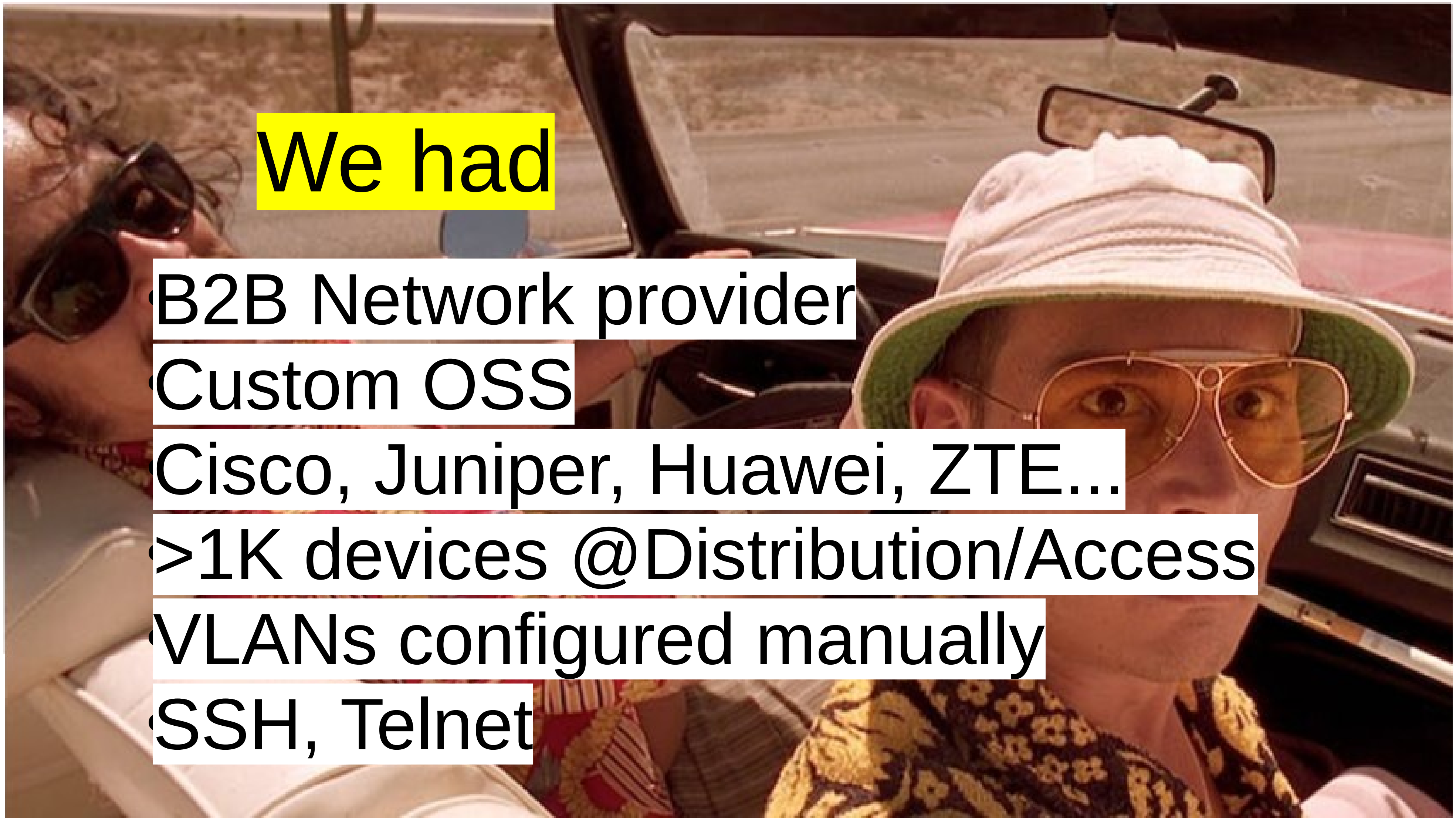
How we configured >100 routers in .1 sec

Note the dot

The Contents

Intro

0x1 of history



We had

- B2B Network provider
- Custom OSS
- Cisco, Juniper, Huawei, ZTE...
- >1K devices @Distribution/Access
- VLANs configured manually
- SSH, Telnet

We
CTO
(Network Engineer)
Me

Network Provider

- B2B
- medium-to-large size
  thousands to tens thousands clients
- Network services: VPN, PtP, VoIP etc.
- Custom OSS (Operation Support System)

