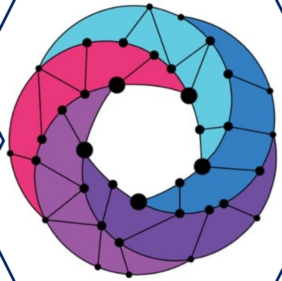
Automating Day-1 and Day-2 VNF Operations with OSM Primitives

What are day-1 and day-2 operations?

How to perform these operations in a VNF?



Day-1 and Day-2 Operations

Day-1

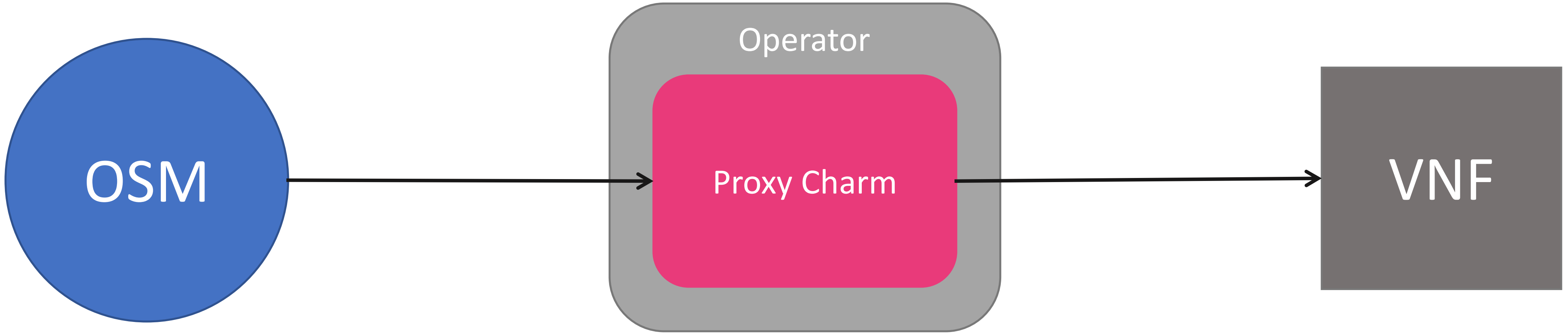
Installation
Setup
Configuration

Day-2

Lights-on
Maintenance
Optimization

How to use Juju Charms to perform VNF day-1 and day-2 operations?

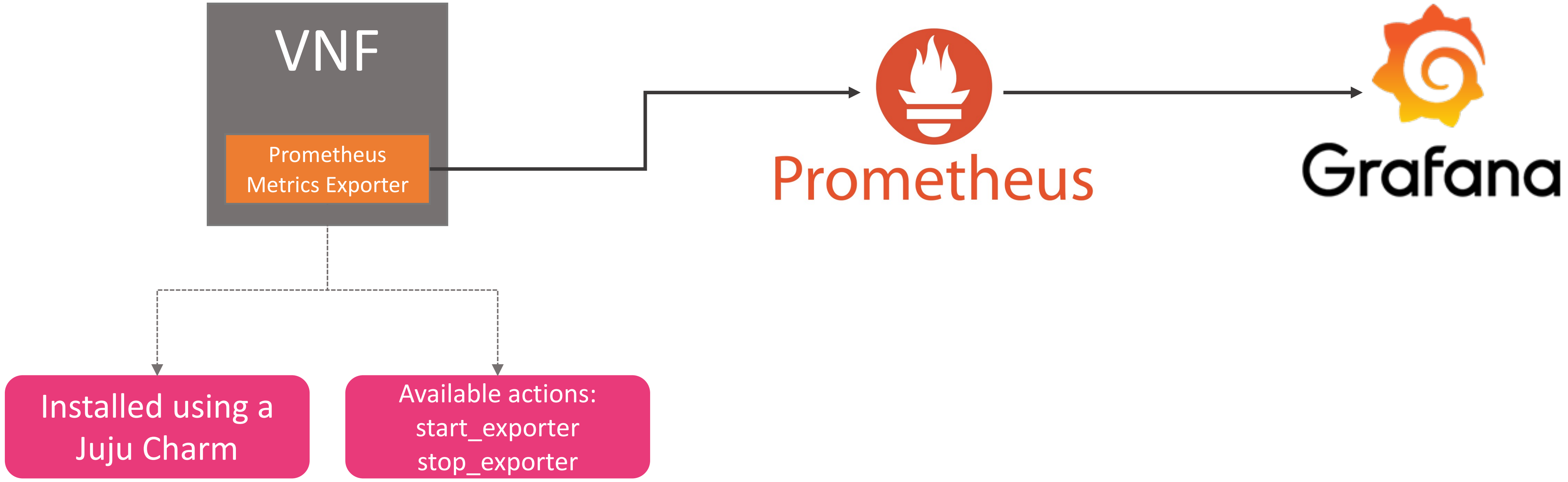
We will use proxy Charms to perform these operations on the VNFs



Adapted from OSM MR10 Hackfest

Our Goal

Create a Juju Charm that makes available a Prometheus Node Metrics Exporter on a VNF. To do so, we will use an OSM SSH Proxy Charm.



Background Concepts

To get the most out of this tutorial you should know:

How to create a VNF

Addressed in our “Build your VNF from Scratch” tutorial

How to develop a Juju Charm

Addressed in our “Introducing OSM Primitives and Juju Charms” tutorial

Base Resources

During this tutorial we will use, as a starting point:

The VNF developed during the “Build your VNF from Scratch” tutorial

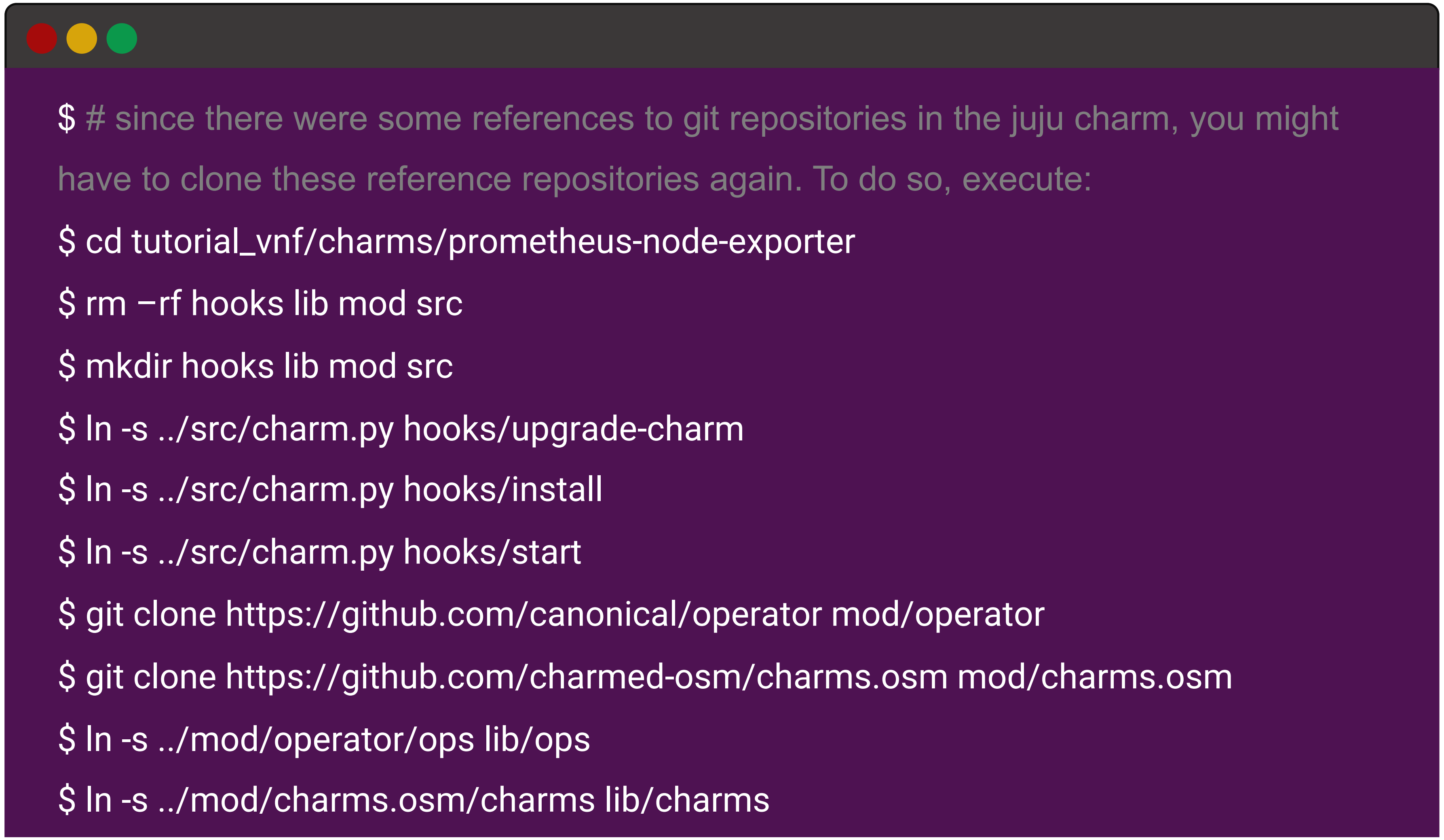
The Juju Charm developed during the “Introducing OSM Primitives and Juju Charms” tutorial

How to use a Juju Charm to perform day-1 and day-2 operations in a VNF?

First of all, start by downloading all the base resources

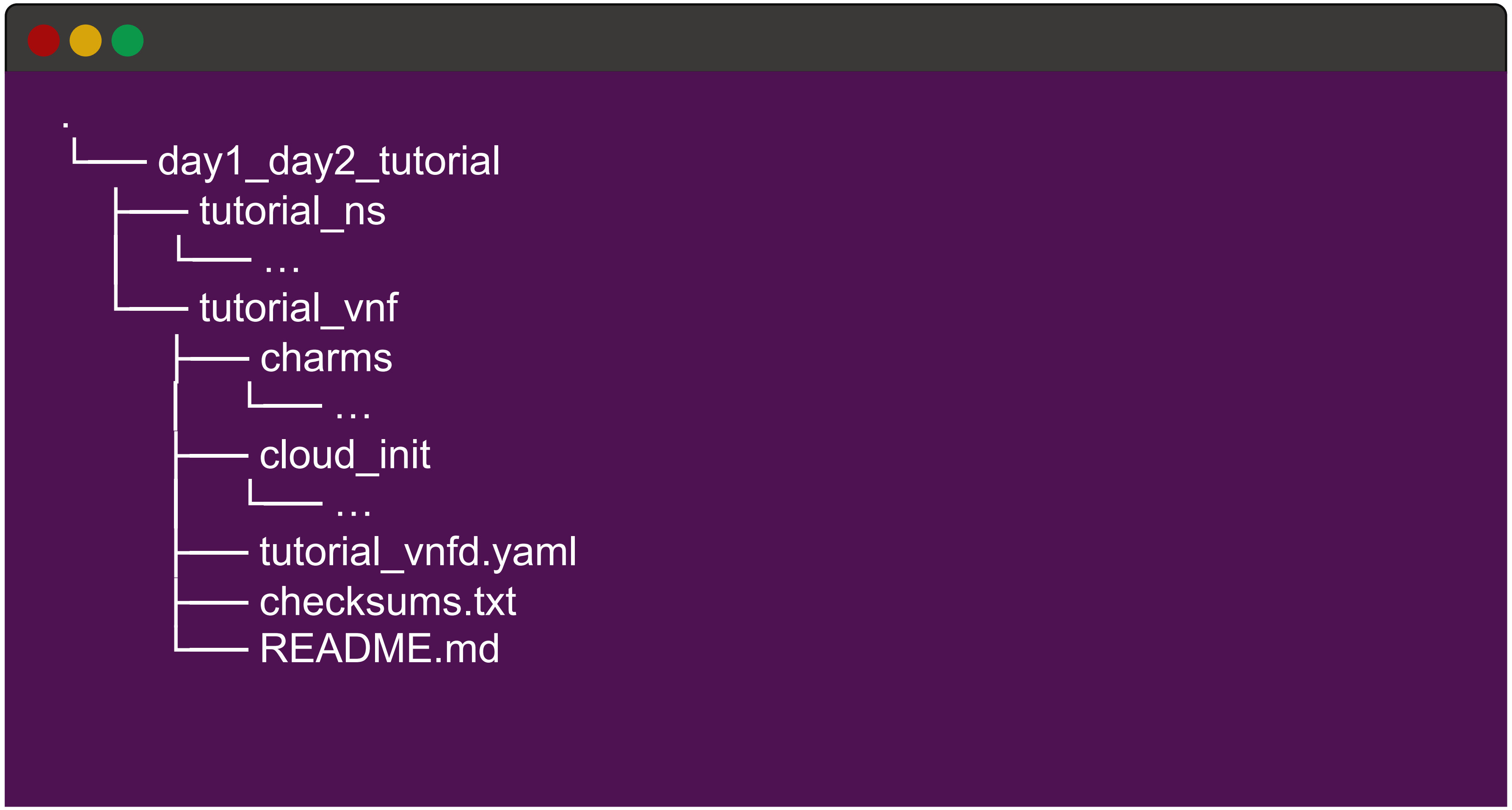
```
$ # download the base VNF and NS
$ curl https://codeload.github.com/5gasp/tutorials/tar.gz/master |  tar -xz --
strip=2 tutorials-master/1-build_your_vnf_from_scratch/outputs
$ # organize your file structure
$ mv outputs/* .
$ rm outputs –d
$ # add the base juju charm to the VNF file structure
$ cd tutorial_vnf
$ curl https://codeload.github.com/5gasp/tutorials/tar.gz/master |  tar -xz --
strip=2 tutorials-master/2-introducing_osm_primitives_and_juju_charms
```

