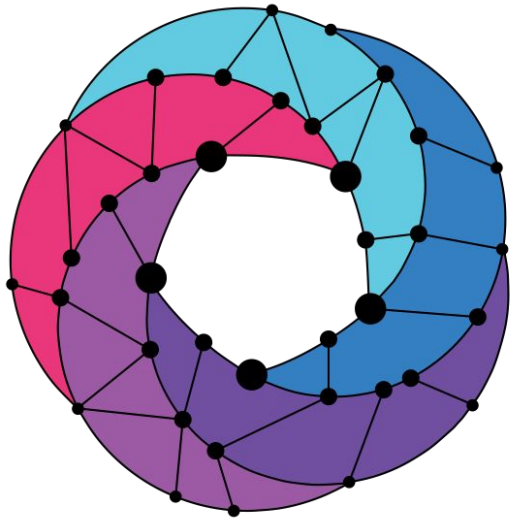


Open Source
MANO

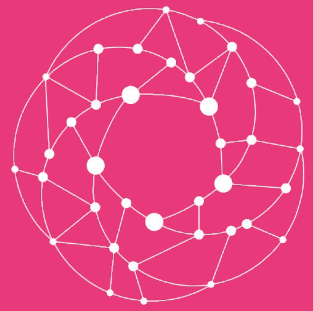
OSM Hackfest – Session 7.1

Introduction to Proxy Charms

Gianpietro Lavado (Whitestack)

© ETSI 2019

ETSI



Open Source
MANO

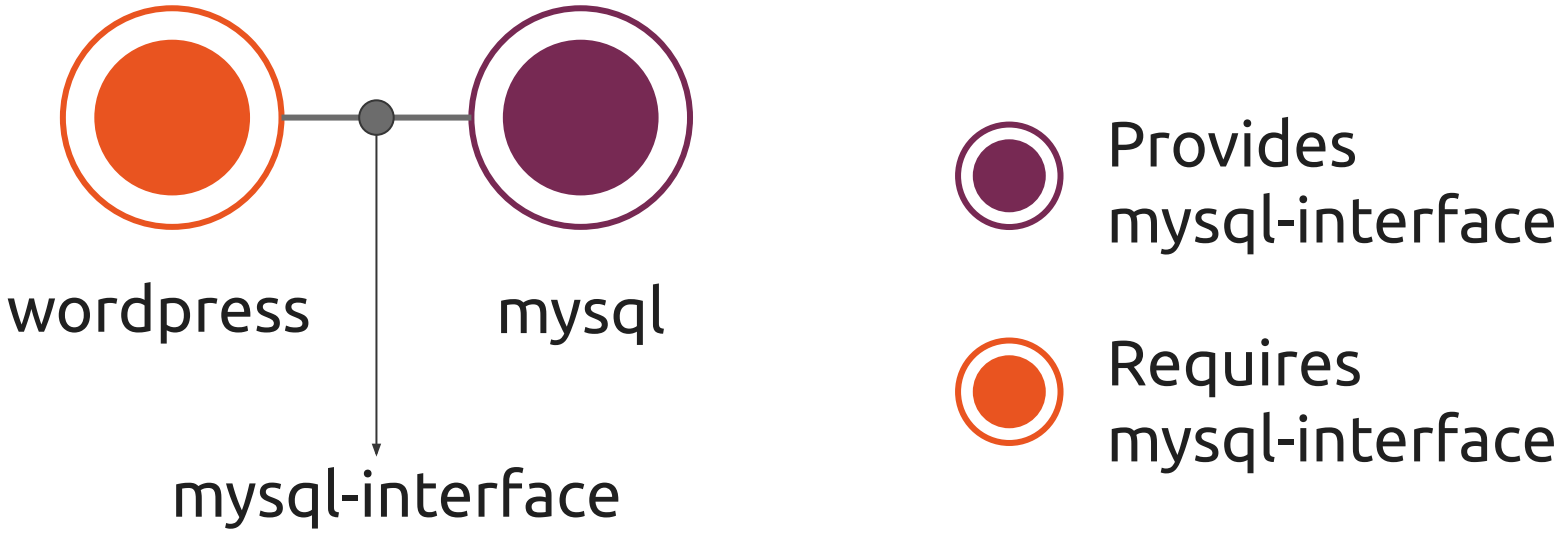
Introduction to charms

© ETSI 2019

ETSI

What is a charm?

- A charm is a set of scripts for deploying and operating software
  - Event handling built in
  - It's organized by layers → Helps reusing code
  - It can provide/require interfaces to exchange data with other charms
  - Utilizes Juju to deploy across multiple substrates
- Example:



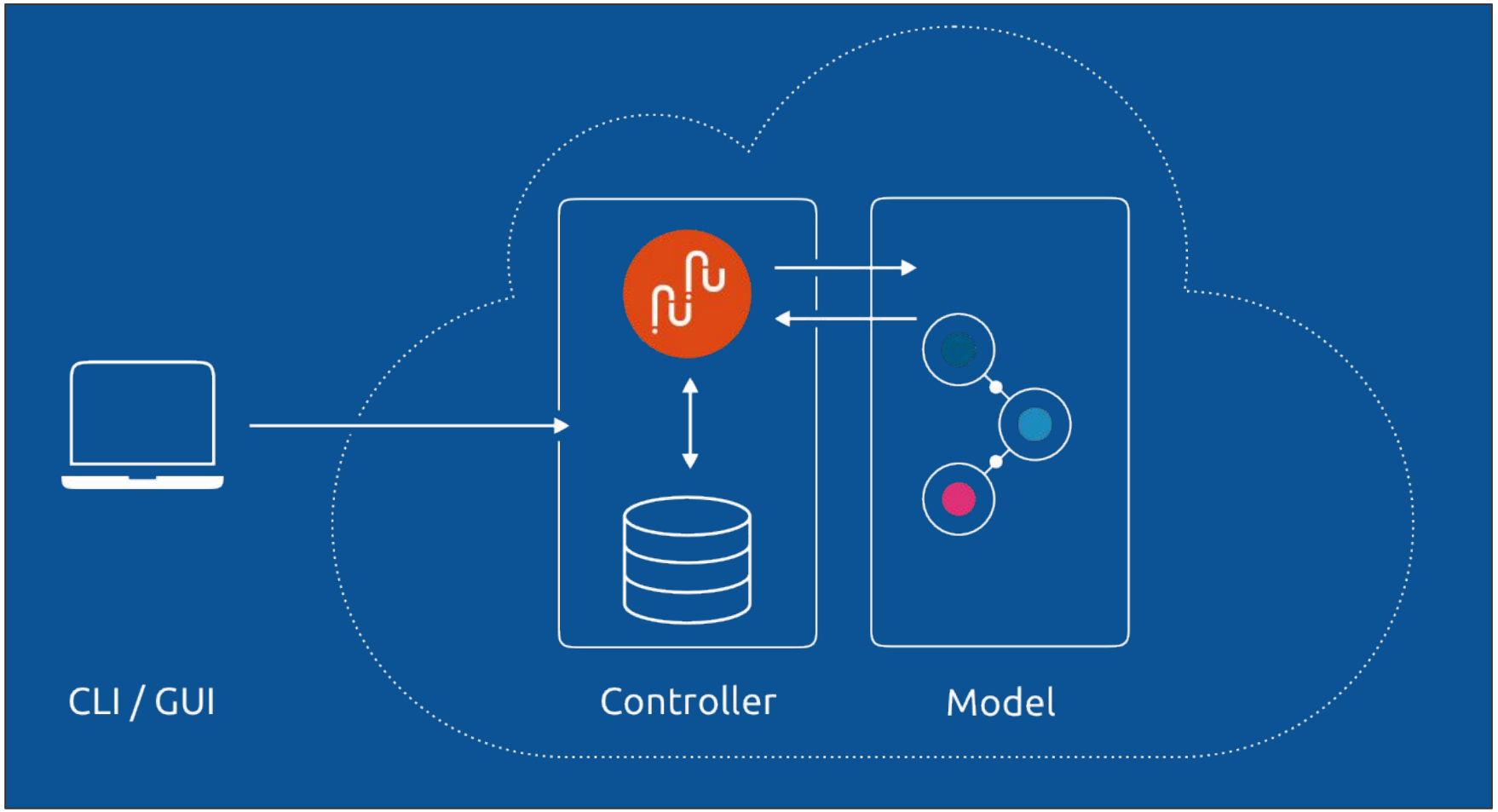
© ETSI 2019

What is Juju?

- Juju is an open source modeling tool, composed of a controller, models, and charms, for **operating software in the cloud**.
- Juju can handle **configuration**, **relationships** between services, **lifecycle** and **scaling**.
- This ensures that common elements such as databases, messaging systems, key value stores, logging infrastructure and other ‘glue’ functions are available as charms for automatic integration, reducing the burden on vendors and integrators.

