# Rethink testbeds with blueprints

on behalf of the SLICES community

11/30/2023

# The blueprint concept in a nutshell

- Help collaboration between engineers and non-engineers to collaborate on developing (business) applications.
- Define a common terminology that doesn't require in-depth, technical knowledge
  - to produce consistent vision on the application with a focus on the future objective.
- Pick up the right resources, be iterative, review and validate, and keep it as a baseline.

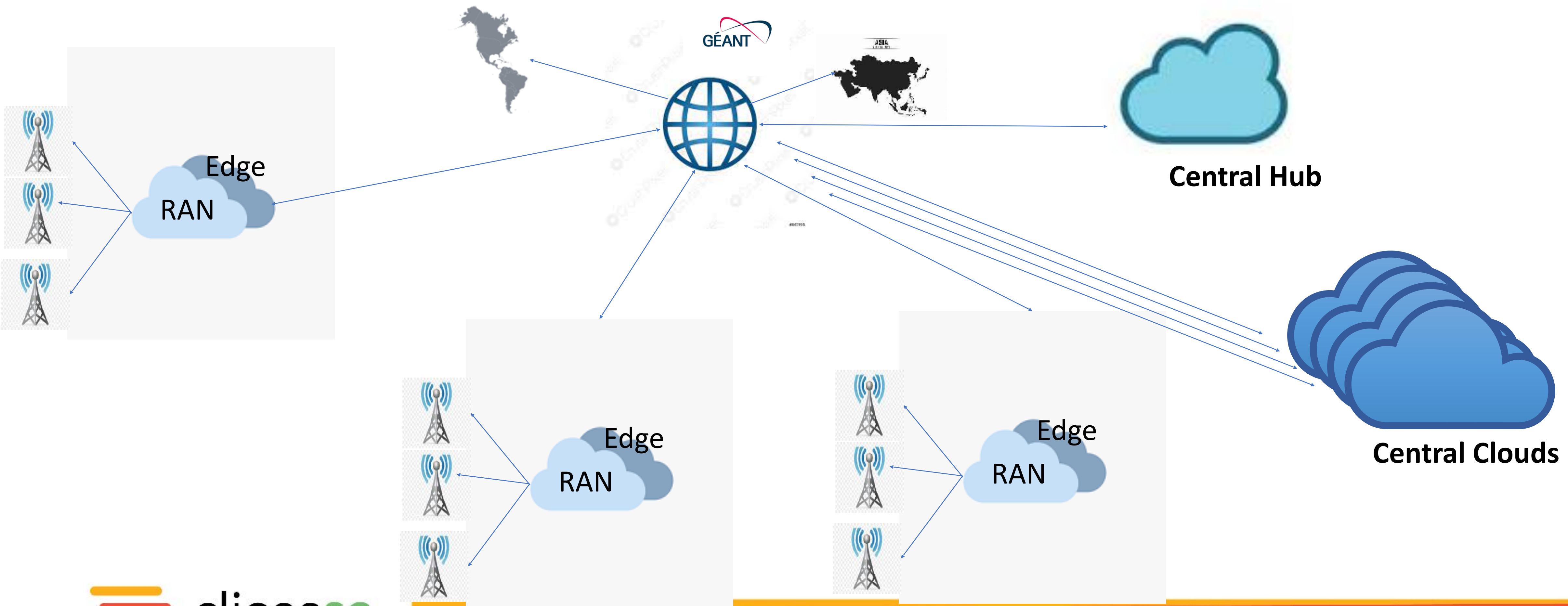
*Adapted from [0a, 0b]*

slicessc

[0a] https://dev.to/jayjayjpg/what-is-a-software-blueprint-5388
[0b] https://www.qrpinternational.fr/blog/faq/blueprint-quest-ce-que-cest/

# SLICES research instrument

# 5G blueprint – High level view




slicessc

# Open, large scale, reproducible

- Reuse and contribute to open-source initiatives
  - Common software/hardware base
- Complex deployments:
  - Multi-region
  - Multi-tenancy
  - Multi-management
- Full documentation
- Fine-grain automatic control

slicessc

# What for?

slicessc

# Type 1 experiments: vertical service integration and testing

- post-5G system is a blackbox and users embed software
  - a) in the network infrastructure user-plane, either in the a1) cloud center or a2) at the RAN/Edge (low-latency services)
  - b) terminals (e.g. drones, robots, fixed stations).
- Users can add HW (“bring your own device”) to the terminal end for special experiments (e.g. multimedia or connected robotics). Potentially also in the network infrastructure.

# Type 2 experiments: Software Defined Networking

- The user has access to well-defined interfaces to the network functions (e.g., xAPP) and writes applications which stimulate the interface to alter the network behaviour or collect KPI.
- This typically involves writing software in the framework of a controller.
- Eventually it could even be in a DPU or programmable switches (P4) for low-level real-time behaviour.

slicessc

# Type 3 experiments: radio/network development software

- Users without specific radio/processing HW can use a site for development and testing purposes (e.g., ssh access) with the objective of pushing software improvements to companion OSS projects.
- A user develops improved network function (e.g., PHY, MAC) implementations using one of the CI/CD frameworks of companion projects (e.g. OAI,ORAN O-SC) used by SLICES-RI sites for CD. These go through the normal CI procedures (testing) and get deployed on SLICES-RI for larger-scale testing.

# Type 4 experiments: low-level access to radio resources

- Will be to experiment with candidate 6G technologies. This will typically involve insertion of hardware elements into sites such as
  - novel RF devices such as antennas, RIS, optical wireless devices, THz radios, etc.
  - hardware accelerators for key functions like channel decoders, cryptographic functions, user-plane packet-processing.
  - real-time edge devices (TSN, real-time multimedia, industrial IoT, etc.).
- Once inserted (or provided by sites) users can reserve time to develop and test scenarios using these devices.

# Type 5 experiments: joint use of post-5G infrastructure and HPC resources

- Example 1: Real-time Digital twin of radio network
  - GPU farms can be used as real-time 3D radio emulators. When interconnected with radio and core network infrastructure can make a digital twin of a deployed network (e.g., Converge project)
  - This requires tight interconnection between radio processing infrastructure and the GPU farm but can be used to perform experiments not possible on the real network (large number of terminals). Can run on current cloud infrastructure (e.g. AWS).
  - Novel aspect, joint radio and digital twin. This requires proximity of HPC and real radio infrastructure.
- Example 2: Code analysis and bug fixing
  - Protocol implementations are bug-ridden. In the CI/CD Type 3 experiment, developers of OSS networking software can make use of SLICES-RI GPU farms for code analysis and bug fixing.
  - Today in projects like OAI, CI makes use of "simple" tools like cppcheck to analyze community contributions. Use of AI/ML tools will take this to another level.

# Why 5G?

# 5G key principles [1]

- Separate the User Plane functions from the Control Plane functions
- Modularize the function design
- Define the set of interactions between network functions as services
- Enable direct interactions between network functions
- Minimize dependencies between the Access Network and the Core Network
- Support "stateless" NFs (i.e., compute decoupled from storage)
- Support capability exposure

