



# Use Case Automation

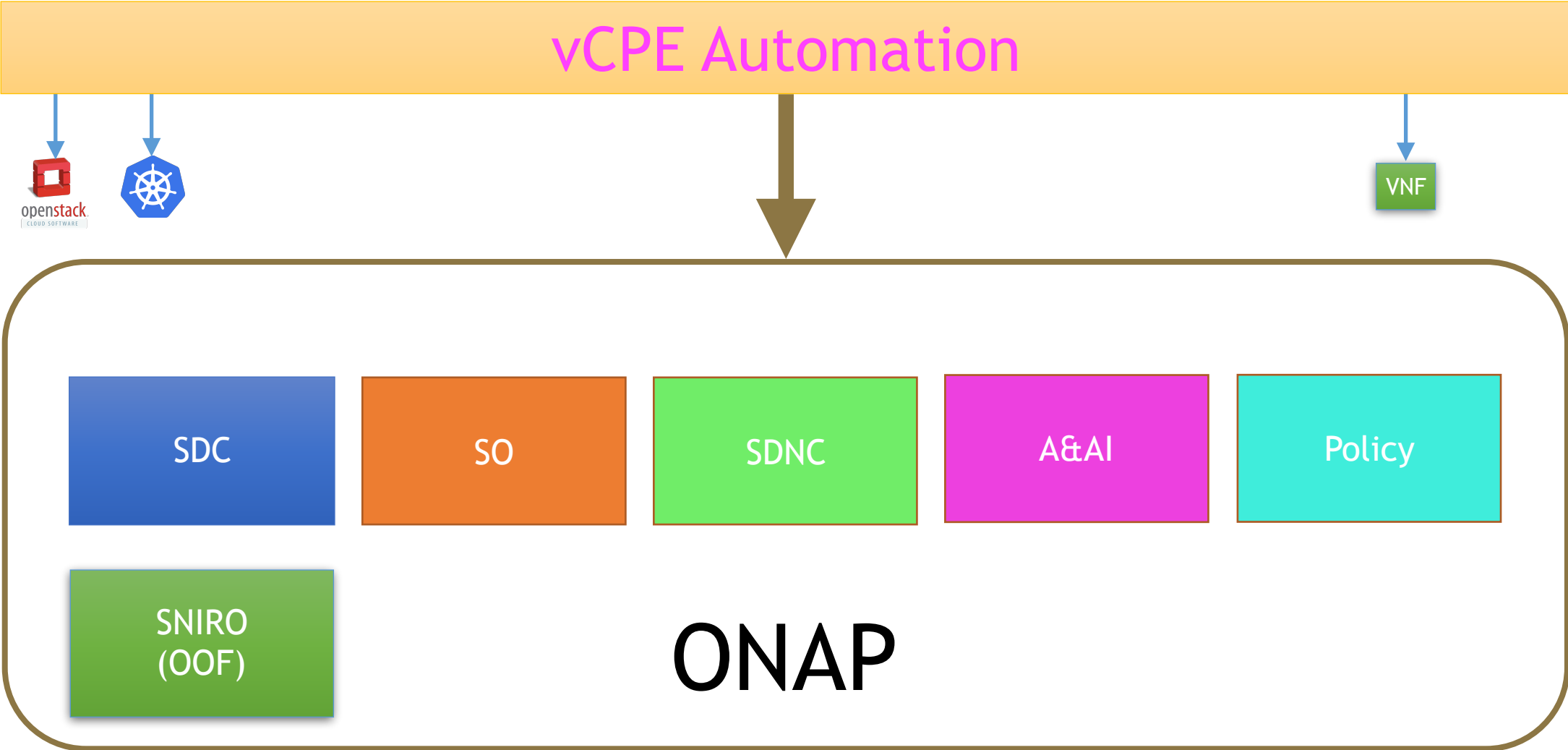
(An example of OSS integration)

