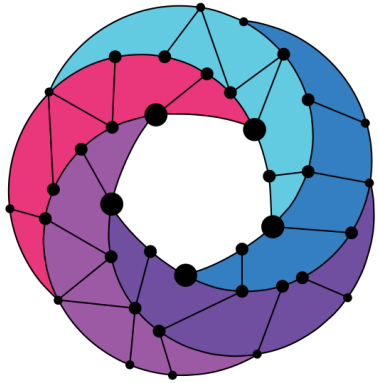


Open Source
MANO

# OSM Hackfest – Pre-session 5
# VNF package generation from command line
# Introduction to charms and VNF primitives

Gerardo García (Telefónica)



ETSI

Open Source
MANO

# How to generate VNF package from command line

# Creating VNF package from terminal (1/2)

- Wiki page: https://osm.etsi.org/wikipub/index.php/Creating_your_own_VNF_package_(Release_THREE)
- Clone the devops repo:
  - git clone https://osm.etsi.org/gerrit/osm/devops
- Create a skeleton folder with all the files required for a single-VM VNF package:
  - ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd --image <IMAGE_NAME> -c <VNF_NAME>
- Go to the VNF_NAME_vnfd folder and edit the descriptor
- Add artifacts (charms, icons, cloud-init files, etc.)

# Creating VNF package from terminal (2/2)

- Once done, you can generate the tar.gz VNF package with the command:
  - ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd –N <VNF_NAME>_vnfd
  - Note: the argument -N is optional and is intended to keep the package files after creating the package
- The tool generate_descriptor_pkg.sh, jointly with other tools for VNF package creation and validation, will be distributed in future releases in a package ‘osm-tools’.
- When editing the descriptor, use the IM tree representation of VNFD as a reference:
  - http://osm-download.etsi.org/ftp/osm-doc/vnfd.html

# Validating the VNF descriptor

- A descriptor can be validated against the IM using this command:
  - ./devops/descriptor-packages/tools/upgrade_descriptor_version.py --test <VNF_DESCRIPTOR_FILE>

  Note: this tool might evolve in future releases to have different scripts for upgrade and for validation.

- The tool upgrade_descriptor_version.py requires the python-osm-im package to be installed.
  - Update your /etc/apt/sources.list with the following line:

    deb [arch=amd64] http://osm-download.etsi.org/repository/osm/debian/ReleaseTHREE stable osmclient IM
  - sudo apt-get update
  - sudo apt-get install python-osm-im

