Customizing CloudStack Network with VNF Appliances

The introduction of VNF framework
in Apache CloudStack 4.23

Wei Zhou

- Joined Apache CloudStack community in 2012
- Apache CloudStack committer since 2013.05
- Apache CloudStack PMC member since 2017.03

- Senior Software Architect @ Shapeblue
- Member of Kubernetes org

- Email: weizhou@apache.org
- Github: @weizhouapache

Content

- VNF integration recapping
- Why a VNF framework
- Introduction of the VNF framework
- Key Considerations and Samples
- Conclusion & Discussion

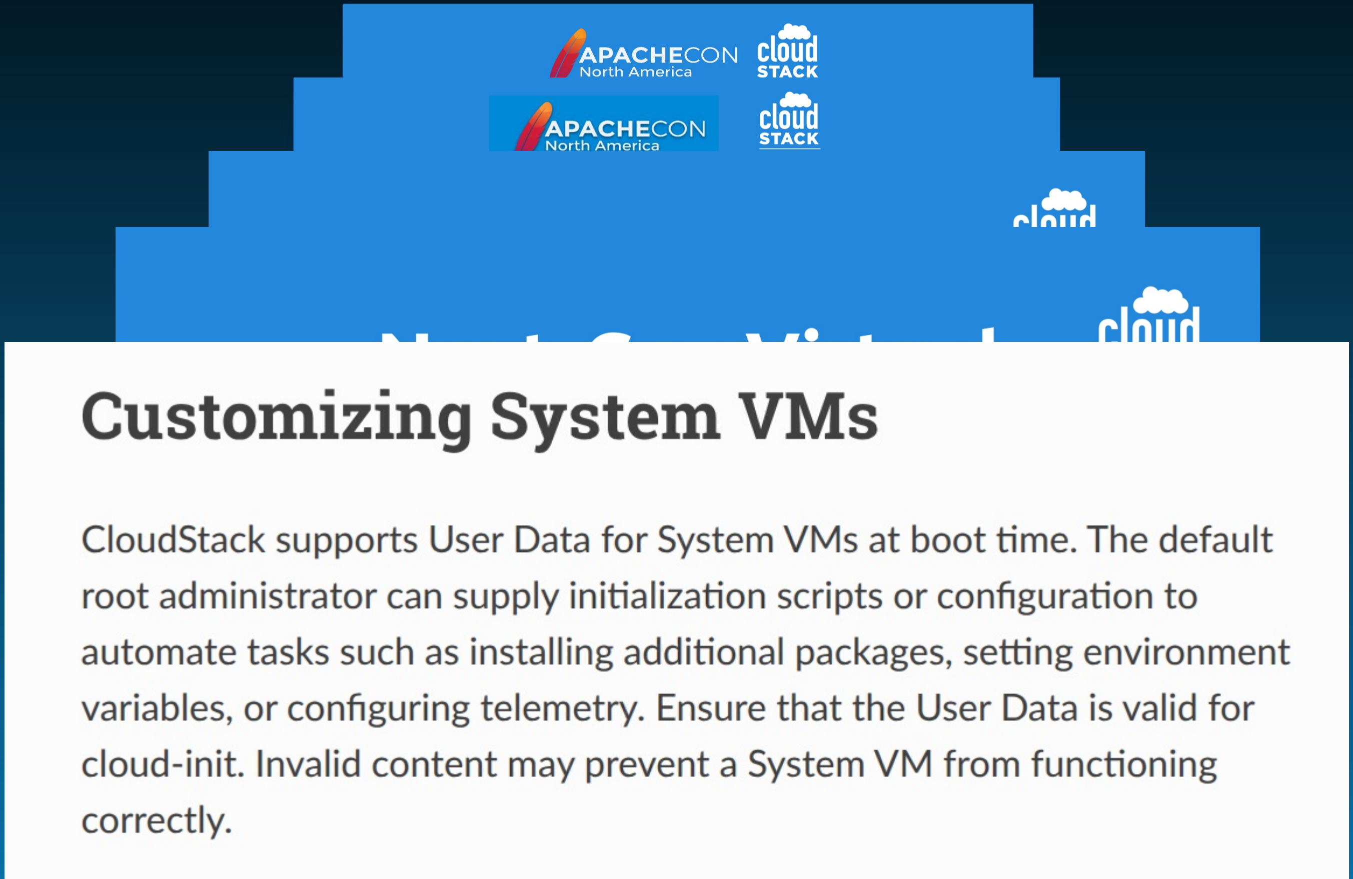
What is VNF

- VNF: Virtualized Network Functions
  - **Virtualized**: appliance running in VMs
  - **Network functions**: provide network services
    - Firewall, Source NAT, Destination NAT, LB, VPN, Security ……
  - Source: vendors, cloud providers, marketplace, BYOV

CloudStack Solution: Virtual Router

- It is Amazing !
- It is very powerful: provides features
  - SourceNat, StaticNat; Firewall; PortForwarding, Load balancing; S2S/RSA VPN; DHCP, DNS, Userdata, Password server; IPv6, etc
  - **Dynamic Routing (since ACS 4.20)**
- Limitations: Tightly coupled with Apache CloudStack
  - Specific software solutions (iptables, haproxy, etc)
  - Lack some features (OpenVPN, Security solutions, etc)

Past discussions



Customizing System VMs

CloudStack supports User Data for System VMs at boot time. The default root administrator can supply initialization scripts or configuration to automate tasks such as installing additional packages, setting environment variables, or configuring telemetry. Ensure that the User Data is valid for cloud-init. Invalid content may prevent a System VM from functioning correctly.

cloudstack
open source cloud computing

Network Function Virtualization with Apache CloudStack – a tale of

cloudstack
open source cloud computing

Ne
ACS
VNI

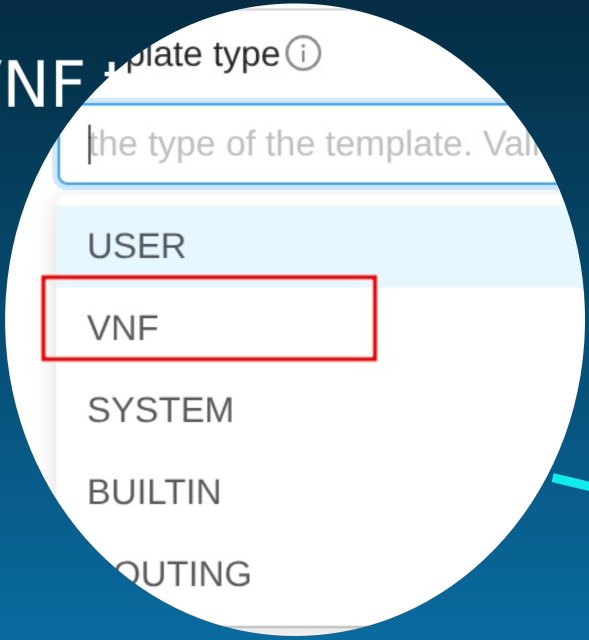
VNF Integration and Support in CloudStack 4.19

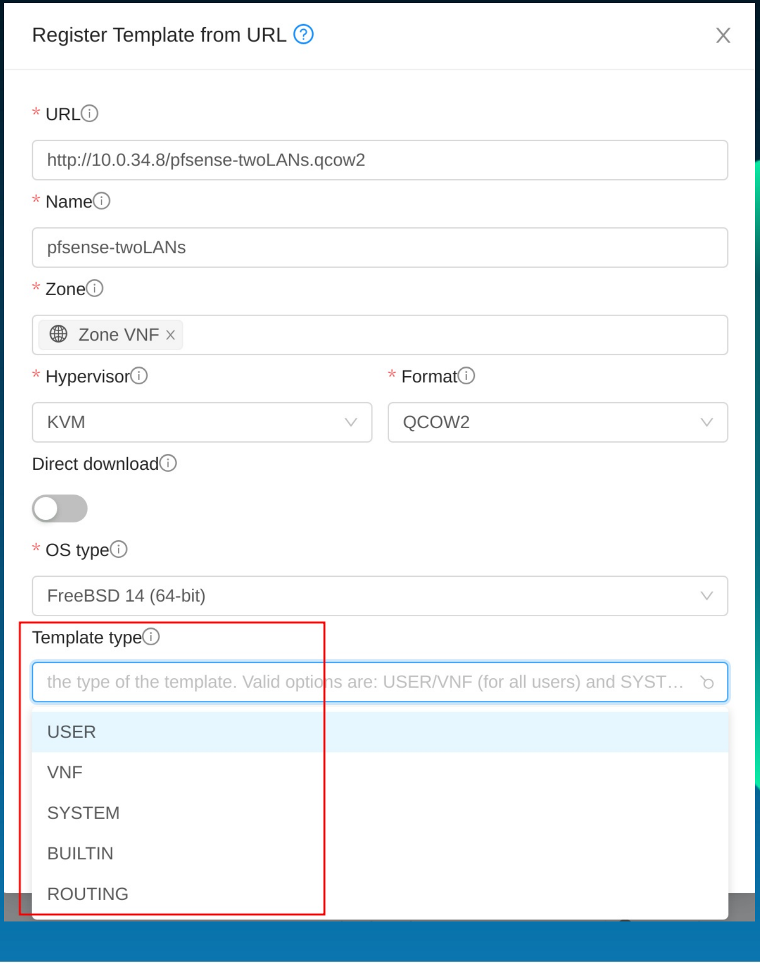
CloudStack Collaboration Conference, 23 - 24 November 2023

VNF integration recapping

- Supported since ACS 4.19

- VNF




Register Template from URL ?

