

PYTHON

FOR NETWORK ENGINEERS

Onsite Training Session
July 2020

Day4 Schedule



P Y T H O N
FOR NETWORK ENGINEERS

- Jinja2 Templating
- Deploying Jinja2 Generated Configurations
- Integrating to the operating system with subprocess (optional)
- Pulling data from a CSV file (optional)
- Intro to Unit Testing and CI/CD

- Nornir

Variables



Templates

Output Files



Jinja2 Templating



```
import jinja2

my_dict = {"a": "whatever"}

my_template = """
some
text
of
something
{{ a }}
something
"""

t = jinja2.Template(my_template)
print(t.render(my_dict))
```

Reference Material in:

`{{ github_repo }}/jinja2_example/jinja2_simple.py`
`{{ github_repo }}/jinja2_example/jinja2_bgp.py`



```
some
text
of
something
whatever
something
```

Jinja2 Templating

Loading Template from a File



```
import jinja2


template_file = "juniper_bgp.j2"
with open(template_file) as f:
    bgp_template = f.read()

my_vars = {
    "peer_as": "22",
    "neighbor1": "10.10.10.2",
    "neighbor2": "10.10.10.99",
    "neighbor3": "10.10.10.220",
}

template = jinja2.Template(bgp_template)
print(template.render(my_vars))
```

Reference Material in:

`{{ github_repo }}/jinja2_example/jinja2_bgp_file.py`



```
protocols {
  bgp {
    group external-peers {
      type external;
      peer-as 22;
      neighbor 10.10.10.2;
      neighbor 10.10.10.99;
      neighbor 10.10.10.220;
    }
  }
}
```

Reference Material in:

{{ github_repo }}/jinja2_example/jinja2_env.py

Jinja2 Template - Environment

```
from jinja2 import FileSystemLoader, StrictUndefined
from jinja2.environment import Environment

env = Environment(undefined=StrictUndefined)
env.loader = FileSystemLoader([".", "./templates/"])

my_vars = {"bgp_as": 22, "router_id": "1.1.1.1", "peer1": "10.20.30.1"}

template_file = "bgp_config.j2"
template = env.get_template(template_file)
output = template.render(**my_vars)
print(output)
```



Jinja2 Templating - Conditionals

```
interface GigabitEthernet0/0/0
  {%- if primary_ip %}
  ip address 10.220.88.22 255.255.255.0
  {%- endif %}
  negotiation auto
```

Jinja2 Templating - Loops



```
protocols {
  bgp {
    group external-peers {
      type external;
      {% for neighbor_ip, neighbor_as in my_list %}
        neighbor {{ neighbor_ip }} {
          peer-as {{ neighbor_as }};
        }
      {% endfor %}
    }
  }
}
```

Reference Material in:

{{ github_repo }}/jinja2_example/jinja2_bgp_loop.py

Jinja2 - Other Topics



- Jinja2 Whitespace Stripping
- Jinja2 Create Variables
- Jinja2 Filters
- Jinja2 Macros
- Jinja2 Includes / Hierarchy

Exercises:

`./day4/jinja2/ex1.txt`

`./day4/jinja2/ex2.txt`

`./day4/jinja2/ex3.txt`



Jinja2 - Deploying Generated Configurations

```
def render_configs(j2_env, template_file, my_vars):  
    # Render configs  
    template = j2_env.get_template(template_file)  
    config_section = template.render(**my_vars)  
    return config_section  
  
def load_configs(host, config, junos_format="text"):  
  
    # PyEZ connection  
    a_device = Device(host=host, user=USER, password=PASSWORD)  
    a_device.open()  
    a_device.timeout = 90  
  
    # Load config object  
    cfg = Config(a_device)  
    cfg.load(config, format=junos_format, merge=True)  
    return cfg
```