Integration Team

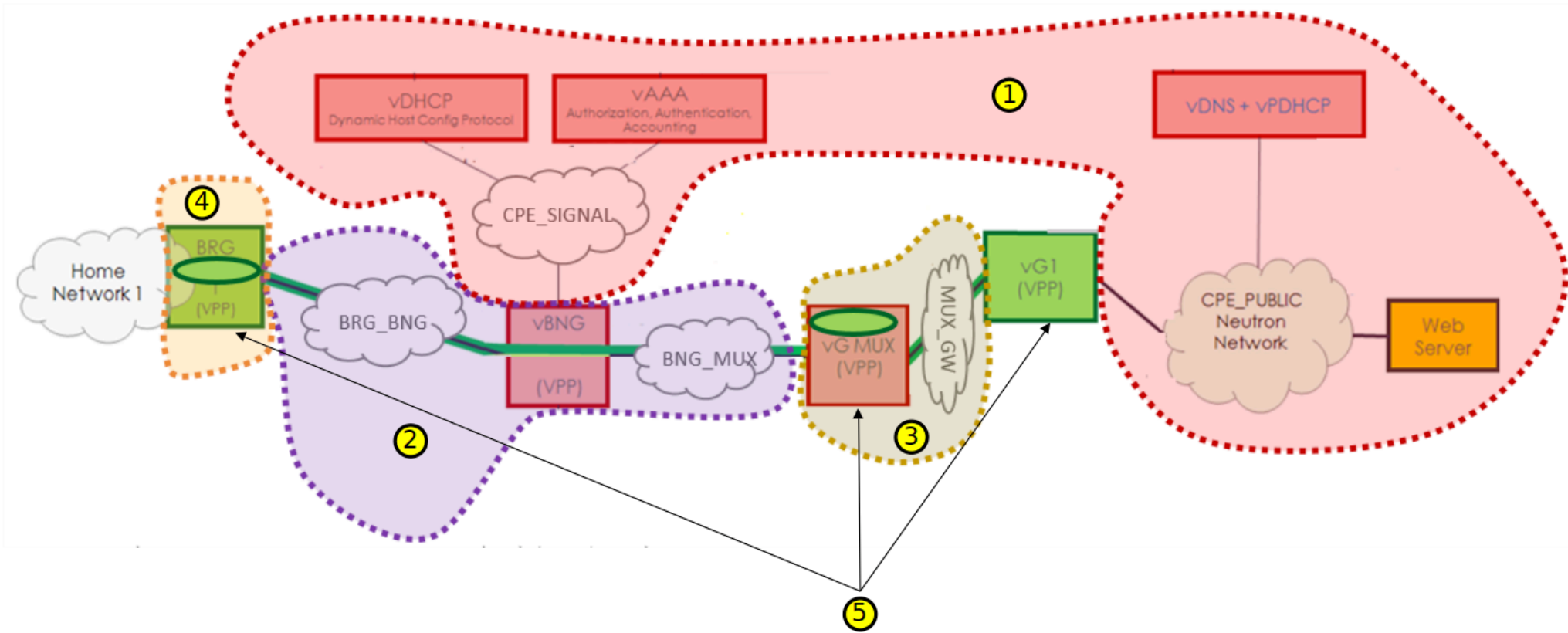
# TSC Must-Have for Dublin

Category	Summary	Owner	Impacted Components	GO/NO GO	Ranking
TSC Must Have	Improve our E2E Process Automat	TSC	AAI, SO, SDC, APPC, VFC, CCSDK + Documentati	GO based on bootcamp details during F2F event	1
	SP3 - Footprint Optimization	TSC	OOM + ALL	GO	1
	CI/CD	TSC	Infrastructure	Already in progress / Demo planned in Paris	1
	Document as You Code	Documentation Project	ALL - Tracked through tickets - Weekly updates	Already in progress	1
	Security by Design	Security Subcommittee	ALL - Part of Milestone checklist	GO - Need to enhance Milestone Checklist & Final	1