[1] ETSI TS 123 501

# General core architecture

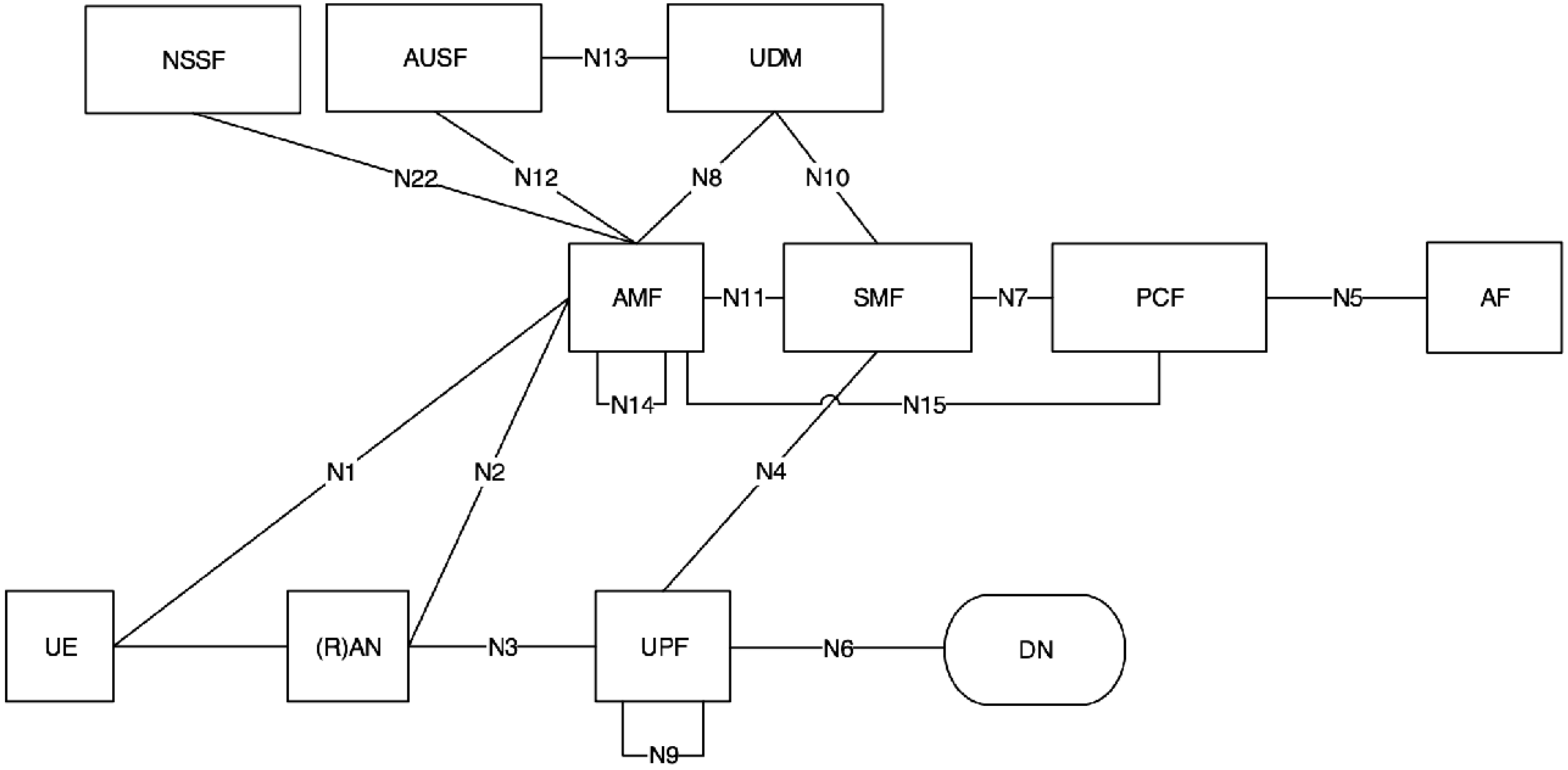



**Figure 4.2.3-2: Non-Roaming 5G System Architecture in reference point representation [1]**

AF: Application Function
AMF: Access and Mobility Management Function
AUSF: Authentication Server Function
DN: Data Network (DN)
NSSF: Network Slice Selection Function
PCF: Policy Control Function
(R)AN: (Radio) Access Network
SMF: Session Management Function
UDM: Unified Data Management
UE: User Equipment
UPF: User Plane Function

[1] ETSI TS 123 501

# Split-RAN

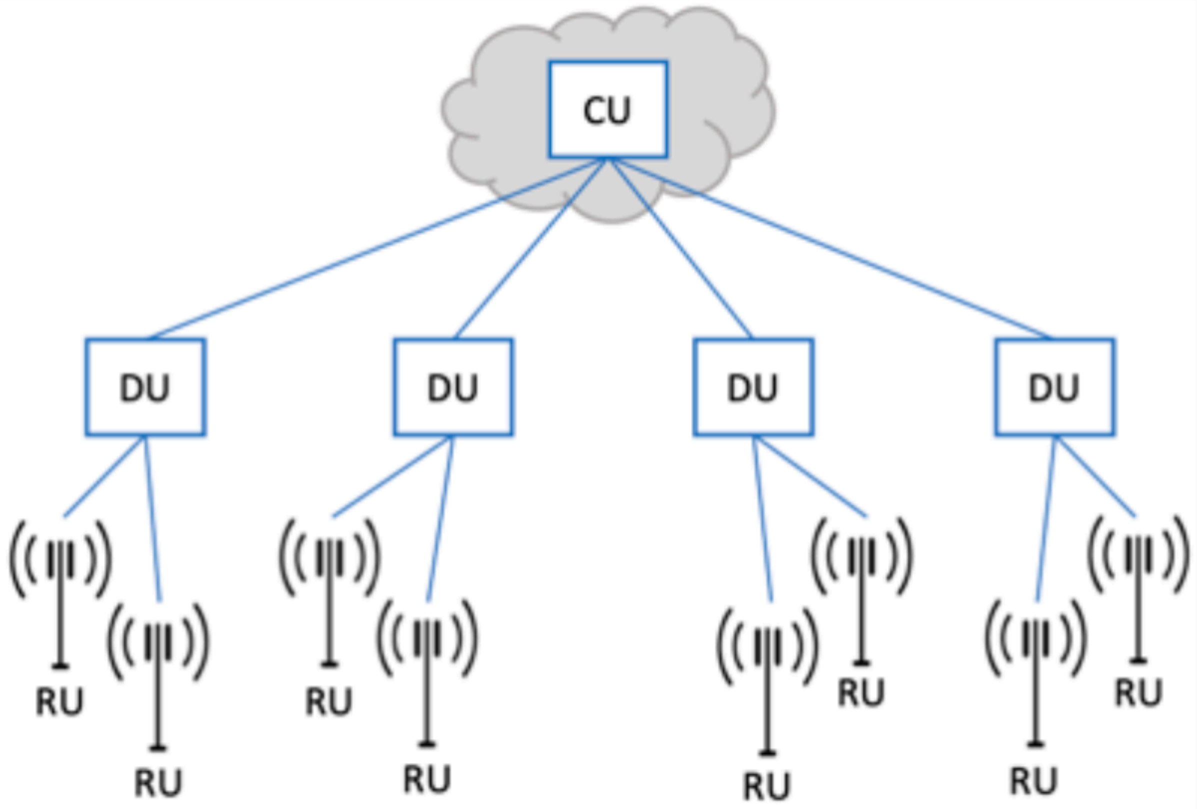



Figure 26. *Split RAN hierarchy, with one CU serving multiple DUs, each of which serves multiple RUs.* [2]

slicessc

[2]Private 5G: A Systems Approach v1.1-dev

# Split-RAN

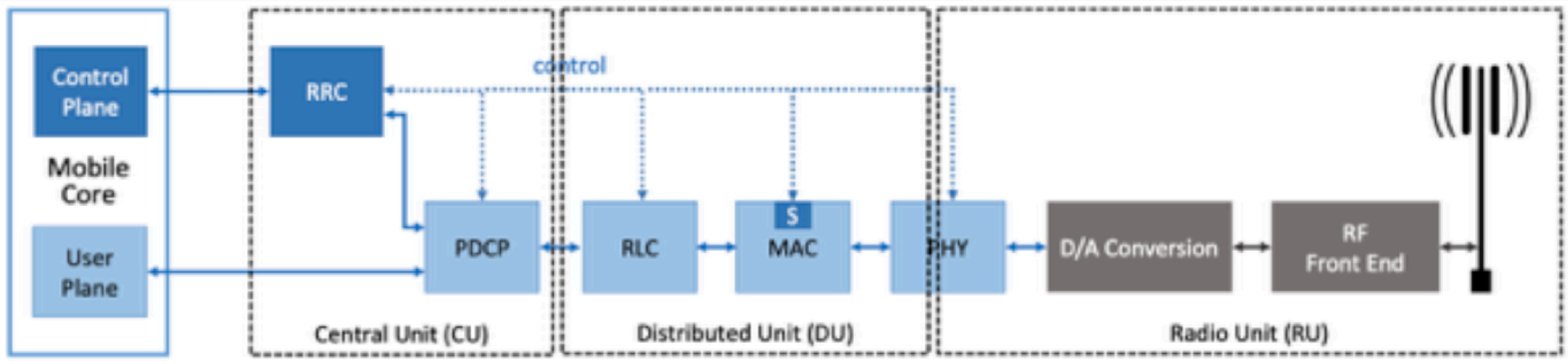


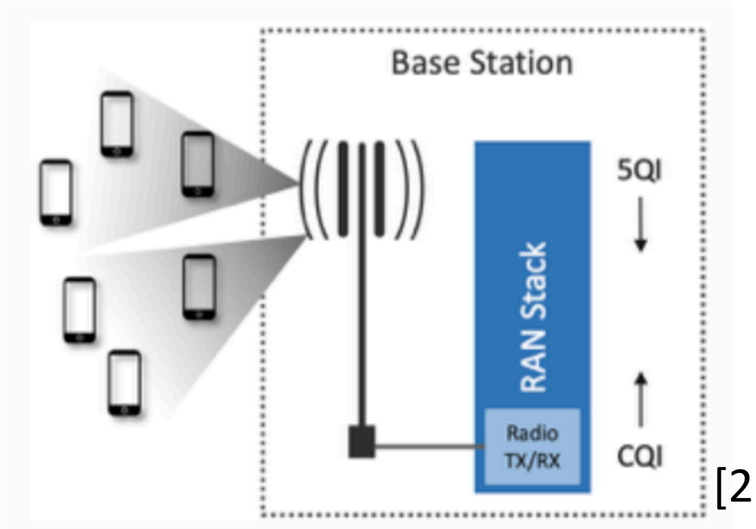

Figure 25. *Split RAN processing pipeline distributed across a Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU).* [2]

