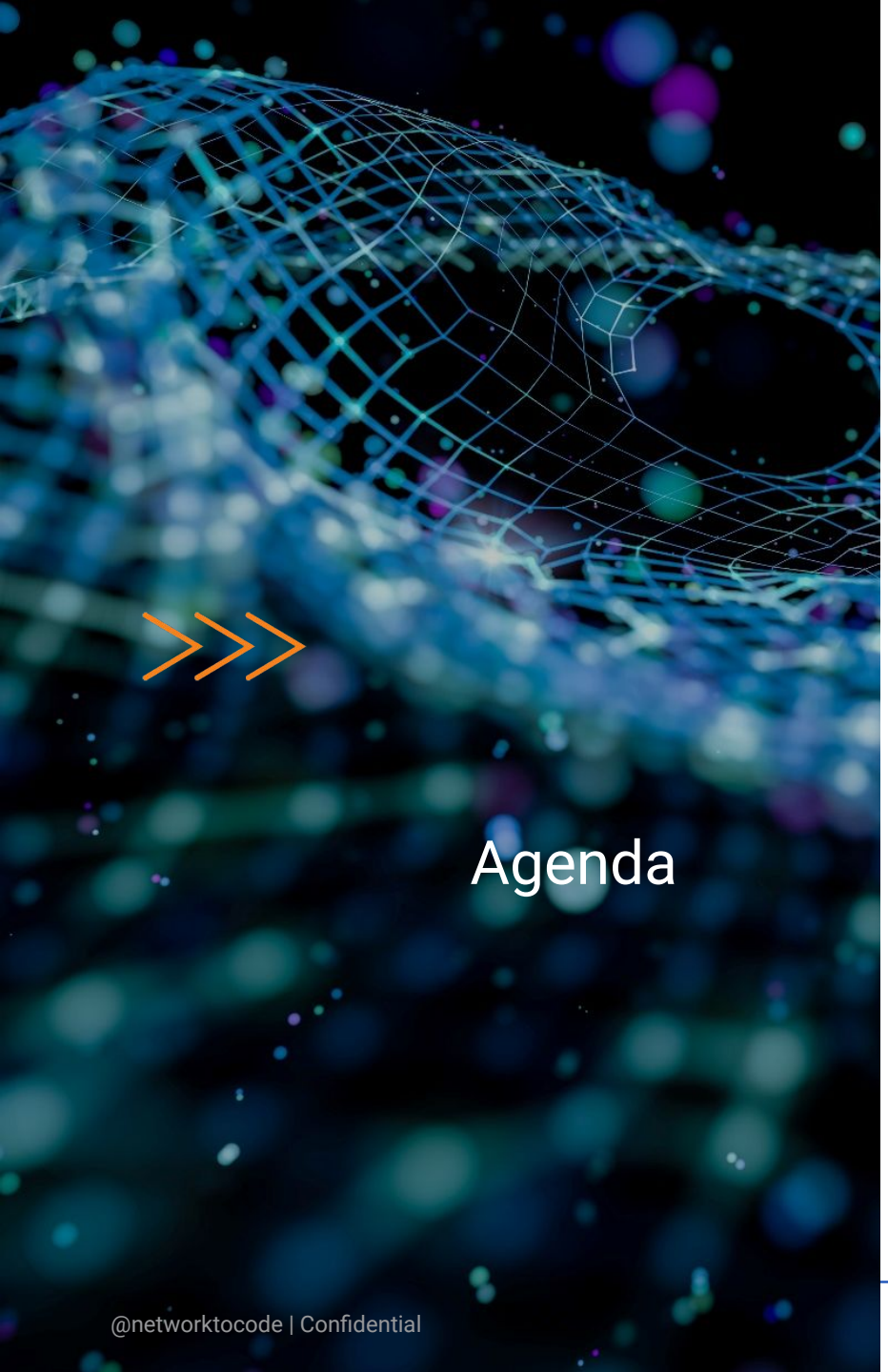
Introduction to Nornir

Workshop



Agenda

- **Introduction to Nornir**
- **Understanding the Nornir Inventory**
- **Using Filters with the Inventory**
  - **Lab 01** - Creating the Nornir Inventory
  - **Lab 02** - Exploring the Nornir Inventory
  - **Lab 03** - Filtering the Inventory File
- **Nornir Plugins: Nautobot Inventory, NAPALM, Netmiko**
  - **Lab 04** - Getting Inventory Data Dynamically from Nautobot
  - **Lab 05** - Exploring NAPALM and Netmiko Task Modules to Gather Data
  - **Lab 06** - Exploring NAPALM and Netmiko Task Modules for Configuration Changes

>>> Housekeeping

Course Materials

You MUST be logged in to your GitHub account to view

https://github.com/ntc-training/workshop-introduction-nornir

- Lectures and Demos - **Timing**
- **Breaks**
- Feedback / **Q&A**

Part 1

Understanding Nornir and its Inventory

>>> Introducing Nornir

- **Nornir** is a **Python**-based, **multi-threaded**, automation framework with built-in **inventory** management.
- **Why?**
  - Avoids the limitations of Domain Specific Languages (DSL).
  - Use the full power of the Python language to work with complex data structures and build advanced workflows.
  - Much faster than more abstracted tools (like Ansible, Salt, Puppet etc.)
  - Native integration with other Python frameworks (like Flask).
  - Cleaner code that leverages great tooling like linters, debuggers, and the testing frameworks of the Python ecosystem.

>>> Getting Started with Nornir

- **Useful links**
  - Project page: https://nornir.tech/nornir/
  - Documentation: https://nornir.readthedocs.io/
- **Nornir** is a Python package
  - pip install nornir
- It also has additional functionality provided as **plugins**
  - Main list: https://nornir.tech/nornir/plugins/
  - Most of them can also be installed via pip (e.g. pip install nornir-napalm)

>>> The Nornir Inventory

- One of the most important pieces of Nornir