OSS

- common for Network Service Providers, This one custom made in-shop
- Web Application
- Functions like client service records, visual network calculator, etc.
- Billing is separate

Vendors

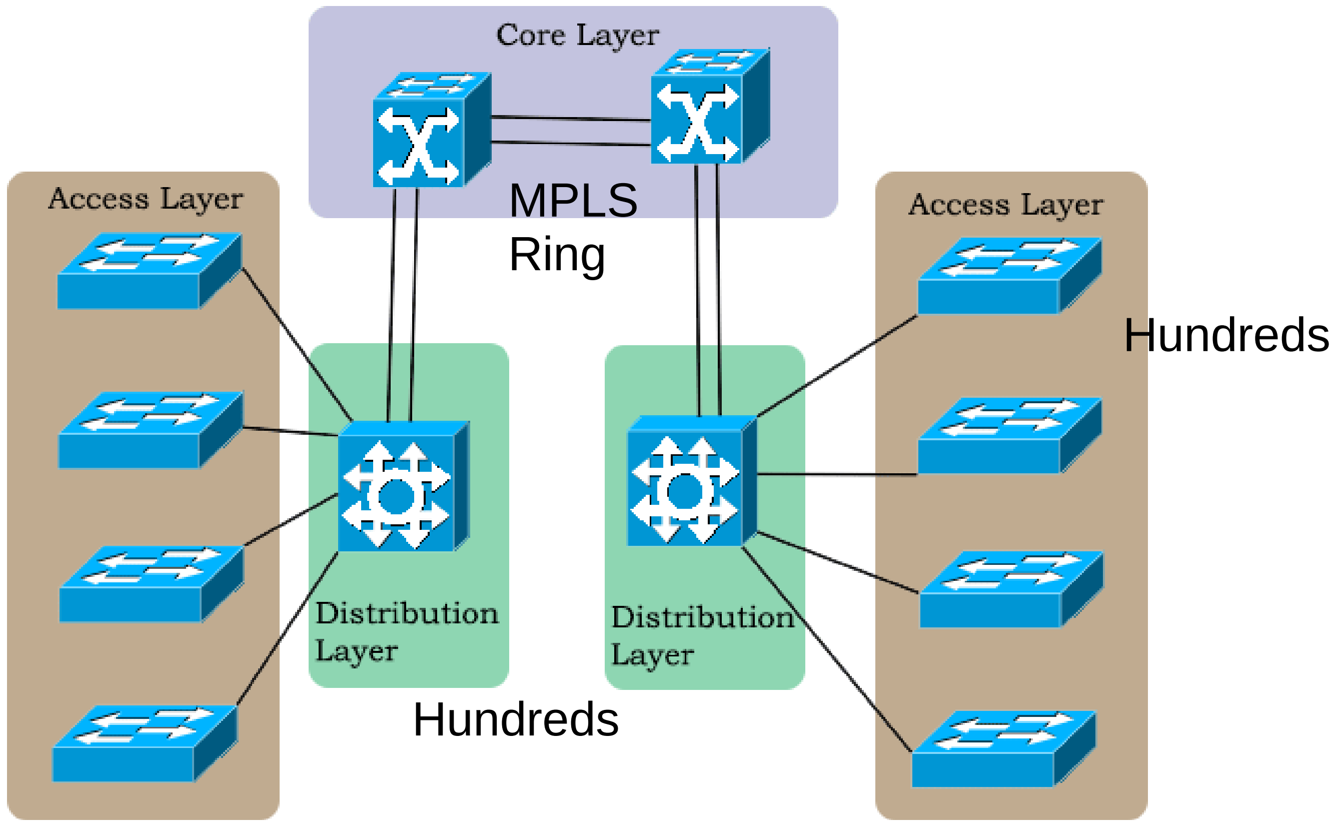
Cisco
Huawei
Eltex
Juniper
Extreme
Raisecom

UPM ABB Outotec NOKIA FINNAIR RELEX Nordea VISMA KON ESTE

Network Redesign

- Chaotic
- Then good
- Then we want automate

Core, Distribution, Access



VLANs

- L2, L3 VLAN Service @Distribution/Access
- Configured manually via SSH/Telnet
- 30 – 90 minutes for operator network engineer
- Lots of possibilities for error

SSH/Telnet

- CLI
- Notepad++
- Manually checking port links
- Coffee breaks

  when operator wants to have a coffee and goes away from keyboard in the middle of VLAN finding/configure operation, it’s common that something breaks