# vCPE Use Case Automation

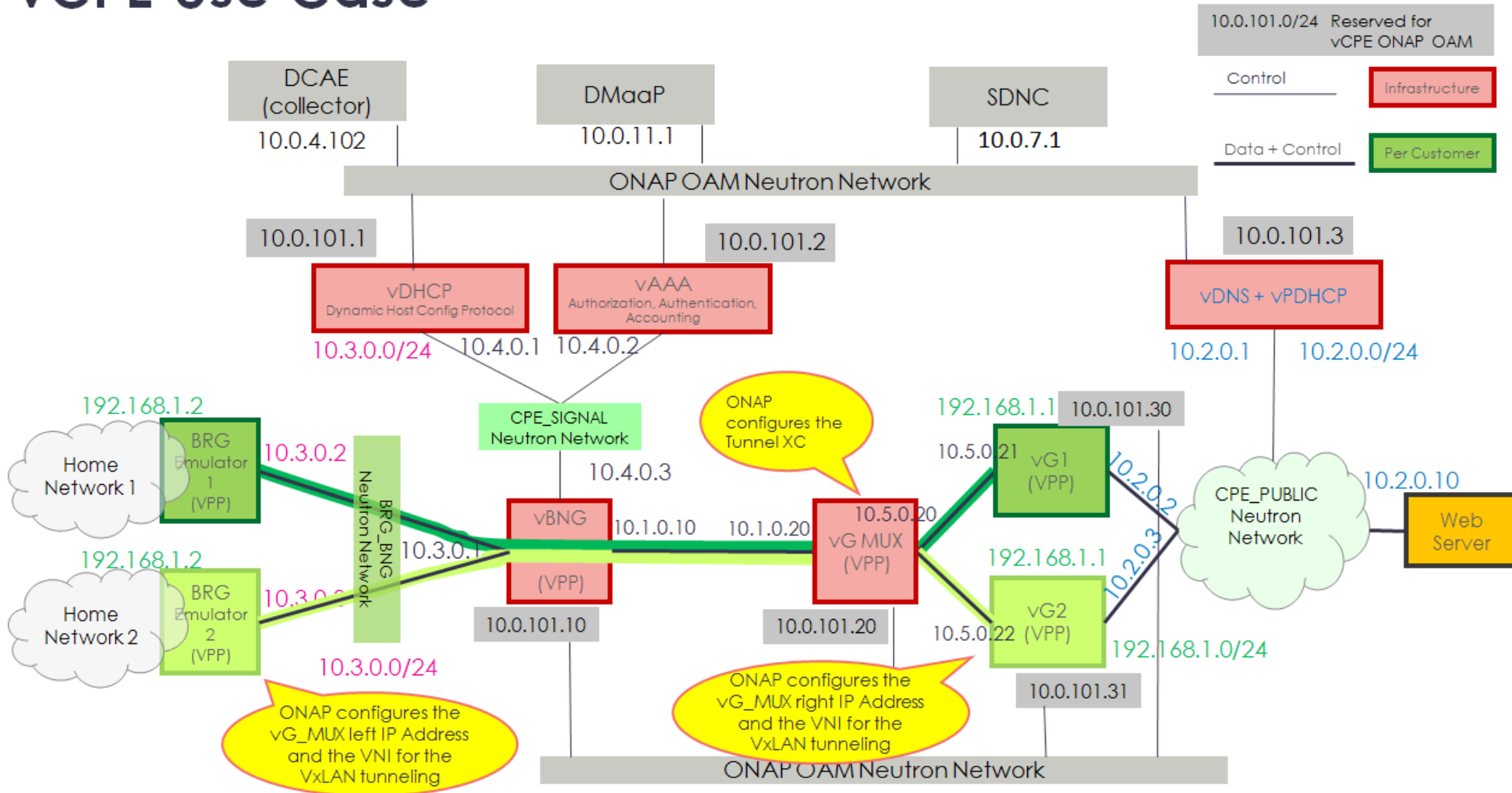


# 5 Services in vCPE Use Case



# Residential Broadband vCPE Use Case

## vCPE Use Case



# vCPE Service Tasks

#	Tasks	Category
1	Add new subcategory BRG to SDC Allotted Resource from Category Management page	Provision
2	Initialize demo data	Provision
3	VNFs onboarding	VNF onboarding service design and distribution
4	Create services for infra, vbng, vbrg and vgmux	
5	Download vbng and bgmux csars	
6	Create allotted resource for vbng and bgmux	
7	Create customer service	
8	Distribute all services	
9	Create availability zone in A&AI	Provision
10	Add customer SDN-ETHERNET-INTERNET in A&AI	
11	Run on SDNC cluster node <code>`ip route add 10.3.0.0/24 via 10.0.101.10 dev ens3`</code>	
12	Run on SDNC pod <code>`/opt/sdnc/bin/addIpAddresses.sh VGW 10.5.0 22 250`</code>	
13	Configure SNIRO	
14	Preload network and vmodules for services of infra, vbng, vbrg and vgmux and deploy those services	Service instantiation
15	Run heatbridge for vgmux	Provision
16	Run healthcheck-k8s.py to verify VNFs	Verification
17	Add custom flow trigger in SO	Provision
18	Preload and deploy customer service	Service instantiation
19	Push closed loop policy from pap pod	Provision
20	Service assurance	Verification

# vCPE Service Tasks with Automation

#	Tasks	Automation Method
1	Add new subcategory BRG to SDC Allotted Resource from Category Management page	SDC REST API
2	Initialize demo data	Robot script
3	VNFs onboarding	
4	Create services for infra, vbng, vbrg and vgmux	
5	Download vbng and bgmux csar packages	
6	Create allotted resource for vbng and bgmux	
7	Create customer service	
8	Distribute all services	
9	Create availability zone in A&AI	
10	Add customer SDN-ETHERNET-INTERNET in A&AI	A&AI REST API
11	Run on SDNC cluster node <code>`ip route add 10.3.0.0/24 via 10.0.101.10 dev ens3`</code>	Shell command
12	Run on SDNC pod <code>`/opt/sdnc/bin/addIpAddresses.sh VGW 10.5.0 22 250`</code>	Shell command
13	Configure SNIRO	Python script
14	Preload network and vmodule for services of infra, vbng, vbrg and vgmux and deploy them	
15	Create heatbridge for vgmux	
16	Run healthcheck-k8s.py to verify VNFs	
17	Add custom workflow in SO	
18	Deploy customer service	Shell command
19	Push closed loop policy from pap pod	
20	Service assurance	Shell commands

# vCPE Service Design with SDC

The screenshot shows the ONAP SDC v1.1.0 interface in the Composition view for service `vcpevc_infra_1222a`. The main workspace displays a diagram with three components: `cpe_signal`, `vcpevc_infra_1222a`, and `cpe_public`. A left-hand sidebar lists available network elements, including `ExtVL` and `Generic Neutron`. A right-hand sidebar lists various network-related artifacts such as `network_assignments`, `network_ecomp_naming`, and `network_role`. The interface includes navigation tabs for `HOME`, `SERVICE: vcpevc_infra_1222a`, and `Composition`.

The screenshot shows the ONAP SDC v1.1.0 interface in the Properties Assignment view for service `vcppear_tunnelconn_1222a`. The main workspace displays a table for property assignment with columns for `Property Name`, `Type`, `ES`, and `Value`. The table contains several rows, including `targeted...` (string), `providing...` (string), `max_inst...` (integer), and `min_insta...` (integer). A terminal window in the foreground shows the contents of a `tosca_definitions_version: tosca_simple_yaml_1_1` file, with red arrows pointing from specific lines in the terminal to the corresponding rows in the table. The terminal content includes:

```
invariantUUID: 74cb1cde-04ae-40a4-9cb9-357bb7367212
UUID: 382181dd-eb88-49d5-b430-4a069ad1f80c
name: vcpevc_vgmux_1222a
description: VGMUX
type: Service
category: Network L1-3
serviceType: ''
serviceRole: ''
serviceEcompNaming: true
ecompGeneratedNaming: true
namingPolicy: ''
imports:
- nodes:
  file: nodes.yml
- datatypes:
  file: data.yml
- capabilities:
  file: capabilities.yml
- relationships:
  file: relationships.yml
groups:
  file: groups.yml
policies:
  file: policies.yml
service-vcpevc_vgmux_1222a-interface:
  file: service-VcpevcVgmux1222a-template-interface.yml
resource-vcpevspvgmux:
  file: resource-Vcpevspvgmux-template.yml
resource-vcpevspvgmux-interface:
  file: resource-Vcpevspvgmux-template-interface.yml
topology_template:
  node_templates:
    vcpevspvgmux 0:
      type: org.openecomp.resource.vf.Vcpevspvgmux
      metadata:
        invariantUUID: 8a169dc2-c0f7-48df-b5a0-a28b1ded2435
Type :quit<Enter> to exit Vim 1,1 Top
```



# Service Design Automation with SDC BE and FE API

The screenshot displays the ONAP SDC BE and FE API interface. The main window shows the 'Properties Assignment' page for a service named 'TunnelXConn\_2019-01-02'. The page is in 'IN DESIGN CHECK OUT' state. A table lists various properties with their names, types, and values. The 'role' property is highlighted.