slicessc

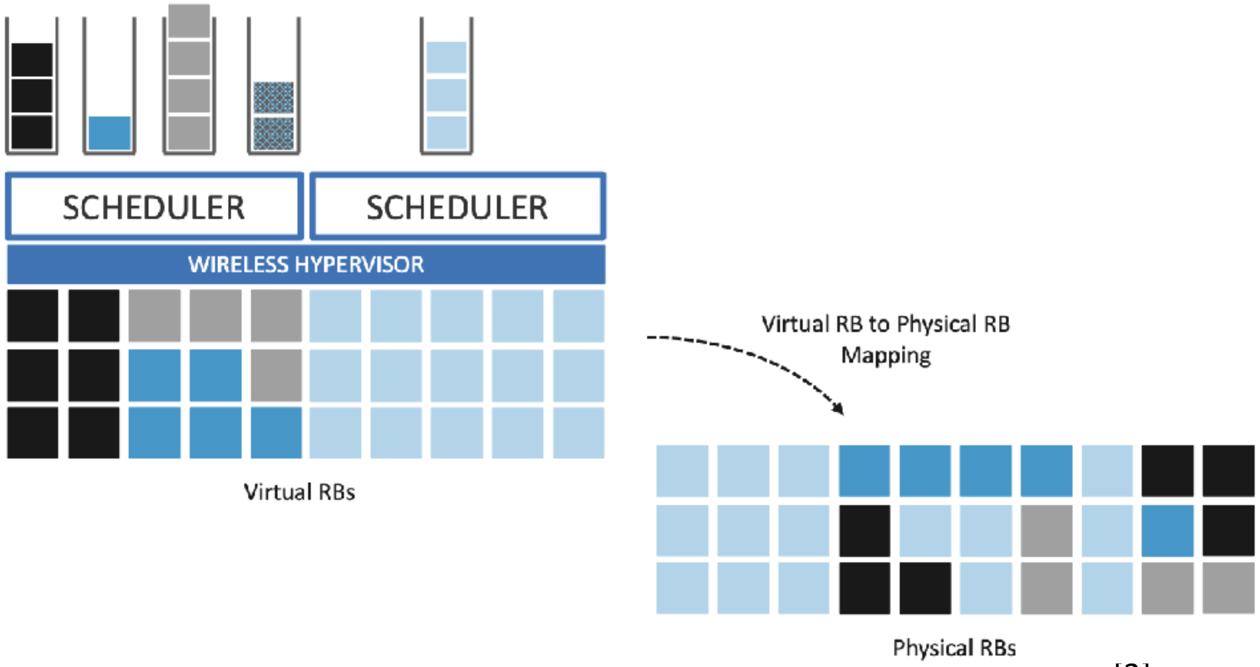
[2]Private 5G: A Systems Approach v1.1-dev

# Radio optimization

- High utilisation of spectrum, sub-millisecond control loop




[2]




[2]

5QI: 5G QoS Identifier
CQI: Channel Quality Indicator

slicessc

[2]Private 5G: A Systems Approach v1.1-dev

# HW-based UPF implementations

- Intel Tofino silicon is discontinued
- DPDK, DDP, QAT... is not enough (~10Gbps, ~1ms)
- DPU/IPU-based solutions (e.g., Marvel OCTEON, Nvidia Bluefield 3) will fill this need
  - P4 may still be relevant but more conventional solutions (C/C++ control SW on ARM with HW accelerators)
- Now studying the use of such processors for UPF and/or CU-UP

slicessc

# HW-based UPF implementations

- Intel Tofino silicon is discontinued
- DPDK, DDP, QAT... is not enough (~10Gbps, ~1ms)

**How to be cloud-nativize?**

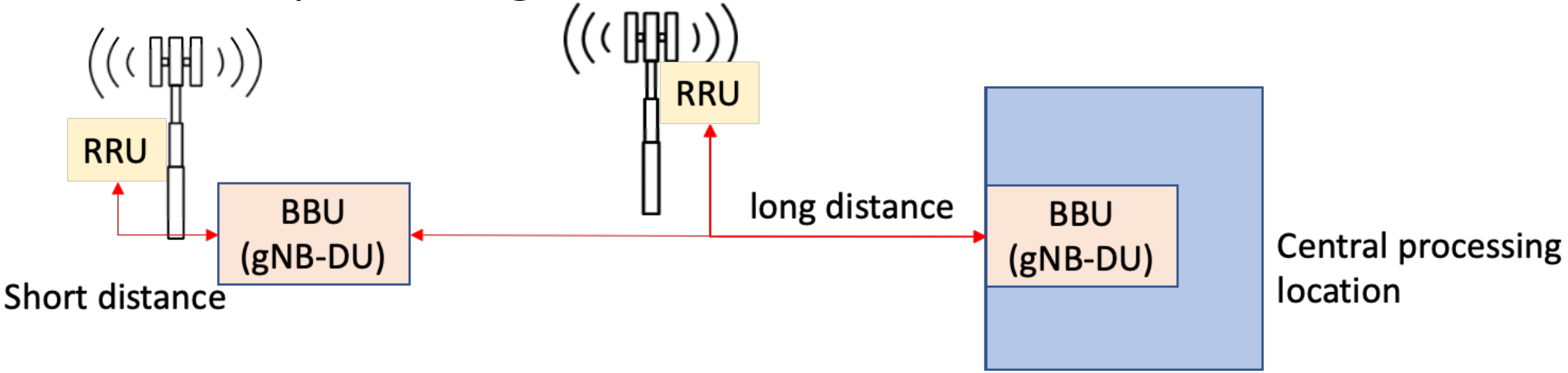
control SW on ARM with HW accelerators)

- Now studying the use of such processors for UPF and/or CU-UP

slicessc

# Fronthaul

- Transport of radio signals over a network:
  - to allow for statistical multiplexing of multiple radio sites sharing common fiber links
  - to share processing between radio-units and baseband units

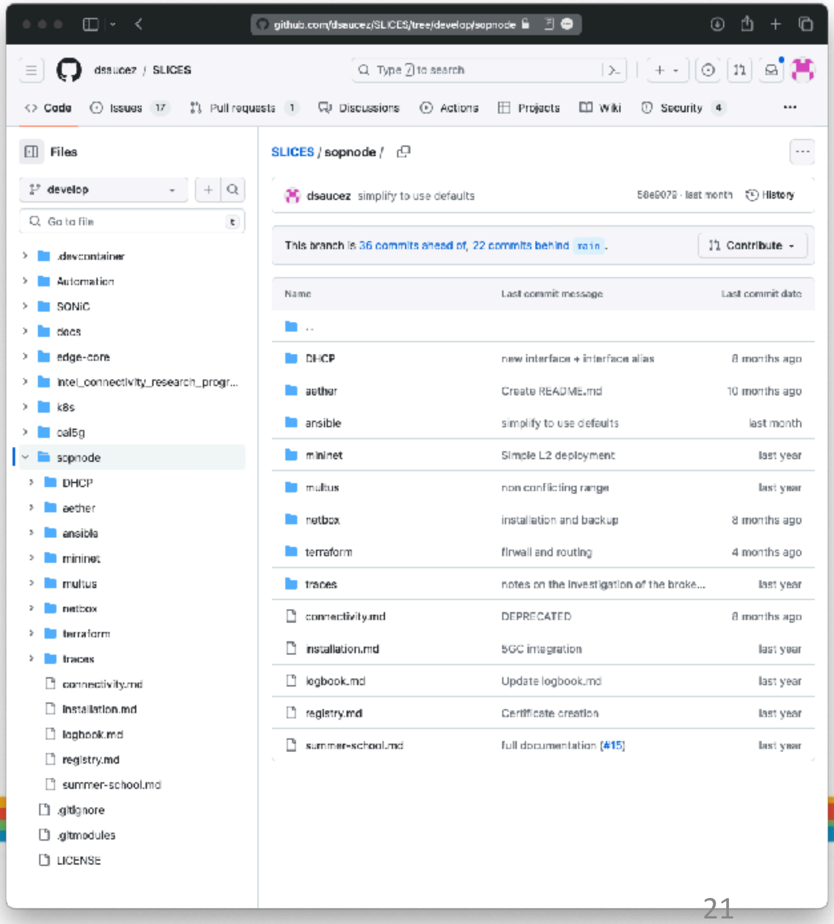


# Fronthaul implementation

- O-RAN Open FHI in OpenAirInterface
- Library pptimized for Intel-based x86-64 (e.g., AVX512)
  - How to move to AMD and ARM?
- Complexity in network infrastructure
  - Raw Ethernet
  - Usually 3 VLAN (VMDq not enough, need sr-iov): management, control/user plane
  - PTP distribution and PTP HW time stamping on NICs

