
NESi
Network Equipment Simulator


Our Goals

# Our Goals

- Simulation of certain points of a network (DSLAMs as of now)
  - Simulating the interaction between cards, ports and CPEs
- Offering vendor specific management interfaces supporting command line operations
- Creating a testing environment for network management and automation


History

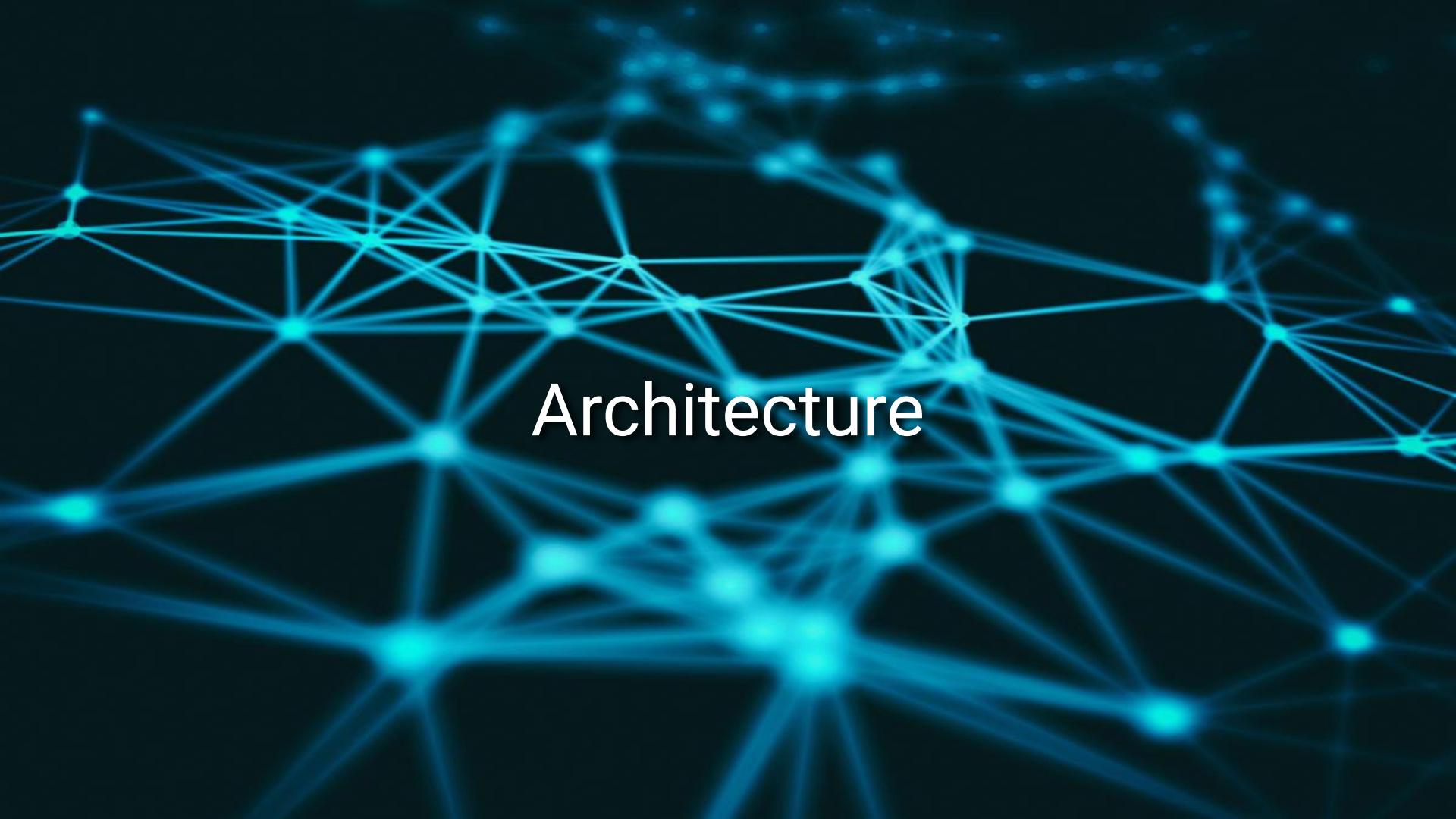
# From Softboxen...

