neops
network automation
orchestration

SwiNOG #41

Building a Vendor-Agnostic E2E Testing Ecosystem with Netlab

Leandro Lerena & Andy Griesbeck

28.04.2026

Agenda

Building a Vendor-Agnostic E2E Testing Ecosystem with Netlab

Workflow as a Transaction

- Motivation for a transactional model
- How to execute a workflow as a transaction
- Remaining real-world challenges

Leandro Lerena
Software Architect
zebbra AG

Testing your Code

- Building network automation that behaves like software
- CI-runnable tests against Cisco IOL, SR Linux, FRR — pick your fixture
- Vendor agnosticism as a consequence of good abstractions, not a marketing claim

Andy Griesbeck
Software Engineer
zebbra AG

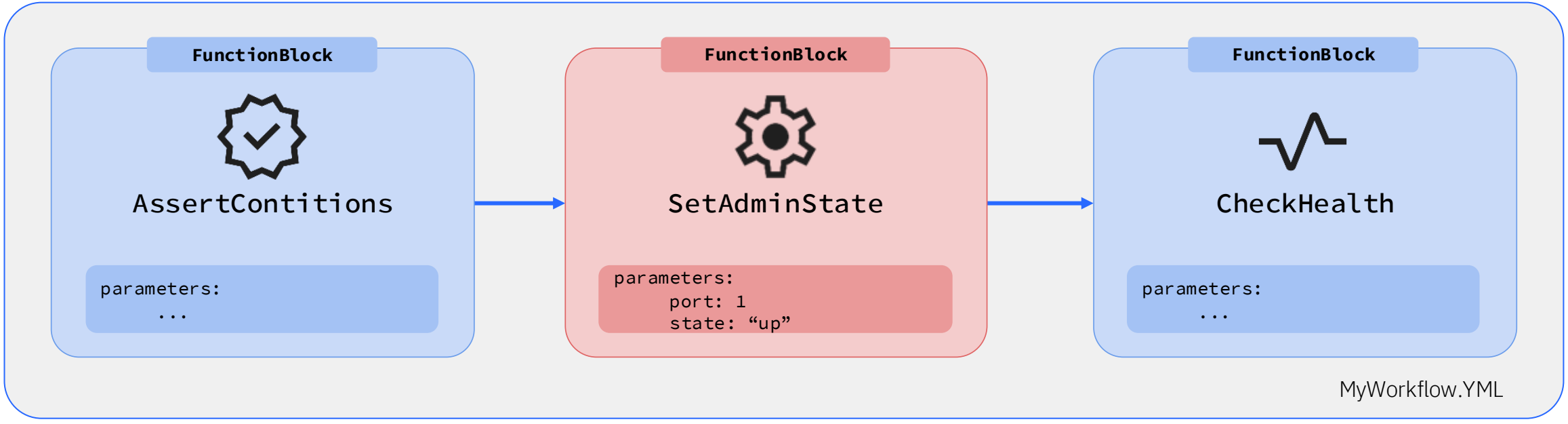
Workflow as a Transaction

aka.

a nice perfect-in-theory transactional model for workflows

Leandro Lerena
Software Architect
zebbra AG

Consequences from a missing transactional model



Interpretability

What does a **failing operation** mean? Is it reason to **escalate** or can we **analyze it calmly** tomorrow?