# Reference implementation (1/3)

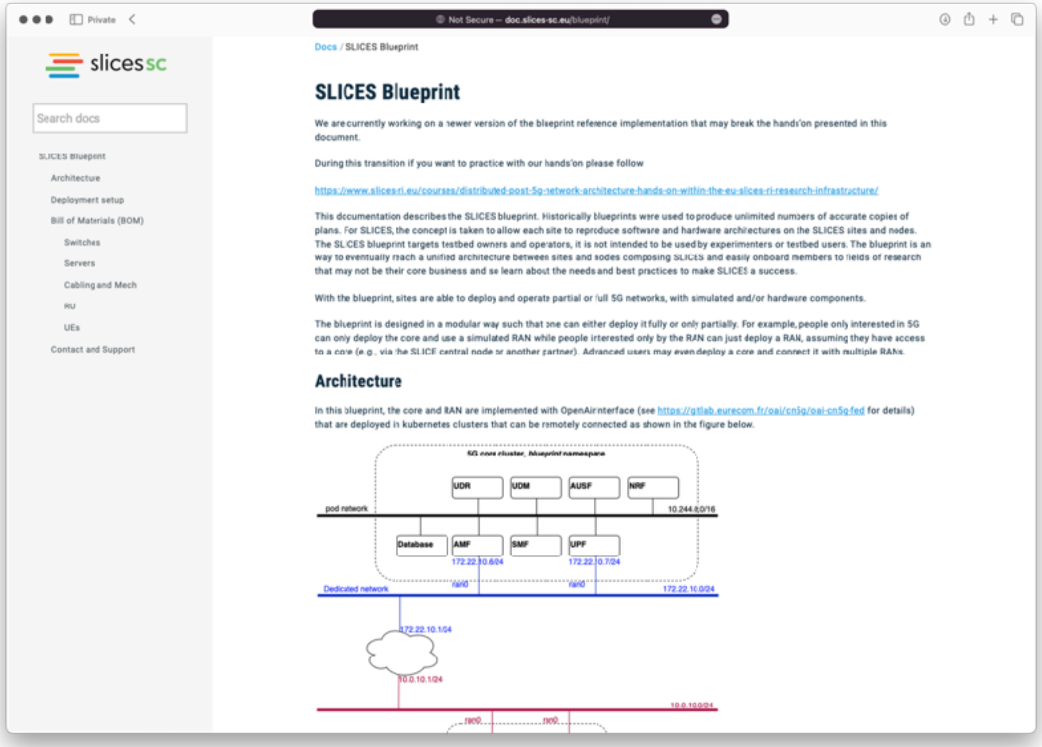
- Terraform (support of GCP) + Ansible for deployments
- kubernets + docker to manage resources
- OpenAirInterface
  - 5G core implementation
  - RAN implementation
- Software and/or hardware
  - USRP/AW2S
  - Tofino/Tofino2 with SD-Fabric

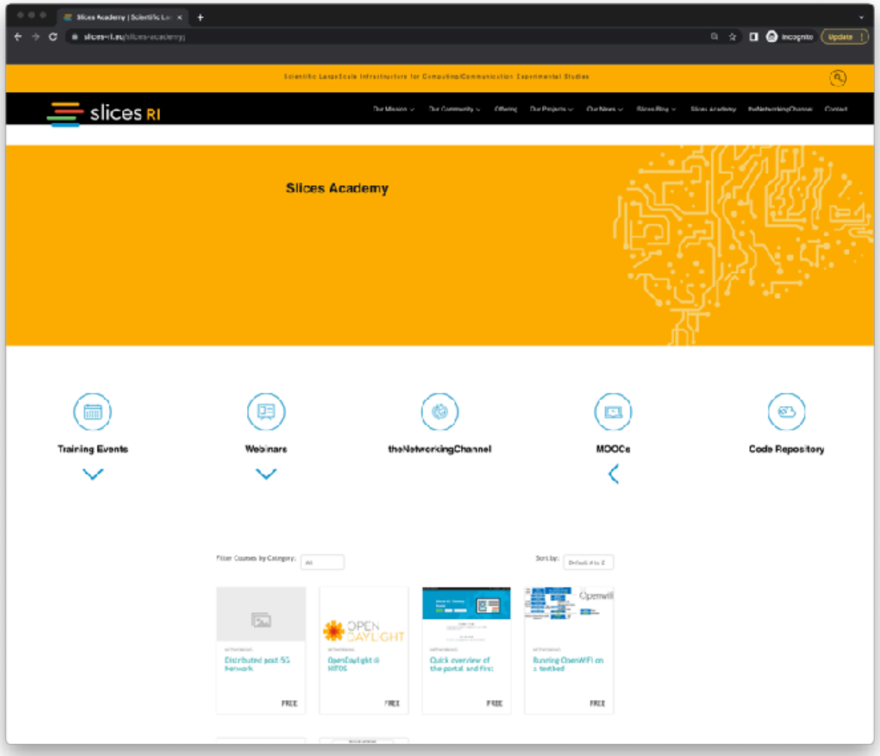



slicessc

# Reference implementation (2/3)

- Complete documentation and SLICES academy integration



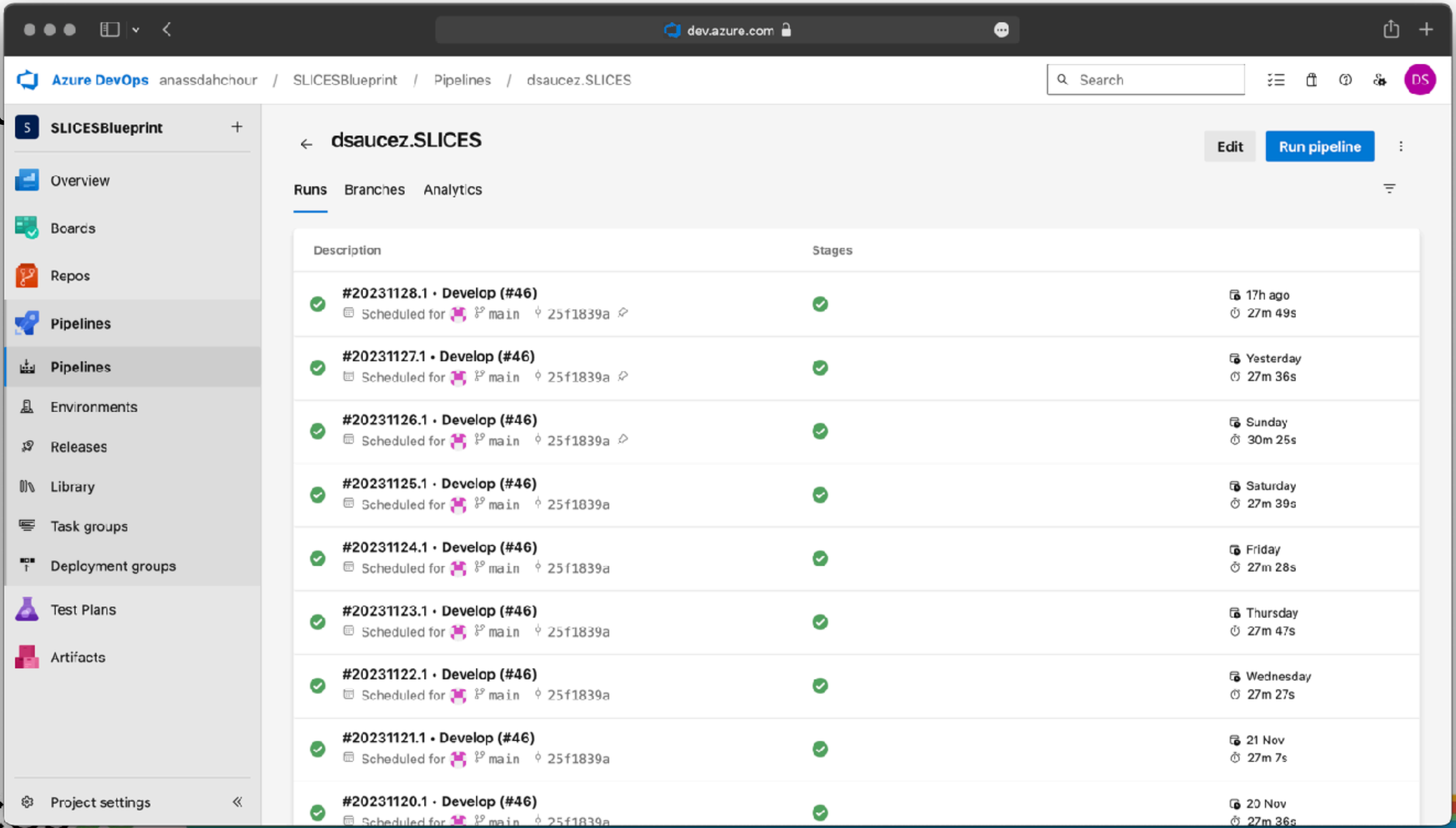



# Reference implementation (3/3)

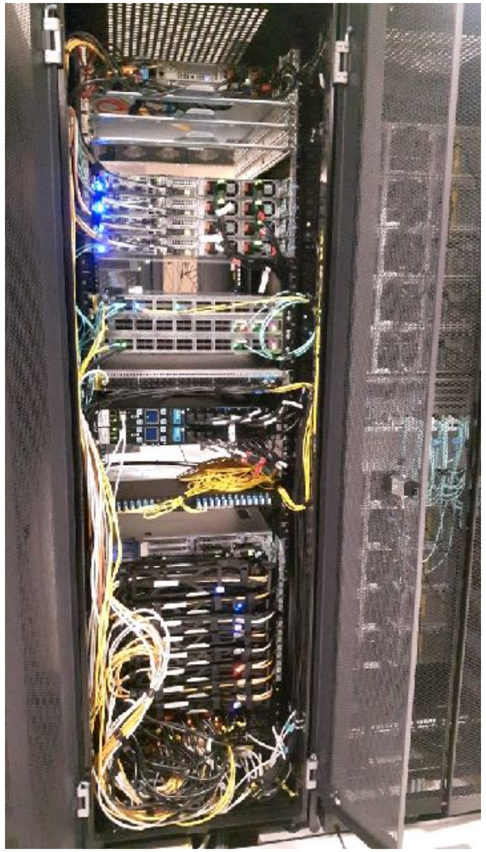
- Continuous testing

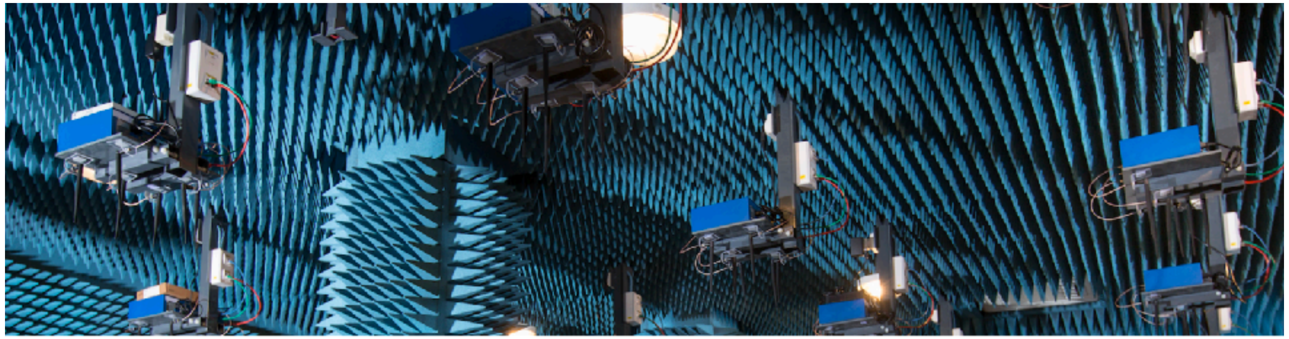
# Reference implementation (3/3)

