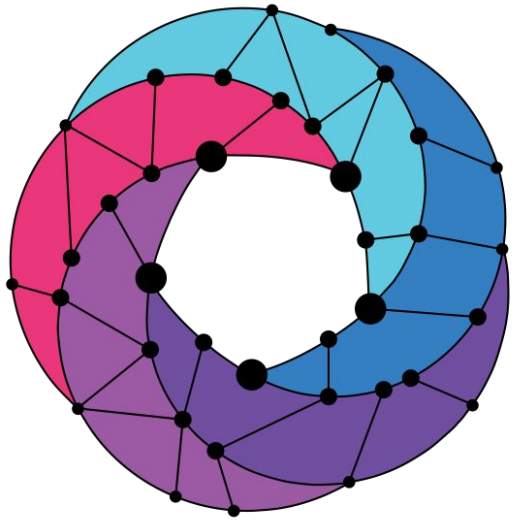


Open Source
MANO

OSM MR Hackfest – Hack 3
Automating VNF Day-1 & 2 operations

David Garcia (Canonical)
Dominik Fleischmann (Canonical)
Dmitrii Shcherbakov (Canonical)

© ETSI 2020

ETSI

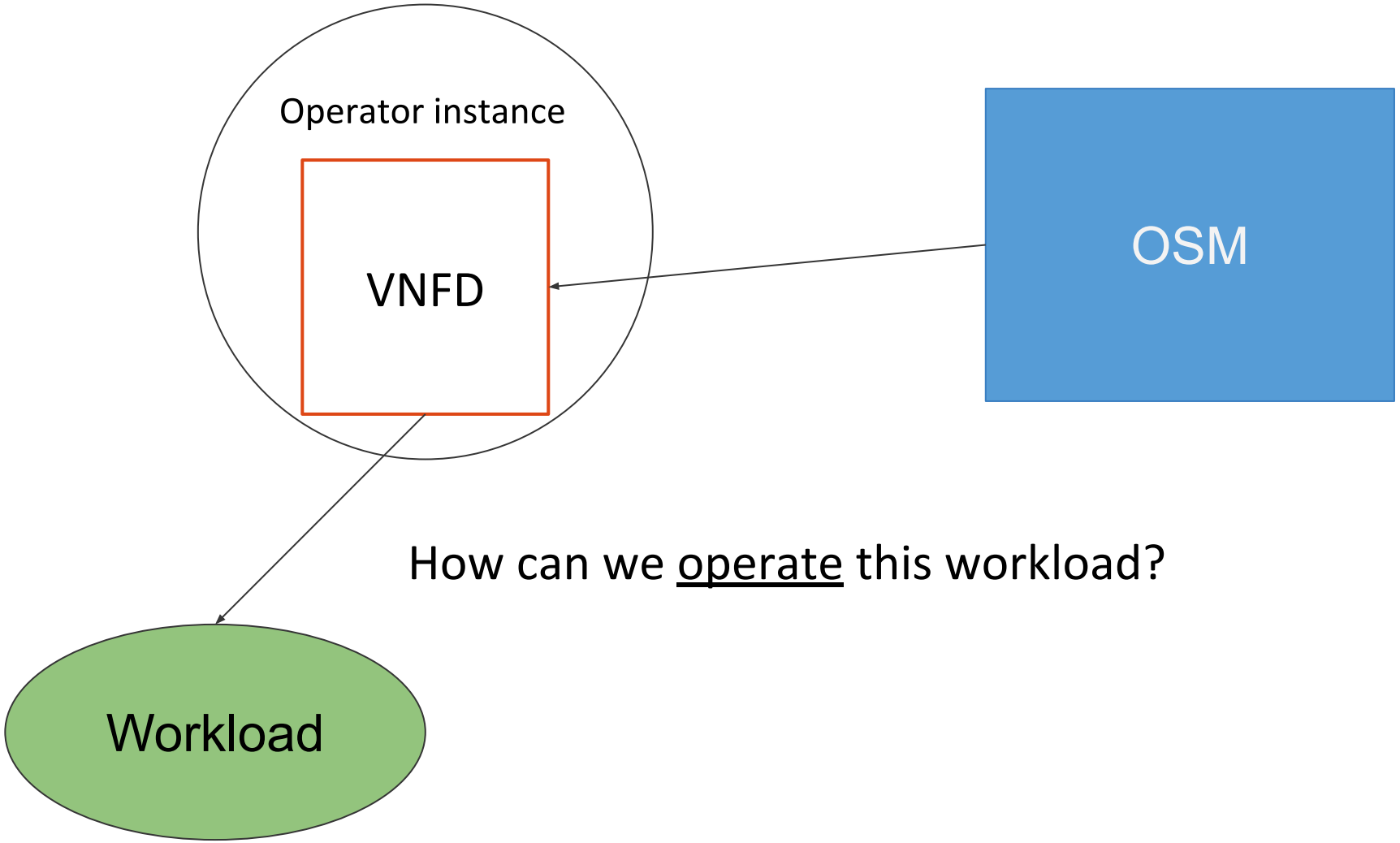
Open Source
MANO

Introduction to VNF operations

© ETSI 2019

ETSI

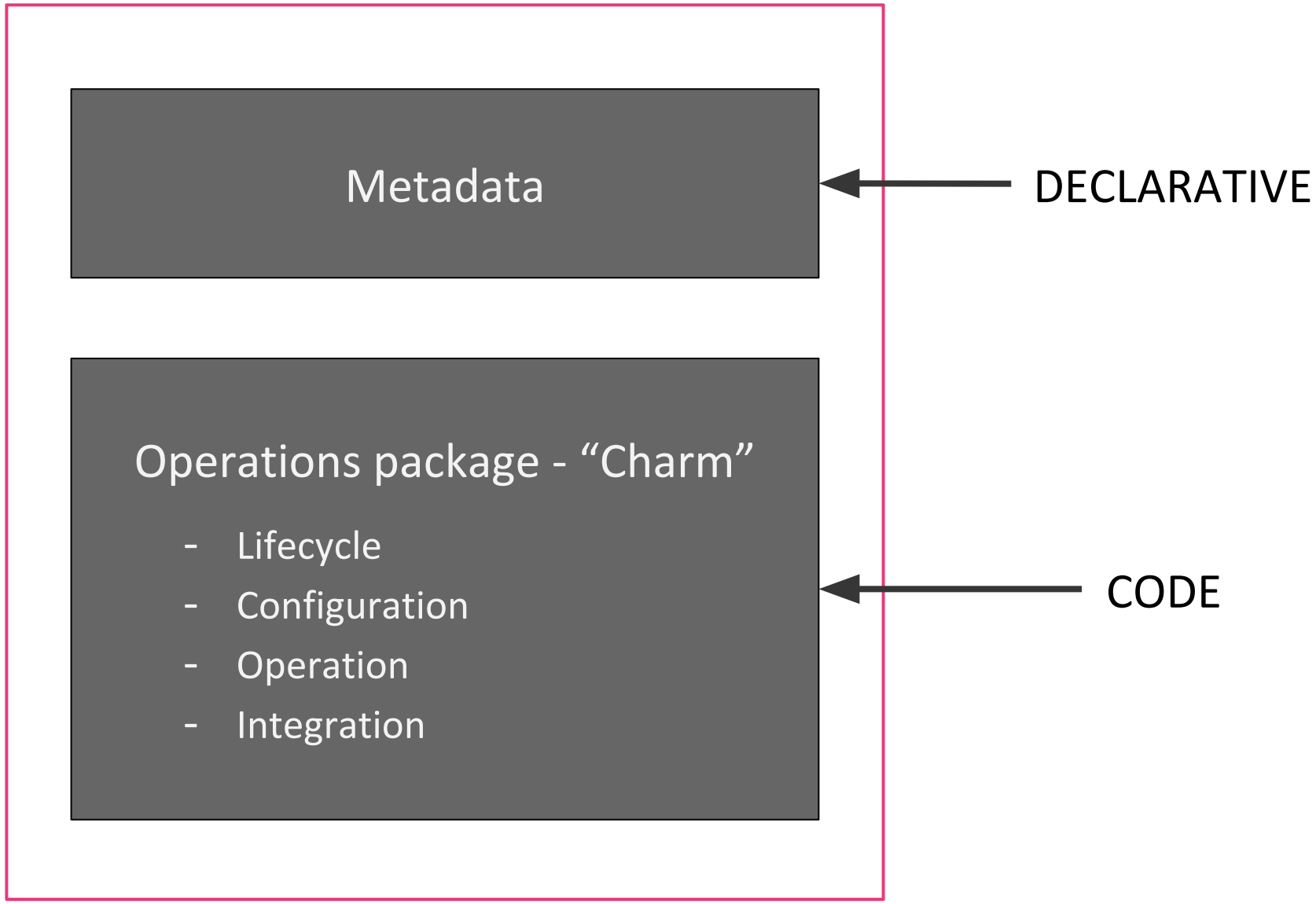
Operating workloads



How can we operate this workload?

