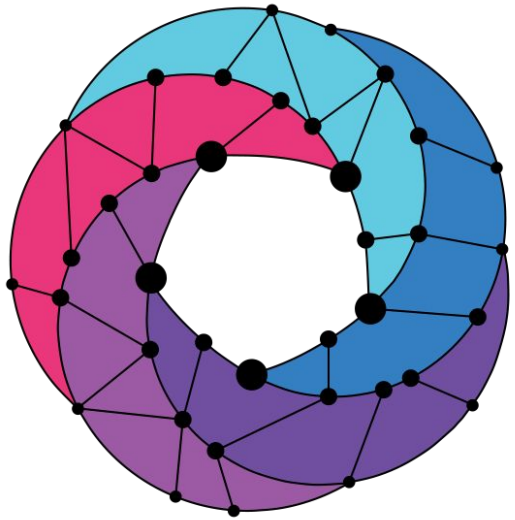


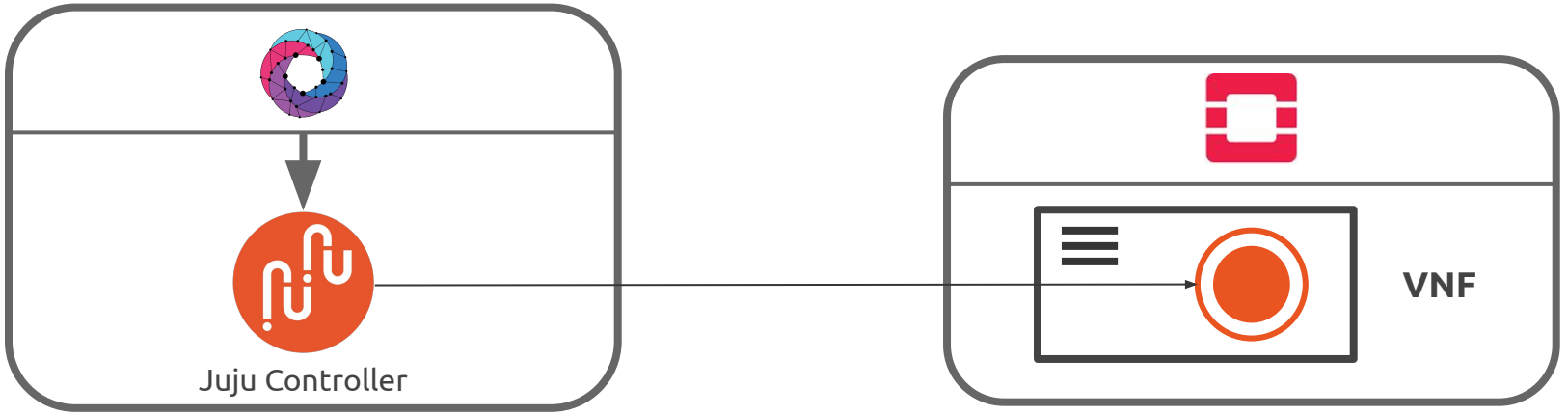
Open Source
MANO

OSM Hackfest – NEW
Native charms and Charm Tech framework

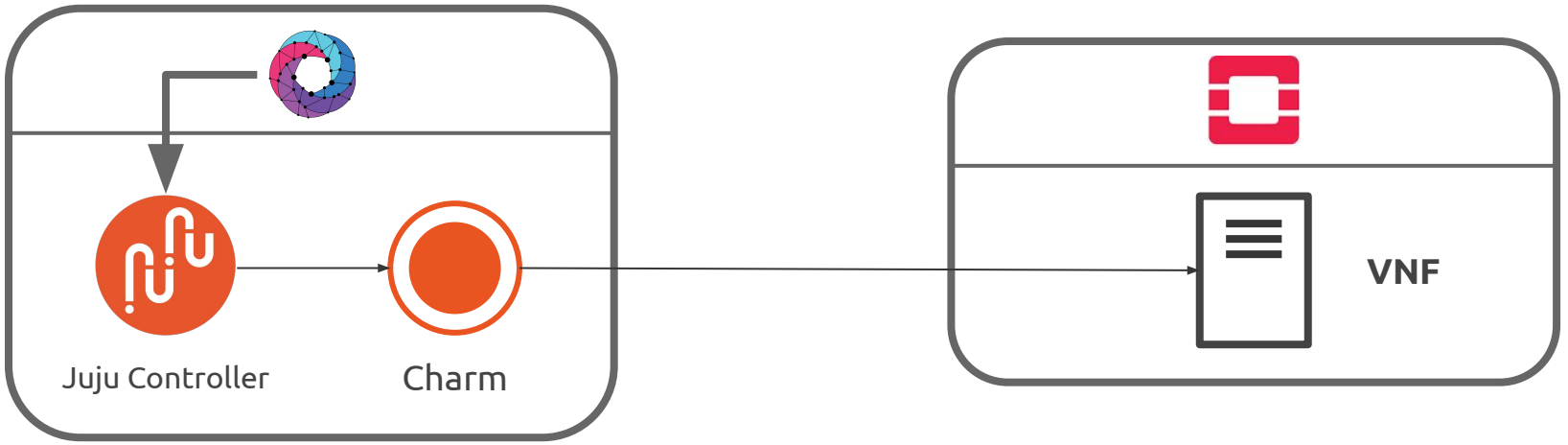
Dominik Fleischmann, David Garcia, Tytus Kurek (Canonical)

ETSI

Native Charms vs Proxy Charms



NATIVE CHARMS



PROXY CHARMS

Objective

- Build a Native Charm
  - Install apt packages (hello)
  - Install a snap (microk8s)
  - Configure microk8s
  - Implement action
