Introduction to OSM Primitives

David Garcia (Canonical)
Primitives are actions exposed by the operator
Operator Pattern

Install Configure
Upgrade Integrate
Backup Restore

MySQL®

Human operator
Operator Pattern

Install  Configure
Upgrade  Integrate
Backup   Restore

Human operator

MySQL®
Operator Pattern

Charm

- Install
- Upgrade
- Backup

- Configure
- Integrate
- Restore

MySQL®
Charms are universal operators
Reality is messy and mixed
One VNF is many apps and integration

VNFD

Metadata

App Charm

App Charm

App Charm

App Charm
VNFD

Metadata

“Charms” - operations code
- Lifecycle
- Configuration
- Integration
- Actions / Primitives

DECLARATIVE

CODE
VNFD

Metadata

“Charms” - operations code
- Lifecycle
- Configuration
- Integration
- Actions / Primitives
Primitives are actions exposed by the operator
OSM Primitives are **actions exposed by the Charm**
Operating “proxy” workloads

‘Operator’ instance

Proxy Charm

Workload
PNF/VNF/KNF

OSM

Where can we run our own operations code for this workload?
Operating “native” workloads

Workloads optimised for OSM have a charm that drives the workload directly.
Integration is first-class in the VNFd

Lines of integration in the VNFd
Charms declare typed integration points

“I can use a MySQL database”

“I can use LDAP”

“I can send my logs to a syslog”
Charms declare typed integration points

“I can use LDAP”

“I can provide LDAP”
Matching integration points can be related

Lines of integration between matching integration points on different charms
Composition gives complex integrations
VNFds can describe complex integrations

Lifecycle scripts
Config scripts
Integration scripts
Action scripts
OSM primitives are Charm Action scripts

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

Each ‘primitive’ is a charm action script that takes parameters and produces output.
Charms are packages of scripts to drive apps

Lifecycle scripts
- install
- config
- update
- remove
- scale

"Action" scripts are OSM Primitives
- "action: backup"
- "action: restore"
- "action: scan-viruses"
- "action: health-check"
- "action: add-repo"
- "action: ...
- "action: ...
- "action: ...

Integration scripts
- relate-mysql
- relate-ldap
- relate-proxy
- relate-...

These are your operations primitives.

© ETSI 2020
Charm describes Action parameters

Charm metadata describes the action parameters.

Each Action is a script, usually in Python or Bash.

```
addurl:
  description: "Add squid config"
  params:
    url:
      description: "URL that will be allowed"
      type: string
      default: ""

deleteurl:
  description: "Delete allowed URL squid config"
  params:
    url:
      description: "URL that will stop to be allowed"
      type: string
      default: ""
```
#!/bin/bash

URL=`action-get url`

if ! grep -Fxq "http_access allow allowedurls" /etc/squid/squid.conf
then
    sed -i '/^# And finally deny all .*/i http_access allow allowedurls\n' /etc/squid/squid.conf
fi

sed -i "/^http_access allow allowedurls.*$/i acl allowedurls dstdomain \.$URL" /etc/squid/squid.conf

kill -HUP `cat /var/run/squid.pid`
def on_deleteurl_action(self, event):
    """Handle the deleteurl action."""
    url = event.params["url"]

    line_to_delete = "acl allowedurls dstdomain .{}".format(url)
    line_deleted = False

    with open("/etc/squid/squid.conf", "r") as f:
        lines = f.readlines()
    with open("/etc/squid/squid.conf", "w") as f:
        for line in lines:
            if line_to_delete not in line:
                f.write(line)
            else:
                line_deleted = True

    if line_deleted:
        event.set_results({"output": "URL deleted succesfully"})
        subprocess.check_output(
            "kill -HUP `cat /var/run/squid.pid`", shell=True)
    else:
        event.fail("No URL was deleted")
Juju Controller

Operator Lifecycle Manager
OSM Architecture

NBI
“Northbound Interface”

UI
“User Interface”

LCM
“Lifecycle Manager”

RO
“Resource Orchestrator”

VCA
“VNF Configuration Abstraction”

© ETSI 2020
Juju drives application operations on machine and kubernetes substrates

Install, update, configure, scale, integrate, and actions.
Juju architecture

- Juju client
- Juju Controller
- LCM
- VCA
- Application model
- Charm
- Machine
- Workload
- PNF/VNF/KNF
Juju controller manages multiple models
VCA uses multiple models for scenario
VCA can be high availability
VCA coordinates all OSM Primitives
Configure

https_proxy: xxx
ca_cert: yyy
...

© ETSI 2020
Do backup!

Charm

LCM  VCA

Workload PNF/VNF/KNF

© ETSI 2020
Integration

Integrate!

LCM → VCA

Charm → Charm → Charm → Charm

PNF, VNF, KNF
Reality is messy and mixed