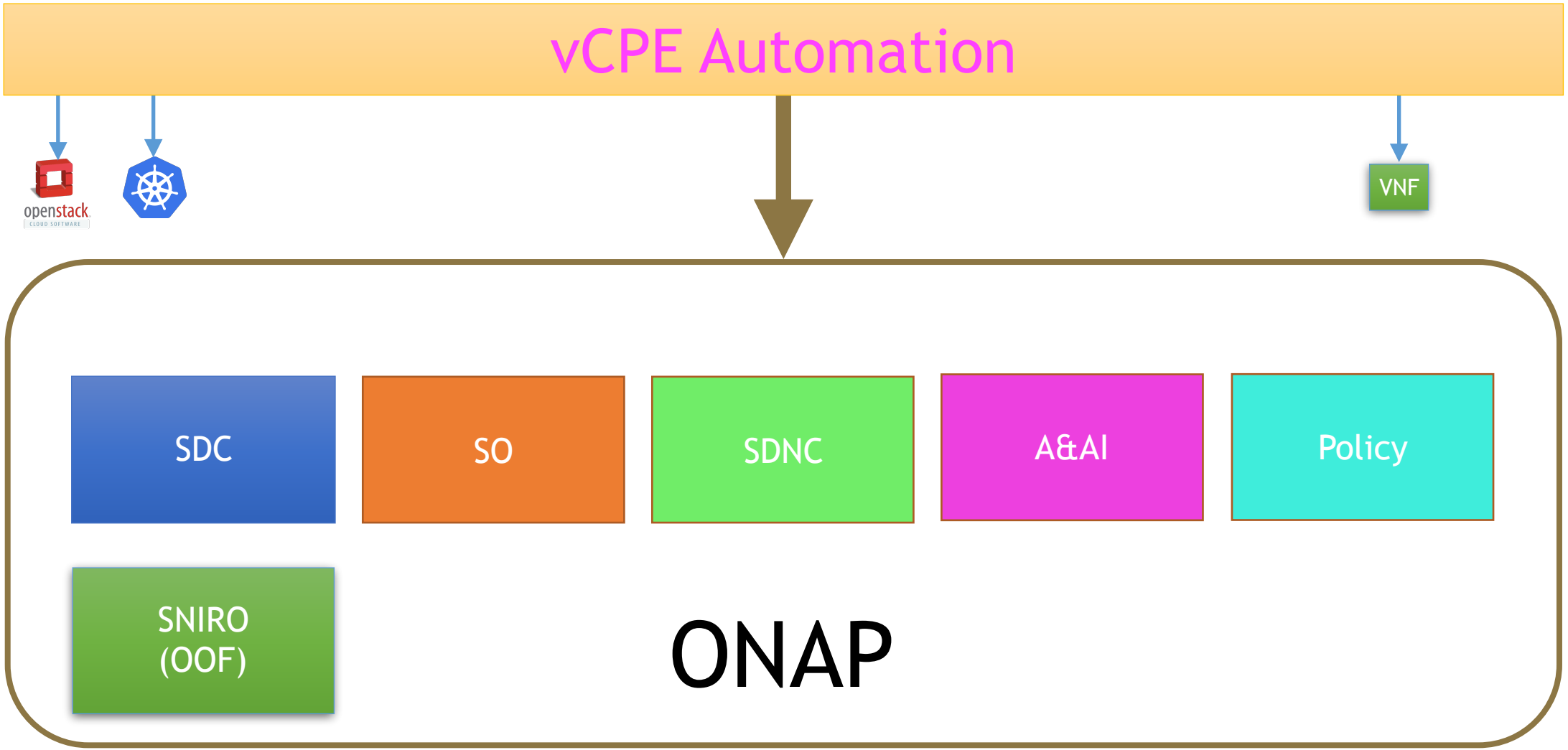
Use Case Automation
(An example of OSS integration)

