Funded by
the European Union

CONVINCE

# sit-aw-anchoring – Hands-on

ROSCon Fr/De W5

18/11/2025

ISTITUTO ITALIANO DI TECNOLOGIA · Università di Genova · cea · CITTA' DI TORINO · TIMELEX · UNIVERSITY OF BIRMINGHAM · BOSCH · INVENTYA Your success. Our commitment · FONDAZIONE TORINO MUSEI · CLEIA Engineering Innovation Automation

*Horizon Europe Programme under Grant Agreement No 101070227. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.*

# Agenda

- Software overview *(architecture, how to adapt for your application domain, how to launch, how to use) [~10+ mins]*

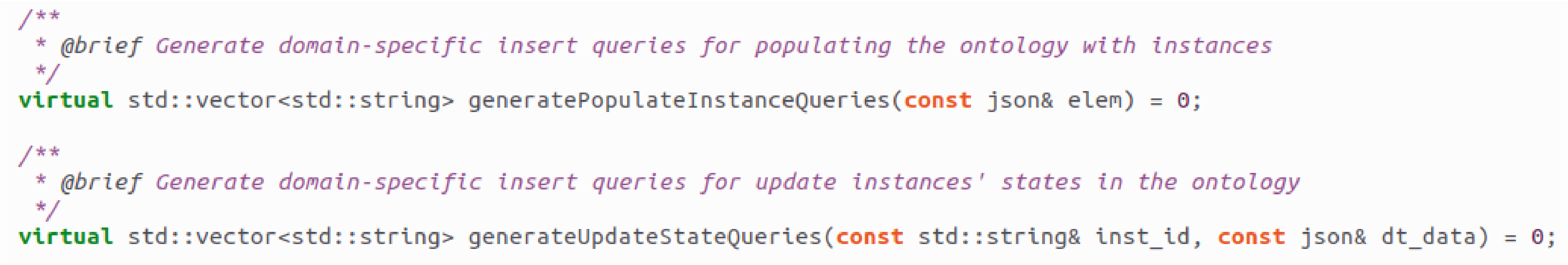
- Run *ready* example *(just run, no modifs) [~10- mins]*

- Add a simple customization *(as time permits) [~10 mins]*

Funded by
the European Union

# Architecture

- pkgs organization
  - **`anchoring_process`** — implements the anchoring process; the node to deploy.
  - **`anchoring_process_interfaces, process_msgs`** — interface defs for action servers.
  - **`anchoring_core`** — base plugin for entity managers; the one to specialize for your ontology.
- wait, plugins?
  - custom data process for your specific domain concepts (which data into the KG, how to map data from DT, …)



```
/**
 * @brief Generate domain-specific insert queries for populating the ontology with instances
 */
virtual std::vector<std::string> generatePopulateInstanceQueries(const json& elem) = 0;

/**
 * @brief Generate domain-specific insert queries for update instances' states in the ontology
 */
virtual std::vector<std::string> generateUpdateStateQueries(const std::string& inst_id, const json& dt_data) = 0;
```

Funded by
the European Union

# Plugin Example `anchoring_cubesworld_plugin`

- Plugins are ***THE "thing"*** to implement to use semantic anchoring for your application domain
- Unfamiliar with plugins? https://docs.ros.org/en/rolling/Tutorials/Beginner-Client-Libraries/Pluginlib.html
- **`src/examples/anchoring_cubesworld_plugin/src/anchoring_cubesworld_plugin.cpp`**
  - specializes `anchoring_core::AnchoringManager`
  - weak ptr to parent managed node (for custom operations in configure, activate, cleanup, …)
  - knows about the mapping rules btw DT data and KG concepts data, e.g., `pkg_share_dir+"/rules/mapping"`
  - knows how to manage specificities of concepts' creation in the KG, e.g., cube also need a grasp pose *L108—L123*
- **`src/examples/anchoring_cubesworld_plugin/rules/schemas/`**
  - the schemas that formalize the TQL conceptualization of a domain, e.g., the one we will refer to *"CubesWorld"* (name is free) in the following.

