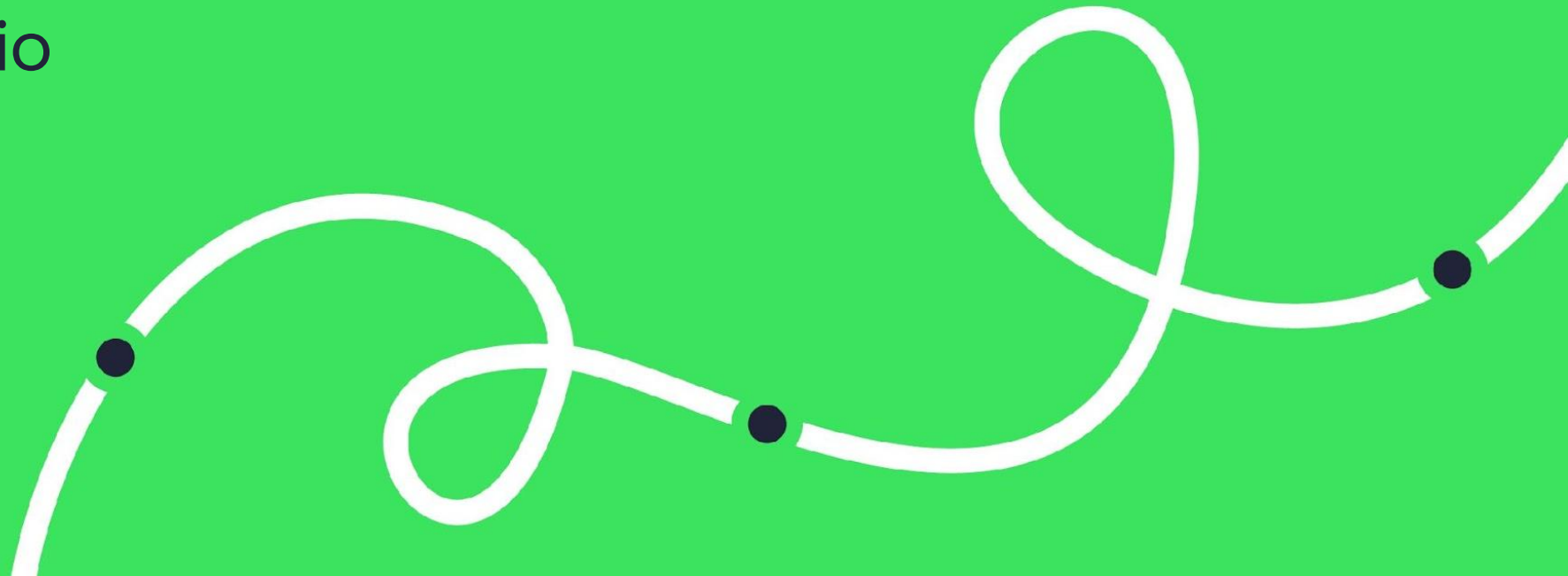


# Semplificare l'automazione multivendor client RESTCONF da moduli YANG

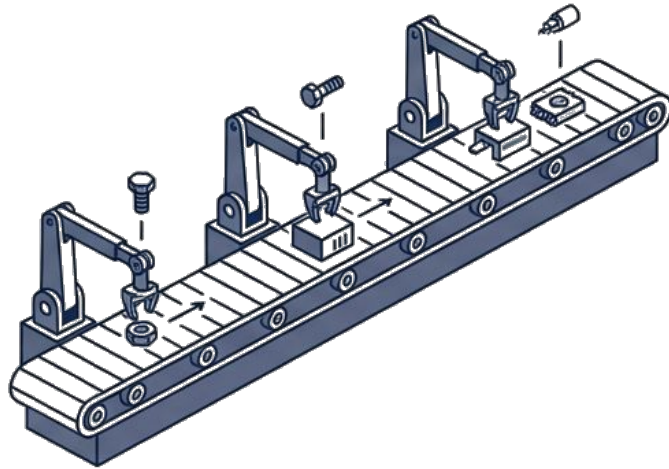
Pydantic-based IDE-friendly RESTCONF clients from YANG

Matteo Colantonio



# Network Orchestration

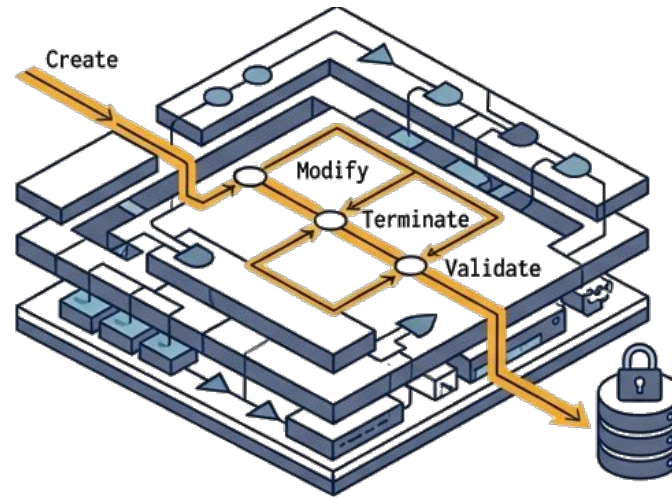
## Automation



Executing repetitive tasks reliably.

