Automate Your Network in 5 Easy Steps With Python and Netmiko

Luca Gubler

DevNet Expert 2023::4

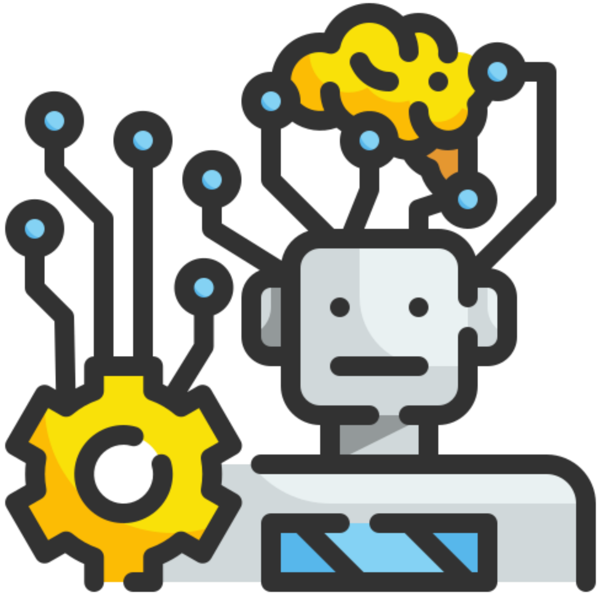
24.10.2024

Agenda

- What Is Network Automation?
- What Is Netmiko?
- 5 Steps to Automate Your Network
- Next Steps

After This Talk, You’ll Be Able To

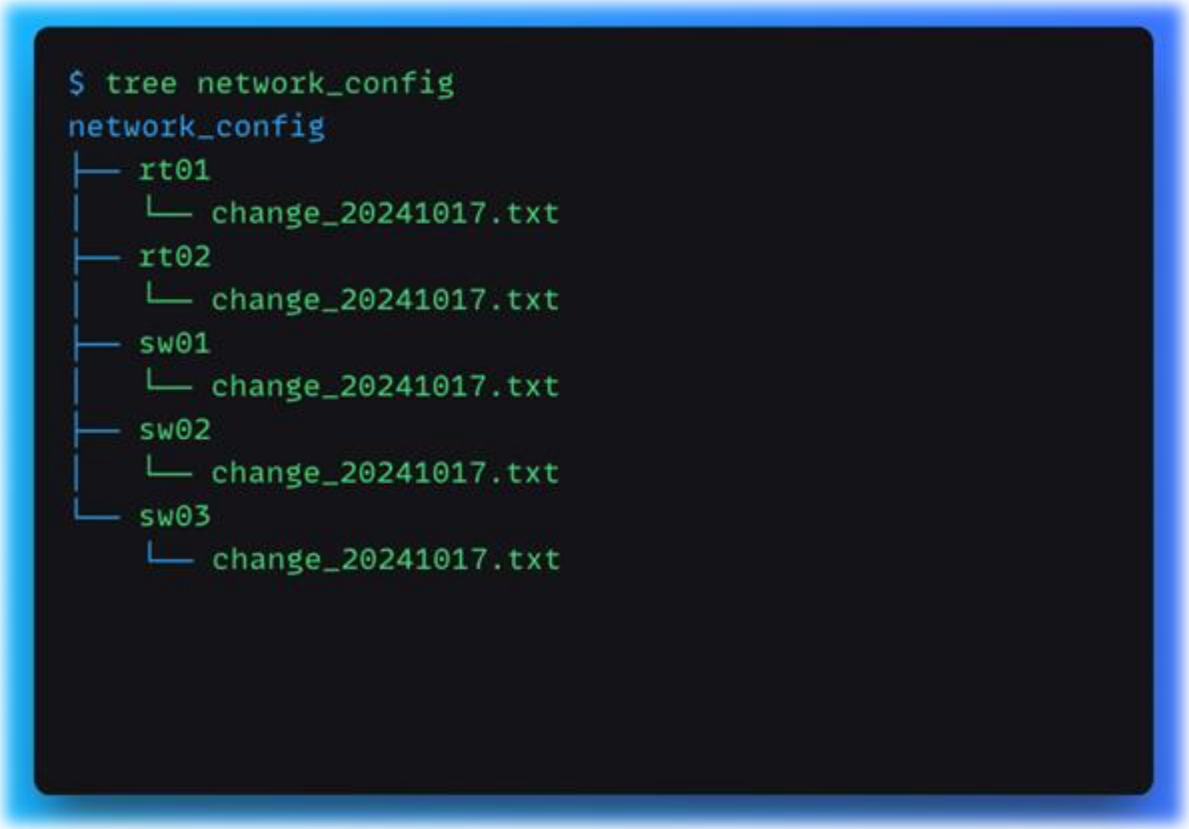
- … understand challenges network engineers face
- … understand the need for network automation
- … write basic Netmiko scripts



What Is Network Automation?

Traditional Network Change Workflow

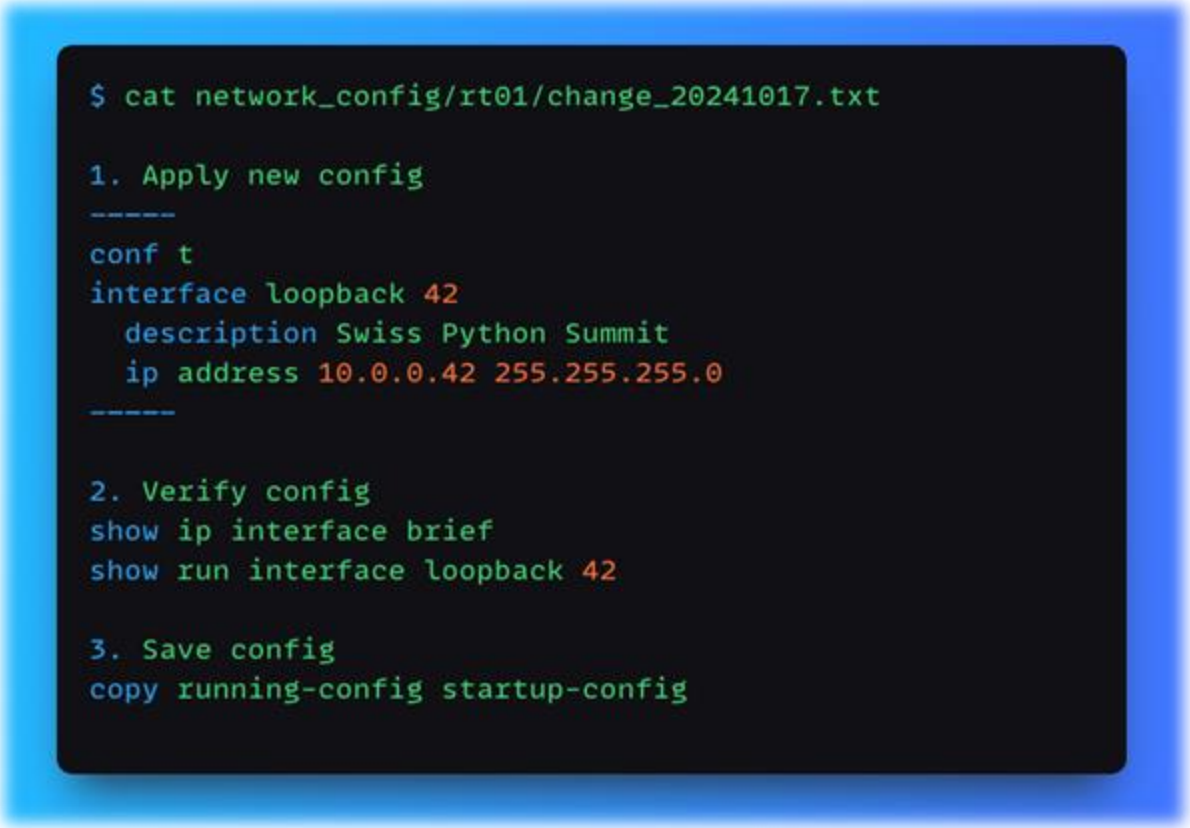
- Prepare config changes in text files



```
$ tree network_config
network_config
├── rt01
│   └── change_20241017.txt
├── rt02
│   └── change_20241017.txt
├── sw01
│   └── change_20241017.txt
├── sw02
│   └── change_20241017.txt
└── sw03
    └── change_20241017.txt
```