© ETSI 2020

VNFD



© ETSI 2020

Lifecycle

- Install
- Scale out
- Upgrade
- Report status

© ETSI 2020

Configuration

- Initial configuration
- Change configuration
- YAML

```
options:
  auth-type:
    type: string
    description: Sets the authentication type
    default: keystone
  port:
    type: int
    description: Sets the port
```

© ETSI 2020

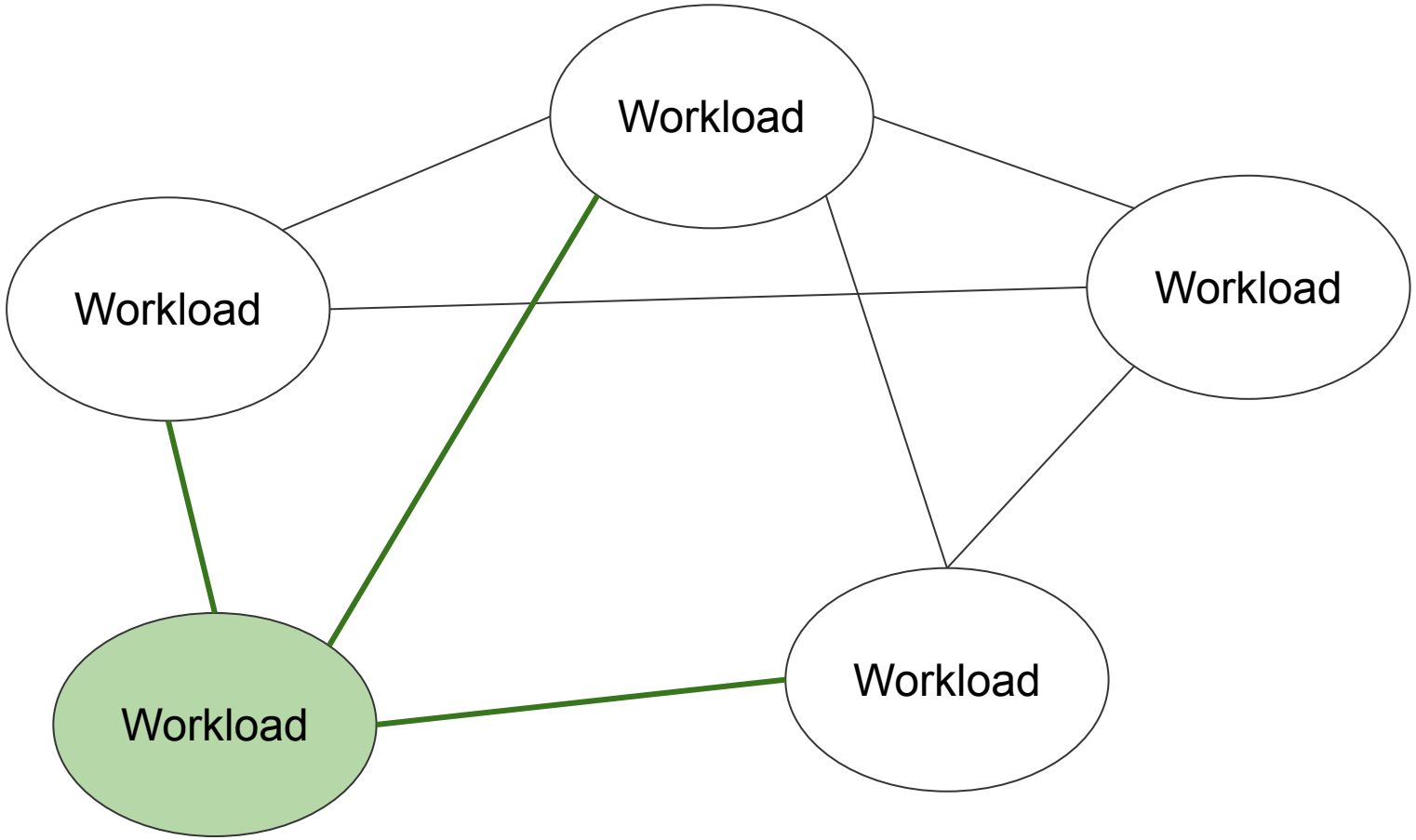
Operations… for example

- Backup
- Monitor
- Debug
- Add users, policies, rules, etc.
- Manage certificates, keys, etc.
- Rotate logs
- **Custom operations**

Each ‘operations primitive’ is a function call that takes parameters and produces a result.

