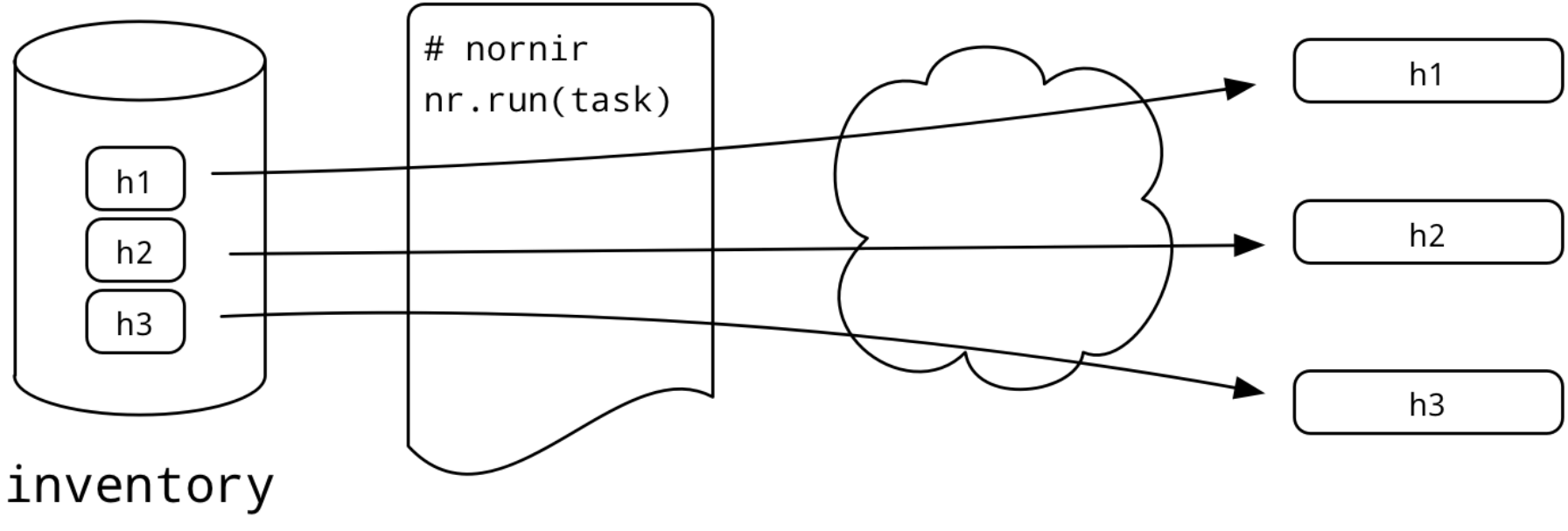
NORNIR

David Barroso Pardo @dbarrosop

Introduction

What's Nornir?

Pluggable multi-threaded framework with inventory management to help operate collections of devices



Why Nornir

Because it's written in python and meant to be used with python

- Orders of magnitude faster than YAML-based alternatives
- Integrate natively with other python frameworks like flask, django, click, etc...
- Easier to extend
- Cleaner logic
- Leverage linters, debuggers and loggers and IDEs for python

```
from nornir import InitNornir
from nornir.plugins.tasks.commands import command
from nornir.plugins.functions.text import print_result

nr = InitNornir(config_file="1_intro/config.yaml")
result = nr.run(task=command,
                command="echo Hi!")
print_result(result, vars=["stdout"])
```

```
command**************************************************************************
* leaf00.bma ** changed : False ************************************************
vvvv command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Hi!

^^^^ END command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* leaf01.bma ** changed : False ************************************************
vvvv command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Hi!

^^^^ END command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* spine00.bma ** changed : False ***********************************************
vvvv command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Hi!

^^^^ END command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* spine01.bma ** changed : False ***********************************************
vvvv command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Hi!

^^^^ END command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Installation

To install nornir execute the following in your python environment:

```
>>> pip install nornir
```

Running this presentation

This presentation is a jupyter (https://jupyter.org/) playbook and all the code is embedded and runnable. To do so:

1. Go to your python environment
2. Clone the repo https://github.com/dravetech/nornir-workshop (https://github.com/dravetech/nornir-workshop)
3. Execute:

```
cd nornir-workshop
pip install -r requirements.txt
jupyter notebook notebooks/
```

4. Open the URL http://localhost:8888 (http://localhost:8888) in your browser
5. Open any file with the extension `.ipynb`
6. Have fun!

Initializing Nornir

Configuration

To initialize nornir you need configuration. There are three ways of configuring Nornir:

1. With a configuration file
2. Directly in the code
3. With a combination of (1) and (2)

With a configuration file

```
!cat 1_intro/config.yaml
```

```
---
num_workers: 20
inventory:
    options:
        host_file: 1_intro/inventory/hosts.yaml
        group_file: 1_intro/inventory/groups.yaml
        defaults_file: 1_intro/inventory/defaults.yaml
```

```
from nornir import InitNornir
nr = InitNornir(config_file="1_intro/config.yaml")
print("Number of threads:", nr.config.num_workers)
print("Hosts file:", nr.config.inventory.options["host_file"])
print("Groups file:", nr.config.inventory.options["group_file"])
print("Defaults file:", nr.config.inventory.options["defaults_file"])
```

```
Number of threads: 20
Hosts file: 1_intro/inventory/hosts.yaml
Groups file: 1_intro/inventory/groups.yaml
Defaults file: 1_intro/inventory/defaults.yaml
```

With code

```
from nornir import InitNornir
nr = InitNornir(
    num_workers=100,
    inventory={
        "options": {
            "host_file": "1_intro/inventory/hosts.yaml",
            "group_file": "1_intro/inventory/groups.yaml",
            "defaults_file": "1_intro/inventory/defaults.yaml",
        }
    }
)
print("Number of threads:", nr.config.num_workers)
print("Hosts file:", nr.config.inventory.options["host_file"])
print("Groups file:", nr.config.inventory.options["group_file"])
print("Defaults file:", nr.config.inventory.options["defaults_file"])
```

```
Number of threads: 100
Hosts file: 1_intro/inventory/hosts.yaml
Groups file: 1_intro/inventory/groups.yaml
Defaults file: 1_intro/inventory/defaults.yaml
```

With a combination

```
from nornir import InitNornir
nr = InitNornir(
    config_file="1_intro/config.yaml",
    num_workers=100,
)
print("Number of threads:", nr.config.num_workers)
print("Hosts file:", nr.config.inventory.options["host_file"])
print("Groups file:", nr.config.inventory.options["group_file"])
print("Defaults file:", nr.config.inventory.options["defaults_file"])
```

```
Number of threads: 100
Hosts file: 1_intro/inventory/hosts.yaml
Groups file: 1_intro/inventory/groups.yaml
Defaults file: 1_intro/inventory/defaults.yaml
```

The inventory

The inventory is arguably the most important piece of nornir. Everything revolves around it.

Inventory Data

The inventory has three pieces of information:

hosts - Hosts contain a mapping about each existing host, how to connect to them and may also contain user defined data. It also describes group membership and data may be inherited from parent groups or defaults.

groups - Similar to hosts

defaults - You can define here default values that will be used when a certain data point is not defined by a host or any of its parent groups.

Understanding the Inventory by example

```
from nornir import InitNornir
nr = InitNornir(
    inventory={
        "options": {
            "host_file": "2_inventory/inventory/hosts.yaml",
            "group_file": "2_inventory/inventory/groups.yaml",
            "defaults_file": "2_inventory/inventory/defaults.yaml",
        }
    }
)
```

```
nr.inventory.hosts
```

```
{'spine00.bma': Host: spine00.bma,
 'spine01.bma': Host: spine01.bma,
 'leaf00.bma': Host: leaf00.bma,
 'leaf01.bma': Host: leaf01.bma,
 'spine00.cmh': Host: spine00.cmh,
 'spine01.cmh': Host: spine01.cmh,
 'leaf00.cmh': Host: leaf00.cmh,
 'leaf01.cmh': Host: leaf01.cmh}
```

```
nr.inventory.groups
```

```
{'bma': Group: bma,
 'cmh': Group: cmh,
 'eos': Group: eos,
 'junos': Group: junos}
```

```
nr.inventory.defaults
```

```
<nornir.core.inventory.Defaults at 0x10a53fa08>
```

Example: `spine00.bma`

```
!sed '2,14!d' 2_inventory/inventory/hosts.yaml
```

```
spine00.bma:
    # well-known attributes; hostname, username, password, platform and port
    hostname: spine00.bma.acme.com
    platform: junos

    groups: # group membership
        - bma
        - junos

    data:  # user-defined data
        site: bma
        role: spine
        region: EU
```

```
!sed '2,4!d' 2_inventory/inventory/groups.yaml
```

```
bma:
    data:
        asn: 65100
```

```
!sed '14,16!d' 2_inventory/inventory/groups.yaml
```

```
junos:
    username: junos_auto
    port: 5022
```

```
!sed '2,3!d' 2_inventory/inventory/defaults.yaml
```

```
data:
    domain: acme.com
```

```
nr.inventory.hosts["spine00.bma"].hostname # comes from the device data
```

```
'spine00.bma.acme.com'
```

```
nr.inventory.hosts["spine00.bma"].username # comes from the group "junos"
```

```
'junos_auto'
```

```
nr.inventory.hosts["spine00.bma"]["asn"] # comes from the group "bma"
                                         # user-defined data is accessed in a dict-like fashion
```

```
65100
```

```
nr.inventory.hosts["spine00.bma"]["domain"] # comes from the defaults
```

```
'acme.com'
```

Example: `leaf01.bma`

```
!sed '39,49!d' 2_inventory/inventory/hosts.yaml
```

```
leaf01.bma:
    hostname: leaf01.bma.acme.com
    platform: eos
    groups:
        - bma
        - eos
    data:
        site: bma
        role: leaf
        region: EU
        asn: 65111
```

```
nr.inventory.hosts["leaf01.bma"].username # comes from the group "eos"
```

```
'eos_auto'
```

```
nr.inventory.hosts["leaf01.bma"]["asn"] # comes from the host itself this time
```

```
65111
```

Groups and defaults behave the same way:

```
nr.inventory.groups["junos"].username
```

```
'junos_auto'
```

```
nr.inventory.groups["bma"]["asn"]
```

```
65100
```

```
print(nr.inventory.defaults.password)
```

```
None
```

```
nr.inventory.defaults.data["domain"] # note that unlike hosts and groups
                                     # there is a data attribute here
```

```
'acme.com'
```

You can easily add/modify/delete data at runtime. Let's start by setting a default password:

```
print(nr.inventory.defaults.password)
```

```
None
```

```
print(nr.inventory.hosts["leaf01.bma"].password)
```

```
None
```

```
# To avoid storing the password in plain files or in the db we could
# store it in some vault, use ssh keys, env variables or we could ask the user
nr.inventory.defaults.password = input("Dear user, enter the password: ")
```

```
Dear user, enter the password: p4ssw0rd
```

```
print(nr.inventory.defaults.password)
```

```
p4ssw0rd
```

```
print(nr.inventory.hosts["leaf01.bma"].password)
```

```
p4ssw0rd
```

```
# to remove it
nr.inventory.defaults.password = None
```

Now, let's add user-defined data:

```
nr.inventory.hosts["leaf01.bma"]["custom_attr"] = "set at runtime"
```

```
nr.inventory.hosts["leaf01.bma"]["custom_attr"]
```

```
'set at runtime'
```

```
# to remove it
nr.inventory.hosts["leaf01.bma"]["custom_attr"] = None
```

Filtering

Nornir doesn't have the concept of "roles" or "classes" or anything like that. Instead, it provides different mechanisms for you to classify and filter hosts so you can implement the mechanism that fits your workflow better.

Those three mechanisms are:

1. Simple filtering
2. Advanced filtering
3. Functions

Let's start by initializing nornir.