First of all, start by downloading all the base resources



```
$ # since there were some references to git repositories in the juju charm, you might
have to clone these reference repositories again. To do so, execute:
$ cd tutorial_vnf/charms/prometheus-node-exporter
$ rm –rf hooks lib mod src
$ mkdir hooks lib mod src
$ ln -s ../src/charm.py hooks/upgrade-charm
$ ln -s ../src/charm.py hooks/install
$ ln -s ../src/charm.py hooks/start
$ git clone https://github.com/canonical/operator mod/operator
$ git clone https://github.com/charmed-osm/charms.osm mod/charms.osm
$ ln -s ../mod/operator/ops lib/ops
$ ln -s ../mod/charms.osm/charms lib/charms
```

After running these commands, you should have the following file structure



```
.
└── day1_day2_tutorial
    ├── tutorial_ns
    │   └── …
    └── tutorial_vnf
        ├── charms
        │   └── …
        ├── cloud_init
        │   └── …
        ├── tutorial_vnfd.yaml
        ├── checksums.txt
        └── README.md
```

Edit the *tutorial_vnf/tutorial_vnfd.yaml* file. Add the following content:

```
vnfd:
  description: A basic VNF descriptor with one VDU
  df:
  - id: default-df
    ...
    # Juju/LCM Actionns
    lcm-operations-configuration:
      operate-vnf-op-config:
        day1-2:
        - config-primitive:
          - name: start-prometheus-exporter
            execution-environment-ref: configure-vnf
          - name: stop-prometheus-exporter
            execution-environment-ref: configure-vnf
          id: tutorial_vnf
          execution-environment-list:
          - id: configure-vnf
            external-connection-point-ref: vnf-cp0-ext
            juju:
              charm: prometheus_node_exporter
              proxy: true
          config-access:
            ssh-access:
              default-user: ubuntu
              required: true
          initial-config-primitive:
          - execution-environment-ref: configure-vnf
            name: config
            parameter:
            - name: ssh-hostname
              value: <rw_mgmt_ip>
            - name: ssh-username
              value: ubuntu
            - name: ssh-password
              value: tutorial
            seq: 1
  ...
```

We will now add code to support the base primitives invoked by OSM.

Start by going to the charm’s directory (tutorial_vnf/ charms/Prometheus_node_exporter).

Add the following to *actions.yaml:*

```yaml
# Standard OSM functions
start:
  description: "Start the service on the VNF."
stop:
  description: "Stop the service on the VNF."
restart:
  description: "Restart the service on the VNF."
reboot:
  description: "Reboot the VNF virtual machine."
upgrade:
  description: "Upgrade the software on the VNF."
```

We will now add code to support the base primitives invoked by OSM.

Add the following to src/*charm.py:*

```python
class SampleProxyCharm(SSHProxyCharm):
    def __init__(self, framework, key):
        super().__init__(framework, key)

        # Listen to charm events
          ...

        # Listen to the touch action event
          ...
        # Custom actions
          ...

        # OSM actions (primitives)
        self.framework.observe(self.on.start_action, self.on_start_action)
        self.framework.observe(self.on.stop_action, self.on_stop_action)
        self.framework.observe(self.on.restart_action, self.on_restart_action)
        self.framework.observe(self.on.reboot_action, self.on_reboot_action)
        self.framework.observe(self.on.upgrade_action, self.on_upgrade_action)
```

We will now add code to support the base primitives invoked by OSM.

Add the following to src/*charm.py:*

```python
class SampleProxyCharm(SSHProxyCharm):
    def __init__(self, framework, key):
        super().__init__(framework, key)
        ...


    ###############
    # OSM methods #
    ###############
    def on_start_action(self, event):
        """Start the VNF service on the VM."""
        pass

    def on_stop_action(self, event):
        """Stop the VNF service on the VM."""
        pass

    def on_restart_action(self, event):
        """Restart the VNF service on the VM."""
        pass

    def on_reboot_action(self, event):
        """Reboot the VM."""
        if self.unit.is_leader():
          pass

    def on_upgrade_action(self, event):
        """Upgrade the VNF service on the VM."""
        pass
```

Remove all the event.fail(…), event.log(…), and event.set_results(…) calls in *charm.py*. Instead, use a logger.

To enable logging, import the python's logging module:

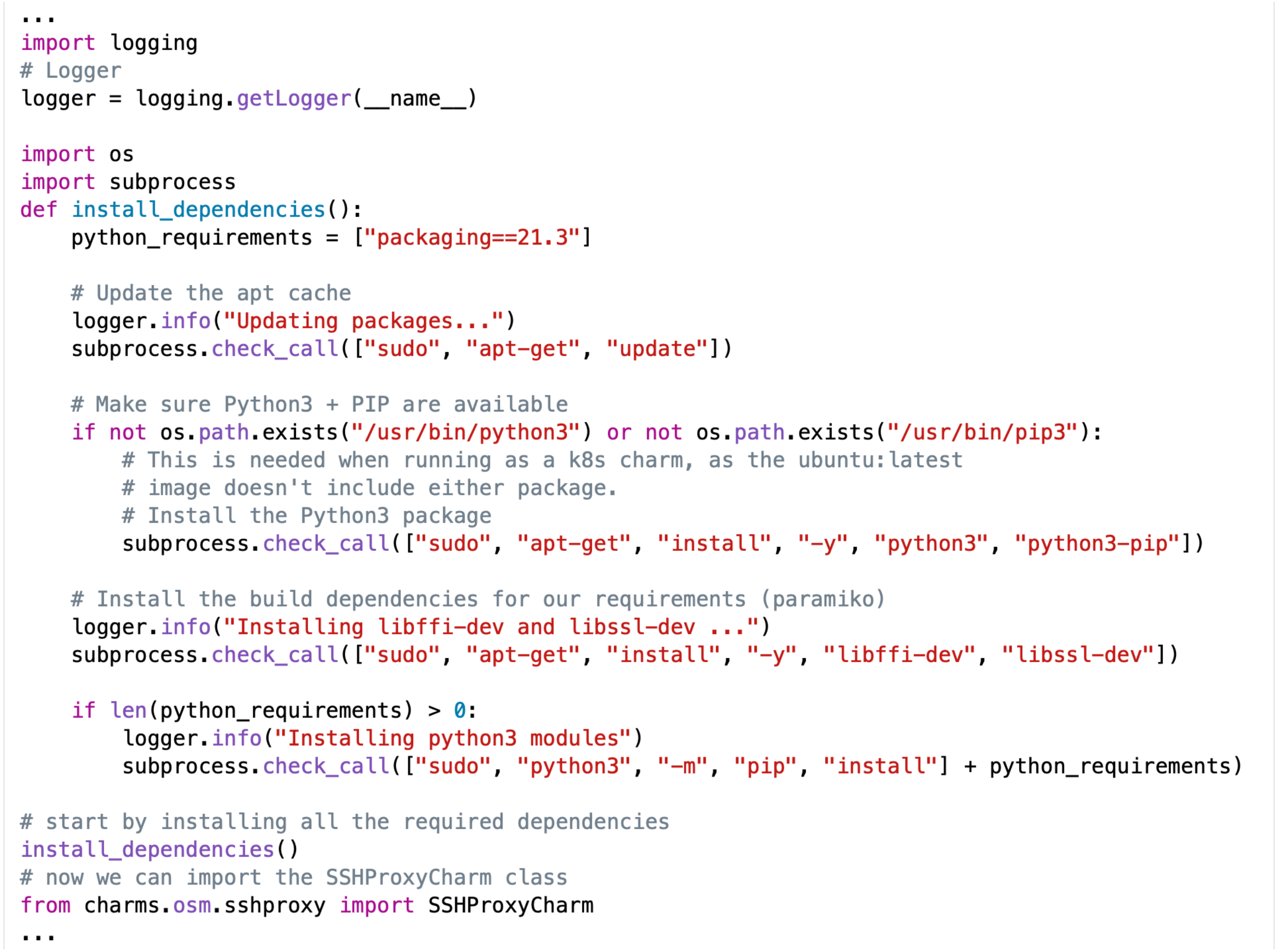
```python
import logging
# Logger
logger = logging.getLogger(__name__)
```

Then you can use ctrl. find&replace to update the following:

- *event.fail(…)* will become *logger.error(…)*
- *event.set_results(…)* will become *logger.info(…)*
- *event.log(…)* will become *logger.info(…)*

The installing of python packages will have to be different than the one used in the juju charm's creation tutorial.