Integration Team

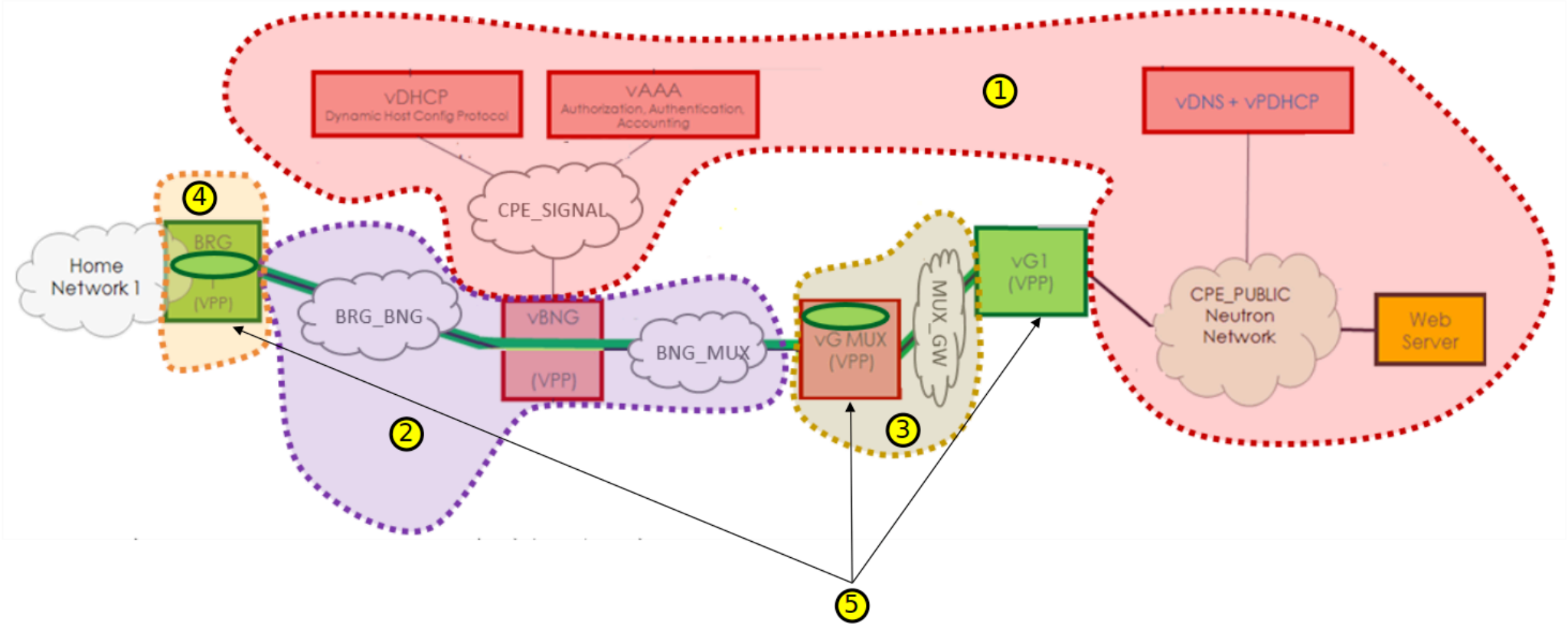
TSC Must-Have for Dublin

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

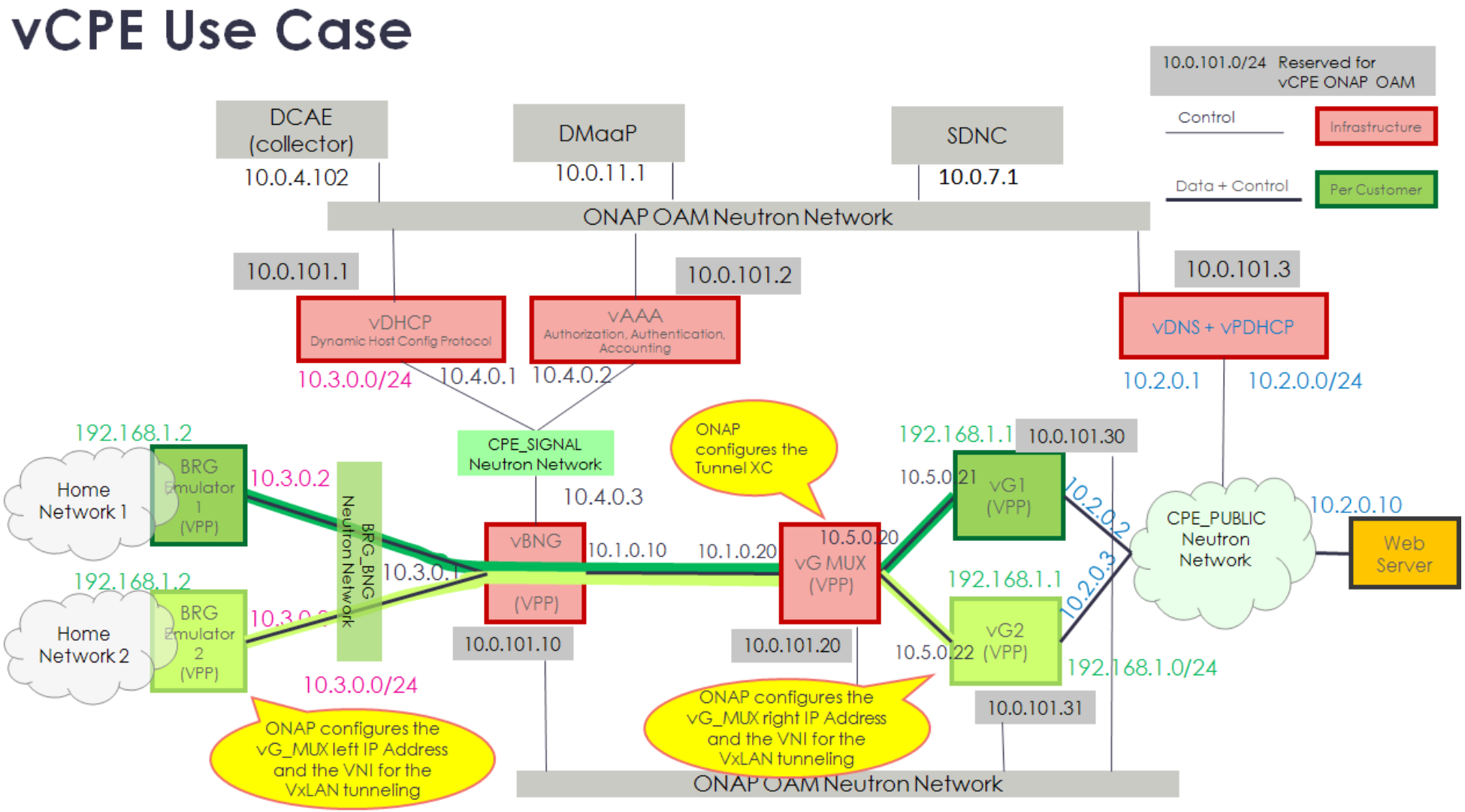
vCPE Use Case Automation



5 Services in vCPE Use Case



Residential Broadband vCPE Use Case



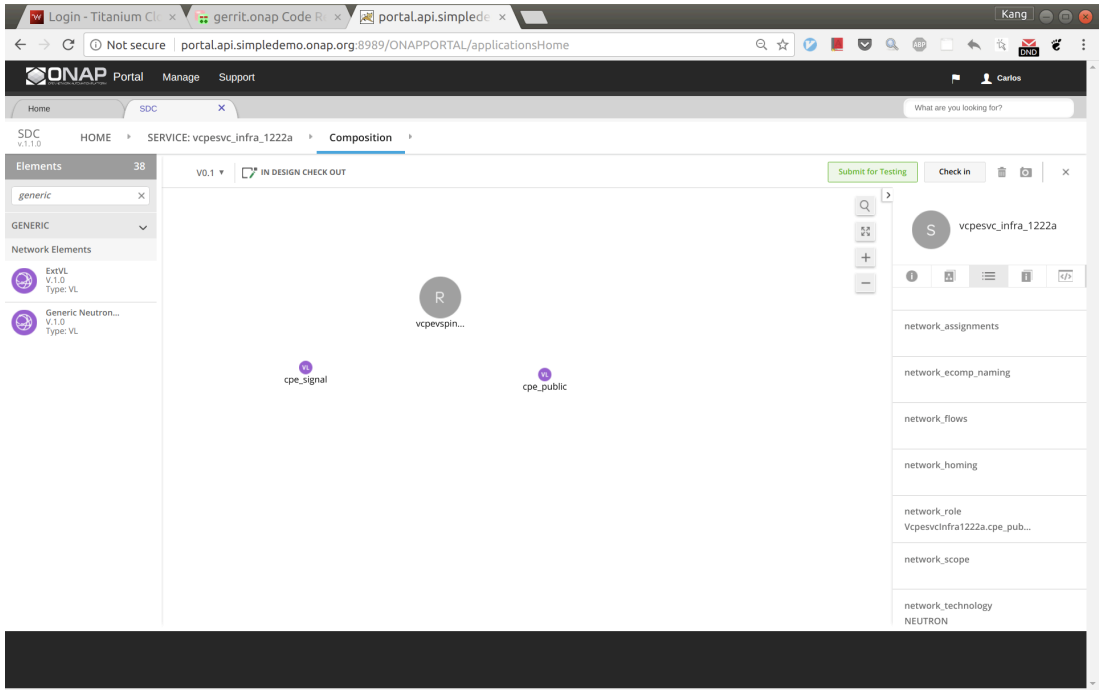
vCPE Service Tasks

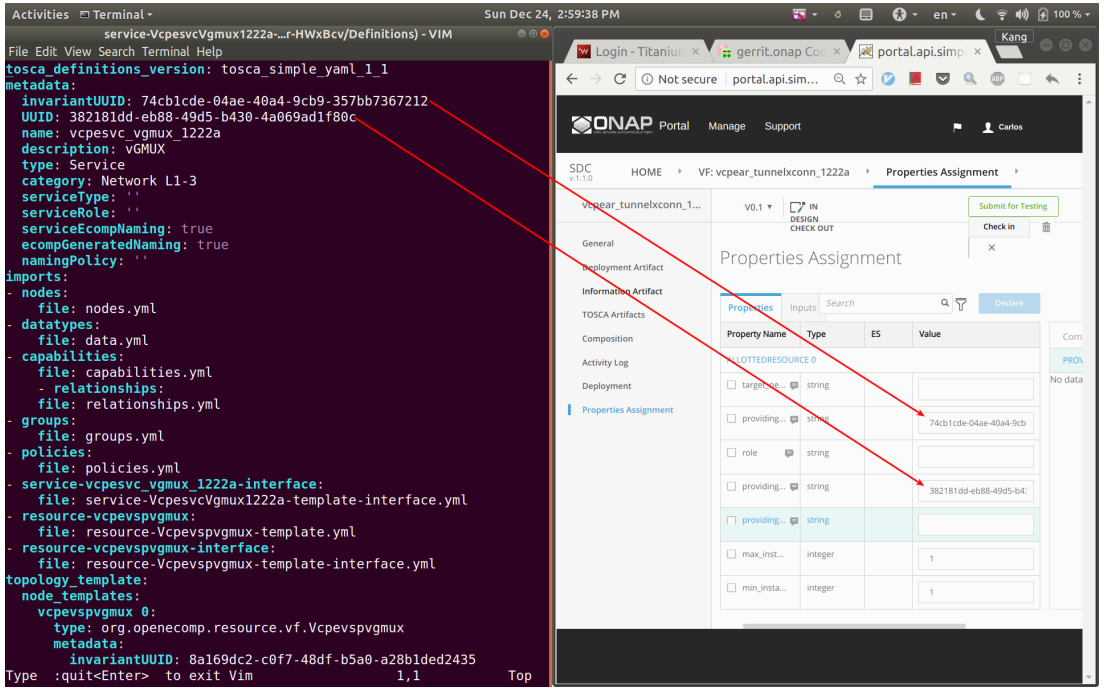
#	Tasks	Category
1	Add new subcategory BRG to SDC Allotted Resource from Category Management page	Provision
2	Initialize demo data	Provision
3	VNFs onboarding	VNF onboarding service design and distribution
4	Create services for infra, vbng, vbrg and vgmux	
5	Download vbng and bgmux csars	
6	Create allotted 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 `ip route add 10.3.0.0/24 via 10.0.101.10 dev ens3`	
12	Run on SDNC pod `/opt/sdnc/bin/addIpAddresses.sh VGW 10.5.0 22 250`	
13	Configure SNIRO	
14	Preload network and vfmodules 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	Verfification
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