* URL ⓘ
http://10.0.34.8/pfsense-twoLANs.qcow2

* Name ⓘ
pfsense-twoLANs

* Zone ⓘ
Zone VNF ×

* Hypervisor ⓘ
KVM

* Format ⓘ
QCOW2

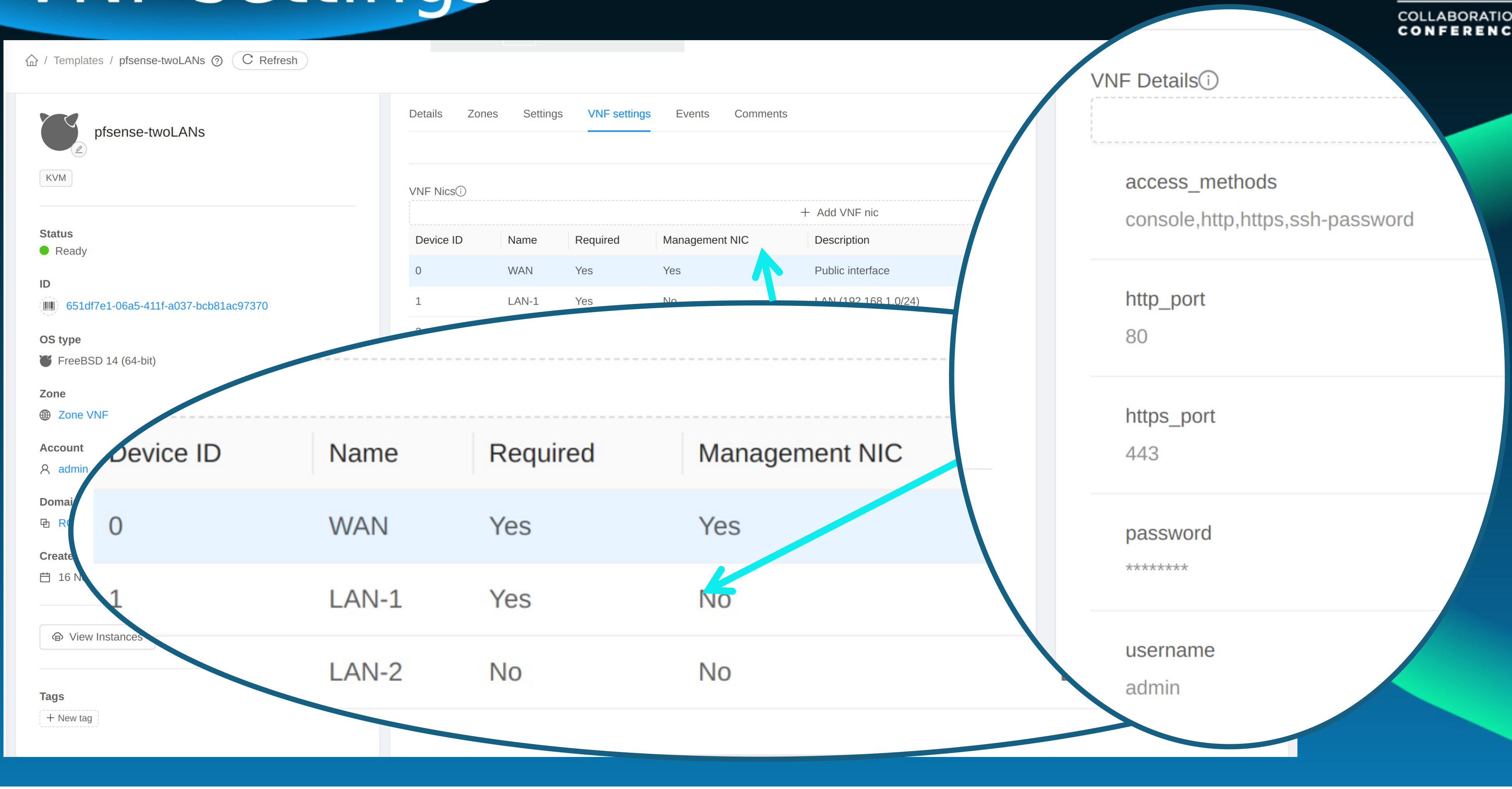
Direct download ⓘ

* OS type ⓘ
FreeBSD 14 (64-bit)

Template type ⓘ
the type of the template. Valid options are: USER/VNF (for all users) and SYST...

USER
VNF
SYSTEM
BUILTIN
ROUTING

VNF settings



Templates / pfsense-twoLANs

pfsense-twoLANs

KVM

Status
Ready

ID
651df7e1-06a5-411f-a037-bcb81ac97370

OS type
FreeBSD 14 (64-bit)

Zone
Zone VNF

Account
admin

Domai
R

Create
16 N

Tags

VNF Nics

Device ID	Name	Required	Management NIC	Description
0	WAN	Yes	Yes	Public interface
1	LAN-1	Yes	No	LAN (192.168.1.0/24)

Device ID	Name	Required	Management NIC
0	WAN	Yes	Yes
1	LAN-1	Yes	No
	LAN-2	No	No

VNF Details

access_methods
console,http,https,ssh-password

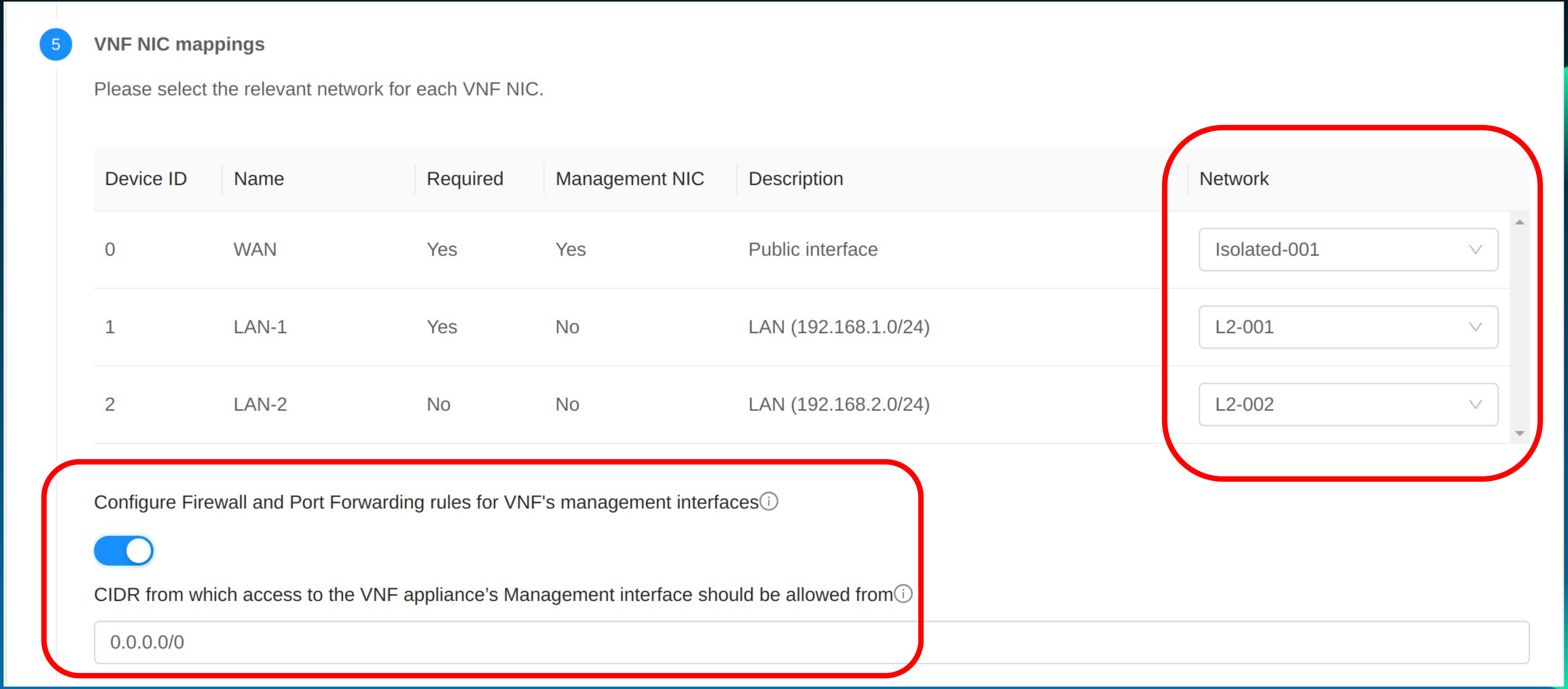
http_port
80

https_port
443

password
****

username
admin

Deploy VM with VNF NIC mappings



5 **VNF NIC mappings**

Please select the relevant network for each VNF NIC.

Device ID	Name	Required	Management NIC	Description	Network
0	WAN	Yes	Yes	Public interface	Isolated-001
1	LAN-1	Yes	No	LAN (192.168.1.0/24)	L2-001
2	LAN-2	No	No	LAN (192.168.2.0/24)	L2-002

Configure Firewall and Port Forwarding rules for VNF's management interfaces ⓘ

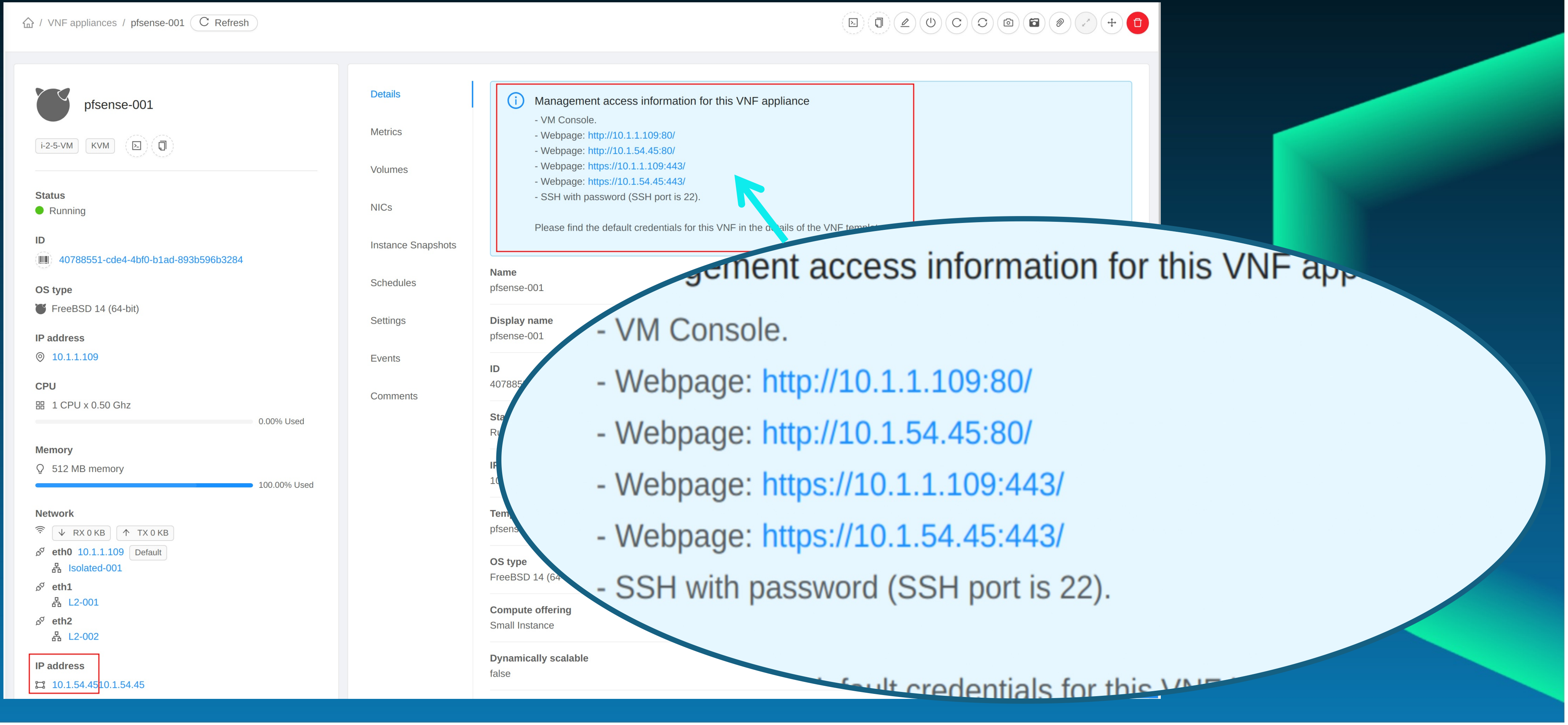
●

CIDR from which access to the VNF appliance’s Management interface should be allowed from ⓘ