The Plan

The Plan

- Automate routine tasks, e.g. VLANs
- Minimal possible action from operator
- Reliable
- Traceable
- Keeping it simple (with limited resources)

KISS

- Various vendors, non-consistent
- SSH/Telnet already used
- Let’s try just imitate operator’s work

🚩 Don’t use telnet

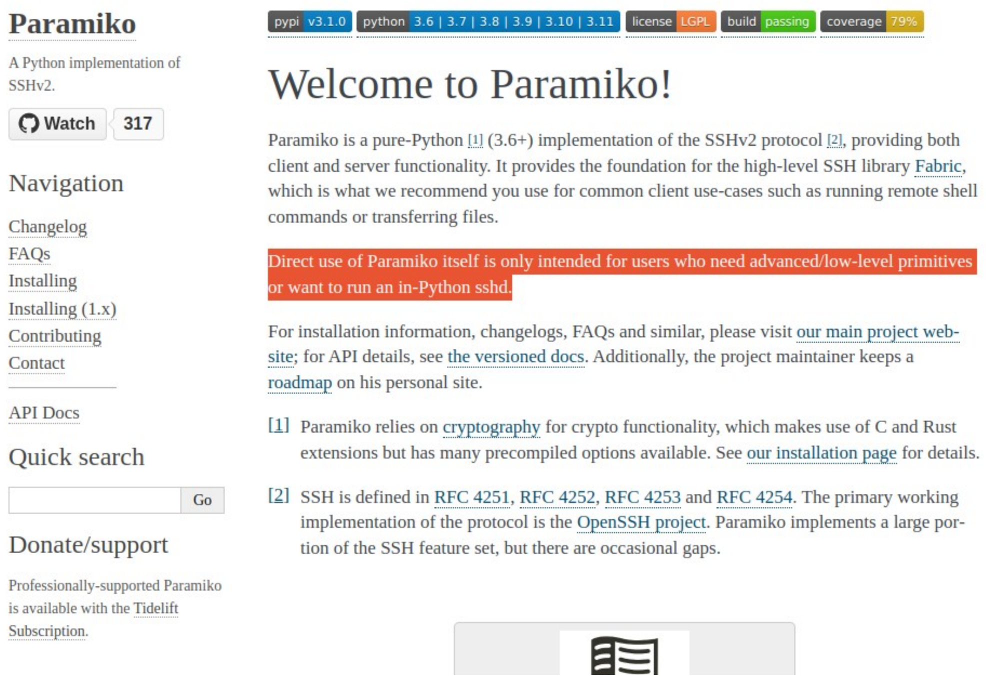
SSH + Python

Pic of python snake doing sssh..

https://github.com/ktbyers/netmiko

NETMIKO

https://github.com/paramiko/paramiko



Paramiko

A Python implementation of SSHv2.

Watch 317

Navigation

- Changelog
- FAQs
- Installing
- Installing (1.x)
- Contributing
- Contact

API Docs

Quick search

Go

Donate/support

Professionally-supported Paramiko is available with the Tidelift Subscription.

pypi v3.1.0 | python 3.6 | 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | license LGPL | build passing | coverage 79%

Welcome to Paramiko!

Paramiko is a pure-Python [1] (3.6+) implementation of the SSHv2 protocol [2], providing both client and server functionality. It provides the foundation for the high-level SSH library Fabric, which is what we recommend you use for common client use-cases such as running remote shell commands or transferring files.

Direct use of Paramiko itself is only intended for users who need advanced/low-level primitives or want to run an in-Python sshd.

For installation information, changelogs, FAQs and similar, please visit our main project website; for API details, see the versioned docs. Additionally, the project maintainer keeps a roadmap on his personal site.

[1] Paramiko relies on cryptography for crypto functionality, which makes use of C and Rust extensions but has many precompiled options available. See our installation page for details.

[2] SSH is defined in RFC 4251, RFC 4252, RFC 4253 and RFC 4254. The primary working implementation of the protocol is the OpenSSH project. Paramiko implements a large portion of the SSH feature set, but there are occasional gaps.