You will have to create a function that runs OS commands to do this.



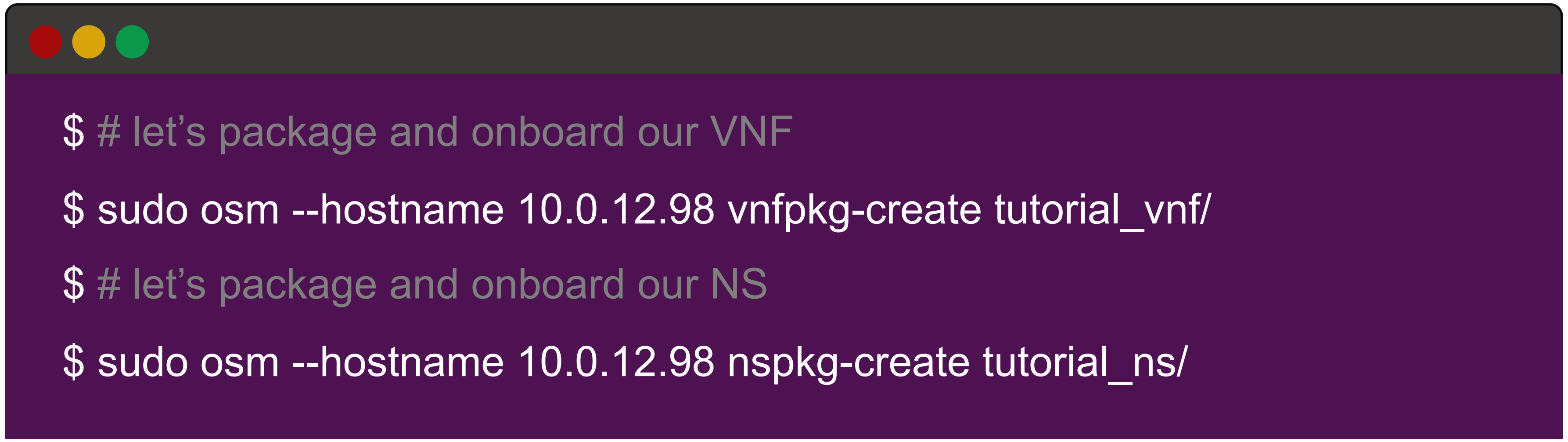
```
...
import logging
# Logger
logger = logging.getLogger(__name__)

import os
import subprocess
def install_dependencies():
    python_requirements = ["packaging==21.3"]

    # Update the apt cache
    logger.info("Updating packages...")
    subprocess.check_call(["sudo", "apt-get", "update"])

    # Make sure Python3 + PIP are available
    if not os.path.exists("/usr/bin/python3") or not os.path.exists("/usr/bin/pip3"):
        # This is needed when running as a k8s charm, as the ubuntu:latest
        # image doesn't include either package.
        # Install the Python3 package
        subprocess.check_call(["sudo", "apt-get", "install", "-y", "python3", "python3-pip"])

    # Install the build dependencies for our requirements (paramiko)
    logger.info("Installing libffi-dev and libssl-dev ...")
    subprocess.check_call(["sudo", "apt-get", "install", "-y", "libffi-dev", "libssl-dev"])

    if len(python_requirements) > 0:
        logger.info("Installing python3 modules")
        subprocess.check_call(["sudo", "python3", "-m", "pip", "install"] + python_requirements)

# start by installing all the required dependencies
install_dependencies()
# now we can import the SSHProxyCharm class
from charms.osm.sshproxy import SSHProxyCharm
...
```

Since we want to automatically install the prometheus node exporter once the VNF is started, we will have to update the *on_start()* function, in *charm.py.*

Update its code to this:

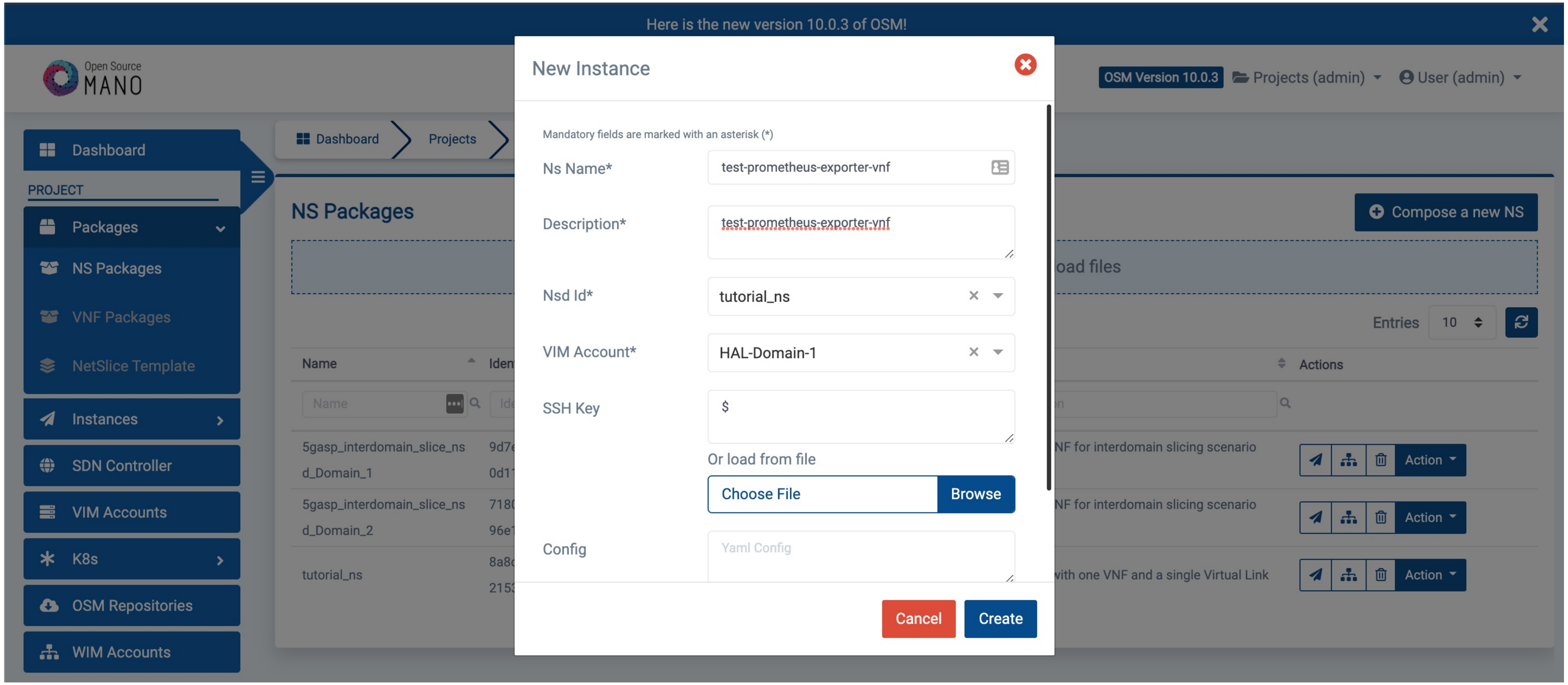
```python
def on_start(self, event):
    """Called when the charm is being started"""
    super().on_start(event)
    # Custom Code
    self.on_start_prometheus_exporter(event)
```