0.0.0.0/0

VNF appliance details

cloud STACK COLLABORATION CONFERENCE



/ VNF appliances / pfsense-001 Refresh

pfsense-001

i-2-5-VM KVM

Status
Running

ID
40788551-cde4-4bf0-b1ad-893b596b3284

OS type
FreeBSD 14 (64-bit)

IP address
10.1.1.109

CPU
1 CPU x 0.50 Ghz
0.00% Used

Memory
512 MB memory
100.00% Used

Network
RX 0 KB TX 0 KB
eth0 10.1.1.109 Default
Isolated-001
eth1
L2-001
eth2
L2-002

IP address
10.1.54.4510.1.54.45

Management access information for this VNF appliance
- VM Console.
- Webpage: http://10.1.1.109:80/
- Webpage: http://10.1.54.45:80/
- Webpage: https://10.1.1.109:443/
- Webpage: https://10.1.54.45:443/
- SSH with password (SSH port is 22).

Please find the default credentials for this VNF in the d ails of the VNF templa

Name
pfsense-001

Display name
pfsense-001

ID
407885

Sta
R

IP
10

Tem
pfsens

OS type
FreeBSD 14 (64

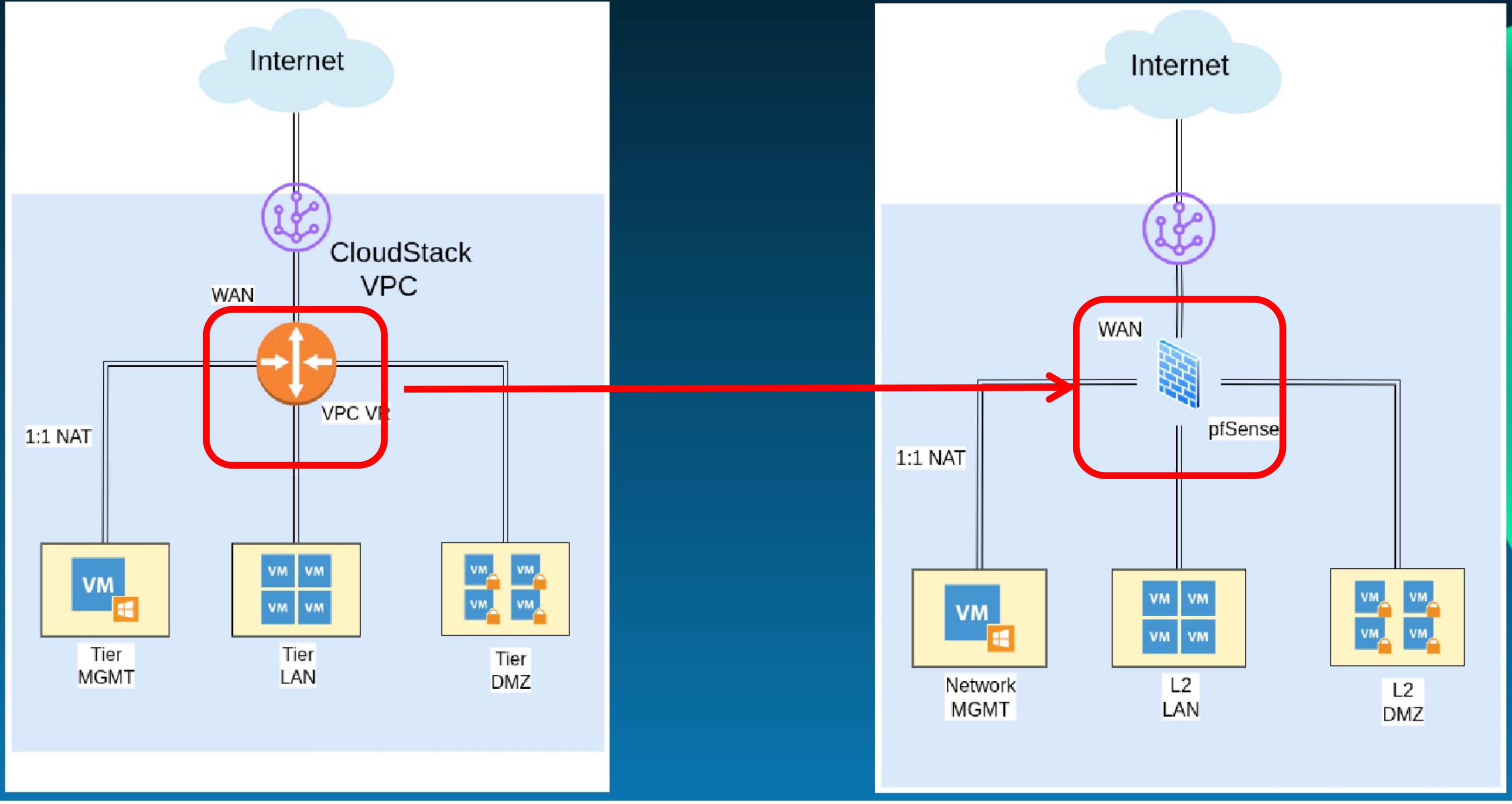
Compute offering
Small Instance

Dynamically scalable
false

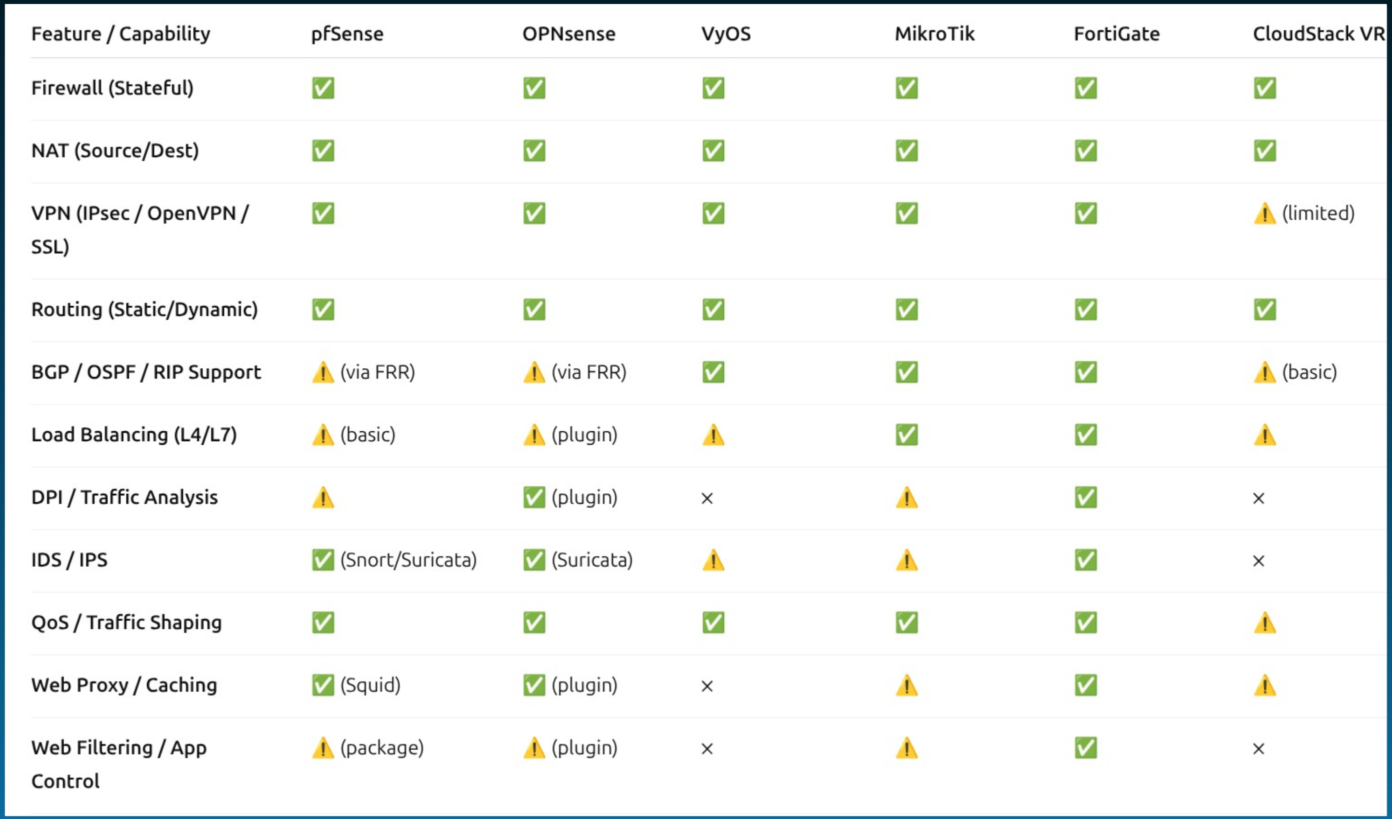
gement access information for this VNF app
- VM Console.
- Webpage: http://10.1.1.109:80/
- Webpage: http://10.1.54.45:80/
- Webpage: https://10.1.1.109:443/
- Webpage: https://10.1.54.45:443/
- SSH with password (SSH port is 22).
fault credentials for this VNF

