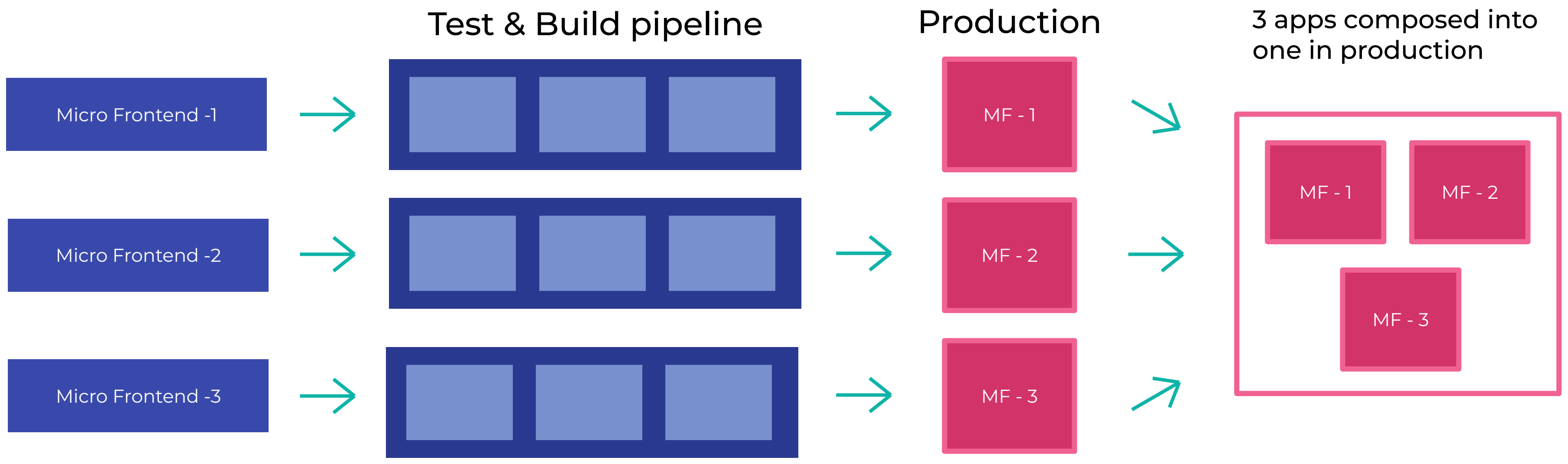


diagram reproduced from https://martinfowler.com/articles/micro-frontends.html

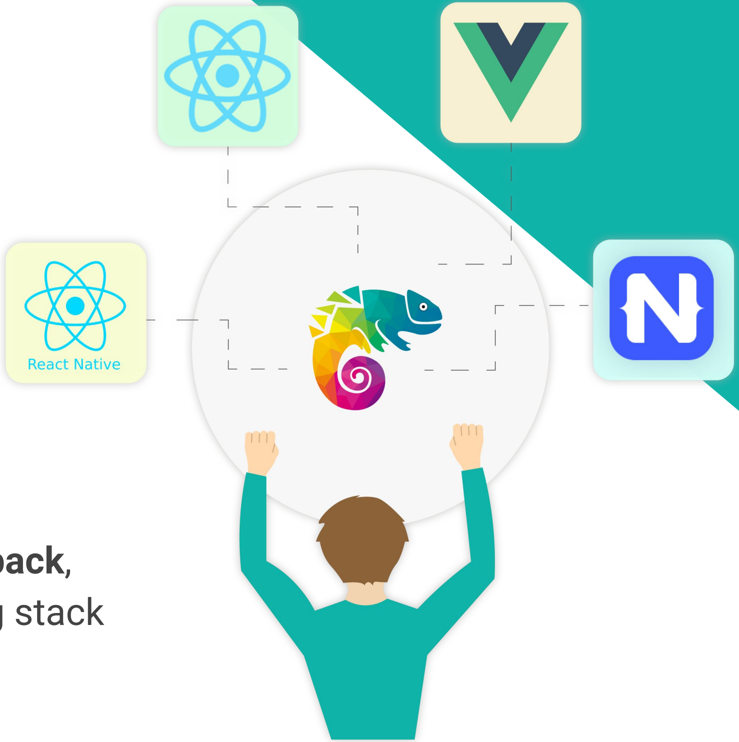
Framework Support

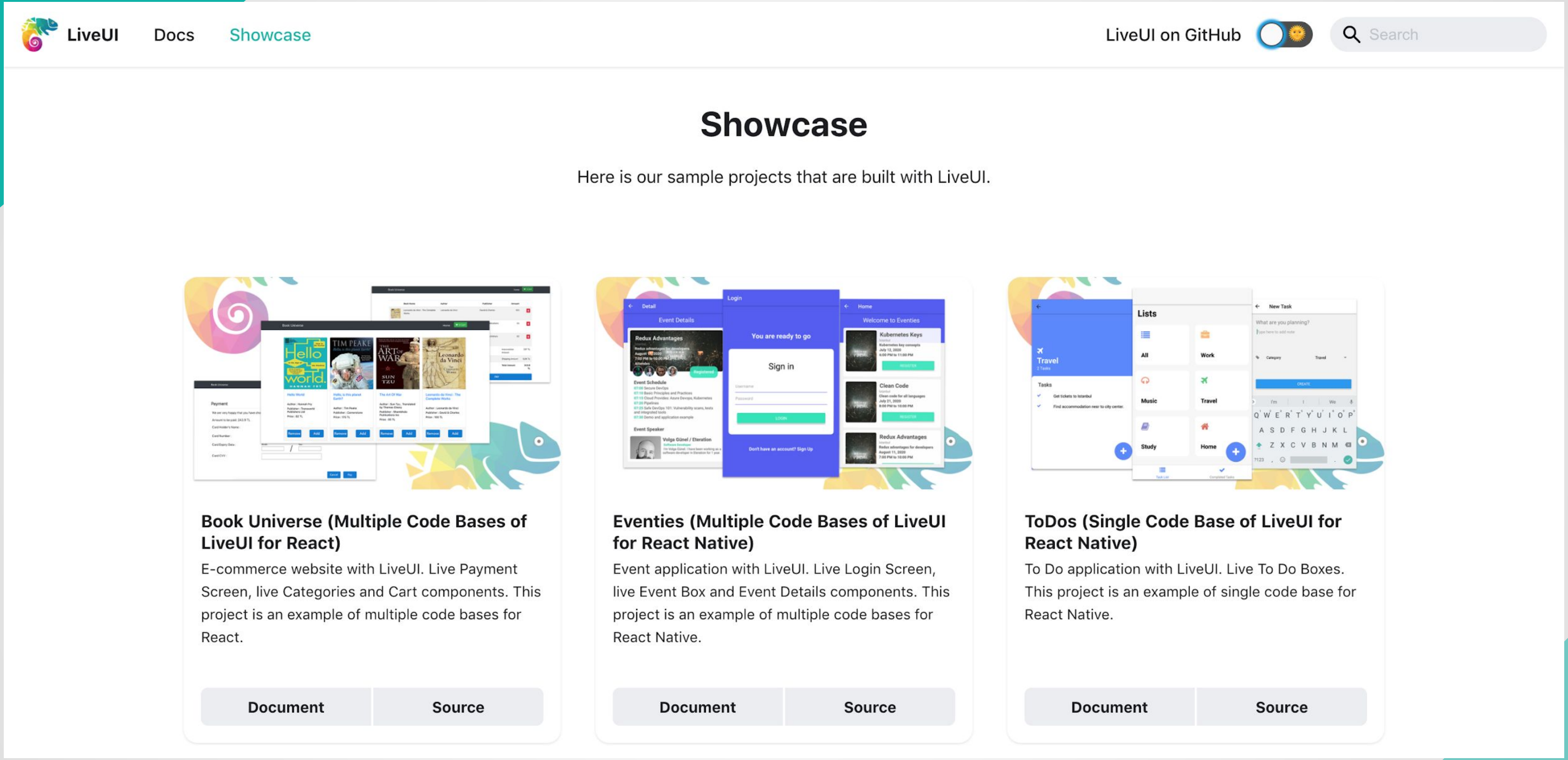
LiveUI is framework and bundler agnostic.

LiveUI works with

- **React.js**
- **React Native**
- **Vue.js**
- **NativeScript**

You can use with the bundler of your choice such as **Webpack**, **Metro** and others. It will reuse support from the underlying stack such as the upcoming Module federation from webpack 5 automatically, or default to its own.





Showcase

Here is our sample projects that are built with LiveUI.

Book Universe (Multiple Code Bases of LiveUI for React)

E-commerce website with LiveUI. Live Payment Screen, live Categories and Cart components. This project is an example of multiple code bases for React.

Document | Source

Eventies (Multiple Code Bases of LiveUI for React Native)

Event application with LiveUI. Live Login Screen, live Event Box and Event Details components. This project is an example of multiple code bases for React Native.

Document | Source

ToDos (Single Code Base of LiveUI for React Native)

To Do application with LiveUI. Live To Do Boxes. This project is an example of single code base for React Native.

Document | Source

Thank you !

- *https://projects.eclipse.org/proposals/eclipse-muto*
  - *https://liveui.composiv.ai*
  - *Soon: github.com/eclipse-muto*