  - The **SimpleInventory** plugin uses YAML files that are parsed and used to create core inventory objects (within Python).
  - Other plugins for Ansible, Nautobot etc.
- Three Components
  - **hosts**: device info, connection details, user defined data, groups
  - **groups**: logical grouping of devices, data inheritance, scale
  - **defaults**: define data that applies to all hosts as a baseline

>>> A hosts YAML Example

```
cat ./files/inventory/simple_inventory_files/multi_devices.yml
```

```
---

csr1:
  hostname: 100.96.0.18
  username: ntc
  password: ntc123
  platform: cisco_ios
csr2:
  hostname: 100.96.0.20
  username: ntc
  password: ntc123
  platform: cisco_ios
csr3:
  hostname: 100.96.0.22
  username: ntc
  password: ntc123
  platform: cisco_ios
```

>>> Adding custom data

```
cat ./files/inventory/simple_inventory_files/multi_devices_with_data.yml
```

```
---

csr1:
  hostname: 100.96.0.18
  username: ntc
  password: ntc123
  platform: cisco_ios
  data:
    snmp_ro: ntc_course

csr2:
  hostname: 100.96.0.20
  username: ntc
  password: ntc123
  platform: cisco_ios
  data:
    snmp_ro: ntc_course
```

>>> A group definition example

```
---

iosxe:
  platform: cisco_ios
  data:
    snmp_location: NYC

nxos:
  platform: cisco_nxos
  data:
    snmp_location: Orlando
```

```
csr3:
  hostname: 100.96.0.22
  username: ntc
  password: ntc123
  platform: cisco_ios
  data:
    snmp_ro: ntc_course
  groups:
    - iosxe

nxos-spine1:
  hostname: 100.96.0.14
  username: ntc
  password: ntc123
  platform: cisco_nxos
  data:
    snmp_ro: ntc_course
  groups:
    - nxos
```

>>> Adding defaults

```
cat ./files/inventory/defaults/defaults.yml
```

```
---

username: ntc
password: ntc123
data:
  snmp_ro: ntc_course
```

>>> Tie it all together - Nornir Configuration

- To initialize Nornir, you need configuration.
  - From a file.
  - Directly in the code.
  - A combination of files and code.
- Code takes precedence over configuration loaded from files.