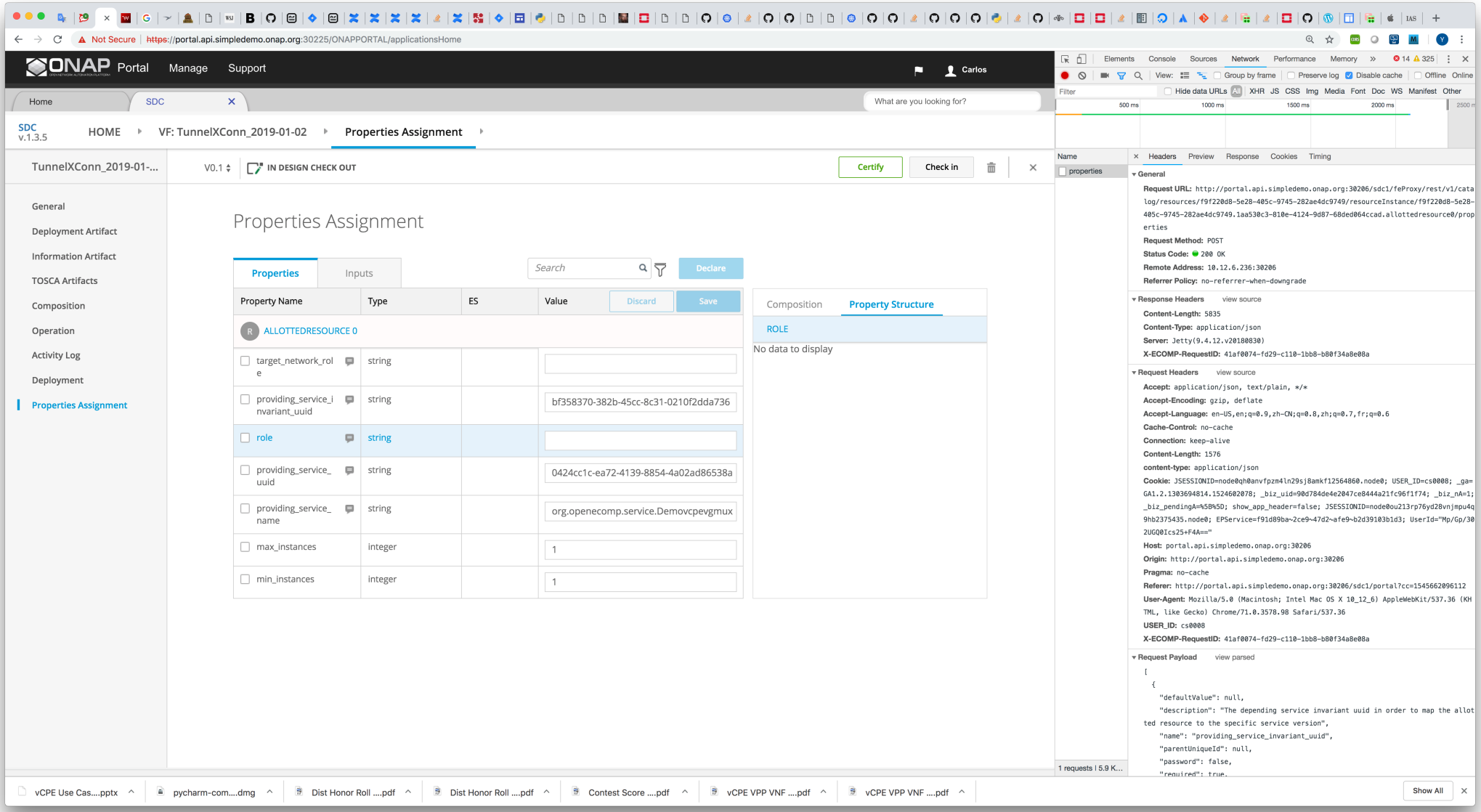
#	Tasks	Automation Method
1	Add new subcategory BRG to SDC Allotted Resource from Category Management page	SDC REST API
2	Initialize demo data	Robot script
3	VNFs onboarding	
4	Create services for infra, vbng, vbrg and vgmux	
5	Download vbng and bgmux csar packages	
6	Create allotted resource for vbng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

vCPE Service Design with SDC



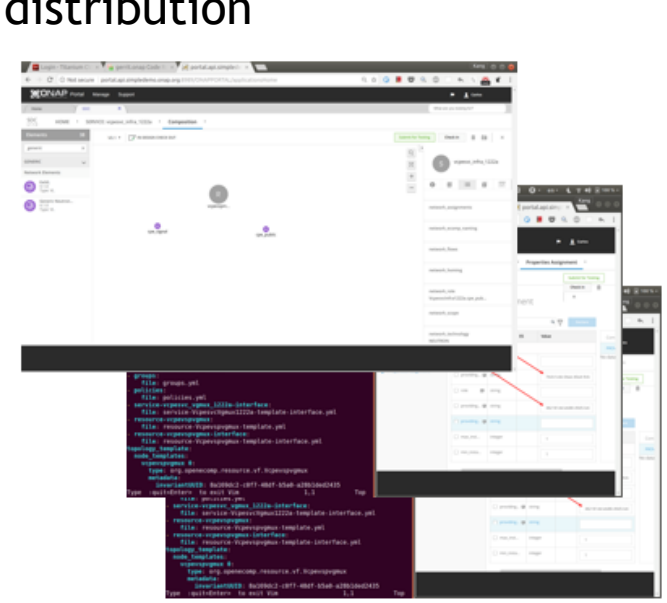


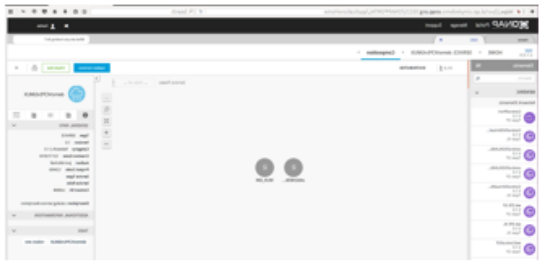
Service Design Automation with SDC BE and FE API

