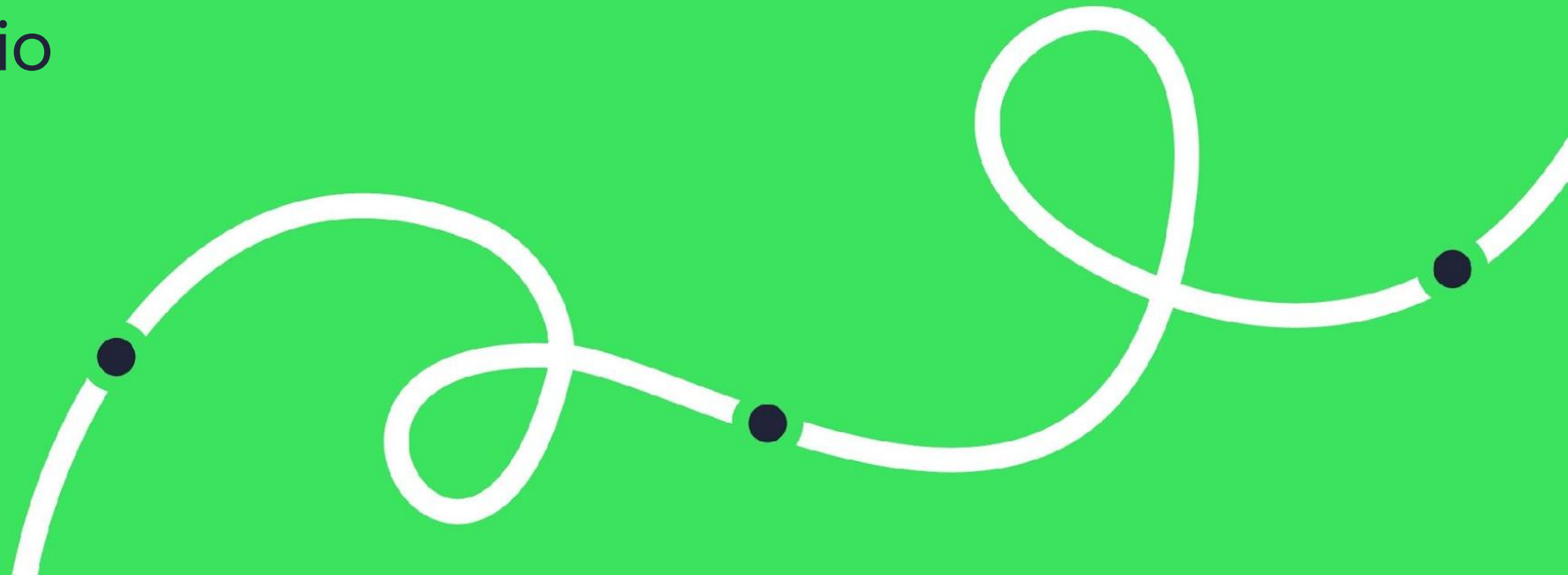
Semplificare l'automazione multivendor
client RESTCONF da moduli YANG

Pydantic-based IDE-friendly RESTCONF clients from YANG

Matteo Colantonio

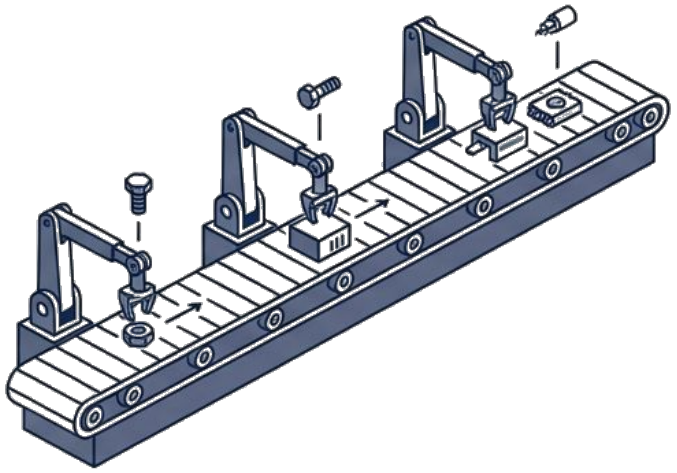


Consortium GARR

Network Orchestration

workfloworchestrator.org

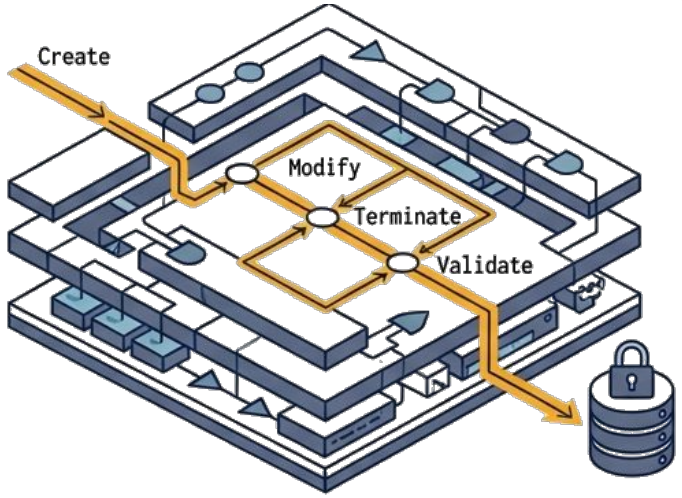
Automation



Executing repetitive tasks reliably.