```
from nornir import InitNornir
nr = InitNornir(
    inventory={
        "options": {
            "host_file": "3_filtering/inventory/hosts.yaml",
            "group_file": "3_filtering/inventory/groups.yaml",
            "defaults_file": "3_filtering/inventory/defaults.yaml",
        }
    }
)
```

Simple filtering

It allows filtering by matching on `<k,v>` pairs. It's the simplest way of filtering hosts, however, it is not very flexible.

```python
# role and region are user-defined data
leafs = nr.filter(role="leaf")
for host, host_data in leafs.inventory.hosts.items():
    print(f"{host}: {host_data['role']}")
```

```
leaf00.bma: leaf
leaf01.bma: leaf
leaf00.cmh: leaf
leaf01.cmh: leaf
```

```python
leafs_eu = nr.filter(role="leaf", region="EU")
for host, host_data in leafs_eu.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.bma: leaf, EU
leaf01.bma: leaf, EU
```

```python
# filtering is additive
my_devs = nr.filter(role="leaf")
my_devs = my_devs.filter(region="EU")
for host, host_data in my_devs.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.bma: leaf, EU
leaf01.bma: leaf, EU
```

```python
# you can also filter using well-known attributes
junos_devs = nr.filter(platform="junos")
for host, host_data in junos_devs.inventory.hosts.items():
    print(f"{host}: {host_data.platform}")
```

```
spine00.bma: junos
leaf00.bma: junos
spine00.cmh: junos
leaf00.cmh: junos
```

Advanced Filtering

Extremely powerful and flexible, however, it's not as straightforward.

First, you will need to import the `F` object:

```
from nornir.core.filter import F
```

Let's start by porting the examples in "simple filtering".

```
# role and region are user-defined data
leafs = nr.filter(F(role="leaf"))
for host, host_data in leafs.inventory.hosts.items():
    print(f"{host}: {host_data['role']}")
```

```
leaf00.bma: leaf
leaf01.bma: leaf
leaf00.cmh: leaf
leaf01.cmh: leaf
```

```
leafs_eu = nr.filter(F(role="leaf") & F(region="EU"))
for host, host_data in leafs_eu.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.bma: leaf, EU
leaf01.bma: leaf, EU
```

```
# filtering is additive
my_devs = nr.filter(F(role="leaf"))
my_devs = my_devs.filter(F(region="EU"))
for host, host_data in my_devs.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.bma: leaf, EU
leaf01.bma: leaf, EU
```

```
# you can also filter using well-known attributes
junos_devs = nr.filter(F(platform="junos"))
for host, host_data in junos_devs.inventory.hosts.items():
    print(f"{host}: {host_data.platform}")
```

```
spine00.bma: junos
leaf00.bma: junos
spine00.cmh: junos
leaf00.cmh: junos
```

`F` objects support logical operations:

```
# leafs in the EU
leafs_eu = nr.filter(F(role="leaf") & F(region="EU"))
for host, host_data in leafs_eu.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.bma: leaf, EU
leaf01.bma: leaf, EU
```

```
# leaves NOT in the EU
leafs_not_eu = nr.filter(F(role="leaf") & ~F(region="EU"))
for host, host_data in leafs_not_eu.inventory.hosts.items():
    print(f"{host}: {host_data['role']}, {host_data['region']}")
```

```
leaf00.cmh: leaf, NA
leaf01.cmh: leaf, NA
```

```
# either ios or eos switches
ios_or_eos = nr.filter(F(platform="ios") | F(platform="eos"))
for host, host_data in ios_or_eos.inventory.hosts.items():
    print(f"{host}: {host_data.platform}")
```

```
spine01.bma: eos
leaf01.bma: eos
spine01.cmh: ios
leaf01.cmh: ios
```

```
# junos acting as leaves located outside the EU
junos_leaves_not_in_eu = nr.filter(F(platform="junos") & F(role="leaf") & ~F(region="EU"))
for host, host_data in junos_leaves_not_in_eu.inventory.hosts.items():
    print(f"{host}: {host_data.platform}, {host_data['role']}, {host_data['region']}")
```

```
leaf00.cmh: junos, leaf, NA
```

The F objects also let's you filter by nested data

```
#  we have added similar information to all the hosts to represent with image they are running
!sed '10,18!d' 3_filtering/inventory/hosts.yaml
```

```
    data:  # user-defined data
        site: bma
        role: spine
        region: EU
        system:
            image: 17.2R2   # image version
            uptime: 100     # in days, somehow the inventory knows this
```

```
junos_image = nr.filter(F(platform="junos") & F(system__image="14.1X53-D46"))
for host, host_data in junos_image.inventory.hosts.items():
    print(f"{host}: {host_data.platform}, {host_data['system']['image']}")
```

```
leaf00.bma: junos, 14.1X53-D46
leaf00.cmh: junos, 14.1X53-D46
```

The `F` object also lets you perform certain operations on objects:

```
junos_17 = nr.filter(F(platform="junos") & F(system__image__startswith="17"))
for host, host_data in junos_17.inventory.hosts.items():
    print(f"{host}: {host_data.platform}, {host_data['system']['image']}")
```

```
spine00.bma: junos, 17.2R2
spine00.cmh: junos, 17.2R1
```

```
such_uptime = nr.filter(F(system__uptime__ge=50))
for host, host_data in such_uptime.inventory.hosts.items():
    print(f"{host}: {host_data['system']['uptime']}")
```

```
spine00.bma: 100
spine01.bma: 100
spine01.cmh: 50
leaf00.cmh: 60
```

```
various_images = nr.filter(F(platform="junos") & F(system__image__any=["14.1X53-D46", "17.2R1"]))
for host, host_data in various_images.inventory.hosts.items():
    print(f"{host}: {host_data.platform}, {host_data['system']['image']}")
```

```
leaf00.bma: junos, 14.1X53-D46
spine00.cmh: junos, 17.2R1
leaf00.cmh: junos, 14.1X53-D46
```

Functions

You can also filter objects by providing your own function. Such function will accept a `host` as parameter and needs to return either `True` (accept) or `False` (filter).

```
import re

def eos_maintenance_images(host):
    """ EOS images end either in M (maintenance images) or F (feature images)"""
    return bool(re.match(".*M$", host["system"]["image"]))
```

```
maintenance_images = nr.filter(filter_func=eos_maintenance_images)
for host, host_data in maintenance_images.inventory.hosts.items():
    print(f"{host}: {host_data.platform}, {host_data['system']['image']}")
```

```
spine01.bma: eos, 4.19.4M
leaf01.bma: eos, 4.17.4M
```

Running tasks

Let's start by initializing nornir.

```
from nornir import InitNornir
nr = InitNornir(
    inventory={
        "options": {
            "host_file": "4_tasks/inventory/hosts.yaml",
            "group_file": "4_tasks/inventory/groups.yaml",
            "defaults_file": "4_tasks/inventory/defaults.yaml",
        }
    }
)
```

Running a task

```
from nornir.plugins.tasks.networking import napalm_get

# we filter the hosts (optional, this would be the equivalent of "role")
spine_bma = nr.filter(role="spine", site="bma")

# we run the task, saving the result in the variable `r`
r = spine_bma.run(name="Get facts",
                  task=napalm_get,
                  getters=["facts"])
```

You can use the helper function `print_result` to print the result on the screen:

```
from nornir.plugins.functions.text import print_result

print_result(r)
```

```
Get facts********************************************************************
* spine00.bma ** changed : False ***********************************************
vvvv Get facts ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
{ 'facts': { 'fqdn': 'localhost',
             'hostname': 'localhost',
             'interface_list': ['Ethernet1', 'Ethernet2', 'Management1'],
             'model': 'vEOS',
             'os_version': '4.20.1F-6820520.4201F',
             'serial_number': '',
             'uptime': 317919,
             'vendor': 'Arista'}}
^^^^ END Get facts ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* spine01.bma ** changed : False ***********************************************
vvvv Get facts ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
{ 'facts': { 'fqdn': 'vsrx',
             'hostname': 'vsrx',
             'interface_list': ['ge-0/0/0', 'ge-0/0/1', 'vlan'],
             'model': 'FIREFLY-PERIMETER',
             'os_version': '12.1X47-D20.7',
             'serial_number': '5f5d14568760',
             'uptime': 4338,
             'vendor': 'Juniper'}}
^^^^ END Get facts ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

You can also work with the result programmatically:

```
# running a task will return a dict-like object
# where the keys are the hosts' names and the values
# a list of results (more on why a list later)
for host, task_results in r.items():
    print(f"{host}: {task_results}")
```

```
spine00.bma: MultiResult: [Result: "Get facts"]
spine01.bma: MultiResult: [Result: "Get facts"]
```

```
for host, task_results in r.items():
    my_task = task_results[0]          # the only task we ran
    my_task_result = my_task.result    # this should contain the exact output from napalm's `get_facts`
    print(f"{host}: {my_task_result['facts']['os_version']}")
```

```
spine00.bma: 4.20.1F-6820520.4201F
spine01.bma: 12.1X47-D20.7
```