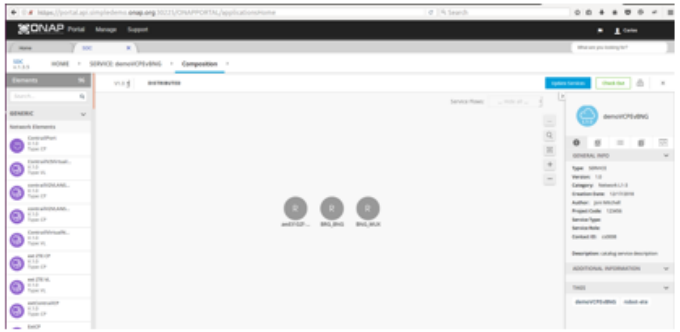


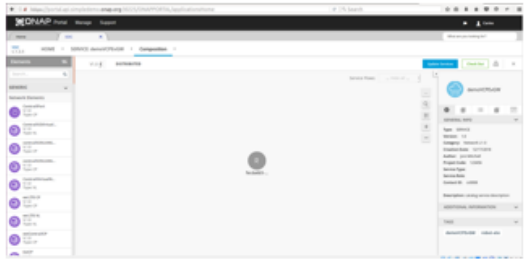
CI Automation – Before and After

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









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

```
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 ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vFWCL, ${DEMO_PREFIX}VFWCL
KEYWORD BuiltIn. Log To Console Distibuting vLB
KEYWORD ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vLB, ${DEMO_PREFIX}VLB
KEYWORD BuiltIn. Log To Console Distibuting vCPEInfra
KEYWORD ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vCPEInfra, ${DEMO_PREFIX}VCPEInfra
KEYWORD BuiltIn. Log To Console Distibuting vCPEvBNG
KEYWORD ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vCPEvBNG, ${DEMO_PREFIX}VCPEvBNG
KEYWORD BuiltIn. Log To Console Distibuting vCPEvBRGEMU
KEYWORD ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vCPEvBRGEMU, ${DEMO_PREFIX}VCPEvBRGEMU
KEYWORD BuiltIn. Log To Console Distibuting vCPEvGMUX
KEYWORD ${status}, ${value} = BuiltIn. Run Keyword And Ignore Error Distribute Model, vCPEvGMUX, ${DEMO_PREFIX}VCPEvGMUX
KEYWORD BuiltIn. Log To Console Distibuting vCPEvGW (this is not vCPEResCust service)
KEYWORD ${status}, ${value} = 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...

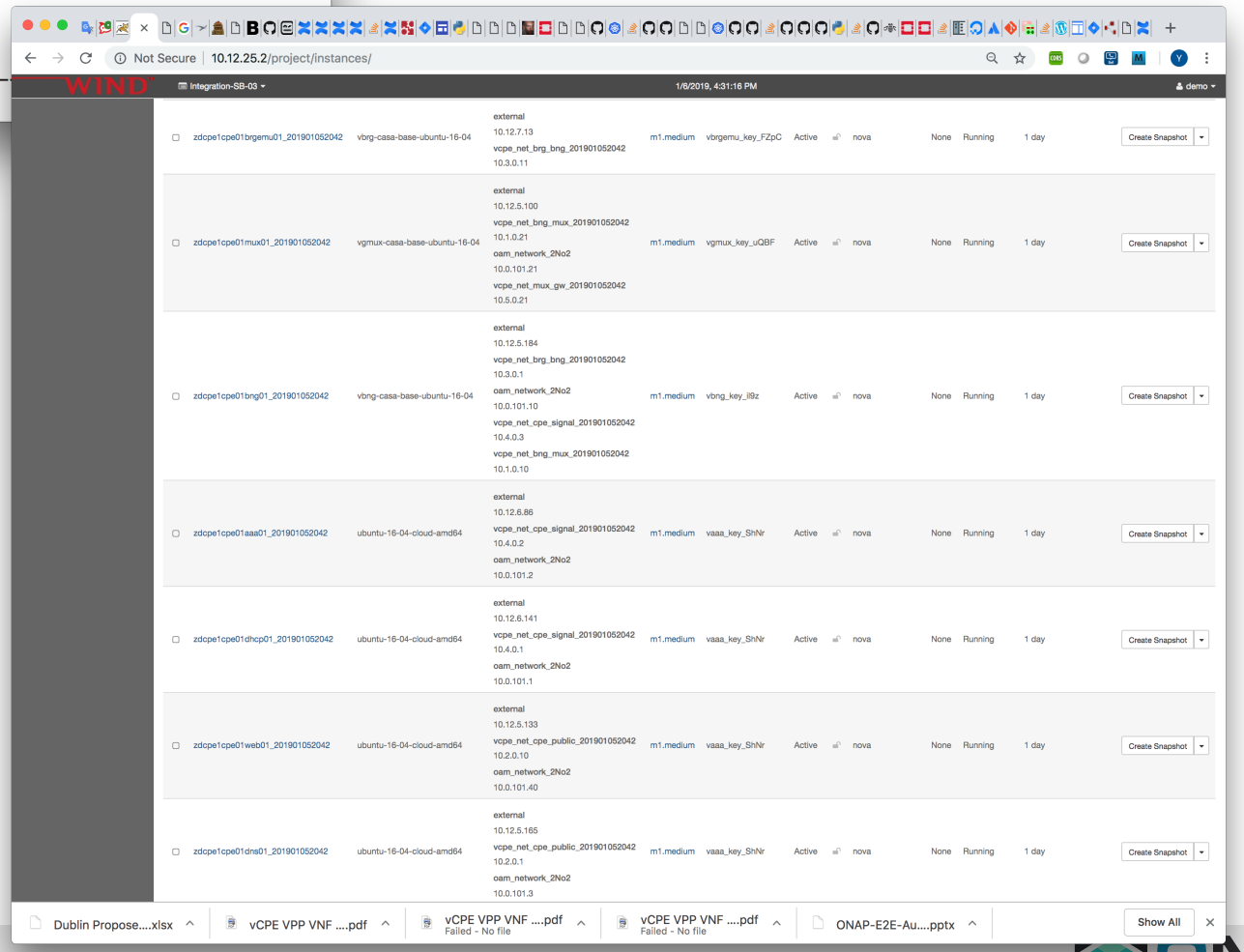
```
{
  "VNF-API:input": {
    "VNF-API:request-information": {
      "VNF-API:request-id": "robot0012",
      "VNF-API:notification-url": "http://so.onap.org",
      "VNF-API:order-number": "robot0012",
      "VNF-API:request-sub-action": "SUPP",
      "VNF-API:request-action": "PreloadNetworkRequest",
      "VNF-API:source": "robot",
      "VNF-API:order-version": "1.0"
    },
    "VNF-API:network-topology-information": {
      "VNF-API:network-topology-identifier": {