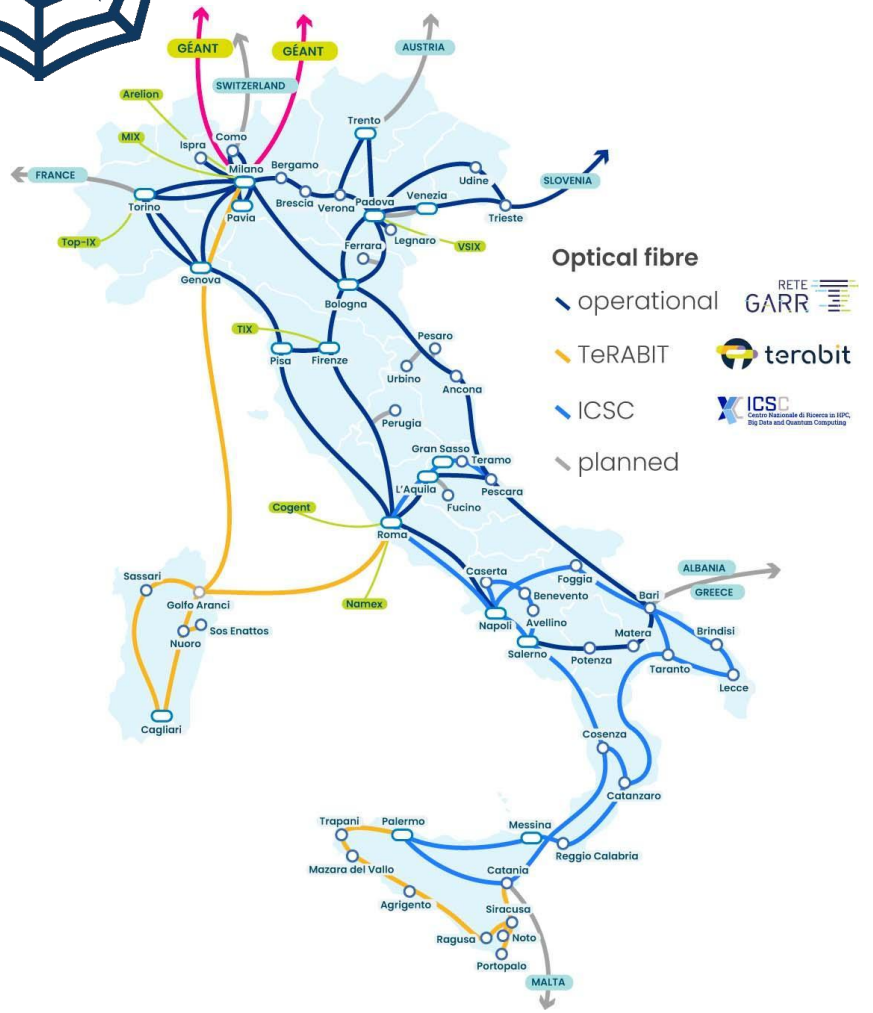
Focuses on single actions (e.g. config templating) with no or low abstraction

Orchestration

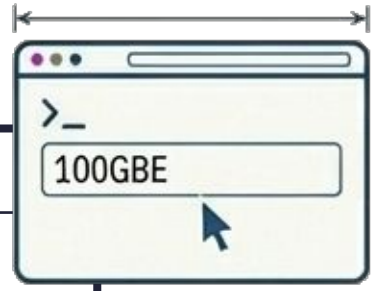


Adding **abstract** and **stateful** layer decoupled from HW.

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

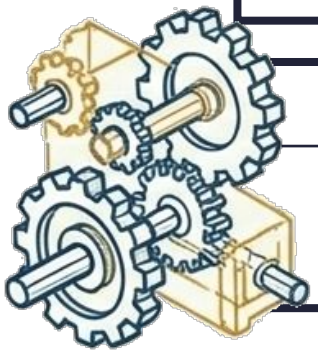


Network Orchestration



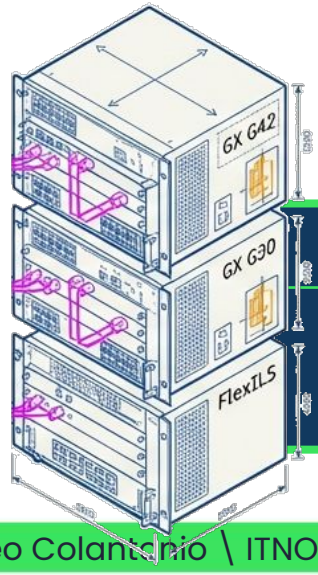
Intent

Orchestrator UI & API
User input through guided forms



Engine

PostgreSQL: current network **abstract state** & lifecycle history
Workflows: Python code with **retriable polymorphic** steps



Vendor NMS

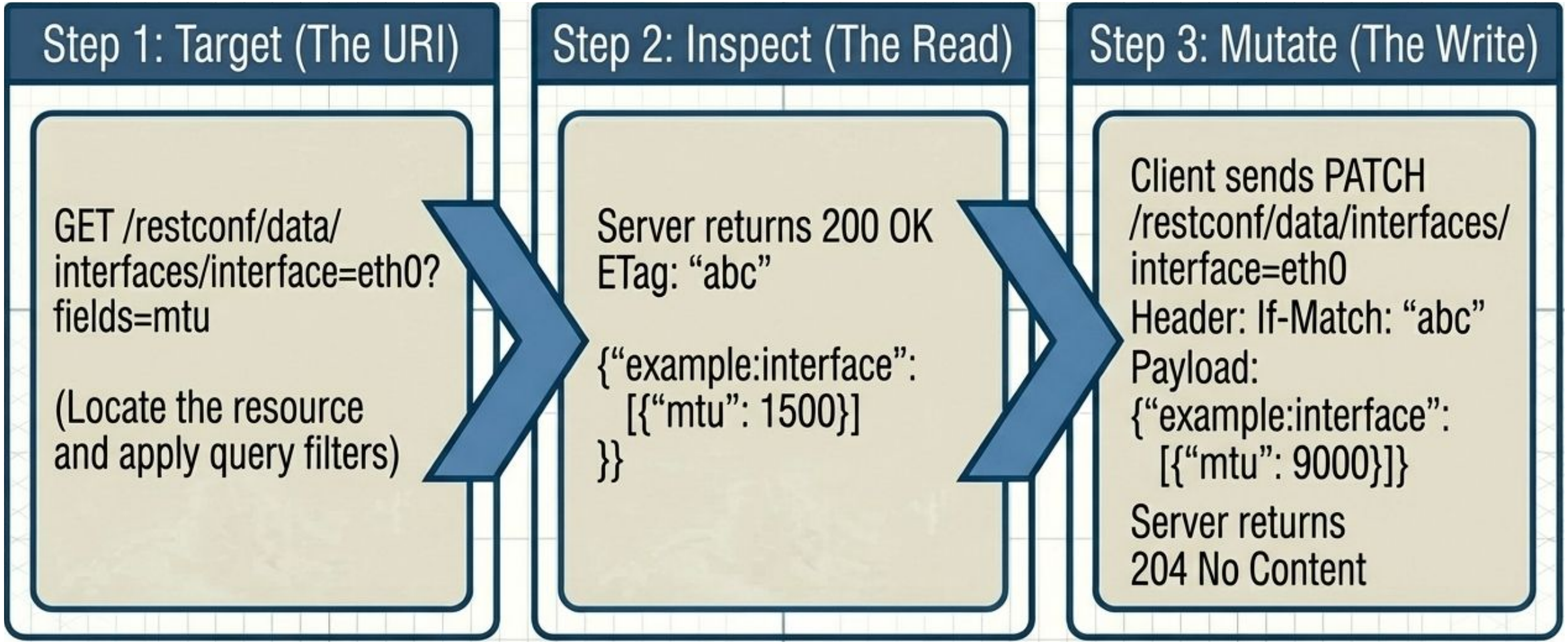
RESTCONF / TL1

```
{
  "data": {
    "interface":
     "GX G42",
    "frequency":
     "192.30THz",
    ...
  }
}
```