© ETSI 2019

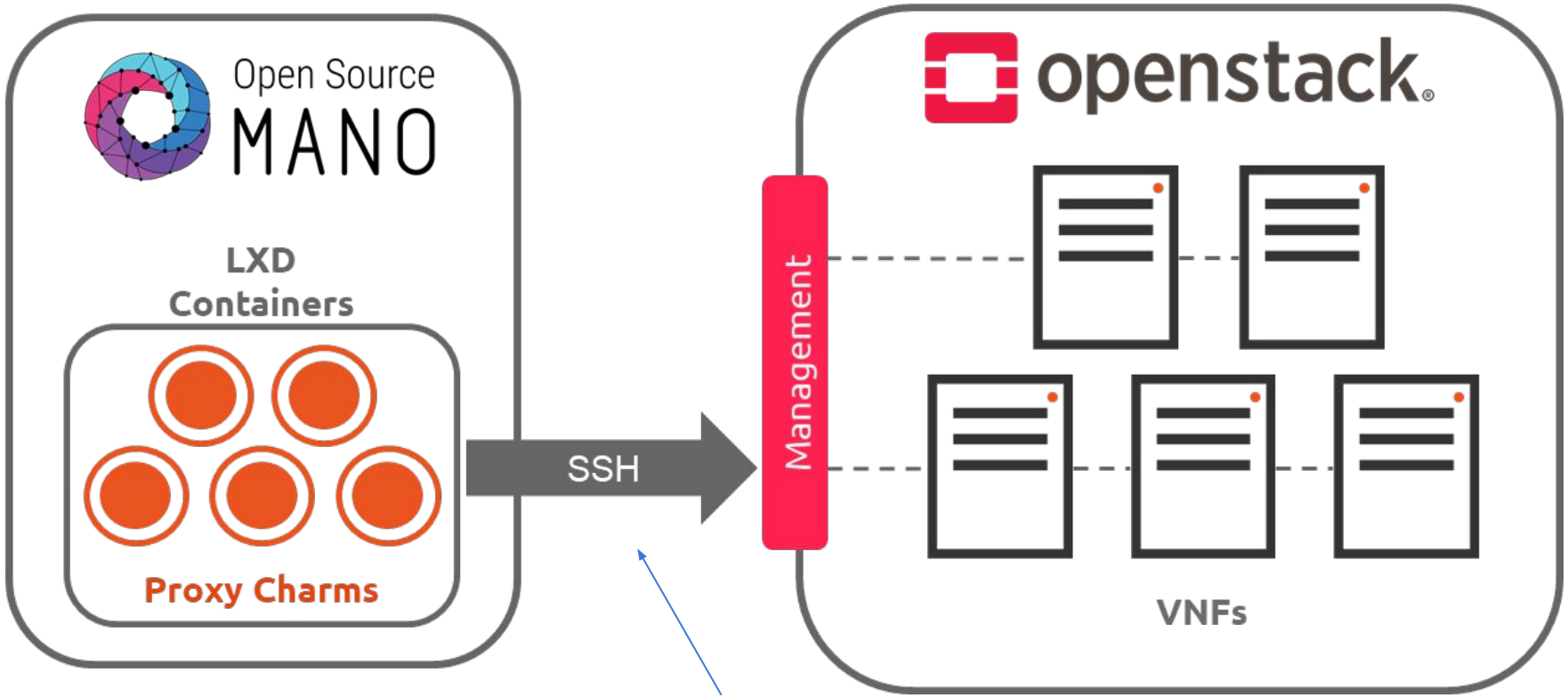
What is Juju?



© ETSI 2019

Proxy charms in OSM

As opposed to classical “Native charms”, Proxy charms run from outside the application. In particular, it runs within a model instantiated in a LXC container, that configures the VNFs through their management interface.