Traditional Network Change Workflow

- Config file content



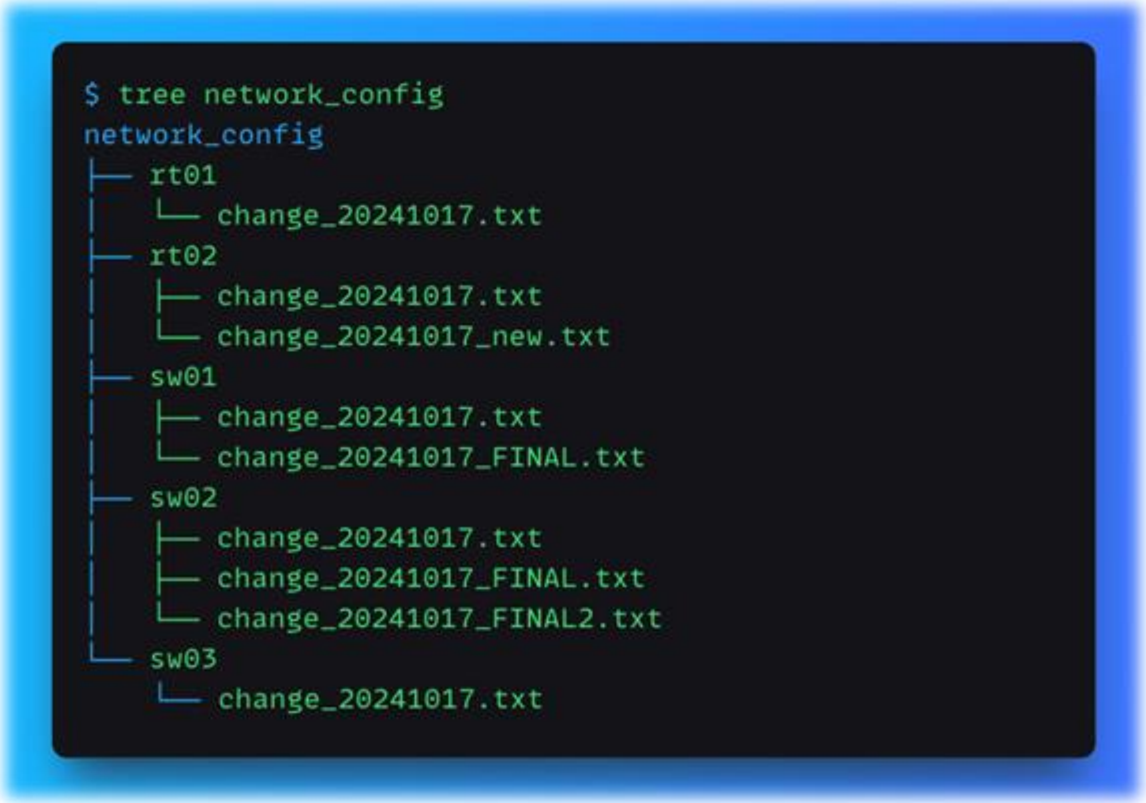
```
$ cat network_config/rt01/change_20241017.txt

1. Apply new config
------
conf t
interface loopback 42
  description Swiss Python Summit
  ip address 10.0.0.42 255.255.255.0
------

2. Verify config
show ip interface brief
show run interface loopback 42

3. Save config
copy running-config startup-config
```

Traditional Network Change Workflow

- Prepare config changes in text files
- Working hours
  - Best case → 21:00 – 22:00

Traditional Network Change Workflow

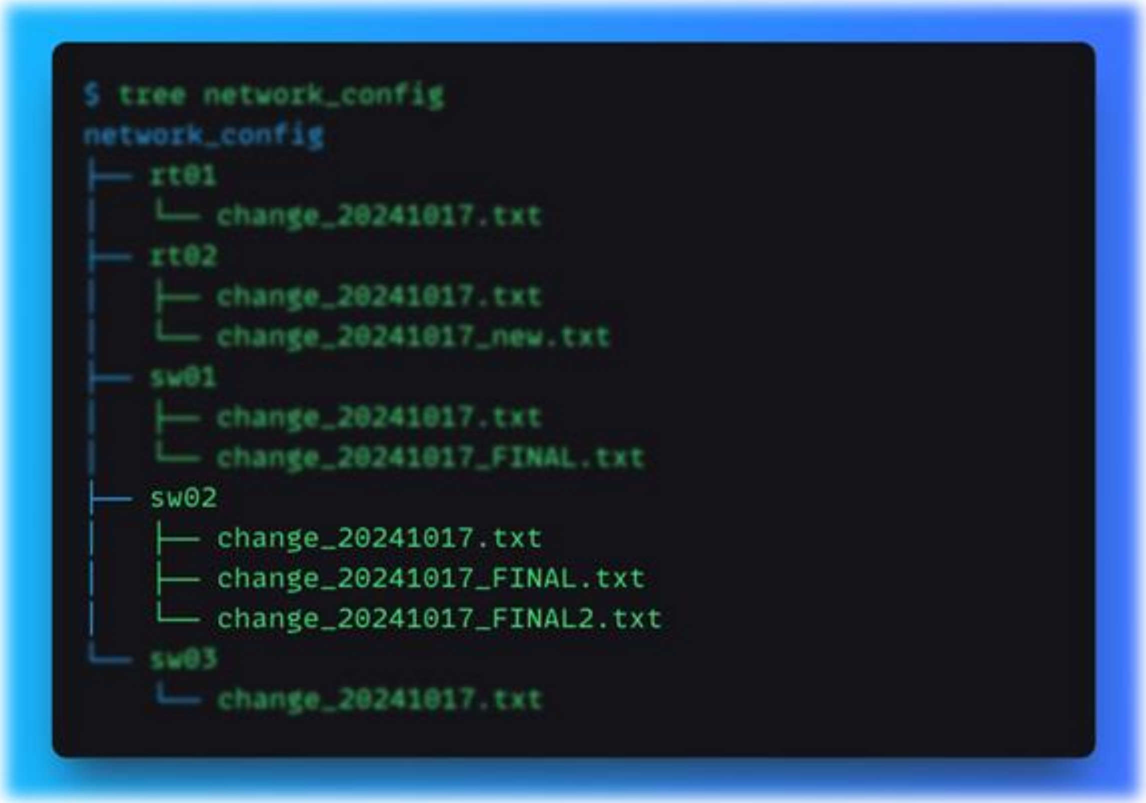
- Prepare config changes in text files



```
$ tree network_config
network_config
├── rt01
│   └── change_20241017.txt
├── rt02
│   ├── change_20241017.txt
│   └── change_20241017_new.txt
├── sw01
│   ├── change_20241017.txt
│   └── change_20241017_FINAL.txt
├── sw02
│   ├── change_20241017.txt
│   ├── change_20241017_FINAL.txt
│   └── change_20241017_FINAL2.txt
└── sw03
    └── change_20241017.txt
```