- Build the VNF and NS Descriptors
- Instantiate and invoke actions

Open Source
MANO

Build a Native Charm

bit.ly/native_charm

ETSI

Build a Native Charm

Open Source MANO

Prepare environment:

```
mkdir -p ~/charms/build
export CHARM_BUILD_DIR=~/charms/build
```

Create a charm:

```
sudo snap install charm --classic
cd charms
charm create native-charm
cd native-charm
```

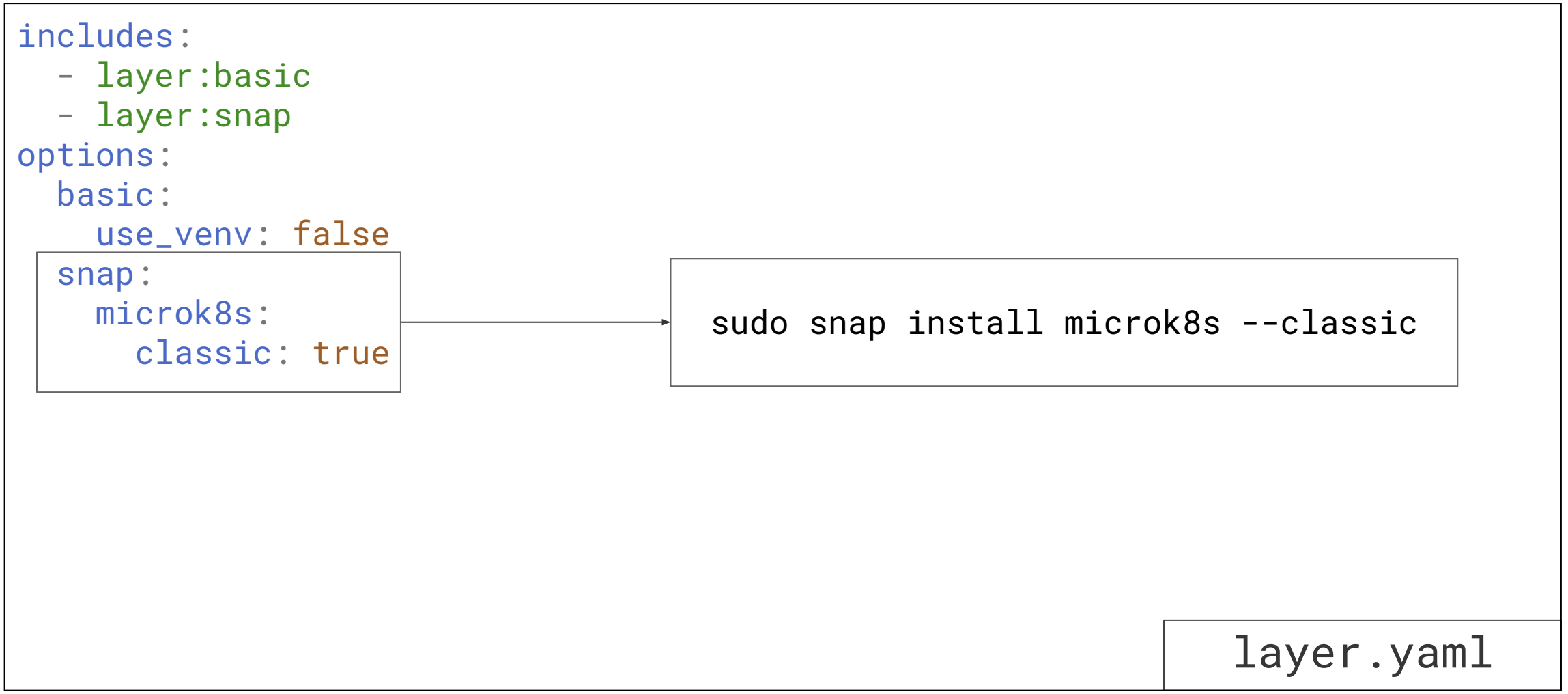
Build a Native Charm *(charms/native-charm)*