© ETSI 2019

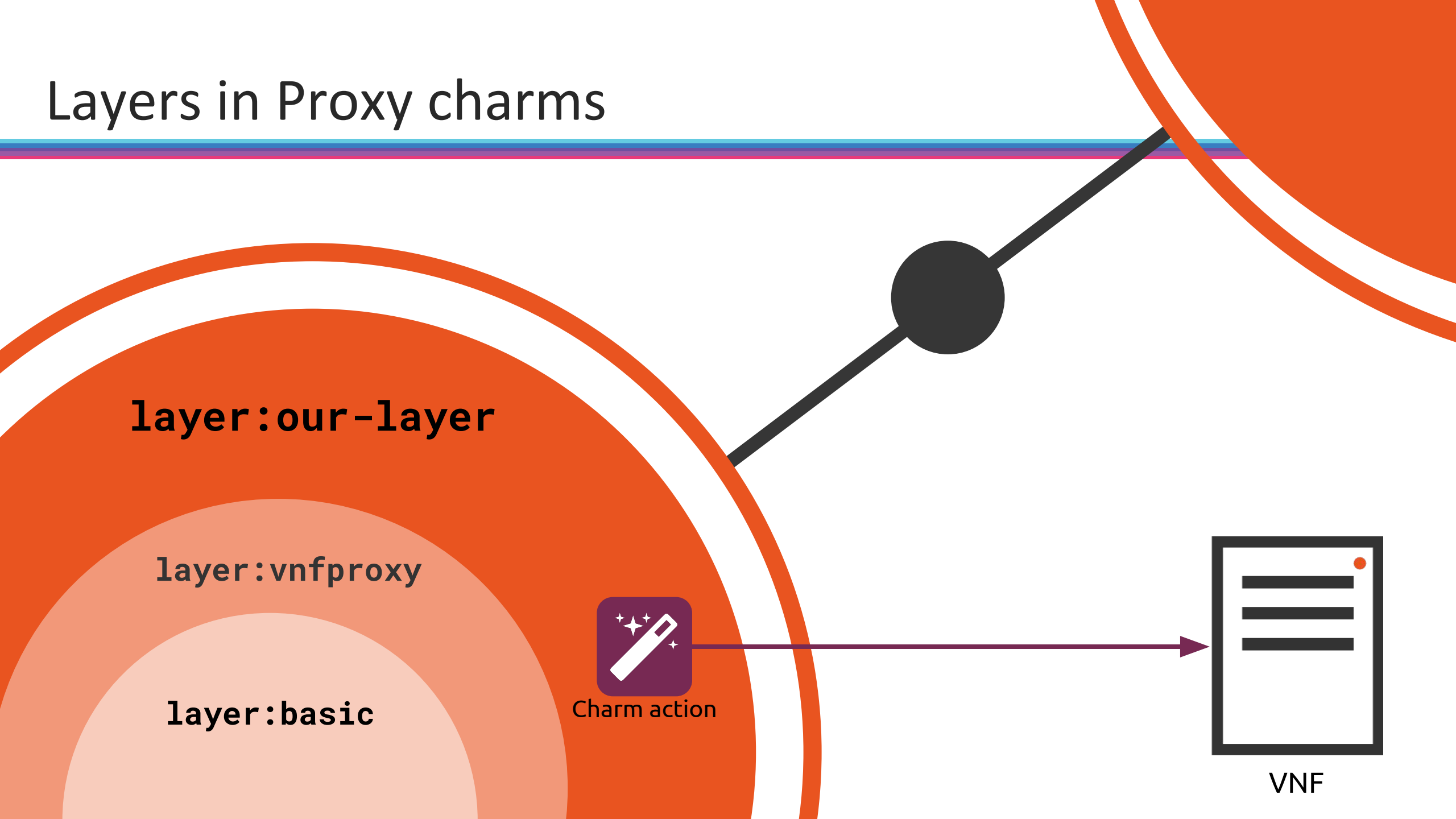
Proxy charms in OSM

- VNF configuration is done in three “days”
  - **Day-0:** The machine gets ready to be managed
    - E.g. import ssh-keys, create users/pass, network configuration, etc.
  - **Day-1:** The machine gets configured for providing services
    - E.g.: Configure (install packages, edit config files, execute commands, etc.)
  - **Day-2:** The machine configuration and management is updated
    - E.g.: Do **on-demand** actions (dump logs, backup mysql database, etc.)
- Proxy charms cover day-1 and day-2 configuration

Proxy Charms

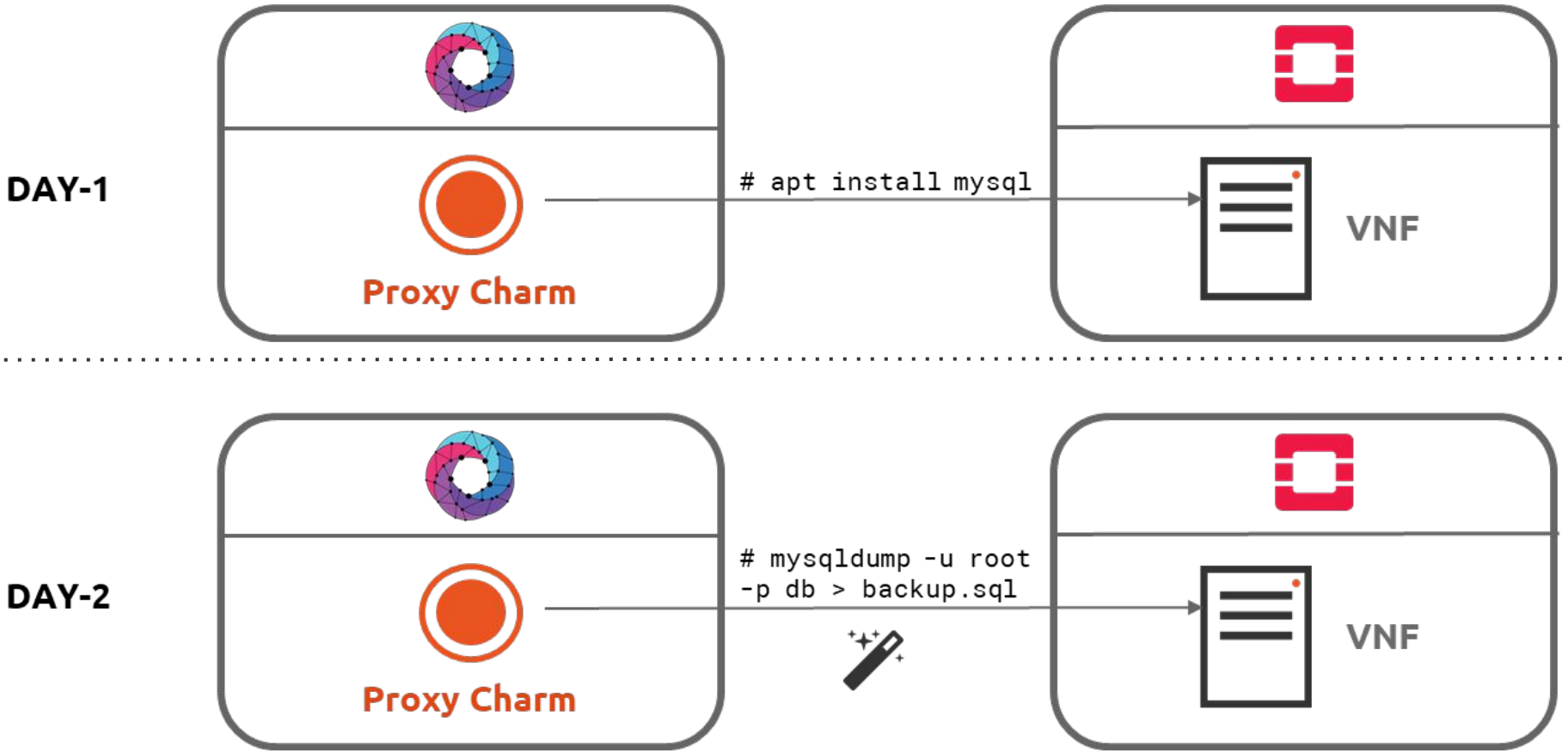
© ETSI 2019

Layers in Proxy charms

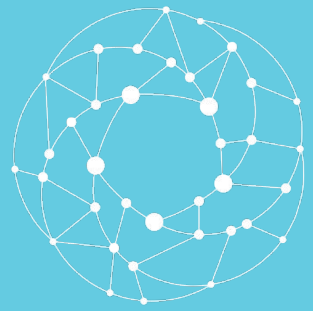


Charm actions - VNF primitives

MySQL simplified example



© ETSI 2019



Open Source
MANO

Building your charm

© ETSI 2019

ETSI

Setting up a charming environment

```
sudo snap install charm --classic  # already installed in using shared OSM
```