Traditional Network Change Workflow

- Prepare config changes in text files

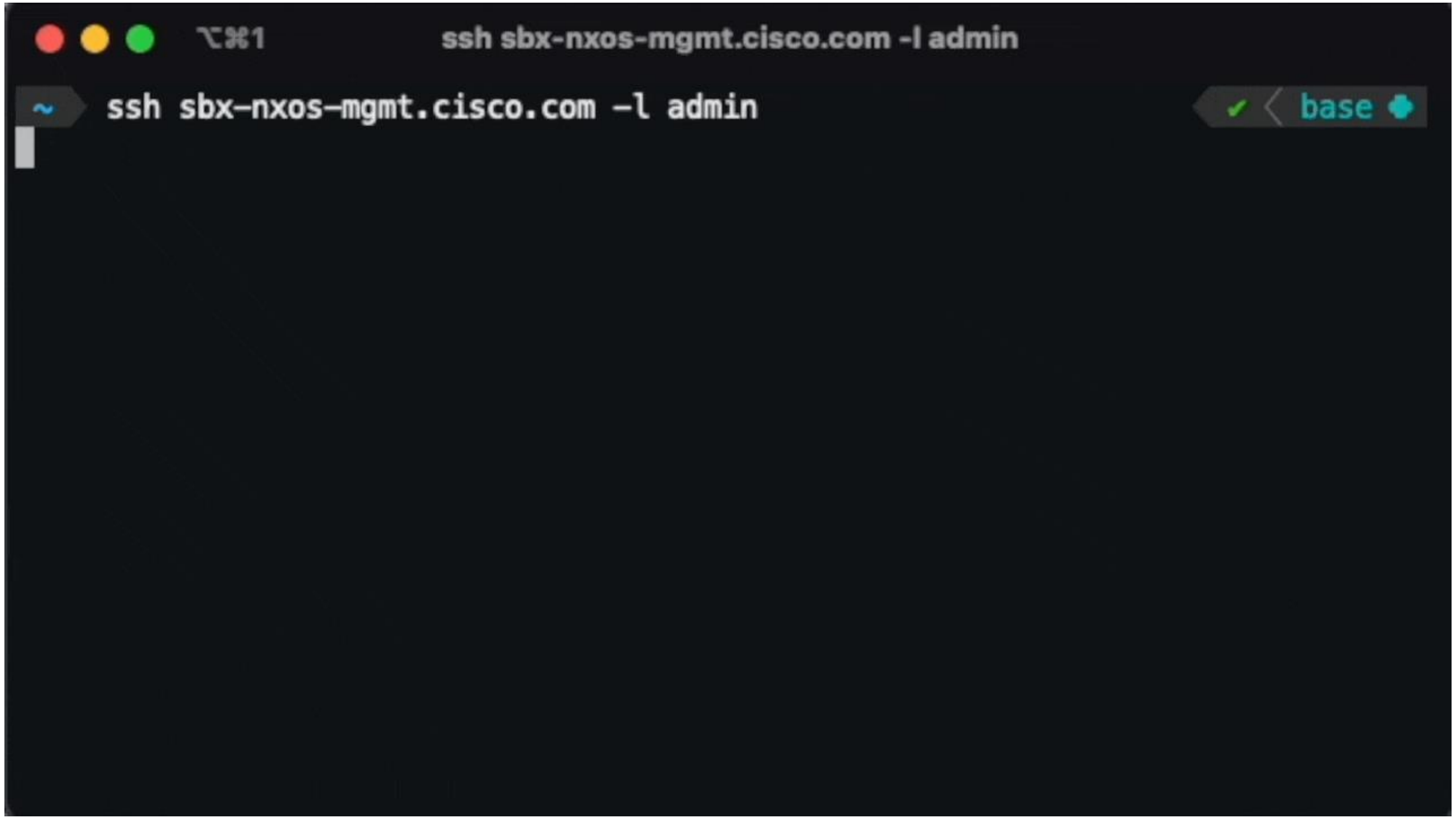


```
$ tree network_config
network_config
├── rt01
│   └── change_20241017.txt
├── rt02
│   ├── change_20241017.txt
│   └── change_20241017_new.txt
├── sw01
│   ├── change_20241017.txt
│   └── change_20241017_FINAL.txt
├── sw02
│   ├── change_20241017.txt
│   ├── change_20241017_FINAL.txt
│   └── change_20241017_FINAL2.txt
└── sw03
    └── change_20241017.txt
```

Traditional Network Change Workflow

- Prepare config changes in text files
- Working hours
  - Best case → 21:00 – 22:00
  - Worst case → 21:00 – 5:00

Manual Config Change



```
~ ssh sbx-nxos-mgmt.cisco.com -l admin
```

Challenges

- Human error
  - Copy paste error
  - Wrong config to wrong device
  - Completely forget a device
- Versioning

Benefits of Network Automation

- Consistency
- Efficiency
- Reduced Downtime
- Scalability

Word of Caution

What Is Netmiko?