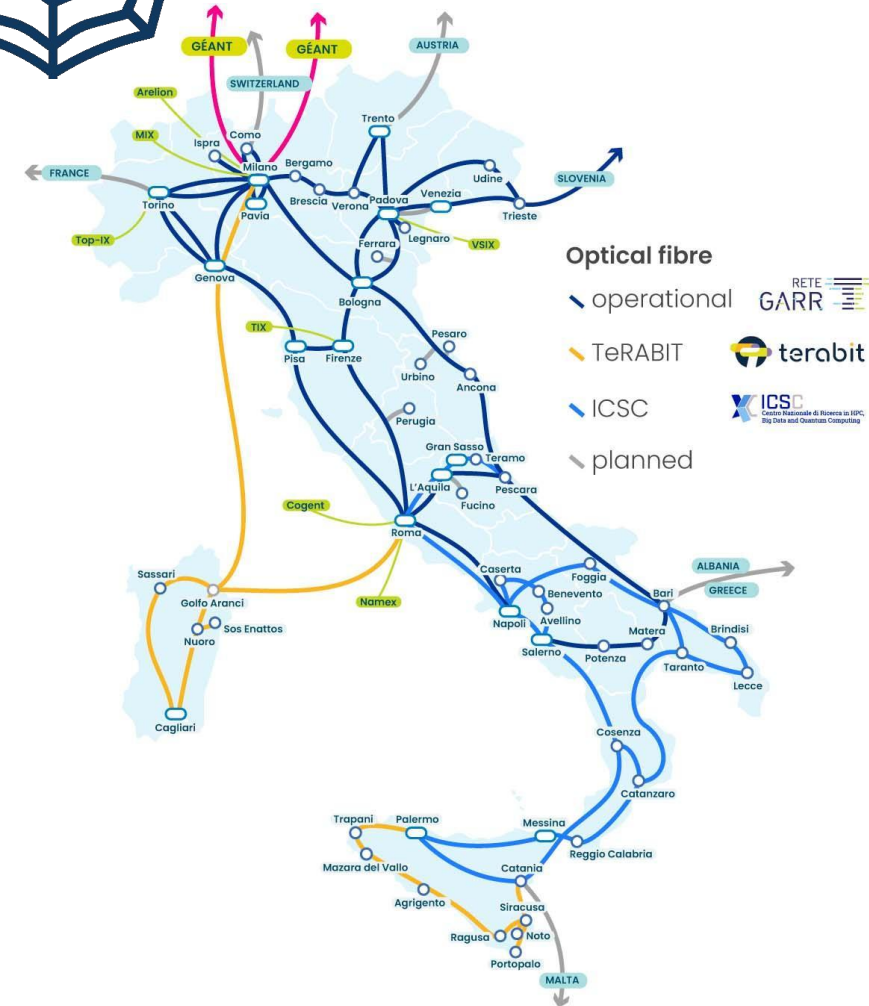
Focuses on single actions (e.g. config templating) with no or low abstraction