Running multiple tasks: a simple example

```
from nornir.plugins.tasks.networking import napalm_cli


# we group the tasks
def a_bunch_of_tasks(task):
    task.run(name="Get facts",
             task=napalm_get,
             getters=["facts"])
    task.run(name="Run CLI command",
             task=napalm_cli,
             commands=["show version"])
    return "I am done!!!"


r = spine_bma.run(name="Running a bunch of tasks",
                  task=a_bunch_of_tasks)
```

```
print_result(r)
```

```
Running a bunch of tasks**********************************************************
* spine00.bma ** changed : False ***********************************************
vvvv Running a bunch of tasks ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
I am done!!!
---- Get facts ** changed : False ------------------------------------------------ INFO
{ 'facts': { 'fqdn': 'localhost',
             'hostname': 'localhost',
             'interface_list': ['Ethernet1', 'Ethernet2', 'Management1'],
             'model': 'vEOS',
             'os_version': '4.20.1F-6820520.4201F',
             'serial_number': '',
             'uptime': 317919,
             'vendor': 'Arista'}}
---- Run CLI command ** changed : False ------------------------------------------ INFO
{ 'show version': 'Arista vEOS\n'
                  'Hardware version:\n'
                  'Serial number:\n'
                  'System MAC address:  0800.273e.22fd\n'
                  '\n'
                  'Software image version: 4.20.1F\n'
                  'Architecture:           i386\n'
                  'Internal build version: 4.20.1F-6820520.4201F\n'
                  'Internal build ID:\n'
                  '790a11e8-5aaf-4be7-a11a-e61795d05b91\n'
                  '\n'
                  'Uptime:                 1 hour and 16 minutes\n'
                  'Total memory:           2017324 kB\n'
                  'Free memory:            1237124 kB\n'}
^^^^ END Running a bunch of tasks ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* spine01.bma ** changed : False ***********************************************
vvvv Running a bunch of tasks ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
I am done!!!
---- Get facts ** changed : False ------------------------------------------------ INFO
{ 'facts': { 'fqdn': 'vsrx',
             'hostname': 'vsrx',
             'interface_list': ['ge-0/0/0', 'ge-0/0/1', 'vlan'],
             'model': 'FIREFLY-PERIMETER',
             'os_version': '12.1X47-D20.7',
             'serial_number': '5f5d14568760',
             'uptime': 4338,
             'vendor': 'Juniper'}}
---- Run CLI command ** changed : False ------------------------------------------ INFO
{ 'show version': 'Hostname: vsrx\n'
                  'Model: firefly-perimeter\n'
                  'JUNOS Software Release [12.1X47-D20.7]\n'}
^^^^ END Running a bunch of tasks ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
for host, task_results in r.items():
    grouped_result = task_results[0].result
    get_facts_result = task_results[1].result
    cli_result = task_results[2].result
    print(f"{host}: {get_facts_result['facts']['os_version']}")
    print(f"{cli_result['show version']}")
    print(f"---> {grouped_result}")
    print("=================")
```

```
spine00.bma: 4.20.1F-6820520.4201F
Arista vEOS
Hardware version:
Serial number:
System MAC address:  0800.273e.22fd

Software image version: 4.20.1F
Architecture:           i386
Internal build version: 4.20.1F-6820520.4201F
Internal build ID:
790a11e8-5aaf-4be7-a11a-e61795d05b91

Uptime:                 1 hour and 16 minutes
Total memory:           2017324 kB
Free memory:            1237124 kB

---> I am done!!!
=================
spine01.bma: 12.1X47-D20.7
Hostname: vsrx
Model: firefly-perimeter
JUNOS Software Release [12.1X47-D20.7]

---> I am done!!!
=================
```

DIY Intent based networking

The objective of this module is to show you a few things:

1. How to create your own "complex" tasks by leveraging existing tasks
2. How complex workflows become simple thanks to the power of python

To do that, we are going to create a task that synchronizes users; it will add missing users, modify the ones that need to be modified and remove old users. This task will be platform agnostic and will support even platforms without hierarchical config.

As usual, let's initializate the nornir object:

```
from nornir import InitNornir
nr = InitNornir(
    inventory={
        "options": {
            "host_file": "5_manage_users/inventory/hosts.yaml",
            "group_file": "5_manage_users/inventory/groups.yaml",
            "defaults_file": "5_manage_users/inventory/defaults.yaml",
        }
    }
)
```

We are going to store the users in a yaml file different than the inventory:

```
!cat 5_manage_users/data/users.yaml
```

```
---
joe:
jane:
admin:
```

Now we are going to create a template per platform that takes:

1. `remove_users` - users we want to remove
2. `desired_users` - users we want in the system

```
!cat 5_manage_users/templates/eos/users.j2
```

```
{% for user in remove_users %}
no username {{ user }}
{% endfor %}

{% for user in desired_users %}
username {{ user }} privilege 15 role network-admin secret sha512 $6$KxgYSk1jaMjZxxBs$1Qmn8UKx7rj3tREiXr
dbhXKCC2Mw0w7LwysWCY/xjAX3QDOe0whVdSbn6O50pCsAhNAqB3pk9kY0Nw2OB/Uhz1
{% endfor %}
```

```
!cat 5_manage_users/templates/junos/users.j2
```

```
{% for user in remove_users %}
delete system login user {{ user }}
{% endfor %}

{% for user in desired_users %}
set system login user {{ user }} uid 2000
set system login user {{ user }} class super-user
{% endfor %}
```