Execution

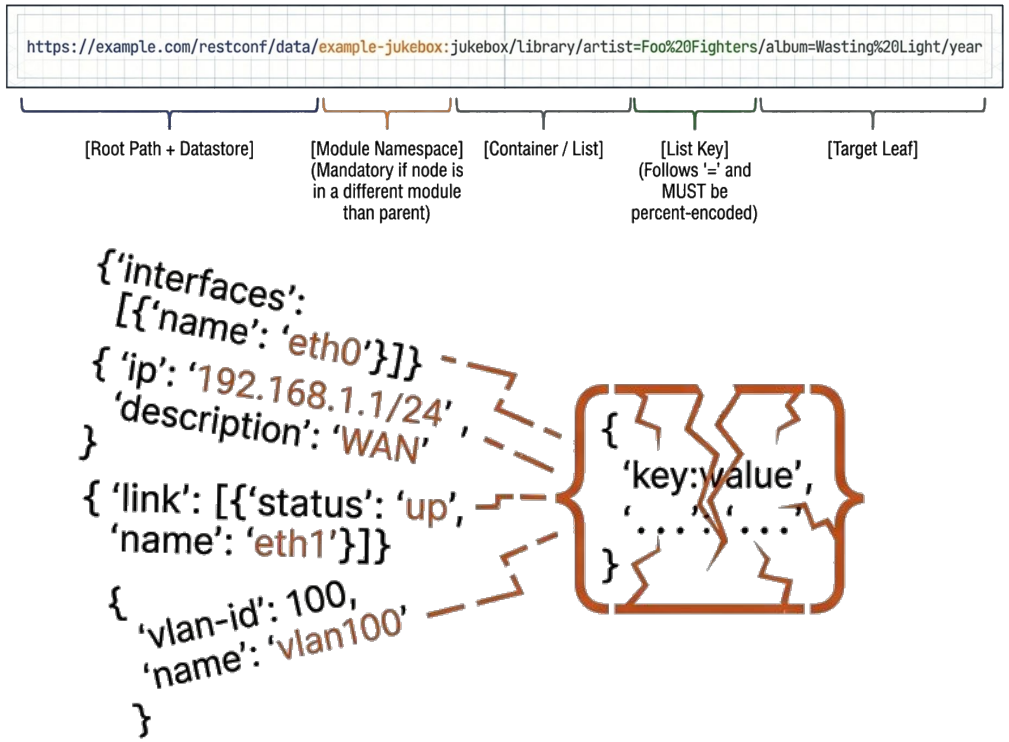
Network Devices [Nokia FlexILS, G30, G42]
& Systems [Netbox]