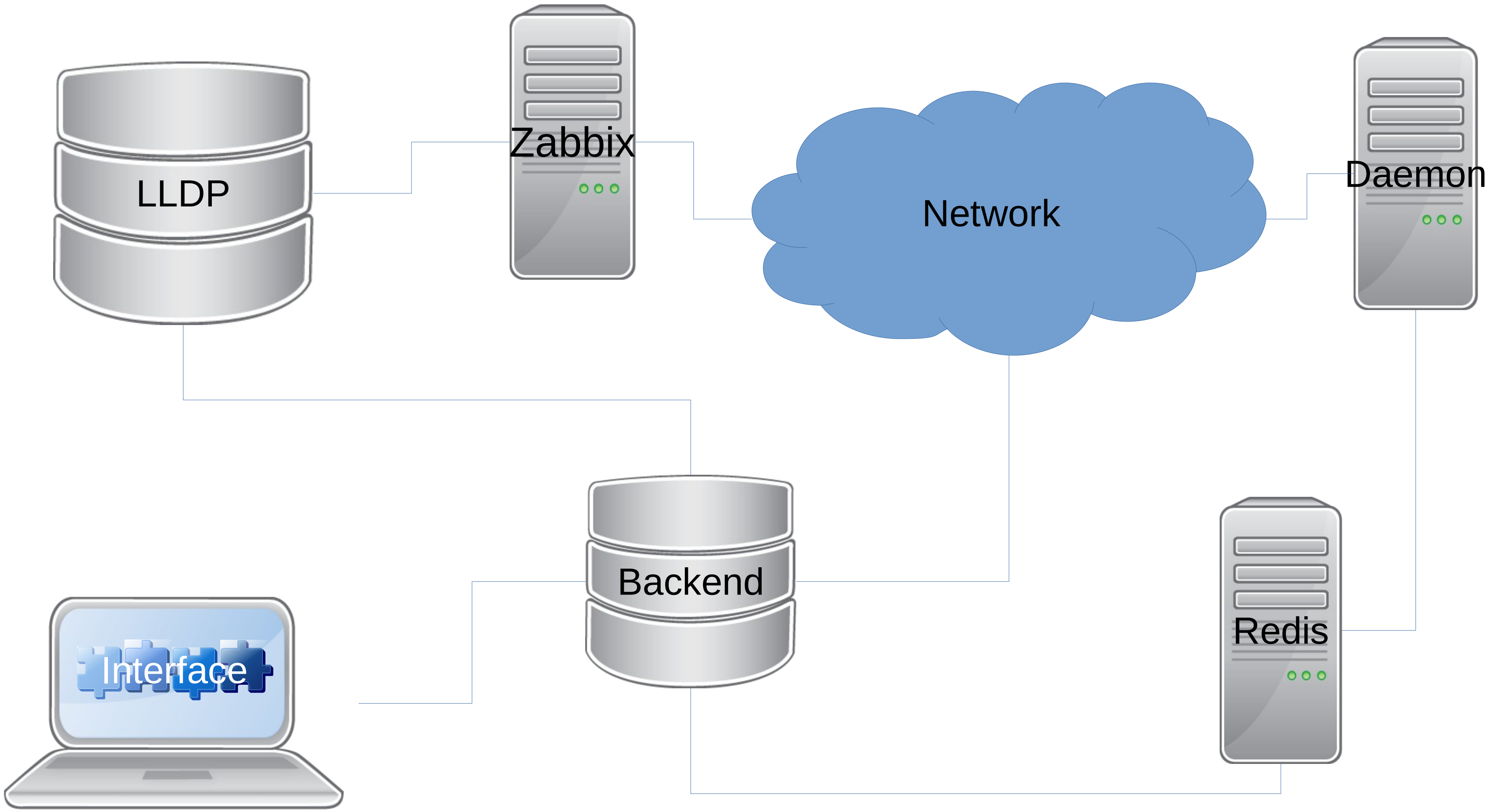
“Direct use of Paramiko itself is only intended for users who need advanced/low-level primitives...”

“Direct use of Paramiko itself is only intended for users who need advanced/low-level primitives...”

