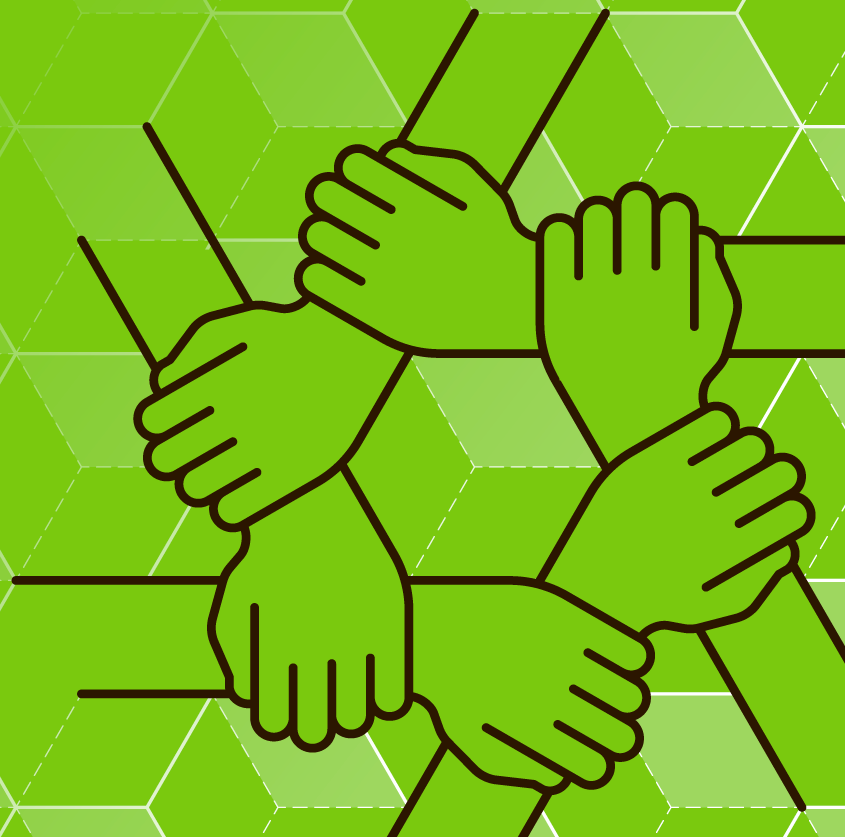
# Test tooling for 100% of SAI use cases



EMPOWERING OPEN.

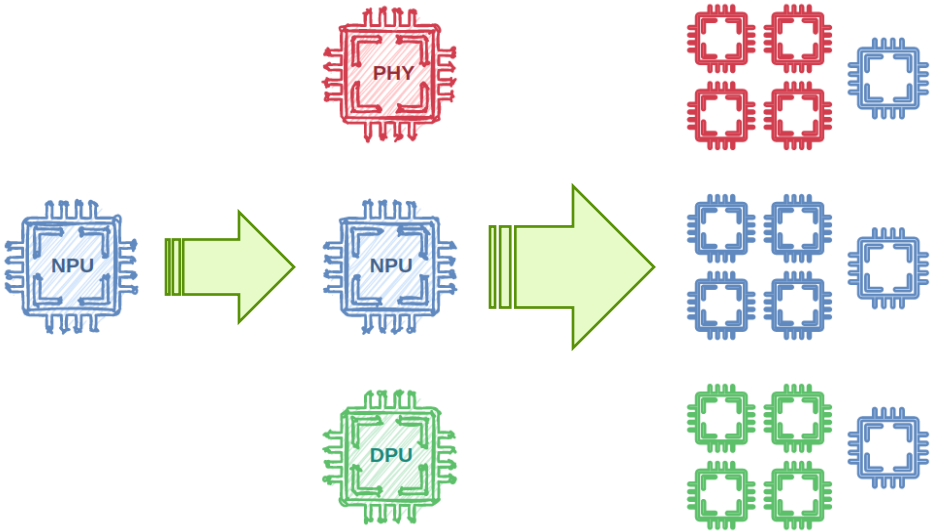
OCP REGIONAL SUMMIT

APRIL 19-20, 2023
PRAGUE, CZ

# Test tooling for 100% of SAI use cases

Andriy Kokhan, Solutions Architect, PLVision
Vlad Laslău, Senior Technical Product Manager, Keysight Technologies