## Orchestration

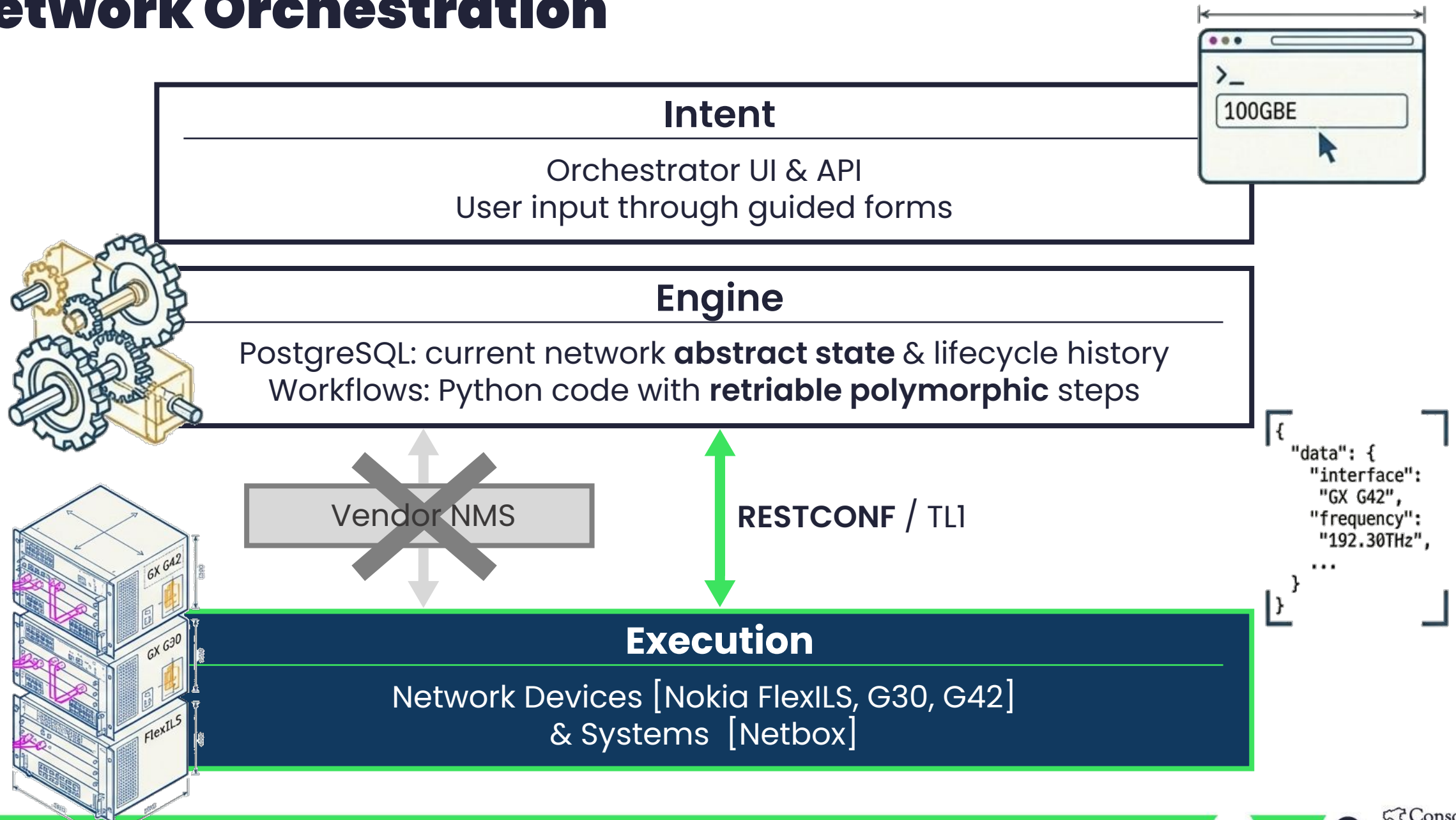


Adding **abstract** and **stateful** layer decoupled from HW.

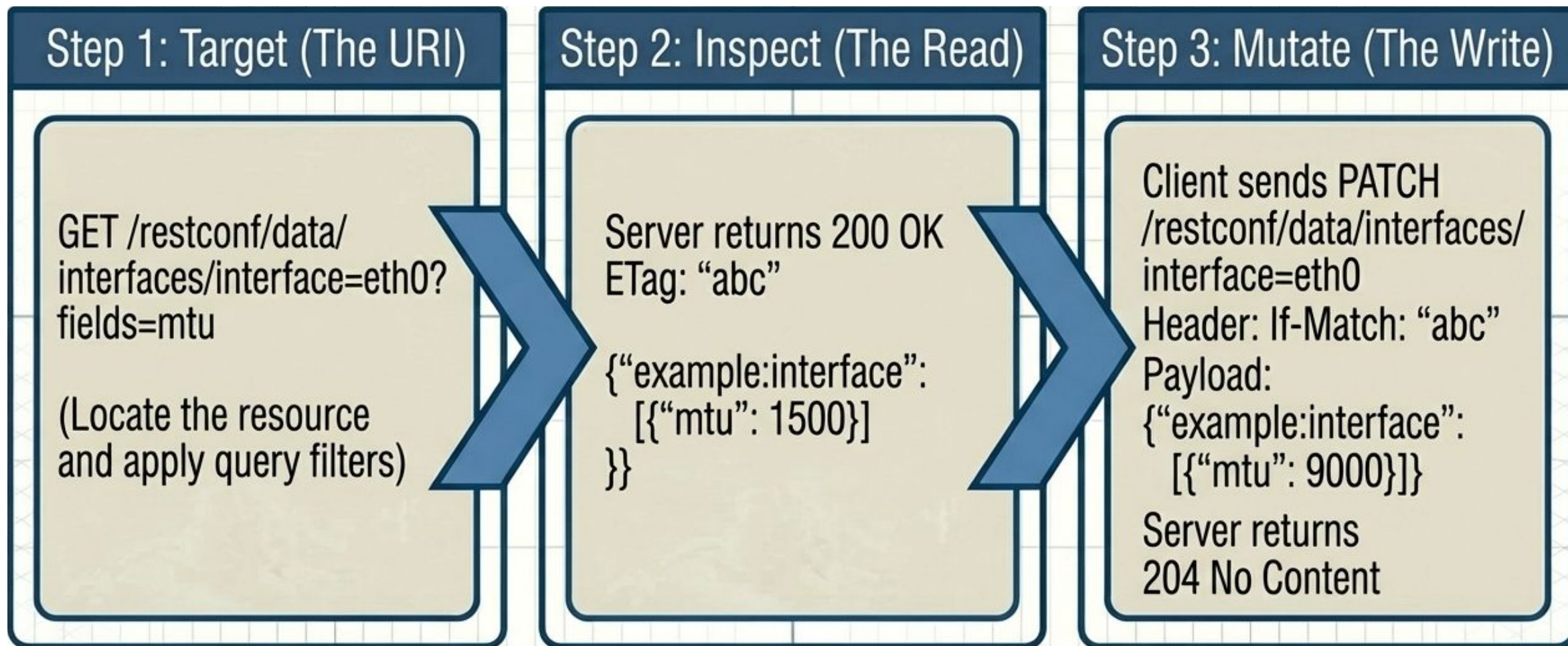
Managing device and service **lifecycles** with full state awareness and **audit trail**.