First, we create a function that does what we need:

```
from nornir.plugins.tasks.networking import napalm_configure, napalm_get
from nornir.plugins.tasks.text import template_file

def manage_users(task, desired_users):
    # get users from device
    state_users = task.run(task=napalm_get,
                           getters=["users"],
                           severity_level=logging.DEBUG)

    # let's verify if the users we got are in desired_users
    # if they are not we have to remove them
    users_to_remove = []
    for user in state_users.result["users"]:
        if user not in desired_users:
            users_to_remove.append(user)

    # we render the template for the platform passing desired_users and users_to_remove
    users_config = task.run(task=template_file,
                            path=f"5_manage_users/templates/{task.host.platform}",
                            template="users.j2",
                            desired_users=desired_users,
                            remove_users=users_to_remove,
                            severity_level=logging.DEBUG)

    # we load the resulting configuration into the device
    task.run(task=napalm_configure,
             configuration=users_config.result)
```

```
import logging

import ruamel.yaml

from nornir.plugins.functions.text import print_result

# we load from a yaml file the users we want
yaml = ruamel.yaml.YAML()
with open("5_manage_users/data/users.yaml", "r") as f:
    desired_users = yaml.load(f.read())

spines = nr.filter(role="spine")

# we call manage_users passing the users we loaded from the yaml file
r = spines.run(task=manage_users,
               desired_users=desired_users)
```

```
print_result(r)
```

```
manage_users*********************************************************************
* spine00.bma ** changed : True ************************************************
vvvv manage_users ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_configure ** changed : True ---------------------------------------- INFO
@@ -14,8 +14,9 @@
 !
 aaa root secret sha512 $6$XoN/NKq7rqSxD9ov$xeiYtli3w.fEiNCsYQ4aJkWWZCor6tNVOwaTwB0UinZTTtV0qBk0VvNBlR4K
csctbf6atsrKt3iTLc9wzOqoT1
 !
-username bob privilege 15 role network-admin secret sha512 $6$6ToHoIeHQFIQeBTL$2iEVU6ReAVTNaoaxdSawgsa2
e/dOxclhMb2D7X7vKYpNKmsXRyEs6YhTxij24i10d.IozA5cKVok5zm2lmV0v/
+username joe privilege 15 role network-admin secret sha512 $6$KxgYSk1jaMjZxxBs$1Qmn8UKx7rj3tREiXrdbhXKC
C2Mw0w7LwysWCY/xjAX3QDOe0whVdSbn6O50pCsAhNAqB3pk9kY0Nw2OB/Uhz1
+username jane privilege 15 role network-admin secret sha512 $6$KxgYSk1jaMjZxxBs$1Qmn8UKx7rj3tREiXrdbhXK
CC2Mw0w7LwysWCY/xjAX3QDOe0whVdSbn6O50pCsAhNAqB3pk9kY0Nw2OB/Uhz1
 !
 interface Ethernet1
 !
^^^^ END manage_users ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* spine01.bma ** changed : True ************************************************
vvvv manage_users ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_configure ** changed : True ---------------------------------------- INFO
[edit system login]
+    user admin {
+        uid 2000;
+        class super-user;
+    }
+    user joe {
+        uid 2000;
+        class super-user;
+    }
+    user jane {
+        uid 2000;
+        class super-user;
+    }
-    user bob {
-        uid 2000;
-        class super-user;
-        authentication {
-            ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7gH
zIw+niNltGEFHzD8+v1I2YJ6oXevct1YeS0o9HZyN1Q9qgCgzUFtdOKLv6IedplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo
9D8HfdOV0IAdx4O7PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR61RGL+GPXQ2MWZWFYbAGjyiYJnAmCP3NOTd0jMZEnDkbUvx
hMmBYSdETk1rRgm+R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insec
ure public key"; ## SECRET-DATA
-        }
-    }
^^^^ END manage_users ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Python goodies

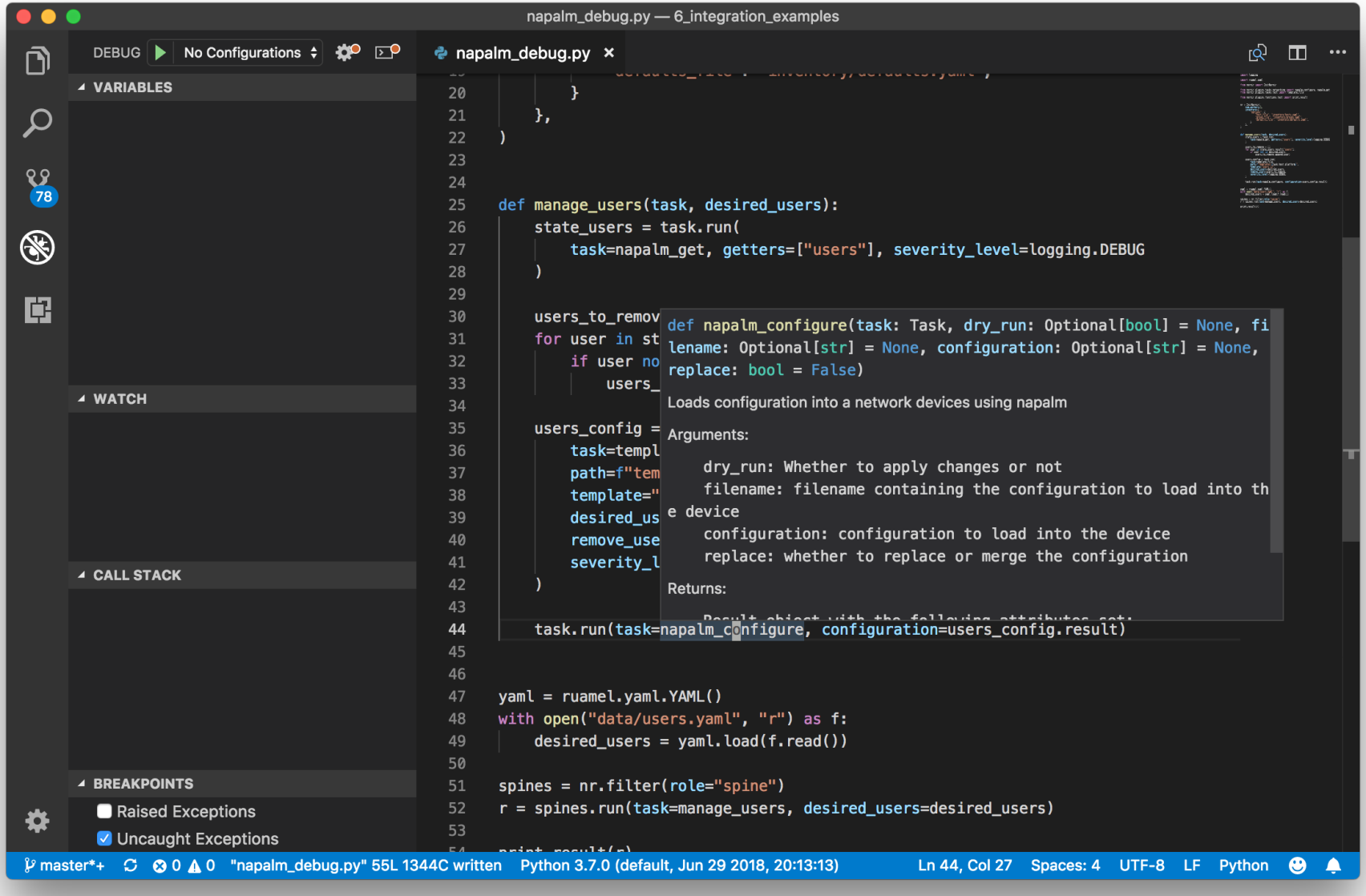
IDE integration

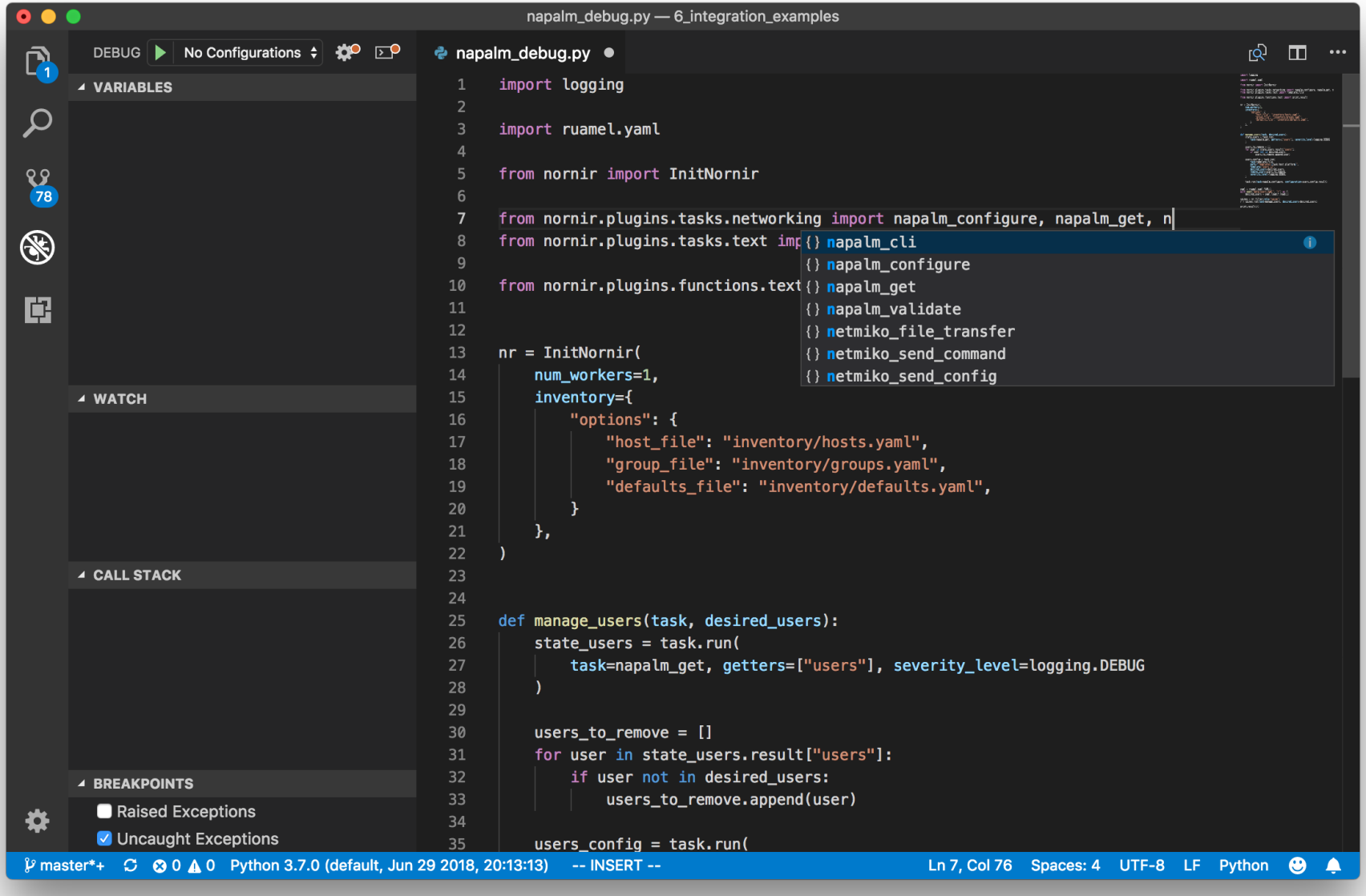
IDE stands for Integrated development environment. It's usually a text editor that understands the code and provides things like:

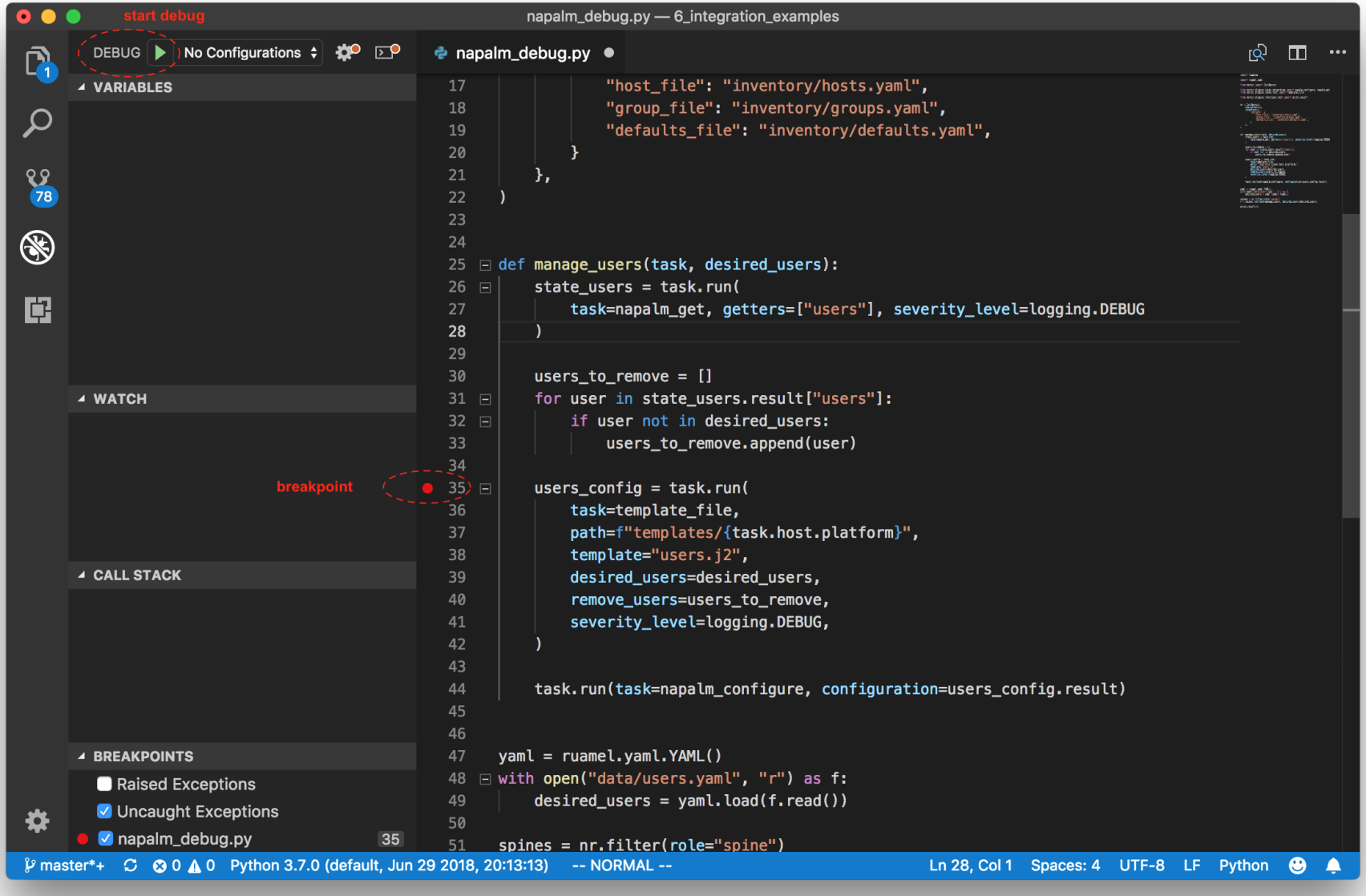
- hinting
- code completion
- debugging capabilities
- etc

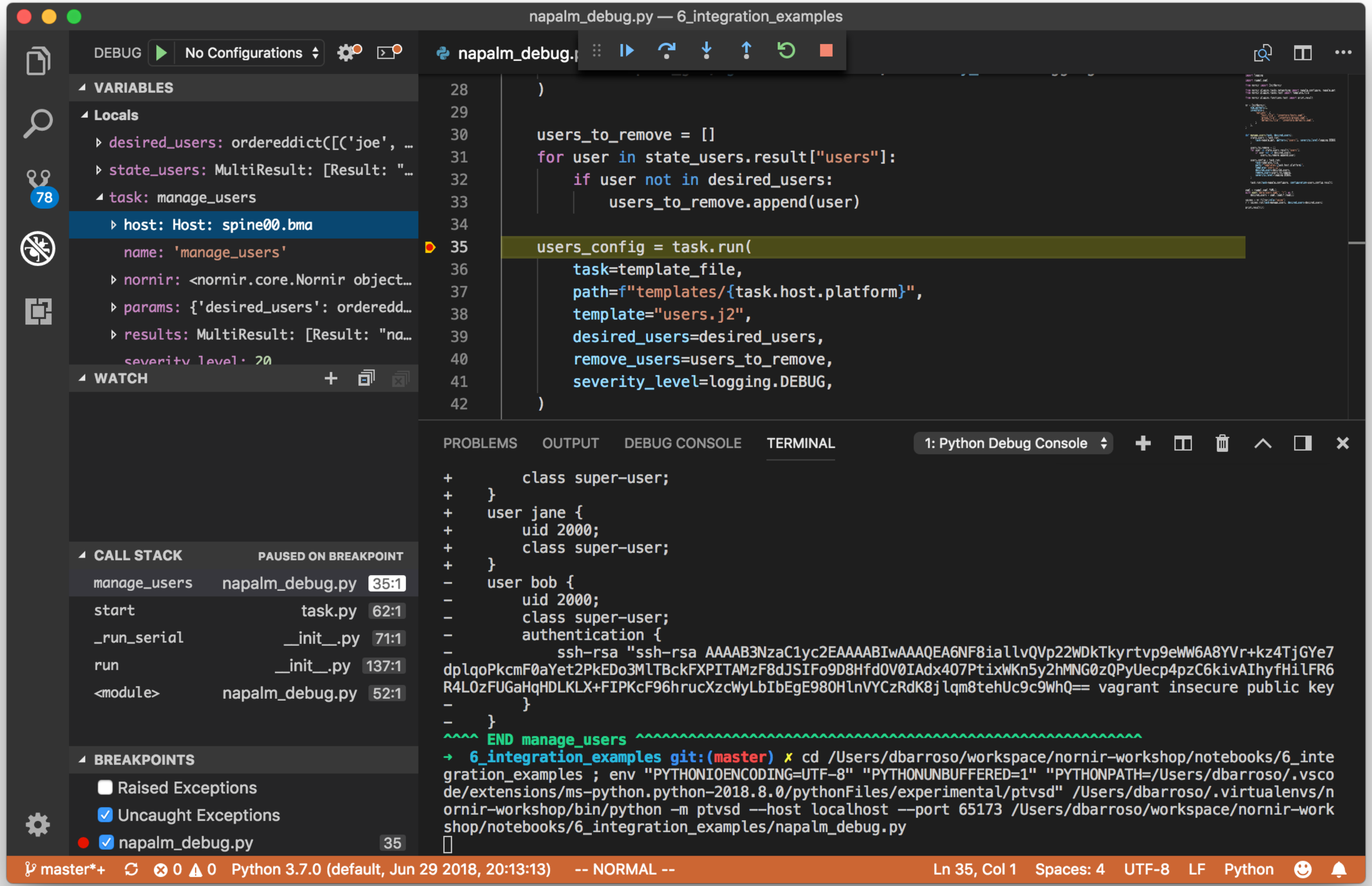