# The application area of SAI




- SONiC device with external PHYs
- CLOS topology
- Multi-ASIC (CLOS-in-a-box, VoQ)
- Disaggregated Chassis
- DASH
- Smart Switch

Dec 2014
3.2T / 100G

about 9 years

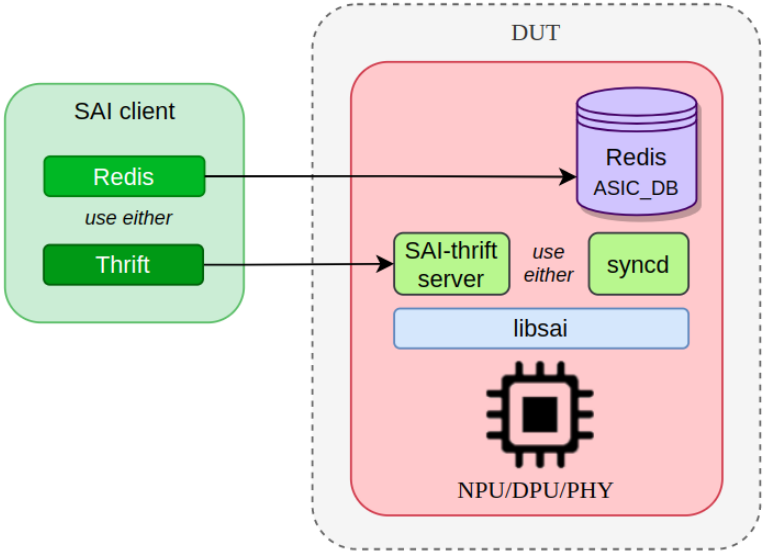
Feb 2023
51.2T / 800G

# Key SAI testing pillars

- **Decoupling of SAI RPC implementation from the API** – written once, the test case can be executed repeatedly using different RPC mechanisms:
  - Thrift – generic use case independent scenario
  - Redis – SONiC scenario
- **Traffic generator agnostic interface** – possibility to use both SW and HW traffic generators to fulfill the demand for the line rate scenarios testing
- **Testbed's flexible description** – a simple and clear definition of all aspects of a certain testbed (the type of SAI RPC mechanism, the type of the traffic generator, connections, etc.)

# SAI configuration modes

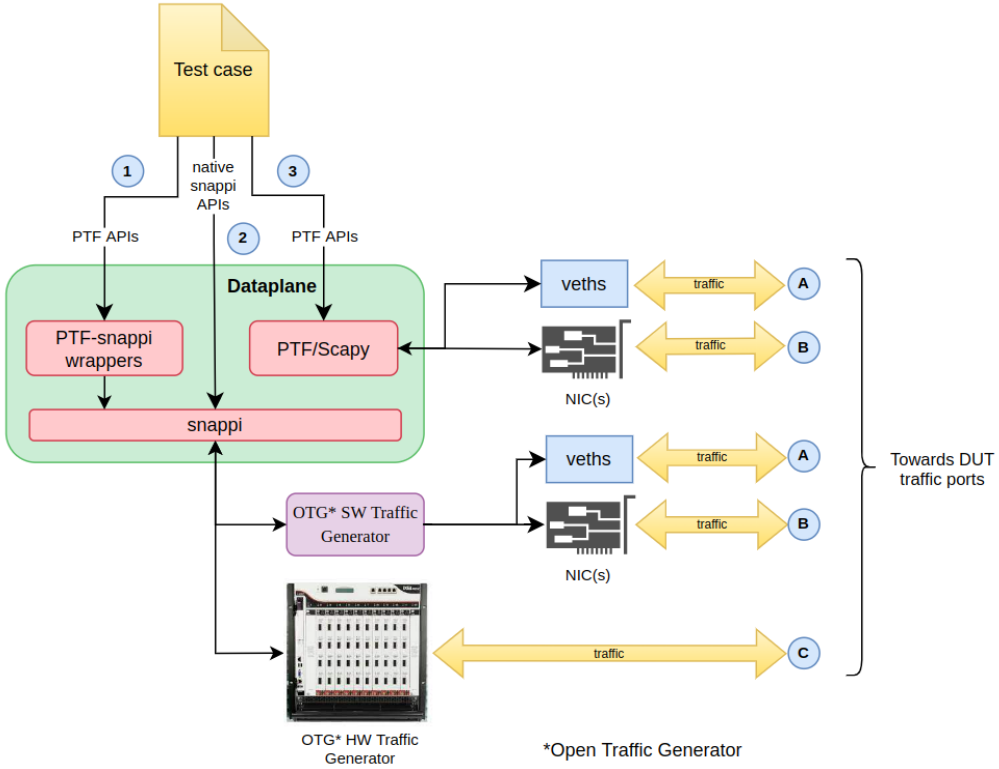
- Supports both saithrift and sairedis APIs
- Runs over either API without change
- The test configuration file selects which API to use