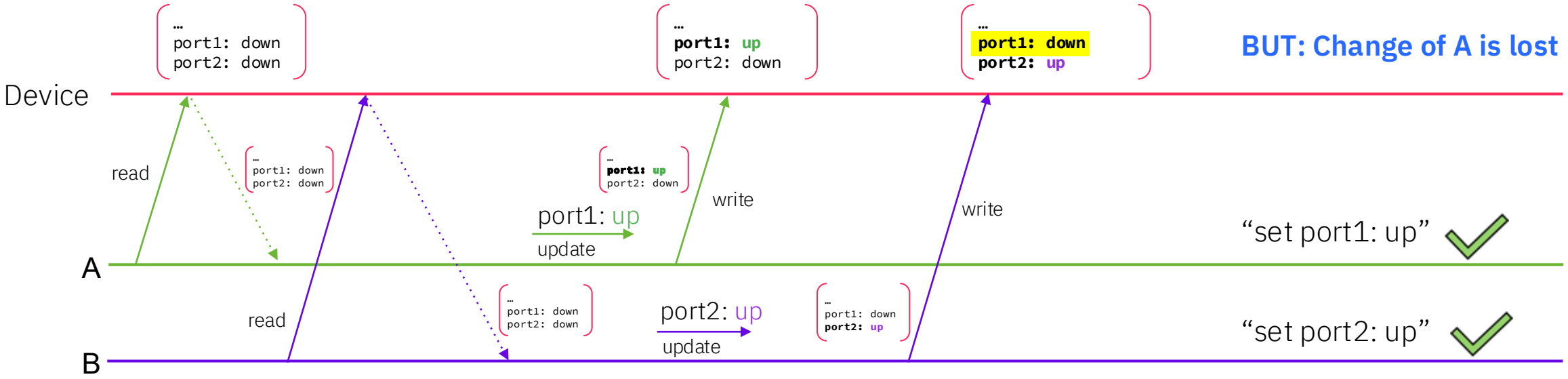
Auditability

What **operations** did (potentially) **touch adevice**? What was the **intent**?

Race conditions

Even when executing simple scripts, **race conditions** that e.g. override an effect may occur, leading to **inconsistent state.**