End-to-end RESTCONF interaction



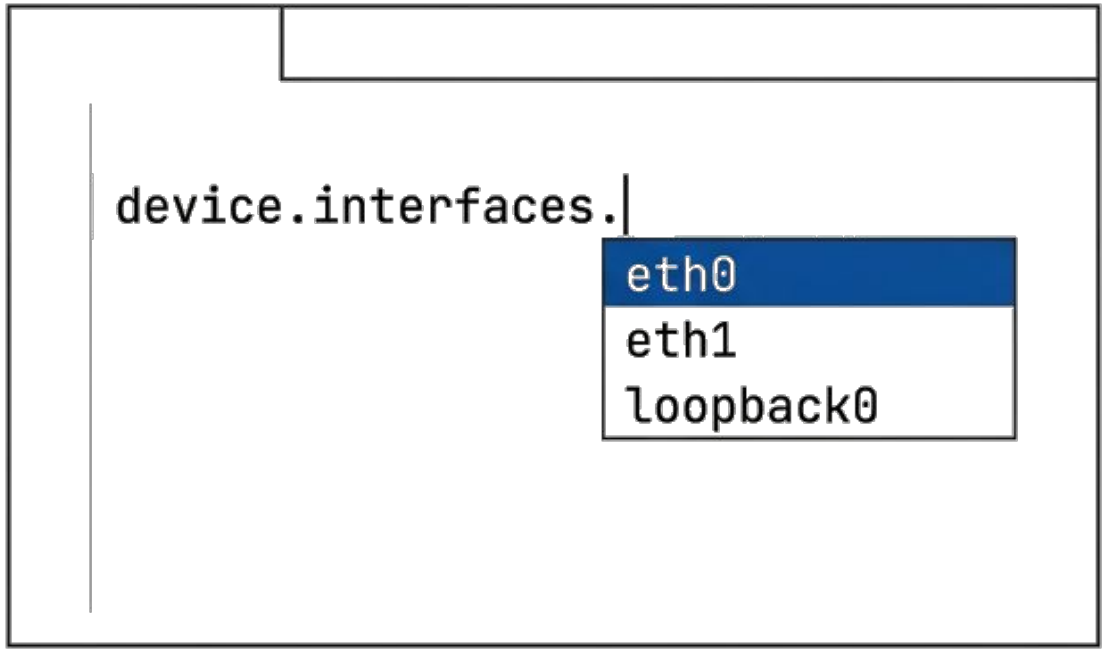
Using raw RESTCONF is fragile

Raw



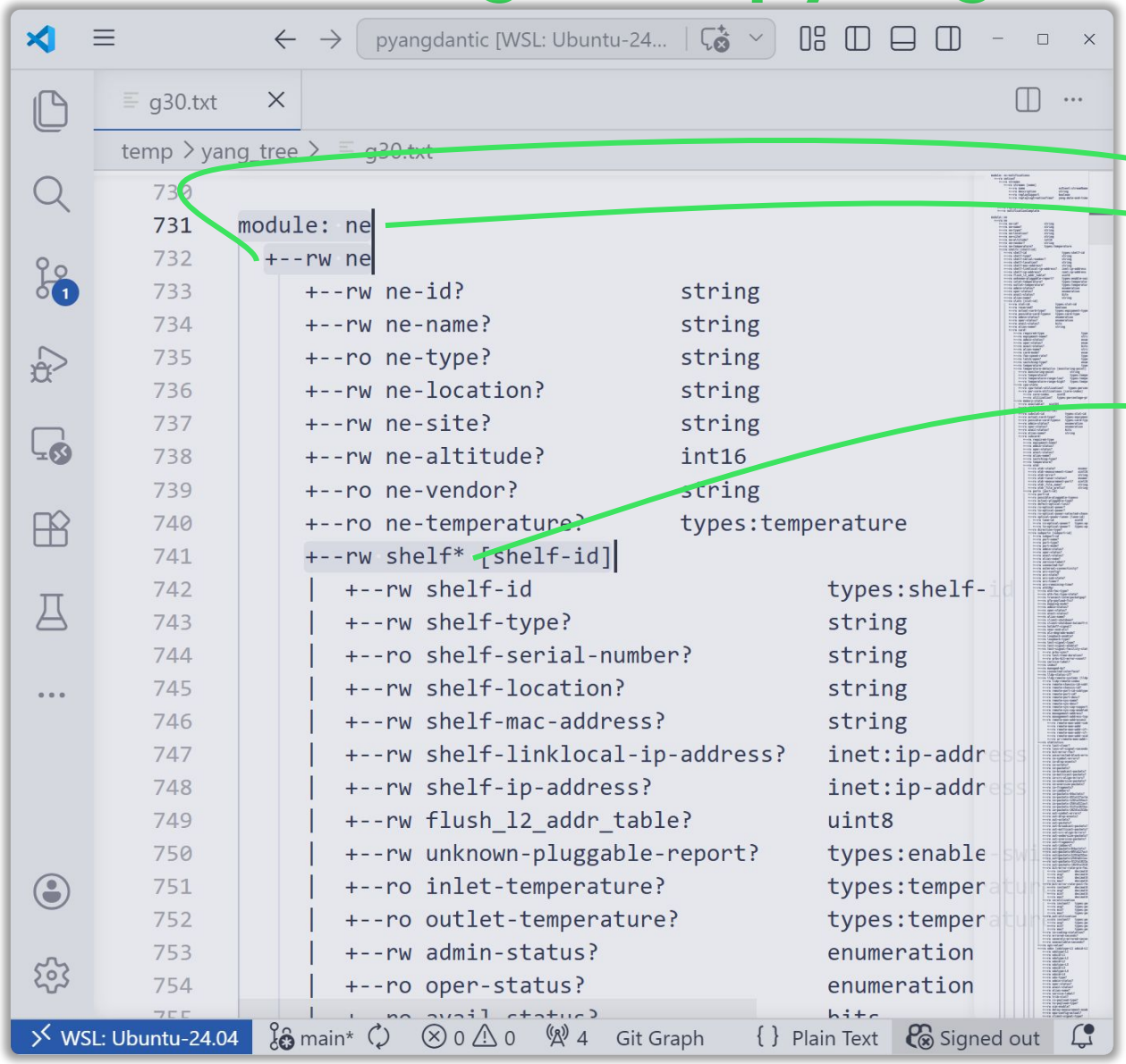
Managing unvalidated JSON and URIs leads to runtime **errors**, **tedious** boilerplate, and **brittle** automation scripts.