# Traffic generator variations

(1) PTF-snappi wrappers, OTG Tgen; packet at a time.
(2) Native OTG/snappi, flow-based packets.
(3) PTF utils, Scapy Traffic generator; packet at a time.

(A) Virtual testing at CPU speeds.
(B) Physical DUT testing at CPU speeds.
(C) Physical testing, high-speed and scale.



# Open Traffic Generator (OTG) API – Overview

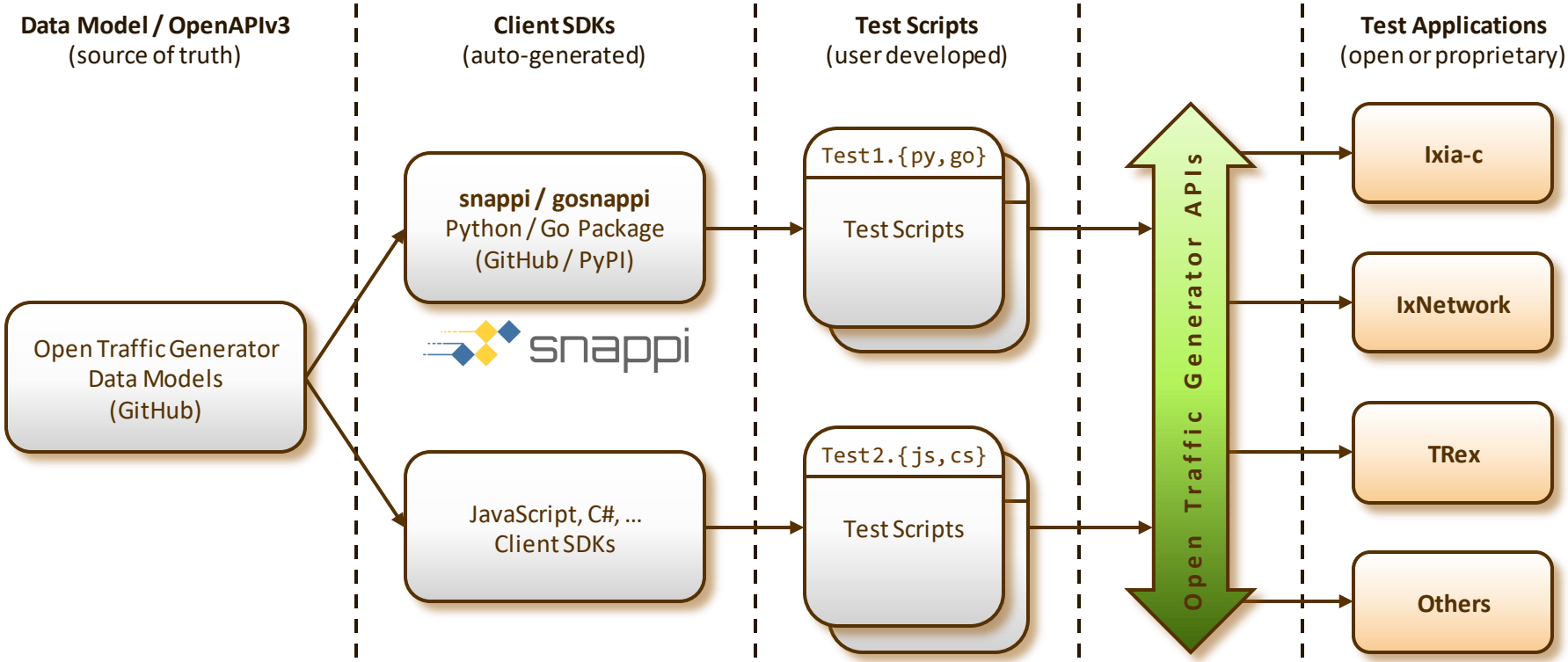
- Open model based declarative API. Designed for network testing. Vendor agnostic ecosystem.
- Open API v3 definition with auto-generated client SDKs in multiple languages.
- Separation between Data Model & API. Well documented Data Model. Flat API.
- Write once and run everywhere. Speeds from slow simulators up to line rate.