Create needed directories for building the charm

```
mkdir -p ~/charms/layers
mkdir -p ~/charms/interfaces
```

Juju and charms environment variables (Add to ~/.bashrc)

```
export JUJU_REPOSITORY=~/charms
export CHARM_LAYERS_DIR=$JUJU_REPOSITORY/layers
export CHARM_INTERFACES_DIR=$JUJU_REPOSITORY/interfaces
```

© ETSI 2019

Creating a Proxy charm layer

```
cd $JUJU_REPOSITORY/layers

charm create simple

cd simple/
```



```
$JUJU_REPOSITORY/layers
 |-- simple
     |-- README.ex
     |-- config.yaml
     |-- icon.svg
     |-- layer.yaml
     |-- metadata.yaml
     |-- reactive
     |   |-- simple.py
     |-- tests
         |-- 00-setup
         |-- 10-deploy
```

© ETSI 2019

Metadata.yaml

Metadata.yaml includes all the high level information of our charm

```
name: simple
summary: A simple VNF proxy charm
maintainer: Name <user@domain.tld>
subordinate: false
series: ['xenial','bionic']
```

metadata.yaml

© ETSI 2019

Layer.yaml

Layer.yaml states all the layers on which our layer is based.

```
includes: ['layer:basic', 'layer:vnfproxy']
options:
    basic:
        use_venv: false
```

layer.yaml

© ETSI 2019

Actions.yaml

Actions.yaml contains the high level description of the actions that will be implemented in our charm.

```
touch:
  description: "Touch a file on the VNF."
  params:
    filename:
      description: "The name of the file to touch."
      type: string
      default: ""
  required:
  - filename
```

actions.yaml

© ETSI 2019

Reactive/simple.py

Open Source MANO

Reactive/simple.py contains the actual code of our Proxy charm.

```
from charmhelpers.core.hookenv import action_get, action_fail, action_set, status_set
from charms.reactive import clear_flag, set_flag, when, when_not
import charms.sshproxy


@when('sshproxy.configured')
@when_not('simple.installed')
def install_simple_proxy_charm():
    """Set the status to active when ssh configured."""
    set_flag('simple.installed')
    status_set('active', 'Ready!')
```

reactive/simple.py

© ETSI 2019

Actions/touch

```
$ mkdir actions
$ nano actions/touch
$ chmod +x actions/touch
```

```
#!/usr/bin/env python3
import sys
sys.path.append('lib')
from charms.reactive import main, set_state
from charmhelpers.core.hookenv import action_fail, action_name

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

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

actions/touch

Note: Every Proxy charm action uses the same code. The actual action is defined in the reactive directory.

© ETSI 2019

Implementing the action

Append the implementation of the action to reactive/simple.py

```
@when('actions.touch')
def touch():
    """Touch a file."""
    err = ''
    try:
        filename = action_get('filename')
        cmd = ['touch {}'.format(filename)]
        result, err = charms.sshproxy._run(cmd)
    except:
        action_fail('command failed: {}'.format(err))
    else:
        action_set({'output': result})
    finally:
        clear_flag('actions.touch')
```

reactive/simple.py

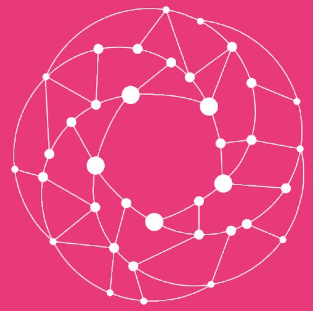
© ETSI 2019

Building a Proxy charm layer