Use case 1: VR Alternatives

- Router OS

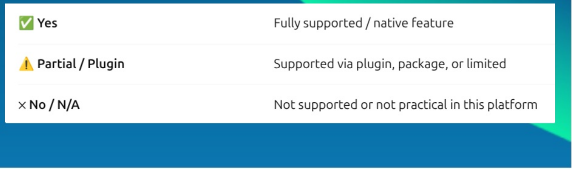


Popular Router OSes



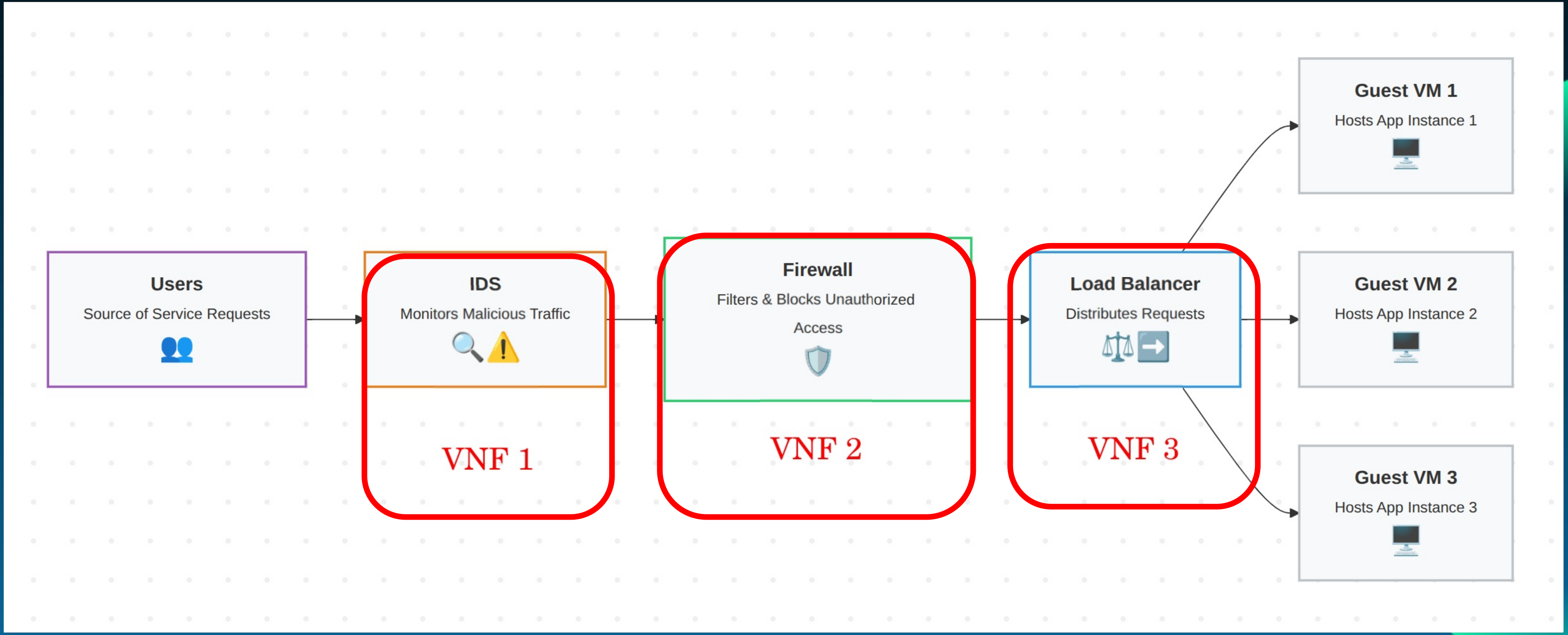
Feature / Capability	pfSense	OPNsense	VyOS	MikroTik	FortiGate	CloudStack VR
Firewall (Stateful)	✅	✅	✅	✅	✅	✅
NAT (Source/Dest)	✅	✅	✅	✅	✅	✅
VPN (IPsec / OpenVPN / SSL)	✅	✅	✅	✅	✅	⚠️ (limited)
Routing (Static/Dynamic)	✅	✅	✅	✅	✅	✅
BGP / OSPF / RIP Support	⚠️ (via FRR)	⚠️ (via FRR)	✅	✅	✅	⚠️ (basic)
Load Balancing (L4/L7)	⚠️ (basic)	⚠️ (plugin)	⚠️	✅	✅	⚠️
DPI / Traffic Analysis	⚠️	✅ (plugin)	×	⚠️	✅	×
IDS / IPS	✅ (Snort/Suricata)	✅ (Suricata)	⚠️	⚠️	✅	×
QoS / Traffic Shaping	✅	✅	✅	✅	✅	⚠️
Web Proxy / Caching	✅ (Squid)	✅ (plugin)	×	⚠️	✅	⚠️
Web Filtering / App Control	⚠️ (package)	⚠️ (plugin)	×	⚠️	✅	×

Source: ChatGPT

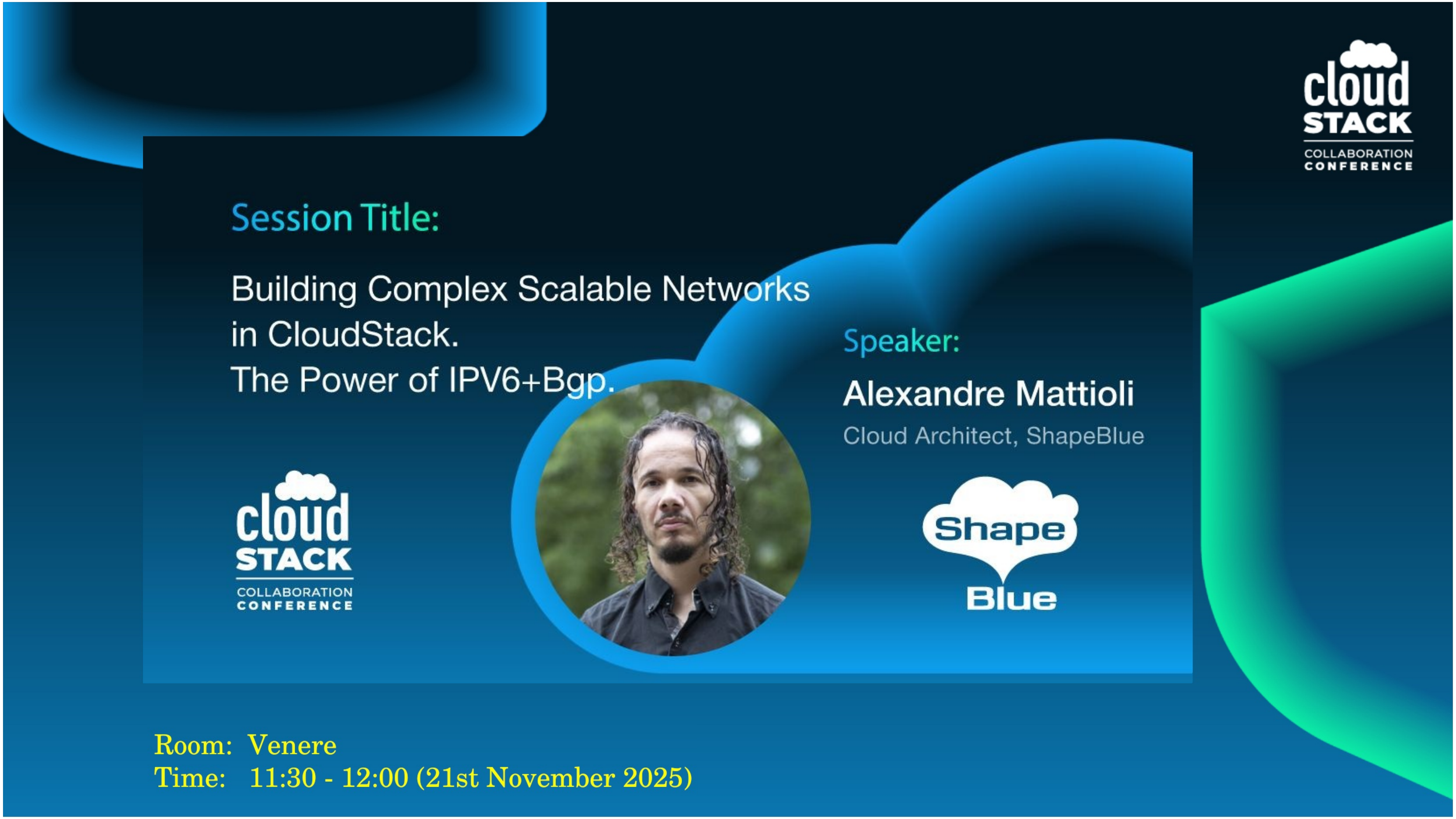


✅ Yes	Fully supported / native feature
⚠️ Partial / Plugin	Supported via plugin, package, or limited
× No / N/A	Not supported or not practical in this platform

Use case: Network service chain



Ref to: ETSI GS NFV-IFA 014: Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Network Service Templates Specification



Session Title:

Building Complex Scalable Networks in CloudStack.
The Power of IPV6+Bgp.

Speaker:

Alexandre Mattioli

Cloud Architect, ShapeBlue

Room: Venere
Time: 11:30 - 12:00 (21st November 2025)

Why a VNF Framework ?

- VRs are fully managed by ACS, but VNFs are not.
- Unified framework: From Ad-Hoc to Strategic
  - Pluggable Architecture
  - Standardized Lifecycle Management
  - Declarative Configuration
  - Diverse Service Operations
- Target to Apache CloudStack 4.23.0

Pluggable Architecture

Problem Statement

- Develop CloudStack Plugins in Java source code, which requires deep Java and CloudStack internals knowledge.
- It is not easy to introduce new vendor and maintain vendor plugins.

Framework Solution

- The framework is designed to be extensible.
- CloudStack creates a clean VNF Provider Interface, handles orchestration, providers handle VNF-specific logic.

Lifecycle Management

Problem Statement

- CloudStack can deploy the VNF appliances, but everything after that was a black box.
- No way to configure VNF appliances like CloudStack VRs in CloudStack.

Framework Solution

- Supports both Orchestration and Configuration

Declarative Configuration

Problem Statement

- The initial configuration relied entirely on the VNF template. Some VNF templates support cloud-init and userdata.
- Lack of health monitoring.

Framework Solution

- Supports complex bootstrap process to establish initial communication
- Introduces health check (incl. Basic network, API endpoint, service status, etc).