- Client-side SDK for Open Traffic Generator API.
- Pythonic and available on PyPI. Can be installed via **pip install snappi**.
- OO, default values, client-side validations, factory methods, single import, …
- Serialize / deserialize whole configuration or objects, IntelliSense, …

```python
# Test TCP ACL on ASIC
# Increment TCP Source Port

import snappi

api  = snappi.api(location = "https://100.1.1.1")
cfg  = api.config()

flow = cfg.flows.flow(name = 'Traffic-Flow')[-1]
flow.size.fixed      = 1518
flow.rate.percentage = 10
flow.metrics.enable  = True

eth, ip, tcp  = flow.packet.ethernet().ipv4().tcp()

eth.src.value = "00:11:22:33:44:55"
eth.dst.value = "00:AA:BB:CC:DD:EE"

ip.src.value  = "1.1.1.1"
ip.dst.value  = "1.1.1.2"

tcp.src.increment.start = 5000
tcp.src.increment.step  = 2
tcp.src.increment.count = 10
```

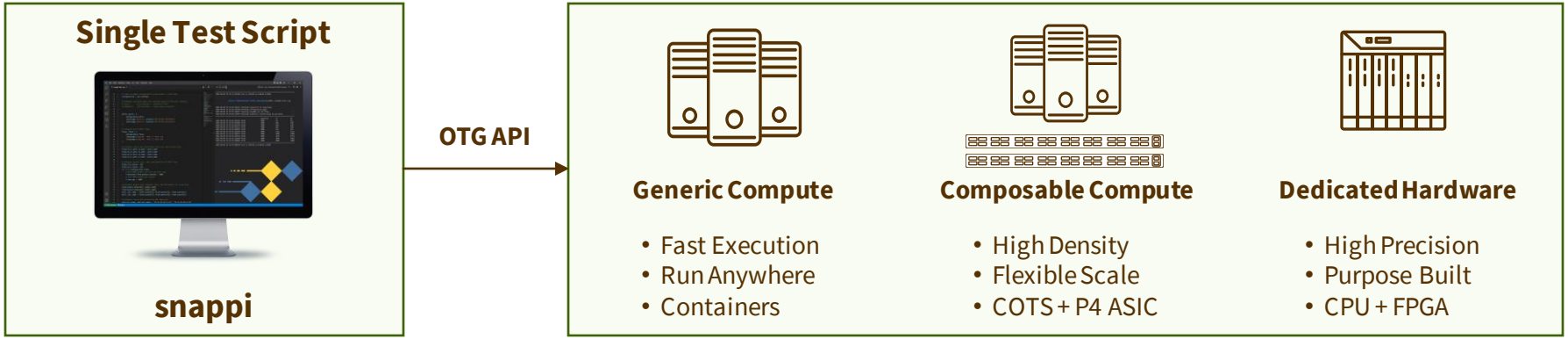
**https://otg.dev**

# snappi : From API to Test Script to Test Execution



# Open Traffic Generator (OTG) API – Status

- **Traffic Generation** : Layer 2 / 3 / 4 Traffic Generation & Statistics Analysis
- **Device Emulation** : ETH, LACP, LAG, LLDP, PFC, IPv4, IPv6, BGP, ISIS, RSVP
- **Ecosystem Presence** : SAI Challenger, SONiC, DASH, OpenConfig Feature Profiles
- **Platforms** : Software (KNE + Container Lab) & Hardware (Composable + Dedicated)
- **Availability** : Now (version 0.x available) & 2023 Q3 (version 1.0 planned)