- NESi originates from Ilya Etingof's Softboxen (https://github.com/etingof/softboxen)
- Started as a basic CLI simulation of a dummy Cisco Switch
- Just the basic CLI mode
- Basic CLI functionalities
- Heavily relied on tox

# ...to NESi

NESi now supports:

- lightweight debugging options
- a lightweight testing framework
- multiple network socket modes
- Advanced CLI options (history, arrow navigation, etc.)


Architecture

# REST API

- Software is backed by a REST API
- API has to run permanently in the background (as of now)
- API is the communication backbone between database and CLI
- Allowing the user to create, clone or delete devices and sub-components
- API structure is backed by the flask python module

# REST API - Overview

```python
@app.route(PREFIX + '/boxen/<box_id>/ports', methods=['GET'])
def show_ports(box_id):
    if flask.request.args is None:
        req = {}
    else:
        req = flask.request.args

    response = show_components(PortsSchema(), Port, req, box_id)
    return response, 200


@app.route(PREFIX + '/boxen/<box_id>/ports/<id>', methods=['GET'])
def show_port(box_id, id):
    response = show_component(Port, box_id, id)
    return response, 200


@app.route(PREFIX + '/boxen/<box_id>/ports', methods=['POST'])
def new_port(box_id):
    req = flask.request.json
    ...
```

## REST API Structure:

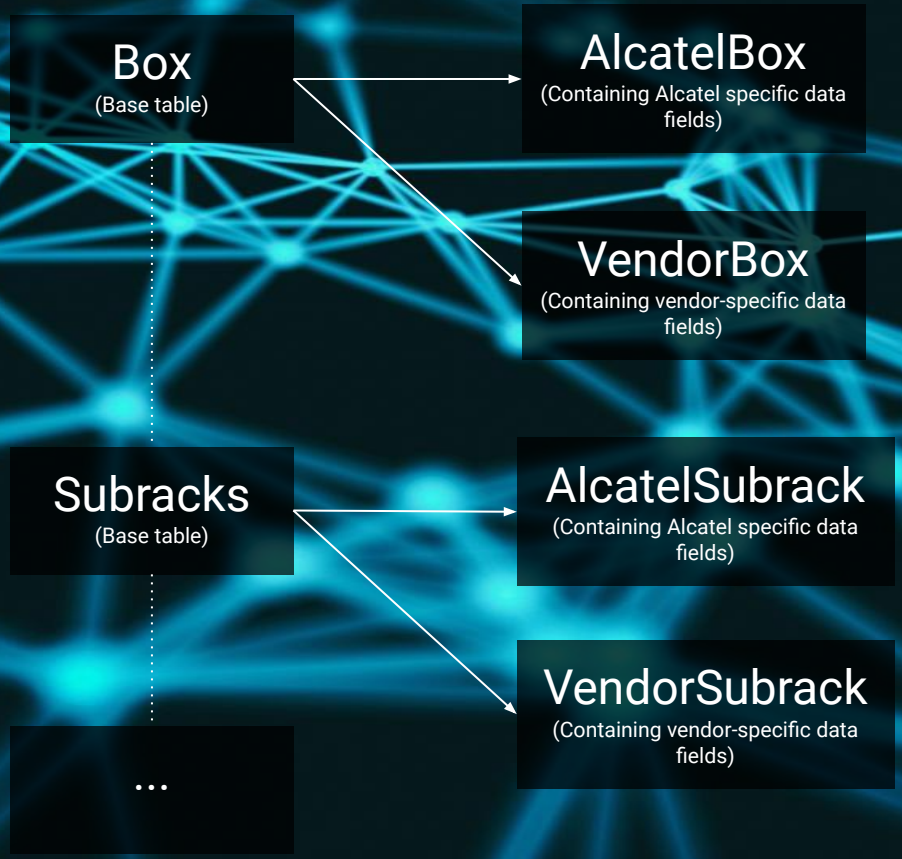
https://127.0.0.1:5000/nesi/v1/boxen/<device-id>/...

- ...subracks/<subrack-id>
- ...cards/<card-id>
- ...ports/<port-id>
- ...