Funded by
the European Union

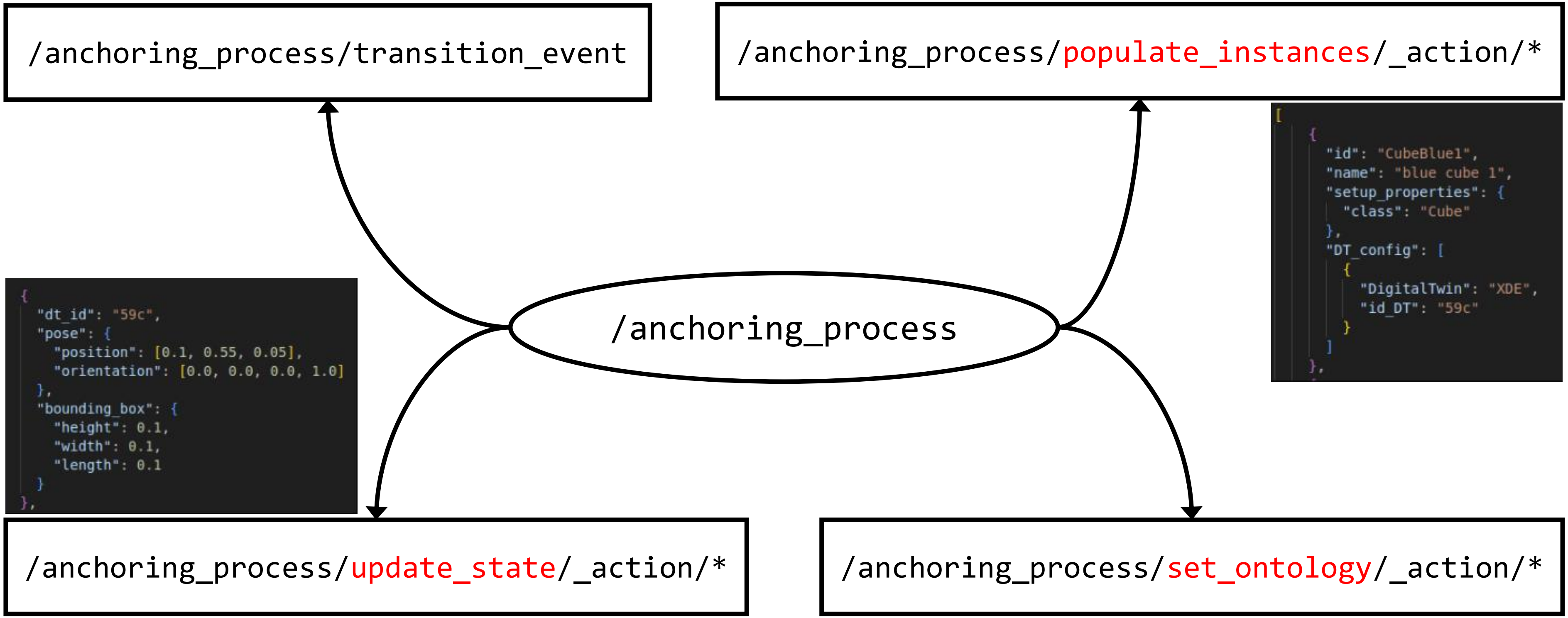
# Deployment Example: `pick_place_uc`

- **`src/examples/pick_place_uc/launch/cfg/params.yaml`**
  - register *entity managers* — *keys must be the entity names* in your domain *(L4—L6)*
    and, for each manager, the corresponding plugin *(L7—L10)*
  - register knowledge domains — keys are arbitrary *(L11—L12)*
    and, for each domain, a DB name, pkg and paths where schemas are defined *(L13—L19)*
  - set the DB serveraddr *(L3)*