# Testbed description

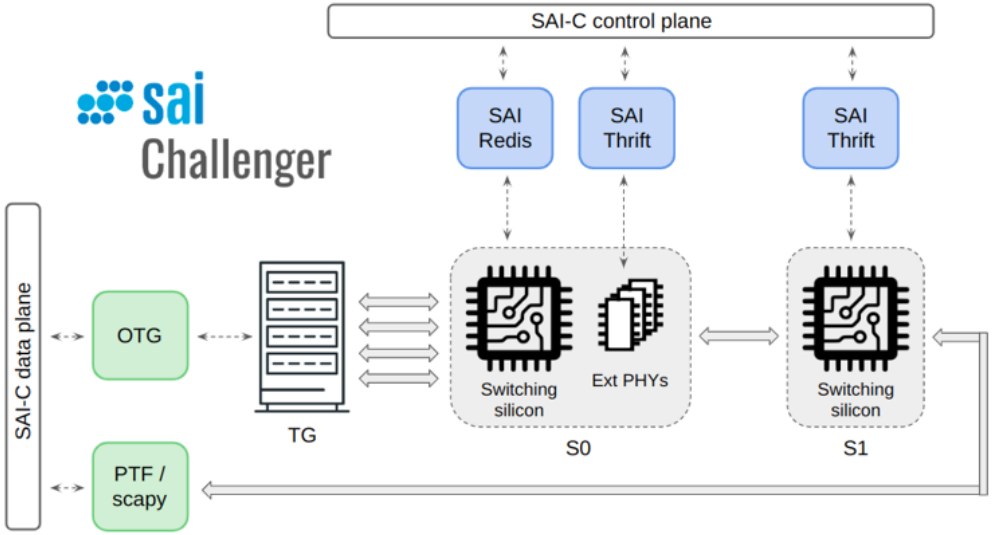
```
"npu": [
  {
    "alias": "npu1",
    "asic": "trident3",
    "target": "AS7726-32X",
    "sku": "32x100g",
    "client": {
      "type": "redis",
      "config": {
          "ip": "192.168.1.10",
          "port": "6379"
      }
    }
  },
  {
     // NPU/DUT #2
  }
]
```

```
"phy": [
  {
    "alias": "phy1",
    "asic": "trident2",
    "target": "saivs",
    "sku": null,
    "client": {
      "type": "thrift",
      "config": {
          "ip": "192.168.1.10",
          "port": "9092"
      }
    }
  },
  {
     // PHY #2
  }
]
```

```
"dataplane": [
  {
    "alias": "ptf",
    "type": "ptf",
    "mode": "eth",
    "port_groups": [
      {"alias": 0, "name": "veth1"},
      {"alias": 1, "name": "veth2"}
    ]
  },
  {
    "alias": "snappi",
    "type": "snappi",
    "mode": "ixia_c",
    "controller": "https://127.0.0.1:443",
    "port_groups": [
      {"alias": 0, "name": "veth3", "speed": "10G"},
      {"alias": 1, "name": "veth4", "speed": "10G"}
    ]
  }
]
```

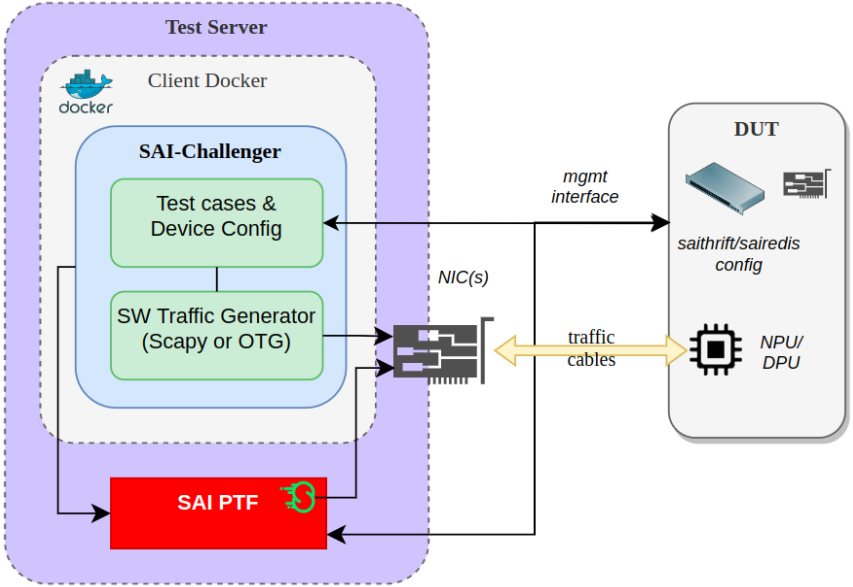
# Multi-DUT multi-ASIC topology

- Data plane diversity
- Control plane diversity
- Test cases definition flexibility
- Multi-ASIC (NPU, DPU, PHY)
- Multi-DUT