Open Source
MANO

# Charms and VNF primitives in OSM

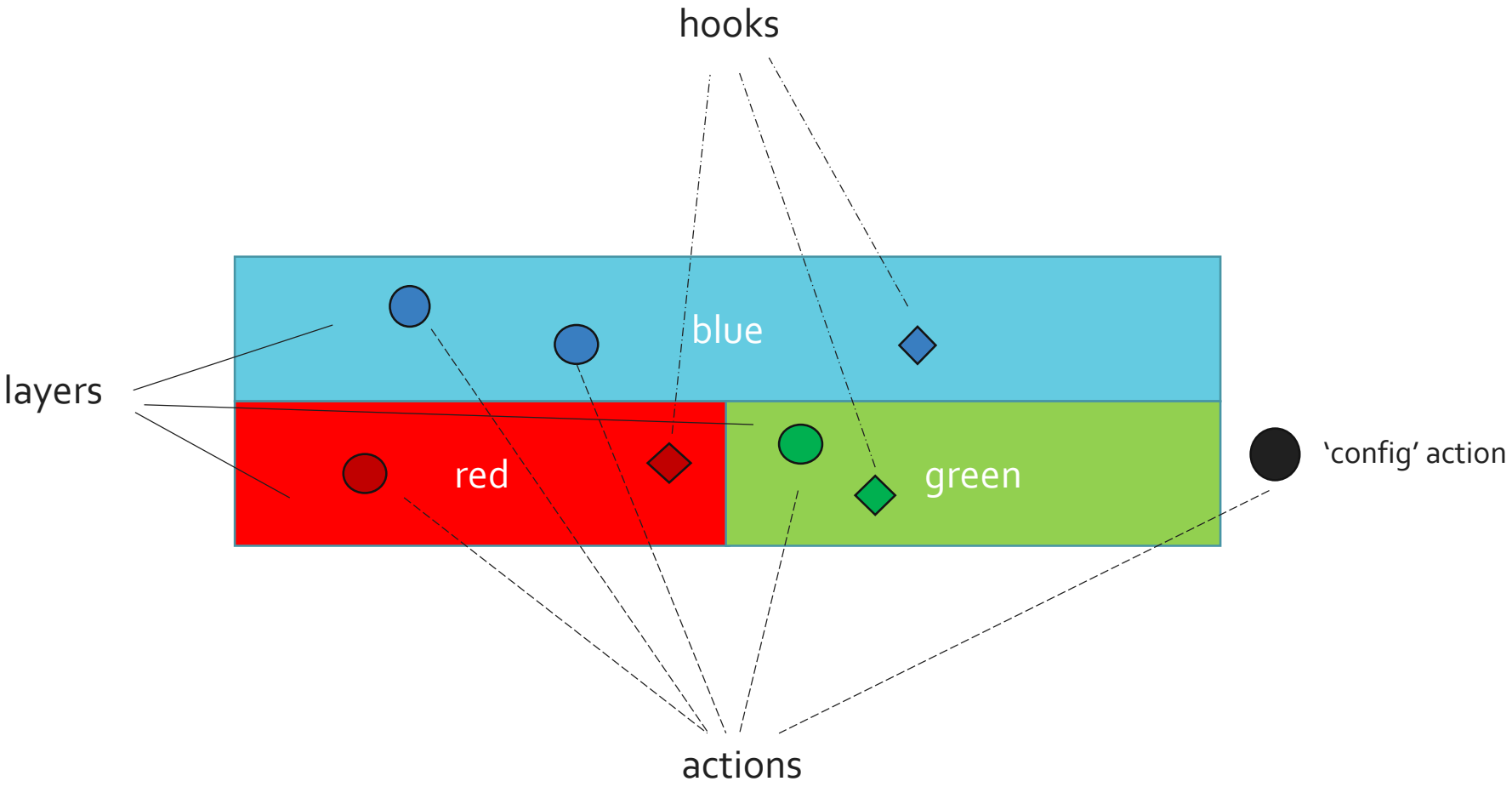


ETSI

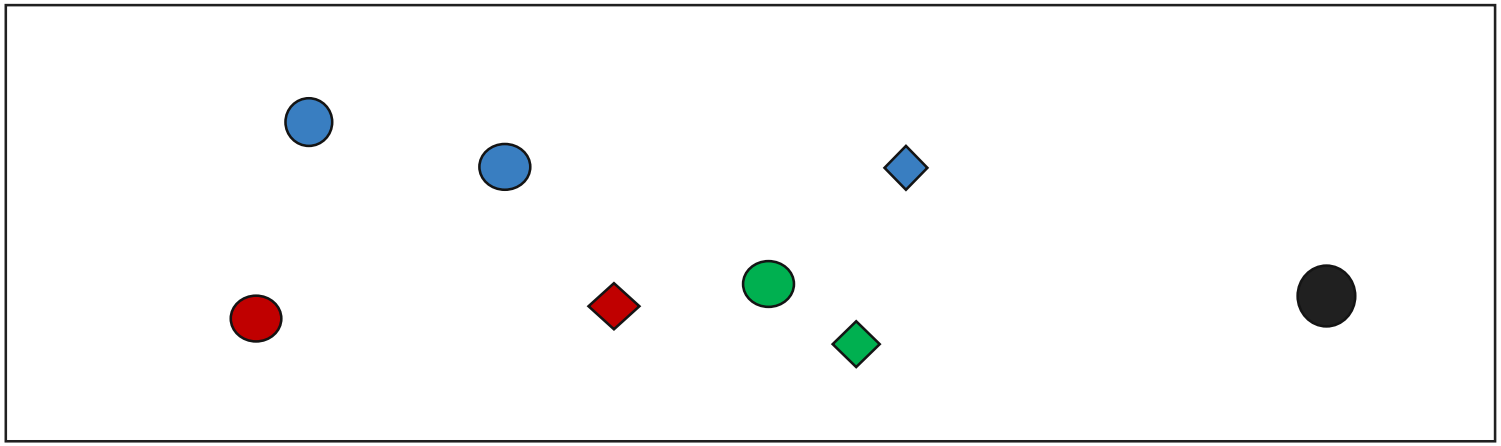
# What is a charm?


- A charm is a set of actions and hooks
  - Actions are programs
  - Hooks are events/signals
- For commodity and reusability, those actions and hooks are grouped in layers
- A charm will always have one layer:
  - That layer has some actions and hooks
  - In addition, that layer can import other layers
- The resulting charm has all the actions and hooks from all the layers joined together, plus additional default actions and hooks (e.g. ‘config’ action)

# What is a charm?
# Charm design



# What is a charm?
# Charm build

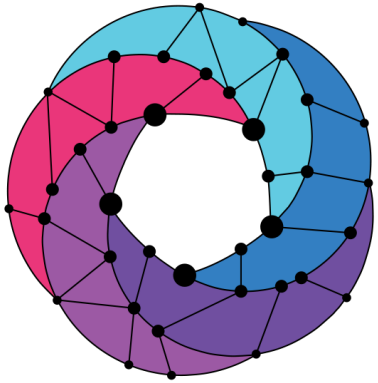


# VNF primitives in OSM

- Initial-config-primitive (day-1): invoked from the SO at instantiation time
- Config-primitives (day-2): invoked from the SO at operator demand (or demanded through the SO NB API e.g. from an OSS)
- Others out of scope (pre and post scaling primitives)

# Mapping between VNF primitives and charm actions and hooks in the descriptor

- VNF primitives have to be mapped to actions in the VNF descriptor
- Initial-config-primitive: maps to a sequence of actions or hooks where the first must be always ‘config’ (action)
- Config-primitives: maps 1to1 to an action
- When writing that mapping in the descriptor, actions and the parameters have to be explicitly written again



Open Source
MANO

Find us at:

osm.etsi.org
osm.etsi.org/wikipub



ETSI