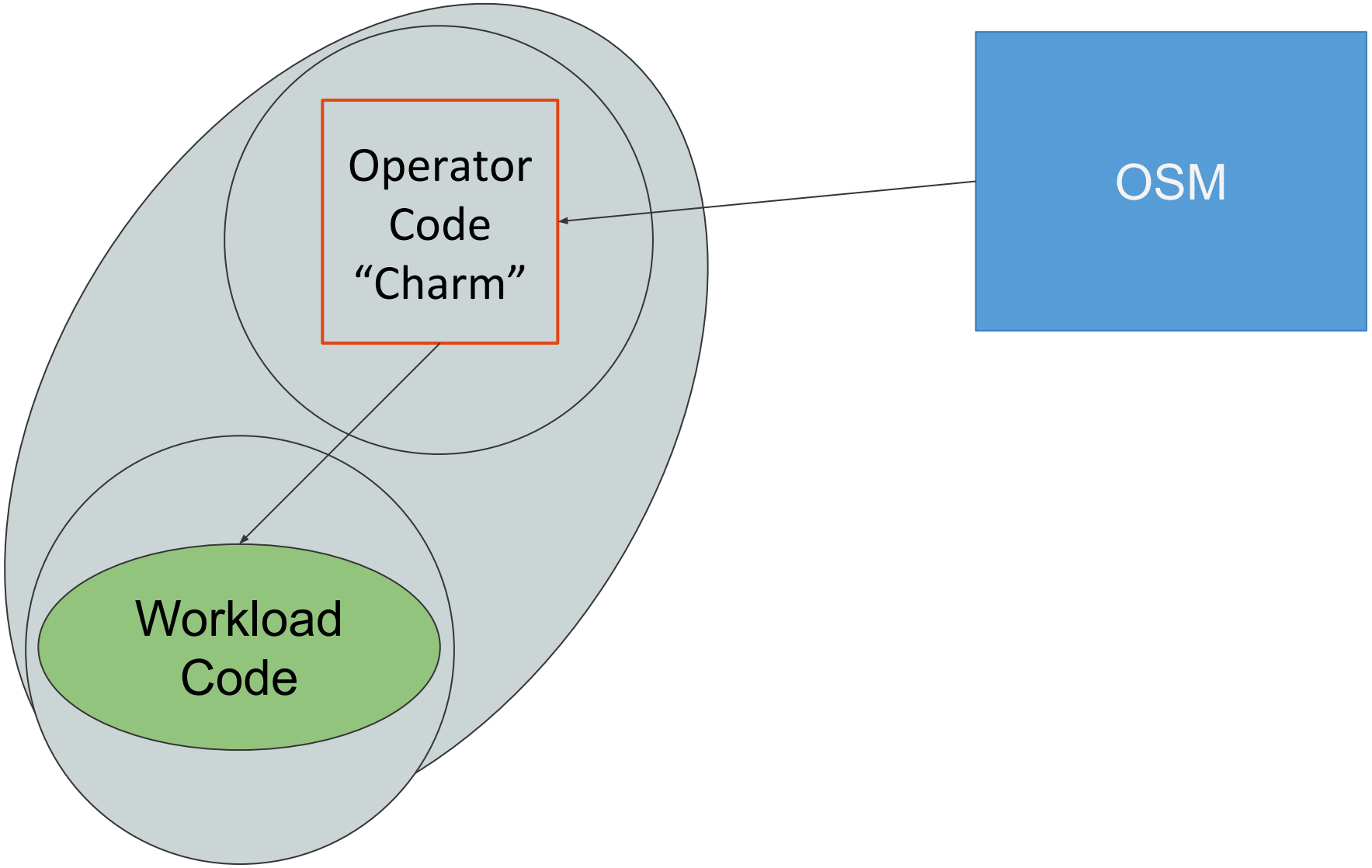
© ETSI 2020

Integration



© ETSI 2020

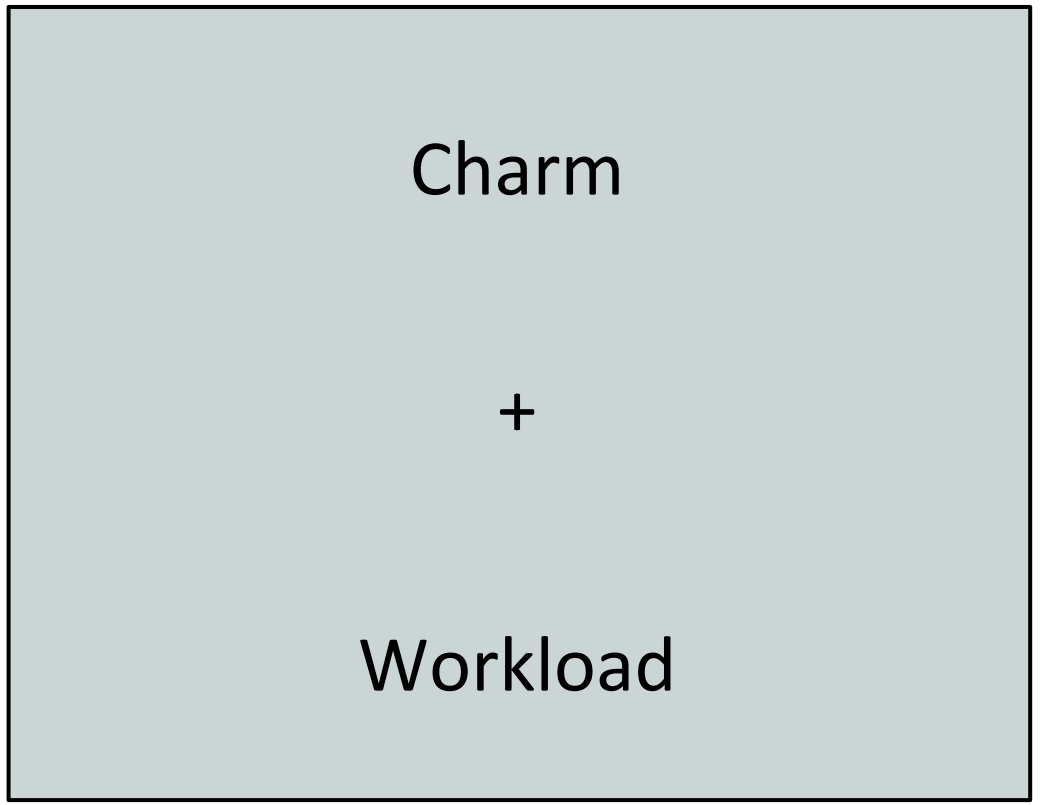
Native and Proxy Charms



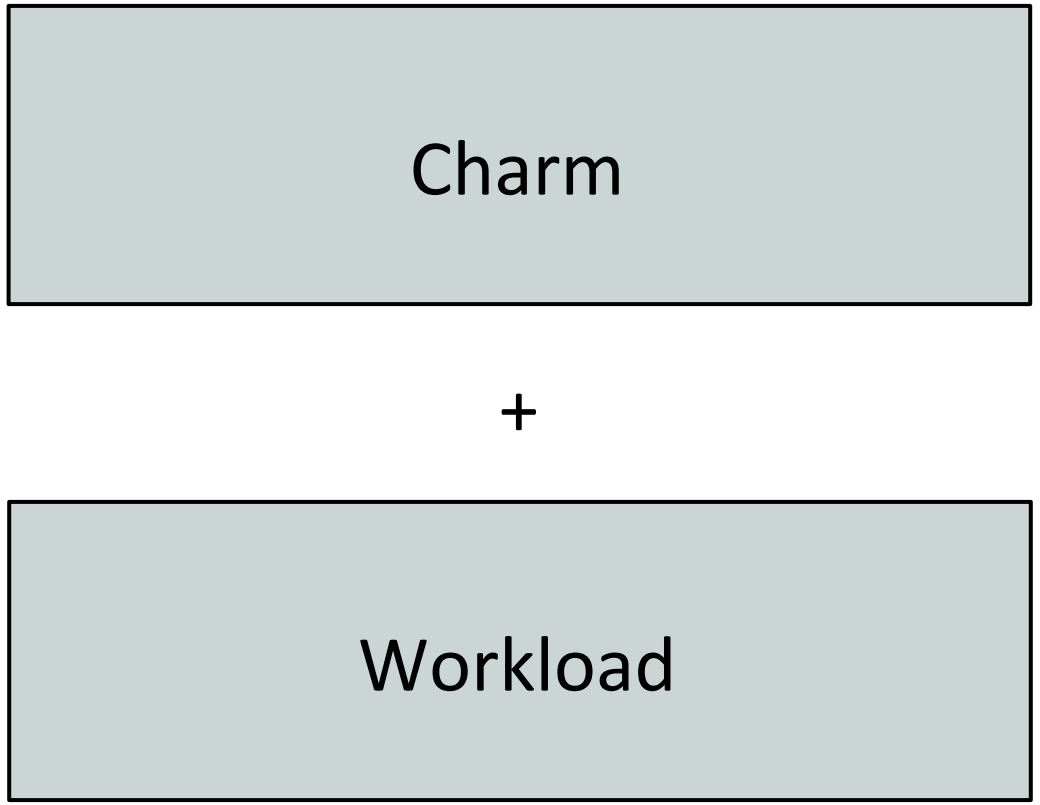
© ETSI 2020

Open Source MANO

“Native” Charm



“Proxy” Charm

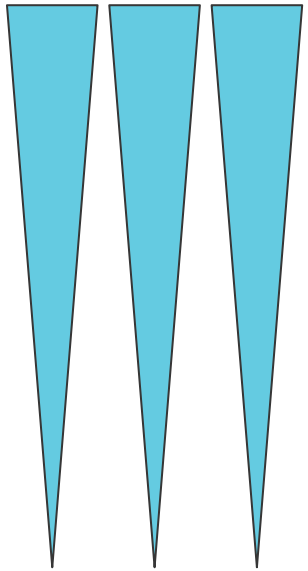


© ETSI 2020

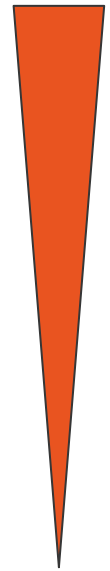
DEMO

© ETSI 2020

Kubernetes “operators” are charms



Workload containers



Charm

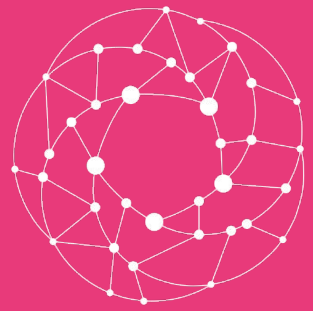
Operator container

© ETSI 2020

Summary of charms

- Part of VNFD
- Manage:
  - Lifecycle
  - Configuration
  - Operations
  - Integration
- Machine and kubernetes workloads
- **Charms are <u>code</u>**

© ETSI 2020



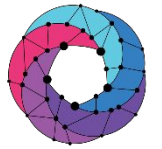
Open Source
MANO

Python Operator Framework

© ETSI 2020

ETSI

Python Operator Framework



"Make it easy to write a Kubernetes operator in pure Python"

© ETSI 2020

Python Operator Framework

- Class hierarchy
  - for modeling services and integrations;
- Event system;
- Persistence layer;
- Minimal dependencies.

© ETSI 2020

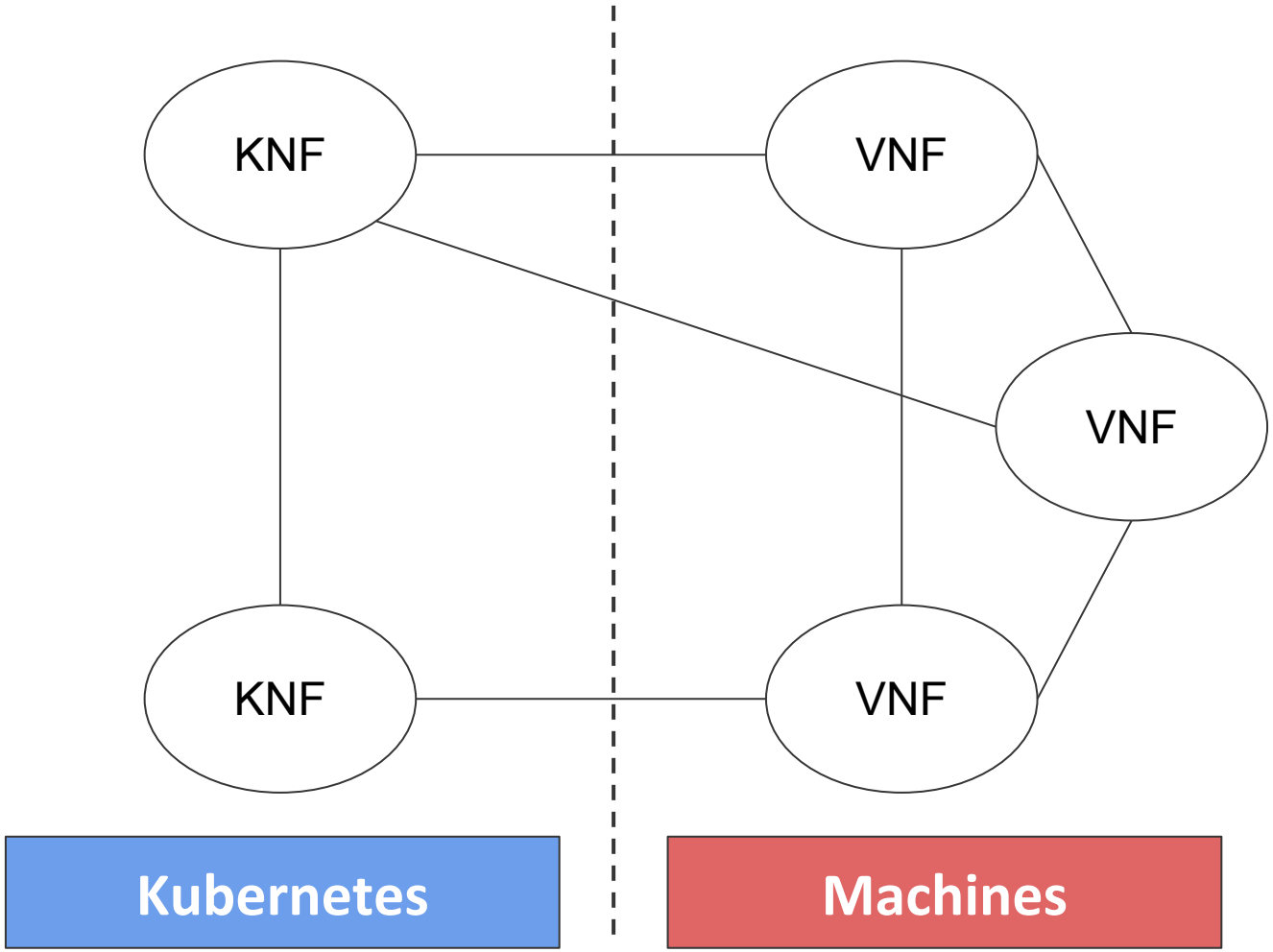
Why Pure Python?

Open Source MANO

- High-level
- Widely known
- Low entry barrier
- Good for writing integration code
- Simple debugging & testing
- Cross-architecture

© ETSI 2020

Unified operator framework for K8s and legacy workloads



© ETSI 2020

Event Driven Programming Model

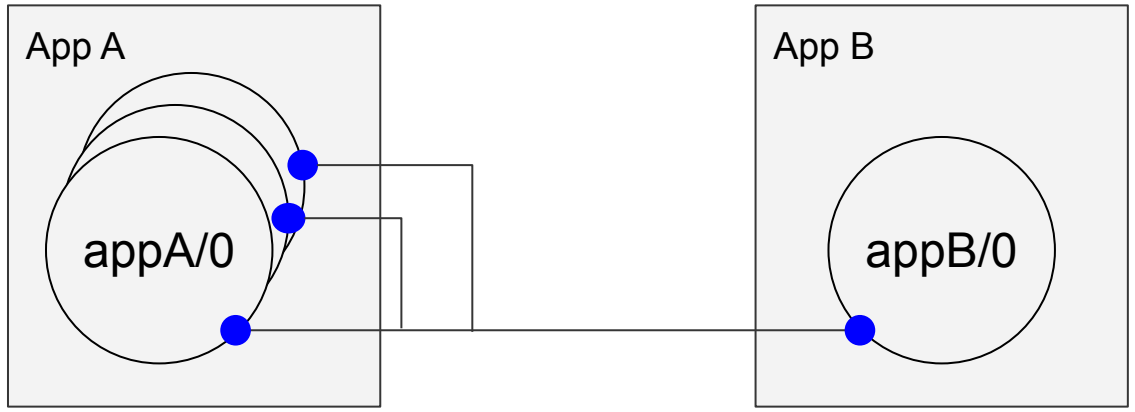
© ETSI 2020

Event handlers are Charm object methods


```
class MyCharm(CharmBase):

    def __init__(self, *args):
        super().__init__(*args)
        self.framework.observe(self.on.install, self.on_install)
        self.framework.observe(self.on.config_changed, self.on_config_changed)
        self.framework.observe(self.on.start, self.on_start)
        self.framework.observe(self.on.leader_elected, self.on_leader_elected)
        self.framework.observe(self.on.db_relation_joined, self.on_db_relation_joined)
        self.framework.observe(self.on.db_relation_changed, self.on_db_relation_changed)

    def on_install(self, event):
        packages = self.model.config['packages']
        # Install relevant packages…

    def on_config_changed(self, event):
        # Re-render a template here.
```