Now, let's create our VNF and NS packages, and onboard them to OSM



```
$ # let's package and onboard our VNF
$ sudo osm --hostname 10.0.12.98 vnfpkg-create tutorial_vnf/
$ # let's package and onboard our NS
$ sudo osm --hostname 10.0.12.98 nspkg-create tutorial_ns/
```

Deploy the NS


Here is the new version 10.0.3 of OSM!

New Instance

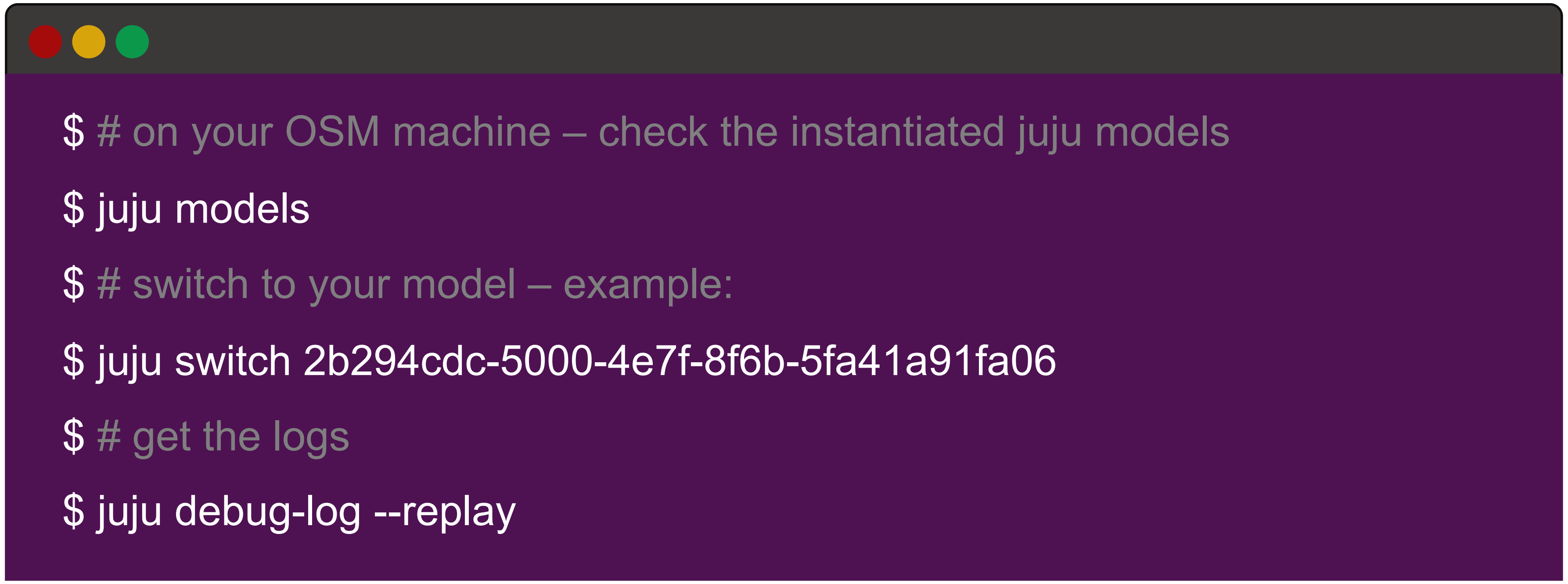
Mandatory fields are marked with an asterisk (*)

Ns Name*	test-prometheus-exporter-vnf
Description*	test-prometheus-exporter-vnf
Nsd Id*	tutorial_ns
VIM Account*	HAL-Domain-1
SSH Key	$
	Or load from file
Config	Yaml Config

NS Packages

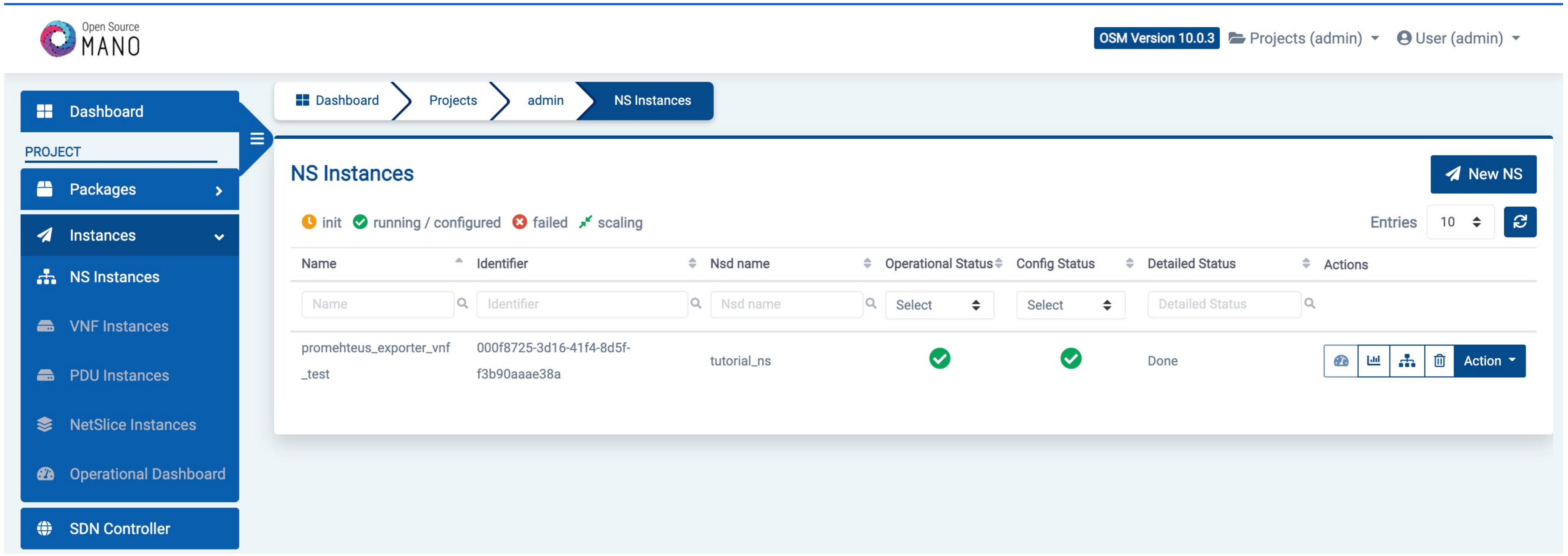
Name	Iden		Actions
5gasp_interdomain_slice_ns d_Domain_1	9d7 0d1	NF for interdomain slicing scenario	Action
5gasp_interdomain_slice_ns d_Domain_2	718 96e	NF for interdomain slicing scenario	Action
tutorial_ns	8a8 215	with one VNF and a single Virtual Link	Action