        "VNF-API:network-role": "${network_role}",
        "VNF-API:network-technology": "neutron",
        "VNF-API:service-type": "${service_type}",
        "VNF-API:network-name": "${network_name}",
        "VNF-API:network-type": "${network_type}"
      },
      "VNF-API:provider-network-information": {
        "VNF-API:is-external-network": "true",
        "VNF-API:physical-network-name": "${network_name}",
        "VNF-API:is-provider-network": "true",
        "VNF-API:is-shared-network": "true"
      },
      "VNF-API:subnets": [
        {
          "VNF-API:start-address": "${subnet_start_ip}",
          "VNF-API:cidr-mask": "24",
          "VNF-API:ip-version": "4",
          "VNF-API:dhcp-enabled": "N",
          "VNF-API:gateway-address": "${subnet_gateway}"
        }
        ]
    },
    "VNF-API:sdnc-request-header": {
      "VNF-API:svc-action": "reserve",
      "VNF-API:svc-notification-url": "http://so.onap.org",
      "VNF-API:svc-request-id": "robot0012"
    }
  }
}
:set nu
```

json — root@oom-rancher: ~/integration/test/vcpe — ssh -i ~/.ssh/onap_dev root@10.12.6.194 — 94×43

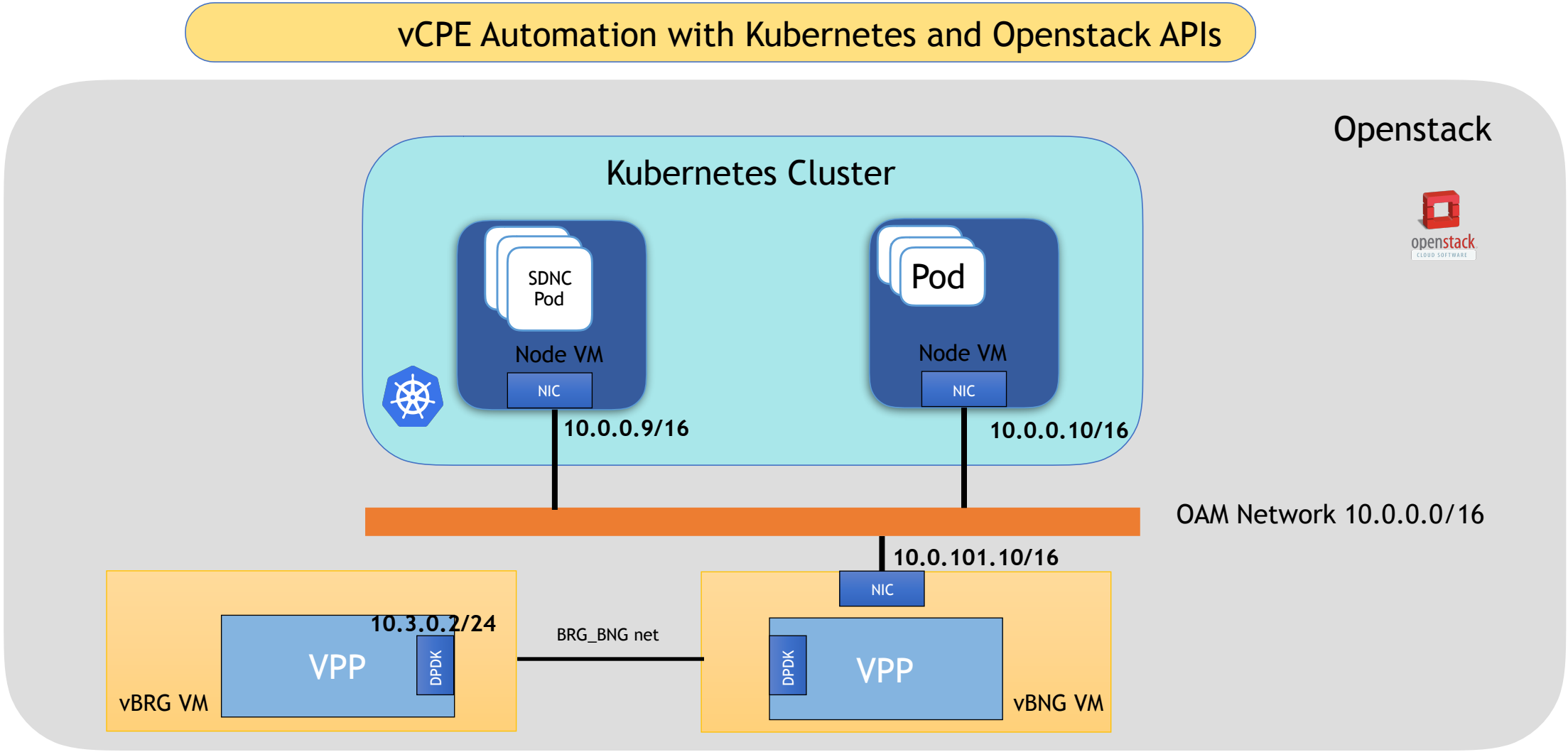
```
{
    "VNF-API:input": {
        "VNF-API:sdnc-request-header": {
            "VNF-API:svc-action": "reserve",
            "VNF-API:svc-notification-url": "http://so.onap.org",
            "VNF-API:svc-request-id": "robot0012"
        },
        "VNF-API:network-topology-information": {
            "VNF-API:network-topology-identifier": {
                "VNF-API:network-role": "cpe_public",
                "VNF-API:network-name": "vcpe_net_cpe_public_201901051754",
                "VNF-API:network-technology": "neutron",
                "VNF-API:network-type": "Generic NeutronNet",
                "VNF-API:service-type": "vCPE"
            },
