Low-Level Network Device Interactions

From “Mastering Python Networking - Eric Chou”

Agenda

- Why Netmiko & where it fits
- Netmiko basics: install, connect, show/config, multi-device
- Nornir overview: inventory, tasks, results
- Nornir + Netmiko plugin workflow
- CLI automation caveats and idempotency
- Best practices & next steps

What is Netmiko?

- Python library built on top of Paramiko
- Simplifies SSH to network devices (prompts, modes, paging)
- Great for quick, direct interactions with many vendors

Install Netmiko

- Use a **virtualenv** for isolation
- `pip install netmiko`
- Keep device credentials out of code

Minimal Netmiko Example

```
from netmiko import ConnectHandler

net = ConnectHandler(device_type="cisco_ios",
host="192.168.2.51", username="cisco",
password="cisco")

prompt = net.find_prompt()  # e.g., "lax-edg-r1#"
```

Show Commands with Netmiko

- `output = net.send_command("show ip int brief")`
- Netmiko handles terminal length and prompt syncing
- You get clean text output ready to parse

Configuration with Netmiko

- `cfg = ["logging buffered 19999"]`
- `result = net.send_config_set(cfg)`
- Netmiko enters/exits config mode for you

Multiple Devices Pattern

- `devices = [{"host": "192.168.2.51"}, {"host": "192.168.2.52"}, ...]`
- Loop: Connect → run show/config → collect output → disconnect
- Store results per device for later processing

Session & Prompt Handling

- find_prompt() confirms you"re at the expected mode
- send_command() vs send_config_set()
- Use enable() if your platform requires an enable secret

Troubleshooting Netmiko

- Mismatched device_type causes odd prompt/expect issues
- Slow/loaded devices? add_delay_factor or global_delay_factor
- Log sessions when debugging parsing problems

When to Choose Netmiko

- Need fast, imperative scripts
- Small-to-medium blast radius per run
- Great building block for frameworks (Ansible/Nornir)

Limitations of Raw CLI Automation

- Screen-scraped, unstructured output is fragile
- CLI formatting can change across versions
- Harder to be idempotent without extra checks

Netmiko: Quick Recap

- Simple API for SSH to network gear
- send_command() for show, send_config_set() for config
- Use loops to scale across many devices

Bridge to Nornir

- Netmiko solves "how to talk to one device" elegantly
- Nornir adds inventory, concurrency, results handling
- Together: scalable workflows with clean code

What is Nornir?

- Pure-Python automation framework
- Inventory + task runner + results model
- Pluggable ecosystem (e.g., nornir_netmiko, nornir_utils)

Install Nornir + Plugins

- `pip install nornir nornir_utils nornir_netmiko`
- Keep it inside a **virtualenv**
- Version-pin in requirements.txt for reproducibility