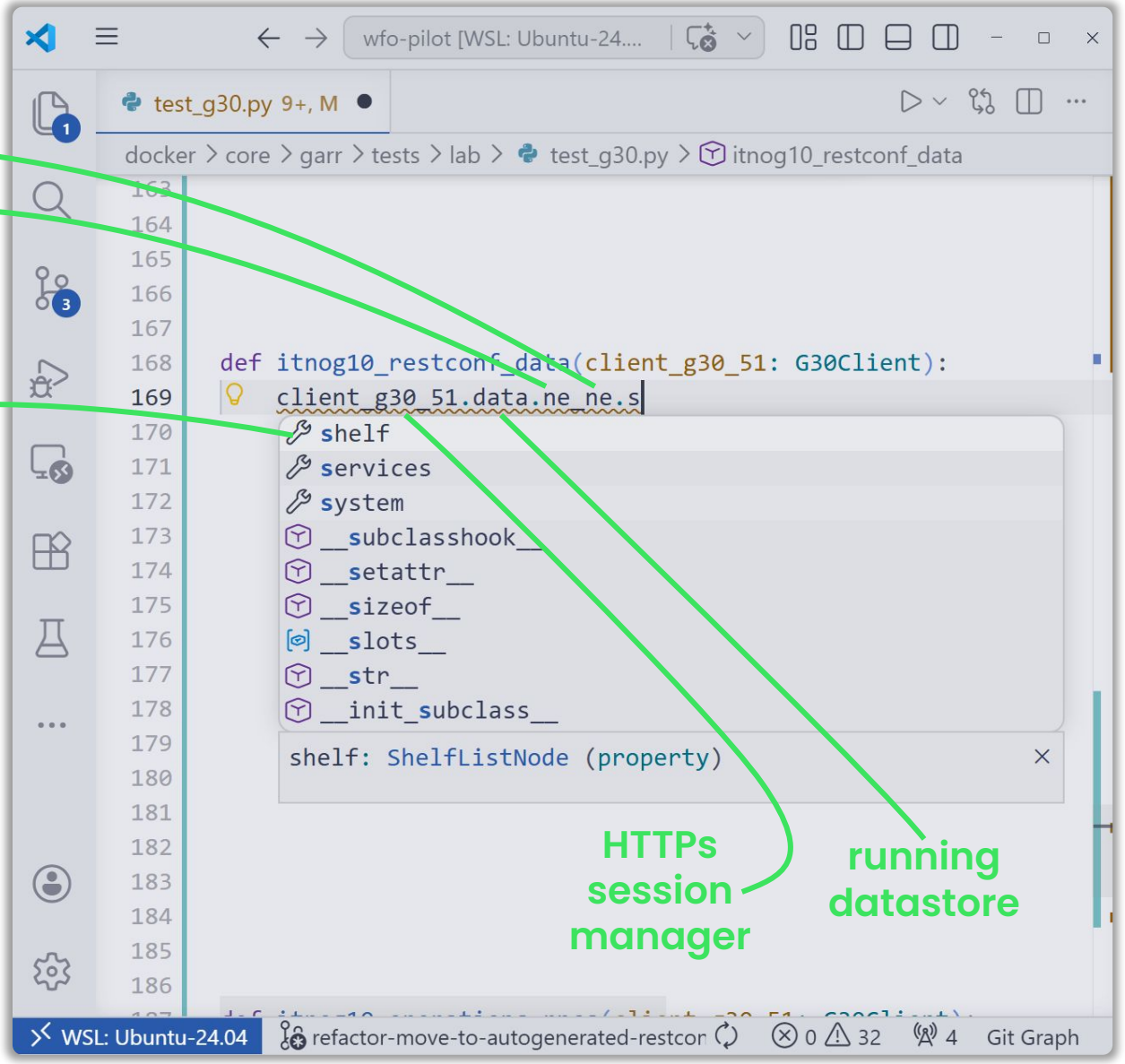
Ideal



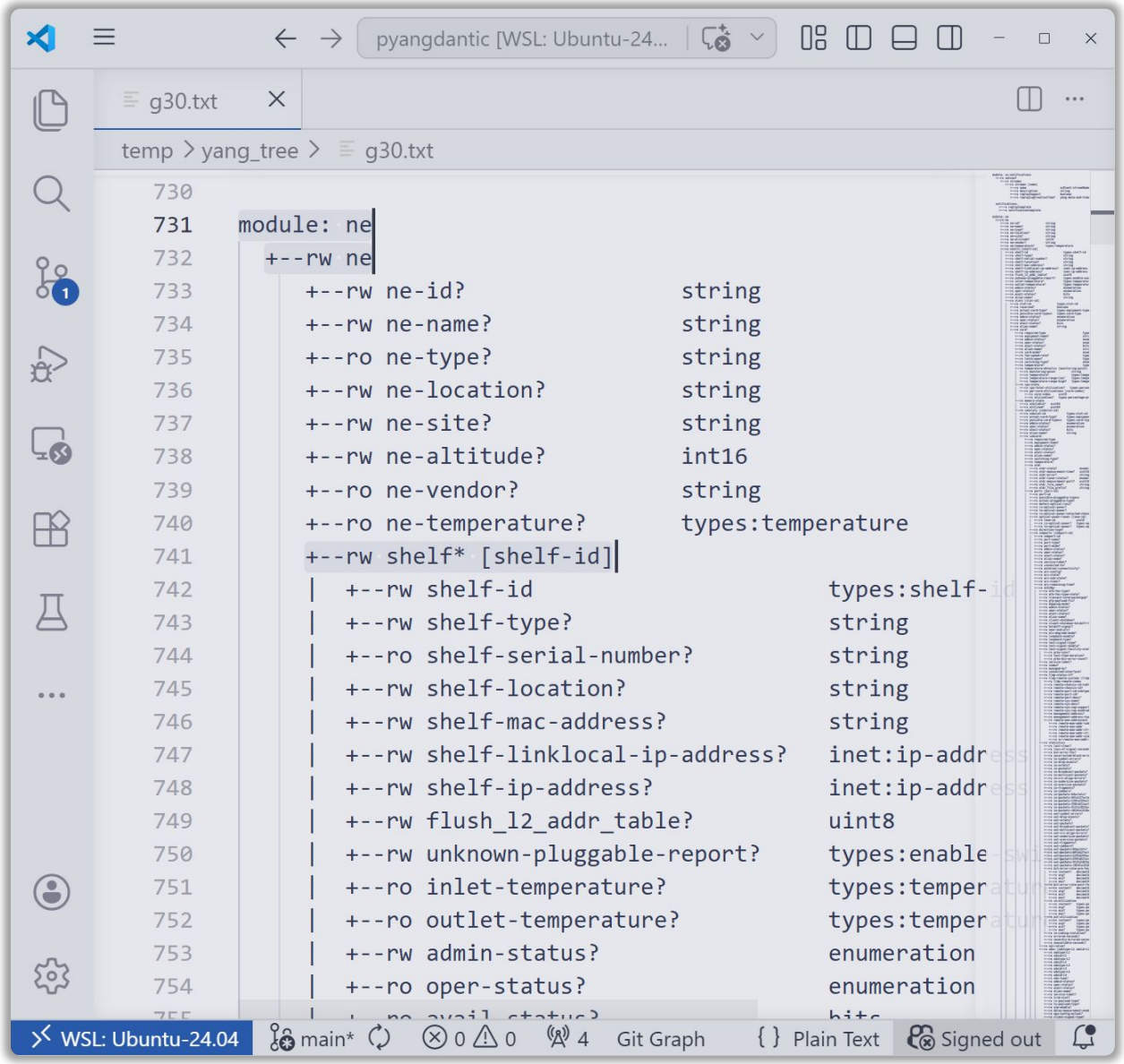
I want an IDE-friendly interface with **autocomplete** and **validation** that tracks **YANG** modules and their release changes