>>> YAML Configuration File

```yaml
#config.yaml
---

inventory:
 plugin: SimpleInventory
 options:
   host_file: "hosts.yml"
   group_file: "groups.yml"
   defaults_file: "defaults.yml"

runner:
 plugin: threaded
 options:
   num_workers: 100
```

```python
from nornir import InitNornir

nr = InitNornir(config_file="config.yaml")
```

>>> Configuration as Code

```
from nornir import InitNornir

nr = InitNornir(
    runner={
        "plugin": "threaded",
        "options": {
            "num_workers": 100,
        },
    },
    inventory={
        "plugin": "SimpleInventory",
        "options": {
            "host_file": "hosts.yaml",
            "group_file": "groups.yaml"
        },
    },
)
```

>>> Hybrid Configuration

```
from nornir import InitNornir

nr = InitNornir(
    config_file="config.yaml",
    runner={
        "plugin": "threaded",
        "options": {
            "num_workers": 50,
        },
    },
)
```

>>> Explore the Inventory

- **Demo time!**
  - `nr.inventory`
  - `nr.inventory.hosts`
  - `nr.inventory.groups`
  - `nr.inventory.defaults`

>>> Nornir Filtering

- Large inventories require a way to select hosts based on custom criteria (platform, site, location etc.)
- Nornir provides powerful filtering functionality:
  - Simple filtering (based on key/value pairs).
  - Advanced filtering (logical operations).
  - Function based filtering (fully custom code).
- Filtering works on both well-known and user-defined data.

>>> Nornir Filtering Examples

- Simple Filtering
  - `nr.filter(role="leaf")`
  - `nr.filter(role="border", region="EU")`
- Advanced Filtering
  - `nr.filter(F(region="EU") | F(region="US"))`
  - `nr.filter(F(role="border") & ~F(region="US"))`
  - works on nested data and can call “magic” methods
    - `F(boot__image__startswith="9.3")`


LAB TIME
Labs: 01-03

Part 2

Using Plugins to Manage Inventory and Interact with Network Devices

>>> Nornir Plugins

- Plugins provide Nornir with extra functionality:
  - Additional inventory sources (e.g. Ansible, Nautobot, NetBox, IP Fabric, LibreNMS etc.).
  - Loading and printing data (nornir_utils).
  - Connecting and interacting with network devices
    - NAPALM, Netmiko, Scrapli, PyEZ, Netconf etc.

>>> Nornir - Nautobot

- https://docs.nautobot.com/projects/nornir-nautobot/en/latest/
- The **Inventory** plugin is used to gather device data from a Nautobot instance. This queries the DCIM endpoint to gather information about the devices.
  - Use all of the Django REST Framework filters via the “filter_parameters” configuration option.
  - https://docs.nautobot.com/projects/core/en/latest/rest-api/filtering/
- It also provides processor and task plugins to help with building automation workflows within Nautobot Apps.

>>> Nornir - NAPALM

- https://github.com/nornir-automation/nornir_napalm
- Uses the NAPALM library to provide the following:
  - Connect via NAPALM to devices.
  - **napalm_cli**
  - **napalm_configure**
  - **napalm_get** - call **get_*** methods
  - **napalm_ping**
  - **napalm_validate**

>>> Nornir - Netmiko

- https://github.com/ktbyers/nornir_netmiko
- Uses the Netmiko library to provide the following:
  - Connect via Netmiko (enhanced SSH) to devices.
  - **netmiko_file_transfer**
  - **netmiko_save_config**
  - **netmiko_send_command**
  - **netmiko_send_config**
  - **netmiko_commit**


LAB TIME!
Labs: 04-06