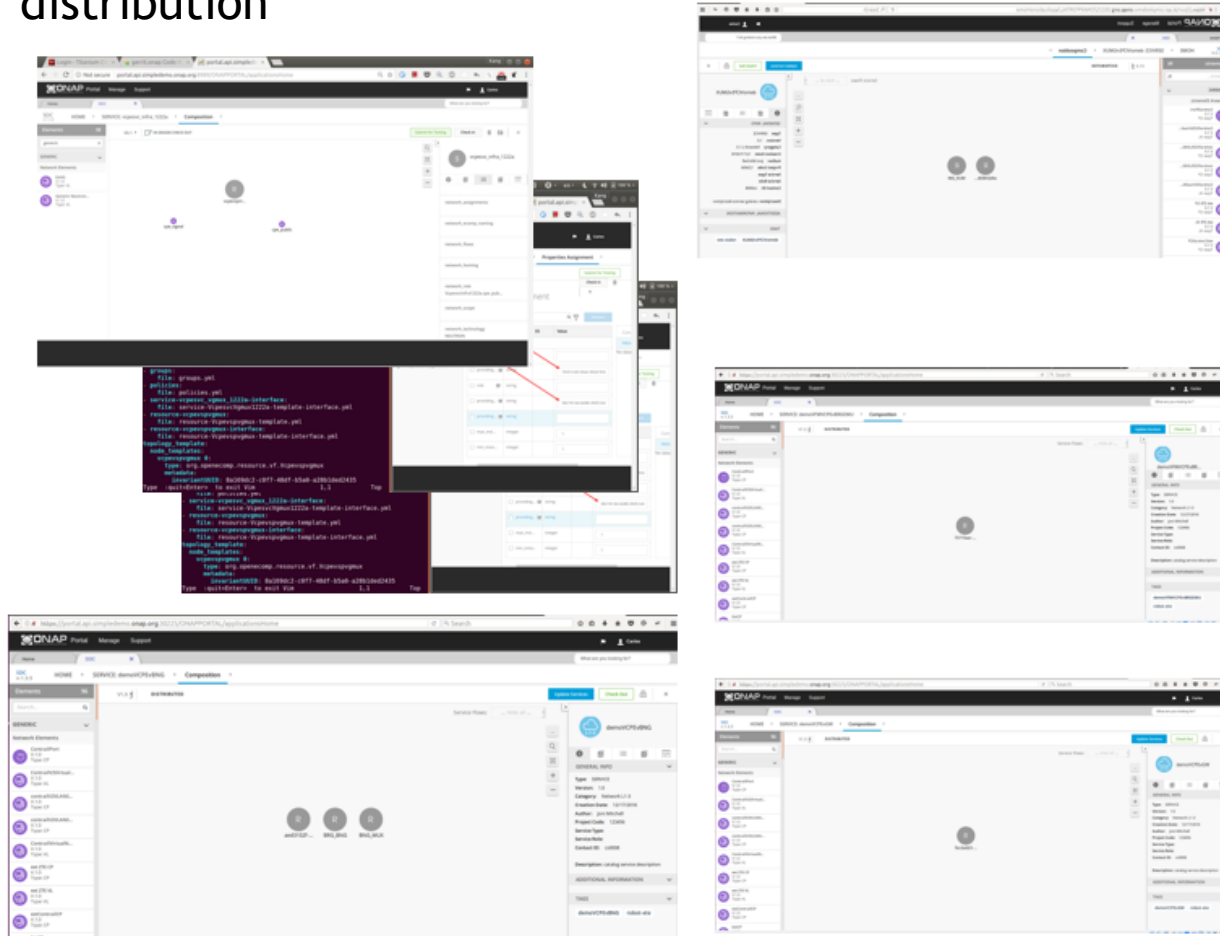
Property Name	Type	ES	Value
target_network_role	string		
providing_service_invariant_uuid	string		bf358370-382b-45cc-8c31-0210f2dda736
role	string		
providing_service_uuid	string		0424cc1c-ea72-4139-8854-4a02ad86538a
providing_service_name	string		org.openecomp.service.Demovcpevgmux
max_instances	integer		1
min_instances	integer		1

The right-hand side of the interface shows the Network Inspector, displaying the request details for the FE API. The request is a POST to the URL `http://portal.api.simpldemo.onap.org:30206/sdc1/feProxy/rest/v1/catalog/resources/f9f220d8-5e28-405c-9745-282ae4dc9749/resourceInstance/f9f220d8-5e28-405c-9745-282ae4dc9749.1aa530c3-810e-4124-9d87-68de064ccad.allottedresource0/properties`. The response status is 200 OK. The response headers include `Content-Type: application/json` and `Server: Jetty(9.4.12.v20180830)`. The request payload is a JSON object with the following structure:

```
{
  "defaultValue": null,
  "description": "The depending service invariant uuid in order to map the allotted resource to the specific service version",
  "name": "providing_service_invariant_uuid",
  "parentUniqueId": null,
  "password": false,
  "required": true
}
```

# CI Automation – Before and After

~3 hours for license, SWP onboarding, service creation, resource onboarding and distribution



3 seconds of typing  
(30 minutes of sipping coffee while robot does the work)

```
KEYWORD demo_preload. Load Models ${customer_name}
Documentation: Use openECOMP to Orchestrate a service.
Start / End / Elapsed: 20181216 17:47:05.836 / 20181216 18:33:50.898 / 00:46:45.062
KEYWORD BuiltIn.Set Test Variable ${CUSTOMER_NAME}, ${customer_name}
KEYWORD BuiltIn.Log To Console ${\n}Distributing vFWCL
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vFWCL, ${DEMO_PREFIX}vFWCL
KEYWORD BuiltIn.Log To Console Distributing vLB
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vLB, ${DEMO_PREFIX}vLB
KEYWORD BuiltIn.Log To Console Distributing vCPEInfra
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vCPEInfra, ${DEMO_PREFIX}vCPEInfra
KEYWORD BuiltIn.Log To Console Distributing vCPEvBNG
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vCPEvBNG, ${DEMO_PREFIX}vCPEvBNG
KEYWORD BuiltIn.Log To Console Distributing vCPEvBRGEMU
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vCPEvBRGEMU, ${DEMO_PREFIX}vCPEvBRGEMU
KEYWORD BuiltIn.Log To Console Distributing vCPEvGMUX
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vCPEvGMUX, ${DEMO_PREFIX}vCPEvGMUX
KEYWORD BuiltIn.Log To Console Distributing vCPEvGW (this is not vCPEResCust service)
KEYWORD BuiltIn.Run Keyword And Ignore Error Distribute Model, vCPEvGW, ${DEMO_PREFIX}vCPEvGW
17:47:05.83 TRACE Arguments: [ ${customer_name}='Demonstration' ]
6
```