# Database

- SQLite as lightweight database solution in combination with pythons marshmallow library
- Dynamically created on each API launch
- Database structures backed by models and schema files
- Database is populated via shell scripts
- Base tables extended by vendor specific tables

# Database - Overview



# CLI

- CLI backend written in Python
- CLI can be operated in either daemon or gui mode
- Retrieving data from DB and dynamically filling Jinja2 templates to generate output
- CLI is based on a REPR loop
- Advanced CLI operations (local history, arrow navigation)

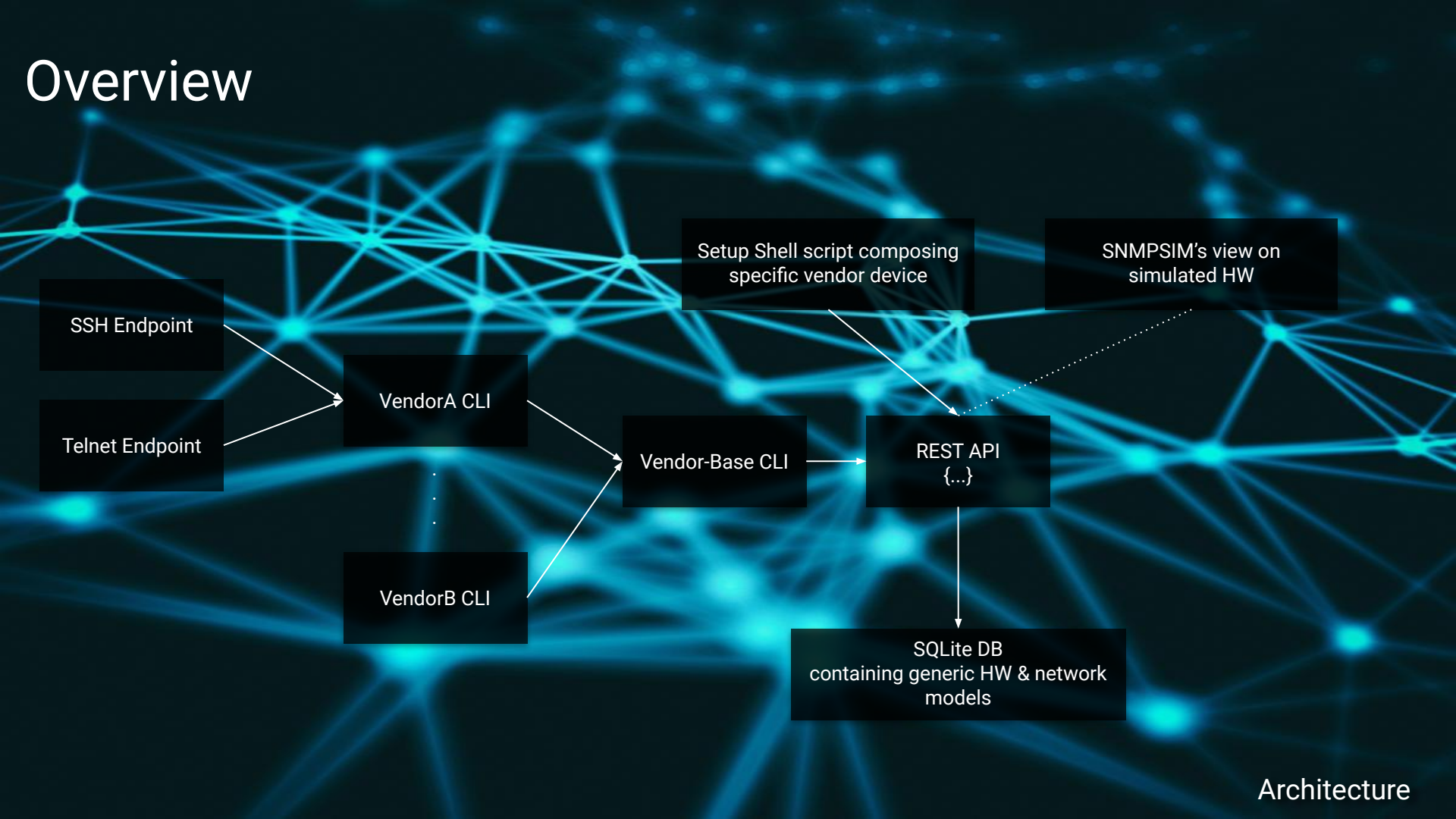
# REPR Loop

Read - Evaluate - PRint

The command processor implementing this loop operates like the following:

- Reading user input up to <CR>
- Try matching input against known CLI commands
- Applying command to underlying device
- Locating correct Jinja2 template and presenting it to the user

# Overview



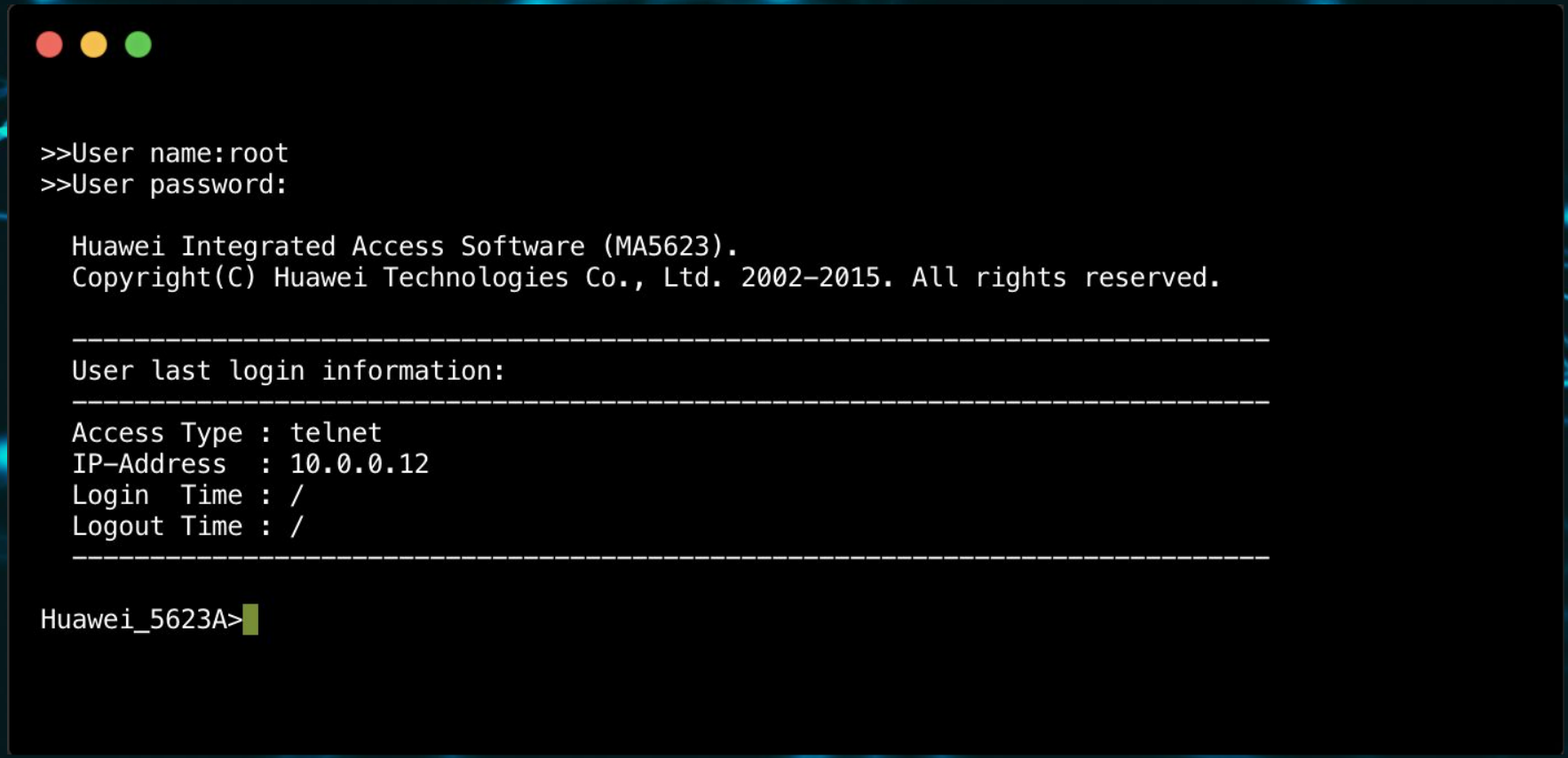

Architecture


Vendors

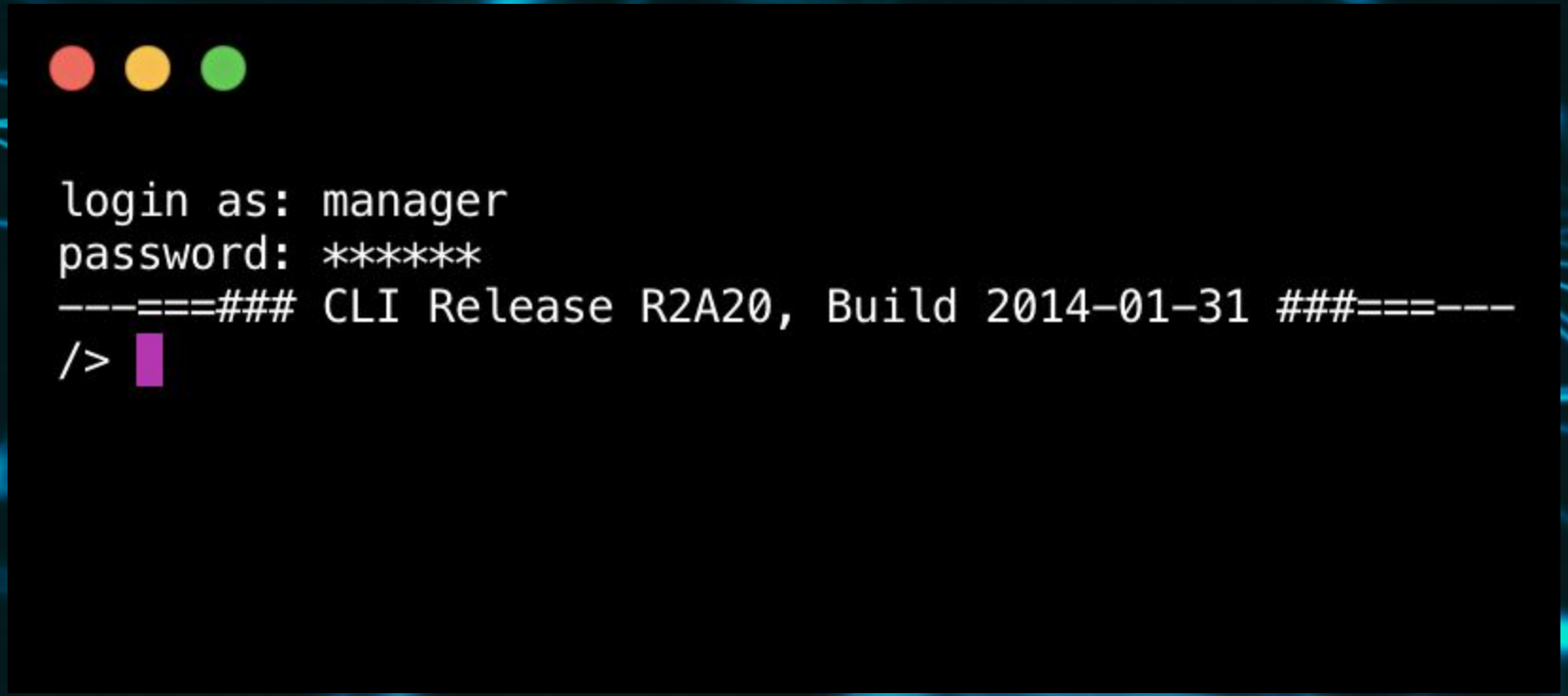
# Alcatel


login:admin
password:
Welcome to Alcatel_7360
last login : 01/03/2021 14:09:57
Alcatel_7360>#