If you want to do some debug…



```
$ # on your OSM machine – check the instantiated juju models
$ juju models
$ # switch to your model – example:
$ juju switch 2b294cdc-5000-4e7f-8f6b-5fa41a91fa06
$ # get the logs
$ juju debug-log --replay
```

If everything goes accordingly, you should have a green icon in the NS's operational status and config status



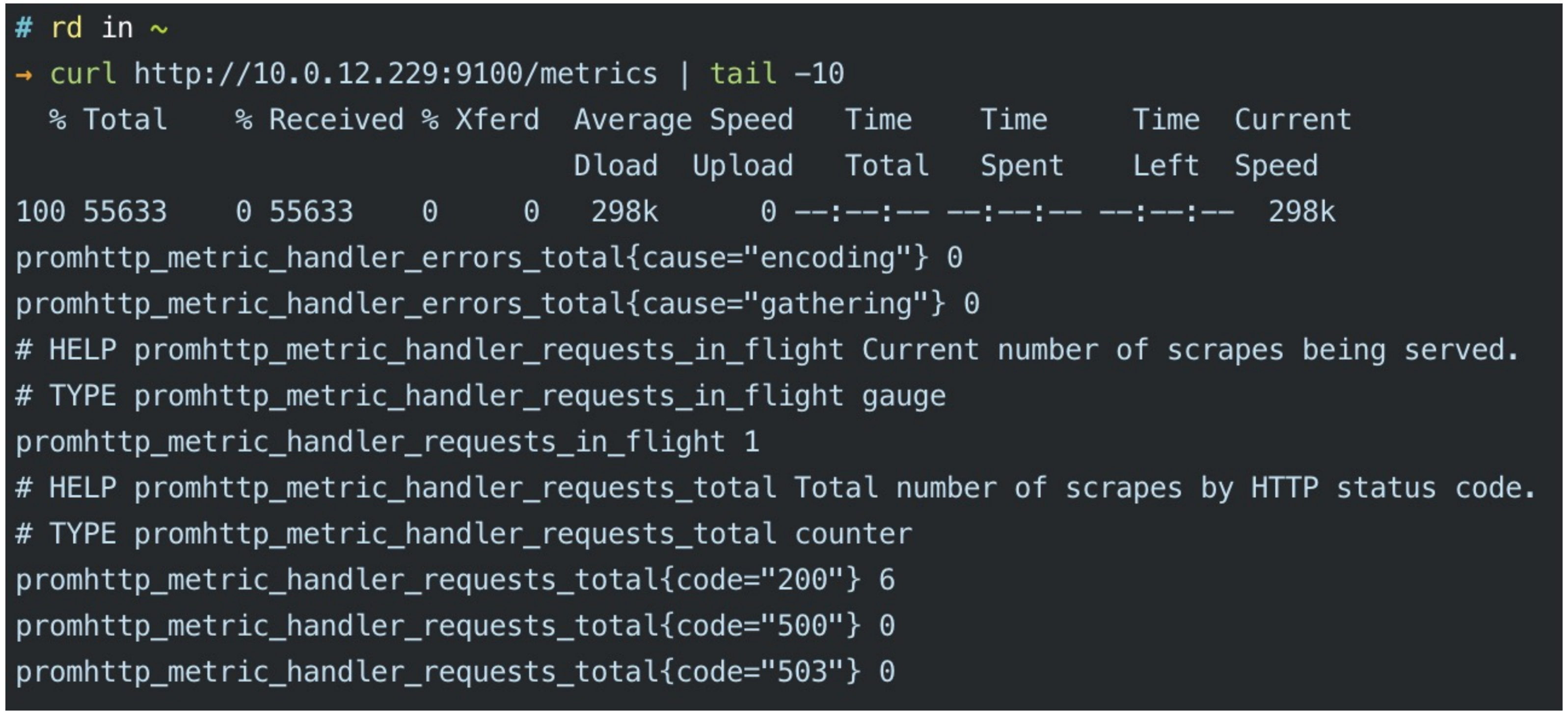
NS Instances

● init ✓ running / configured ✗ failed scaling

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
promehteus_exporter_vnf_test	000f8725-3d16-41f4-8d5f-f3b90aaae38a	tutorial_ns	✓	✓	Done	Action

Now, let’s test if the charm performed the desired operations…

To do so, execute a curl to the metrics endpoint, and verify if there are metrics being collected.

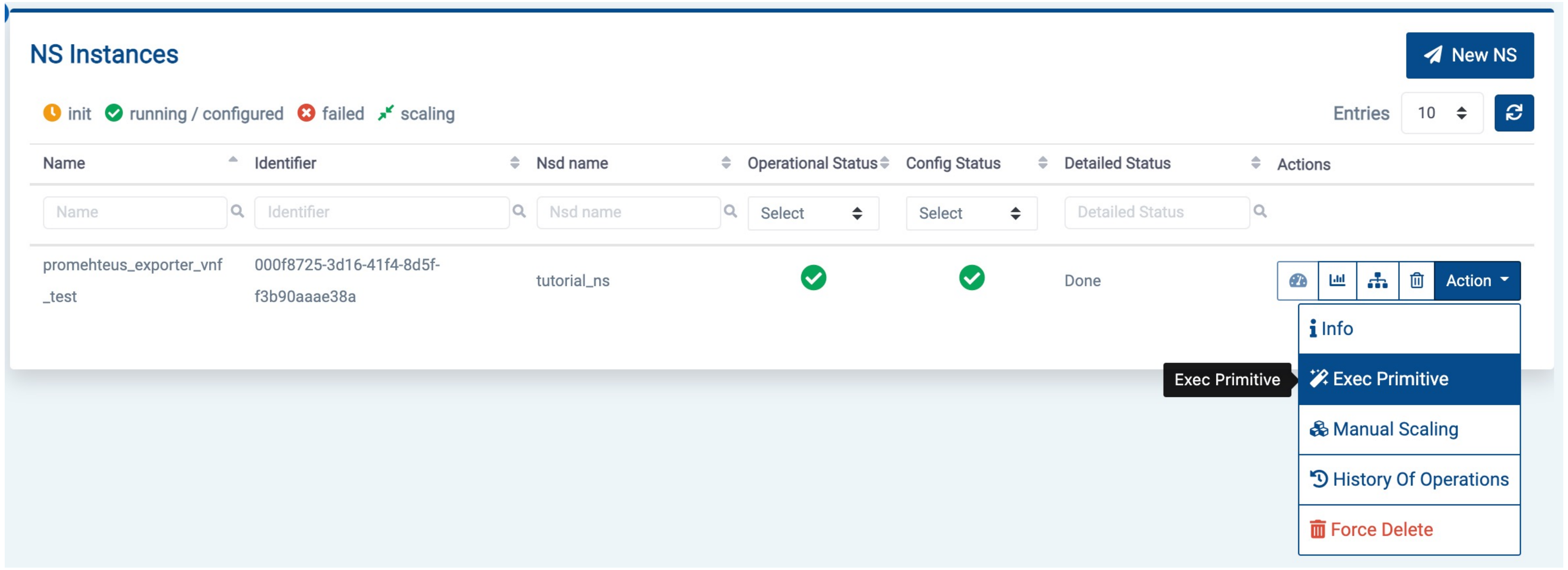


```
# rd in ~
→ curl http://10.0.12.229:9100/metrics | tail -10
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 55633    0 55633    0     0   298k      0 --:--:-- --:--:-- --:--:--  298k
promhttp_metric_handler_errors_total{cause="encoding"} 0
promhttp_metric_handler_errors_total{cause="gathering"} 0
# HELP promhttp_metric_handler_requests_in_flight Current number of scrapes being served.
# TYPE promhttp_metric_handler_requests_in_flight gauge
promhttp_metric_handler_requests_in_flight 1
# HELP promhttp_metric_handler_requests_total Total number of scrapes by HTTP status code.
# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 6
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
```