Reference Material in:
{{ github_repo }}/jinja2_deploy/



P Y T H O N
FOR NETWORK ENGINEERS

Subprocess - *Integrating to the System Operating System*

```
import os

print()
print("Current working directory")
start_dir = os.getcwd()
print(os.getcwd())

print()
print("Path of module we are executing")
print(os.path.realpath(__file__))

print()
print("Is this a file?")
print(os.path.isfile(__file__))

print()
print("Change directory into /tmp")
os.chdir("/tmp")
print(os.getcwd())
```



P Y T H O N
FOR NETWORK ENGINEERS

Subprocess - *Integrating to the System Operating System*

```
import subprocess

def subprocess_wrapper(cmd_list):
    """Wrapper to execute subprocess including byte to UTF-8 conversion."""
    proc = subprocess.Popen(cmd_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    std_out, std_err = proc.communicate()
    (std_out, std_err) = [x.decode("utf-8") for x in (std_out, std_err)]

    return (std_out, std_err, proc.returncode)

cmd_list = ["ls", "-a", "-l"]
std_out, std_err, return_code = subprocess_wrapper(cmd_list)
print()
print(f"Return Code: {return_code}")
print(std_out)
print()
```

Exercises:

`./day4/subprocess/ex1.txt`

Reference Material in:

`{{ github_repo }}/subprocess_example`



P Y T H O N
FOR NETWORK ENGINEERS

CSV Examples

Reference Material in:
[{{ github_repo }}/csv_example](#)

```
-----  
device_name,device_type,host,username,password  
pynet-rtr1,cisco_ios,184.105.247.70,pyclass,my_pass  
pynet-rtr2,cisco_ios,184.105.247.71,pyclass,my_pass  
-----
```

```
file_name = "test_net_devices.csv"  
with open(file_name) as f:  
    read_csv = csv.DictReader(f)  
    for entry in read_csv:  
        print(entry)
```

If it doesn't happen automatically; it didn't happen.



GitLab



Travis CI



Azure Pipelines



circleci

Continuous Integration using GitHub Actions.

Define a
.[github/workflows/commit.yaml](#)

Specify Events

Setup Environment

Install Dependencies

Add linting/tests

```
name: build
on: [push, pull_request]
jobs:
  std_tests:
    runs-on: ubuntu-latest
    strategy:
      max-parallel: 3
      matrix:
        python-version: [3.6, 3.7, 3.8]

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Setup Python ${ matrix.python-version }
        uses: actions/setup-python@v2
        with:
          python-version: ${ matrix.python-version }

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run black
        run: |
          black --check .
```

Day5 Schedule - Nornir



- Nornir Overview
- Inventory
- Nornir Tasks
- Results
- Nornir and Networking Plugins
- Custom Tasks and Grouped Tasks
- Nornir + Jinja2
- Using Netmiko and NAPALM Directly in Tasks
- Failed Tasks
- Debugging





P Y T H O N

FOR NETWORK ENGINEERS

Why was Nornir created? What problem is it trying to solve?

1. Systematic approach to inventory and data.
2. Concurrency
3. Using all Python



Why use a framework?

- * Systematic Inventory Management
- * Modular integration to other libraries
- * Integrated Concurrency
- * Systematizes automation in your organization

Ansible - Nornir Comparisons



Ansible Pluses

- + Easy getting started path
- + Use of SSH as a primary transport (familiarity)
- + Large community of network engineers using Ansible
- + Large organization behind it (Red Hat)



Nornir Pluses

- + All Python - Single, general purpose language
- + Easier debugging/troubleshooting
- + Use of Python Tool Chain (linters, debuggers, code testing)
- + Good performance
- + Tighter integrations to NAPALM and Netmiko