Network Automation Tools

ANSIBLE

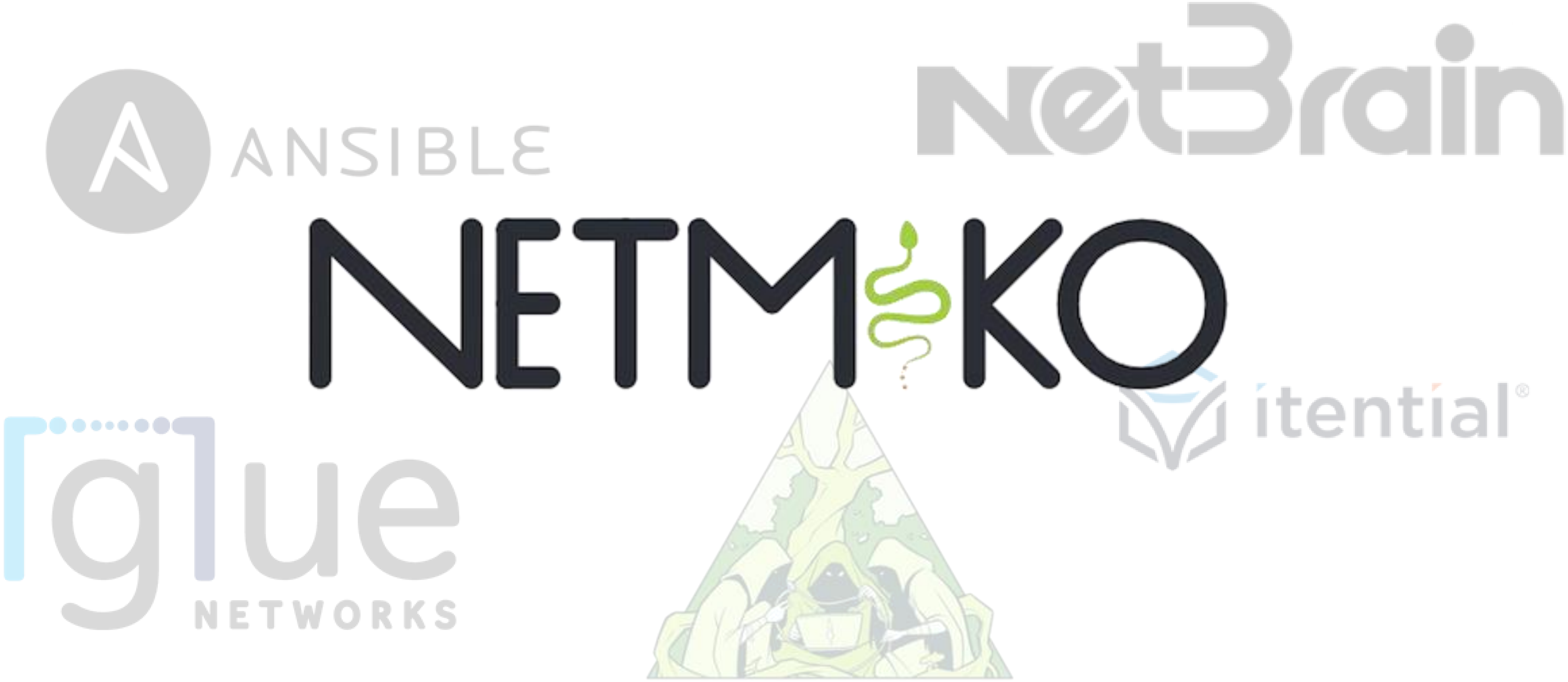
NetBrain

NETMIKO

itential®

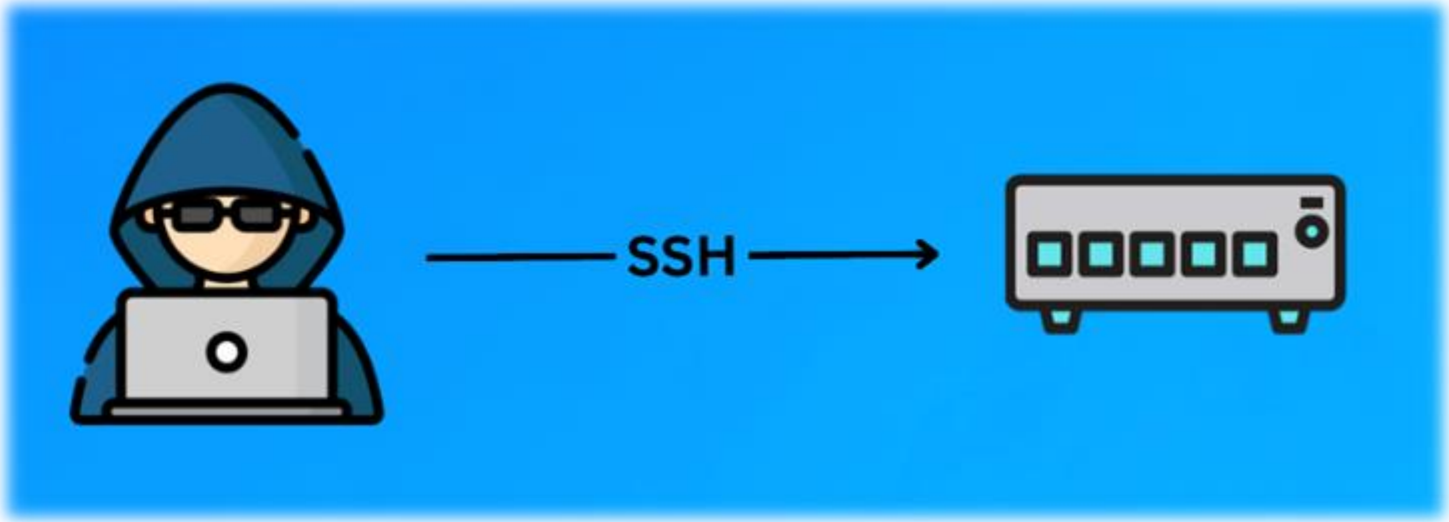
glue NETWORKS

Network Automation Tools



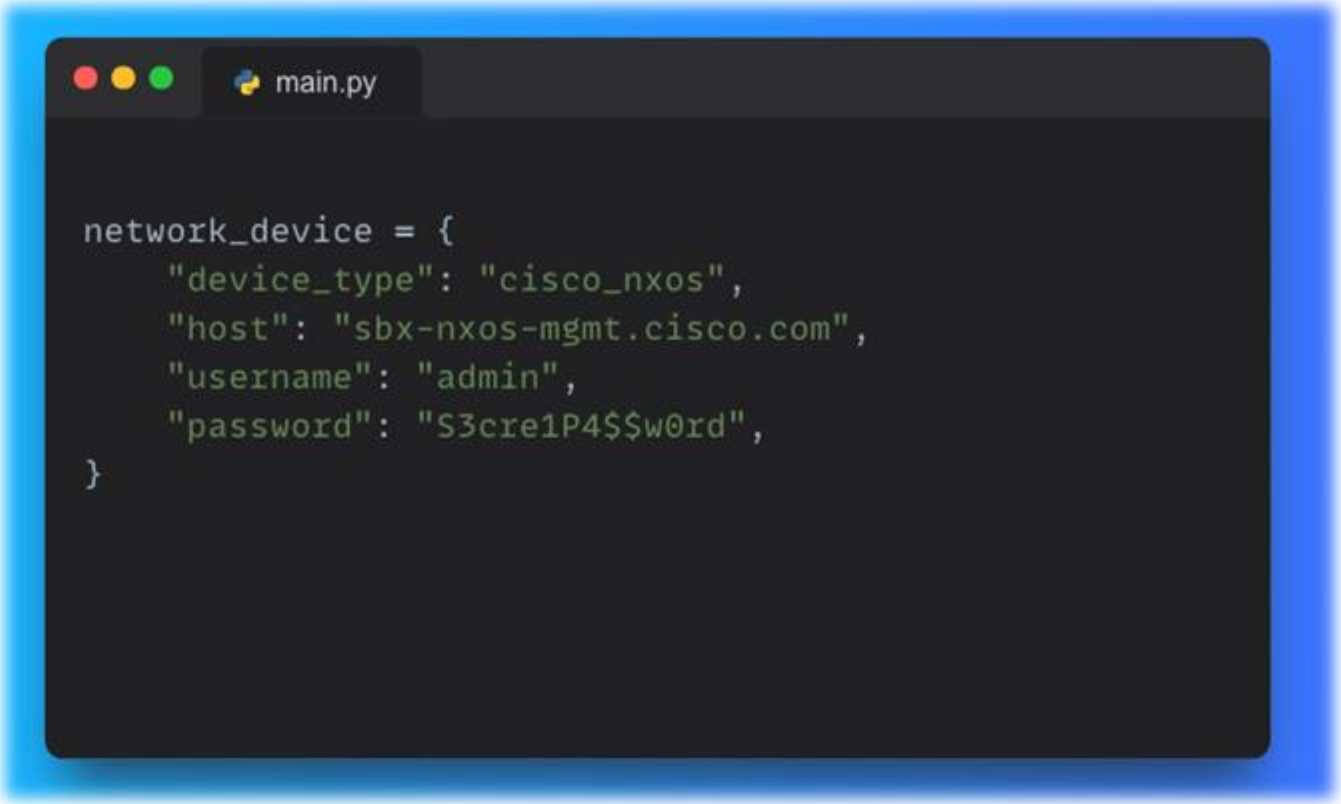
How It Works

- Establish SSH connection
- Configure something
- Verify config



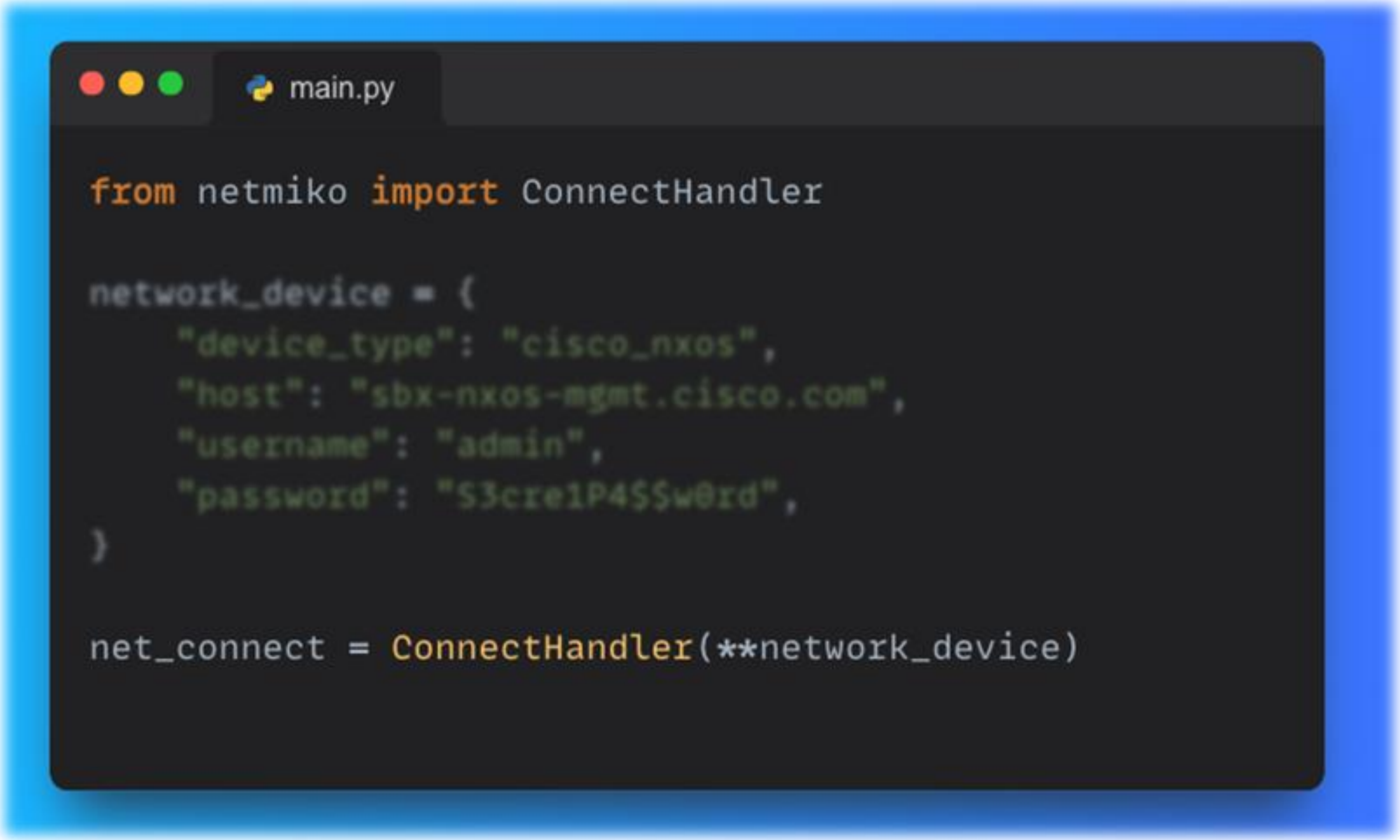
5 Steps to Automate Your Network

Step 1 – Define Your Inventory



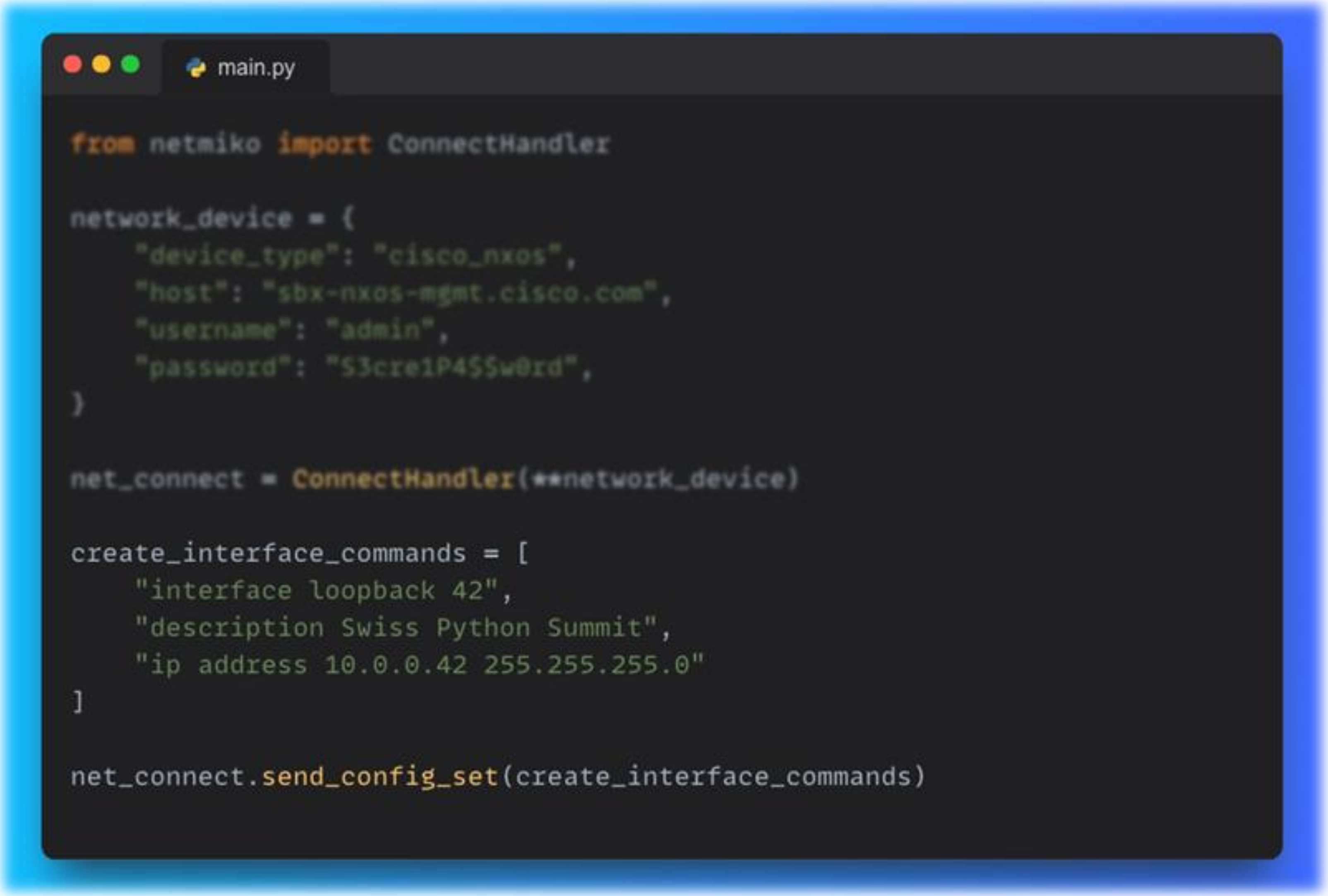
```python
network_device = {
    "device_type": "cisco_nxos",
    "host": "sbx-nxos-mgmt.cisco.com",
    "username": "admin",
    "password": "S3cre1P4$$w0rd",
}
```

Step 2 – Connect To Devices



```python
from netmiko import ConnectHandler

network_device = {
    "device_type": "cisco_nxos",
    "host": "sbx-nxos-mgmt.cisco.com",
    "username": "admin",
    "password": "S3cre1P4$$w0rd",
}

net_connect = ConnectHandler(**network_device)
```