© ETSI 2020

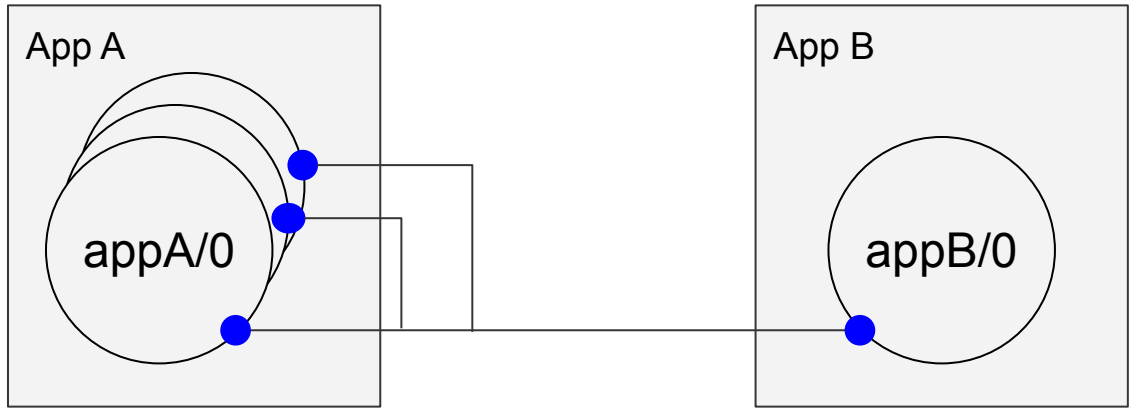
Graph change events



“A multi-VDU VNF integrated with a single-VDU VNF”

© ETSI 2020

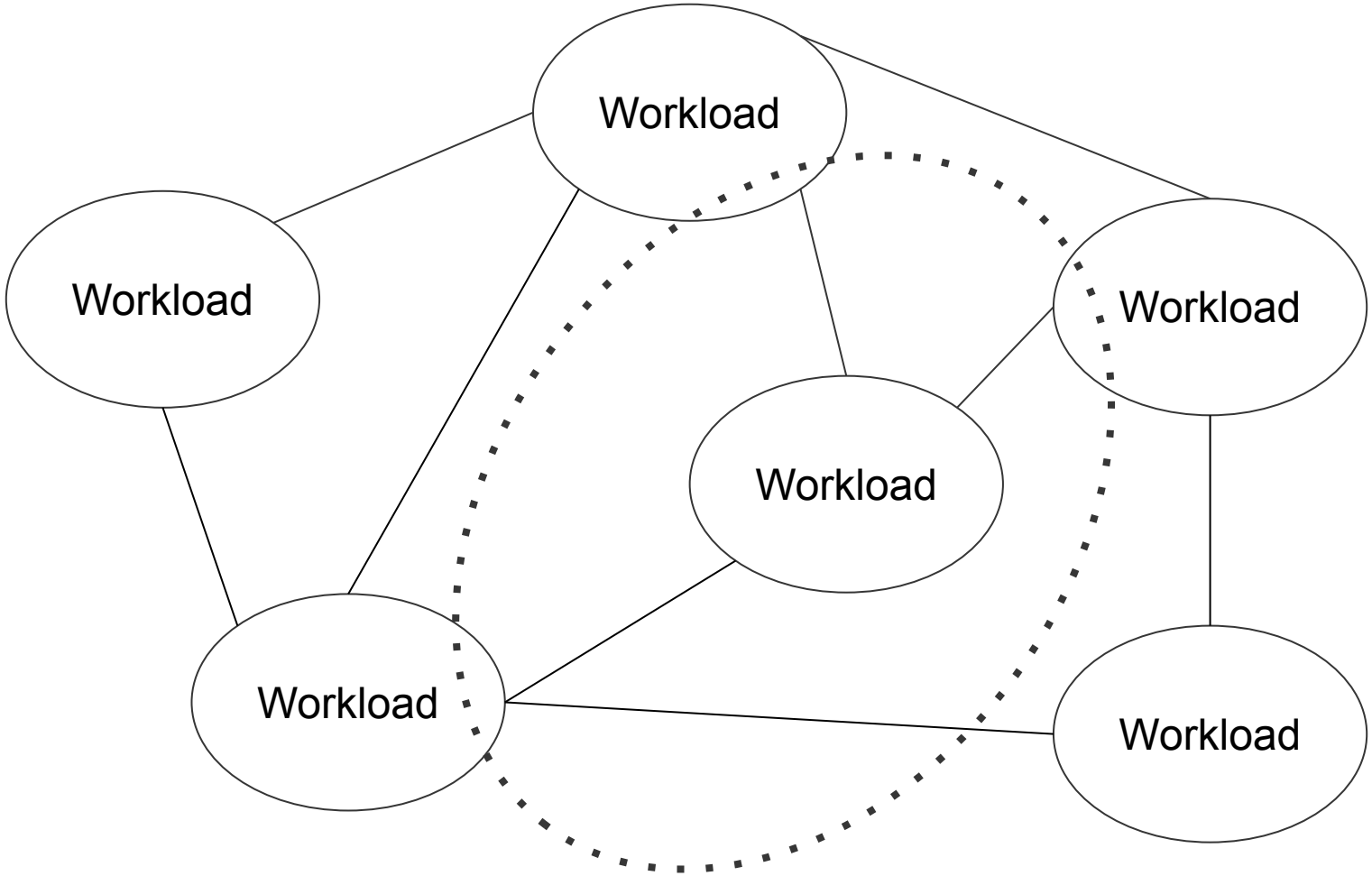
Graph change events




```
Event 1: Deploy 1 unit of app A
Event 2: Deploy 1 unit of app B
Event 3: Relate app A and app B
Event 4: appA/0 and appB/0 observe each other
Event 5: Scale app A: add 1 unit
Event 6: Scale app A: add 1 unit
Event 7: appA/1 and appB/0 observe each other
```

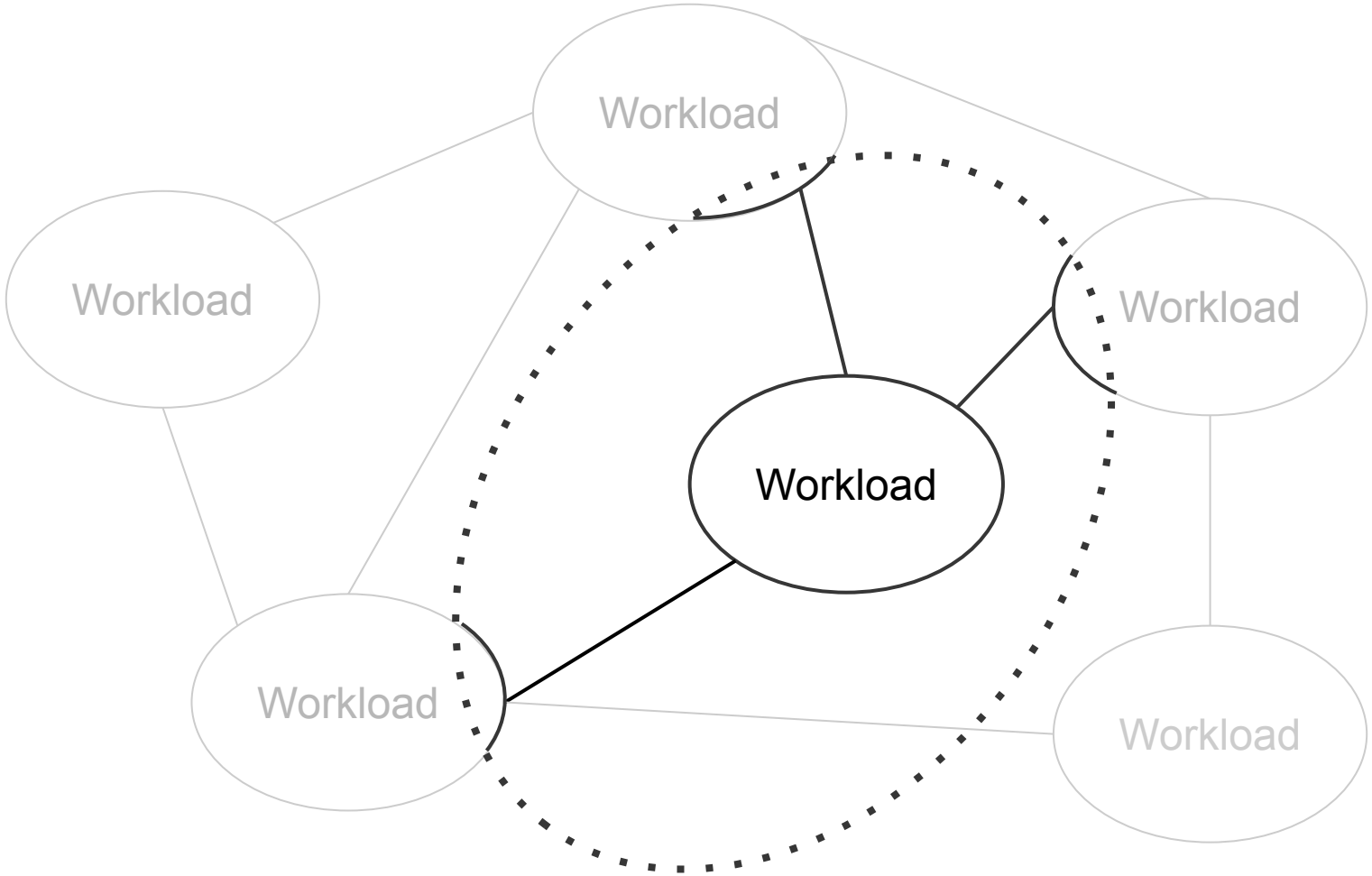
© ETSI 2020

Event Horizon

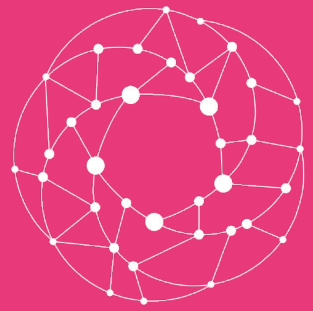


© ETSI 2020

Event Horizon



© ETSI 2020



Open Source
MANO

How to write a charm?