Diverse Service Operations

Problem Statement

- Implementing new features is very complex
- Every VNF has different services (firewall, NAT, routing) with different operations (create, read, update, delete).

Framework Solution

- Support self-defined services and operations.
- Supports multiple data models

The Goals

Goal	What It Enables
Standardized Lifecycle Management	Predictable automation Reliable operations
Declarative Configuration	Stanard Model-driven management Configuration as Code
Pluggable Architecture	Ecosystem growth Technology Independence
Operator-Friendly	Centralized Management Consistent Experience Less skills required

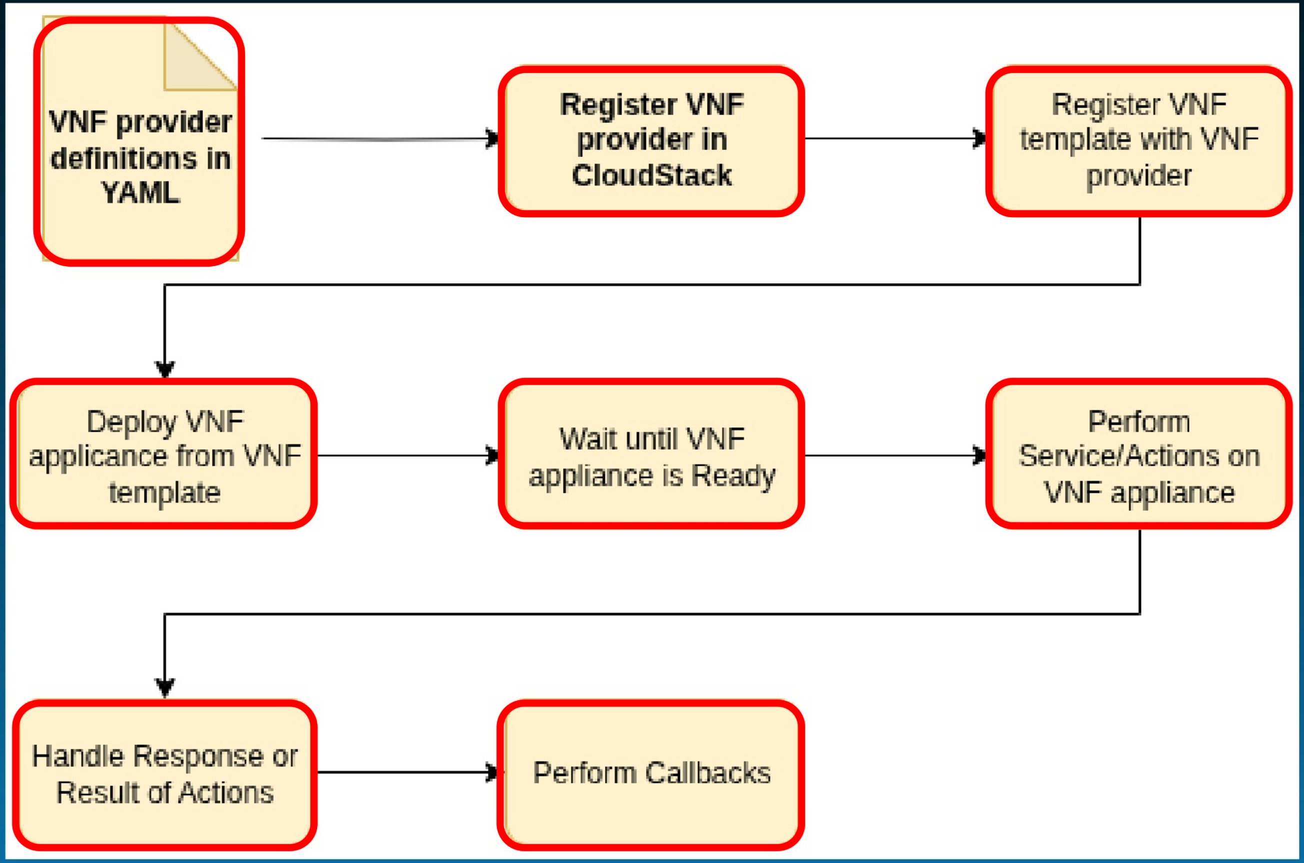
Introduction of VNF framework

- The Solution: **A Declarative YAML-Driven Provider Engine**

- An Intelligent Generic Provider
  - Parses YAML dictionary and executes the defined workflows.
  - Supports VNF-Specific YAML dictionary. For each VNF type (FortiGate, MikroTik, etc.), Users can create a YAML dictionary that describes how to interact with it.

- **Configuration as Code**

Overall Workflow



VNF provider definitions in YAML

Register VNF provider in CloudStack

Register VNF template with VNF provider

Deploy VNF applicance from VNF template

Wait until VNF appliance is Ready

Perform Service/Actions on VNF appliance

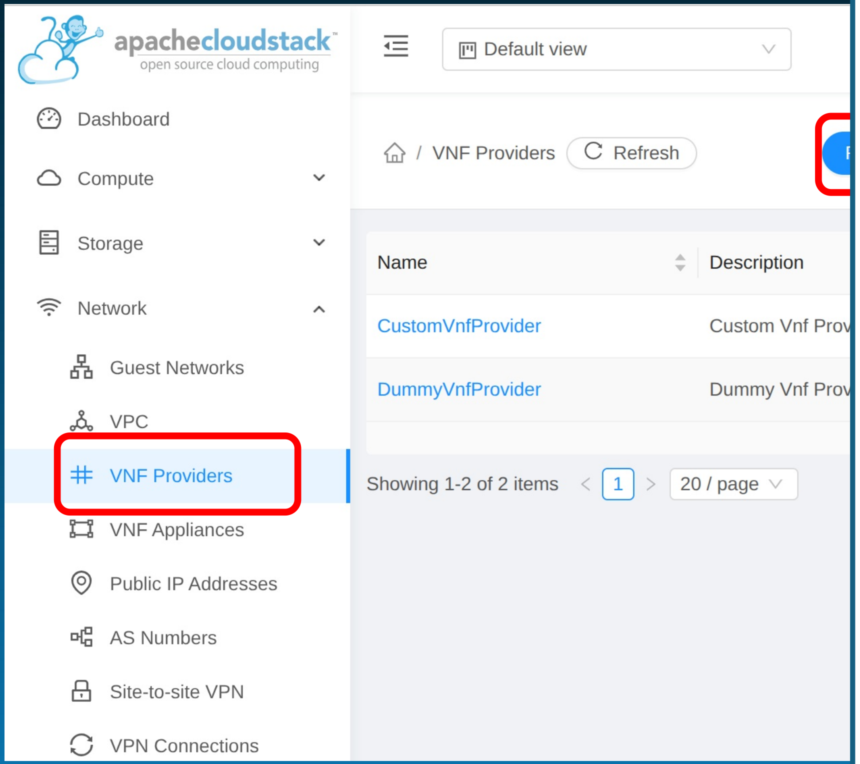
Handle Response or Result of Actions

Perform Callbacks

bold: administrator only

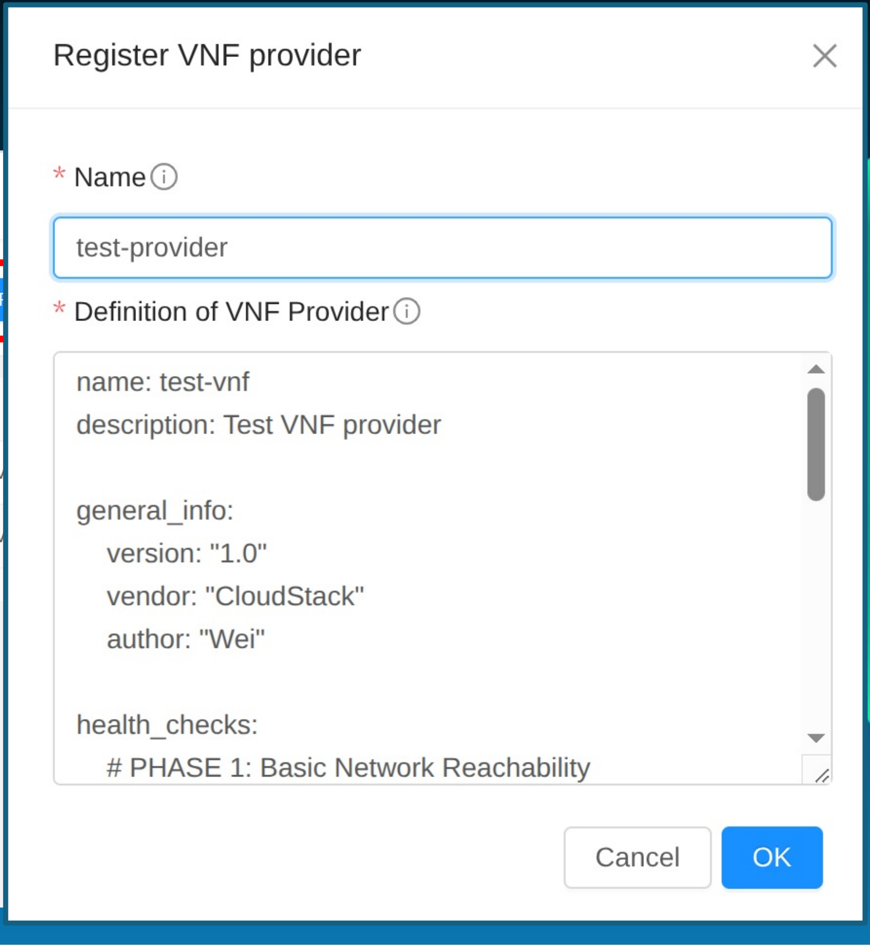
Feature Quick Look

- VNF providers



Name	Description
CustomVnfProvider	Custom Vnf Prov
DummyVnfProvider	Dummy Vnf Prov