If a config update is not atomic, with actors A and B



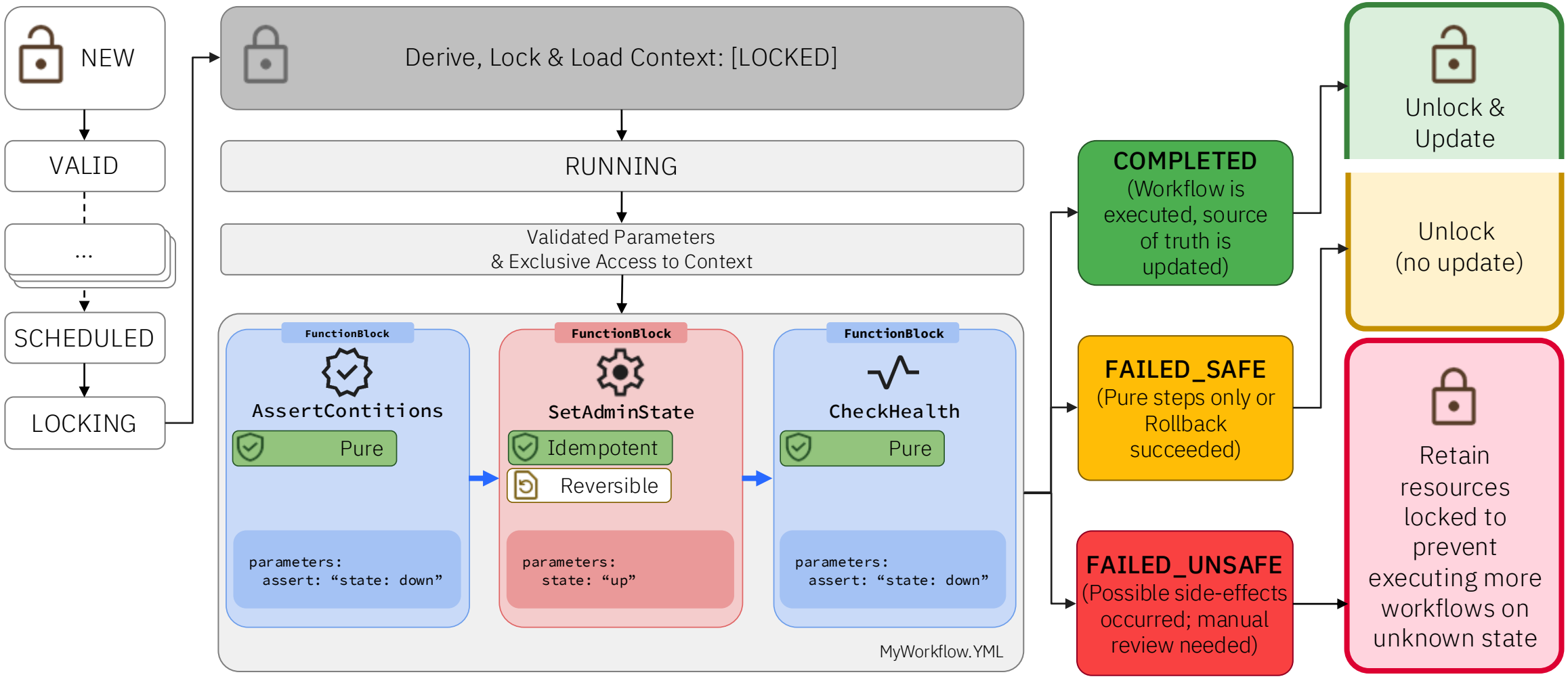
Atomicity

Most operations are **not atomic by nature.**

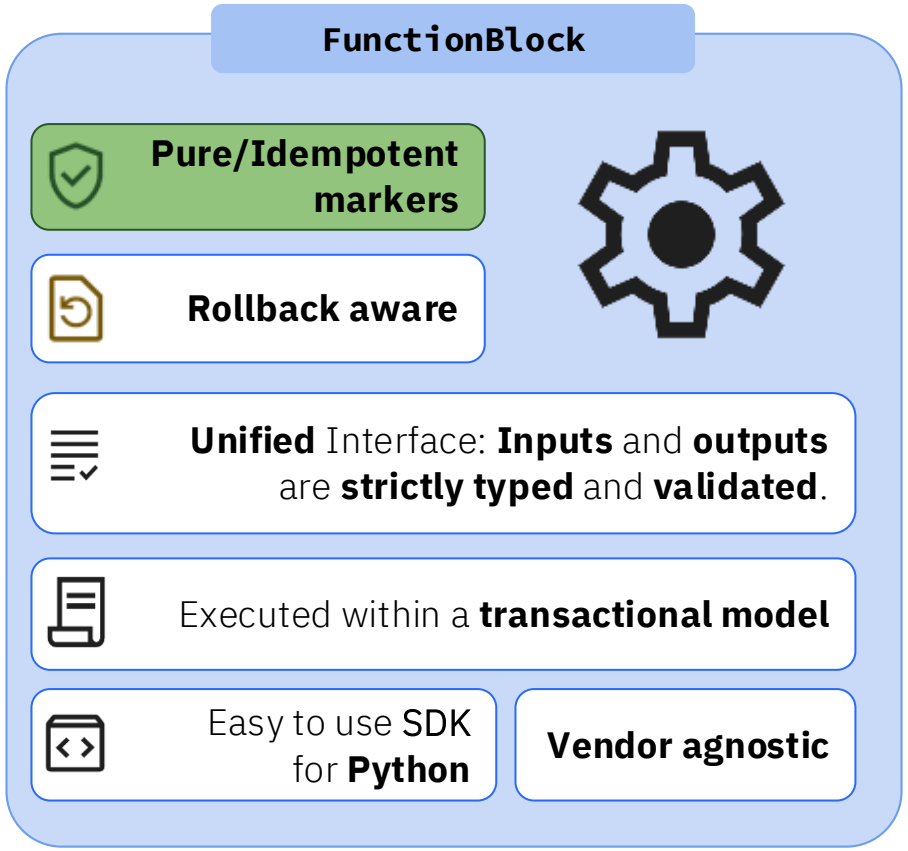
An unreachable goal

For most **complex operations**, **atomicity** is an **unreachable goal**. Need of a **central orchestrator.**