Nornir is a first class python citizen which means you get all these features.

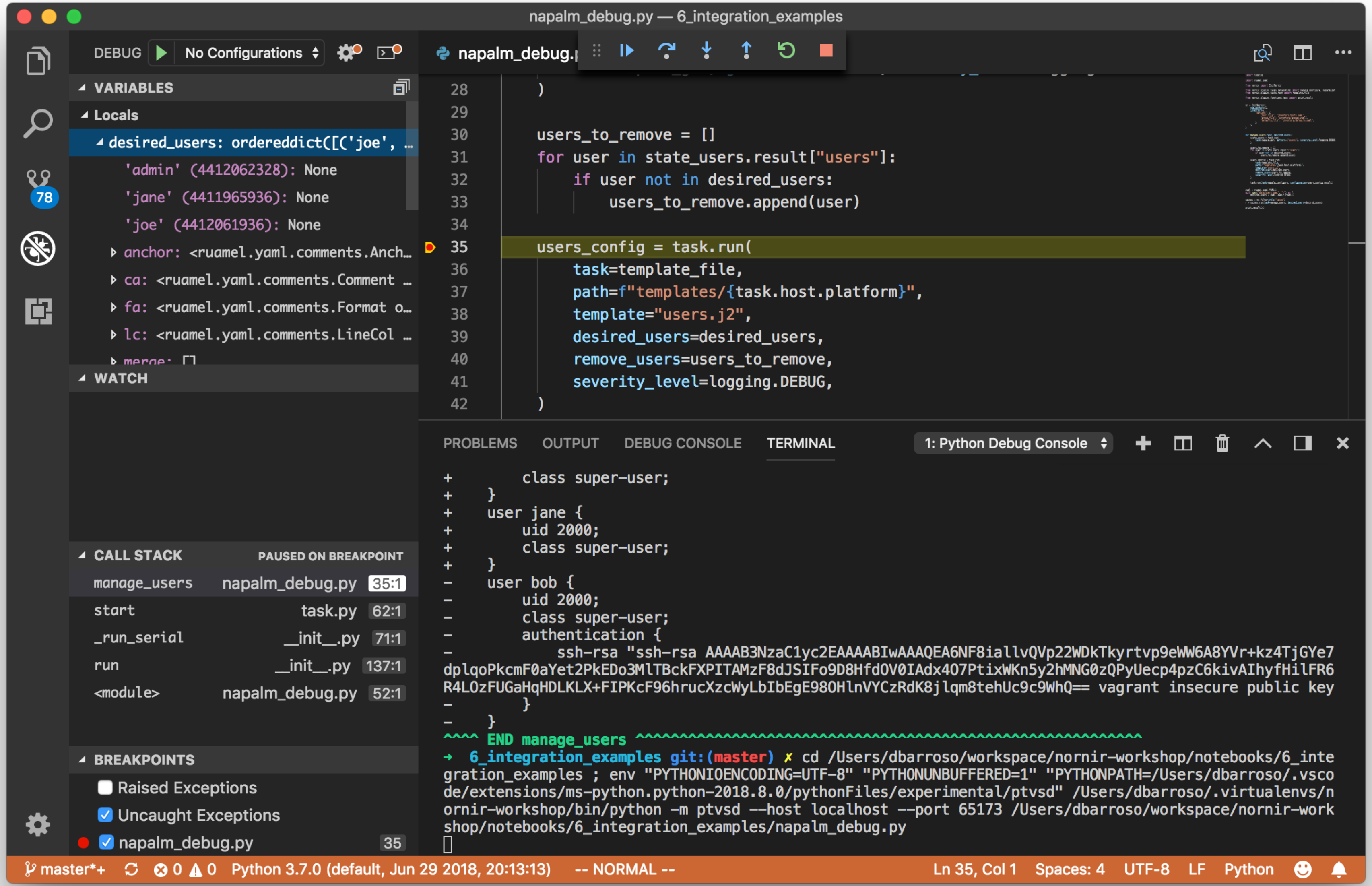
Let's see a quick example using Visual Studio Code (https://code.visualstudio.com)

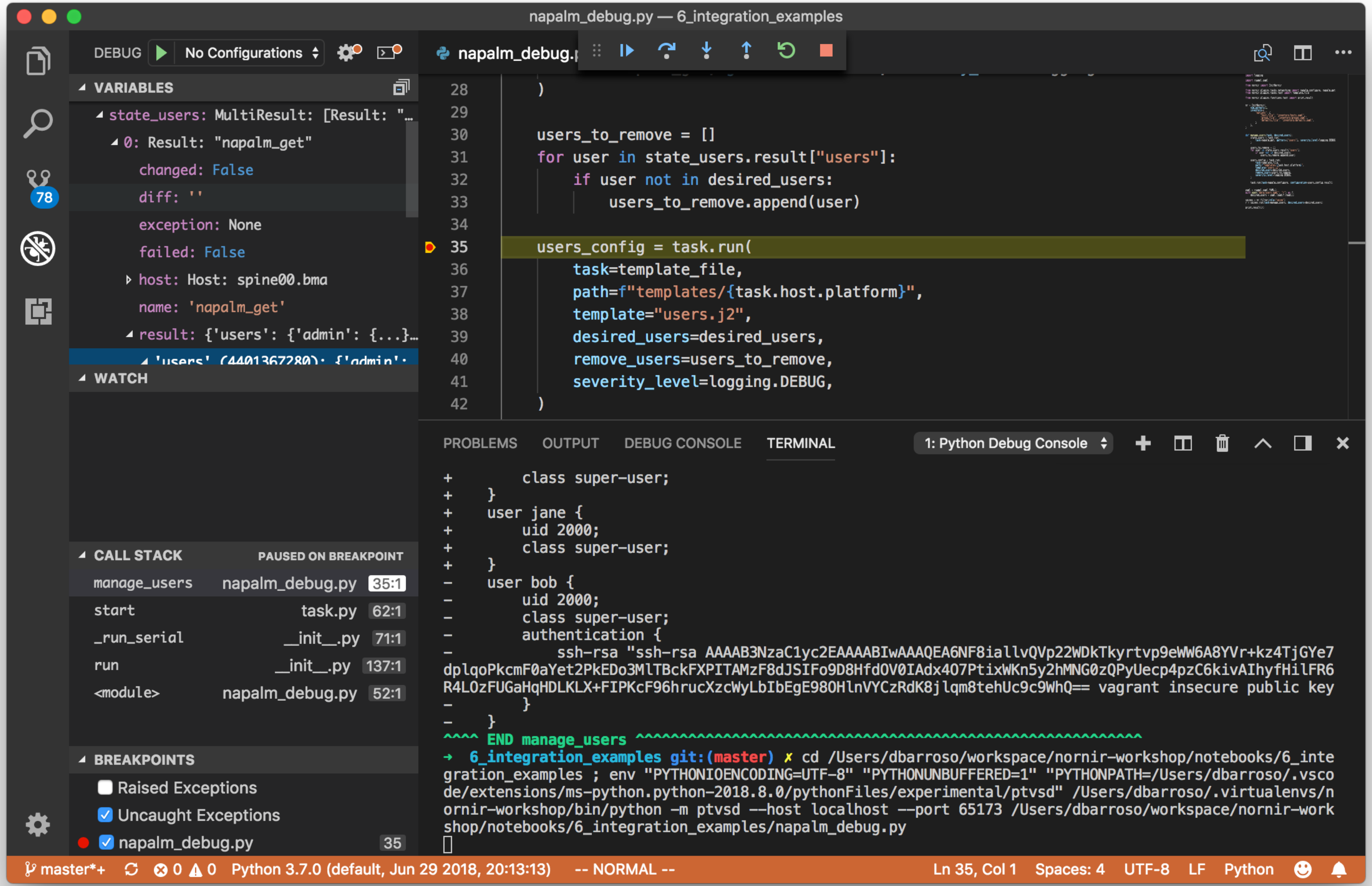

napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
def napalm_configure(task: Task, dry_run: Optional[bool] = None, filename: Optional[str] = None, configuration: Optional[str] = None, replace: bool = False)
Loads configuration into a network devices using napalm
Arguments:
dry_run: Whether to apply changes or not
filename: filename containing the configuration to load into the device
configuration: configuration to load into the device
replace: whether to replace or merge the configuration
Returns:
task.run(task=napalm_configure, configuration=users_config.result)
master*+
"napalm_debug.py" 55L 1344C written
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
Ln 44, Col 27
Spaces: 4
UTF-8
LF
Python

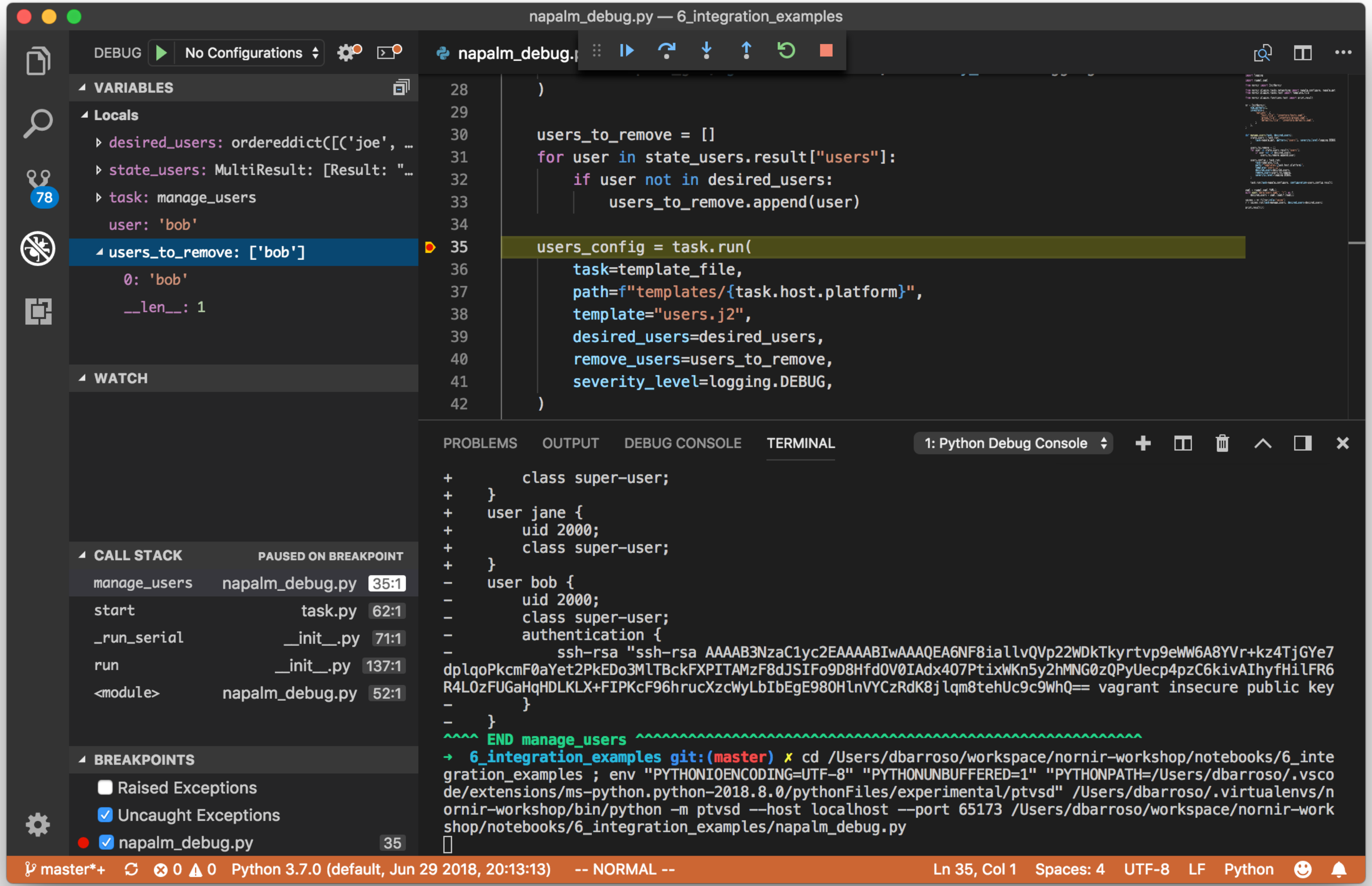

napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
import logging
import ruamel.yaml
from nornir import InitNornir
from nornir.plugins.tasks.networking import napalm_configure, napalm_get, n
from nornir.plugins.tasks.text imp
from nornir.plugins.functions.text
nr = InitNornir(
num_workers=1,
inventory={
"options": {
"host_file": "inventory/hosts.yaml",
"group_file": "inventory/groups.yaml",
"defaults_file": "inventory/defaults.yaml",
}
},
)
def manage_users(task, desired_users):
state_users = task.run(
task=napalm_get, getters=["users"], severity_level=logging.DEBUG
)
users_to_remove = []
for user in state_users.result["users"]:
if user not in desired_users:
users_to_remove.append(user)
users_config = task.run(
napalm_cli
napalm_configure
napalm_get
napalm_validate
netmiko_file_transfer
netmiko_send_command
netmiko_send_config
master*+
0 0
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
-- INSERT --
Ln 7, Col 76
Spaces: 4
UTF-8
LF
Python


start debug
napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
breakpoint
"host_file": "inventory/hosts.yaml",
"group_file": "inventory/groups.yaml",
"defaults_file": "inventory/defaults.yaml",
}
},
)
def manage_users(task, desired_users):
state_users = task.run(
task=napalm_get, getters=["users"], severity_level=logging.DEBUG
)
users_to_remove = []
for user in state_users.result["users"]:
if user not in desired_users:
users_to_remove.append(user)
users_config = task.run(
task=template_file,
path=f"templates/{task.host.platform}",
template="users.j2",
desired_users=desired_users,
remove_users=users_to_remove,
severity_level=logging.DEBUG,
)
task.run(task=napalm_configure, configuration=users_config.result)
yaml = ruamel.yaml.YAML()
with open("data/users.yaml", "r") as f:
desired_users = yaml.load(f.read())
spines = nr.filter(role="spine")
master*+ 0 0 Python 3.7.0 (default, Jun 29 2018, 20:13:13) -- NORMAL -- Ln 28, Col 1 Spaces: 4 UTF-8 LF Python

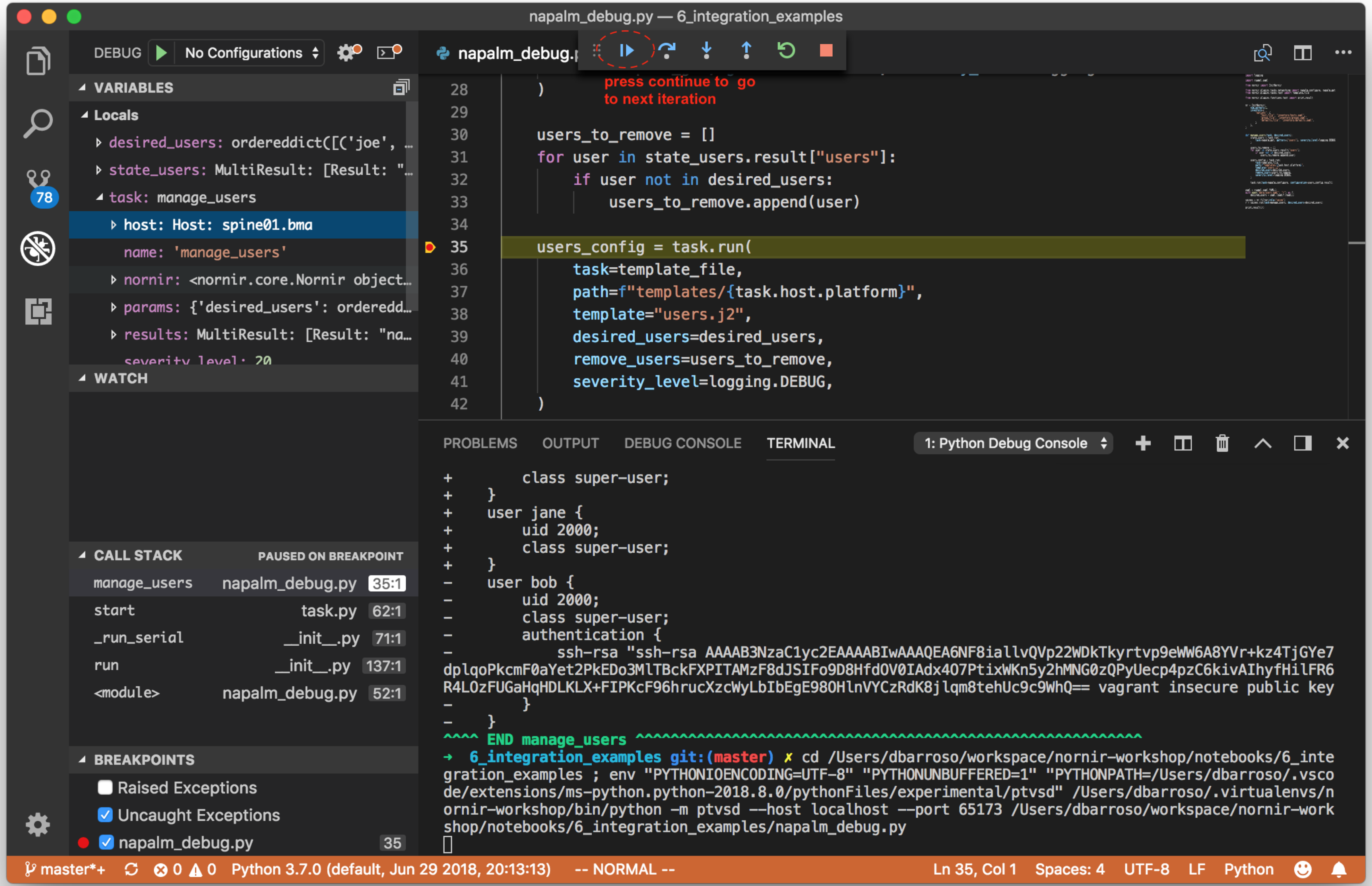

napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
Locals
desired_users: orderedict([('joe', …
state_users: MultiResult: [Result: "…
task: manage_users
host: Host: spine00.bma
name: 'manage_users'
nornir: <nornir.core.Nornir object…
params: {'desired_users': orderedd…
results: MultiResult: [Result: "na…
severity_level: 20
WATCH
CALL STACK
PAUSED ON BREAKPOINT
manage_users napalm_debug.py 35:1
start task.py 62:1
_run_serial __init__.py 71:1
run __init__.py 137:1
<module> napalm_debug.py 52:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
78
)
users_to_remove = []
for user in state_users.result["users"]:
if user not in desired_users:
users_to_remove.append(user)
users_config = task.run(
task=template_file,
path=f"templates/{task.host.platform}",
template="users.j2",
desired_users=desired_users,
remove_users=users_to_remove,
severity_level=logging.DEBUG,
)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Console
+ class super-user;
+ }
+ user jane {
+ uid 2000;
+ class super-user;
+ }
- user bob {
- uid 2000;
- class super-user;
- authentication {
- ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7dplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx407PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR6R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE980HlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