© ETSI 2020

ETSI

Charm Declaration

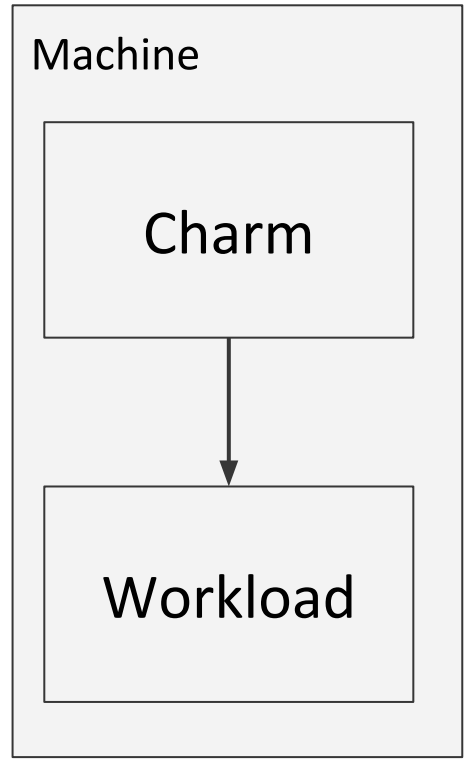
```
my-charm/
    metadata.yaml
    config.yaml
    ...
    src/charm.py
    lib/
      ...python modules
```

```
from ops.main import main
from ops.charm import CharmBase


class MyCharm(CharmBase):

    ...


if __name__ == '__main__':
    main(MyCharm)
```

© ETSI 2020

Installing a Workload on a Machine

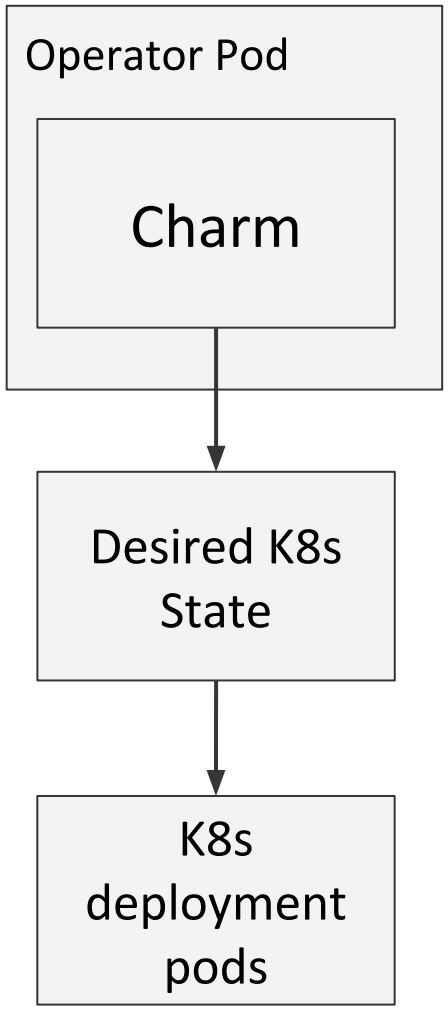
```
class LBCharm(CharmBase):

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

    def on_install(self, event):
        packages = self.model.config["packages"]
        # install the specified packages...
        apt_install(packages)
```



© ETSI 2020

Installing a K8s Workload

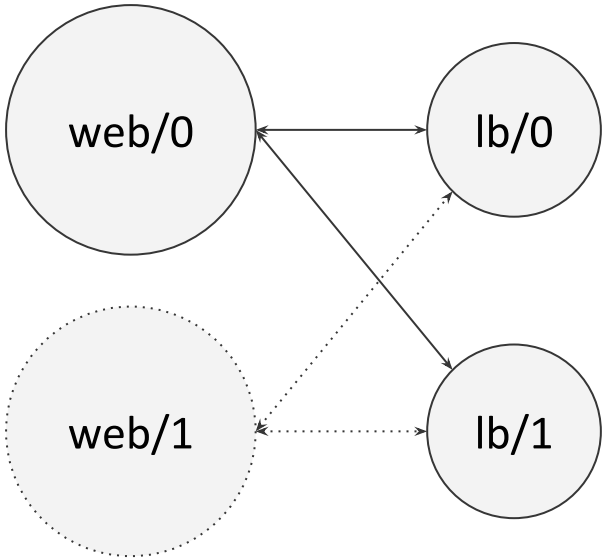
```
def on_install(self, event):

    self.model.pod.set_spec({
        'containers': [{
            'name': self.meta.name,
            'ports': [{
                'containerPort': self.model.config['http_port'],
                'protocol': 'TCP',
                }],
            'imageDetails': {
                'imagePath': self.model.config['image-path']
                'username': self.model.config['registry-user']
                'password': self.model.config['registry-password']
            },
            'config': {
                'CONFIG_KEY': 'CONFIG_VALUE', # ...
            }
    })
```



© ETSI 2020

Scaling

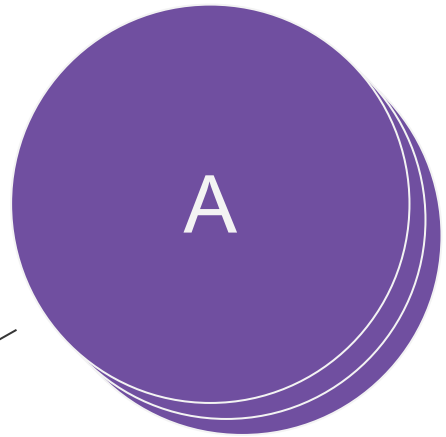
```
class LBCharm(CharmBase):

    # web/x unit added to the model and seen by a load-balancer
    def on_backend_relation_joined(self, event):
        relation_data = event.relation.data[event.unit]
        backend_addr = relation_data.get('ingress-address')

        # Trigger load-balancer re-configuration
        self.on.backend_added.emit(backend_addr)
```

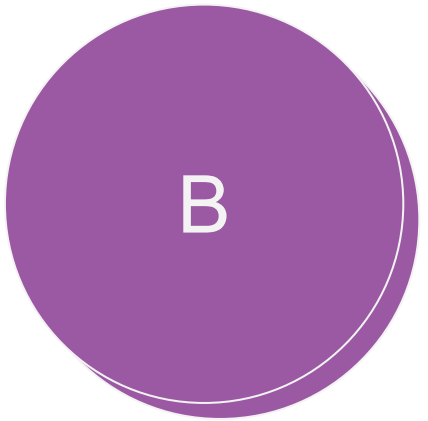


© ETSI 2020

Relation Data

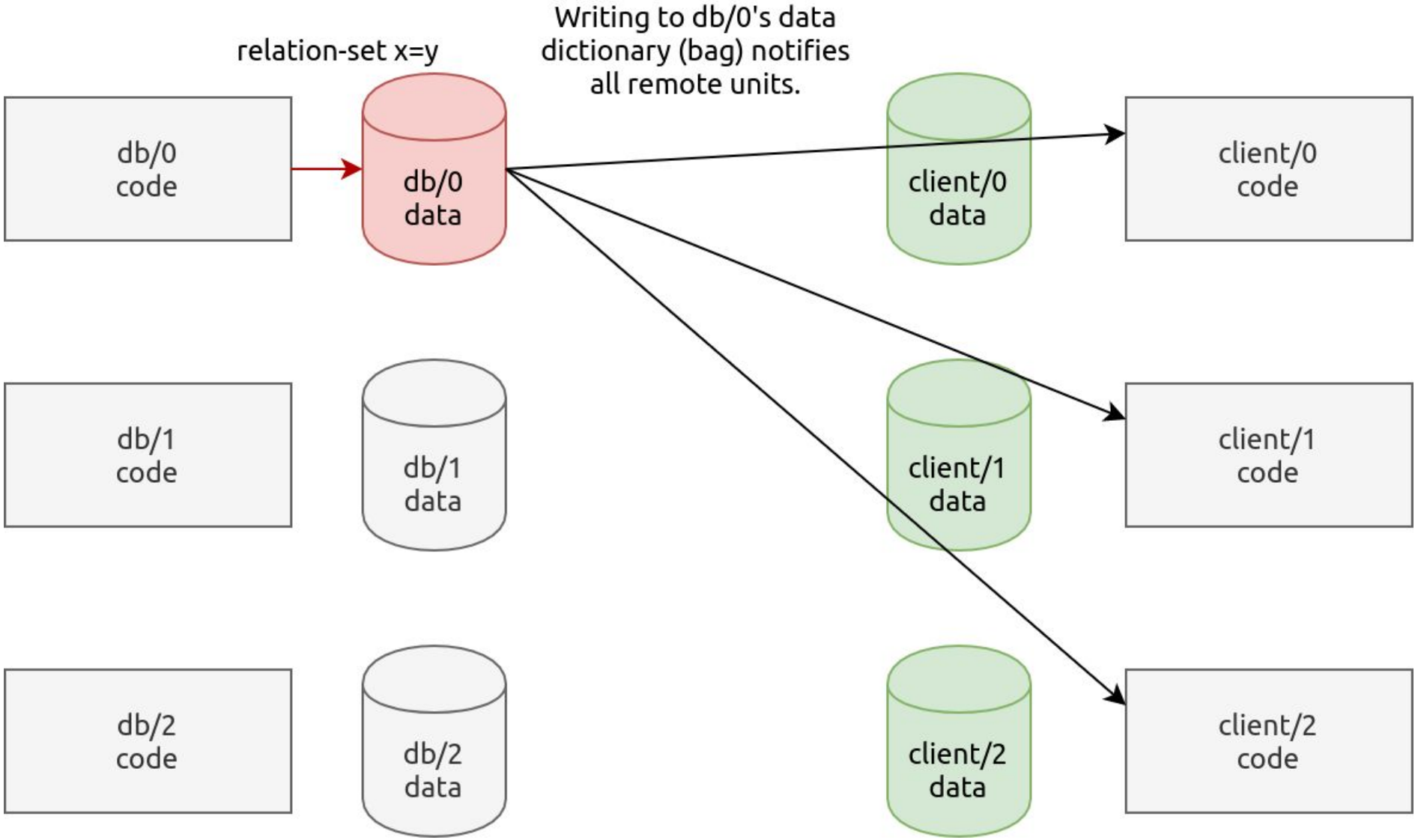
App A Data	App B Data
A/0 Data A/1 Data A/2 Data	B/0 Data B/1 Data