```
$ charm build
$ ls $JUJU_REPOSITORY/builds/simple
```

```
actions           bin          copyright   hooks       layer.yaml     Makefile
reactive          README.md    simple      tox.ini     actions.yaml   config.yaml
deps              icon.svg     lib         README.ex   metadata.yaml  test
requirements.txt  wheelhouse
```

© ETSI 2019



Open Source
MANO

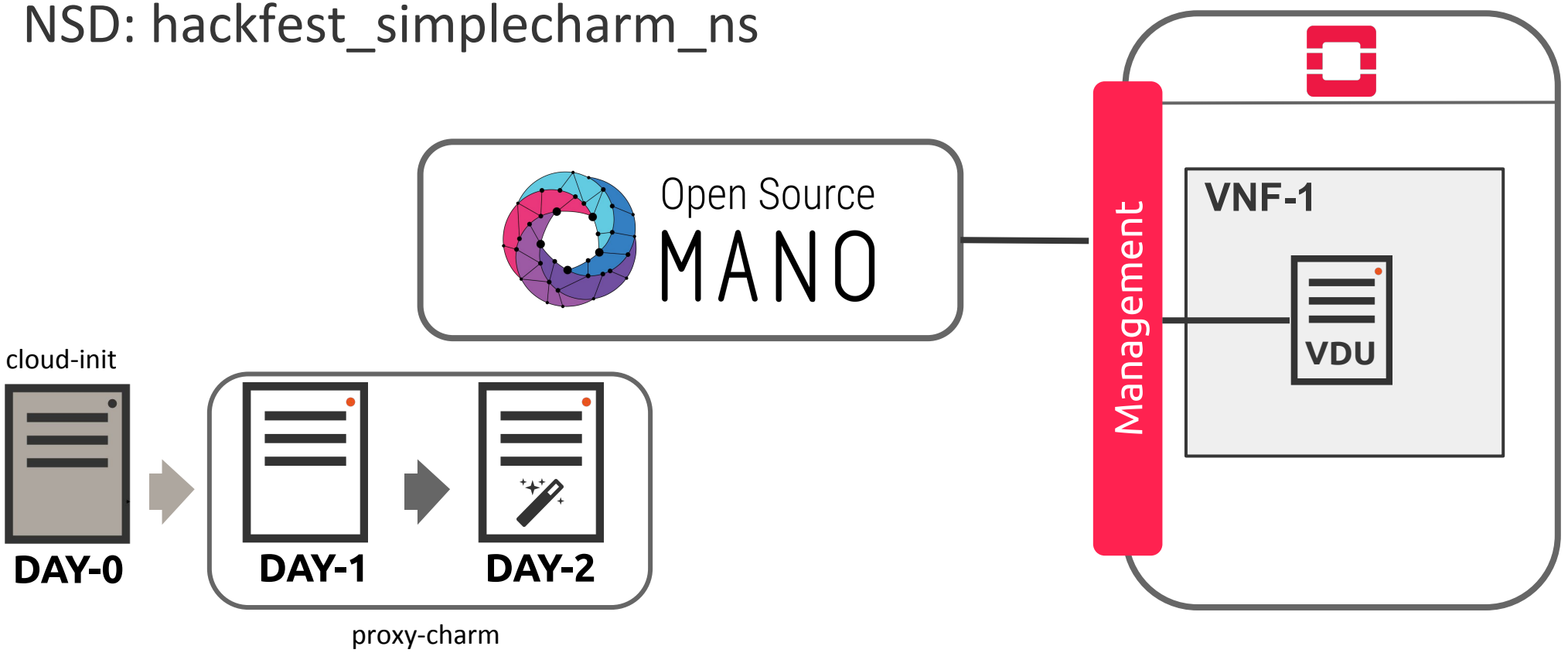
Building your VNF/NS packages

© ETSI 2019

ETSI

Context

- VNFD: hackfest_simplecharm_vnf
- NSD: hackfest_simplecharm_ns



© ETSI 2019

Get a VNFD template to get started

- Download the following incomplete VNF package (just missing the charm!)

```
wget http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_simplecharm_vnfd_incomplete.tar.gz
```

- Untar it to start working on the YAML file over the next slides

```
tar -xvzf hackfest_simplecharm_vnfd_incomplete.tar.gz
```

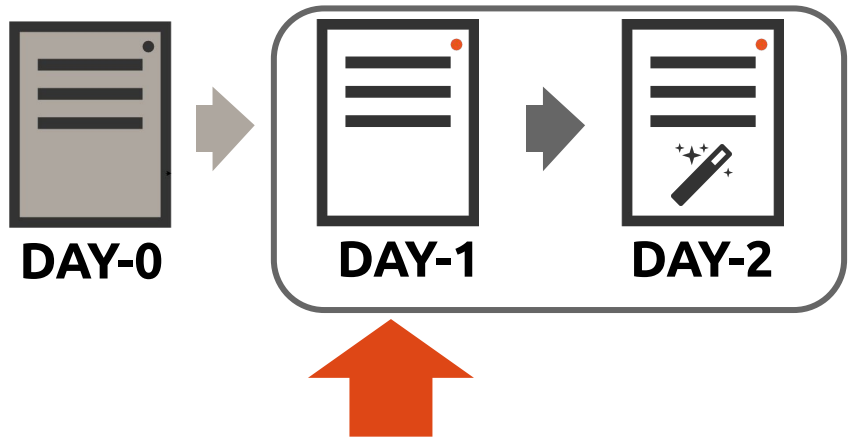
- Our final result will be the below package (for reference purposes):

```
wget http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_simplecharm_vnf.tar.gz
```

© ETSI 2019

Adding the charm to your descriptor

- Edit the file to add your charm!
- Start by creating the vnf-configuration section and put the name of our charm in the *juju:charm:* section.