Showing 1-2 of 2 items 1 20 / page

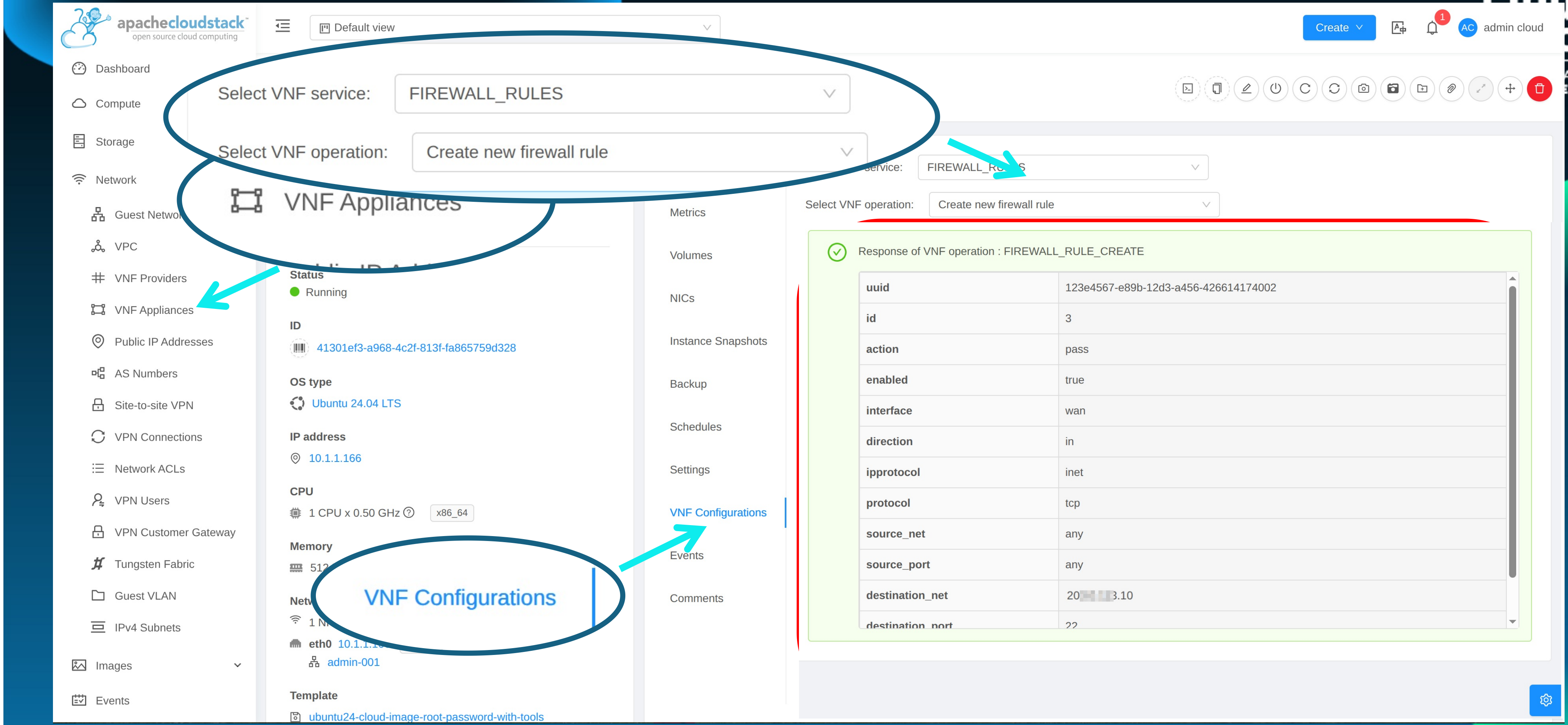


Register VNF provider

* Name: test-provider

* Definition of VNF Provider:

```
name: test-vnf
description: Test VNF provider

general_info:
  version: "1.0"
  vendor: "CloudStack"
  author: "Wei"

health_checks:
  # PHASE 1: Basic Network Reachability
```



Select VNF service: FIREWALL_RULES

Select VNF operation: Create new firewall rule

Status
Running

ID
41301ef3-a968-4c2f-813f-fa865759d328

OS type
Ubuntu 24.04 LTS

IP address
10.1.1.166

CPU
1 CPU x 0.50 GHz x86_64

Memory
512

Netw
1 N
eth0 10.1.1.1
admin-001

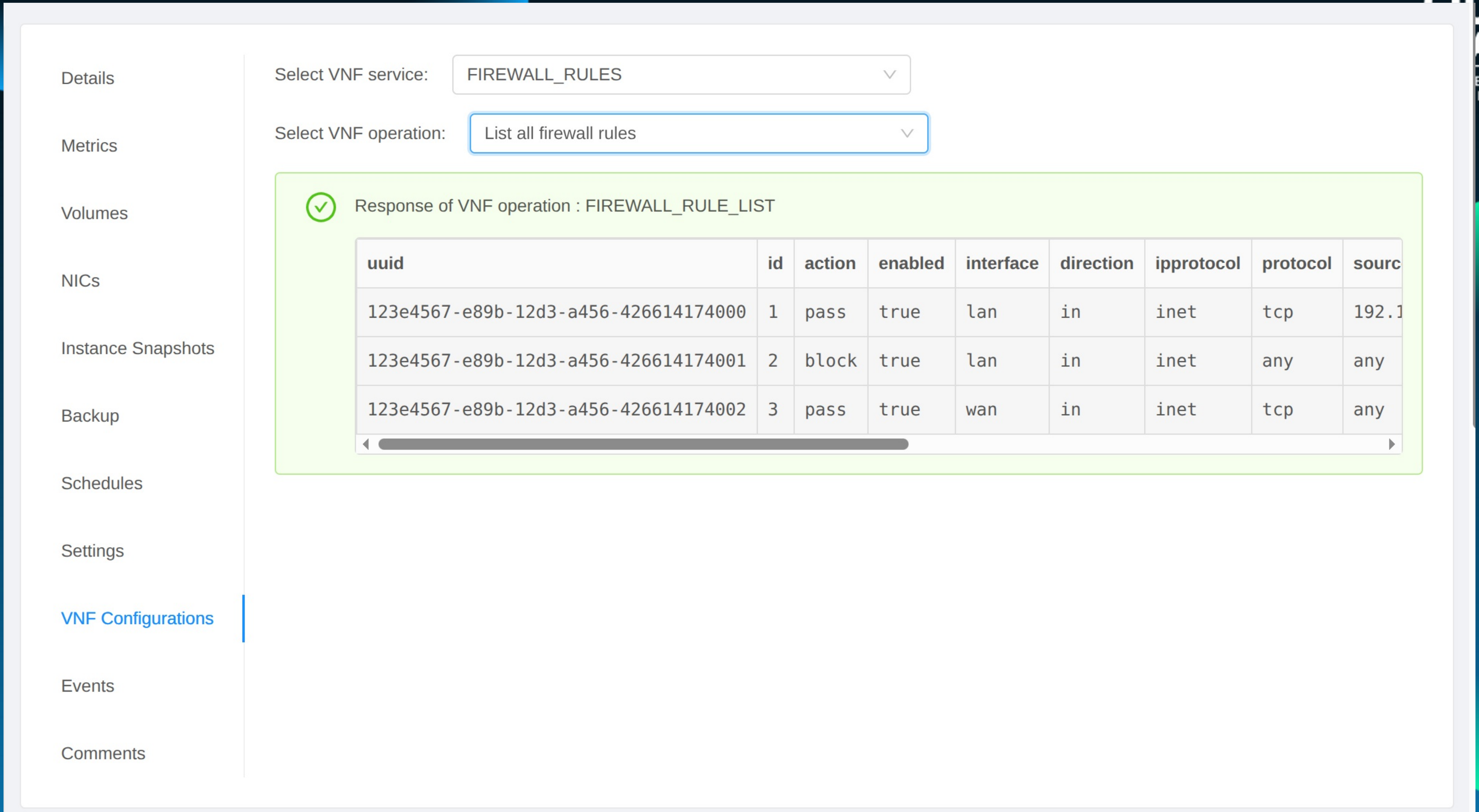
Template
ubuntu24-cloud-image-root-password-with-tools

Service: FIREWALL_RULES

Select VNF operation: Create new firewall rule

Response of VNF operation : FIREWALL_RULE_CREATE

uuid	123e4567-e89b-12d3-a456-426614174002
id	3
action	pass
enabled	true
interface	wan
direction	in
ipprotocol	inet
protocol	tcp
source_net	any
source_port	any
destination_net	20[redacted]3.10
destination_port	22



Select VNF service: FIREWALL_RULES

Select VNF operation: List all firewall rules

Response of VNF operation : FIREWALL_RULE_LIST

uuid	id	action	enabled	interface	direction	ipprotocol	protocol	sourc
123e4567-e89b-12d3-a456-426614174000	1	pass	true	lan	in	inet	tcp	192.1
123e4567-e89b-12d3-a456-426614174001	2	block	true	lan	in	inet	any	any
123e4567-e89b-12d3-a456-426614174002	3	pass	true	wan	in	inet	tcp	any

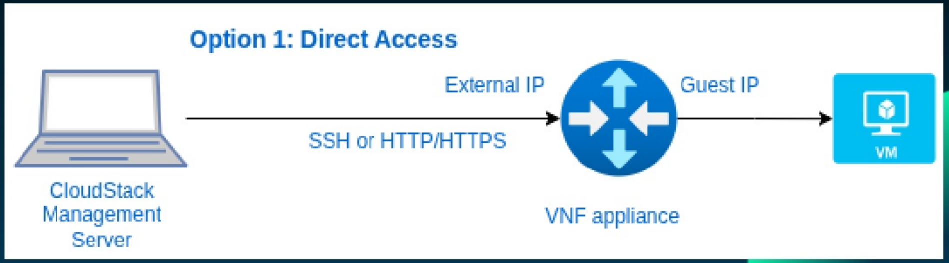
Considerations of VNF framework

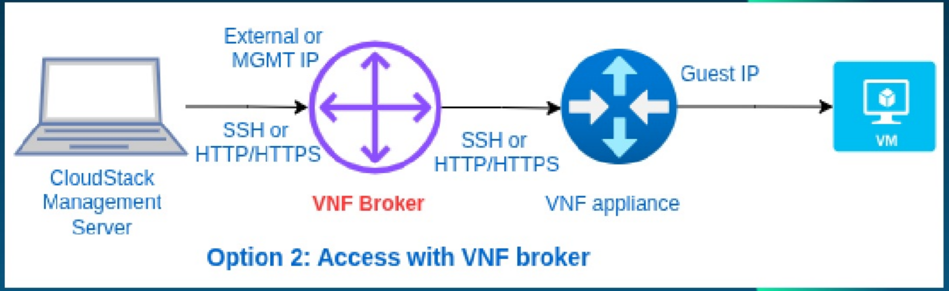
- Complex Connectivity & Access
- Bootstrapping Steps
- Diverse Services and Operations
- State Management Complexity

Complex Connectivity & Access

Problem Statement

- Direct or Indirect?
- How to connect?
  - SSH or HTTP/HTTPS or gRPC ?
- Authentication?