Workflow as a transaction



When concepts hit the real world



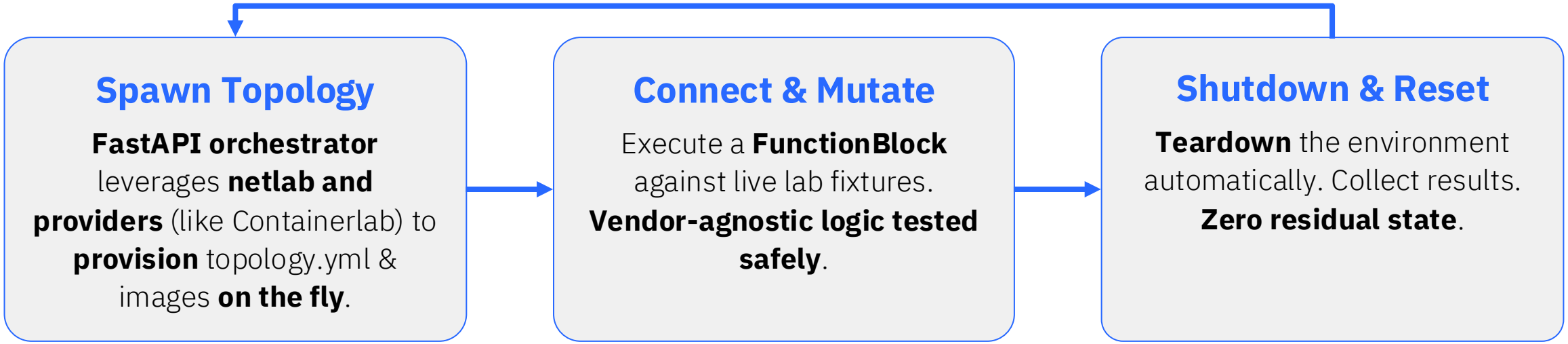
The challenge

- The **quality** of a workflow **depends strongly** on its **function blocks**.
- **Vendor agnosticity** needs **actual testing** on **different** vendors.
- Need to **eliminate** as much **friction** as possible

Netlab to the rescue (SwiNOG #40)

- **Describe** network topologies in a topology.yaml.
- **Spawn virtual networks** from predefined topologies

The Netlab Orchestrator Workflow



Exclusive Access

With requests coming from **IDEs** and **CI/CD** pipelines.

Brings in the power of netlab

Just drop in your images and use them in **topology.yml.**

One simple python decorator

Provide your **parameters**, **assertions** and the **topology.yml**, the rest is abstracted away!

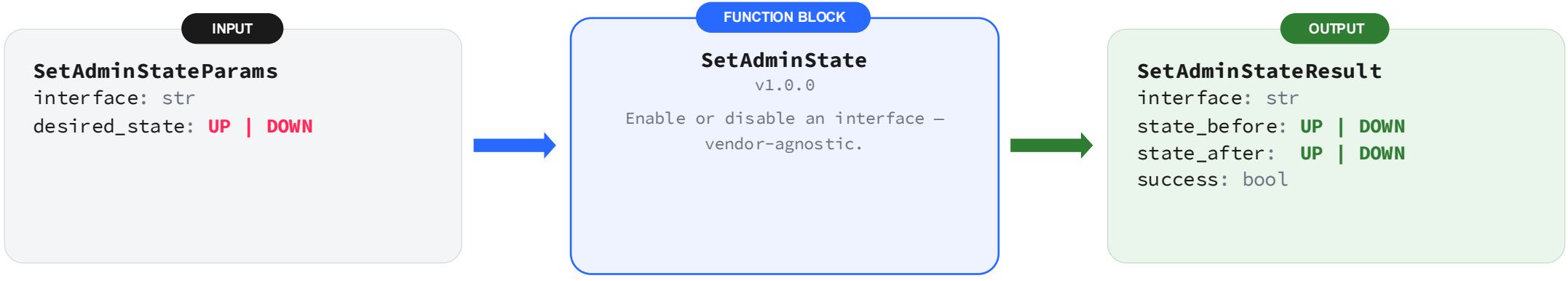
Testing your Code

aka.

the thing everyone knows is great but often skips

Andy Griesbeck
Software Engineer
zebbra AG

Our guinea pig: vendor-agnostic SetAdminState




```python
@register_function_block(
    name          = "set_admin_state",
    version       = (1, 0, 0),
    run_on        = "device",
    is_idempotent = True, # -> if it fails, the orchestrator can retry ;)
)
class SetAdminState(FunctionBlock):
    async def run(self, p, ctx):
         with AdminStateProxy.connect(ctx.device) as conn:
            before = conn.get_interface_admin_state(p.interface)
            conn.set_admin_state(p.interface, p.desired_state)
            after  = conn.get_interface_admin_state(p.interface)
        return FunctionBlockResult(
            success = after is p.desired_state,
            data    = SetAdminStateResult(p.interface, before, after)
        )
```

Testing the guinea pig

FUNCTION BLOCK

SetAdminState

UNIT

```
@fb_test_case(
    ”Fails with no device",
    params={...},
    succeeds=false
)
```

∅

no device — pure logic

LAB

```
@fb_test_case_with_lab(
    ”Shut Ethernet0/1 on Cisco IOL",
    remote_lab_fixture=simple_iol,
    params={...},
    assertions=[port_is_down]
)
```

Cisco IOL lab

LAB

```
@fb_test_case_with_lab(
    ”Bring ethernet-1/1 up on Nokia SR Linux",
    remote_lab_fixture=simple_srl,
    params={...},
    assertions=[port_is_up]
)
```