✓ OK

The System

aka Pushkin


Zabbix
LLDP
Network
Daemon
Backend
Interface
Redis

Moving Parts

- Integrated with OSS (skipping details)
- (Web) interface
  scripting, device map, administration
- Backend
  communicates with Zabbix, Redis, Daemon
- Zabbix with LLDP module
- Redis messaging queue
- Daemon (Custom Python lib daemonized)

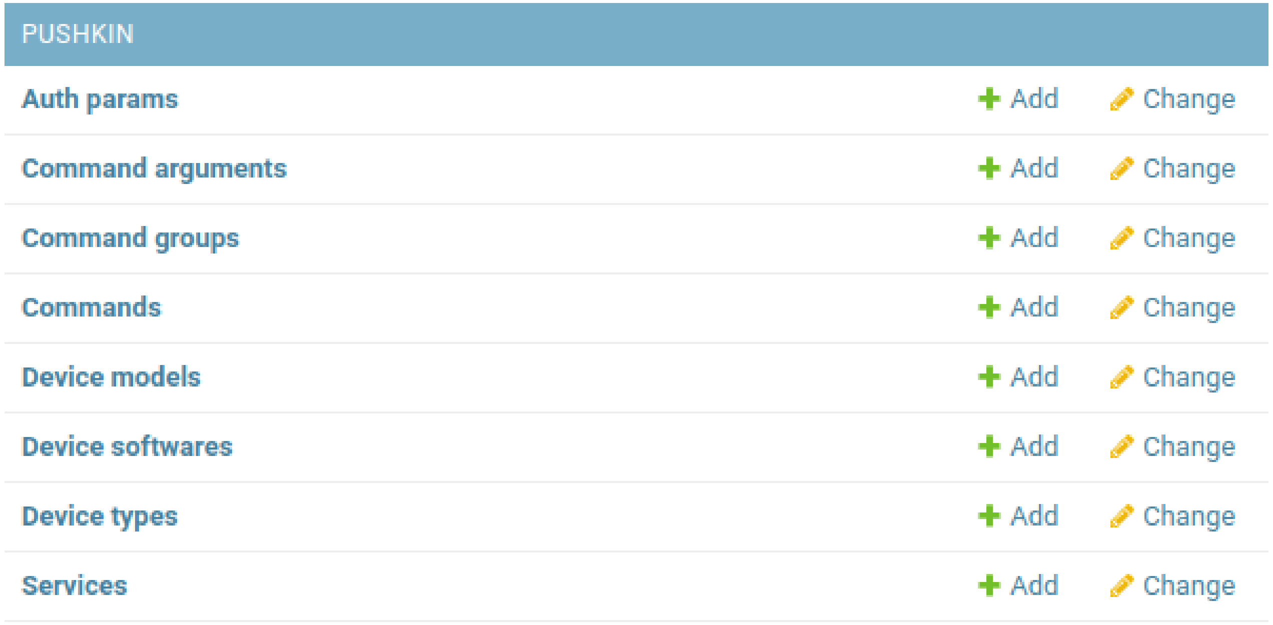
Web Interface

- Django
- Sqlite
- Users, Groups
- Sortedm2m
- bootstrap

/operator

- Blurred image screenshot
- Choose a free resource (e.g. vacant VLANID)
- Choose a termination device (based on device map & client address)
- Hit “Fire” button
- Within 30sec watch for Service Status change to OK
- Make self a coffee (external vendor)

/admin



PUSHKIN		
Auth params	Add	Change
Command arguments	Add	Change
Command groups	Add	Change
Commands	Add	Change
Device models	Add	Change
Device softwares	Add	Change
Device types	Add	Change
Services	Add	Change

/admin

- Auth Params
- Device params
  vendor, model, os version, ...
- Command Groups
- Commands

Scripting inside

- Conceptually a DSL from Jetbrains’ MPS
- Provide a domain-specific lang constructs
- Translates into usual Vendor OS- and version-specific commands for particular device
- DRY
- Bonus: can be rhymed into poem

Authentication
select commands
Vendor specific,
OS version specific

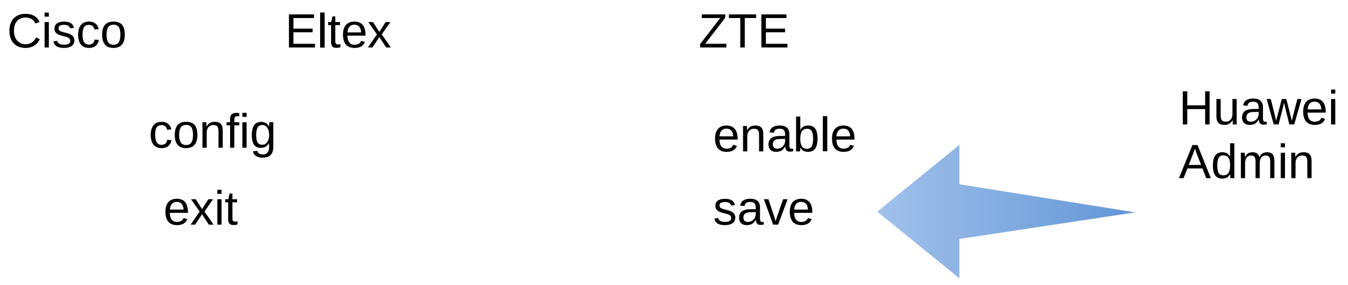
operator

```
Create vlan <id> <name>
```

```
Config
Vlan <id>
Config vlan <id>
Name <name>
Exit
exit
```