Funded by
the European Union

# ROS Graph

CONVINCE



/anchoring_process/transition_event

/anchoring_process/populate_instances/_action/*

/anchoring_process

/anchoring_process/update_state/_action/*

/anchoring_process/set_ontology/_action/*

```
{
  "dt_id": "59c",
  "pose": {
    "position": [0.1, 0.55, 0.05],
    "orientation": [0.0, 0.0, 0.0, 1.0]
  },
  "bounding_box": {
    "height": 0.1,
    "width": 0.1,
    "length": 0.1
  }
},
```

```
[
    {
      "id": "CubeBlue1",
      "name": "blue cube 1",
      "setup_properties": {
        "class": "Cube"
      },
      "DT_config": [
        {
          "DigitalTwin": "XDE",
          "id_DT": "59c"
        }
      ]
    },
```

Funded by
the European Union

# Run the Example (1/5)

- Terminal 1 (start TypeDB server)
  - *Host*
    - *`make start-docker`*
  - *Container*
    - *`typedb server`*

- Terminal 2 (start TypeDB Studio, configure later)
  - Host
    - *`make join-docker`*
  - Container
    - *`typedb-studio`*

Funded by
the European Union

# Run the Example (2/5)

CONVINCE

- Terminal 3 (start DT environment, `setup.json` is already exported)
  - *Host*
    - *make join-docker*
  - *Container*
    - *cd /tmp/dt*
    - *python3 DT_Simulation.py*

- Terminal 4 (prepare to run the simulation)
  - Host
    - *make join-docker*
  - Container
    - *cd /tmp/dt*

Funded by
the European Union

# Run the Example (3/5)

- Terminal 5(*) (launch `pick_place_uc`)
  - *Container*
    - `ros2 launch pick_place_uc pick_place_uc_launch.py`

- Terminal 6 (activate `anchoring_process`)
  - Host
    - `make join-docker`
  - Container
    - `ros2 lifecycle set /anchoring_process configure`
    - `ros2 lifecycle set /anchoring_process activate`

Funded by
the European Union

# Run the Example (4/5) — Setup Phase

- Terminal 6 (cont'd, setup phase for the `anchoring_process`)
  - Container

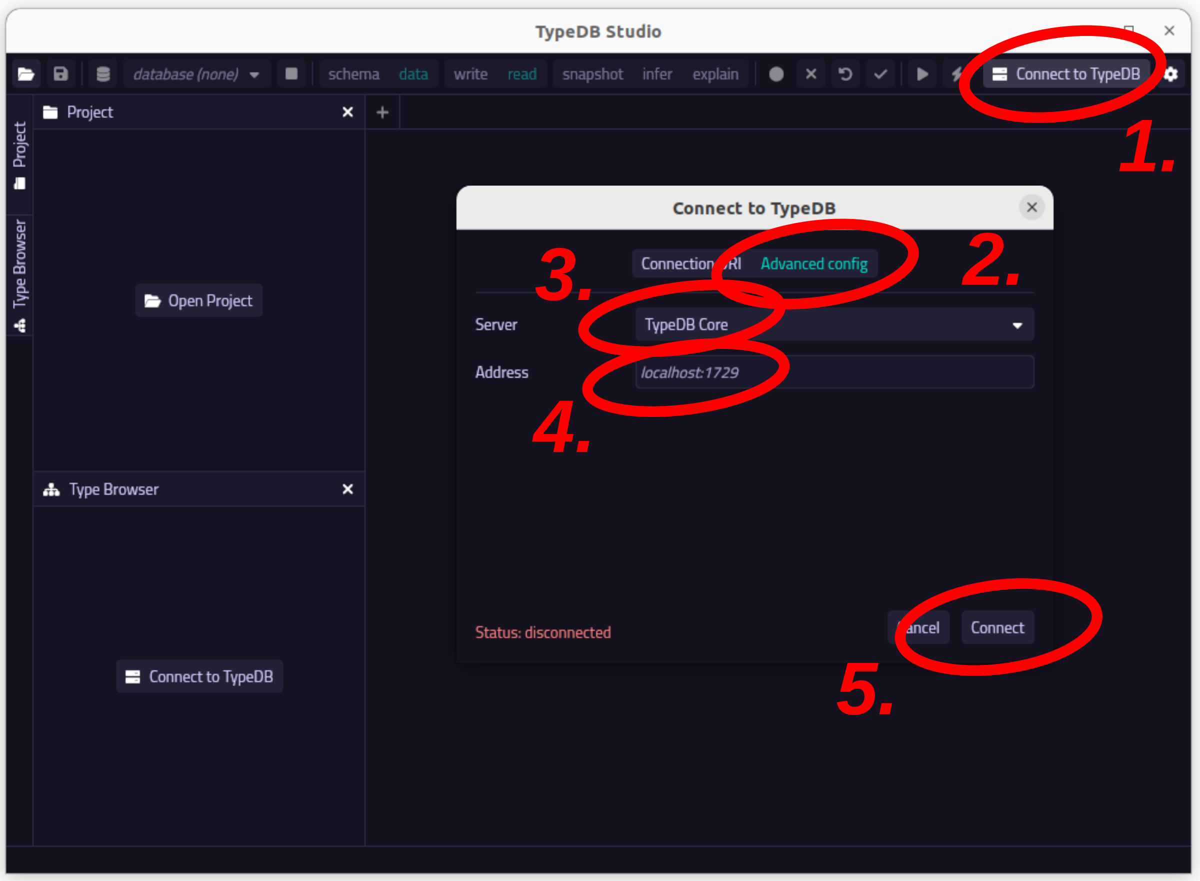
    -
      ```
      ros2 action send_goal /anchoring_process/set_ontology anchoring_process_interfaces/action/SetOntology "{knowledge_domain: 'CubesWorld'}"
      ```