Success!

You can try to execute OSM primitives, via the OSM UI

Let’s invoke the stop-prometheus-exporter primitive



NS Instances

New NS

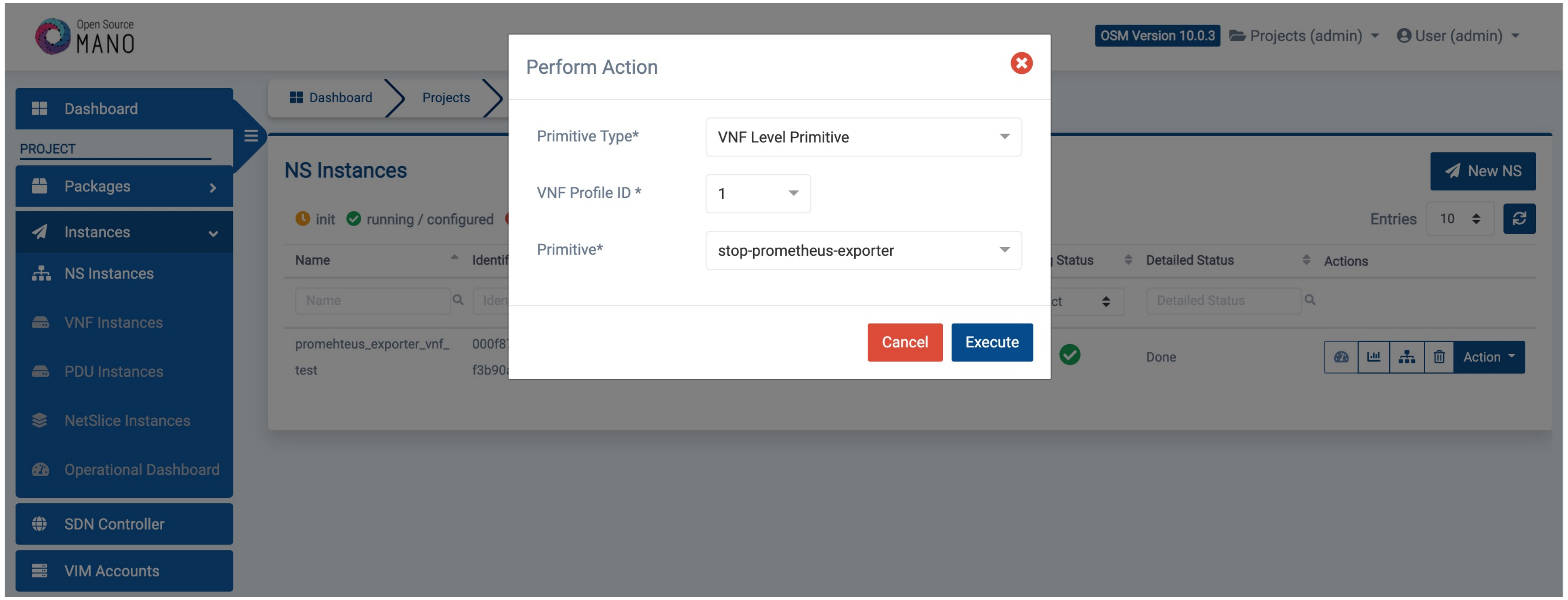
● init ✓ running / configured ✗ failed scaling

Entries 10

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
promehteus_exporter_vnf_test	000f8725-3d16-41f4-8d5f-f3b90aaae38a	tutorial_ns	✓	✓	Done	Action

You can try to execute OSM primitives, via the OSM UI

Let’s invoke the stop-prometheus-exporter primitive



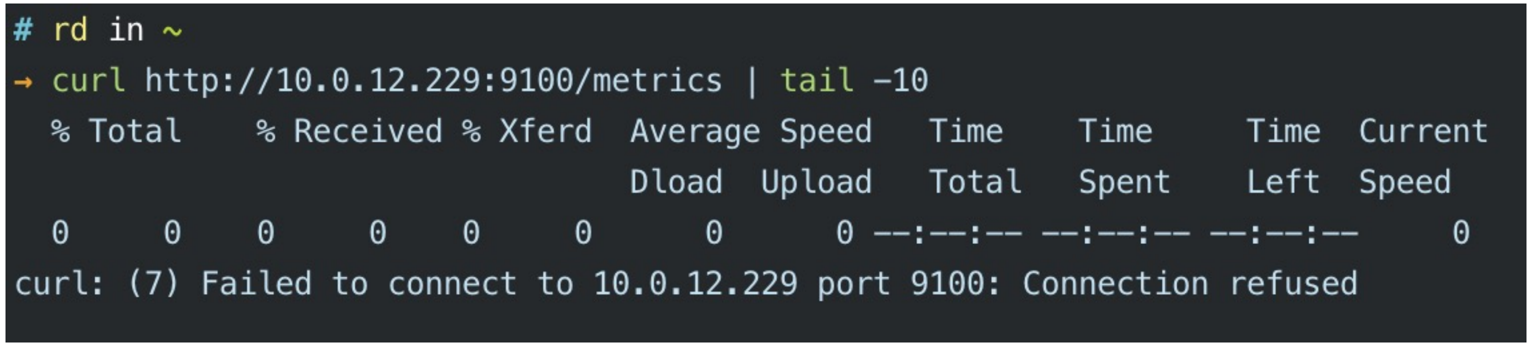
Perform Action

Primitive Type*	VNF Level Primitive
VNF Profile ID *	1
Primitive*	stop-prometheus-exporter