DRY

- Specific commands, no repeat
- Command group (“create vlan”)
- Hence command reuse



	config	enable	exit	save
cisco	✓		✓	
eltex	✓		✓	
huawei		✓		✓
zte		✓	✓	

Pushkin vs. Netmiko part 1

- Command reuse, no need for separate class for each vendor, model, os/version combination
- Store only commands that differ

Command Args

- Simple parser
- Anywhere in the command
- Create vlan <vlanid> <name>
- Create <vlanid> vlan with <name> of client

Port numbering

- GE1/0/1
- Fa0/1
- Port0
- All come to just integer: 0, 2, 5, 15 …
- Make Port <id> access
- Make Port <id> trunk

Man

```
Device selector:
<boxid>[,<boxid>][-<boxid>].<clli>

Id == CLLI-encoded unique name


Command <arg> group <arg> with <portid> optionally

Example:

    sw07-sw09,agw01.blgrd
    create vlan 25 name BGMotorGarage
    add vlan 25 to trunk port 12

    sw19.nvsd
    create vlan 25 name BGMotorGarage
    add vlan 25 to port 2
    make port 2 access
```

Messaging via Redis

- Simple messaging queue with JSON payload
- Example

```
[
 {“device”: “box007.blgrd”,
  “commands”: [“comm1”,”comm2”,…]},
 {“device”: “sw03.blgrd”,
  “commands”: [“comm1”,”comm2”,…]},
 …
]
```

- Not necessary but nice

Network Map Builder

aka Zabbix

- 15K NVPS
- LLDP via SNMP
- Custom LLDP module Made in Japan
- >2TB weekly
- ~15K devices
- ~1K with LLDP

Zabbix

- Was not a best choice
  (confirmed with Zabbix Support Team)

- It did OK
  with some hacky setup with Zabbix Proxies

- Next time Consider OpenNMS or Proprietary
- And don’t mix LLDP and Common Monitoring

Network Map

- All paths were calculated based on LLDP fetched via SNMP
- Build Graph with Dijkstra's Shortest Path
- Operators choose one option:
  termination device
- All links constructed

Daemon

Socket Programming

- Of a healthy person:

```
s = socket()
s.write(“command\n”)
result = s.read()
s.write(“command\n”)
result = s.read()
s.write(“command\n”)
result = s.read()
```

- The smoker:

```
s = socket()
s.write(“command;command;conquer\n”)
result = True
```

Pushkin vs. Netmiko part 2

- 30 sec – 1.5 min
  Netmiko
- 0.1 – 0.5 sec
  Pushkin

Wait, but...

- Hand the socket over to background thread,
  receive response from device,
  log it
- If everything was fine you don’t need it
- If smth bad happen, you read the logs

Test the connectivity

- ARP-tables filled up with mac-address
  when the ping reaches the host
  inside the valid vlan

  So, we got ARP == we made a vlan

  No need to read (and parse) device’s response

General Notion of Service

- Service = commands + test procedure
- Commands configure
- Test asserts success of configure
- And also is a metric which can be used for a Service health monitoring

VLAN Service

- VLAN
  Create vlan <id> <name>

  Add vlan <id> to trunk

  Add vlan <id> to access

  Test
  ping access ip
  check ARP table for src MAC

Backend

- LLDP via SNMP collected
- Device Graph constructed
- Devices with ports in Database
- Free VLANs in Database

Frontend

- Operator selects 2 things
- Termination Device
- VLAN ID
- VLAN name is constructed based on Client’s name, uniqueness and human readability ensured

Outro

Outro

- OSS, business value
- SDN
- NETCONF, OpenFlow
- Vendor-specific
- NetDevOps
- Network as a Service, self configuration
- OSS-enabled (users’ workflow unchanged)
- How to apply Basic building blocks to fit in the big picture

Show me the code

- https://github.com/iberestenko/highload

The End

?