            "VNF-API:subnets": [
                {
                    "VNF-API:start-address": "10.2.0.2",
                    "VNF-API:gateway-address": "10.2.0.1",
                    "VNF-API:ip-version": "4",
                    "VNF-API:dhcp-enabled": "N",
                    "VNF-API:cidr-mask": "24"
                }
            ],
            "VNF-API:provider-network-information": {
                "VNF-API:physical-network-name": "vcpe_net_cpe_public_201901051754",
                "VNF-API:is-external-network": "true",
                "VNF-API:is-provider-network": "true",
                "VNF-API:is-shared-network": "true"
            }
        },
        "VNF-API:request-information": {
            "VNF-API:source": "robot",
            "VNF-API:notification-url": "http://so.onap.org",
            "VNF-API:request-sub-action": "SUPP",
            "VNF-API:order-number": "robot0012",
            "VNF-API:request-id": "robot0012",
            "VNF-API:order-version": "1.0",
            "VNF-API:request-action": "PreloadNetworkRequest"
        }
    }
}
:set nu                                                    1,1           Top
```

Instantiate Infra Services

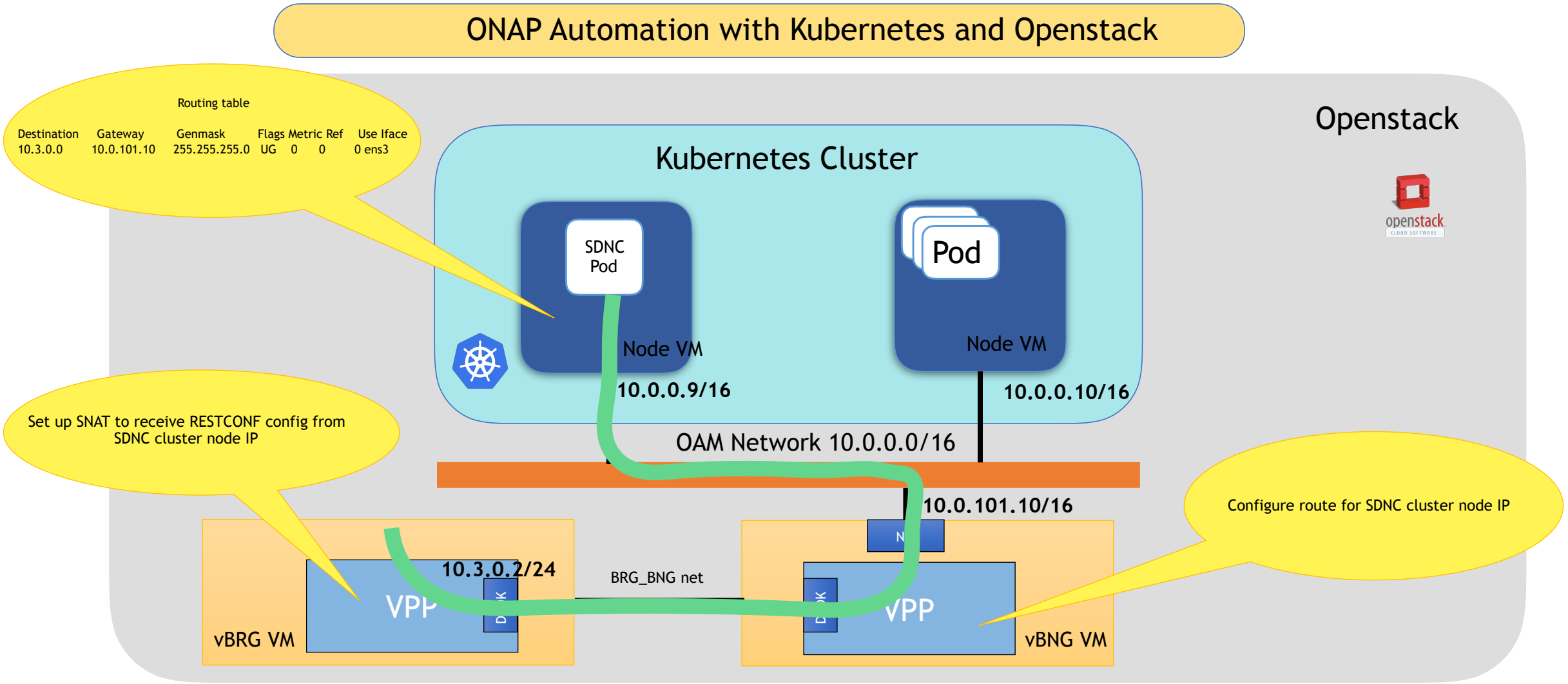
```
yang — root@oom-rancher: ~/integration/test/vcpe — ssh -i ~/.ssh/onap_dev root@10.12.6.194 — 103×10
root@oom-rancher:~/integration/test/vcpe# ./vcpe.py infra
------------------------------------------------------------------------------------------------------------
 vcpe.py:              Brief info 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:
```



Access ONAP Infrastructure via Kubernetes and Openstack API

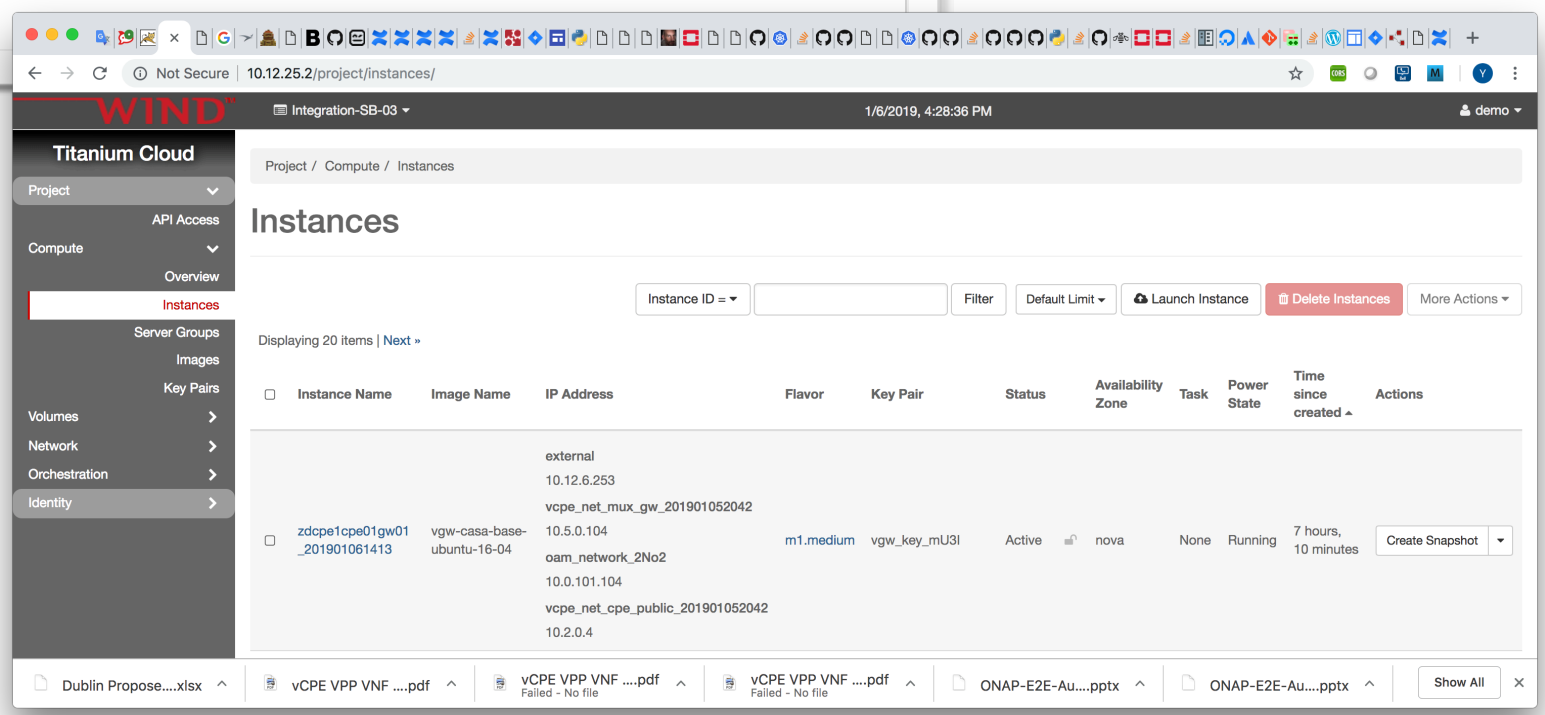


Access ONAP Infrastructure via Kubernetes and Openstack API

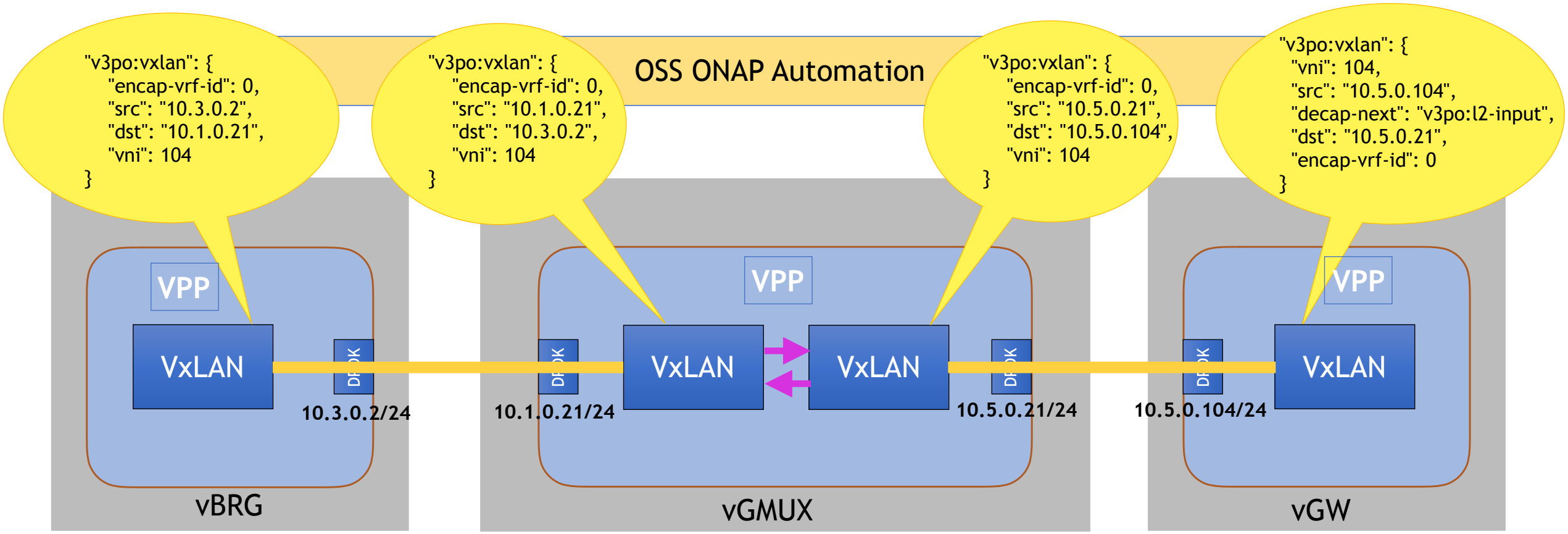


Instantiate Customer Service