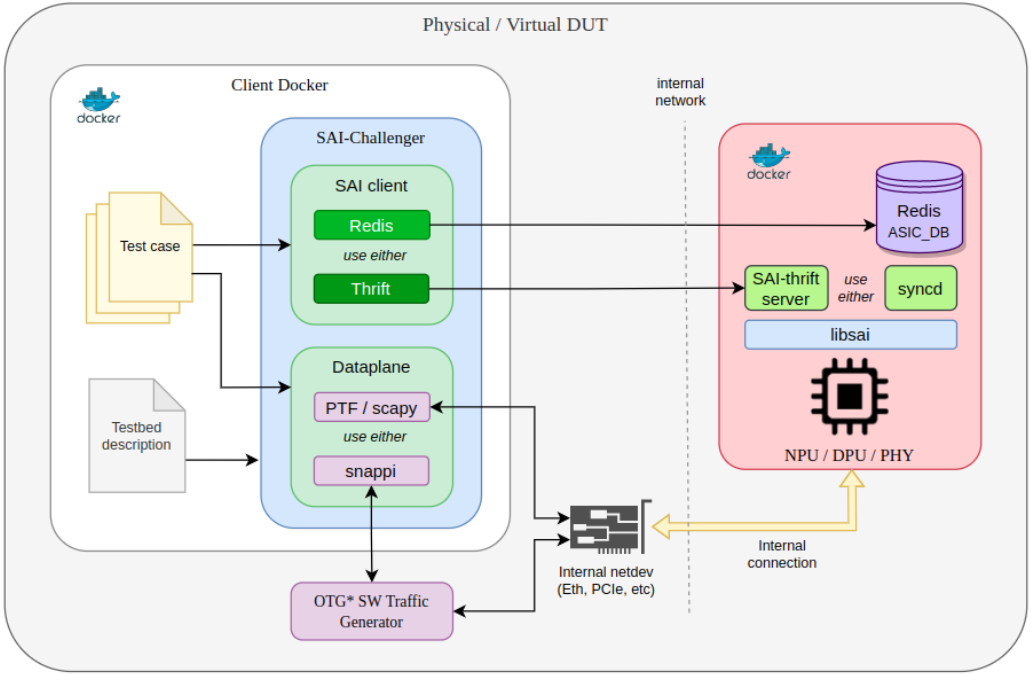
# Use case 1: Executing PTF tests

- Testbed description in JSON
- SAI-C translates JSON into PTF config
- SAI-C executes SAI PTF TCs without changes