Ansible - Nornir Comparisons



Ansible Minuses

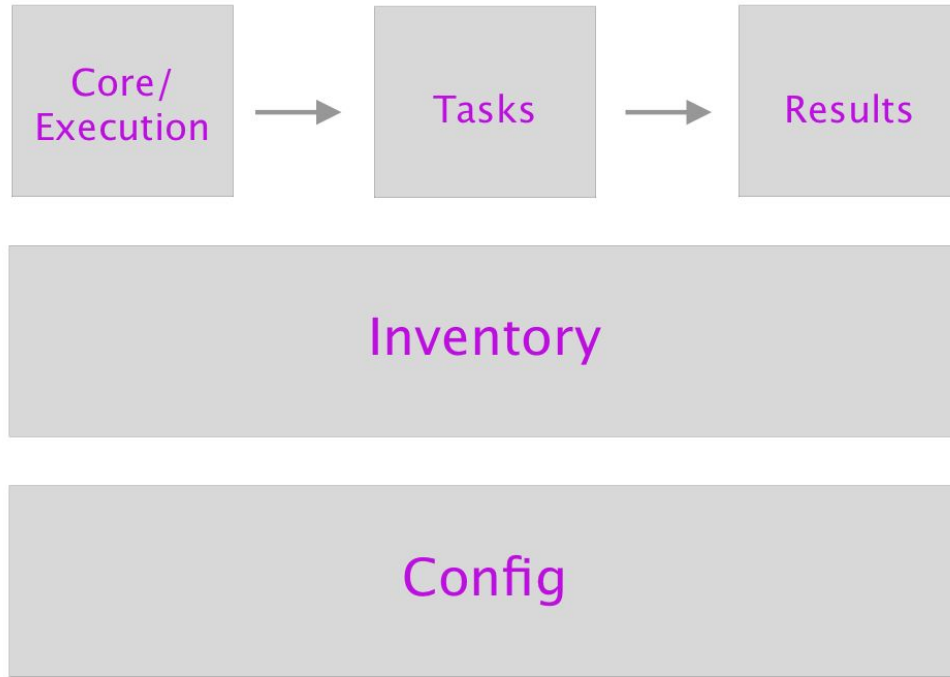
- Complex logic in Ansible is very painful
- Complex, nested data structures in Ansible are painful
- Troubleshooting can be unnecessarily difficult
- Easy things are easy, but somewhat difficult tasks get very hard quickly