# REST API based service/resource onboarding in testsuite

- Add ASDC Catalog Service
- :FOR \${zip} IN @\${model\_zip\_path} - Add VM resources
  - Setup ASDC Catalog Resource
  - Get ASDC Catalog Resource
  - Add ASDC Resource Instance
- :FOR \${network} in @\${networklist} - Add VM external networks
  - Get ASDC Catalog Resource}
  - Add ASDC Resource Instance (including X/Y parameters for the palette)
  - Setup SDC Catalog Resource GenericNeutronNet (this can be replaced with other Virtual Link Types)
- Checkin ASDC Catalog Service
- Request Certify ASDC Catalog Service
- Start Certify ASDC Catalog Service
- Certify ASDC Catalog Service
- Approve ASDC Catalog
- Distribute ASDC Catalog
- Download CSAR

**Model agnostic - add service to service\_mappings.py for service name to zip file directory location**

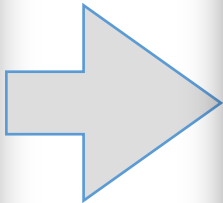
**Model dependent  
For properties/inputs**

**Model agnostic**

← Use by vCPE.py to populate pre-load model data

# Automate Preload by Using JSON Template

```
preload_templates — root@dev-robot-robot-6df6cc5768-lfhzx: /tmp/csar/service-Demovcpevgmux/Definitions — vi template.network.json...
1 {
2   "VNF-API:input": {
3     "VNF-API:request-information": {
4       "VNF-API:request-id": "robot0012",
5       "VNF-API:notification-url": "http://so.onap.org",
6       "VNF-API:order-number": "robot0012",
7       "VNF-API:request-sub-action": "SUPP",
8       "VNF-API:request-action": "PreloadNetworkRequest",
9       "VNF-API:source": "robot",
10      "VNF-API:order-version": "1.0"
11    },
12    "VNF-API:network-topology-information": {
13      "VNF-API:network-topology-identifier": {
14        "VNF-API:network-role": "${network_role}",
15        "VNF-API:network-technology": "neutron",
16        "VNF-API:service-type": "${service_type}",
17        "VNF-API:network-name": "${network_name}",
18        "VNF-API:network-type": "${network_type}"
19      },
20      "VNF-API:provider-network-information": {
21        "VNF-API:is-external-network": "true",
22        "VNF-API:physical-network-name": "${network_name}",
23        "VNF-API:is-provider-network": "true",
24        "VNF-API:is-shared-network": "true"
25      },
26      "VNF-API:subnets": [
27        {
28          "VNF-API:start-address": "${subnet_start_ip}",
29          "VNF-API:cidr-mask": "24",
30          "VNF-API:ip-version": "4",
31          "VNF-API:dhcp-enabled": "N",
32          "VNF-API:gateway-address": "${subnet_gateway}"
33        }
34      ]
35    },
36    "VNF-API:sdnc-request-header": {
37      "VNF-API:svc-action": "reserve",
38      "VNF-API:svc-notification-url": "http://so.onap.org",
39      "VNF-API:svc-request-id": "robot0012"
40    }
41  }
42 }
:set nu
```



```
json — root@oom-rancher: ~/integration/test/vcpe — ssh -i ~/.ssh/onap_dev root@10.12.6.194 — 94x43
1 {
2   "VNF-API:input": {
3     "VNF-API:sdnc-request-header": {
4       "VNF-API:svc-action": "reserve",
5       "VNF-API:svc-notification-url": "http://so.onap.org",
6       "VNF-API:svc-request-id": "robot0012"
7     },
8     "VNF-API:network-topology-information": {
9       "VNF-API:network-topology-identifier": {
10        "VNF-API:network-role": "cpe_public",
11        "VNF-API:network-name": "vcpe_net_cpe_public_201901051754",
12        "VNF-API:network-technology": "neutron",
13        "VNF-API:network-type": "Generic NeutronNet",
14        "VNF-API:service-type": "vCPE"
15      },
16      "VNF-API:subnets": [
17        {
18          "VNF-API:start-address": "10.2.0.2",
19          "VNF-API:gateway-address": "10.2.0.1",
20          "VNF-API:ip-version": "4",
21          "VNF-API:dhcp-enabled": "N",
22          "VNF-API:cidr-mask": "24"
23        }
24      ],
25      "VNF-API:provider-network-information": {
26        "VNF-API:physical-network-name": "vcpe_net_cpe_public_201901051754",
27        "VNF-API:is-external-network": "true",
28        "VNF-API:is-provider-network": "true",
29        "VNF-API:is-shared-network": "true"
30      }
31    },
32    "VNF-API:request-information": {
33      "VNF-API:source": "robot",
34      "VNF-API:notification-url": "http://so.onap.org",
35      "VNF-API:request-sub-action": "SUPP",
36      "VNF-API:order-number": "robot0012",
37      "VNF-API:request-id": "robot0012",
38      "VNF-API:order-version": "1.0",
39      "VNF-API:request-action": "PreloadNetworkRequest"
40    }
41  }
42 }
:set nu 1,1 Top
```