© ETSI 2020

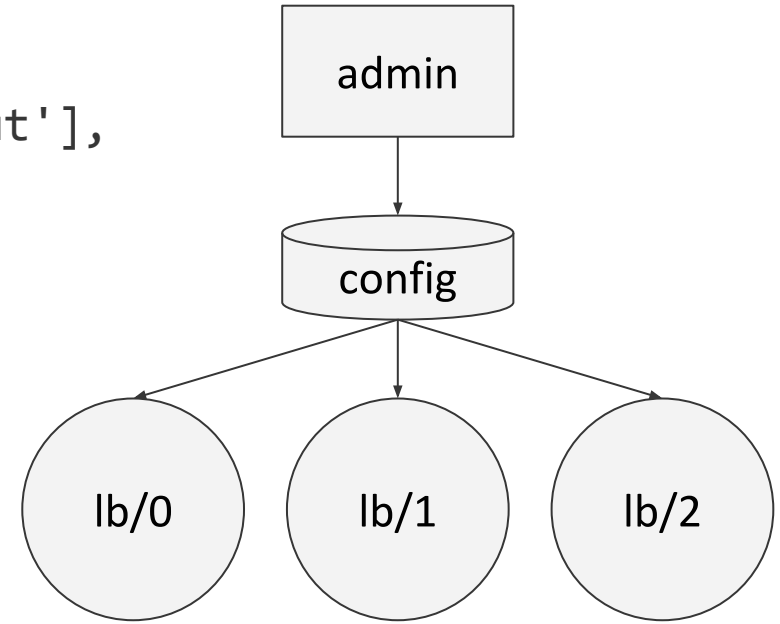
Relation Data Change Event



© ETSI 2020

Configuration Changes

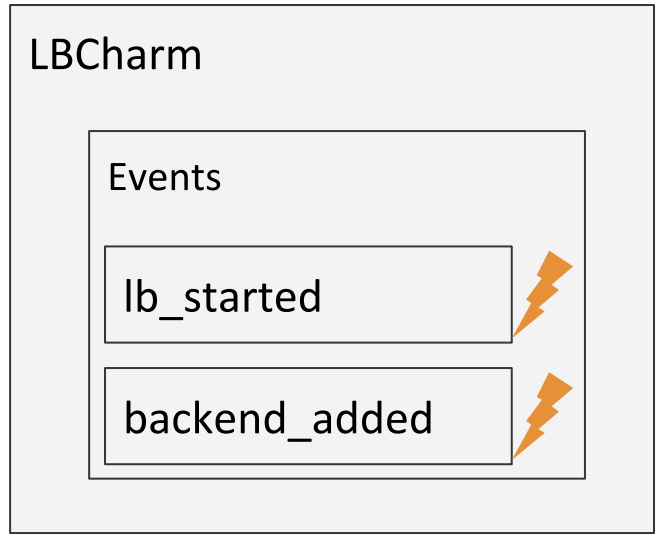
```
class LBCharm(CharmBase):
    def on_config_changed(self, event):
        ctxt = {
            'request_timeout': self.model.config['request-timeout'],
            # Other template context values...
        }
        # Render a jinja2 template with a provided context.
        env = Environment(loader=FileSystemLoader('templates'))
        template = env.get_template('haproxy.conf.j2')
        rendered_content = template.render(ctxt)
```



© ETSI 2020

Charm-defined Events

```
class LBStarted(EventBase): ...

class LBCharmEvents(EventsBase):
    lb_started = EventSource(LBStarted)

class LBCharm(CharmBase):
    on = LBCharmEvents()

    def on_start(self, event):
        # start the lb service ...
        self.on.lb_started.emit()
```



© ETSI 2020

Actions

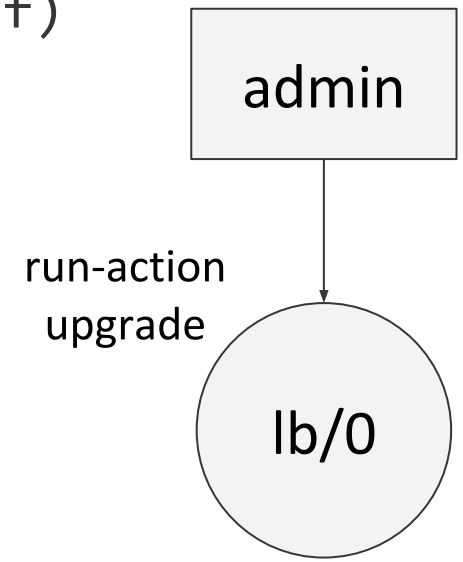
- Admin-triggered one-time events;
- Action event names have an "_action" suffix:

```
self.framework.observe(self.on.upgrade_action, self)
```

- **ActionEvent** type exposes a useful API to work with actions:

```
def on_upgrade_action(self, event):
    my_param = event.params["my_param"]

    event.log("Upgrade progress: 42%")

    event.set_results({"success": "false"})
    event.fail("Almost got it but not quite.")
```

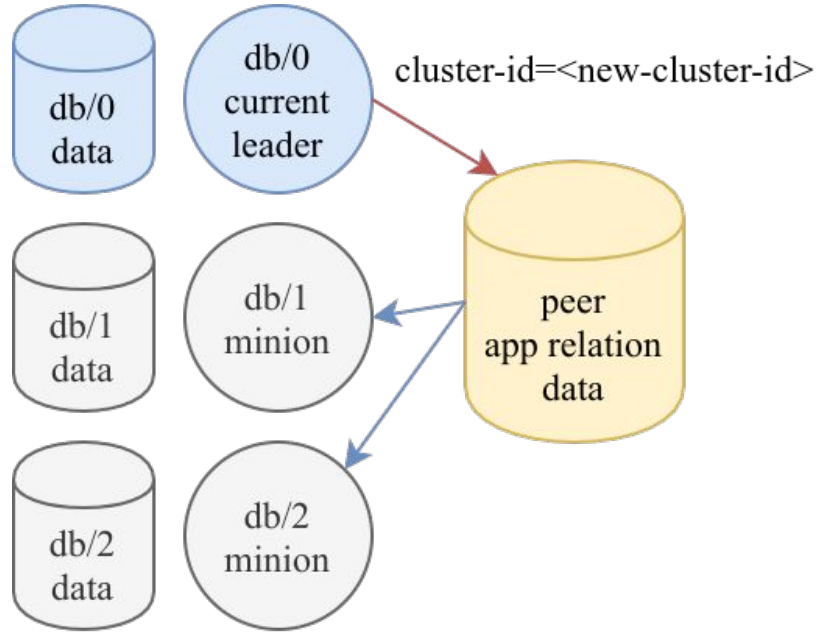
- ActionEvents cannot be deferred.



© ETSI 2020

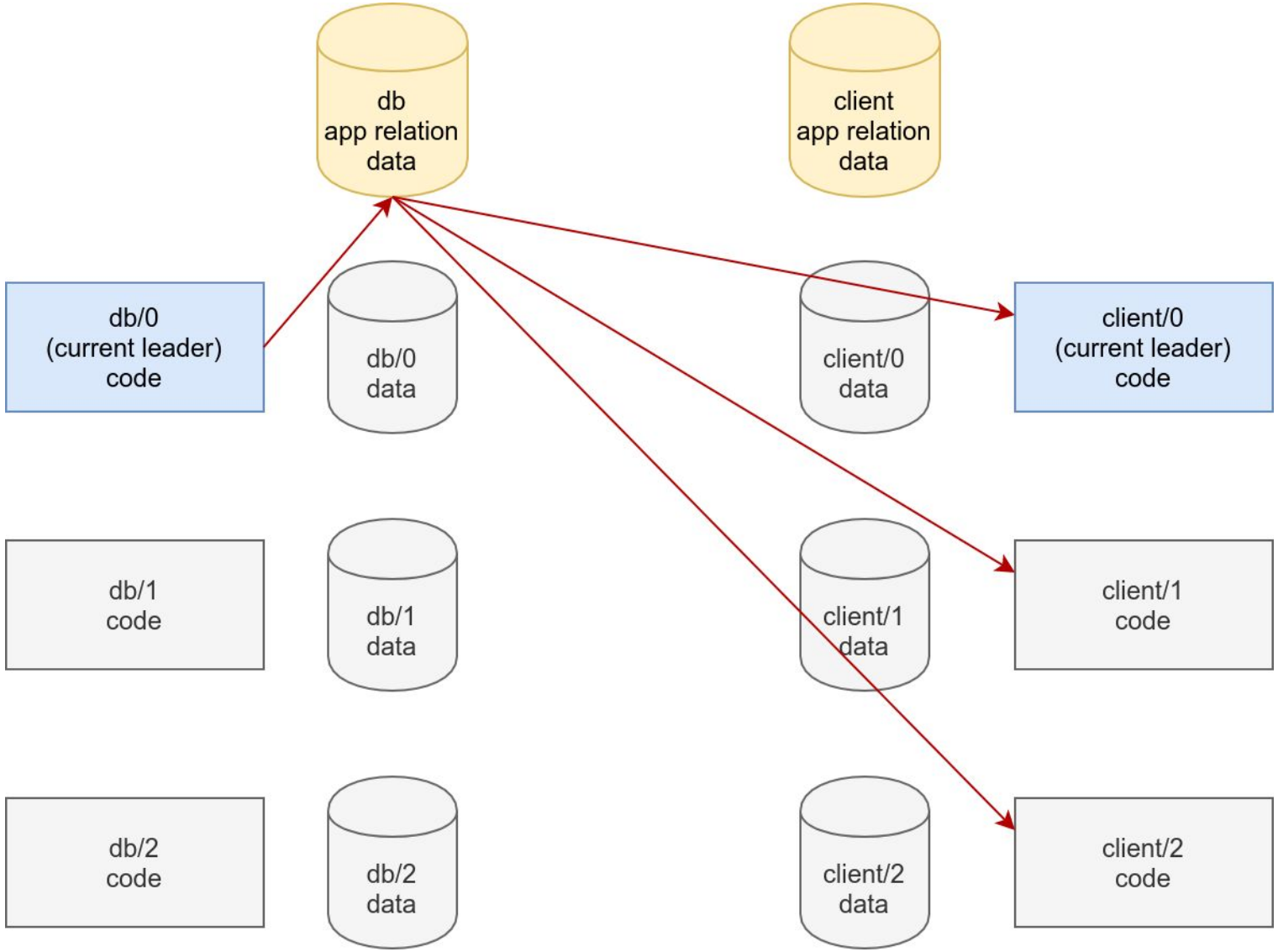
Clustering and Peer App Relation Data

```
class DBCharm(CharmBase):


    def on_cluster_initialized(self, event):
        if not self.model.unit.is_leader():
            raise RuntimeError('initial unit isn’t a leader')


        cluster_relation = self.model.get_relation('cluster')
        cluster_app_data = cluster_relation.data[self.model.app]
        cluster_app_data['initial_unit'] = self.model.unit.name
        cluster_app_data['cluster_id'] = event.cluster_id
```