# Network Orchestration

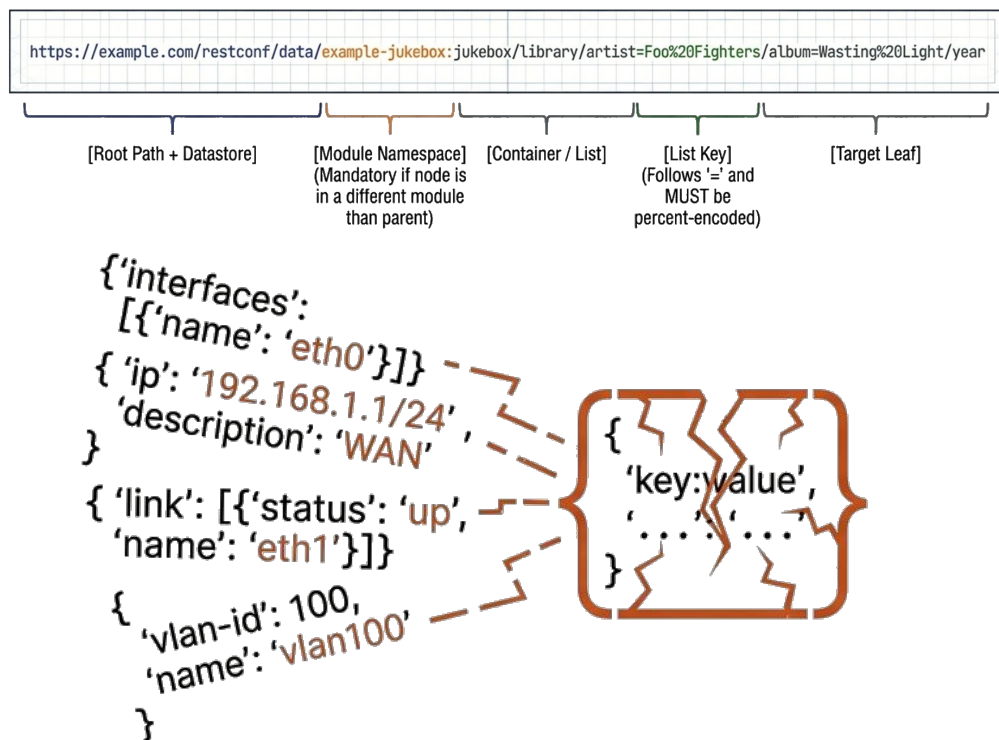


# End-to-end RESTCONF interaction



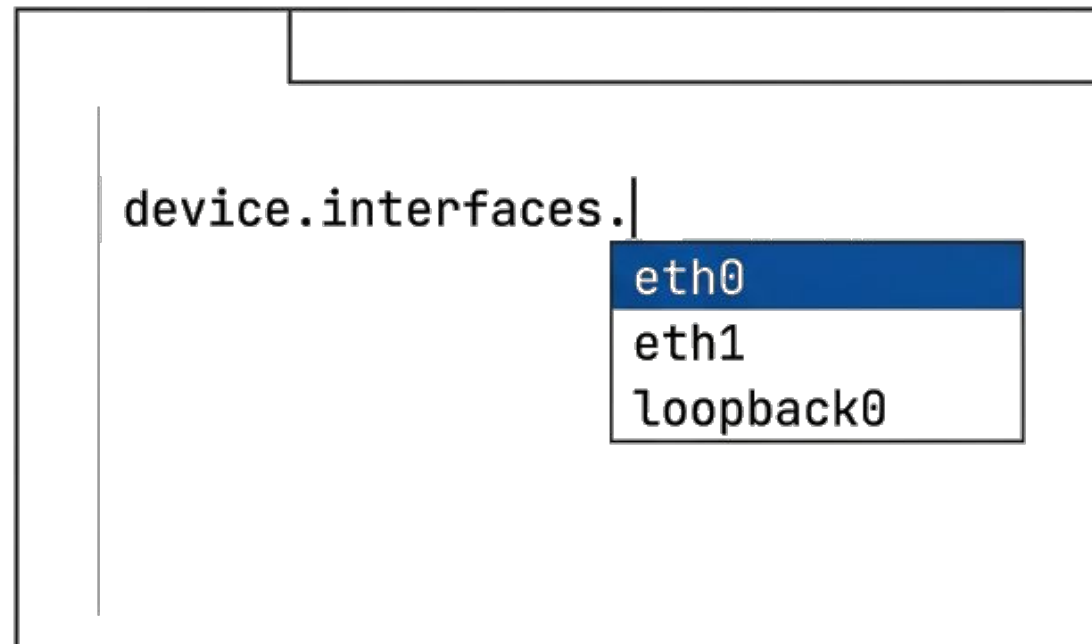
# Using raw RESTCONF is fragile

## Raw



Managing unvalidated JSON and URIs leads to runtime **errors**, **tedious** boilerplate, and **brittle** automation scripts.