Nokia SR Linux lab

1

```
@fb_test_case_with_lab(
   "Shut Ethernet0/1 on Cisco IOL",
   remote_lab_fixture=simple_iol,
   ...
)
```

2

pytest

Discovers the generated tests

3

Remote Lab

Coordinates lab lifecycle

4

netlab + Containerlab

Spins up the topology

5

Virtual Lab

Cisco IOL · real CLI

Runs anywhere
(e.g. locally in IDE, CI/CD)

Runs on remote VM

The guinea pig in action


```python
@fb_test_case(
    "Fails with no device",
    params=SetAdminStateParams(
        interface="Ethernet0/1",
        desired_state=AdminState.DOWN,
    ),
    succeeds=False,
)

@fb_test_case_with_lab(
    "Shut Ethernet0/1 on Cisco IOL",
    remote_lab_fixture=simple_iol,
    params=SetAdminStateParams(
        interface="Ethernet0/1",
        desired_state=AdminState.DOWN,
    ),
    assertions=[lambda r: r.data.state_after == AdminState.DOWN],
)

@fb_test_case_with_lab(
    "Bring ethernet-1/1 up on Nokia SR Linux",
    remote_lab_fixture=simple_srlinux,
    params=SetAdminStateParams(
        interface="ethernet-1/1",
        desired_state=AdminState.UP,
    ),
    assertions=[lambda r: r.data.state_after == AdminState.UP],
)
@register_function_block(..)

class SetAdminState(FunctionBlock):

    async def run(self, p, ctx):
        with AdminStateProxy.connect(ctx.device) as conn:
            before = conn.get_interface_admin_state(p.interface)
            conn.set_admin_state(p.interface, p.desired_state)
            after  = conn.get_interface_admin_state(p.interface)

        return FunctionBlockResult(
            success = after is p.desired_state,
            data    = SetAdminStateResult(p.interface, before, after)
        )
```

```
=== Testing: fails with no device ===
```

```
=== Testing: Shut Ethernet0/1 on Cisco IOL ===
```

```
=== Testing: Shut ethernet-1/1 on Nokia SR Linux ===
```

Wait a minute...?

```
with AdminStateProxy.connect(ctx.device) as conn:
    before = conn.get_interface_admin_state(p.interface)
    conn.set_admin_state(p.interface, p.desired_state)
    after  = conn.get_interface_admin_state(p.interface)
```

FUNCTION BLOCK

SetAdminState
(FunctionBlock)

PROXY

AdminStateProxy

inherits
ConnectionProxy,
AdminStateCapability

CISCO IOL

IOSCiscoScrapliPlugin via **Scrapli**
implements AdminStateCapability

```
def get_admin_state(self, interface):
    out = self.send(f"show interface {interface}")
    return DOWN if "administratively down" in out else UP
def set_admin_state(self, iface, state):
    self.send(f"interface {iface}")
    self.send("shutdown" if state==DOWN else "no shutdown")
```

NOKIA SR LINUX

SRLinuxNetmikoPlugin via **Netmiko**
implements AdminStateCapability

```
def get_admin_state(self, interface):
    out = self.send(f"info from running /interface {interface} admin-state")
    return DOWN if "disable" in out else UP
def set_admin_state(self, interface, state):
    cmd = "enable" if state==UP else "disable"
    self.send(f"set /interface {interface} admin-state {cmd}")
    self.send("commit now")
```

+ ANY OTHER VENDOR

AnyVendorPlugin via **AnyLib**
implements AdminStateCapability

```
def get_admin_state(self, interface): ...
def set_admin_state(self, interface, state): ...
```

CAPABILITY

```
class AdminStateCapability(CapabilityInterface):
    @abstractmethod
    def get_admin_state(self, interface: str) -> AdminState: ...
    @abstractmethod
    def set_admin_state(self, interface: str, state: AdminState) -> None: ...
```

Now the fun starts.

- CI/CD for the network
- Debug like any other code
- New vendor, same function blocks
- AI-written, AI-tested automation
- Refactor without fear *(or at least less…)*

Better sleep. Hopefully.

Thank you for listening!

Remote Lab - Developer Preview:
https://github.com/zebbra/neops-remote-lab

Integrate **Netlab** directly into your tests via **pytest fixtures**
Maintain one lab for multiple engineers and the **CI/CD pipeline**.

"That last percent for fully automated realistic e2e testing"


neops
network automation
orchestration


Automation
you can
trust.
neops
Network automation orchestration