```
oam — root@zdcpe1cpe01brgemu01-201812011854: ~ — ssh -i ~/.ssh/onap_dev root@10.12.5.147 — 110×12
@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
```



Service Assurance

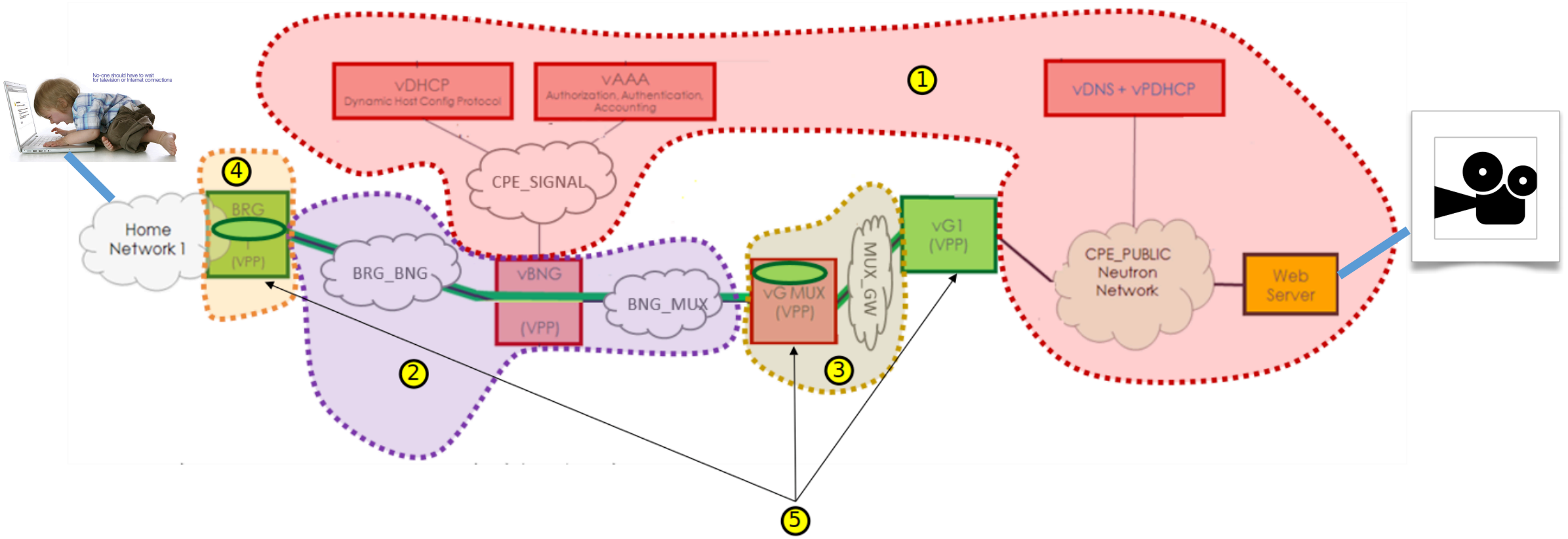


- Interface IPs are assigned correctly and VNIs aligned (healthcheck-k8s.py)
- Data plane verification works (ping and curl commands)
- Closed loop events are sent (vcpe.py)

Suggestion of Using vCPE Use Case to Test New Platform Features

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

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