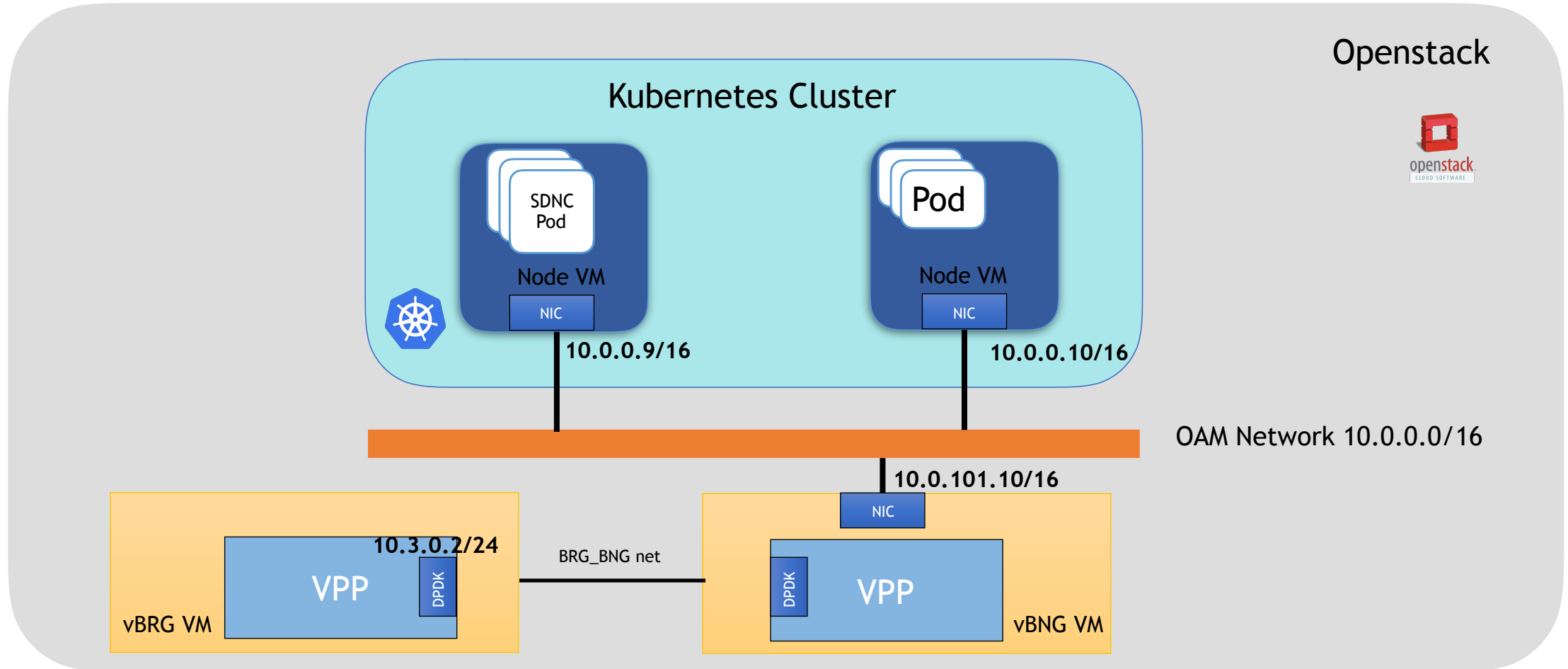
# Instantiate Infra Services

```
yang — root@oom-rancher: ~/integration/test/vcpe — ssh -i ~/.ssh/onap_dev root@10.12.6.194 — 103x10
root@oom-rancher:~/integration/test/vcpe# ./vcpe.py infra
-----
vcpe.py:          Brief info about this program
vcpe.py init:     Add customer service data to SDNC and SO DBs.
vcpe.py infra:   Deploy infrastructure, including DHCP, AAA, DNS, Web Server, vBNG, vGMUX, vBRG.
vcpe.py brg:     Deploy brg only (for testing after infra succeeds).
vcpe.py customer: Deploy customer service, including vGW and VxLANs
vcpe.py loop:    Test closed loop control
-----
Ready to deploy infrastructure? y/n: █
```

ID	Host	IP	Network	Service	Status	Created	Actions
zdcpe1cpe01brgemu01_201901052042	vbrg-casa-base-ubuntu-16-04	10.12.7.13	vcpe_net_brg_bng_201901052042	m1.medium vbrgemu_key_FZpC	Active	1 day	Create Snapshot
zdcpe1cpe01mux01_201901052042	vgmux-casa-base-ubuntu-16-04	10.12.5.100	vcpe_net_bng_mux_201901052042	m1.medium vgmux_key_uQBF	Active	1 day	Create Snapshot
zdcpe1cpe01bng01_201901052042	vbrg-casa-base-ubuntu-16-04	10.12.5.184	vcpe_net_bng_bng_201901052042	m1.medium vbrg_key_8Sz	Active	1 day	Create Snapshot
zdcpe1cpe01aa01_201901052042	ubuntu-16-04-cloud-amd64	10.12.6.86	vcpe_net_cpe_signal_201901052042	m1.medium vaaa_key_ShNr	Active	1 day	Create Snapshot
zdcpe1cpe01dhp01_201901052042	ubuntu-16-04-cloud-amd64	10.12.6.141	vcpe_net_cpe_signal_201901052042	m1.medium vaaa_key_ShNr	Active	1 day	Create Snapshot
zdcpe1cpe01web01_201901052042	ubuntu-16-04-cloud-amd64	10.12.5.133	vcpe_net_cpe_public_201901052042	m1.medium vaaa_key_ShNr	Active	1 day	Create Snapshot
zdcpe1cpe01dns01_201901052042	ubuntu-16-04-cloud-amd64	10.12.5.165	vcpe_net_cpe_public_201901052042	m1.medium vaaa_key_ShNr	Active	1 day	Create Snapshot