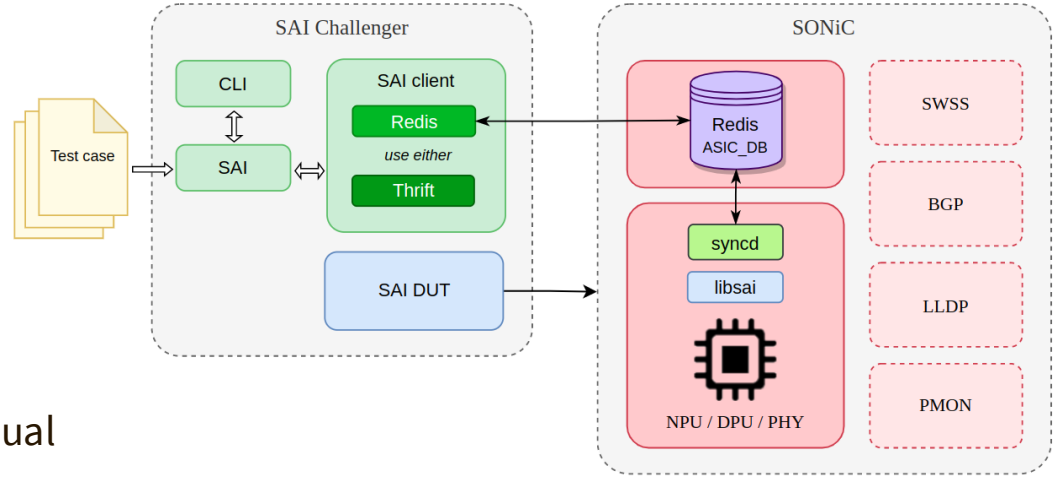
# Use case 2: Self-contained DUT

- SAI-Challenger runs on the DUT itself
- The DUT is a separate physical or virtual device, whether a network switch, DPU or SW dataplane running on a server.
- The DUT is controlled via saithrift or sairedis depending on DUT capabilities, using internal management network.



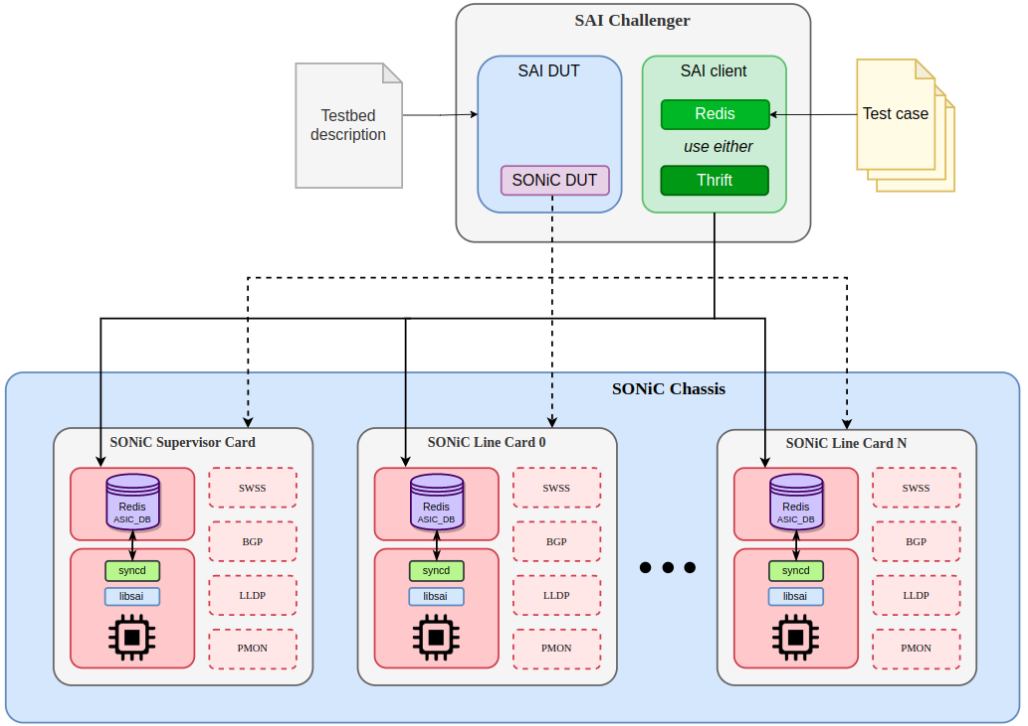
# Use case 3: SONiC testbed as a DUT

- Start SONiC-based device
- Build and start SAI-C Client
- SAI testbed self-provisioning
  - ssh to SONiC device
  - stop SONiC services
  - start Redis and SyncD dockers
- Execute SAI-C test-cases or run manual SAI commands through CLI



# Use case 4: SONiC chassis as a multi-DUT

- Supervisor card and each line cards are designed as the separate SONiC devices
- From SAI-C perspective, SONiC chassis is a regular multi-DUT topology



# Call to Action

- Try it:
  https://github.com/opencomputeproject/SAI-Challenger/blob/main/docs/standalone_mode.md
- Start using:
  https://github.com/opencomputeproject/SAI-Challenger/blob/main/docs/porting_guide.md
- Scale to the line-rate:
  https://otg.dev
  https://github.com/open-traffic-generator
- Adapt and extend:
  https://github.com/opencomputeproject/SAI-Challenger/tree/main/usecases

# Thank you!

## EMPOWERING OPEN.

OCP REGIONAL SUMMIT

APRIL 19-20, 2023
PRAGUE, CZ