© ETSI 2020

Application Relation Data: Example



© ETSI 2020

Application Relation Data

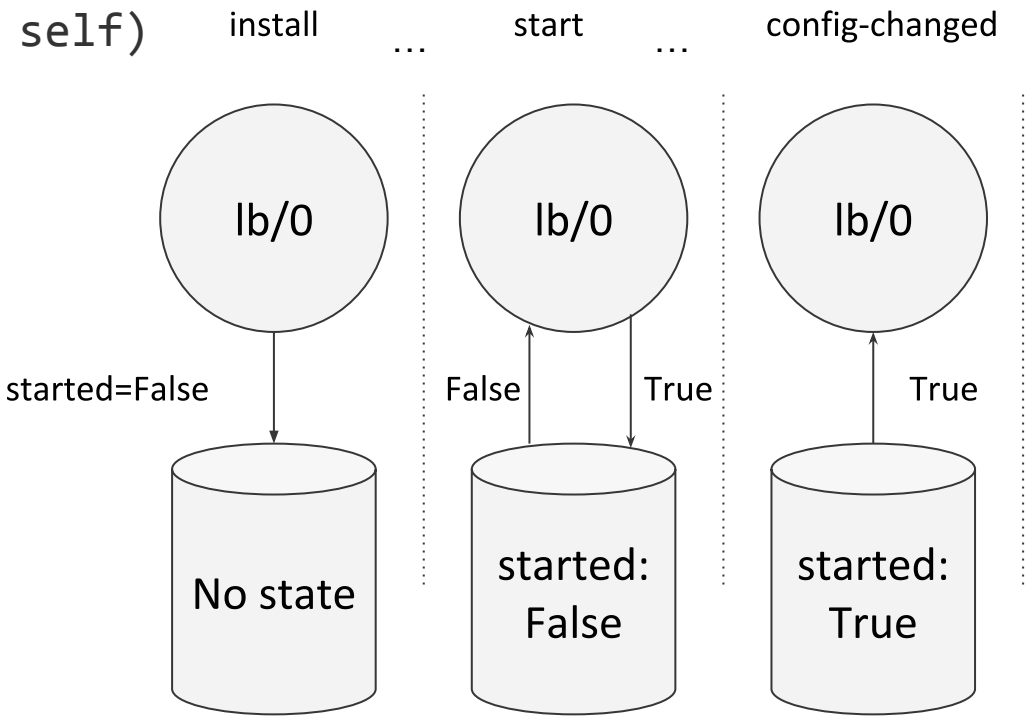
- Relation data of an app per relation;
- Only modifiable by a leader unit;
- Each side of a relation has its own leader unit and app relation data modifiable by that unit;
- App data on a **peer relation** allows data written by a leader unit to be exposed to peers;
- Model representation:

```
rel = self.model.get_relation('cluster')
app_data = rel.data[rel.app]
cluster_id = app_data.get("cluster-id")
```

© ETSI 2020

Stored State: Example

```
class LBCharm(CharmBase):

    state = StoredState()

    def __init__(self, *args):
        self.framework.observe(self.on.start, self)
        self.framework.observe(self.on.config_changed, self)
        self.state.set_default(started=False)

    def on_start(self, event):
        # start the LB service…
        self.state.started = True

    def on_config_changed(self, event):
        if self.state.started:
        # write config files and reload...
```



© ETSI 2020

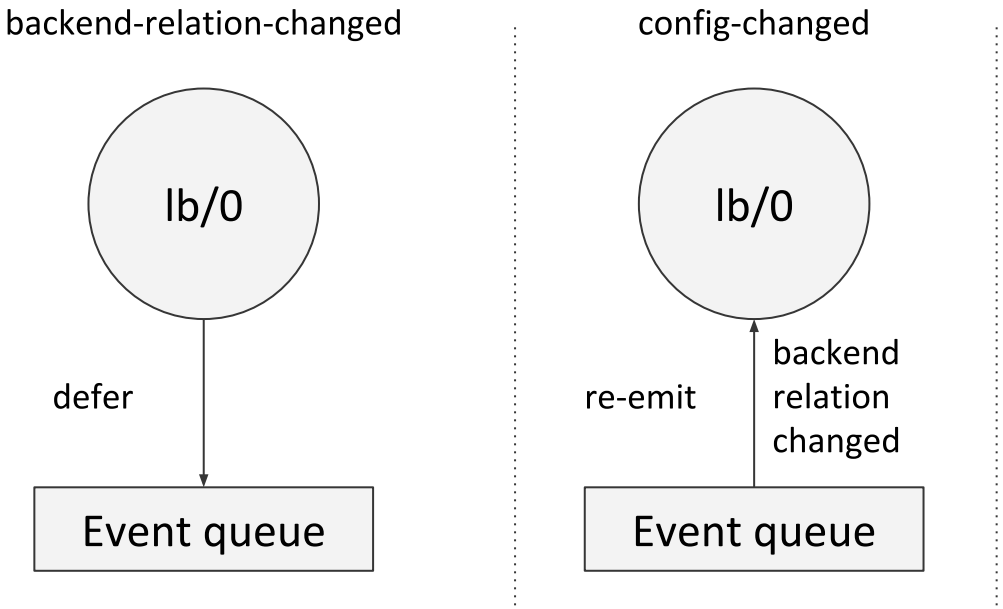
Stored State: Summary

- Each framework **Object** has a **Handle** as its persistent ID;
- Object data is persisted via attributes of type StoredState;
- Only basic Python types may be stored (lists, sets, dicts, ...);
- **StoredState.set_default** is available to initialize the state once.

© ETSI 2020

Deferring Events

```
class LBCharm(CharmBase):

    state = StoredState()


    def backend_relation_changed(self, event):
        if self.state.tls_configured:
            # write config files and reload...
        else:
            # will fire on the next hook execution
            event.defer()
            return
```



© ETSI 2020

Model Representation

```
Model
    unit
    app
    config
    relations[name]
        relation
            name
            id
            data[unit or app]
            units
    storages
    resources
```

© ETSI 2020

Convenience Methods of a Model

Open Source MANO

Easy lookup of relevant objects:

```
def on_config_changed(self, event):
    # raises an exception if multiple relations exist
    db_relation = self.model.get_relation("db")

    binding = self.model.get_binding(db_relation)
    addr = binding.network.ingress_address
```

© ETSI 2020

Creating a "Hello World" Charm Structure


```
mkdir charm-hello-world && cd charm-hello-world && git init

mkdir hooks src lib mod

git submodule add https://github.com/canonical/operator mod/operator

touch metadata.yaml config.yaml src/charm.py

chmod +x src/charm.py


ln -s ../mod/operator/ops lib/ops

ln -s ../src/charm.py hooks/install
```

© ETSI 2020

"Hello World" Charm Metadata

Open Source MANO

metadata.yaml:

```
name: hello-world
summary: a "Hello, World" charm.
description: "Hello, World."
series:
    - bionic
```

© ETSI 2020

"Hello World" Charm Code

https://github.com/dshcherb/charm-hello-world/blob/master/src/charm.py

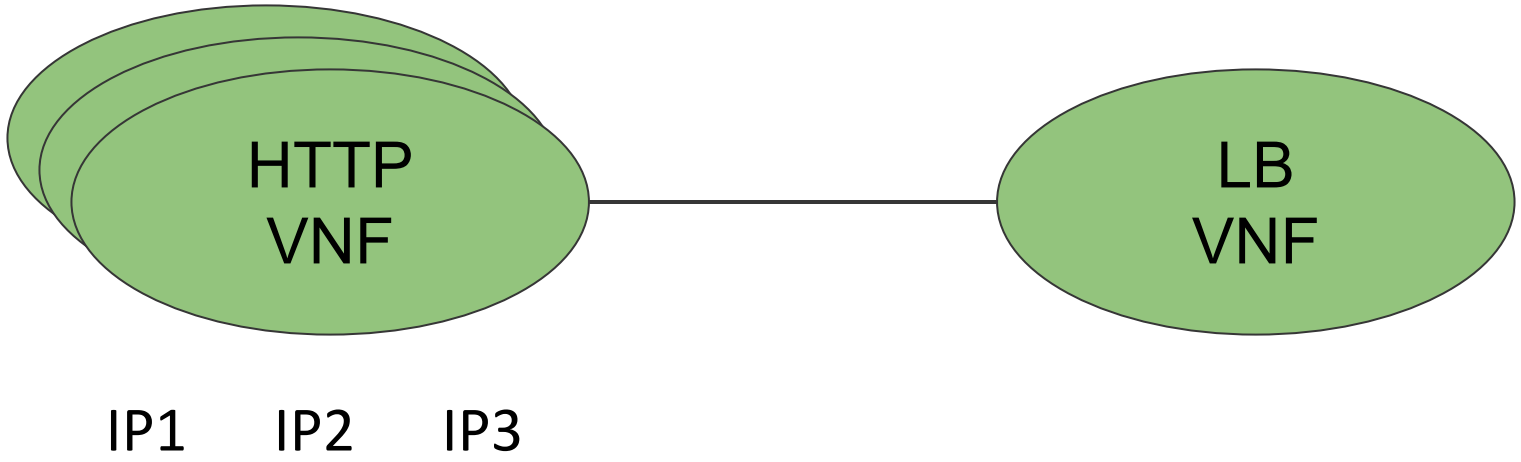
```
# ...
    def __init__(self, *args):
        super().__init__(*args)
        self.state.set_default(logged_hello=False)
        self.framework.observe(self.on.install, self.on_install)

    def on_install(self, event):
        logger.info('Hello, world')
        self.state.logged_hello = True
```