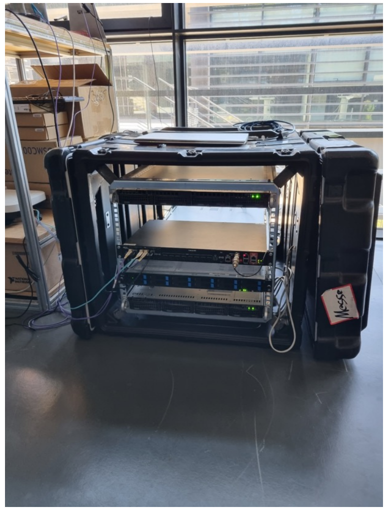
- Continu



# Lessons learned







**core**

Fronthaul

# Diversity of deployments (1/2)

- On-prem
  - Core + HW RAN @ Eurecom
  - Core + HW RAN @ Inria Sophia
  - Core/RAN split
    - Local 200Gbps fiber: Core @ Inria Sophia / HW RAN @ Eurecom
    - International dedicated VLAN 1Gbps/10ms: Core @ imec / RAN @ Inria Sophia
    - Commercial Internet:
      - Core @Sorbonne Université / RAN @ UTH
      - Core @Sorbonne Université / RAN @ Inria Sophia
  - Split 7.2 Core + RAN @ Inria Sophia
- Public cloud aided
  - Core by Inria @ GCP / HW RAN @ Eurecom
  - Core by Inria @ GCP / HW RAN @ UTH
  - Core @ CNR / RAN by Inria @ GCP
- In testbeds
  - Summer school in TUM testbed, Virtual Wall

# Diversity of deployments (2/2)

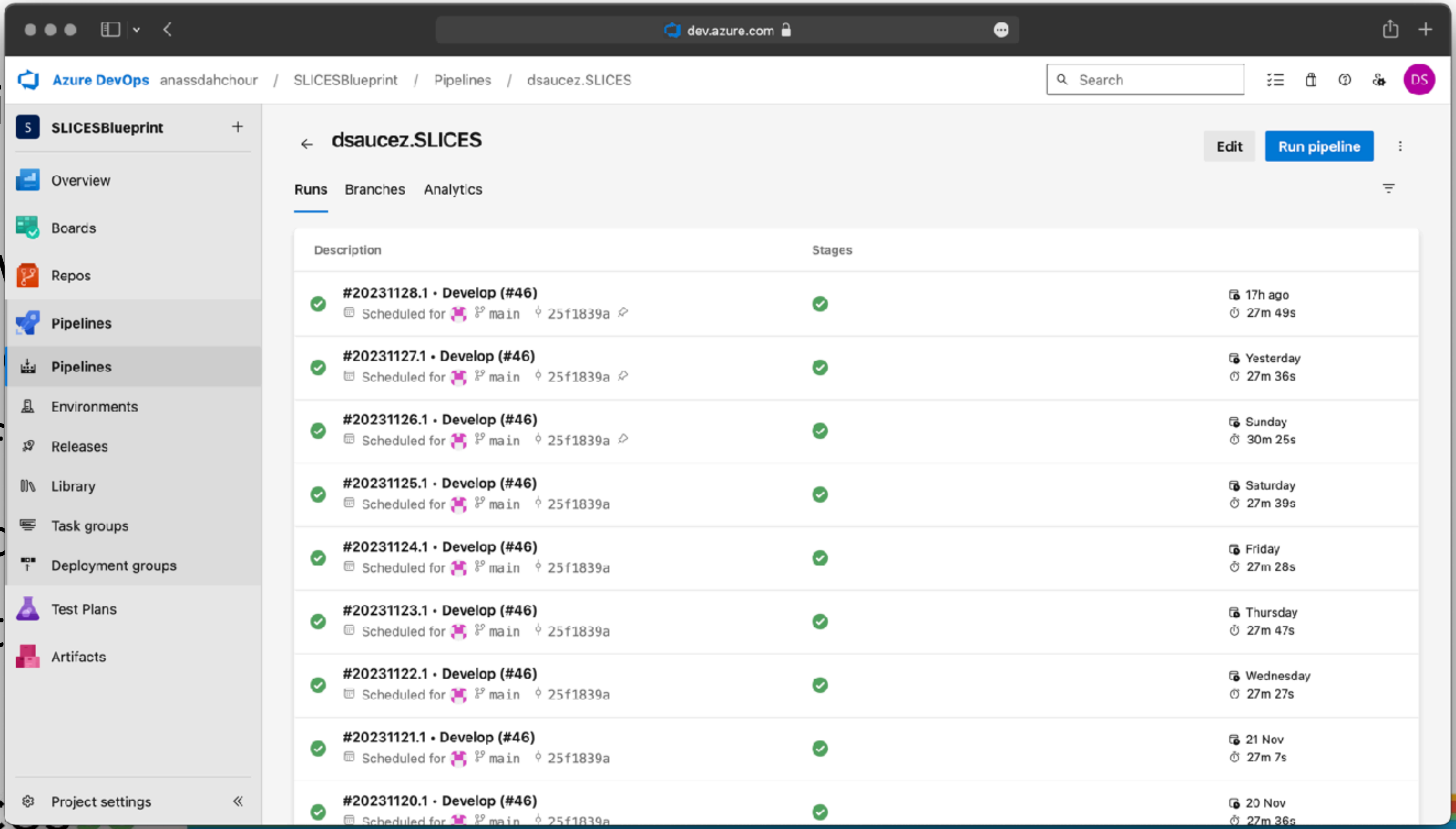
- Network interconnect
  - Commercial Internet accessible
  - Dedicated links
  - VPN service
- Identification / Authentication
  - Users based on the light federation from SLICES (OIDC) is ok
  - PKIs... how to unify certificates
- People test first in virtual environments (must support VM)

# Doomed by dependencies

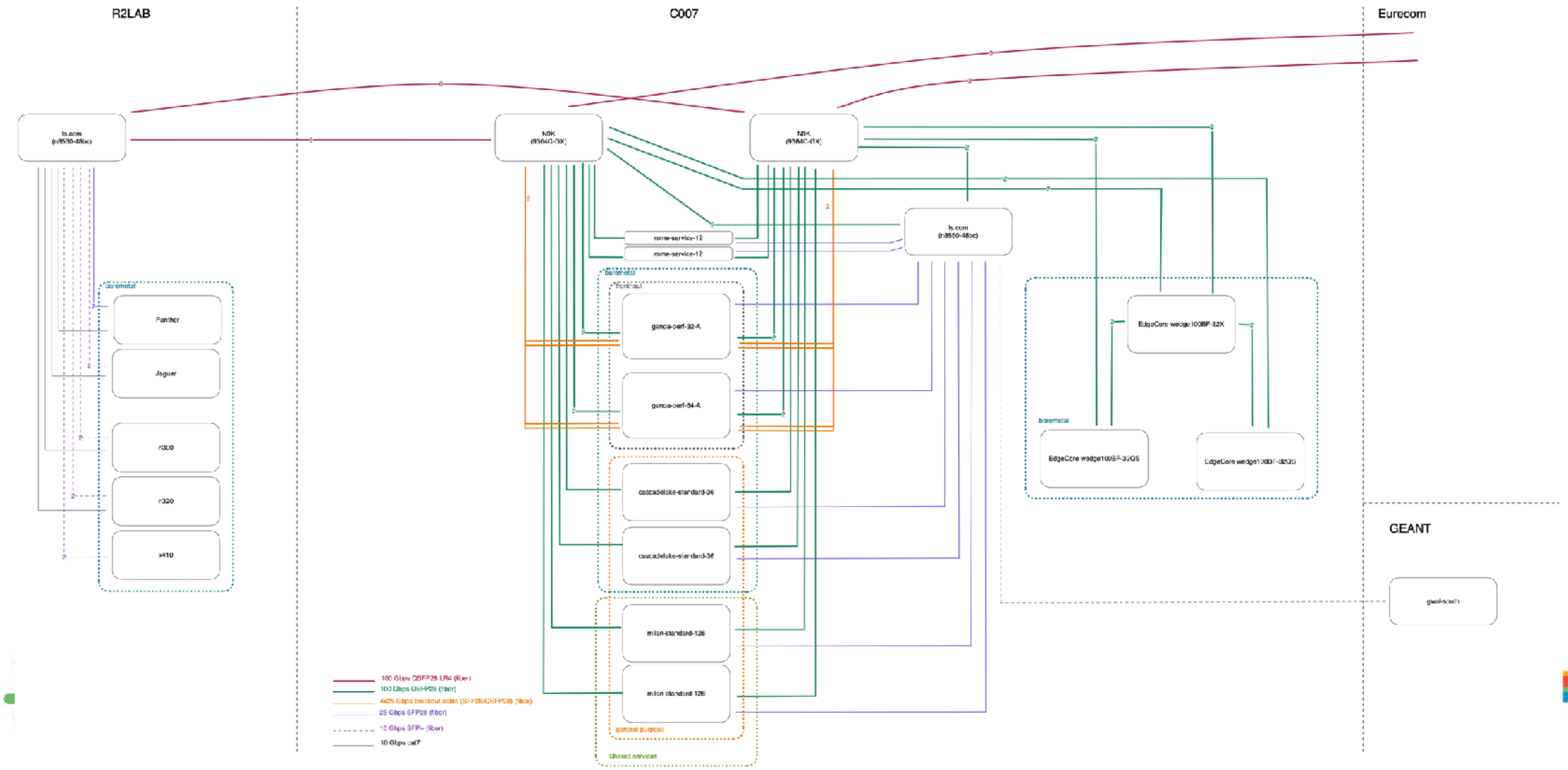
- Blueprint reference implementation relies on independent open source projects
  - all with their own agenda
  - dependencies brake frequently
    - freeze versions whenever possible
    - document dependency graph
    - test, test, test