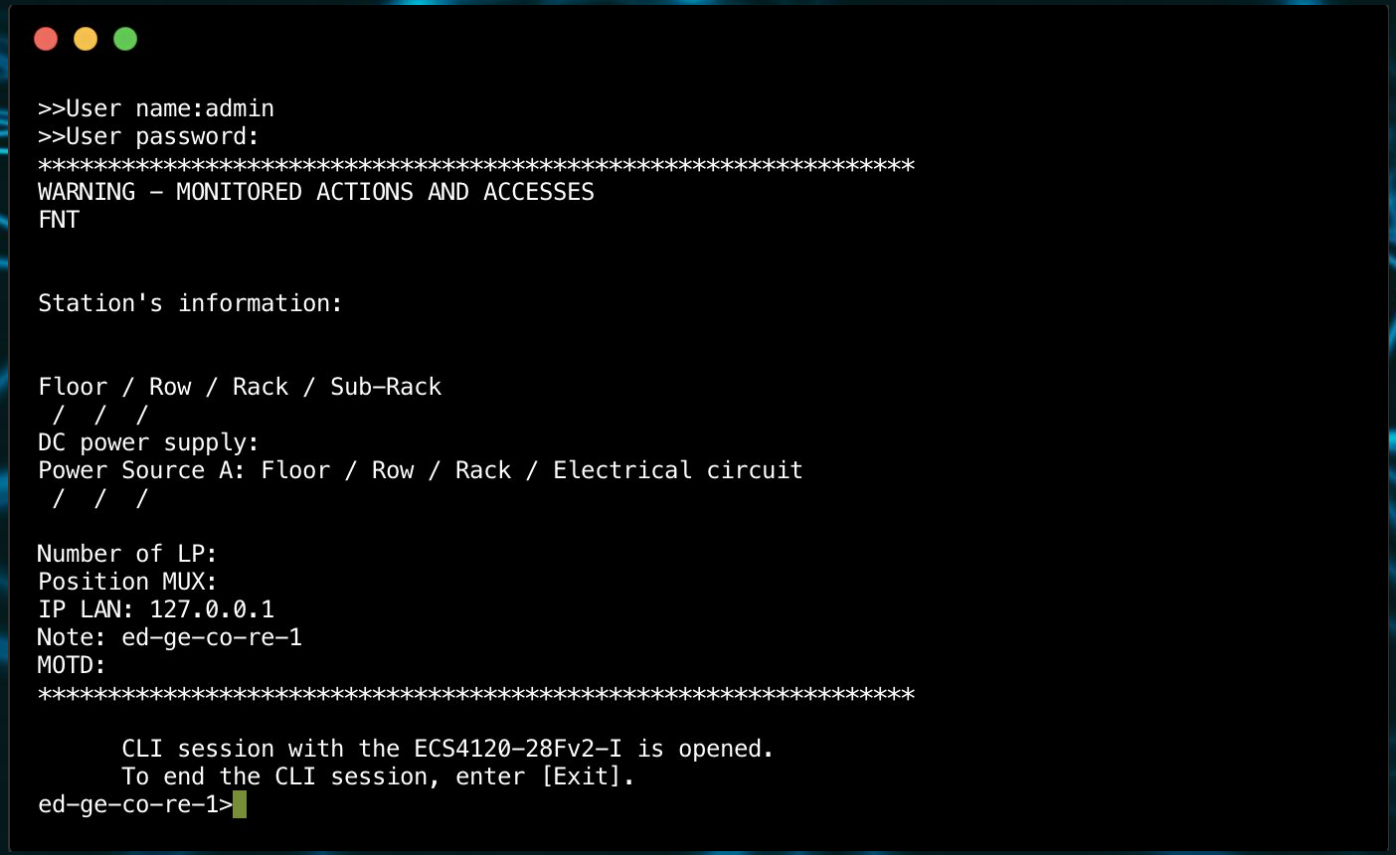
# Huawei


>>User name:root
>>User password:
Huawei Integrated Access Software (MA5623).
Copyright(C) Huawei Technologies Co., Ltd. 2002-2015. All rights reserved.
User last login information:
Access Type : telnet
IP-Address : 10.0.0.12
Login Time : /
Logout Time : /
Huawei_5623A>