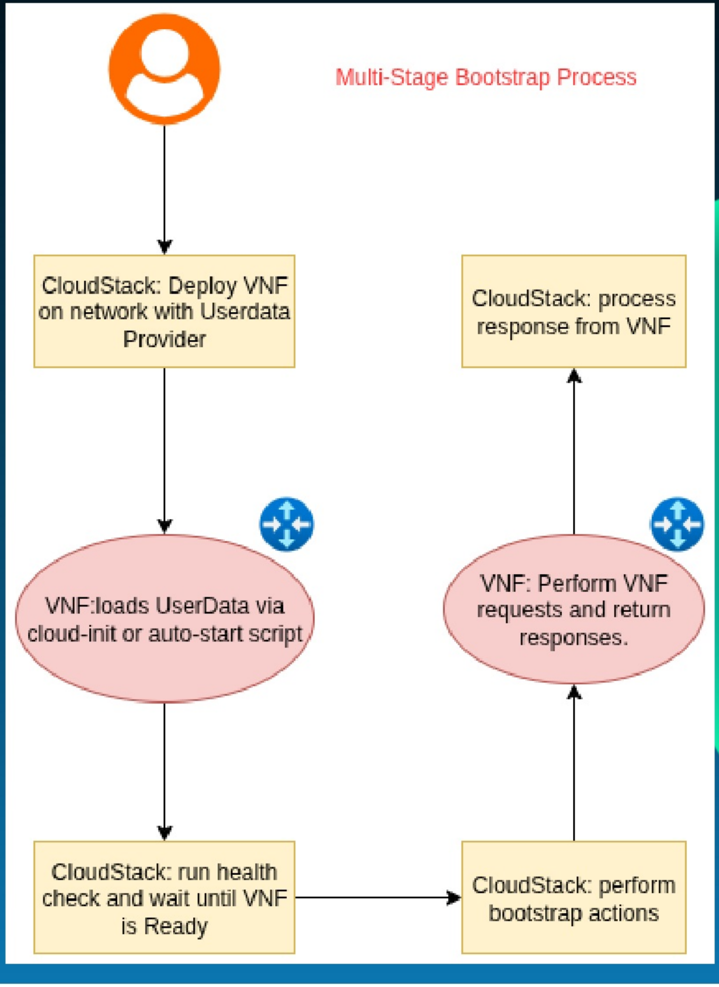
- VNF broker
  - SSH jump host
  - HTTP/HTTPS proxy
  - External or Internal server
  - VM or VR

Bootstrapping and Dependency Hell

Problem Statement

- **"Chicken-and-Egg" Problem**: CloudStack needs to configure the VNF, but the VNF needs to be configured to be reachable and operational.

- define Initial actions
- deal with dependencies of services



Multi-Stage Bootstrap Process

CloudStack: Deploy VNF on network with Userdata Provider

VNF:loads UserData via cloud-init or auto-start script

CloudStack: run health check and wait until VNF is Ready

CloudStack: perform bootstrap actions

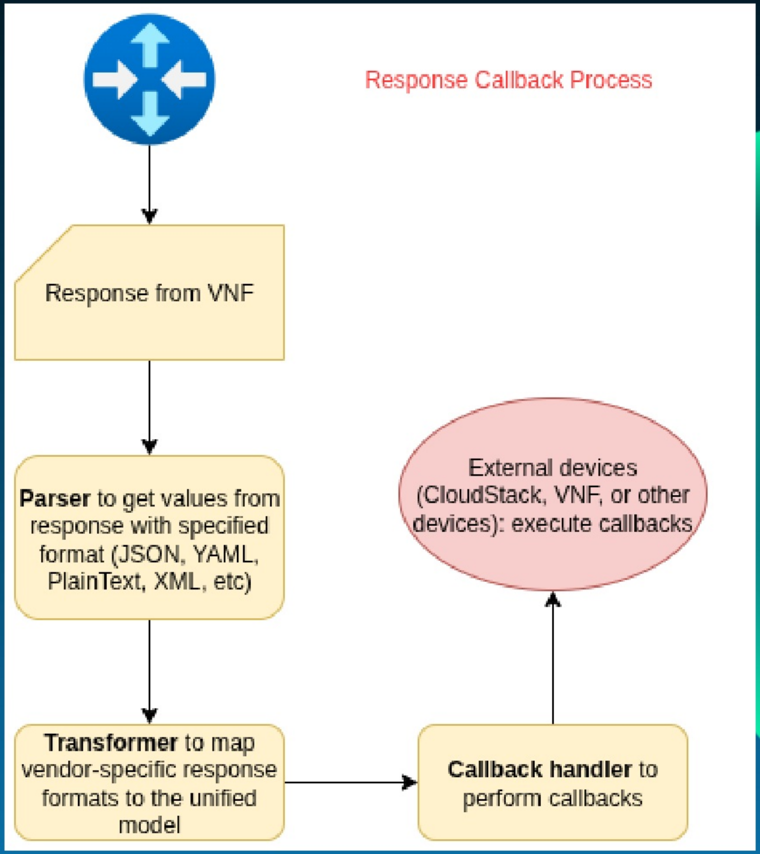
VNF: Perform VNF requests and return responses.

CloudStack: process response from VNF

State Management Complexity

Problem Statement

- CloudStack sends configuration request
- Needs to parse the responses or results with different formats
- handle success or failures



Response Callback Process

Response from VNF

Parser to get values from response with specified format (JSON, YAML, PlainText, XML, etc)

Transformer to map vendor-specific response formats to the unified model

Callback handler to perform callbacks

External devices (CloudStack, VNF, or other devices): execute callbacks

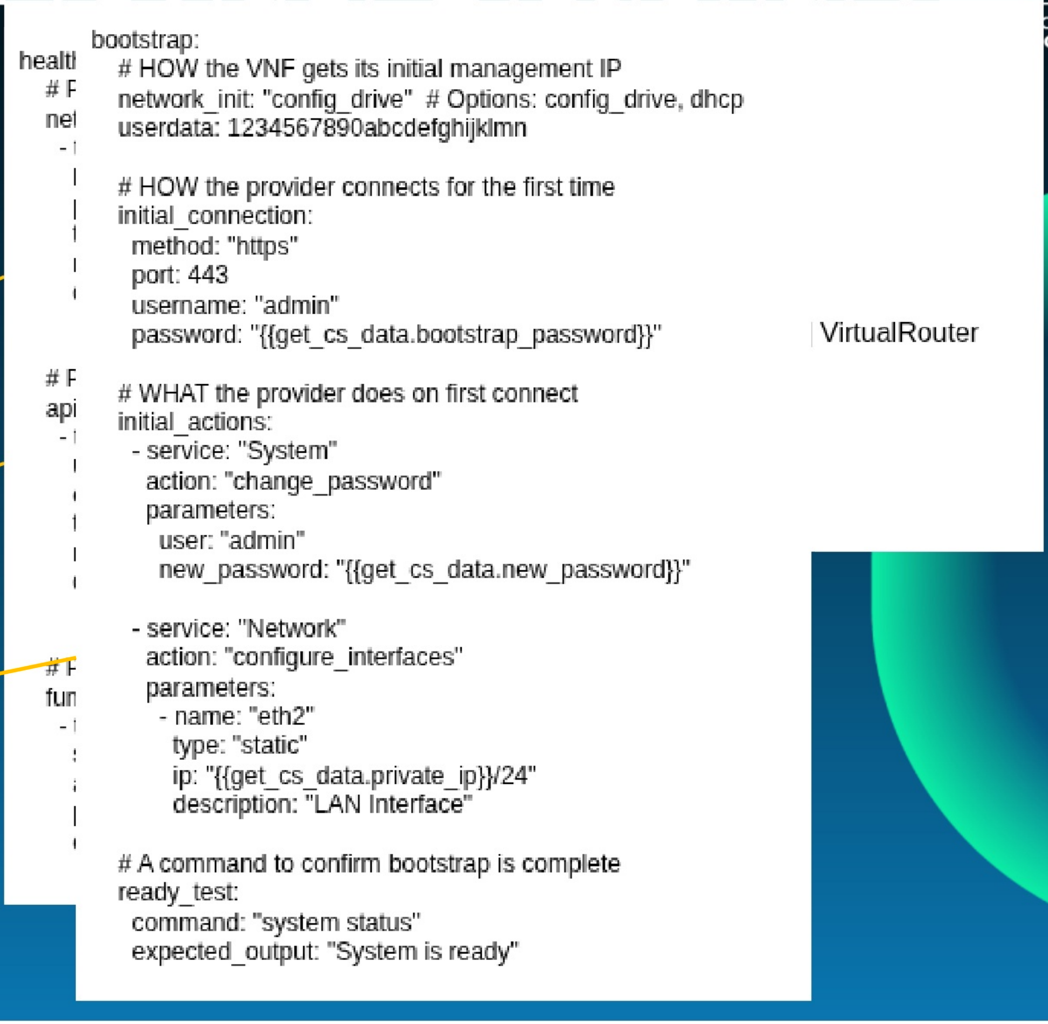
VNF Provider Dictionary in YAML

- Outline

Section	Description
general information	VNF information (name, description, version, debug, global settings)
connections	Access information: SSH/HTTP/HTTPS details, VNF broker, etc
healthchecks	Ways to detect if VNF is healthy
bootstrap	Bootstrap steps and health check during bootstrap
services	Define services and actions

Sample of VNF Provider Dictionary

```
name: RouterOS v3

general_info:
    version: 3.0
    debug: true

connections:
  ...

healthchecks:
  ...

bootstrap:
  ...

services:
  ...
```