# Access ONAP Infrastructure via Kubernetes and Openstack API

## vCPE Automation with Kubernetes and Openstack APIs



# Access ONAP Infrastructure via Kubernetes and Openstack API

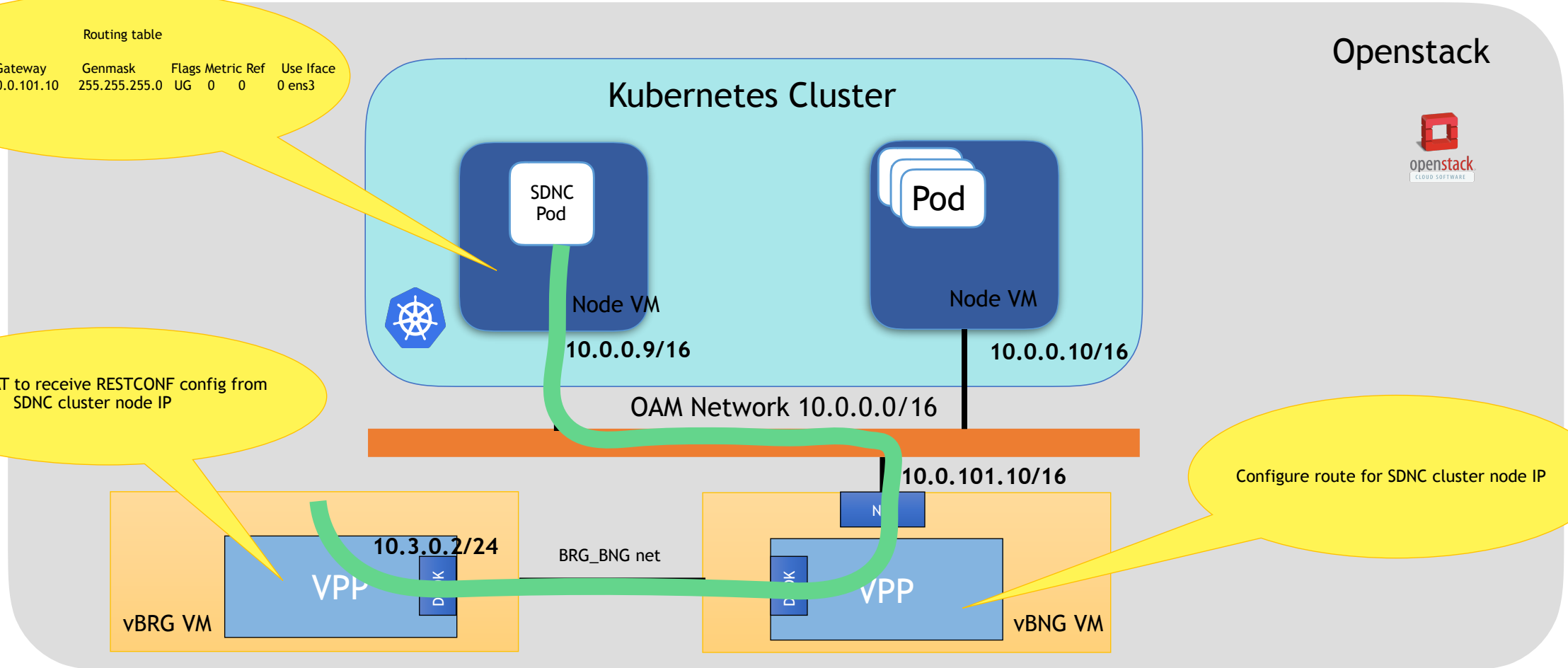
## ONAP Automation with Kubernetes and Openstack

Routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.3.0.0	10.0.101.10	255.255.255.0	UG	0	0		0 ens3

Set up SNAT to receive RESTCONF config from SDNC cluster node IP

Configure route for SDNC cluster node IP



Openstack



# Instantiate Customer Service

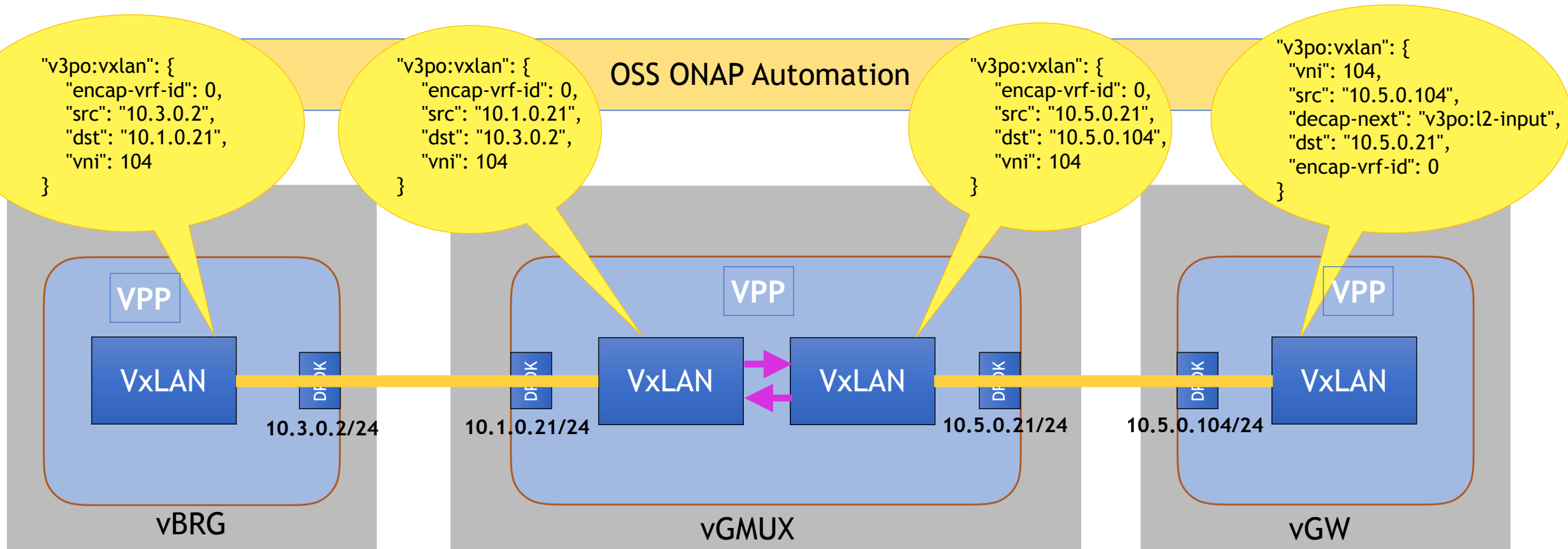
```
oam — root@zdcpe1cpe01brgemu01-201812011854: ~ — ssh -i ~/.ssh/onap_dev root@10.12.5.147 — 110x12
@oom-rancher:~/integration/test/vcpe# ./vcpe.py customer
-----
vcpe.py:          Brief info about this program
vcpe.py init:     Add customer service data to SDNC and SO DBs.
vcpe.py infra:   Deploy infrastructure, including DHCP, AAA, DNS, Web Server, vBNG, vGMUX, vBRG.
vcpe.py brg:     Deploy brg only (for testing after infra succeeds).
vcpe.py customer: Deploy customer service, including vGW and VxLANs
vcpe.py loop:    Test closed loop control
-----
Ready to deploy customer service? y/n: y
Enter the BRG MAC address: fa:16:3e:6e:e1:05
```