    -
      ```
      ros2 action send_goal /anchoring_process/populate_instances anchoring_process_interfaces/action/PopulateInstances "{knowledge_domain: 'CubesWorld', instances: '/tmp/dt/setup.json'}"
      ```

Funded by
the European Union

# Configure typedb-studio

- Connect to TypeDB



Server: TypeDB Core

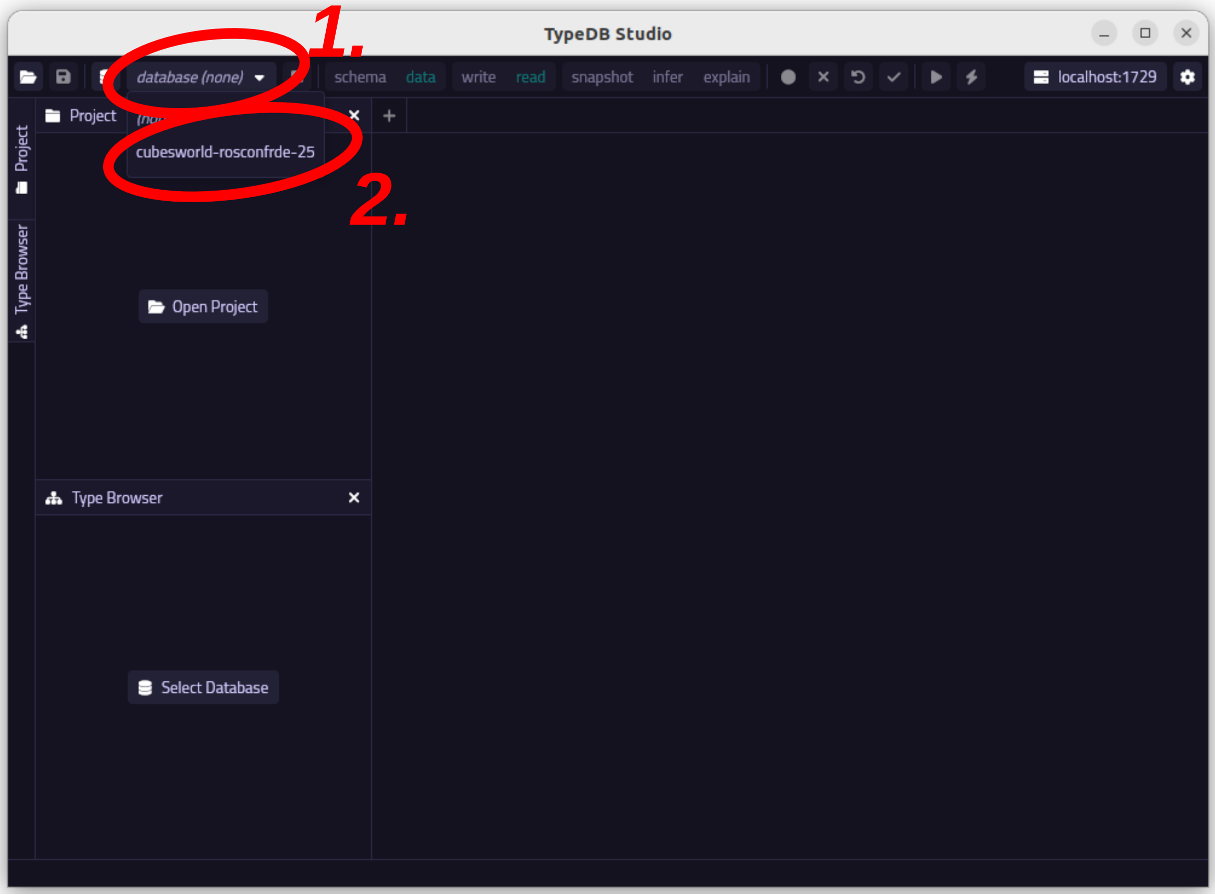
Address: localhost:1729

Status: disconnected