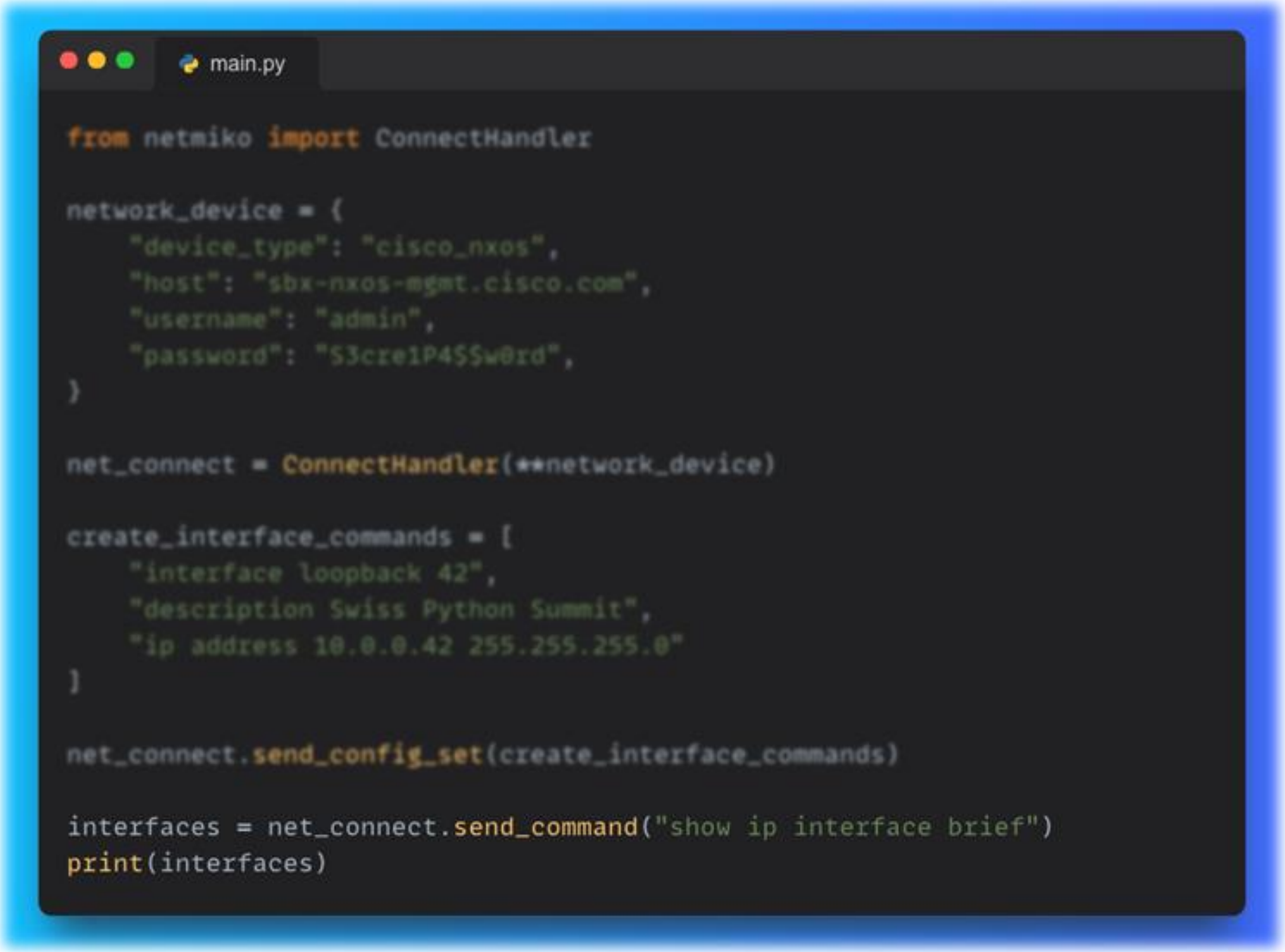
Step 3 – Send Config Commands



main.py

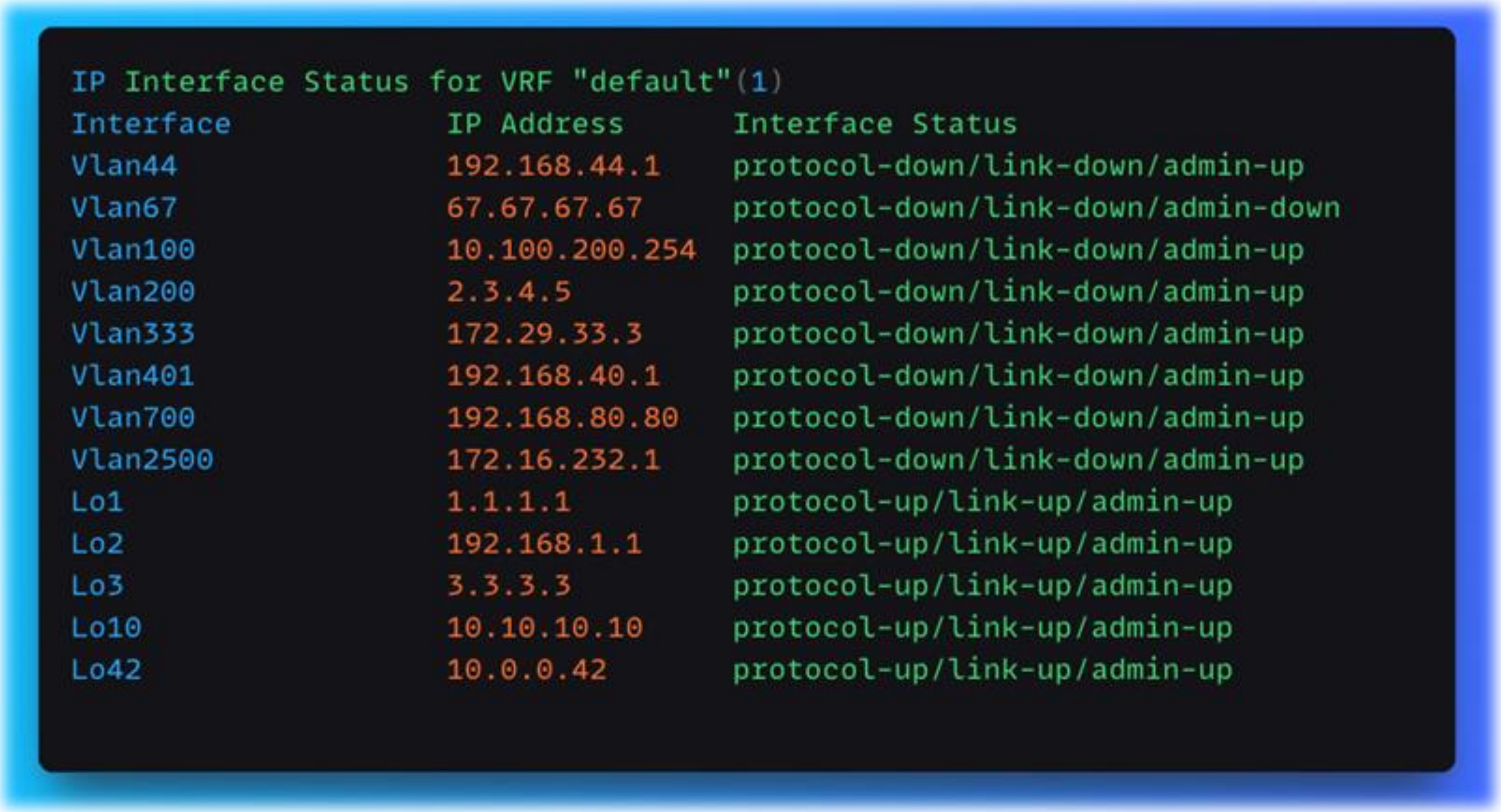
```python
from netmiko import ConnectHandler

network_device = {
    "device_type": "cisco_nxos",
    "host": "sbx-nxos-mgmt.cisco.com",
    "username": "admin",
    "password": "S3cre1P4$$w0rd",
}

net_connect = ConnectHandler(**network_device)

create_interface_commands = [
    "interface loopback 42",
    "description Swiss Python Summit",
    "ip address 10.0.0.42 255.255.255.0"
]

net_connect.send_config_set(create_interface_commands)
```

Step 4 – Send Show Commands



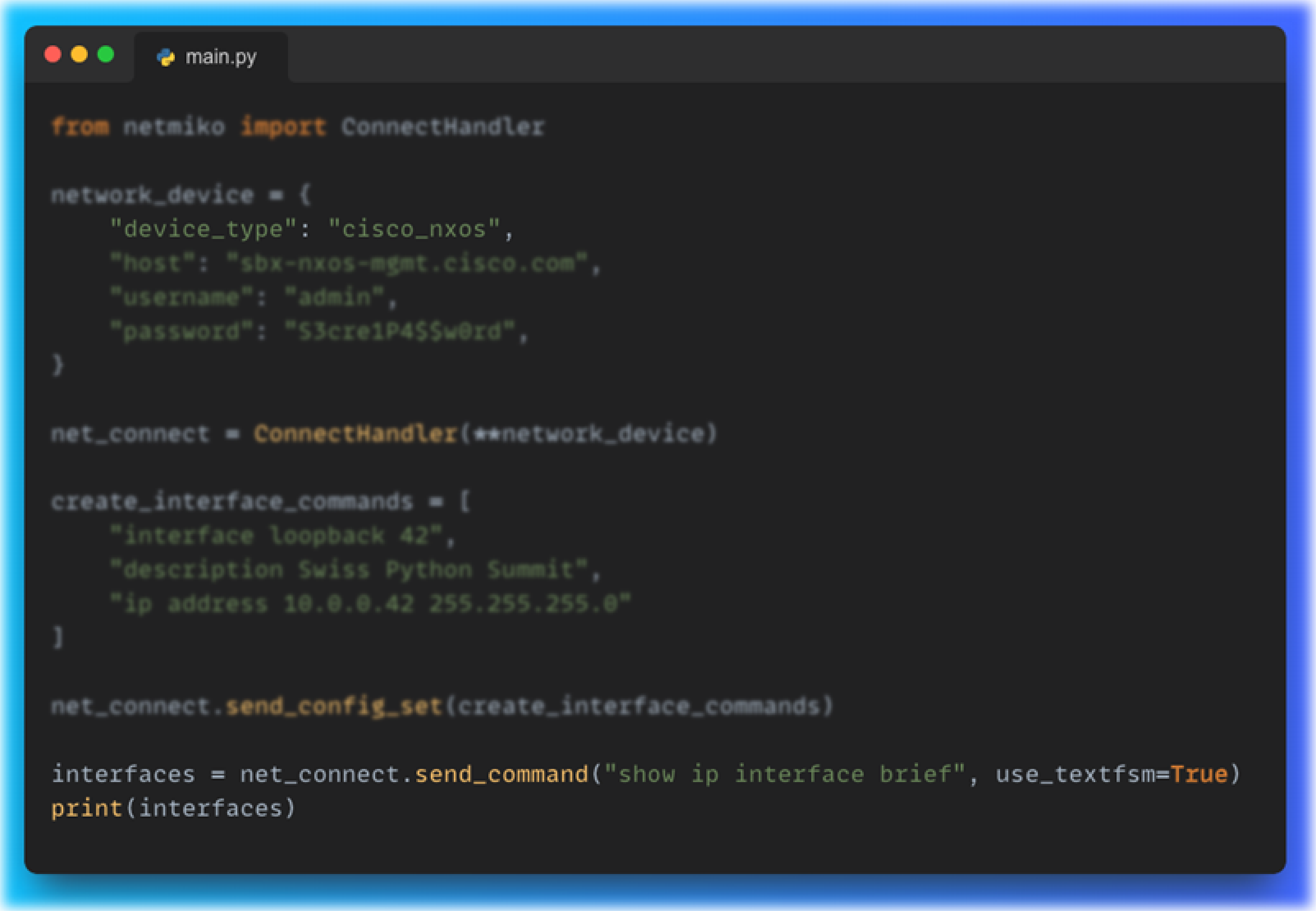
```python
from netmiko import ConnectHandler

network_device = {
    "device_type": "cisco_nxos",
    "host": "sbx-nxos-mgmt.cisco.com",
    "username": "admin",
    "password": "S3cre1P4$$w0rd",
}

net_connect = ConnectHandler(**network_device)

create_interface_commands = [
    "interface loopback 42",
    "description Swiss Python Summit",
    "ip address 10.0.0.42 255.255.255.0"
]

net_connect.send_config_set(create_interface_commands)

interfaces = net_connect.send_command("show ip interface brief")
print(interfaces)
```