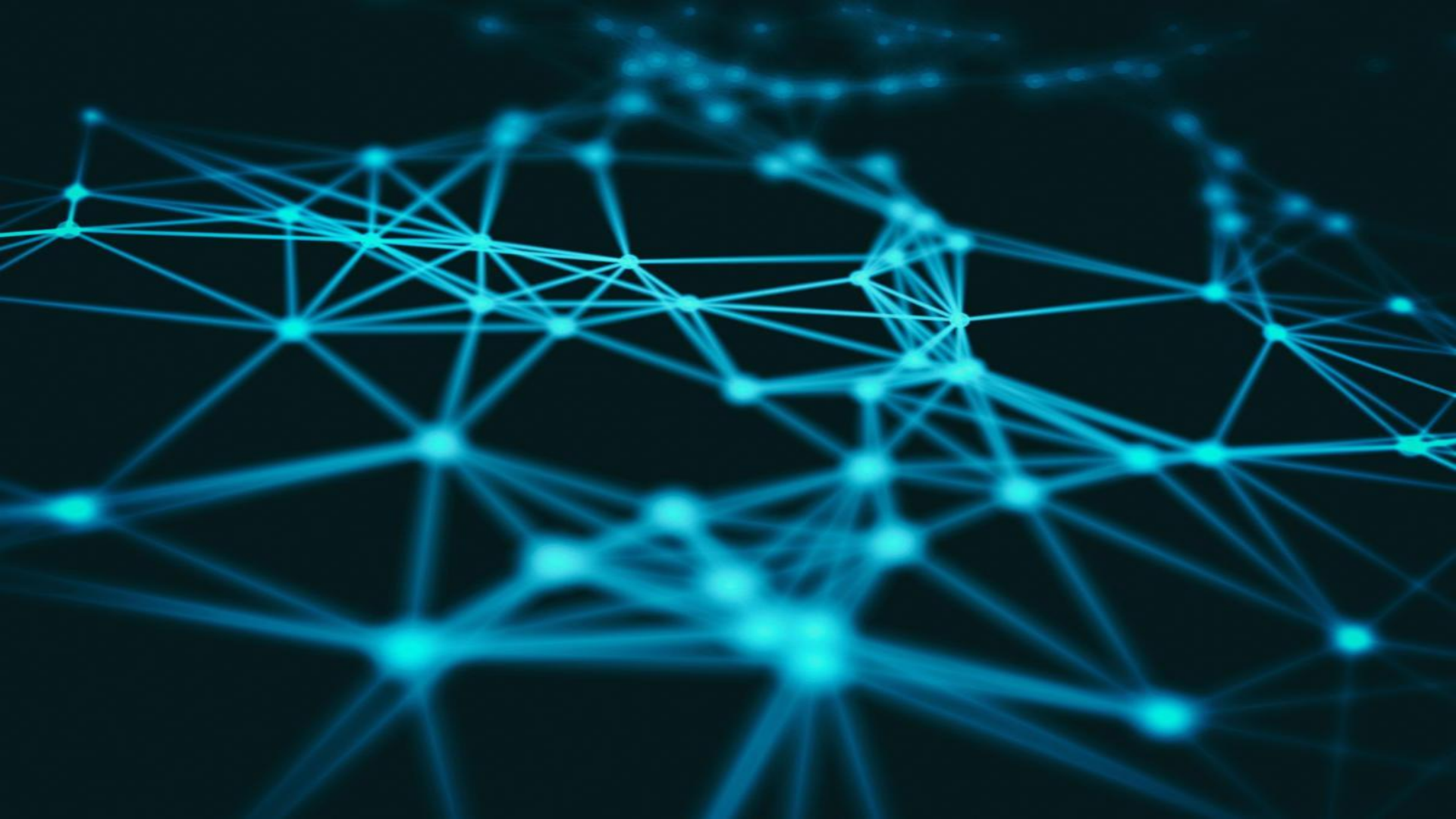
# KeyMile

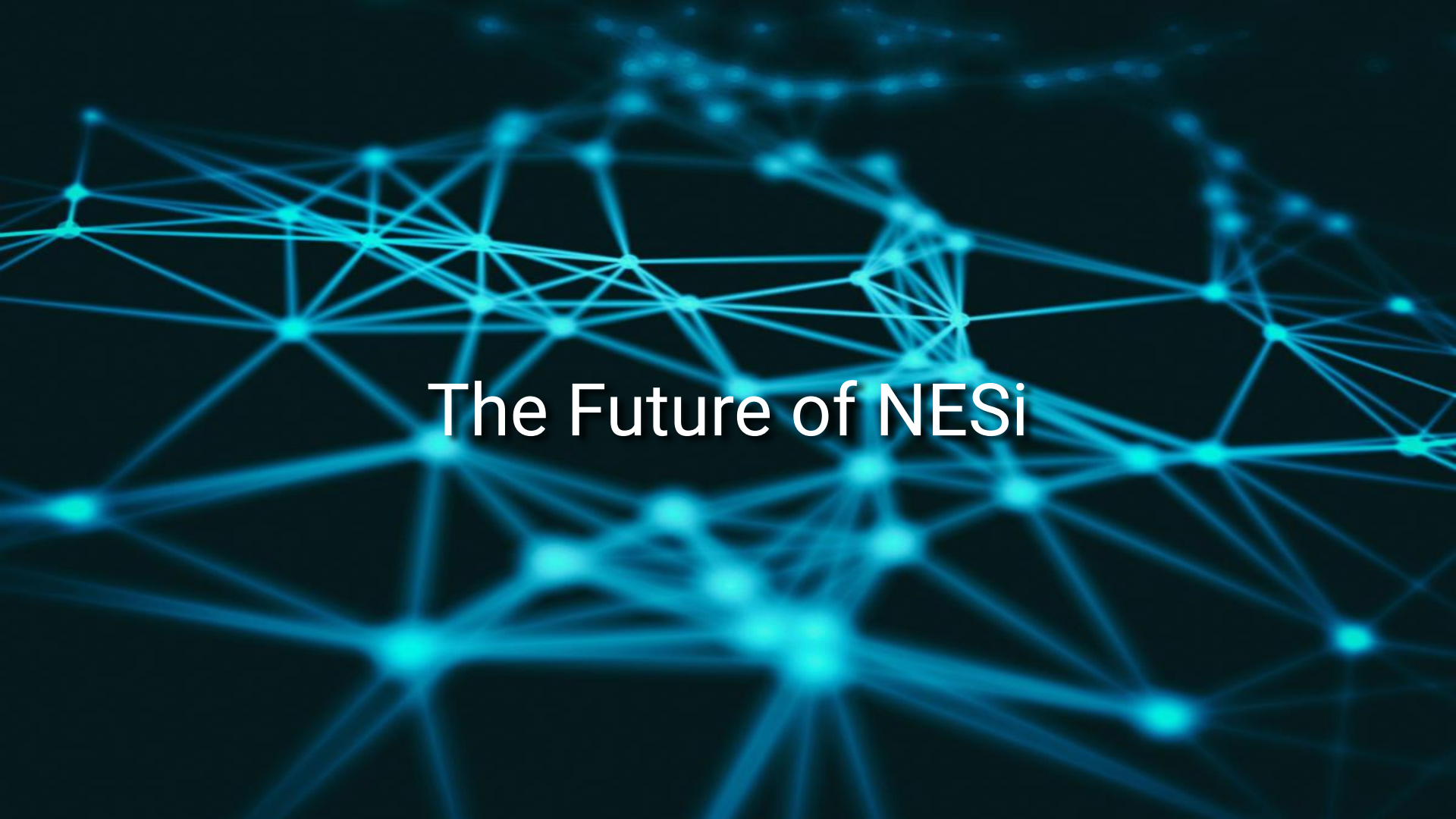

login as: manager
password: ******
---===### CLI Release R2A20, Build 2014-01-31 ###===---
/>

# EdgeCore


>>User name:admin
>>User password:
**********
WARNING - MONITORED ACTIONS AND ACCESSES
FNT
Station's information:
Floor / Row / Rack / Sub-Rack
/ / /
DC power supply:
Power Source A: Floor / Row / Rack / Electrical circuit
/ / /
Number of LP:
Position MUX:
IP LAN: 127.0.0.1
Note: ed-ge-co-re-1
MOTD:
**********
CLI session with the ECS4120-28Fv2-I is opened.
To end the CLI session, enter [Exit].
ed-ge-co-re-1>


Demo




The Future of NESi

# Features to come

- Separated databases for each vendor
- Separation of API and database
- Improved exception-handling
- Python auto-docs
- Improved telnet socket
- Separated physical and logical layer
- Implementation of SNMPSIM

# SNMPSIM

- Can simulate many thousands of different SNMP speaking devices
- Runs over IPv4 and/or IPv6 transports
- Varies response based on SNMP Community, Context, source/destination addresses and ports
- Offers REST API based control plane

in combination with NESi

- Possibility to simulate states of NESi devices from the SNMP point of view
- Implementation as either python module or built in stripped down backend


Question & Feedback


Looking forward to your contributions!

# NESi on GitHub

https://github.com/inexio/NESi

# NESi-UI on GitHub

https://github.com/inexio/NESi-UI

# SNMPSIM on GitHub

https://github.com/inexio/snmpsim


Thank you!