Open Source MANO

```yaml
name: native-charm
summary: Basic Native charm example
maintainer: David Garcia <david.garcia@canonical.com>
description: |
  This is a basic native charm example,
  which shows off how to
  - Install apt packages
  - Install snaps
  - Execute actions
tags:
  - basic
subordinate: false
series: ['xenial', 'bionic']
```

metadata.yaml

Build a Native Charm *(charms/native-charm)*



Build a Native Charm *(charms/native-charm)*

Open Source MANO

```yaml
microk8s-cmd:
  description: "Execute microk8s command (microk8s.<command>)"
  params:
    command:
      description: "The command to execute"
      type: string
      default: ""
  required:
  - command
```

actions.yaml

Build a Native Charm *(charms/native-charm)*

Open Source MANO

```
#!/usr/local/sbin/charm-env python3
import sys
sys.path.append('lib')
from charms.reactive import main, set_flag
from charmhelpers.core.hookenv import action_fail, action_name

set_flag('actions.{}'.format(action_name()))

try:
    main()
except Exception as e:
    action_fail(repr(e))
```

```
mkdir actions
vim actions/microk8s-cmd
chmod +x actions/microk8s-cmd
```

actions/microk8s-cmd

Build a Native Charm *(charms/native-charm)*

Open Source MANO

```
from charms.reactive import (
    when,
    when_not,
    set_flag,
    clear_flag
)
from charmhelpers.core.hookenv import (
    action_get,
    action_set,
    action_fail,
    status_set
)
from charmhelpers.fetch import apt_install
import subprocess
import traceback
import logging

[...]
```

reactive/native_charm.py

Build a Native Charm *(charms/native-charm)*

Open Source MANO

```
[...]
@when('snap.installed.microk8s')
def config_microk8s():
    subprocess.call(
        'sudo usermod -a -G microk8s ubuntu',
        shell=True
    )
    set_flag('native-charm.configured')


@when('native-charm.configured')
@when_not('native-charm.installed')
def install_native_charm():
    apt_install('hello')
    set_flag('native-charm.installed')
    status_set('active', 'Ready!')

[...]
```

reactive/native_charm.py

Build a Native Charm *(charms/native-charm)*

Open Source MANO

```
[...]
@when('actions.microk8s-cmd', 'native-charm.installed')
def microk8s_cmd():
    try:
        command = action_get('command')
        output, _ = subprocess.check_output(
            'microk8s.{}'.format(command),
            shell=True
        )
    except Exception as e:
        action_fail('command failed: {}'.format(e))
        logging.error('Traceback: {}'.format(traceback.format_exc()))
    else:
        action_set({'output': output})
    finally:
        clear_flag('actions.microk8s-cmd')
```

reactive/native_charm.py

Build a Native Charm

Open Source MANO

Build charm:

```
cd ~/charms
charm build native-charm
```

Open Source
MANO

Build the VNF and NS Descriptors

bit.ly/native_charm_nsd

bit.ly/native_charm_vnfd

ETSI

Build the VNF and NS Descriptors

Open Source MANO

Extract descriptors:

```
cd ~
wget bit.ly/native_charm_vnfd -O native_charm_vnfd.tar.gz
wget bit.ly/native_charm_nsd -O native_charm_nsd.tar.gz
tar xf native_charm_vnfd.tar.gz
tar xf native_charm_nsd.tar.gz
```