Step 4 – Send Show Commands



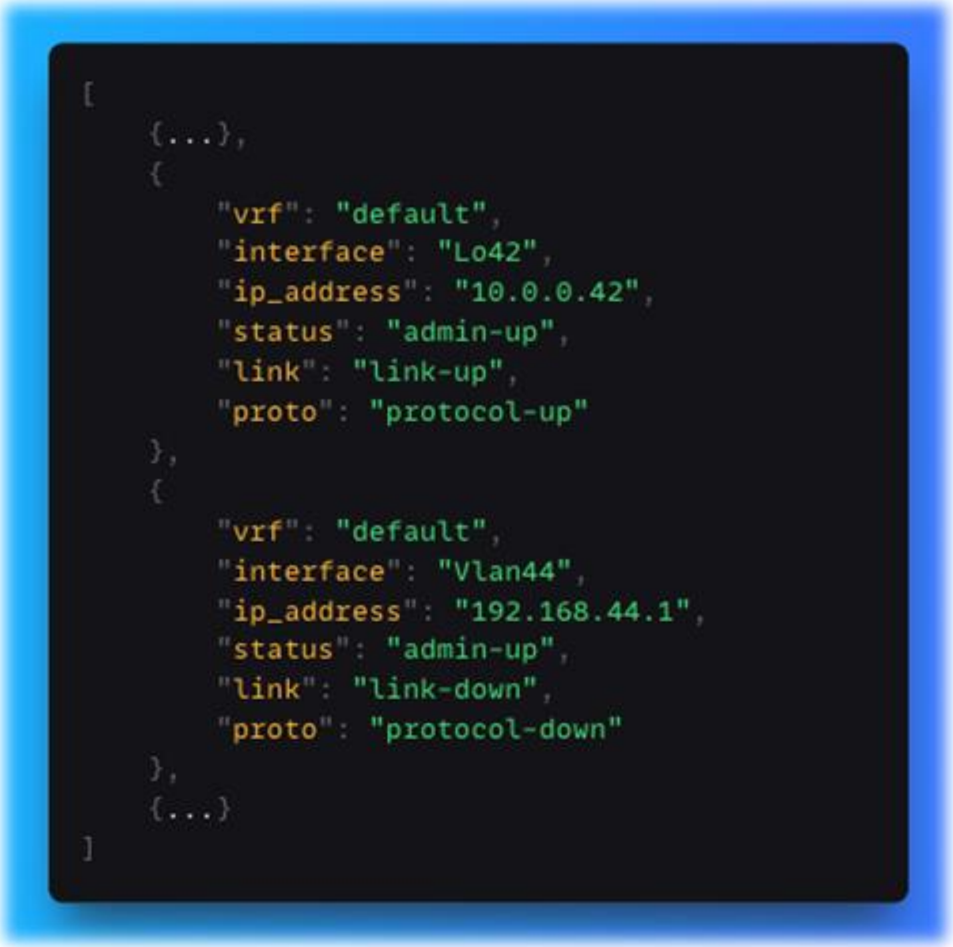
```
IP Interface Status for VRF "default"(1)
Interface            IP Address      Interface Status
Vlan44               192.168.44.1    protocol-down/link-down/admin-up
Vlan67               67.67.67.67     protocol-down/link-down/admin-down
Vlan100              10.100.200.254  protocol-down/link-down/admin-up
Vlan200              2.3.4.5         protocol-down/link-down/admin-up
Vlan333              172.29.33.3     protocol-down/link-down/admin-up
Vlan401              192.168.40.1    protocol-down/link-down/admin-up
Vlan700              192.168.80.80   protocol-down/link-down/admin-up
Vlan2500             172.16.232.1    protocol-down/link-down/admin-up
Lo1                  1.1.1.1         protocol-up/link-up/admin-up
Lo2                  192.168.1.1     protocol-up/link-up/admin-up
Lo3                  3.3.3.3         protocol-up/link-up/admin-up
Lo10                 10.10.10.10     protocol-up/link-up/admin-up
Lo42                 10.0.0.42       protocol-up/link-up/admin-up
```

Step 5 – Parse Output



```python
from netmiko import ConnectHandler

network_device = {
    "device_type": "cisco_nxos",
    "host": "sbx-nxos-mgmt.cisco.com",
    "username": "admin",
    "password": "S3cre1P4$$w0rd",
}

net_connect = ConnectHandler(**network_device)

create_interface_commands = [
    "interface loopback 42",
    "description Swiss Python Summit",
    "ip address 10.0.0.42 255.255.255.0"
]

net_connect.send_config_set(create_interface_commands)

interfaces = net_connect.send_command("show ip interface brief", use_textfsm=True)
print(interfaces)
```

Step 5 – Parse Output

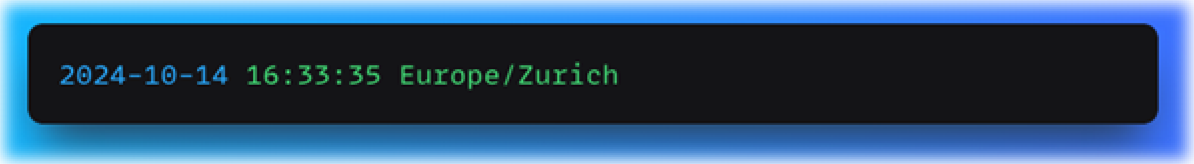


```
[
    {...},
    {
        "vrf": "default",
        "interface": "Lo42",
        "ip_address": "10.0.0.42",
        "status": "admin-up",
        "link": "link-up",
        "proto": "protocol-up"
    },
    {
        "vrf": "default",
        "interface": "Vlan44",
        "ip_address": "192.168.44.1",
        "status": "admin-up",
        "link": "link-down",
        "proto": "protocol-down"
    },
    {...}
]
```