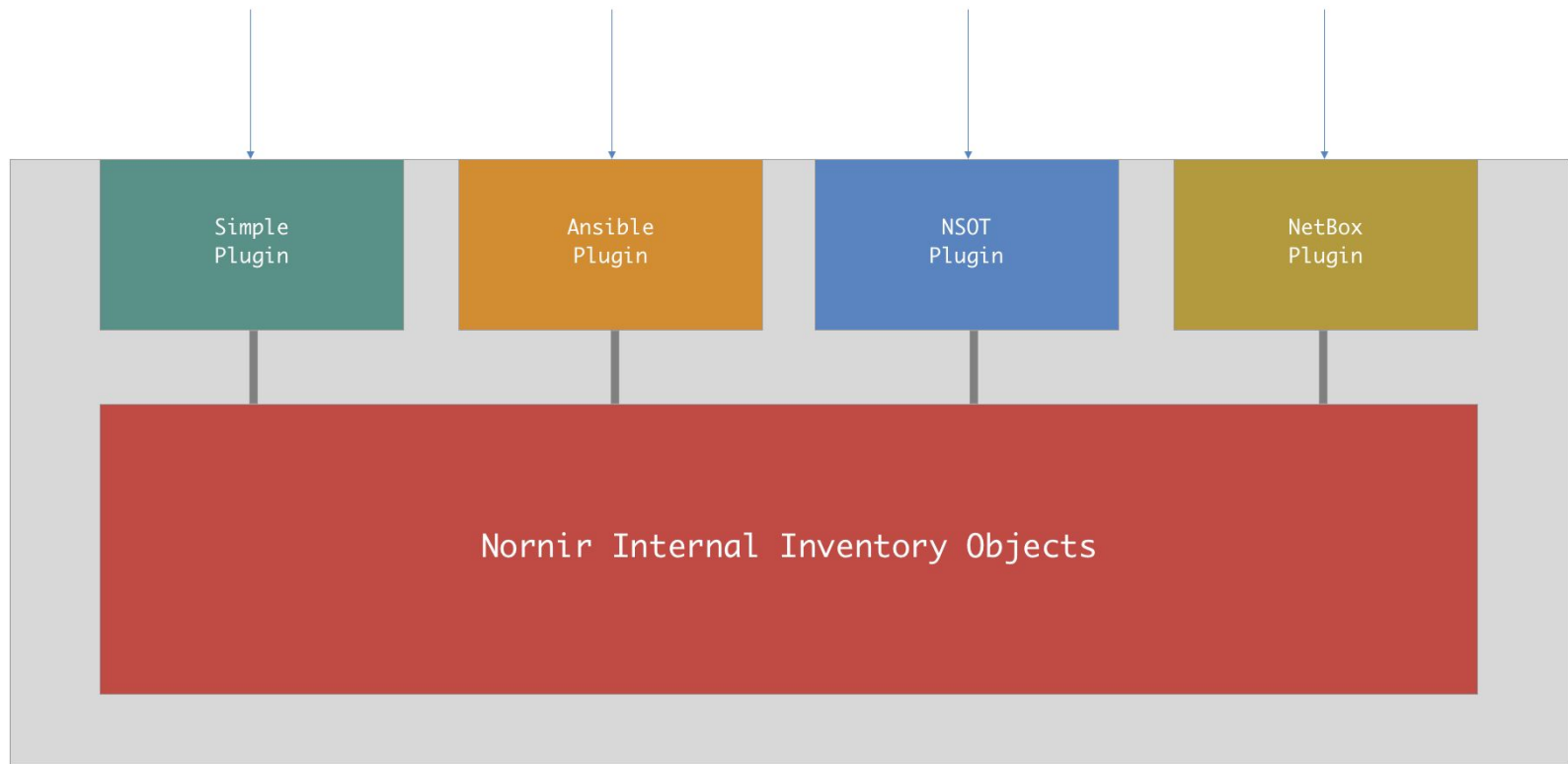


Nornir Minuses

- You need to know Python
- Relatively new project/relatively small number of developers working on it
- Smaller community

Nornir Components





Nornir

SimpleInventory

```
logging:
  enabled: True
inventory:
  plugin: nornir.plugins.inventory.simple.SimpleInventory
  options:
    host_file: "~/nornir_inventory/hosts.yaml"
    group_file: "~/nornir_inventory/groups.yaml"
    defaults_file: "~/nornir_inventory/defaults.yaml"
```

Nornir

SimpleInventory



```
sros1:
  hostname: sros.lasthop.io
  platform: nokia_sros
  connection_options:
    netmiko:
      extras:
        port: 2211
  groups:
    - sros
```




Nornir

Inventory Attribute Traversal

Host -> Group1 -> Group2 -> GroupN -> Defaults

* the "data" exception

The *data inventory exception

Recurses



```
ipdb> nr.inventory.hosts['sros2']['site']  
'Freemont DC'
```

```
ipdb> nr.inventory.hosts['sros2'].data  
{}
```

```
ipdb> █
```

Doesn't Recurse



connection_options

What is going on here?

```
junos:
  platform: junos
  connection_options:
    netmiko:
      platform: juniper_junos
      extras: {}
    napalm:
      extras:
        optional_args: {}
```

Inventory filtering

```
nr = InitNornir(config_file="config.yaml")
tmp_nr = nr.filter(name="sros1")
tmp_nr = nr.filter(platform="nokia_sros")
tmp_nr = nr.filter(hostname="vmx1.lasthop.io")

sros = nr.filter(F(groups__contains="sros"))

all_devices = nr.filter(F(groups__contains="sros") | F(groups__contains="junos"))
```

Exercises:

./day4/nornir_inventory/ex1.txt
./day4/nornir_inventory/ex2.txt
./day4/nornir_inventory/ex3.txt