```
name: hackfest_simplecharm-vnf
...
vnf-configuration:
    juju:
        charm: simple
    initial-config-primitive:
    -   seq: '1'
        name: config
        parameter:
        -   name: ssh-hostname
            value: <rw_mgmt_ip>
        -   name: ssh-username
            value: ubuntu
        -   name: ssh-password
            value: osm4u
    -   seq: '2'
        name: touch
        parameter:
        -   name: filename
            value: '/home/ubuntu/first-touch'
```

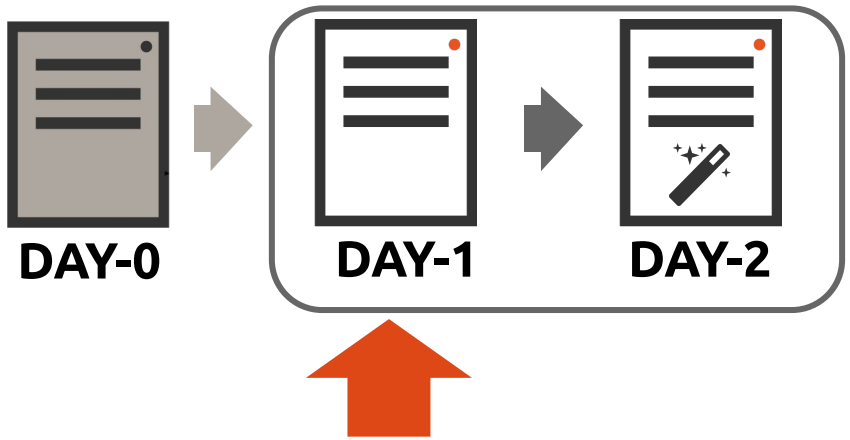
hackfest_simplecharm_vnfd.yaml

© ETSI 2019

Day-1 Configuration

The initial-config-primitive section takes care of Day-1

- The *seq* section states the order in which the initial config primitives will be called.
- The Proxy charm has ssh access to the VNFs thanks to the *config* primitive.



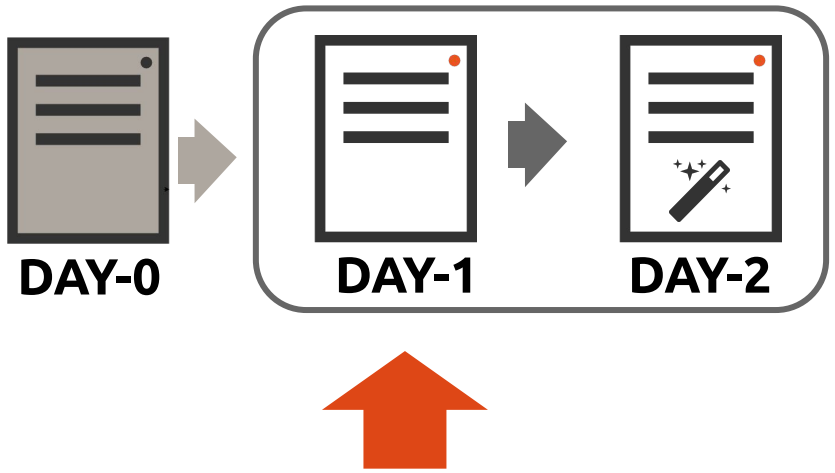

```
name: hackfest_simplecharm-vnf
...
vnf-configuration:
    juju:
        charm: simple
    initial-config-primitive:
    -   seq: '1'
        name: config
        parameter:
        -   name: ssh-hostname
            value: <rw_mgmt_ip>
        -   name: ssh-username
            value: ubuntu
        -   name: ssh-password
            value: osm4u
    -   seq: '2'
        name: touch
        parameter:
        -   name: filename
            value: '/home/ubuntu/first-touch'c
```

hackfest_simplecharm_vnfd.yaml

© ETSI 2019

Day-1 Configuration

- The *touch* primitive is our Day-1 action created in the simple charm.




```
name: hackfest_simplecharm-vnf
...
vnf-configuration:
    juju:
        charm: simple
    initial-config-primitive:
    -   seq: '1'
        name: config
        parameter:
        -   name: ssh-hostname
            value: <rw_mgmt_ip>
        -   name: ssh-username
            value: ubuntu
        -   name: ssh-password
            value: osm4u
    -   seq: '2'
        name: touch
        parameter:
        -   name: filename
            value: '/home/ubuntu/first-touch'
```

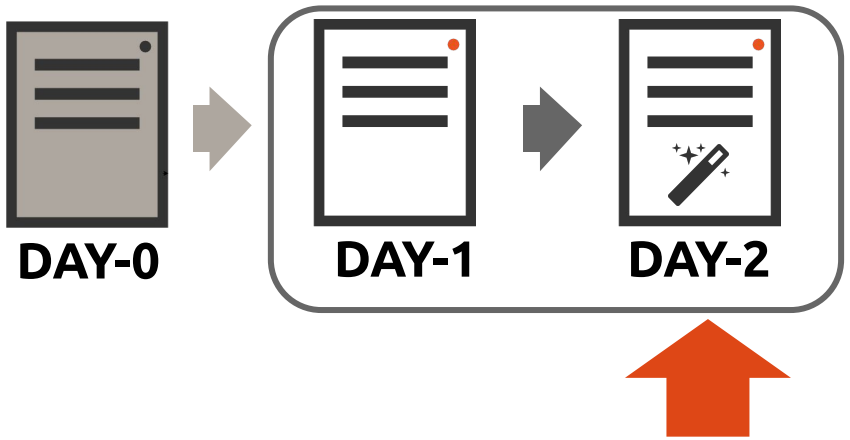
hackfest_simplecharm_vnfd.yaml

© ETSI 2019

Day-2 Configuration

The *config-primitive* section contains the available on-demand actions for Day-2.

- Our same *touch* primitive is included for on-demand calls.




```
initial-config-primitive:
      ...
config-primitive:
-    name: touch
      parameter:
      -    name: filename
           data-type: STRING
           default-value: '/home/ubuntu/touched'
```

hackfest_simplecharm_vnfd.yaml

© ETSI 2019

Generate VNF descriptor

Open Source MANO

Copy the charm you created from your descriptor, validate it and finally rebuild your charm!