Cancel Execute

You can try to execute OSM primitives, via the OSM UI

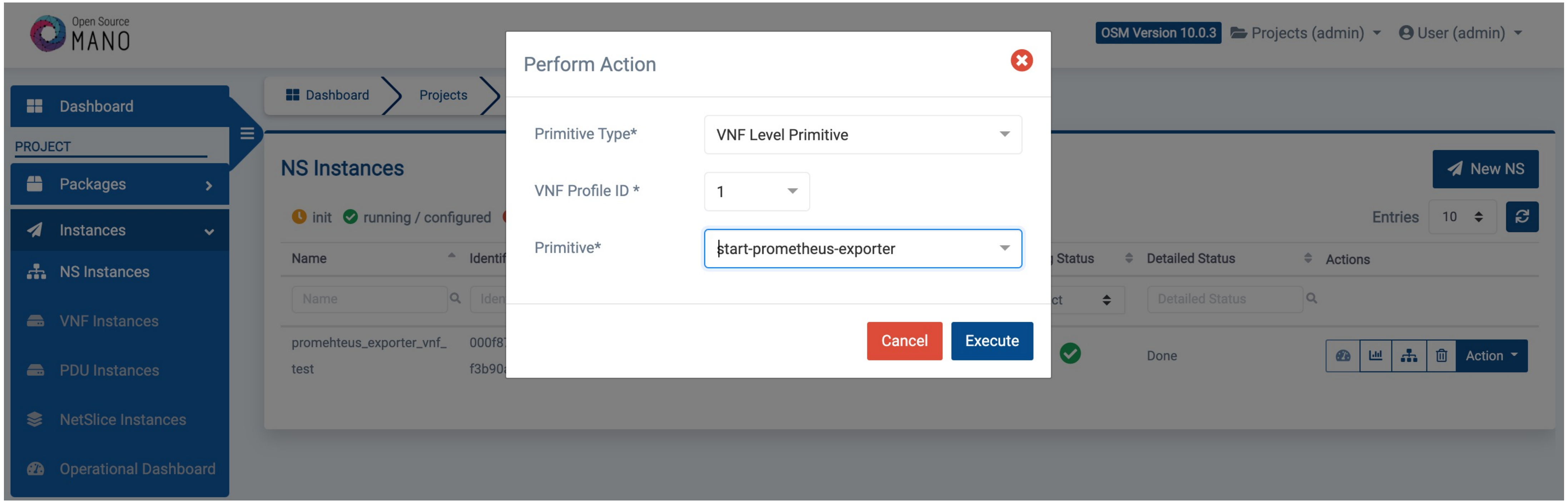
Let’s invoke the stop-prometheus-exporter primitive



```
# rd in ~
→ curl http://10.0.12.229:9100/metrics | tail -10
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (7) Failed to connect to 10.0.12.229 port 9100: Connection refused
```

Success!

Restart the Prometheus Exporter

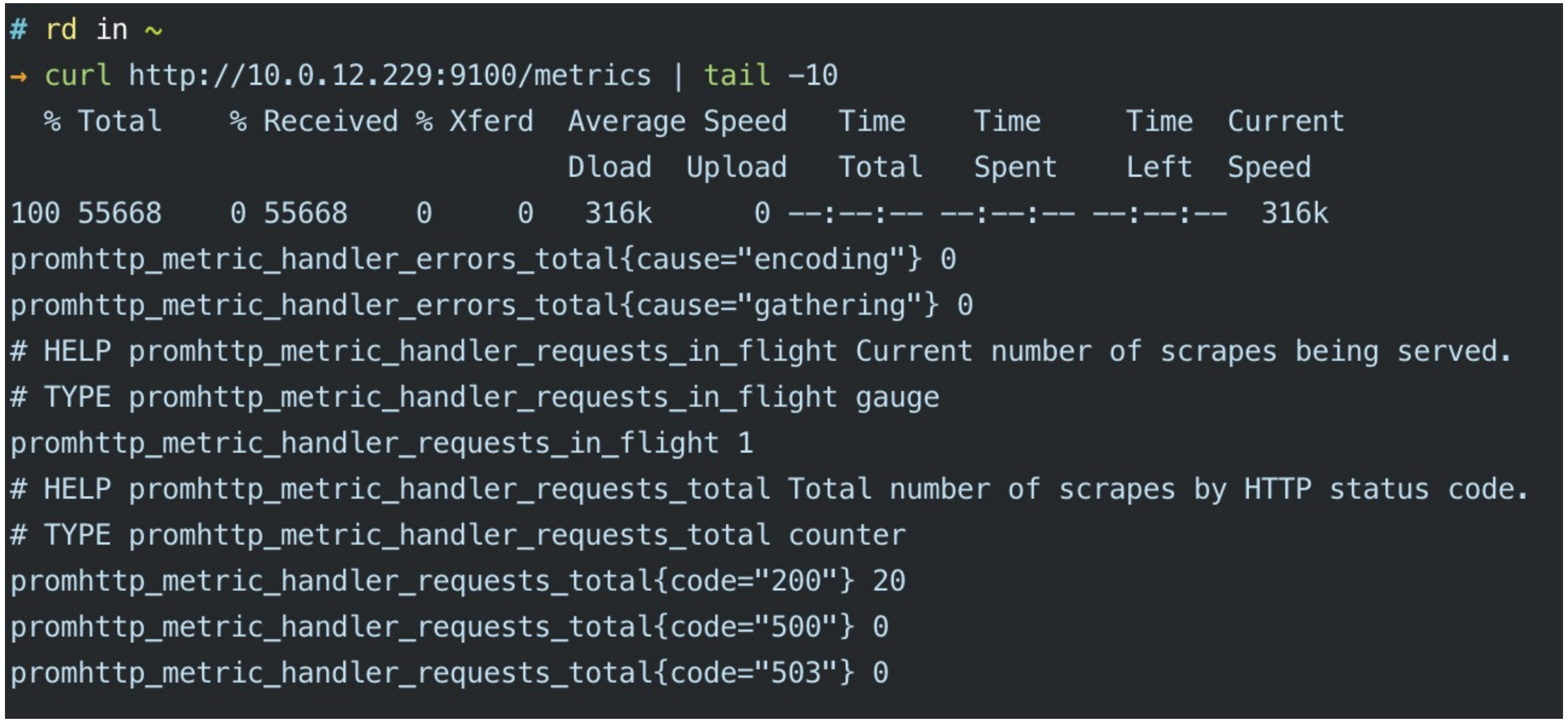


Perform Action

Primitive Type*	VNF Level Primitive
VNF Profile ID *	1
Primitive*	start-prometheus-exporter

Cancel Execute

Restart the Prometheus Exporter



```
# rd in ~
→ curl http://10.0.12.229:9100/metrics | tail -10
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 55668    0 55668    0     0   316k      0 --:--:-- --:--:-- --:--:--  316k
promhttp_metric_handler_errors_total{cause="encoding"} 0
promhttp_metric_handler_errors_total{cause="gathering"} 0
# HELP promhttp_metric_handler_requests_in_flight Current number of scrapes being served.
# TYPE promhttp_metric_handler_requests_in_flight gauge
promhttp_metric_handler_requests_in_flight 1
# HELP promhttp_metric_handler_requests_total Total number of scrapes by HTTP status code.
# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 20
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
```

Success!

The code developed during this tutorial is available at

https://github.com/5gasp/tutorials

If you have any questions regarding the contents addressed in this tutorial you can send an e-mail to Rafael Direito rdireito@av.it.pt, or contact us via contact@5gasp.eu