Funded by
the European Union

# Configure typedb-studio

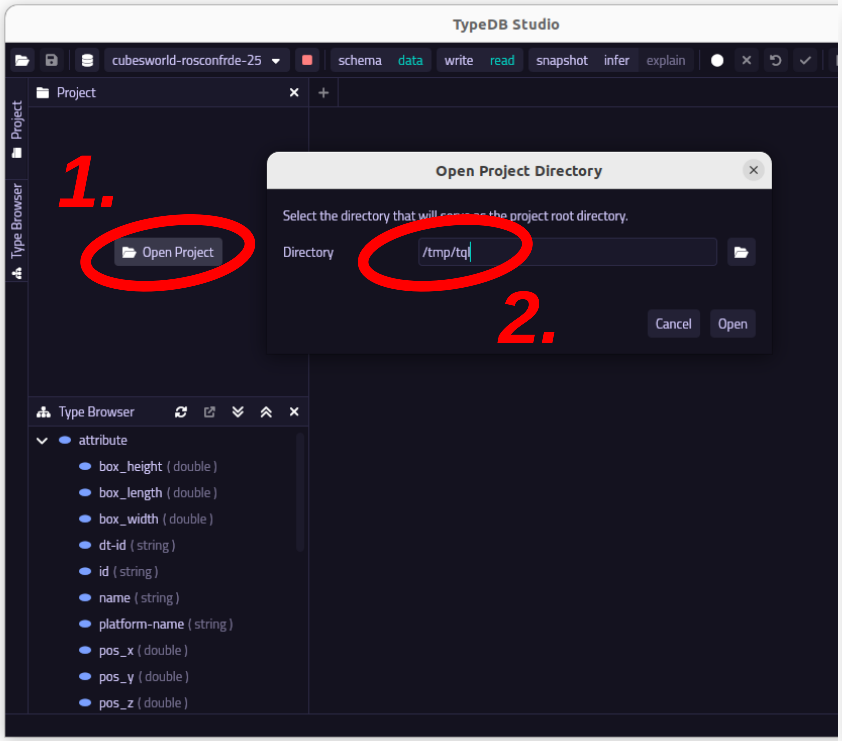
- Select database



Funded by the European Union

# Configure typedb-studio

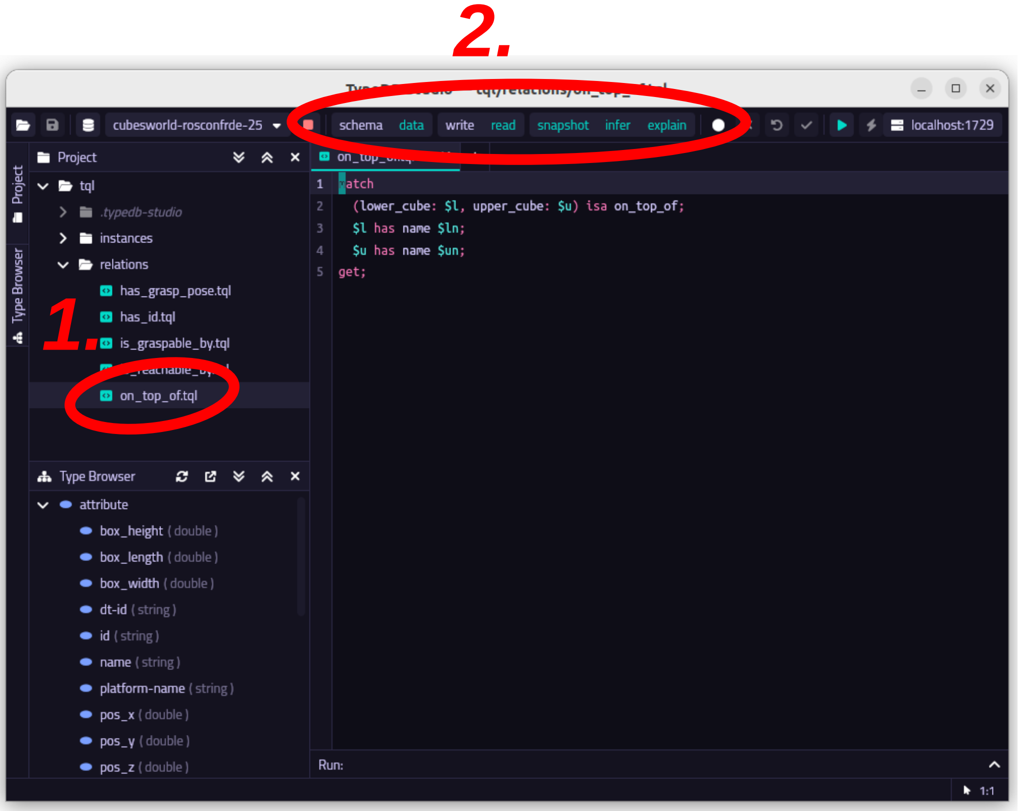
- Open queries



Open Project Directory

Select the directory that will serve as the project root directory.

Directory /tmp/tql



```
match
  (lower_cube: $l, upper_cube: $u) isa on_top_of;