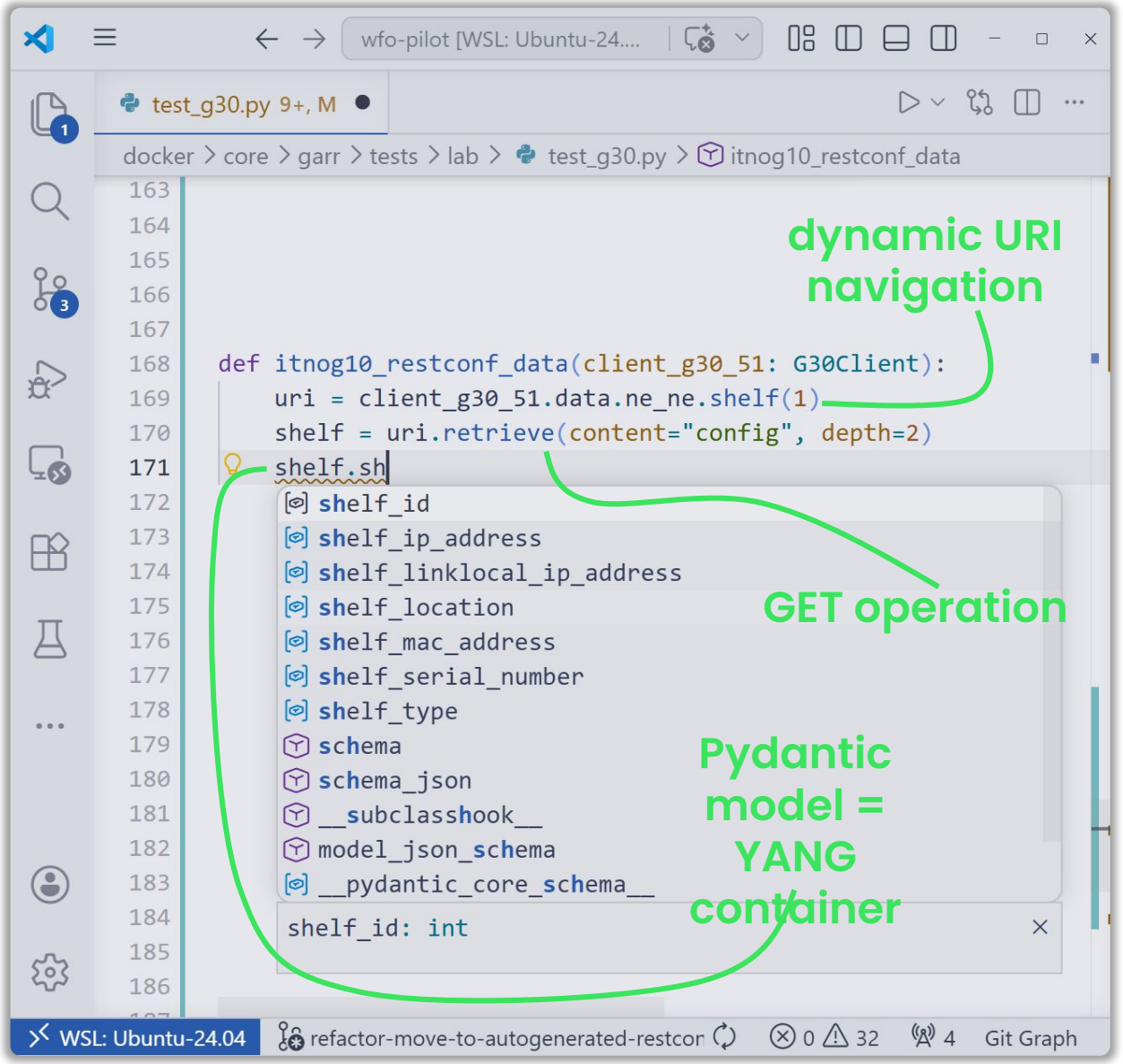
So I built u.garr.it/pyangdantic



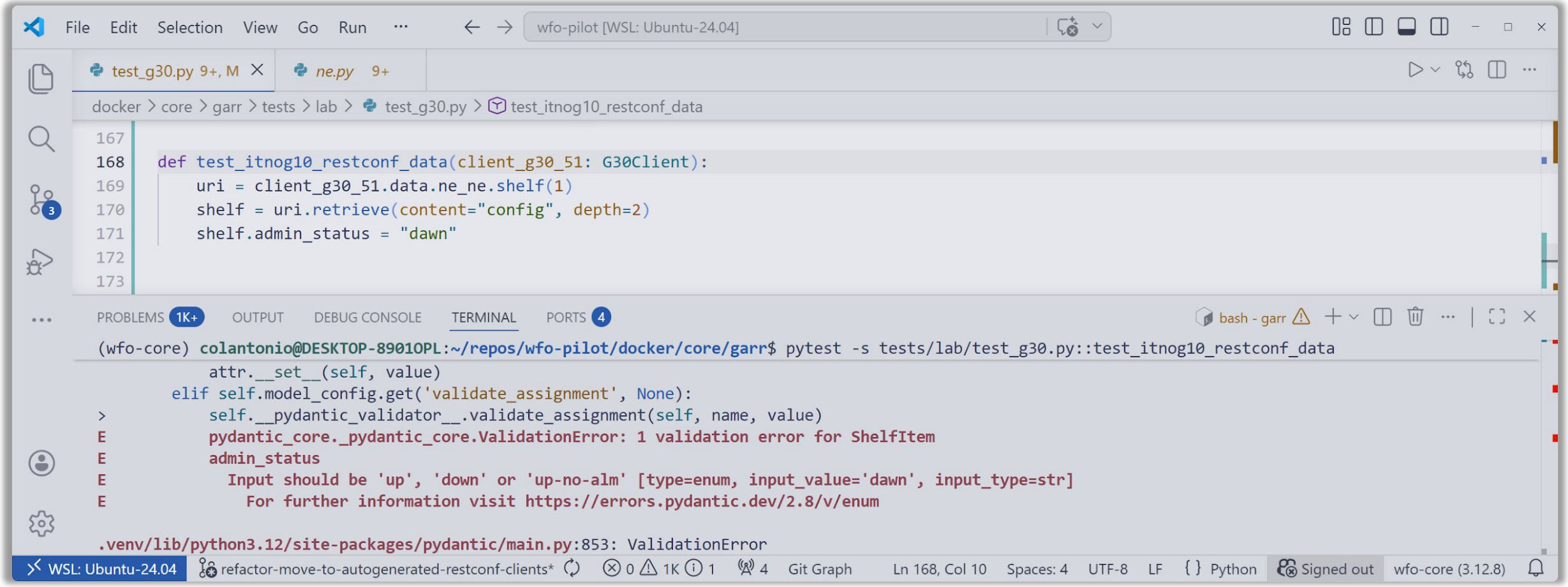


Autocomplete & type safety



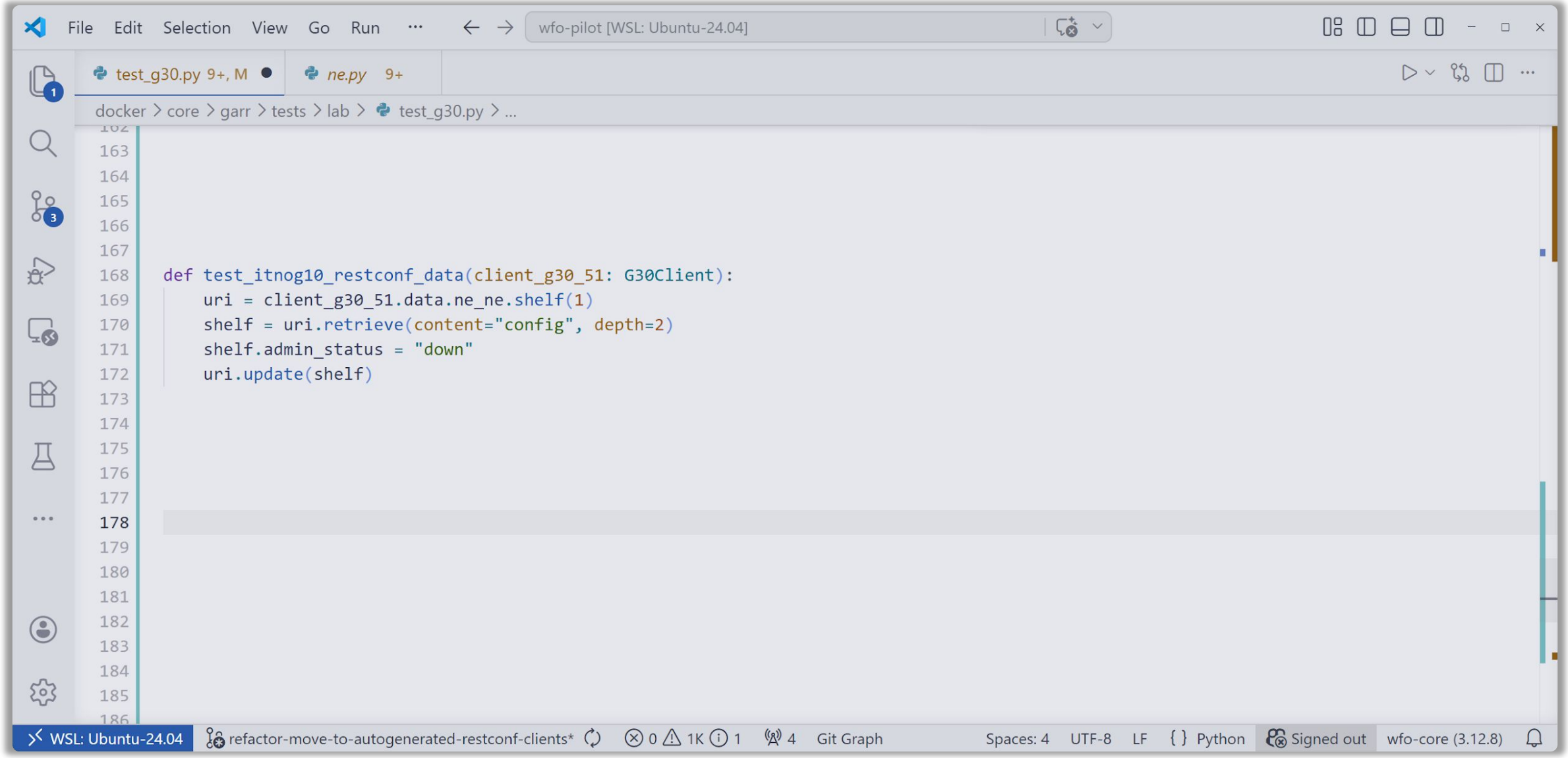


Validation according to YANG rules*

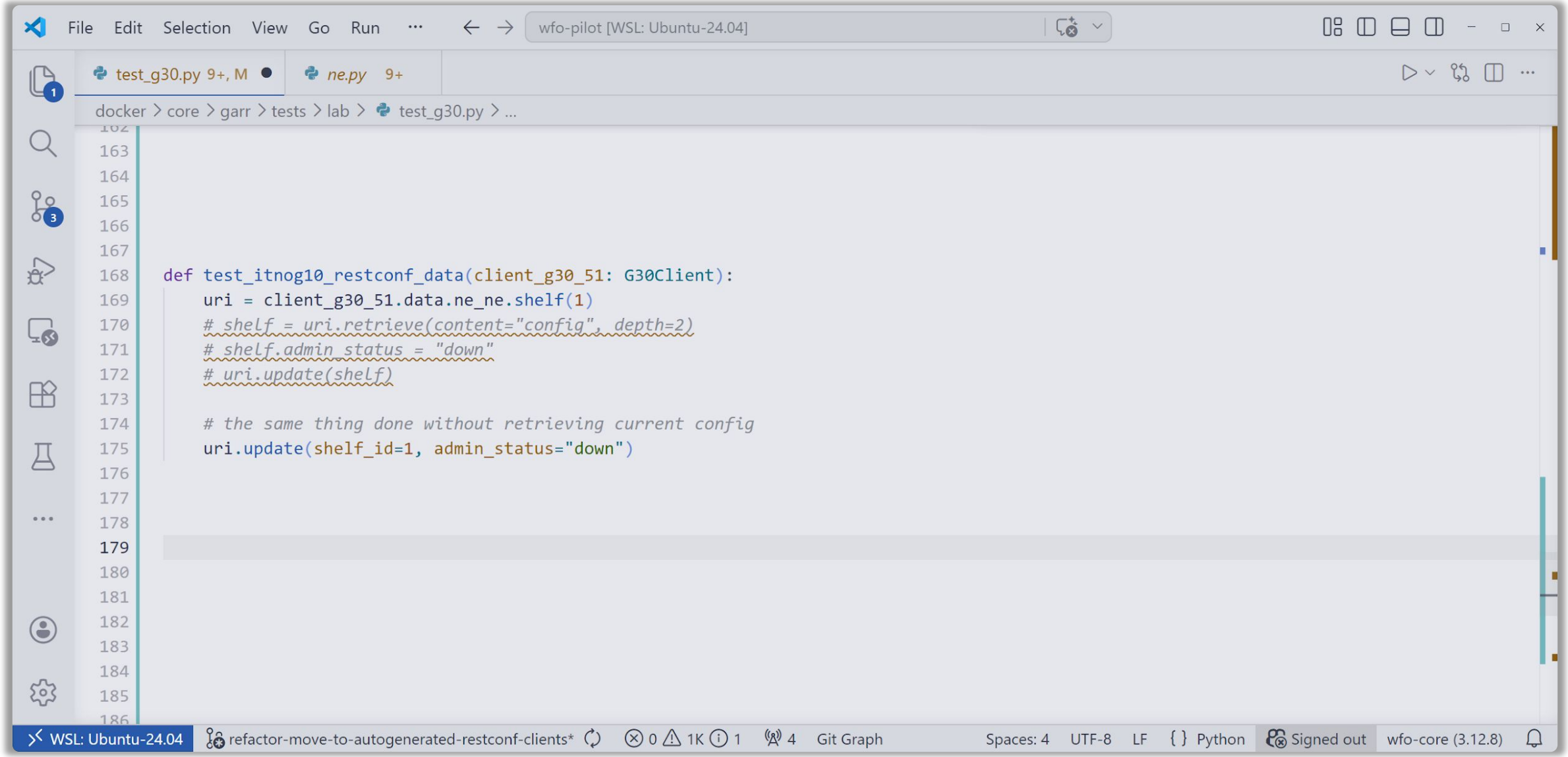


* except for 'choice' and 'case' YANG statements

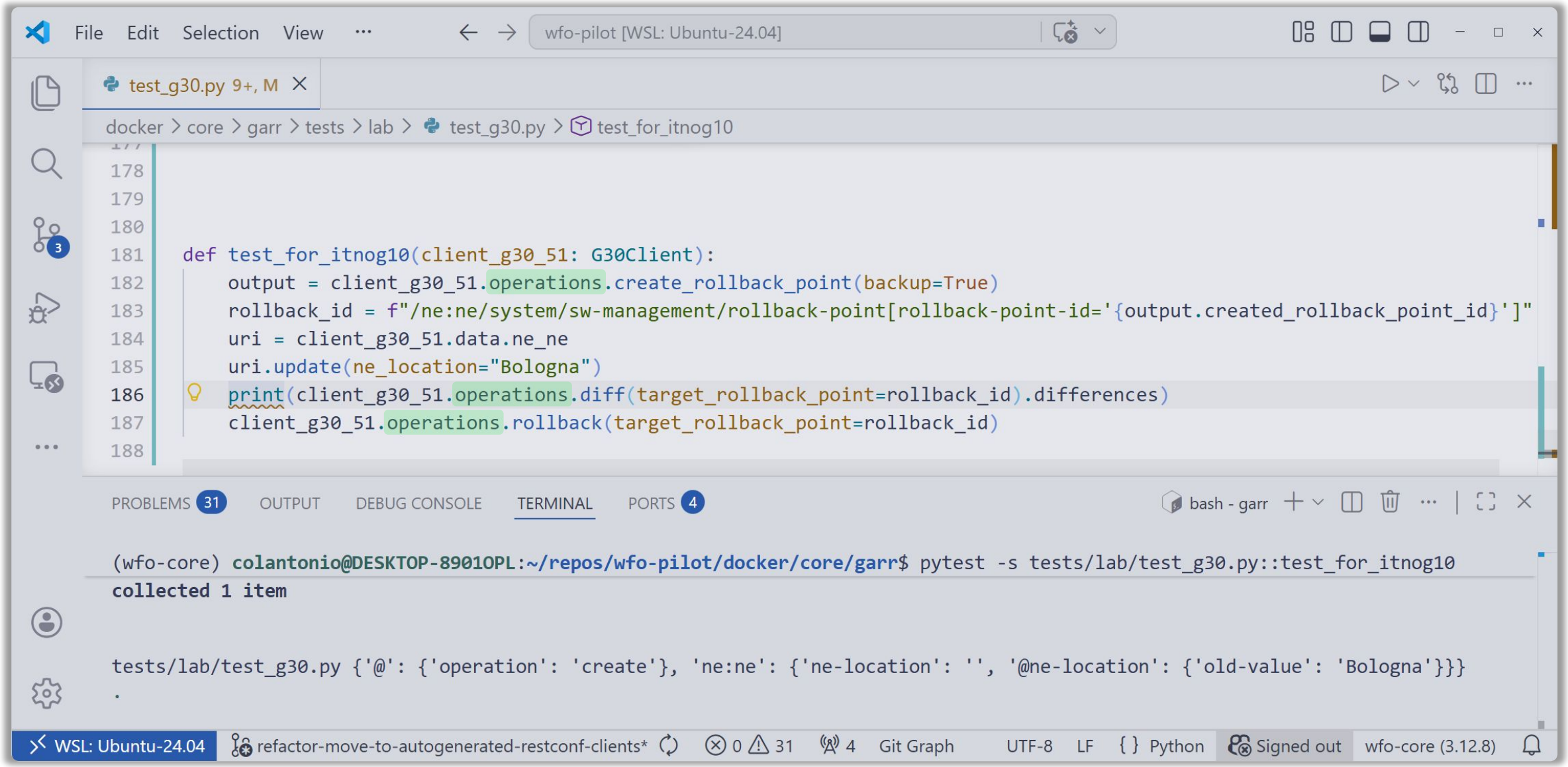
CRUD on running datastore



CRUD on running datastore



RPCs


File Edit Selection View ···
wfo-pilot [WSL: Ubuntu-24.04]
test_g30.py 9+, M
docker > core > garr > tests > lab > test_g30.py > test_for_itnog10
178
179
180
181 def test_for_itnog10(client_g30_51: G30Client):
182 output = client_g30_51.operations.create_rollback_point(backup=True)
183 rollback_id = f"/ne:ne/system/sw-management/rollback-point[rollback-point-id='{output.created_rollback_point_id}']"
184 uri = client_g30_51.data.ne_ne
185 uri.update(ne_location="Bologna")
186 print(client_g30_51.operations.diff(target_rollback_point=rollback_id).differences)
187 client_g30_51.operations.rollback(target_rollback_point=rollback_id)