Build the VNF Descriptor *(native_vnf/)*

```
vnfd:vnfd-catalog:
   vnfd:
   -   id: native-vnf
       vdu:
       -   id: mgmtVM
           name: mgmtVM
           cloud-init-file: cloud-config.txt
           vdu-configuration:
               juju:
                   charm: native-charm
                   proxy: False
               config-access:
                   ssh-access:
                       required: True
                       default-user: ubuntu
               initial-config-primitive:
               -   seq: '1'
                   name: microk8s-cmd
                   parameter:
                   -   name: command
                       value: status
               config-primitive:
               -   name: microk8s-cmd
                   parameter:
                   -   name: command
                       data-type: STRING
```

```
juju:
    charm: native-charm
    proxy: False
```

native_vnfd.yaml

Build the VNF Descriptor *(native_vnf/)*

```
vnfd:vnfd-catalog:
   vnfd:
   -   id: native-vnf
       vdu:
       -   id: mgmtVM
           name: mgmtVM
           cloud-init-file: cloud-config.txt
           vdu-configuration:
               juju:
                   charm: native-charm
                   proxy: False
               config-access:
                   ssh-access:
                       required: True
                       default-user: ubuntu
               initial-config-primitive:
               -   seq: '1'
                   name: microk8s-cmd
                   parameter:
                   -   name: command
                       value: status
               config-primitive:
               -   name: microk8s-cmd
                   parameter:
                   -   name: command
                       data-type: STRING
```

```
config-access:
    ssh-access:
        required: True
        default-user: ubuntu
```

native_vnfd.yaml

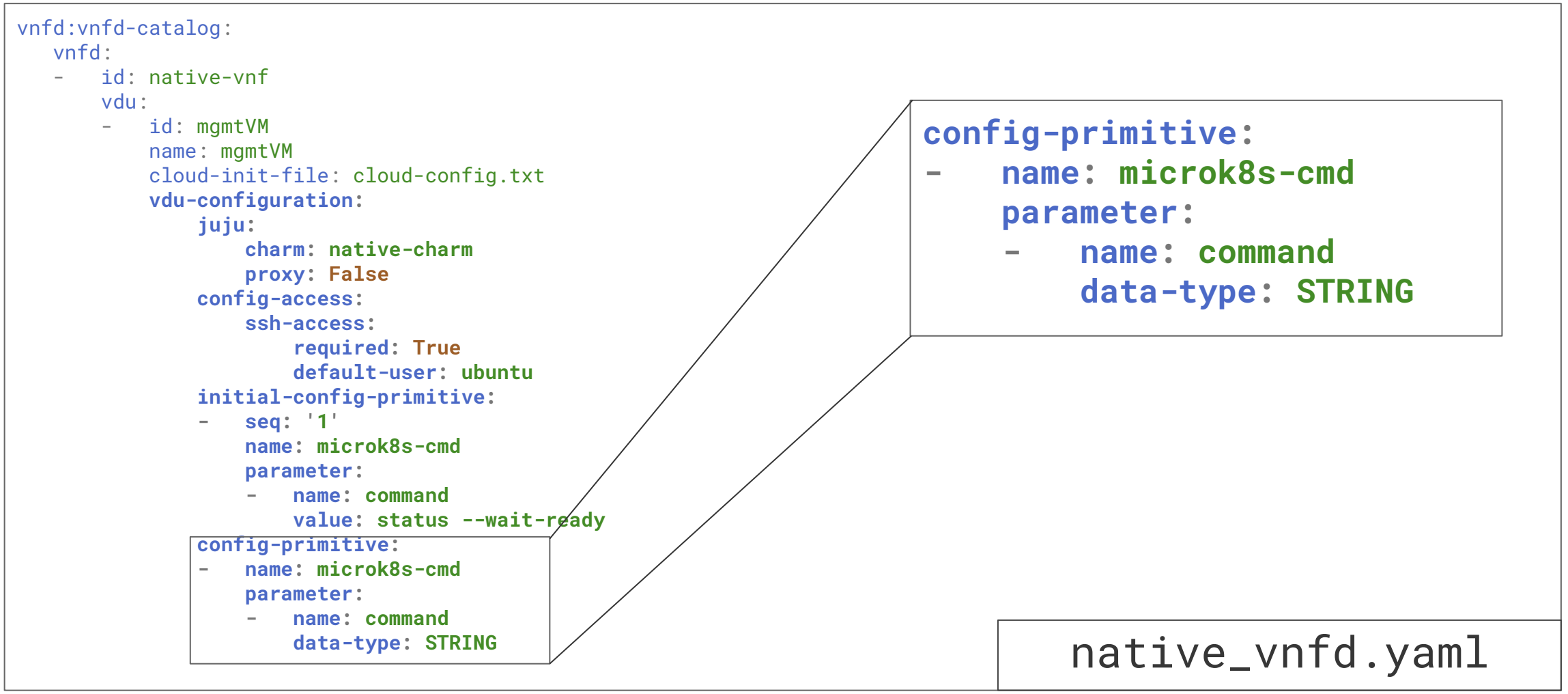
Build the VNF Descriptor *(native_vnf/)*

```
vnfd:vnfd-catalog:
   vnfd:
   -   id: native-vnf
       vdu:
       -   id: mgmtVM
           name: mgmtVM
           cloud-init-file: cloud-config.txt
           vdu-configuration:
               juju:
                   charm: native-charm
                   proxy: False
               config-access:
                   ssh-access:
                       required: True
                       default-user: ubuntu
               initial-config-primitive:
               -   seq: '1'
                   name: microk8s-cmd
                   parameter:
                   -   name: command
                       value: status
               config-primitive:
               -   name: microk8s-cmd
                   parameter:
                   -   name: command
                       data-type: STRING
```

```
initial-config-primitive:
-   seq: '1'
    name: microk8s-cmd
    parameter:
    -   name: command
        value: status
```