  $l has name $ln;
  $u has name $un;
get;
```

Funded by
the European Union

# Run the Example (5/5) — Execution Phase

- Terminal 4 (cont’d, export DT data at regular intervals)
  - Container

    -
      ```
      python3 update_json.py
      ```

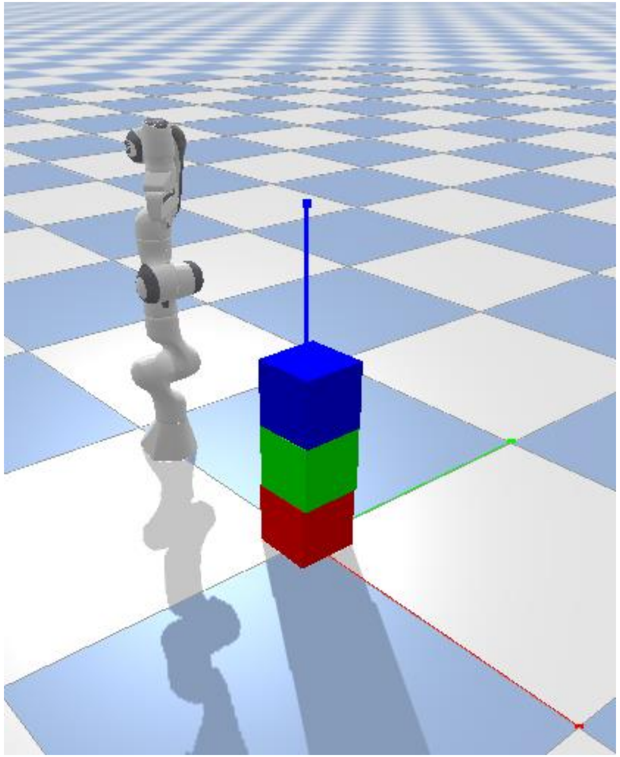
- Terminal 6 (cont’d, execution phase for the `anchoring_process`)
  - Container