```
# Copying it
cp -r $JUJU_REPOSITORY/builds/simple ~/hackfest_simplecharm_vnfd/charms/simple
# Validating it
python ~/validate_descriptor.py ~/hackfest_simplecharm_vnfd/hackfest_simplecharm_vnfd.yaml
# Re-packaging it
cd ~
tar -cvzf hackfest_simplecharm_vnfd.tar.gz hackfest_simplecharm_vnfd/
```

© ETSI 2019

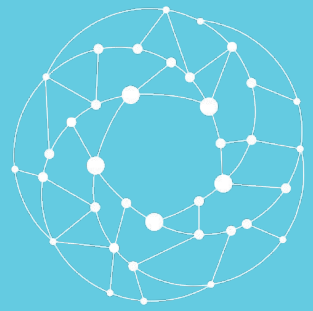
NS Descriptor

- Download the NSD

```
cd ~

wget
http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_simplecharm_ns.tar.gz
```

© ETSI 2019



Open Source
MANO

Final Steps

© ETSI 2019

ETSI

Final Steps: Import VNF and NS

```
osm vnfd-create hackfest_simplecharm_vnfd.tar.gz
osm vnfd-list
+------------------------------+--------------------------------------+
| nfpkg name                   | id                                   |
+------------------------------+--------------------------------------+
| hackfest_simplecharm-vnf     | 3f9294be-1e76-4717-be1f-6ad168551788 |
+------------------------------+--------------------------------------+

osm nsd-create hackfest_simplecharm_ns.tar.gz
osm nsd-list
+------------------------------+--------------------------------------+
| nsd name                     | id                                   |
+------------------------------+--------------------------------------+
| hackfest_simplecharm-ns      | 9481bd6c-41be-4315-a249-afb8eadea544 |
+------------------------------+--------------------------------------+
```

© ETSI 2019

Speeding up the deployment

By default, LXC containers run ‘apt update & upgrade’ when instantiated to run a proxy charm, and they use a default location for debian packages. To speed up charm deployment, you can one or both of these methods:

First method: Disable apt-update/upgrade & use a cached LXC image

```
git clone https://github.com/AdamIsrael/osm-hackfest.git
osm-hackfest/bin/update-juju-lxc-images --xenial
juju model-config enable-os-refresh-update=false enable-os-upgrade=false
```

Second method: Set a closer debian packages source

```
juju model-config apt-mirror=http://fr.archive.ubuntu.com/ubuntu/
```

© ETSI 2019

Launching the NS

If you want to apply the speeding methods, you need to do “model-config” commands at the model level, and since REL6, the model is unique per NS and it matches the NS ID.

So right after instantiating the NS, and before the LXC machine is created, you need to switch to the model and provide the optimizations.

Let’s try the second method while you instantiate your NS:

```
osm ns-create --ns_name hf-simple-XX --nsd_name hackfest_simplecharm-ns
--vim_account ws2 --config '{vld: [ {name: mgmtnet, vim-network-name:
osm-ext} ] }'

# then quickly run the (2nd) speeding method

osm ns-list | grep hf-simple-XX | awk '{print $4}' | xargs -l1 juju switch
juju model-config apt-mirror=http://fr.archive.ubuntu.com/ubuntu/
```

Notes:

- *XX is your POD/tenant number*
- *We use fr.archive.ubuntu.com due to the location of the NFVI*

© ETSI 2019

Juju commands

To monitor your charms, make sure you switch to the model (juju switch [NS-ID]) and run:

```
juju status
```

→ charm status

```
juju debug-log
```

→ live debug messages

```
lxc list
```

→ To check if the proxy charms are getting an ip

```
lxc exec <proxy-charm> bash
```

→ access the container

© ETSI 2019

Final Steps: Check the actions

Checking Day-1

The Day-1 action should have run automatically, so SSH into the VNFand look for a ‘first-touch’ file in the /home/ubuntu directory

ssh ubuntu@VNF-IP-ADDRESS (password: osm4u)

ubuntu@hf-simple-1-mgmtvm-1:~$ ls
first-touch

© ETSI 2019

Final Steps: Check the actions

Open Source MANO

Checking Day-2

The Day-2 action is on demand, you can call it from the UI, or by running this command:

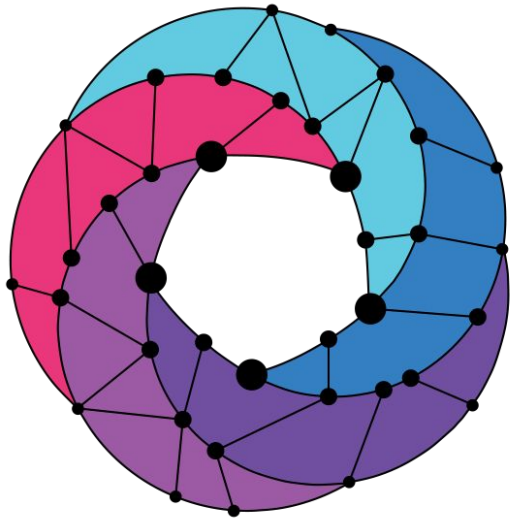
```
osm ns-action hf-simple-XX --vnf_name 1 --action_name touch --params '{filename: /home/ubuntu/mytouch1}'
```

Now go back to the VNF and check if the file exists.

ubuntu@hf-simple-1-mgmtvm-1:~$ ls
first-touch mytouch1

Don't run this yet!
there is an active bug in 6.0.4 for shared environments, when using the same VNFDs

© ETSI 2019



Open Source MANO

Find us at:

osm.etsi.org

osm.etsi.org/wikipub

© ETSI 2019