native_vnfd.yaml

Build the VNF Descriptor *(native_vnf/)*



Build the VNF and NS Descriptors

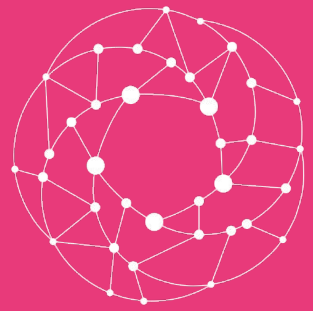
Open Source MANO

Replace charm:

```
rm -rf native_vnf/charms/native-charm
cp -R charms/build/native-charm \
      native_vnf/charms/native-charm
```

Build packages:

```
tar cfz native_vnfd.tar.gz native_vnf/
tar cfz native_nsd.tar.gz  native_ns/
```



Open Source
MANO

Instantiate and invoke actions

ETSI

Instantiate and invoke actions

Upload packages:

```
osm upload-package native_vnfd.tar.gz
osm upload-package native_nsd.tar.gz
```

Instantiate NS:

```
osm ns-create --ns_name native_charms \
              --nsd_name native-ns \
              --vim_account openstack
```

Instantiate and invoke actions

Open Source MANO

Wait until NS is up:

```
watch osm ns-list
```

```
+------------------+--------------------------------------+--------------------+---------------+-----------------+
| ns instance name | id                                   | operational status | config status | detailed status |
+------------------+--------------------------------------+--------------------+---------------+-----------------+
| native_charms    | b13e31f3-5ee5-4955-9d6b-12a0f832473e | running            | configured    | done            |
+------------------+--------------------------------------+--------------------+---------------+-----------------+
```

Switch to juju model:

```
juju switch b13e31f3-5ee5-4955-9d6b-12a0f832473e
```

Instantiate and invoke actions

Open Source MANO

Juju status:

```
juju status
```

```
Model                                  Controller  Cloud/Region         Version  SLA          Timestamp
B13e31f3-5ee5-4955-9d6b-12a0f832473e   osm         localhost/localhost  2.6.10   unsupported  12:26:48Z

App                          Version  Status  Scale  Charm              Store  Rev  OS     Notes
native-charms-b-mgmtvm-aa             active      1  native-charm       local    0  ubuntu

Unit                            Workload  Agent  Machine  Public address  Ports  Message
native-charms-b-mgmtvm-aa/0*    active    idle   0        172.21.248.21          Ready!

Machine  State    DNS           Inst id                Series  AZ  Message
0        started  172.21.248.21  manual:172.21.248.21  xenial      Manually provisioned machine
```