# Doomed by dependencies

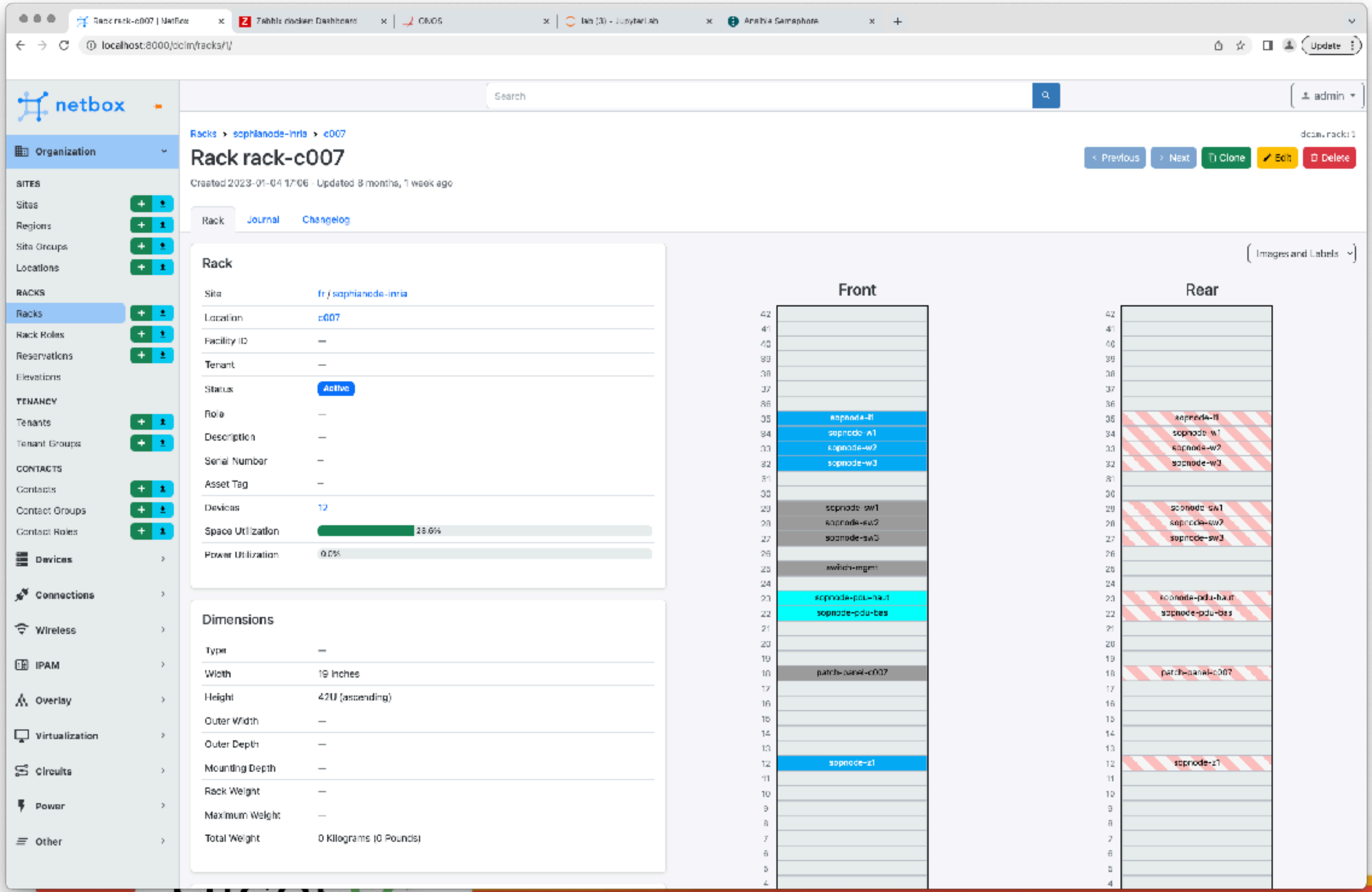
- Bluepri
  source
  - all w
  - dep
    - f
    - c
    - t