- }
- }
^^^^ END manage_users ^^
→ 6_integration_examples git:(master) ✗ cd /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples ; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "PYTHONPATH=/Users/dbarroso/.vscode/extensions/ms-python.python-2018.8.0/pythonFiles/experimental/ptvsd" /Users/dbarroso/.virtualenvs/nornir-workshop/bin/python -m ptvsd --host localhost --port 65173 /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples/napalm_debug.py
master*+
0 0
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
-- NORMAL --
Ln 35, Col 1
Spaces: 4
UTF-8
LF
Python


napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
Locals
desired_users: ordereddict([('joe', …
'admin' (4412062328): None
'jane' (4411965936): None
'joe' (4412061936): None
anchor: <ruamel.yaml.comments.Anch…
ca: <ruamel.yaml.comments.Comment …
fa: <ruamel.yaml.comments.Format o…
lc: <ruamel.yaml.comments.LineCol …
WATCH
CALL STACK
PAUSED ON BREAKPOINT
manage_users napalm_debug.py 35:1
start task.py 62:1
_run_serial __init__.py 71:1
run __init__.py 137:1
<module> napalm_debug.py 52:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
)
users_to_remove = []
for user in state_users.result["users"]:
if user not in desired_users:
users_to_remove.append(user)
users_config = task.run(
task=template_file,
path=f"templates/{task.host.platform}",
template="users.j2",
desired_users=desired_users,
remove_users=users_to_remove,
severity_level=logging.DEBUG,
)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Console
+ class super-user;
+ }
+ user jane {
+ uid 2000;
+ class super-user;
+ }
- user bob {
- uid 2000;
- class super-user;
- authentication {
- ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7
dplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx407PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR6
R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE980HlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
- }
- }
^^^^ END manage_users ^^
→ 6_integration_examples git:(master) ✗ cd /Users/dbarroso/workspace/nornir-workshop/notebooks/6_inte
gration_examples ; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "PYTHONPATH=/Users/dbarroso/.vsco
de/extensions/ms-python.python-2018.8.0/pythonFiles/experimental/ptvsd" /Users/dbarroso/.virtualenvs/n
ornir-workshop/bin/python -m ptvsd --host localhost --port 65173 /Users/dbarroso/workspace/nornir-work
shop/notebooks/6_integration_examples/napalm_debug.py
master*+ 0 0 Python 3.7.0 (default, Jun 29 2018, 20:13:13) -- NORMAL --
Ln 35, Col 1 Spaces: 4 UTF-8 LF Python


napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
state_users: MultiResult: [Result: "…
0: Result: "napalm_get"
changed: False
diff: ''
exception: None
failed: False
host: Host: spine00.bma
name: 'napalm_get'
result: {'users': {'admin': {...}…
'users' (4401367280): {'admin':
WATCH
CALL STACK
PAUSED ON BREAKPOINT
manage_users napalm_debug.py 35:1
start task.py 62:1
_run_serial __init__.py 71:1
run __init__.py 137:1
<module> napalm_debug.py 52:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
78
28)
29
30 users_to_remove = []
31 for user in state_users.result["users"]:
32 if user not in desired_users:
33 users_to_remove.append(user)
34
35 users_config = task.run(
36 task=template_file,
37 path=f"templates/{task.host.platform}",
38 template="users.j2",
39 desired_users=desired_users,
40 remove_users=users_to_remove,
41 severity_level=logging.DEBUG,
42)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Console
+ class super-user;
+ }
+ user jane {
+ uid 2000;
+ class super-user;
+ }
- user bob {
- uid 2000;
- class super-user;
- authentication {
- ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7
dplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx407PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR6
R4L0zFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
- }
- }
^^^^ END manage_users ^^^
→ 6_integration_examples git:(master) ✗ cd /Users/dbarroso/workspace/nornir-workshop/notebooks/6_inte
gration_examples ; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "PYTHONPATH=/Users/dbarroso/.vsco
de/extensions/ms-python.python-2018.8.0/pythonFiles/experimental/ptvsd" /Users/dbarroso/.virtualenvs/n