## Ideal



I want an IDE-friendly interface with **autocomplete** and **validation** that tracks **YANG** modules and their release changes

# So I built [u.garr.it/pyangdantic](https://u.garr.it/pyangdantic)

```
temp > yang tree > g30.txt
730
731 module: ne
732 +--rw ne
733   +--rw ne-id?          string
734   +--rw ne-name?       string
735   +--ro ne-type?       string
736   +--rw ne-location?   string
737   +--rw ne-site?       string
738   +--rw ne-altitude?   int16
739   +--ro ne-vendor?     string
740   +--ro ne-temperature? types:temperature
741   +--rw shelf* [shelf-id]
742     | +--rw shelf-id      types:shelf-id
743     | +--rw shelf-type?   string
744     | +--ro shelf-serial-number? string
745     | +--rw shelf-location? string
746     | +--rw shelf-mac-address? string
747     | +--rw shelf-linklocal-ip-address? inet:ip-address
748     | +--rw shelf-ip-address? inet:ip-address
749     | +--rw flush_l2_addr_table? uint8
750     | +--rw unknown-pluggable-report? types:enable-switch
751     | +--ro inlet-temperature? types:temperature
752     | +--ro outlet-temperature? types:temperature
753     | +--rw admin-status? enumeration
754     | +--ro oper-status? enumeration
755     | +--ro avail-status? enumeration
```

```
wfo-pilot [WSL: Ubuntu-24...
test_g30.py 9+, M
docker > core > garr > tests > lab > test_g30.py > itnog10_restconf_data
163
164
165
166
167
168 def itnog10_restconf_data(client_g30_51: G30Client):
169     client_g30_51.data.ne.ne.s
170     shelf
171     services
172     system
173     __subclasshook__
174     __setattr__
175     __sizeof__
176     __slots__
177     __str__
178     __init_subclass__
179
180     shelf: ShelfListNode (property)
181
182
183
184
185
186
```

HTTPs session manager

running datastore

# Autocomplete & type safety

```
730
731 module: ne
732 +---rw ne
733   +---rw ne-id?          string
734   +---rw ne-name?       string
735   +---ro ne-type?       string
736   +---rw ne-location?   string
737   +---rw ne-site?       string
738   +---rw ne-altitude?   int16
739   +---ro ne-vendor?     string
740   +---ro ne-temperature? types:temperature
741   +---rw shelf* [shelf-id]
742     | +---rw shelf-id      types:shelf-id
743     | +---rw shelf-type?   string
744     | +---ro shelf-serial-number? string
745     | +---rw shelf-location? string
746     | +---rw shelf-mac-address? string
747     | +---rw shelf-linklocal-ip-address? inet:ip-address
748     | +---rw shelf-ip-address? inet:ip-address
749     | +---rw flush_l2_addr_table? uint8
750     | +---rw unknown-pluggable-report? types:enable-switch
751     | +---ro inlet-temperature? types:temperature
752     | +---ro outlet-temperature? types:temperature
753     | +---rw admin-status? enumeration
754     | +---ro oper-status? enumeration
```

```
163
164
165
166
167
168 def itnog10_restconf_data(client_g30_51: G30Client):
169     uri = client_g30_51.data.ne_ne.shelf(1)
170     shelf = uri.retrieve(content="config", depth=2)
171     shelf.sh
172
173     [x] shelf_id
174     [x] shelf_ip_address
175     [x] shelf_linklocal_ip_address
176     [x] shelf_location
177     [x] shelf_mac_address
178     [x] shelf_serial_number
179     [x] shelf_type
180     [x] schema
181     [x] schema_json
182     [x] __subclasshook__
183     [x] model_json_schema
184     [x] __pydantic_core_schema__
185
186     shelf_id: int
```