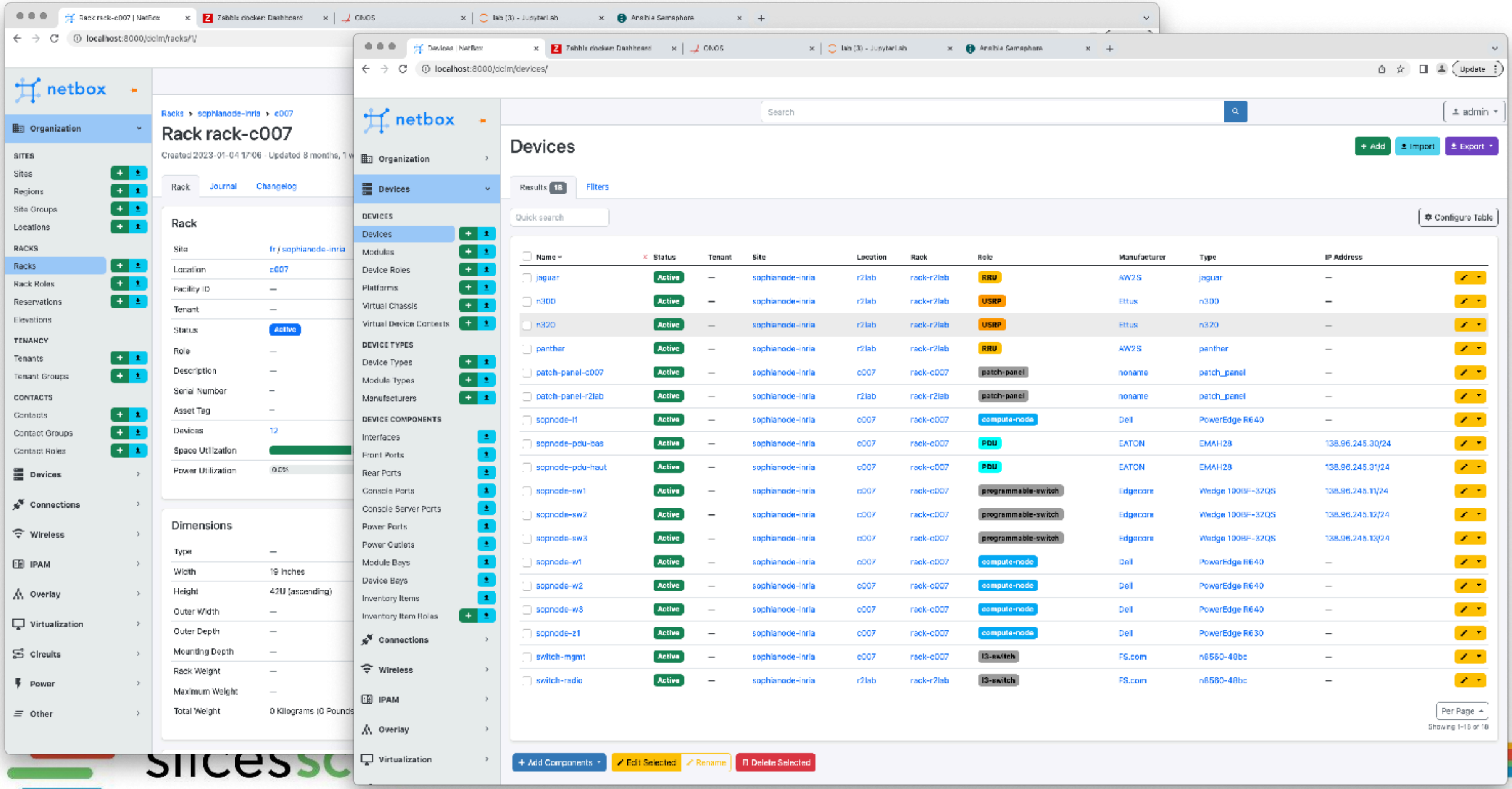
# Heterogenous hw collection



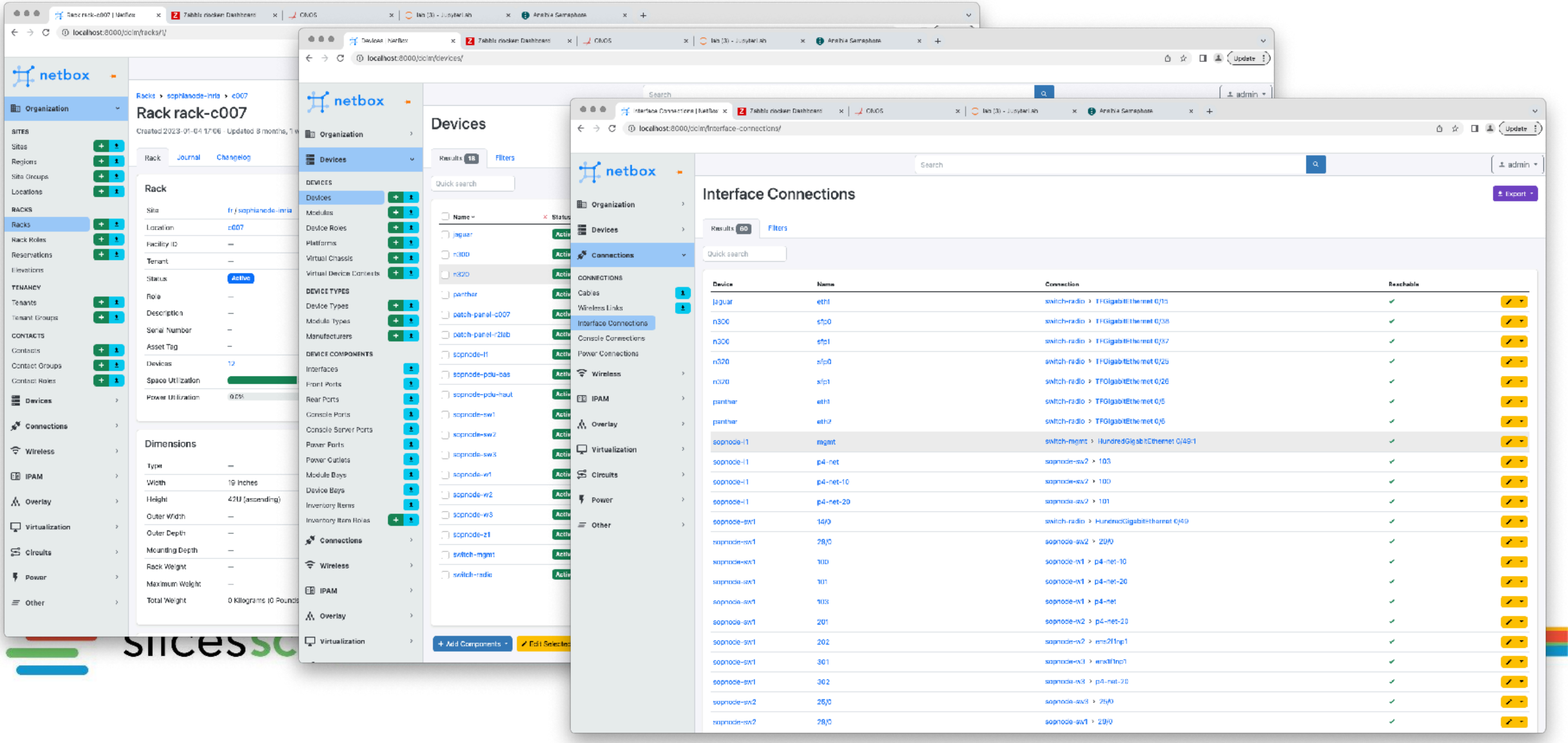
# Source of truth with netbox…



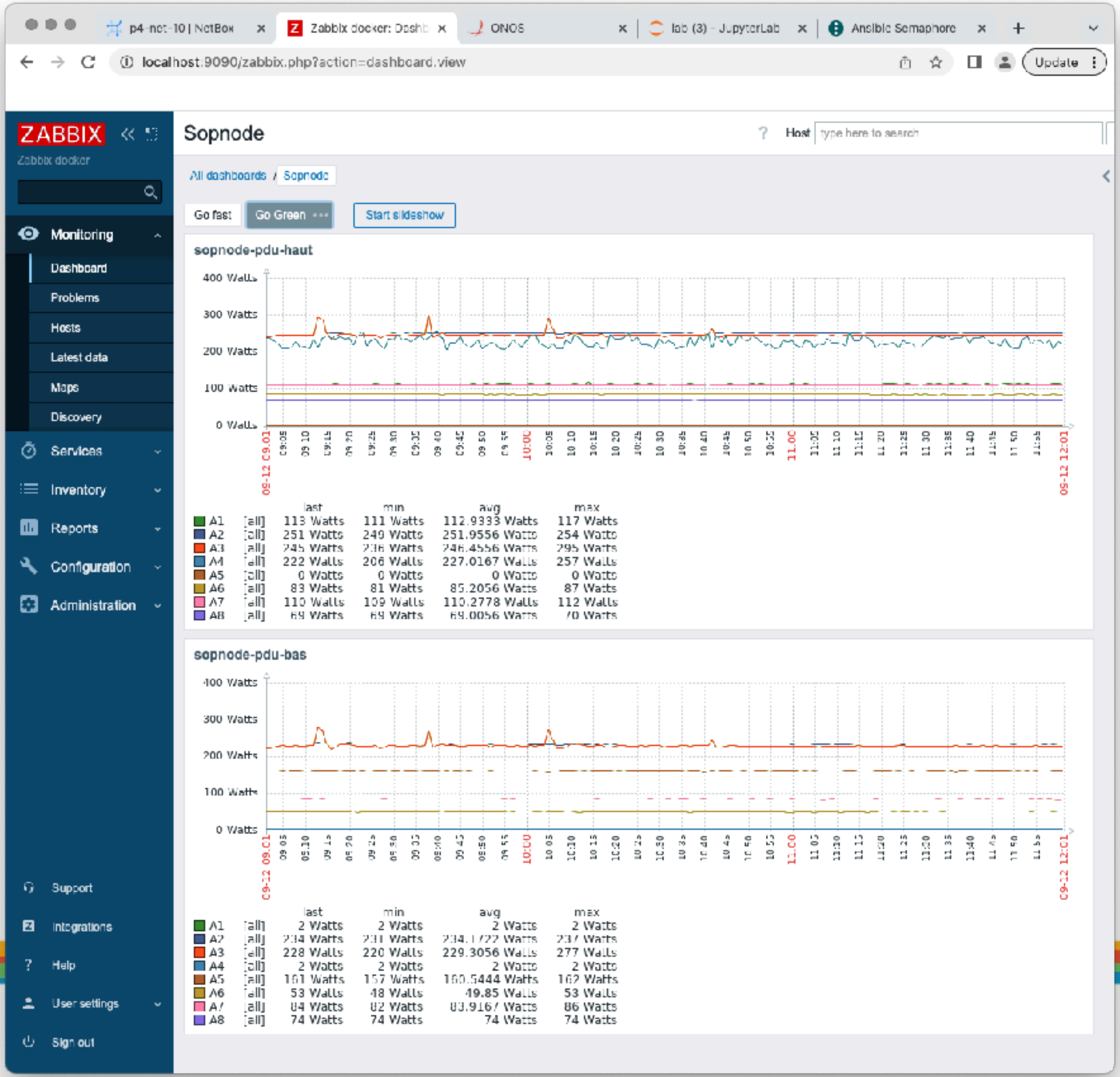
# Source of truth with netbox…



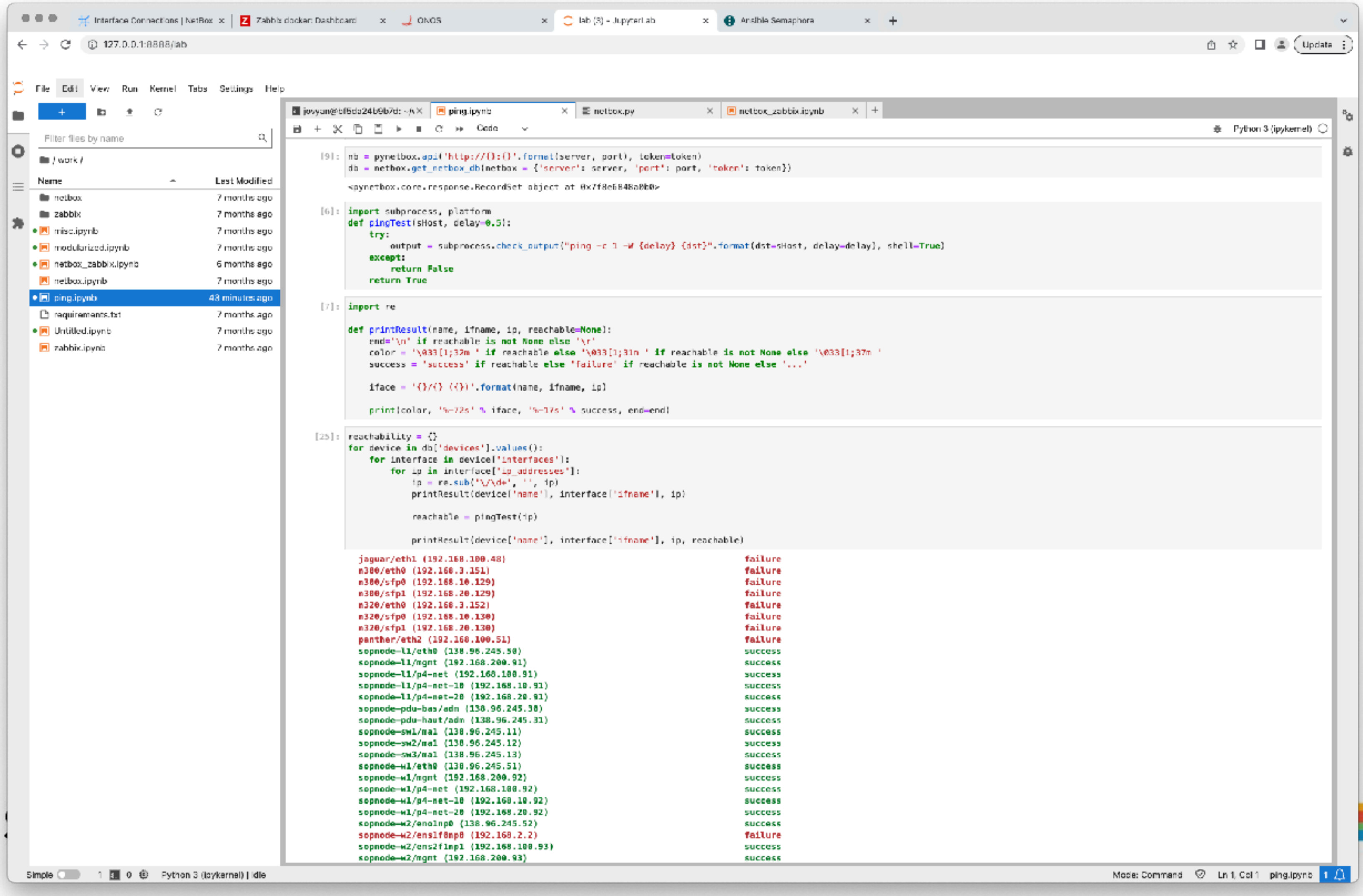
# Source of truth with netbox...



# ... to automate monitoring



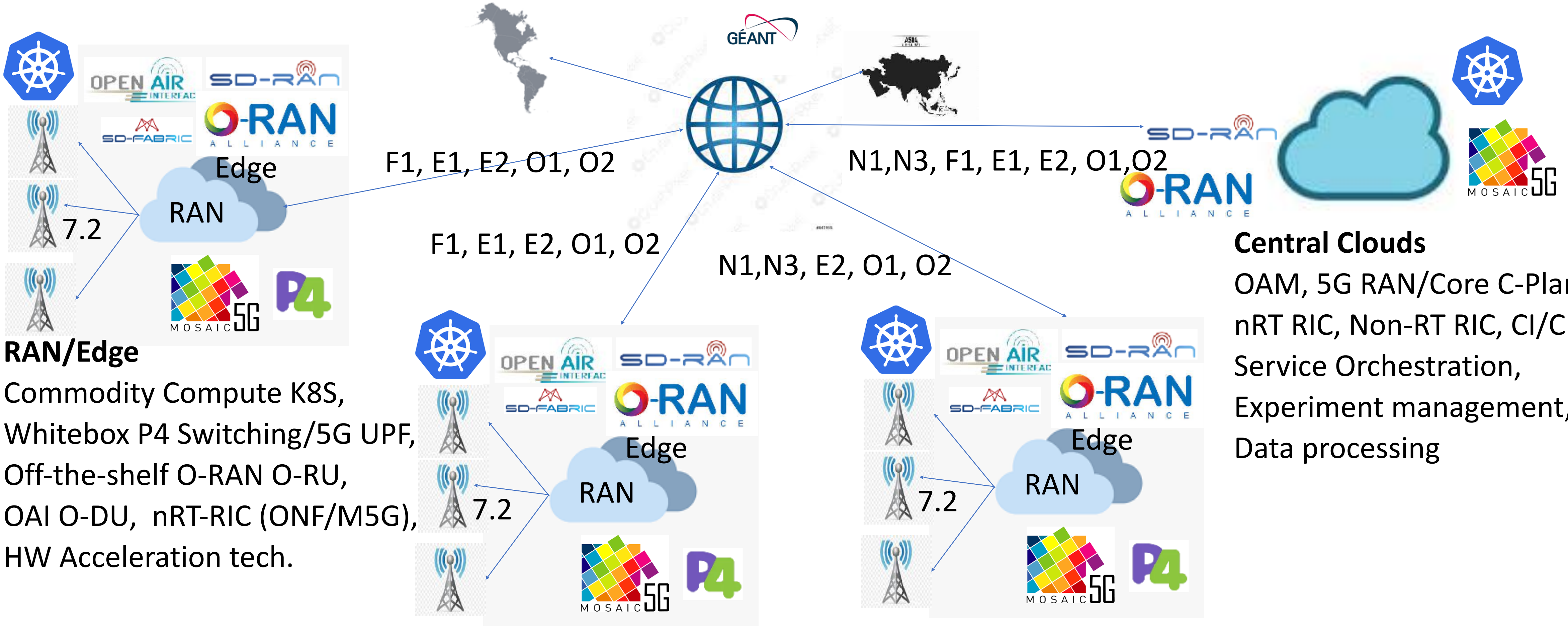
# … to run experiments in a programatic way



slices
SC

# Backup

# Current state




**RAN/Edge**
Commodity Compute K8S,
Whitebox P4 Switching/5G UPF,
Off-the-shelf O-RAN O-RU,
OAI O-DU, nRT-RIC (ONF/M5G),
HW Acceleration tech.

**Central Clouds**
OAM, 5G RAN/Core C-Plane,