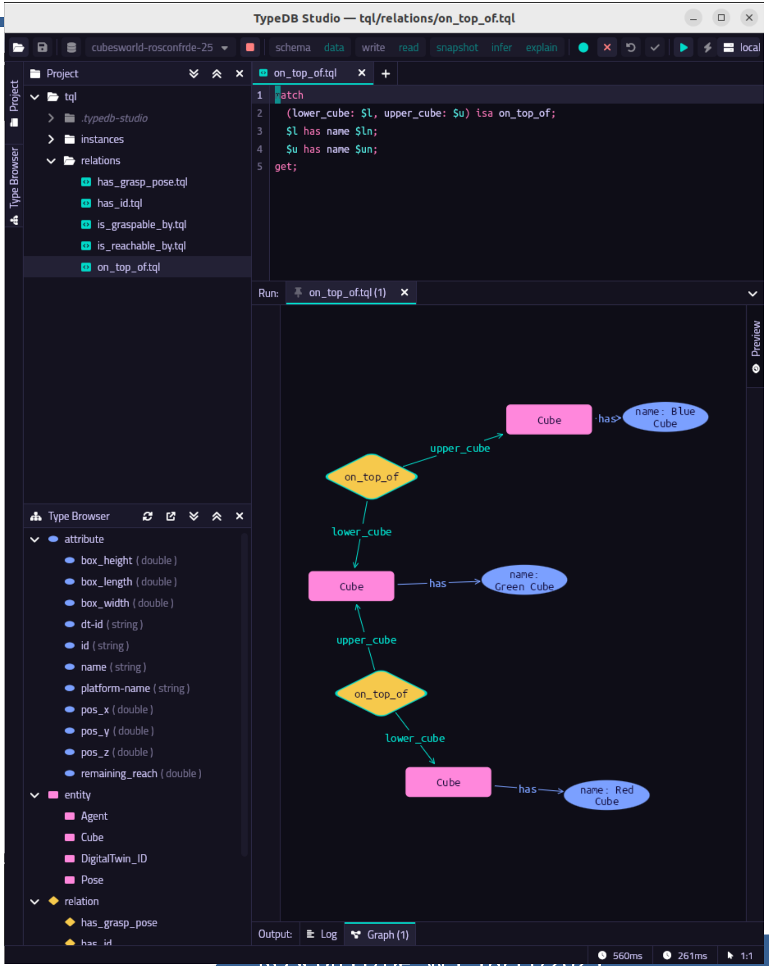
    -
      ```
      ros2 action send_goal /anchoring_process/update_state
      anchoring_process_interfaces/action/UpdateState "{knowledge_domain:
      'CubesWorld', instances: '/tmp/dt/runtime.json'}"
      ```

Funded by
the European Union

# Execute `on_top_of` Query in TypeDB Studio

CONVINCE





TypeDB Studio — tql/relations/on_top_of.tql

```
match
  (lower_cube: $l, upper_cube: $u) isa on_top_of;
  $l has name $ln;
  $u has name $un;
get;
```

Funded by
the European Union

# Play with the Example

- Cubes and robot have interactive markers and can be moved (not the robot base)

- In Terminal 6, execute the update state action for situations of your choice)
  - Container

    -
      ```
      ros2 action send_goal /anchoring_process/update_state anchoring_process_interfaces/action/UpdateState "{knowledge_domain: 'CubesWorld', instances: '/tmp/dt/runtime.json'}"
      ```

Funded by
the European Union

# Questions?