ornir-workshop/bin/python -m ptvsd --host localhost --port 65173 /Users/dbarroso/workspace/nornir-work
shop/notebooks/6_integration_examples/napalm_debug.py
master*+ 0 0 Python 3.7.0 (default, Jun 29 2018, 20:13:13) -- NORMAL -- Ln 35, Col 1 Spaces: 4 UTF-8 LF Python



napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
VARIABLES
Locals
desired_users: ordereddict([('joe', …
state_users: MultiResult: [Result: "…
task: manage_users
user: 'bob'
users_to_remove: ['bob']
0: 'bob'
__len__: 1
WATCH
CALL STACK
PAUSED ON BREAKPOINT
manage_users napalm_debug.py 35:1
start task.py 62:1
_run_serial __init__.py 71:1
run __init__.py 137:1
<module> napalm_debug.py 52:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
)
users_to_remove = []
for user in state_users.result["users"]:
if user not in desired_users:
users_to_remove.append(user)
users_config = task.run(
task=template_file,
path=f"templates/{task.host.platform}",
template="users.j2",
desired_users=desired_users,
remove_users=users_to_remove,
severity_level=logging.DEBUG,
)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Console
+ class super-user;
+ }
+ user jane {
+ uid 2000;
+ class super-user;
+ }
- user bob {
- uid 2000;
- class super-user;
- authentication {
- ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7dplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx407PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR6R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
- }
- }
^^^^ END manage_users ^^^
→ 6_integration_examples git:(master) ✗ cd /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples ; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "PYTHONPATH=/Users/dbarroso/.vscode/extensions/ms-python.python-2018.8.0/pythonFiles/experimental/ptvsd" /Users/dbarroso/.virtualenvs/nornir-workshop/bin/python -m ptvsd --host localhost --port 65173 /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples/napalm_debug.py
master*+
0 0
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
-- NORMAL --
Ln 35, Col 1
Spaces: 4
UTF-8
LF
Python


napalm_debug.py — 6_integration_examples
DEBUG
No Configurations
napalm_debug.py
press continue to go
to next iteration
VARIABLES
Locals
desired_users: ordereddict([('joe', …
state_users: MultiResult: [Result: "…
task: manage_users
host: Host: spine01.bma
name: 'manage_users'
nornir: <nornir.core.Nornir object…
params: {'desired_users': orderedd…
results: MultiResult: [Result: "na…
severity_level: 20
WATCH
CALL STACK
PAUSED ON BREAKPOINT
manage_users napalm_debug.py 35:1
start task.py 62:1
_run_serial __init__.py 71:1
run __init__.py 137:1
<module> napalm_debug.py 52:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
napalm_debug.py 35
28)
29
30 users_to_remove = []
31 for user in state_users.result["users"]:
32 if user not in desired_users:
33 users_to_remove.append(user)
34
35 users_config = task.run(
36 task=template_file,
37 path=f"templates/{task.host.platform}",
38 template="users.j2",
39 desired_users=desired_users,
40 remove_users=users_to_remove,
41 severity_level=logging.DEBUG,
42)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Console
+ class super-user;
+ }
+ user jane {
+ uid 2000;
+ class super-user;
+ }
- user bob {
- uid 2000;
- class super-user;
- authentication {
- ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz4TjGYe7dplqoPkcmF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx407PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR6R4LOzFUGaHqHDLKLX+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
- }
- }
^^^^ END manage_users ^^^
→ 6_integration_examples git:(master) ✗ cd /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples ; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "PYTHONPATH=/Users/dbarroso/.vscode/extensions/ms-python.python-2018.8.0/pythonFiles/experimental/ptvsd" /Users/dbarroso/.virtualenvs/nornir-workshop/bin/python -m ptvsd --host localhost --port 65173 /Users/dbarroso/workspace/nornir-workshop/notebooks/6_integration_examples/napalm_debug.py
master*+
0 0
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
-- NORMAL --
Ln 35, Col 1
Spaces: 4
UTF-8
LF
Python

Integration with other frameworks

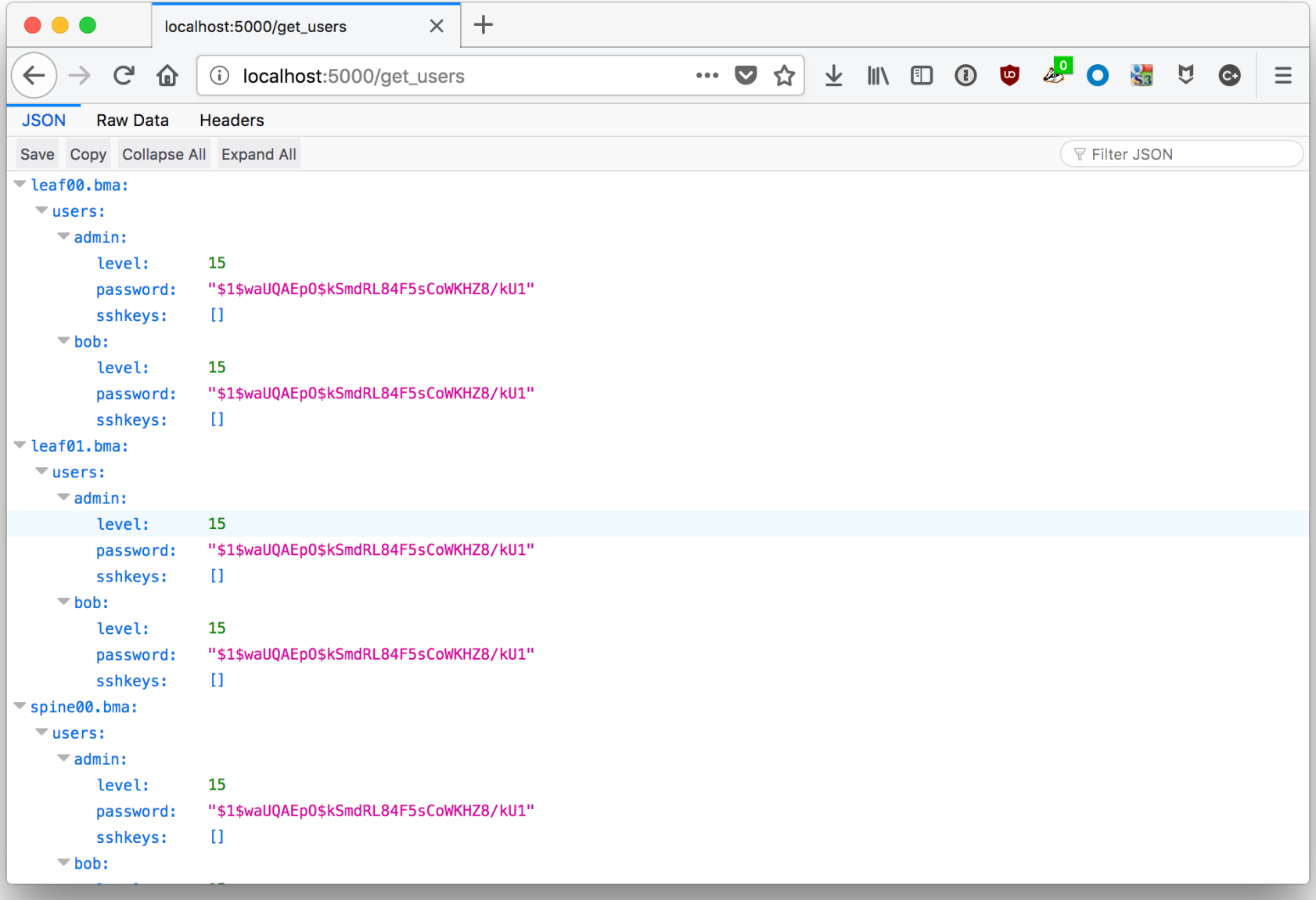
Flask example

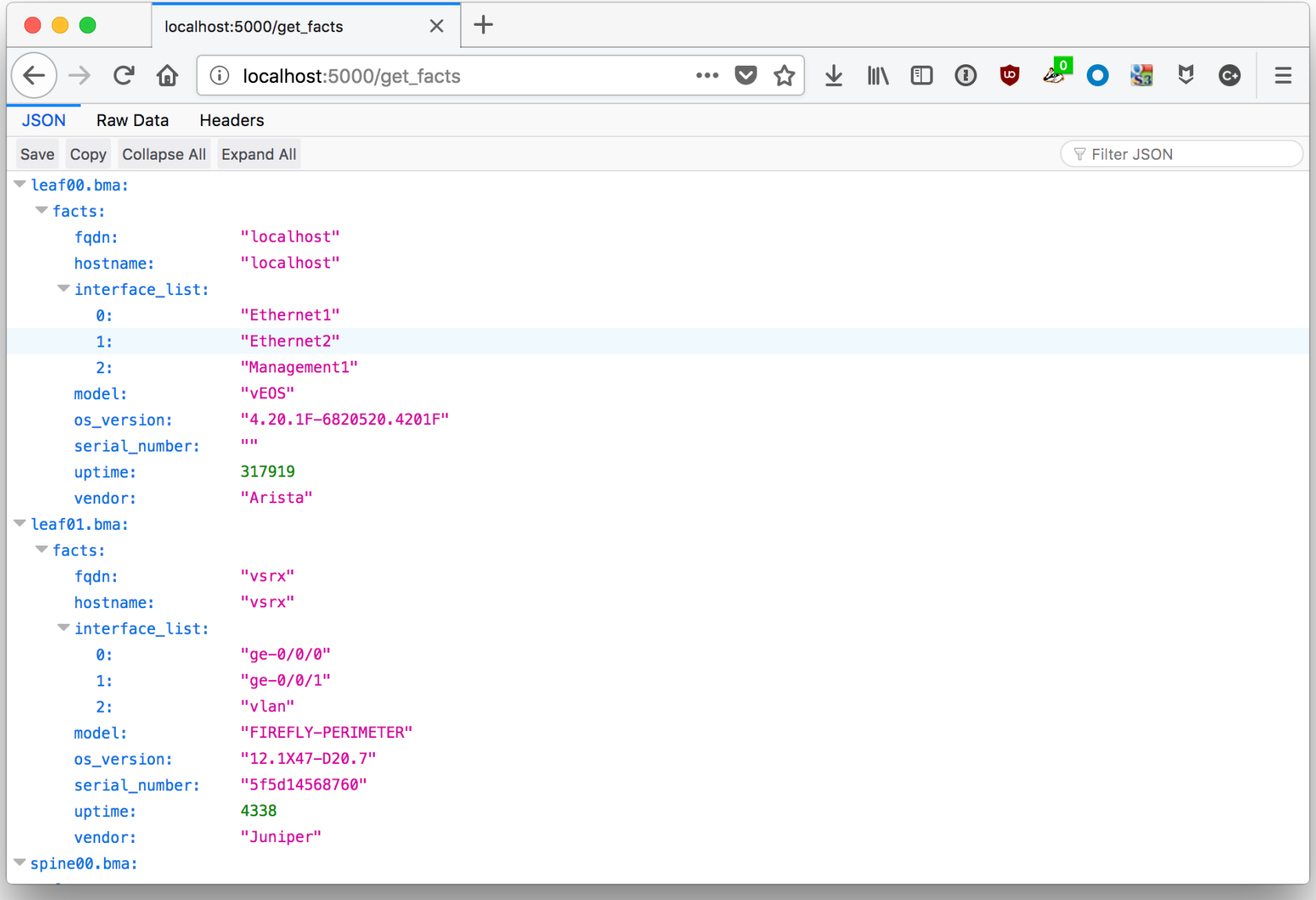
```
!cat 6_integration_examples/napalm_rest.py
```