nRT RIC, Non-RT RIC, CI/CD,
Service Orchestration,
Experiment management,
Data processing

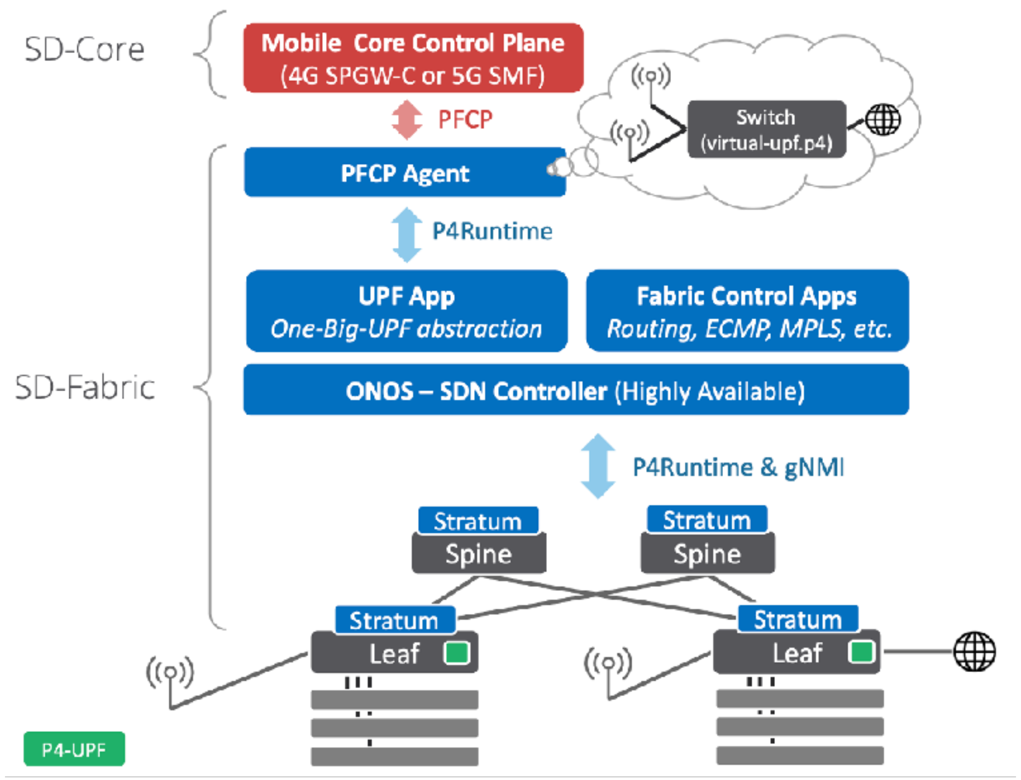
# post5G Experimentation in SLICES

- SLICES first 4-5 years: evolve beyond 5G using open 5G technologies on large-scale end-to-end platforms
- Focus on technologies targeting integration of disaggregated post5G RAN and core with cloud-native deployment framework
- Integration of new applications on experimental post5G infrastructure (SNS C/D)

slicessc

# User-Plane Function blueprints

- Software UPFs are readily available in OSS (OAI SPGW-u, BESS (with free5gcore), Travelping VPP…)
- P4HW UPFs with SD-Fabric




slicessc