188
PROBLEMS 31 OUTPUT DEBUG CONSOLE TERMINAL PORTS 4
bash - garr
(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'}}}
.
WSL: Ubuntu-24.04
refactor-move-to-autogenerated-restconf-clients*
0 31
4
Git Graph
UTF-8
LF
Python
Signed out
wfo-core (3.12.8)

Custom templating

wfo-pilot [WSL: Ubunt...

ne.py .../data_models 9+ | ne.py .../user_templates 1

docker > core > garr > services > nokia > g30 > user_templates > ne.py > ...

```python
@TemplateRegistry.register("InterfaceListNode")
def interface_list_template(
    *,
    lo_ip: str | None,
    lo_name: str | None,
    eth1_ip: str | None,
    eth1_prefix_length: int | None,
    osc_names: list[str] | None,
) -> list[ne.InterfaceItem]:
    from ..data_models.ne import InterfaceItem  # noqa

    interfaces = []

    # Loopback
    if lo_ip: ···

    # eth1
    if eth1_ip: ···

    # OSCX
    for osc_name in osc_names or []: ···

    return interfaces
```

WSL: Ubuntu-24.04 | refactor-move-to-autogenerated-r | 0 1K 1 | 4

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 > ...

```python
    intf_navigator = g30.data.ne_ne.system.networking.interface
    desired_intf_config = intf_navigator.from_template(
        lo_ip=lo_ip,
        lo_name=lo_intf_name,
        eth1_ip=eth1_ip,
        eth1_prefix_length=eth1_prefix_len,
        osc_names=osc_names,
    )

    rtp_navigator = g30.data.ne_ne.system.networking.routing.routing_protocol
    desired_rtp_config = rtp_navigator.from_template(
        ospf_router_id=lo_ip,
        is_ospf_asbr=is_g30_connected_to_switch,
        oscx_intf_names=oscx_intf_names,
        eth1_intf_name=eth1_intf_name,
        eth1_default_gateway=eth1_gateway,
        eth1_default_out_intf_name=eth1_intf_name,
    )

    intf_diffs = compare_pydantic_objects(
        expected=desired_intf_config,
        actual=intf_config,
        unique_id_keys=["if-name"],
    )
```

Ubuntu-24.04 | refactor-move-to-autogenerated-restconf-clients* | 0 1K 1 | 4 | Git Graph

Fine! Domande?

matteo.colantonio@garr.it

www.garr.it

Consortium GARR