```
from flask import Flask, jsonify
from nornir import InitNornir
from nornir.plugins.tasks.networking import napalm_get

app = Flask(__name__)


def get_nr():
    return InitNornir(
        inventory={
            "options": {
                "host_file": "inventory/hosts.yaml",
                "group_file": "inventory/groups.yaml",
                "defaults_file": "inventory/defaults.yaml",
            }
        }
    )


def to_json(results):
    return jsonify({host: result[0].result for host, result in results.items()})


@app.route("/get_users")
def get_users():
    nr = get_nr()
    r = nr.run(task=napalm_get, getters=["users"])
    return to_json(r)


@app.route("/get_facts")
def get_facts():
    nr = get_nr()
    r = nr.run(task=napalm_get, getters=["facts"])
    return to_json(r)
```

Run:

```
cd notebooks/6_integration_examples && FLASK_APP=napalm_rest.py flask run
```


localhost:5000/get_users
localhost:5000/get_users
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
leaf00.bma:
users:
admin:
level: 15
password: "1waUQAEpO$kSmdRL84F5sCoWKHZ8/kU1"
sshkeys: []
bob:
level: 15
password: "1waUQAEpO$kSmdRL84F5sCoWKHZ8/kU1"
sshkeys: []
leaf01.bma:
users:
admin:
level: 15
password: "1waUQAEpO$kSmdRL84F5sCoWKHZ8/kU1"
sshkeys: []
bob:
level: 15
password: "1waUQAEpO$kSmdRL84F5sCoWKHZ8/kU1"
sshkeys: []
spine00.bma:
users:
admin:
level: 15
password: "1waUQAEpO$kSmdRL84F5sCoWKHZ8/kU1"
sshkeys: []
bob:


localhost:5000/get_facts
localhost:5000/get_facts
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
leaf00.bma:
facts:
fqdn: "localhost"
hostname: "localhost"
interface_list:
0: "Ethernet1"
1: "Ethernet2"
2: "Management1"
model: "vEOS"
os_version: "4.20.1F-6820520.4201F"
serial_number: ""
uptime: 317919
vendor: "Arista"
leaf01.bma:
facts:
fqdn: "vsrx"
hostname: "vsrx"
interface_list:
0: "ge-0/0/0"
1: "ge-0/0/1"
2: "vlan"
model: "FIREFLY-PERIMETER"
os_version: "12.1X47-D20.7"
serial_number: "5f5d14568760"
uptime: 4338
vendor: "Juniper"
spine00.bma:

Concluding thoughts

While traditional YAML-based automation frameworks have done a great job at abstracting complex tasks and making them accessible, this same abstraction makes complex workflows to be cumbersome and difficult to implement. Integrating with well-known IDEs and frameworks also becomes a daunting task.

Nornir aims at bringing the best of both worlds; by managing the inventory, providing native multithreading and providing useful building blocks without taking python out of the equation you get the same nice abstractions as with other frameworks while retaining the ability of writing complex workflows in native python, being able to integrate natively with other python frameworks and havint the possibility of using modern IDEs to help you write and debug your code.

Useful links

- nornir (https://github.com/nornir-automation/nornir)
- issues (https://github.com/nornir-automation/nornir/issues)
- documentation (https://nornir.readthedocs.io/en/latest/)
- examples (https://github.com/nornir-automation/nornir-tools)
- repo with this slides (https://github.com/dravetech/nornir-workshop)

Questions?

@dbarrosop