Instantiate and invoke actions

Open Source MANO

Day-1 status:

```
juju show-action-status
```

```
actions:
- action: microk8s-cmd
  completed at: "2019-11-08 12:25:27"
  id: 5071d6f8-0faf-4280-80fa-31ac6ddfa868
  status: completed
  unit: native-charms-b-mgmtvm-aa/0
```

Day-1 output:

```
juju show-action-output 5071d6f8-0faf-...
```

```
results:
  output: 'b''microk8s is running\naddons:\nrbac: disabled\ningress: disabled\ndns:
    disabled\nmetrics-server: disabled\nlinkerd: disabled\nprometheus: disabled\nistio:
    disabled\njaeger: disabled\nfluentd: disabled\ngpu: disabled\nstorage: disabled\ndashboard:
    disabled\nregistry: disabled\n'''
status: completed
timing:
  completed: 2019-11-08 12:25:27 +0000 UTC
  enqueued: 2019-11-08 12:23:20 +0000 UTC
  started: 2019-11-08 12:25:22 +0000 UTC
```

Instantiate and invoke actions

Day-2 actions:

```
juju run-action native-charms-b-mgmtvm-aa/0 \
     microk8s-cmd command='kubectl get namespaces'

juju run-action native-charms-b-mgmtvm-aa/0 \
     microk8s-cmd command='enable dns'
```

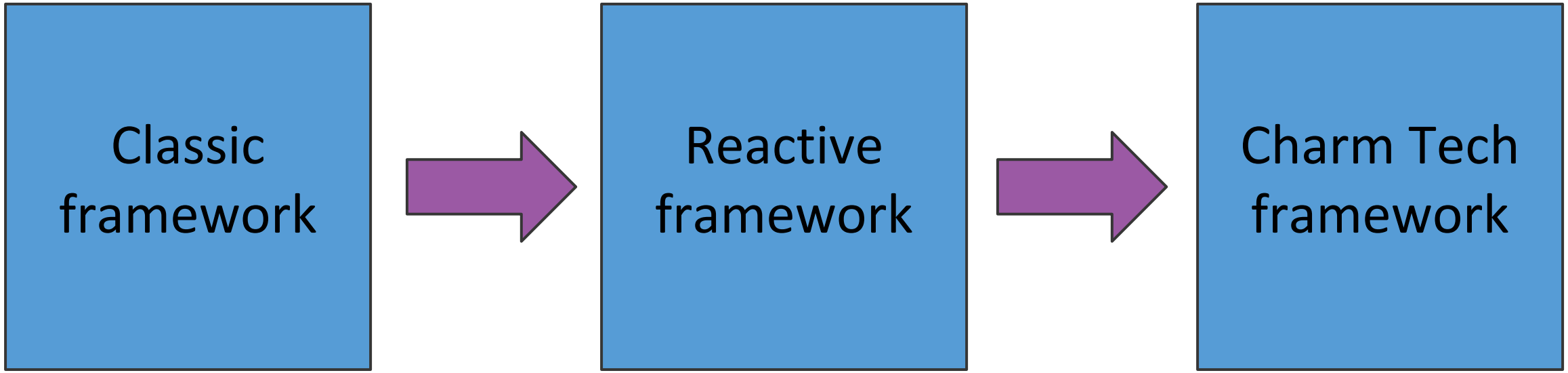
Open Source
MANO

Charm Tech framework

ETSI

Charming used to be tricky …

- … but it is no longer the case!
- Charm Tech framework is coming with Juju 2.7
- Event-based framework (like Java, Swing, Javascript, etc.)



Example: Classic framework

```
...
hooks = Hooks()

@hooks.hook('install')
def install():
   snap_install('microk8s')

@hooks.hook('configure')
def configure():
    cmd = 'sudo usermod -a -G microk8s ubuntu'
    subprocess.call(cmd, shell=True)

def snap_install():
    ...
```

hooks/hooks.py

- Simple, but how do you pass data between the hooks?
- What if the `snap_install` function fails?