dynamic URI navigation

GET operation

Pydantic model = YANG container

# Validation according to YANG rules\*

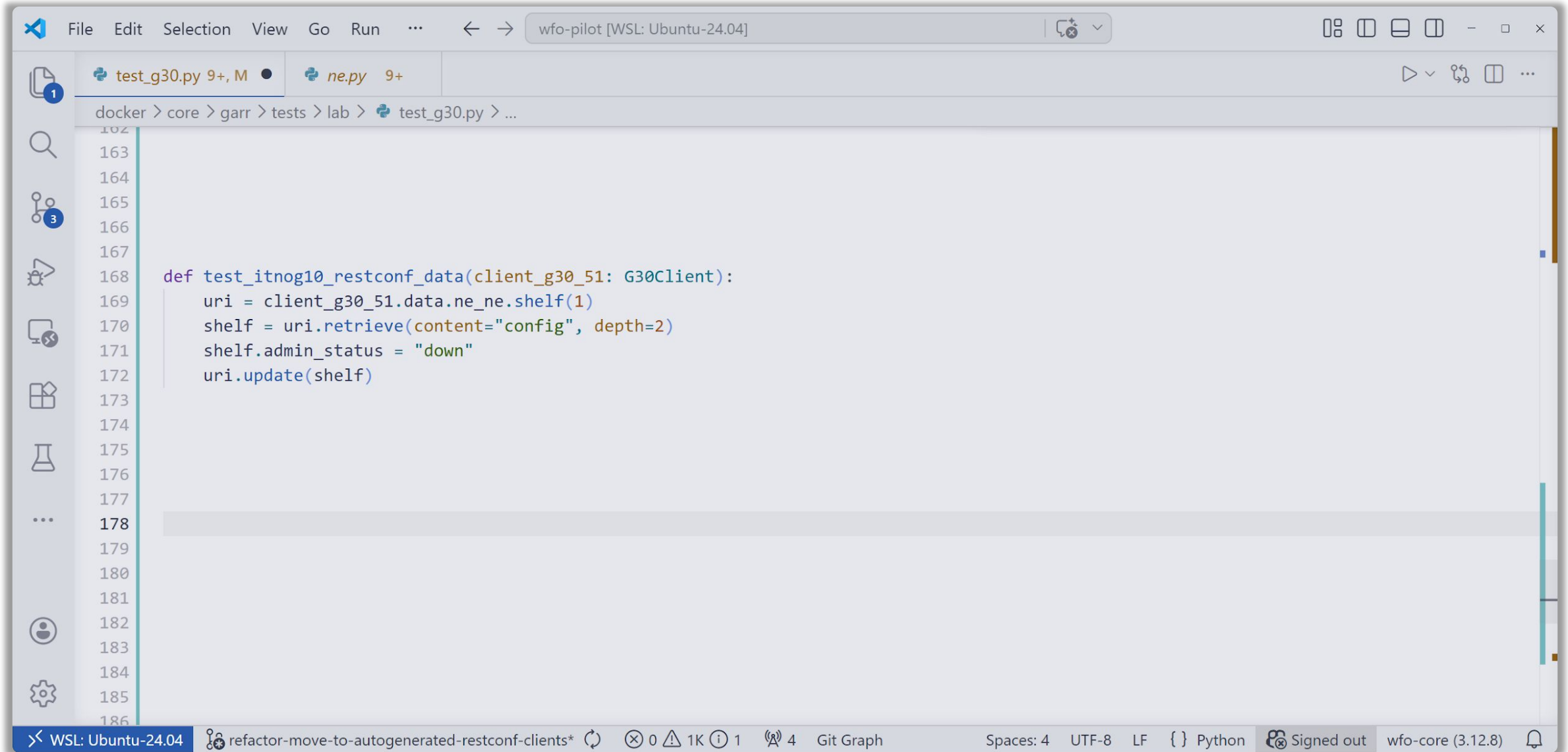
```
167
168 def test_itnog10_restconf_data(client_g30_51: G30Client):
169     uri = client_g30_51.data.ne_ne.shelf(1)
170     shelf = uri.retrieve(content="config", depth=2)
171     shelf.admin_status = "dawn"
172
173
```

```
(wfo-core) colantonio@DESKTOP-89010PL:~/repos/wfo-pilot/docker/core/garr$ pytest -s tests/lab/test_g30.py::test_itnog10_restconf_data
>
E     pydantic_core._pydantic_core.ValidationError: 1 validation error for ShelfItem
E       admin_status
E         Input should be 'up', 'down' or 'up-no-alm' [type=enum, input_value='dawn', input_type=str]
E       For further information visit https://errors.pydantic.dev/2.8/v/enum

.venv/lib/python3.12/site-packages/pydantic/main.py:853: ValidationError
```

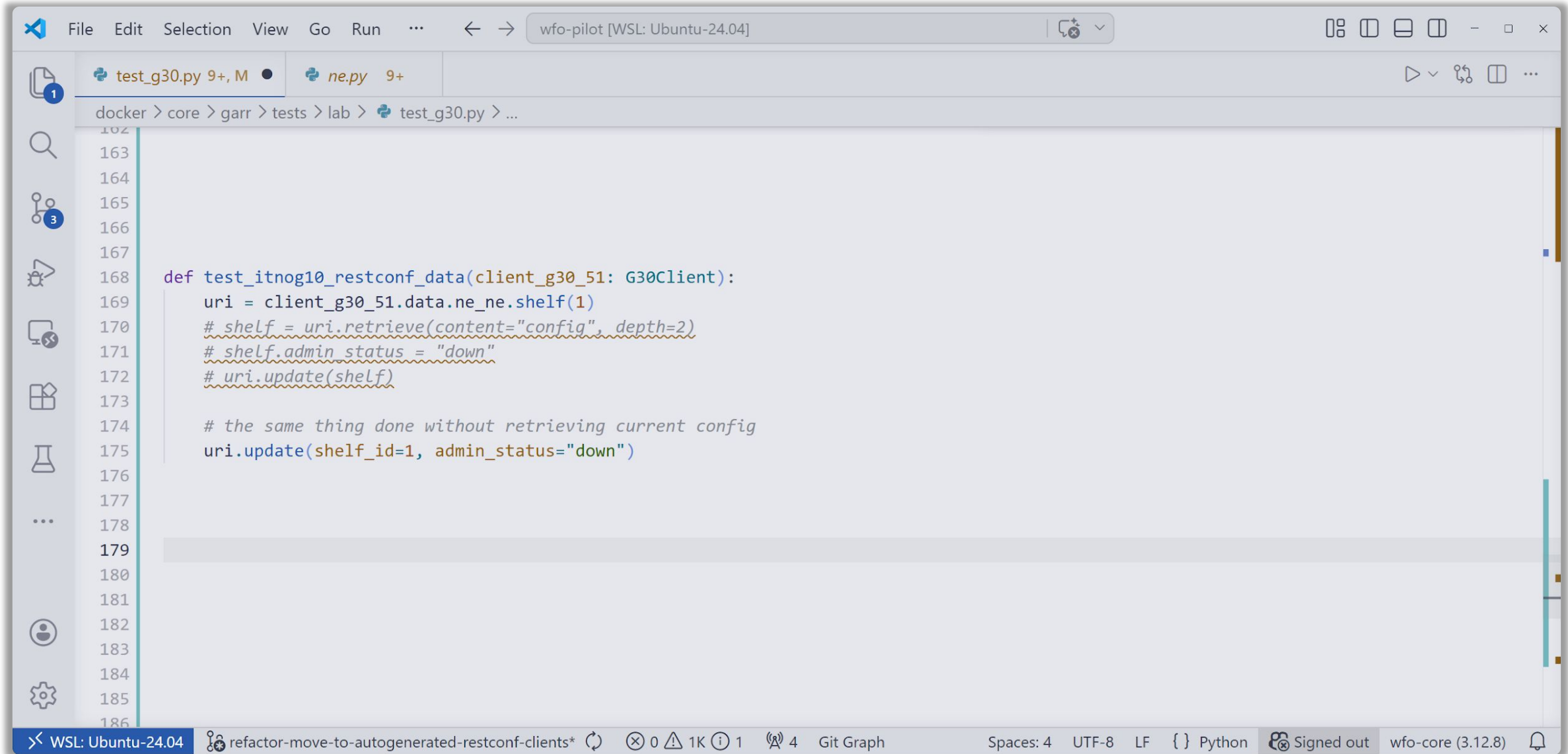
\* except for 'choice' and 'case' YANG statements