What Is TextFSM

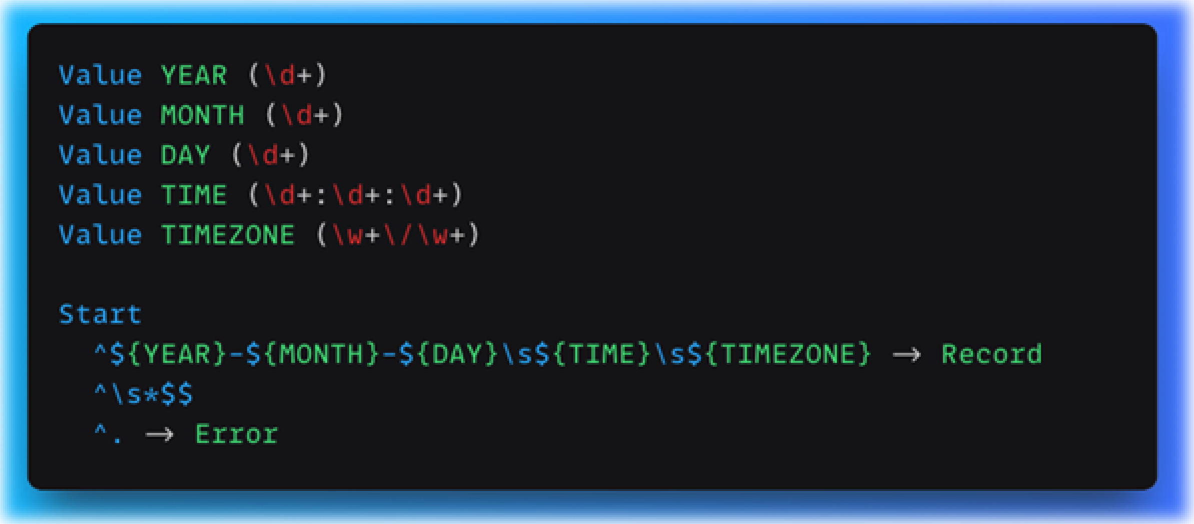
- Open-source tool developed by Google
- Parses semi-structured text
- Access information from CLI of network devices
- Template + raw input = parsed output

Example TextFSM Template



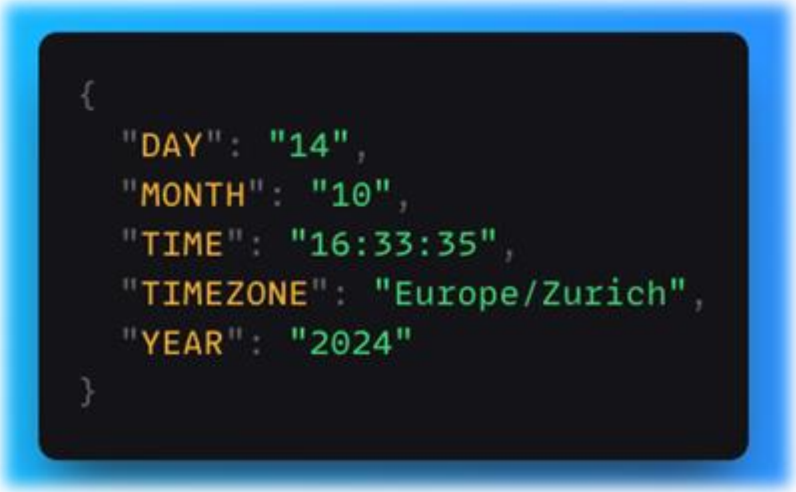
```
2024-10-14 16:33:35 Europe/Zurich
```

+



```
Value YEAR (\d+)
Value MONTH (\d+)
Value DAY (\d+)
Value TIME (\d+:\d+:\d+)
Value TIMEZONE (\w+\/\w+)

Start
  ^${YEAR}-${MONTH}-${DAY}\s${TIME}\s${TIMEZONE} -> Record
  ^\s*$$
  ^. -> Error
```

=



```
{
  "DAY": "14",
  "MONTH": "10",
  "TIME": "16:33:35",
  "TIMEZONE": "Europe/Zurich",
  "YEAR": "2024"
}
```

NTC Templates

- Repo of TextFSM templates
- 50+ vendors and 800+ templates

Conclusion & Outlook

What You’ve Learned

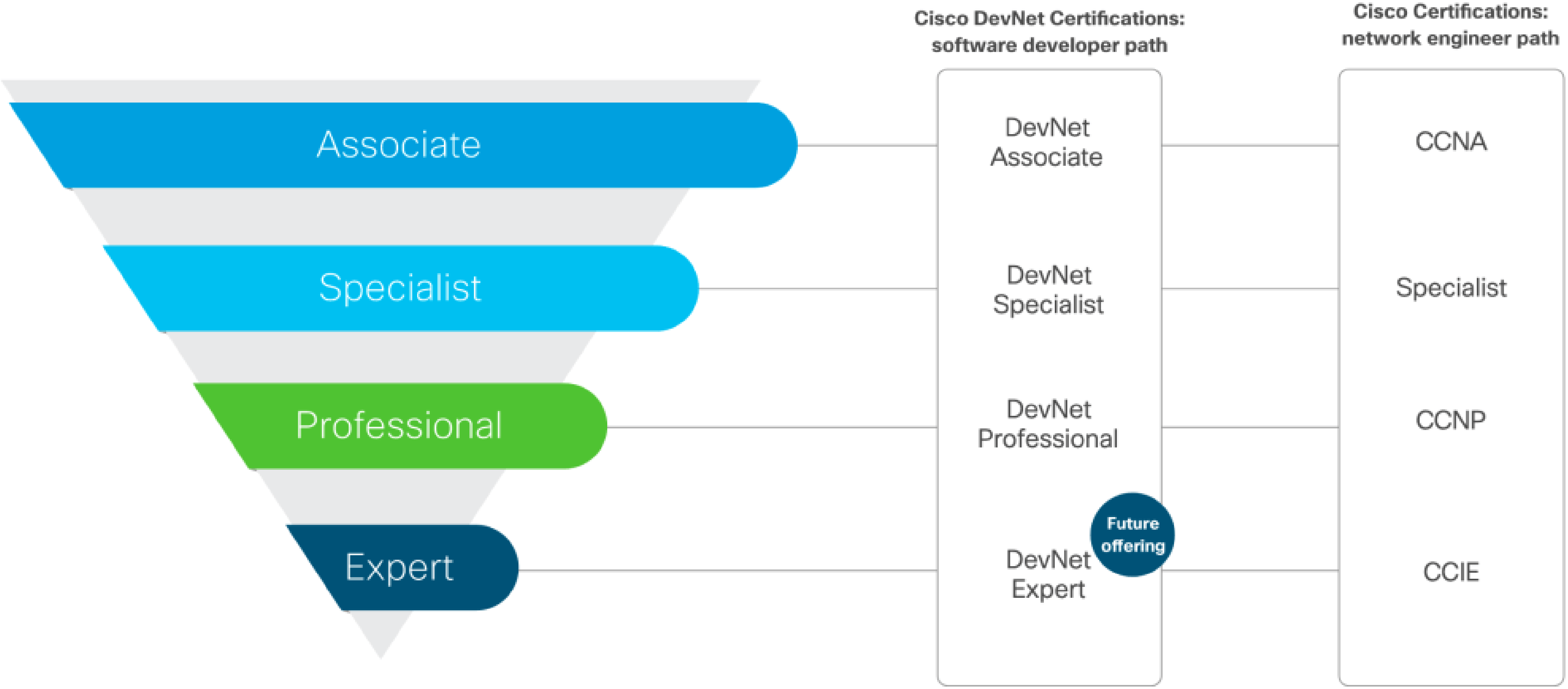
- Network automation fundamentals
- Netmiko basics
- Parsing with TextFSM templates

Where to Go Next

- Use external inventory sources (e.g. NetBox)
- Jinja templates instead of hardcoded commands
- Multithreading or parallel programming

Next Steps

Cisco Certifications



	Cisco DevNet Certifications: software developer path	Cisco Certifications: network engineer path
Associate	DevNet Associate	CCNA
Specialist	DevNet Specialist	Specialist
Professional	DevNet Professional	CCNP
Expert	DevNet Expert Future offering	CCIE

DevNet Academy

Visit onway

- >30 employees
- 6’000 mobile routers
- ~500’000 daily WiFi sessions

THANKS