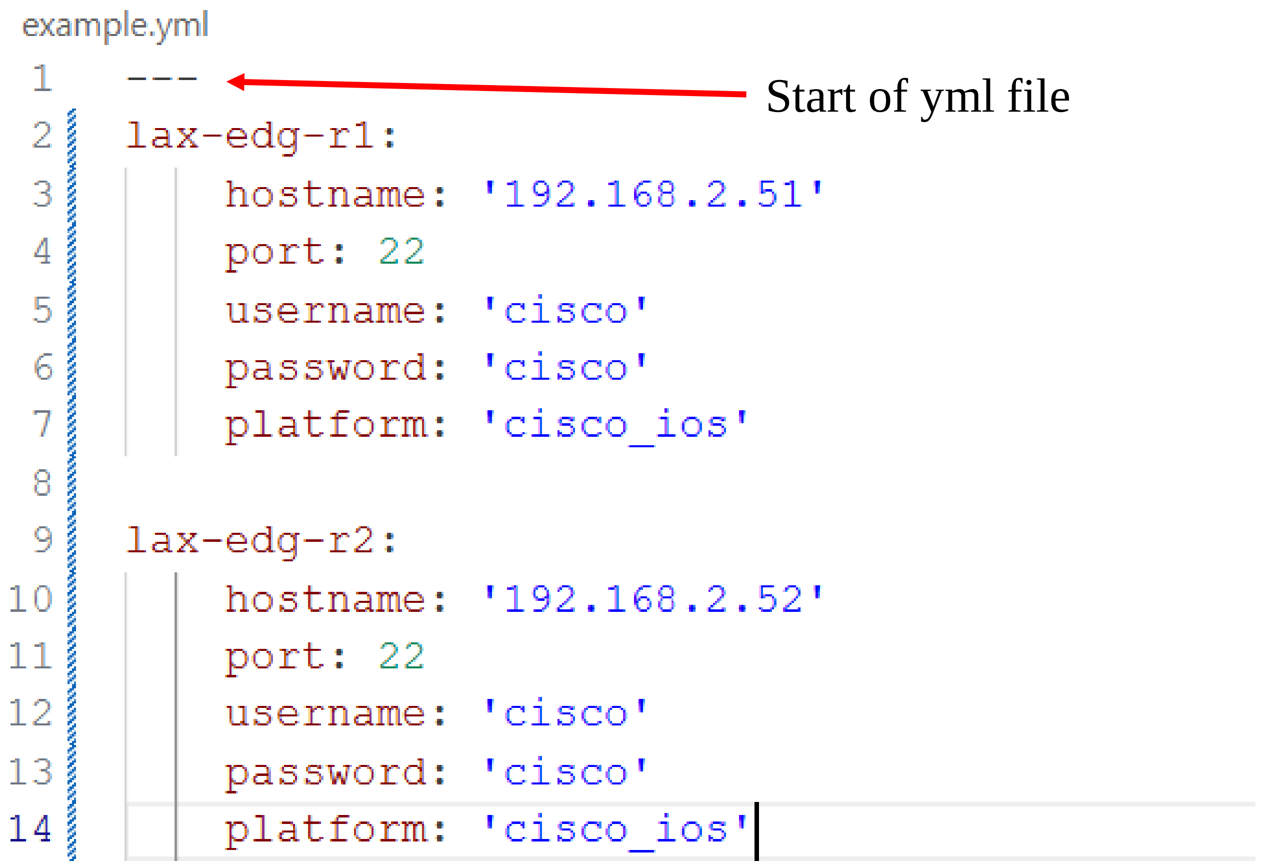
Inventory-First Mindset

- hosts.yaml: per-host hostname, port, username, password, platform
- Group/defaults files can hold shared parameters
- Credential separation keeps code clean

Sample hosts.yaml (Cisco IOS)

- Keep secrets

encrypted in practice



example.yml

```yaml
---
lax-edg-r1:
    hostname: '192.168.2.51'
    port: 22
    username: 'cisco'
    password: 'cisco'
    platform: 'cisco_ios'

lax-edg-r2:
    hostname: '192.168.2.52'
    port: 22
    username: 'cisco'
    password: 'cisco'
    platform: 'cisco_ios'
```

Start of yml file

Your First Nornir Script

```
from nornir import InitNornir

from nornir_netmiko import import
netmiko_send_command

result = nr.run(task=netmiko_send_command,
command_string="show arp")
```

Understanding Results

- `nornir_utils.print_result(result)` renders per-host outputs
- Structured per-host success/failure flags
- Easy to post-process for reporting

Configuration with Nornir + Netmiko

- Use netmiko_send_config to push config lines
- Run across inventory with one call
- Capture and log device responses uniformly

Scaling the Workflow

- Inventory drives scope; task defines the action
- Split tasks into functions for reuse
- Chain tasks (facts → decision → config)

Testing in a Lab

- Use virtual labs (CML, GNS3/EVE-NG) to iterate safely
- Target one or two devices first, then expand
- Keep blast radius small until validated

Extending with Plugins

- nornir_netmiko for SSH/CLI
- NAPALM plugin for higher-level getters/setters
- Pick the right tool per task

Operational Tips

- Isolate dependencies per project (virtualenv)
- Store inventory & code in version control (Git)
- Capture outputs to files/artifacts for auditing

Idempotency Matters

- Repeatable runs should yield the same outcome
- Guardrail checks before/after changes
- Prefer structured state when available

CLI Caveats & Risk

- Unstructured output = brittle parsers
- CLI differences across images/versions
- Bad automation can "go fast wrong"—limit scope, review, test

Wrap-Up & Next Steps

- Start small with Netmiko scripts
- Adopt Nornir for inventory-driven scale
- Look ahead to API-based approaches for structured data