# CRUD on running datastore



```
File Edit Selection View Go Run ... wfo-pilot [WSL: Ubuntu-24.04]
test_g30.py 9+, M ne.py 9+
docker > core > garr > tests > lab > test_g30.py > ...
163
164
165
166
167
168 def test_itnog10_restconf_data(client_g30_51: G30Client):
169     uri = client_g30_51.data.ne_ne.shelf(1)
170     shelf = uri.retrieve(content="config", depth=2)
171     shelf.admin_status = "down"
172     uri.update(shelf)
173
174
175
176
177
178
179
180
181
182
183
184
185
186
WSL: Ubuntu-24.04 refactor-move-to-autogenerated-restconf-clients* 0 1K 1 4 Git Graph Spaces: 4 UTF-8 LF {} Python Signed out wfo-core (3.12.8)
```

# CRUD on running datastore

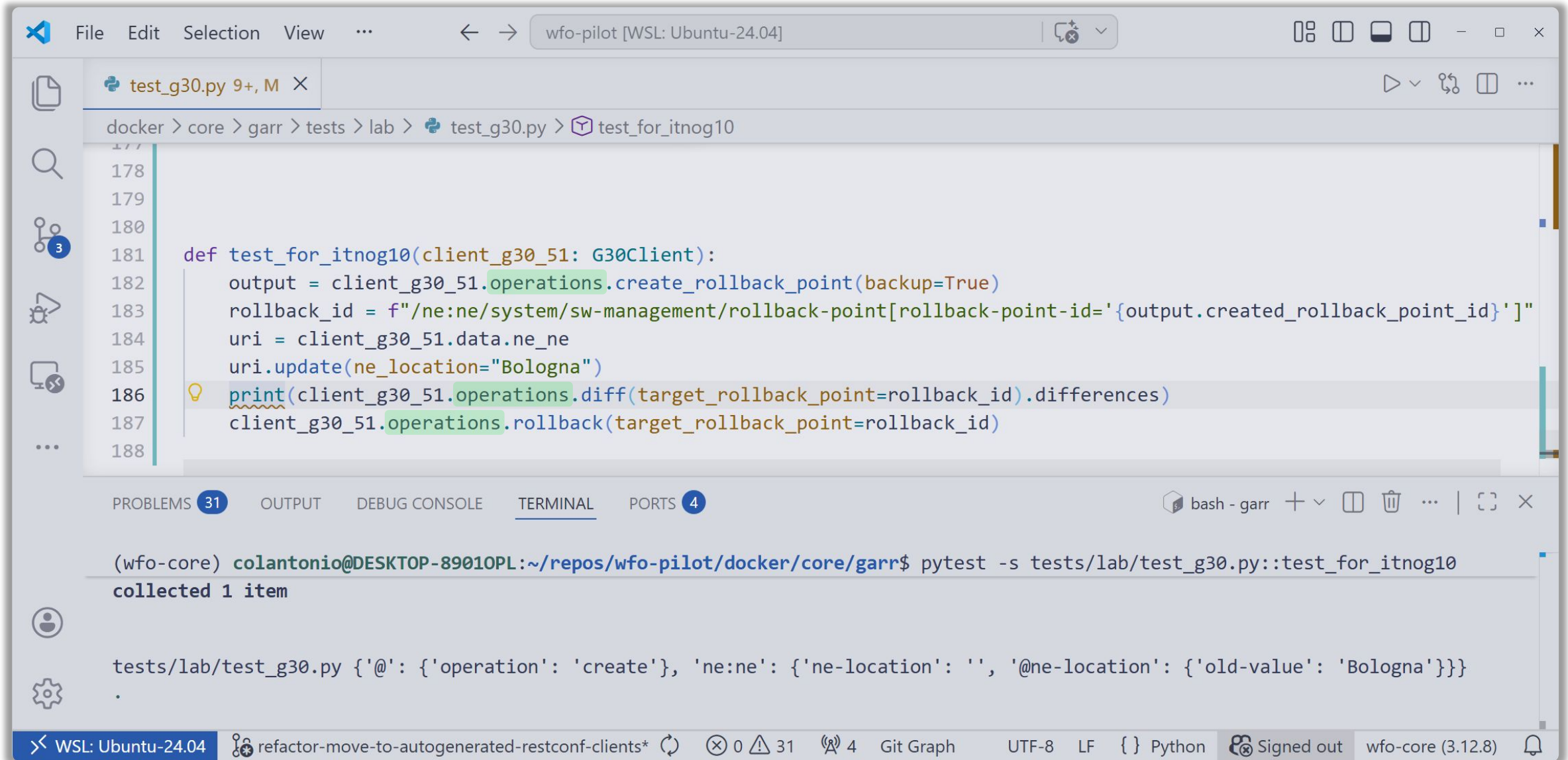


The screenshot shows a VS Code editor window titled "wfo-pilot [WSL: Ubuntu-24.04]". The editor is open to a file named "test\_g30.py" at line 168. The code defines a function "test\_itnog10\_restconf\_data" that interacts with a RESTCONF client. The code includes comments and function calls for retrieving and updating configuration data.

```
163
164
165
166
167
168 def test_itnog10_restconf_data(client_g30_51: G30Client):
169     uri = client_g30_51.data.ne_ne.shelf(1)
170     # shelf = uri.retrieve(content="config", depth=2)
171     # shelf.admin_status = "down"
172     # uri.update(shelf)
173
174     # the same thing done without retrieving current config
175     uri.update(shelf_id=1, admin_status="down")
176
177
178
179
180
181
182
183
184
185
186
```

The status bar at the bottom of the editor shows the following information: WSL: Ubuntu-24.04, refactor-move-to-autogenerated-restconf-clients\*, 0 errors, 1 warning, 1 info, 4 hints, Git Graph, Spaces: 4, UTF-8, LF, Python, Signed out, wfo-core (3.12.8).

# RPCs



The image shows a VS Code editor window with a Python file named `test_g30.py` open. The file is located at `docker > core > garr > tests > lab > test_g30.py`. The code defines a function `test_for_itnog10` that uses a `G30Client` to create a rollback point, update its location to 'Bologna', and then diff the current state against the target rollback point. The terminal output shows the test passing successfully.

```
def test_for_itnog10(client_g30_51: G30Client):
    output = client_g30_51.operations.create_rollback_point(backup=True)
    rollback_id = f"/ne:ne/system/sw-management/rollback-point[rollback-point-id='{output.created_rollback_point_id}']"