© ETSI 2020

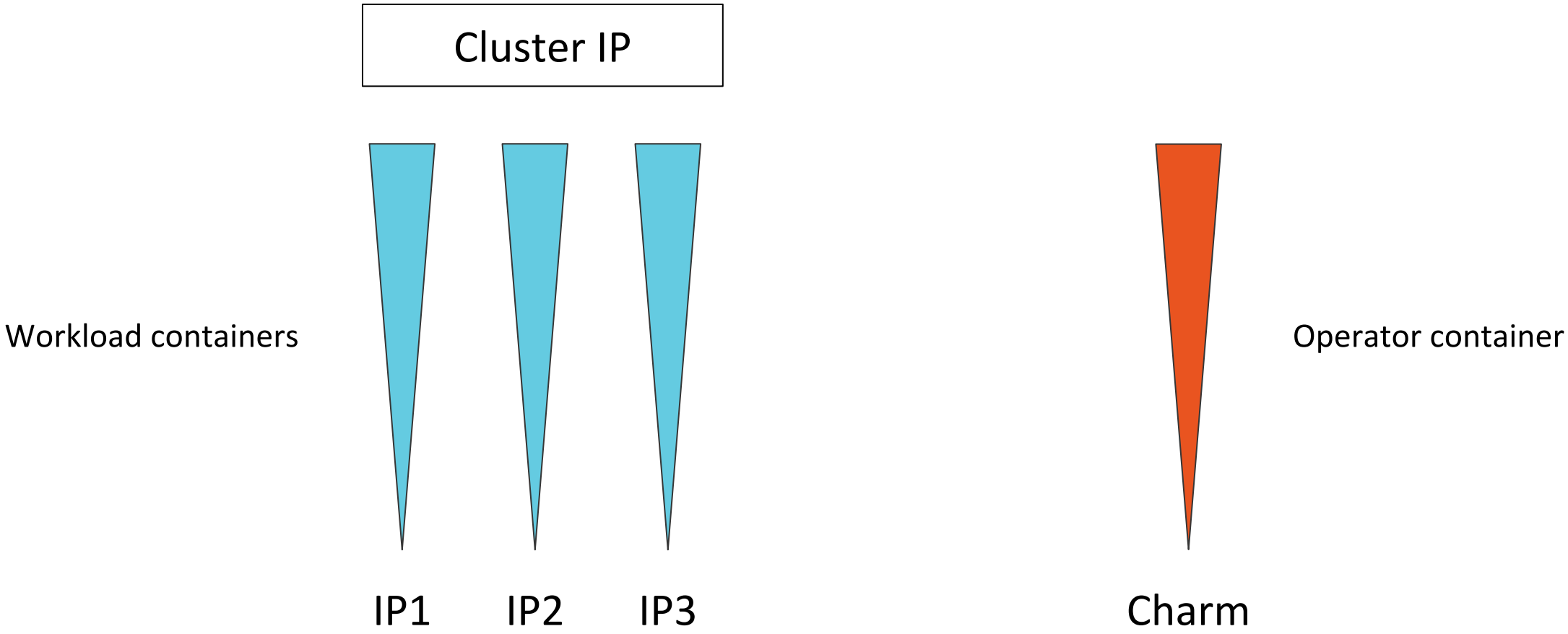
Question

If you have an HTTP service scaled to 1 VDU, and then it’s scaled out to 3 VDUs, does OSM balance the load between the workloads?

© ETSI 2020



© ETSI 2020


Cluster IP
Workload containers
IP1
IP2
IP3
Charm
Operator container

© ETSI 2020

Get ready for hands-on

Ensure you have an Ubuntu installed

Options:

- AWS instance
- Multipass
- VM in Virtualbox

Track your progress here:

https://docs.google.com/spreadsheets/d/1dXpPJH6XfWg8GHfQyNyT_49HSh28dvgVasOe62RGU8o/edit?usp=sharing

© ETSI 2020

Open Source
MANO

Operations in workloads

© ETSI 2019

ETSI

Let’s get hands-on: Initial setup

```
sudo snap install juju --classic
sudo snap install microk8s --classic
sudo adduser ubuntu microk8s
newgrp microk8s
microk8s.status --wait-ready
microk8s.enable storage dns
juju bootstrap microk8s
juju add-model osm
juju deploy osm
```

© ETSI 2020

Operating workloads

Open Source MANO

 # Lifecycle

```
juju scale-application nbi-k8s 3
```

 # Configuration

```
juju config nbi-k8s image=opensourcemano/nbi:7.0.1
```

 # Operation

```
juju run-action mongodb-k8s/0 backup
```

 # Integration

```
juju deploy cs:~charmed-osm/keystone-k8s
juju relate nbi-k8s keystone-k8s
juju relate keystone-k8s mariadb-k8s
```

© ETSI 2020

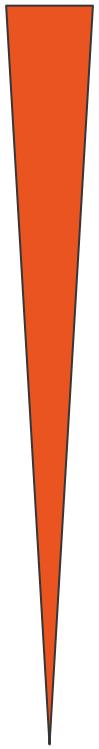
Charms walkthrough

- Zookeeper charm:
  - https://github.com/charmed-osm/charm-k8s-zookeeper
- Kafka charm:
  - https://github.com/charmed-osm/charm-k8s-kafka
- Zookeeper interface
  - https://github.com/charmed-osm/interface-zookeeper

© ETSI 2020

VNFD walkthrough

- K8s charm
  - Base code

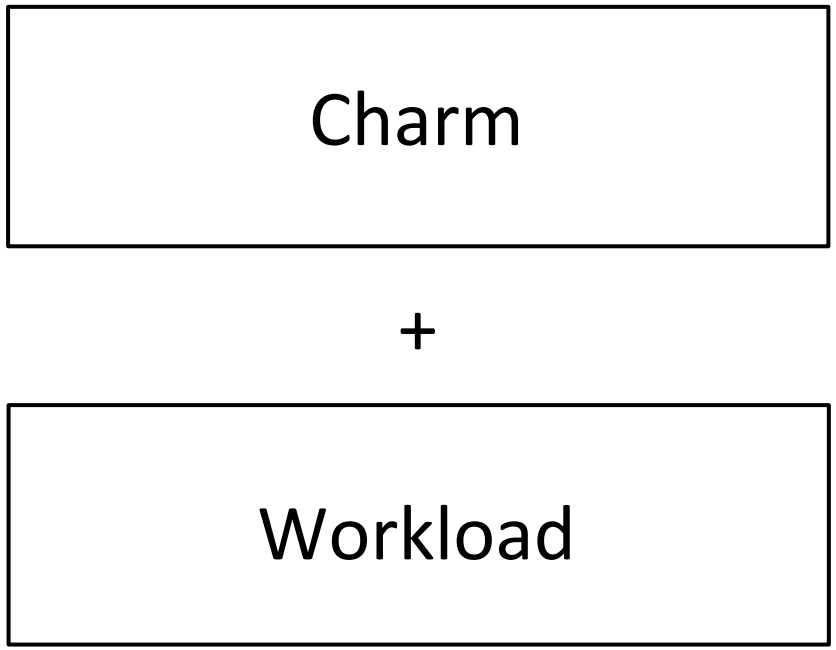


```
vnfd-catalog:
    vnfd:
    -   id: hackfest-simple-k8s-vnfd
        name: hackfest-simple-k8s-vnfd
        connection-point:
        -   name: mgmtnet
        mgmt-interface:
            cp: mgmt
        kdu:
        -   name: mykdu
            juju-bundle: cs:~dominik.f/bundle/hf-k8s-bundle-0
        k8s-cluster:
            nets:
            -   id: mgmtnet
                external-connection-point-ref: mgmt
```

© ETSI 2020

VNFD walkthrough

- Proxy charm
  - Base code

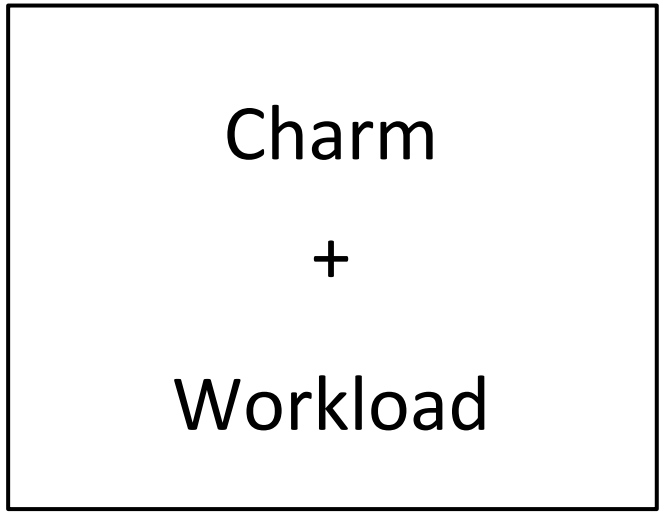



```
vnfd:vnfd-catalog:
    vnfd:
    - ...
      mgmt-interface:
        cp: vnf-mgmt
      ...
      vnf-configuration:
        juju:
          charm: simple-proxy
        initial-config-primitive:
          - seq: "1"
            name: config
            parameter:
              - name: ssh-hostname
                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"
        config-primitive:
          - name: touch
            parameter:
              - name: filename
                default-value: "/home/ubuntu/touched"
                data-type: STRING
```

© ETSI 2020

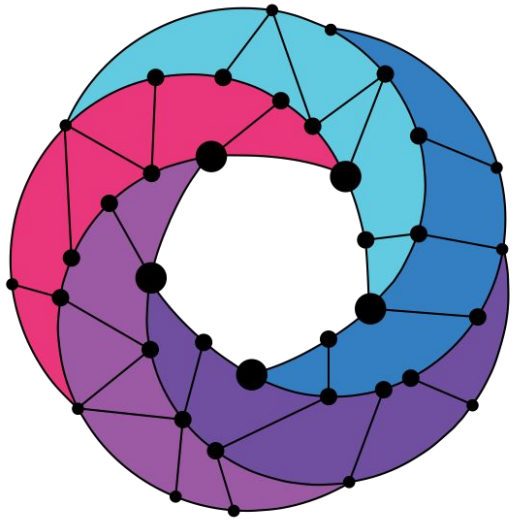
VNFD walkthrough

- Native charm
  - Base code




```
vnfd:vnfd-catalog:
    vnfd:
    - ...
      mgmt-interface:
        cp: vnf-mgmt
      vdu:
        - id: mgmtVM
          ...
          cloud-init-file: cloud-config.txt
          vdu-configuration:
            juju:
              charm: simple-native
              proxy: False
            config-access:
              ssh-access:
                required: True
                default-user: ubuntu
            initial-config-primitive:
              - seq: "1"
                name: touch
                parameter:
                  - name: filename
                    value: "/home/ubuntu/first-touch"
            config-primitive:
              - name: touch
                parameter:
                  - name: filename
                    data-type: STRING
                    default-value: "/home/ubuntu/touch"
```

© ETSI 2020



Open Source MANO

Find us at:

osm.etsi.org

osm.etsi.org/wikipub

© ETSI 2020