Example: Reactive framework

```
...
@when_not('microk8s.installed')
def install_microk8s():
    if snap_install('microk8s'):
        set_flag('microk8s.installed')

@when('microk8s.installed')
@when_not('microk8s.configured')
def configure_microk8s():
    cmd = 'sudo usermod -a -G microk8s ubuntu'
    subprocess.call(cmd, shell=True)
    set_flag('microk8s.configured')

def snap_install():
    ...
```

reactive/charm.py

- This solves the problem, but can become complex very quickly

Example: Charm Tech framework

Open Source MANO

```
...
class Charm(CharmBase):
    state = StoredState()

    def __init__(self, *args):
        super().__init__(*args)
        self.framework.observe(self.on.install, self)
        self.framework.observe(self.on.configure, self)

    def on_install(self, event):
        self.state.installed = snap_install('microk8s')

    def on_configure(self, event):
        if not self.state.installed:
            event.defer()
        cmd = 'sudo usermod -a -G microk8s ubuntu'
        subprocess.call(cmd, shell=True)
...
```

lib/charm.py

Stored State

- Used to pass data between hook execution

```
class Charm(CharmBase):
    state = StoredState()

    ...

    def on_install(self, event):
        self.state.installed = snap_install('microk8s')

    def on_configure(self, event):
        if not self.state.installed:
            event.defer()
        cmd = 'sudo usermod -a -G microk8s ubuntu'
        subprocess.call(cmd, shell=True)
```

lib/charm.py

Events

- Can originate from Juju or the charm code
- Can be observed
- Can be deferred

```
...
        self.framework.observe(self.on.install, self)
        self.framework.observe(self.on.configure, self)
...
    def on_configure(self, event):
        if not self.state.installed:
            event.defer()
        cmd = 'sudo usermod -a -G microk8s ubuntu'
        subprocess.call(cmd, shell=True)
```

lib/charm.py

Relation Events

- Contain information about the context (e.g. related units)

```
def on_related_app_sample_relation_changed(self, event):
    if not self.state.ready:
        event.defer()
        return
    related_app_serialized =
        event.relation.data[event.unit].get('related_app')

# No more structures like that:
#
# for rid in relation_ids(interface):
#     for unit in related_units(rid):
#         rdata = relation_get(unit=unit,
#                              rid=rid)
```

lib/charm.py

Charm structure


```
charm-tech-charm/
    config.yaml
    metadata.yaml
    hooks/
        hook1 -> ../lib/charm.py
        process-hook -> ../lib/charm.py
    lib/
        charm.py
        juju/framework.py
    templates/
```

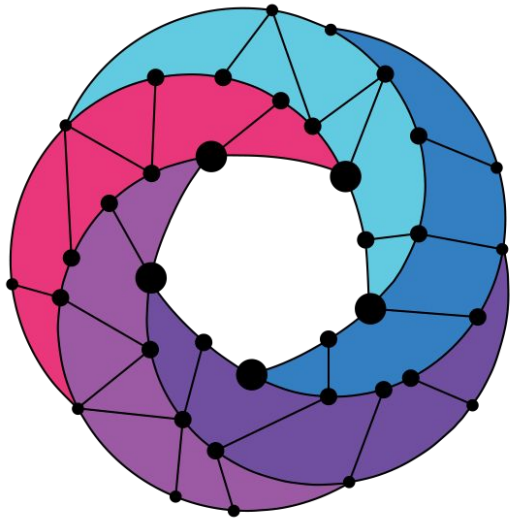
How does it work?

Open Source MANO

- The `lib/charm.py` file contains the **Charm** class
- The **Charm** class is instantiated and managed by the framework
- Event observation is setup
- During the hook execution:
  - Persistent state is loaded
  - Charm instance is created and initialised
  - Deferred events are reemitted
  - The event from the current hook execution context is emitted

Charm Tech framework - summary

- Event-based framework
- More intuitive
- More "pythonic"
- Developer has to provide the **Charm** class (`lib/charm.py`) only
- Next-month work:
  - Support for actions
  - The `process-hook`
  - Higher-level endpoint libraries



Open Source MANO

Find us at:

osm.etsi.org

osm.etsi.org/wikipub