    uri = client_g30_51.data.ne_ne
    uri.update(ne_location="Bologna")
    print(client_g30_51.operations.diff(target_rollback_point=rollback_id).differences)
    client_g30_51.operations.rollback(target_rollback_point=rollback_id)
```

Terminal output:

```
(wfo-core) colantonio@DESKTOP-8901OPL:~/repos/wfo-pilot/docker/core/garr$ pytest -s tests/lab/test_g30.py::test_for_itnog10
collected 1 item

tests/lab/test_g30.py {'@': {'operation': 'create'}, 'ne:ne': {'ne-location': '', '@ne-location': {'old-value': 'Bologna'}}}
```

# Custom templating

```
wfo-pilot [WSL: Ubuntu-24.04]
ne.py .../data_models 9+
ne.py .../user_templates 1 x
docker > core > garr > services > nokia > g30 > user_templates > ne.py > ...
11 @TemplateRegistry.register("InterfaceListNode")
12 def interface_list_template(
13     *,
14     lo_ip: str | None,
15     lo_name: str | None,
16     eth1_ip: str | None,
17     eth1_prefix_length: int | None,
18     osc_names: list[str] | None,
19 ) -> list[ne.InterfaceItem]:
20     from ..data_models.ne import InterfaceItem # noqa
21
22     interfaces = []
23
24     # Loopback
25 > if lo_ip: ...
43
26     # eth1
44 > if eth1_ip: ...
73
27     # OSCX
74
75 > for osc_name in osc_names or []: ...
105
106     return interfaces
107
```

```
wfo-pilot [WSL: Ubuntu-24.04]
ne.py .../data_models 9+
ne.py .../user_templates 1
optical_device.py 1
docker > core > garr > products > services > optical_device.py > ...
345 intf_navigator = g30.data.ne_ne.system.networking.interface
346 desired_intf_config = intf_navigator.from_template(
347     lo_ip=lo_ip,
348     lo_name=lo_intf_name,
349     eth1_ip=eth1_ip,
350     eth1_prefix_length=eth1_prefix_len,
351     osc_names=osc_names,
352 )
353
354 rtp_navigator = g30.data.ne_ne.system.networking.routing.protocol
355 desired_rtp_config = rtp_navigator.from_template(
356     ospf_router_id=lo_ip,
357     is_ospf_asbr=is_g30_connected_to_switch,
358     oscx_intf_names=oscx_intf_names,
359     eth1_intf_name=eth1_intf_name,
360     eth1_default_gateway=eth1_gateway,
361     eth1_default_out_intf_name=eth1_intf_name,
362 )
363
364 intf_diffs = compare_pydantic_objects(
365     expected=desired_intf_config,
366     actual=intf_config,
367     unique_id_keys=["if-name"],
368 )
```

# Fine! Domande?

matteo.colantonio@garr.it

www.garr.it