The screenshot shows the Titanium Cloud web interface. The left sidebar contains navigation options: Project, API Access, Compute, Overview, Instances (highlighted), Server Groups, Images, Key Pairs, Volumes, Network, Orchestration, and Identity. The main content area is titled 'Instances' and shows a table of instances. The table has the following columns: Instance Name, Image Name, IP Address, Flavor, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. One instance is listed with the following details:

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
zdcpe1cpe01gw01_201901061413	vgw-casa-base-ubuntu-16-04	10.5.0.104 oam_network_2No2 10.0.101.104 vcpe_net_cpe_public_201901052042 10.2.0.4	m1.medium	vgw_key_mU3I	Active	nova	None	Running	7 hours, 10 minutes	Create Snapshot



# Service Assurance



- Interface IPs are assigned correctly and VNIs aligned (healthcheck-k8s.py)
- Data plane verification works (ping and curl commands)
- Closed loop events are sent (vcpe.py)

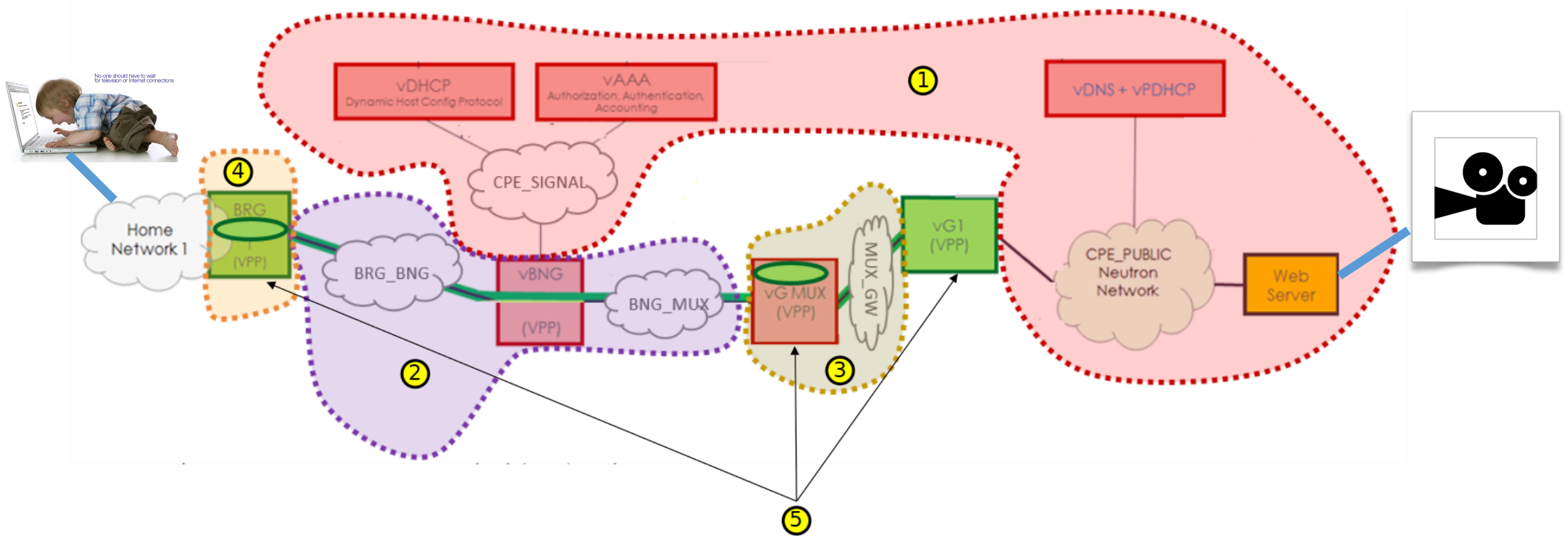
# Suggestion of Using vCPE Use Case to Test New Platform Features

- SO building blocks
- Controller Design Studio for post-instantiation configuration



**Post-instantiation**

# vCPE Demo Setup





# vCPE Demo

<http://vcpedemo.onap.org/bunny.mp4>

# Reference

1. [ONAP E2E Automation](#) - Eric Debeau, etc
2. [vCPE VPP VNF Deep Dive](#) - Eric Multanen, Intel
3. [vCPE Use Case - Customer Service Instantiation](#) - Gil Bullard
4. [Residential Broadband vCPE Drafts for discussion](#)