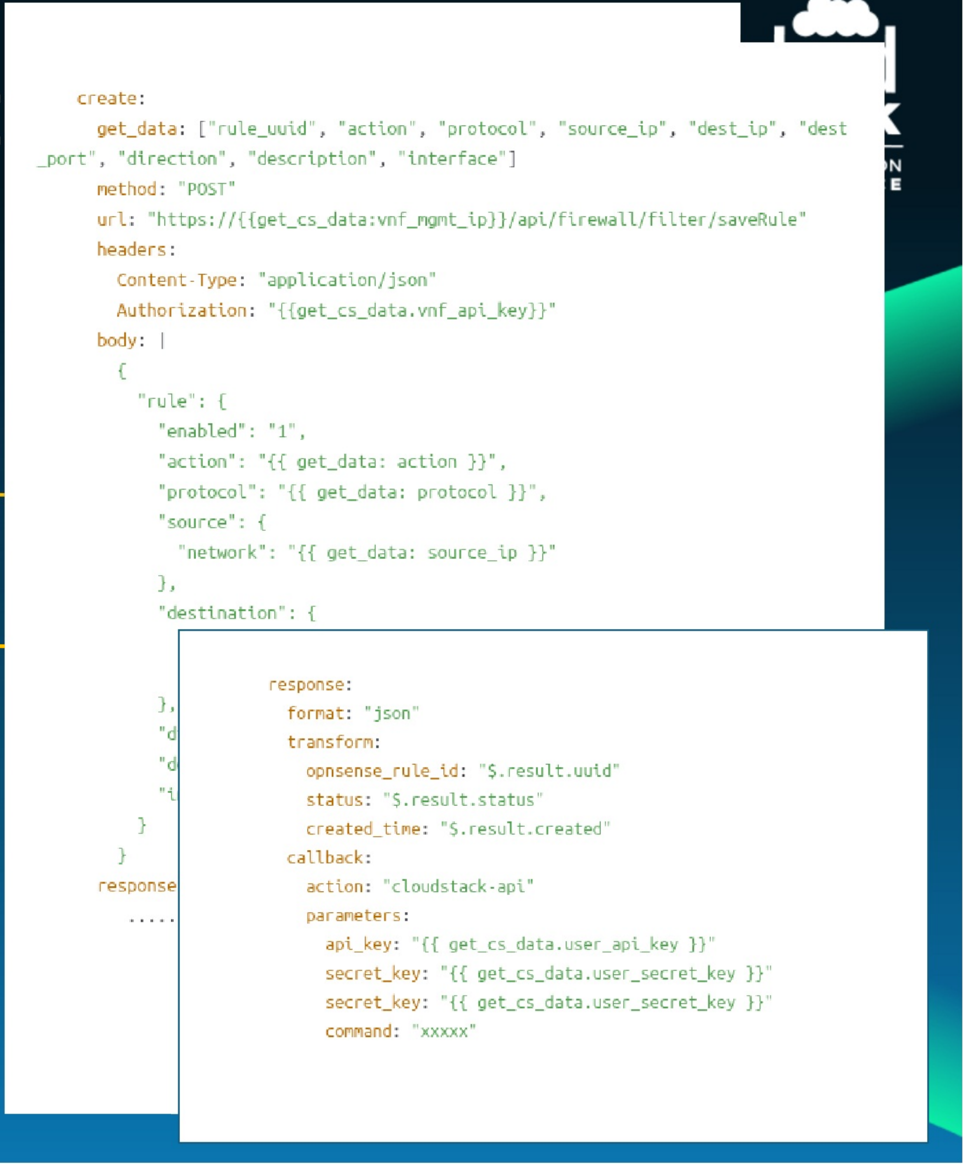


```
bootstrap:
  # HOW the VNF gets its initial management IP
  network_init: "config_drive"  # Options: config_drive, dhcp
  userdata: 1234567890abcdefghijklmn

  # HOW the provider connects for the first time
  initial_connection:
    method: "https"
    port: 443
    username: "admin"
    password: "{{get_cs_data.bootstrap_password}}"

  # WHAT the provider does on first connect
  initial_actions:
    - service: "System"
      action: "change_password"
      parameters:
        user: "admin"
        new_password: "{{get_cs_data.new_password}}"

    - service: "Network"
      action: "configure_interfaces"
      parameters:
        - name: "eth2"
          type: "static"
          ip: "{{get_cs_data.private_ip}}/24"
          description: "LAN Interface"

  # A command to confirm bootstrap is complete
  ready_test:
    command: "system status"
    expected_output: "System is ready"
```

VirtualRouter

Sample of VNF Servic

```
services:
  Firewall:
     data:    # input data from cloudstack API
        ....
     create:  # create a Firewall rule
        ....
     delete:   # delete a Firewall rule
        ...
     update:  # update a Firewall rule
        ...
      list:      # list Firewall rules
        ...
```



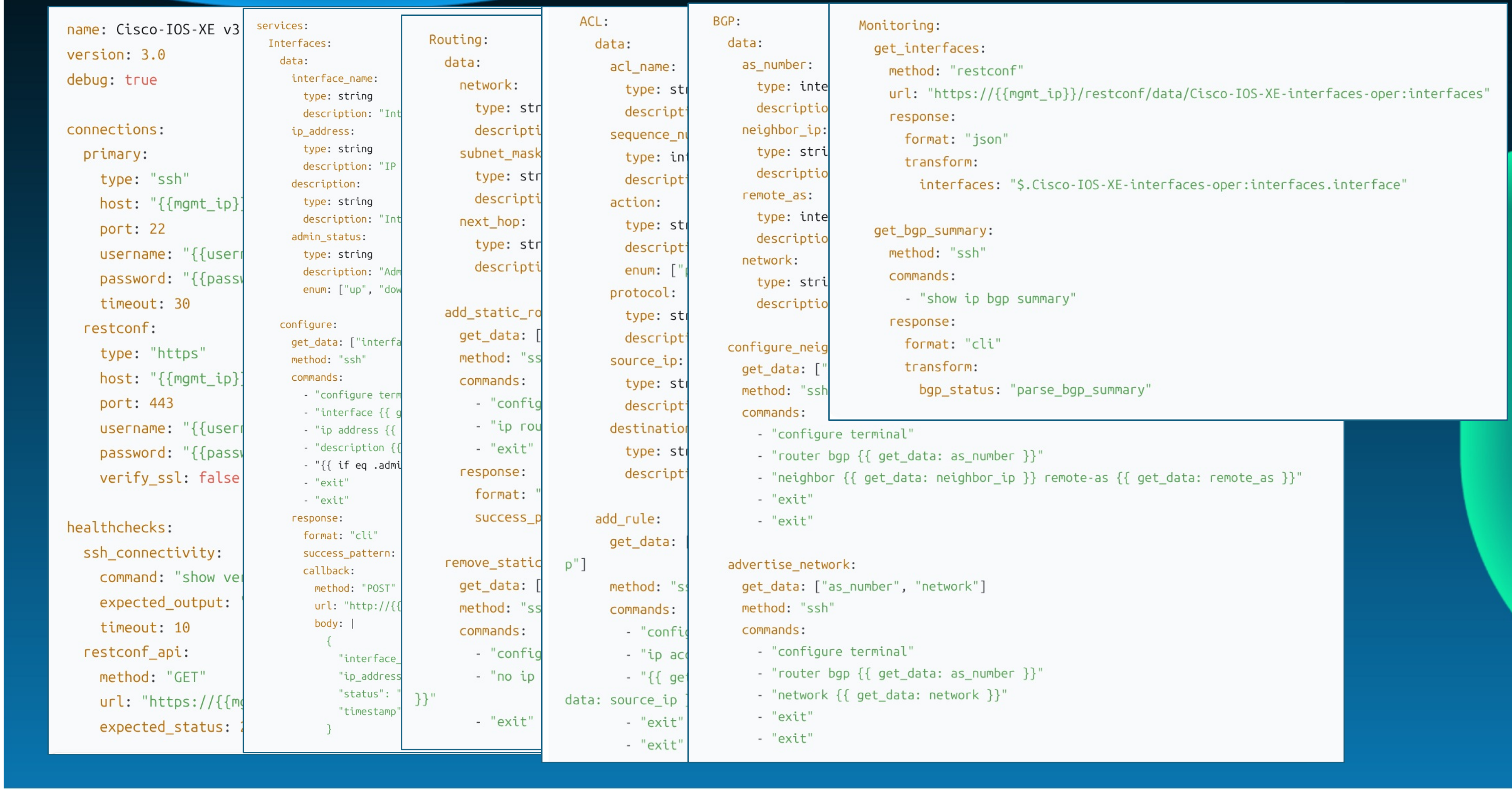
```
    create:
      get_data: ["rule_uuid", "action", "protocol", "source_ip", "dest_ip", "dest
_port", "direction", "description", "interface"]
      method: "POST"
      url: "https://{{get_cs_data:vnf_mgmt_ip}}/api/firewall/filter/saveRule"
      headers:
        Content-Type: "application/json"
        Authorization: "{{get_cs_data.vnf_api_key}}"
      body: |
        {
          "rule": {
            "enabled": "1",
            "action": "{{ get_data: action }}",
            "protocol": "{{ get_data: protocol }}",
            "source": {
              "network": "{{ get_data: source_ip }}"
            },
            "destination": {


            },
            "d
            "d
            "i
          }
        }
      response
        .....
```

```
        response:
          format: "json"
          transform:
            opnsense_rule_id: "$.result.uuid"
            status: "$.result.status"
            created_time: "$.result.created"
          callback:
            action: "cloudstack-api"
            parameters:
              api_key: "{{ get_cs_data.user_api_key }}"
              secret_key: "{{ get_cs_data.user_secret_key }}"
              secret_key: "{{ get_cs_data.user_secret_key }}"
              command: "xxxxx"
```

Conclusion & Takeaways

- VNF framework: A Declarative YAML-Driven Provider Engine
  - Lifecycle Management: Support VNF configurations
  - Lack of Health Monitoring: Introduce health check and status
  - Configuration Passing: support bootstrap configuration
  - No Standardized Way for Advanced Features: Support services and operations defined by YAML
  - A Managed Ecosystem for Users: Support CRUD operation of VNF providers

Configuration as Code



```
name: Cisco-IOS-XE v3
version: 3.0
debug: true

connections:
  primary:
    type: "ssh"
    host: "{{mgmt_ip}}
    port: 22
    username: "{{user
    password: "{{pass
    timeout: 30
  restconf:
    type: "https"
    host: "{{mgmt_ip}}
    port: 443
    username: "{{user
    password: "{{pass
    verify_ssl: false

healthchecks:
  ssh_connectivity:
    command: "show ve
    expected_output: '
    timeout: 10
  restconf_api:
    method: "GET"
    url: "https://{{m
    expected_status: 2
```

```
services:
  Interfaces:
    data:
      interface_name:
        type: string
        description: "Int
      ip_address:
        type: string
        description: "IP
      description:
        type: string
        description: "Int
      admin_status:
        type: string
        description: "Adm
        enum: ["up", "dow

    configure:
      get_data: ["interfa
      method: "ssh"
      commands:
        - "configure term
        - "interface {{ g
        - "ip address {{
        - "description {{
        - "{{ if eq .admi
        - "exit"
        - "exit"
      response:
        format: "cli"
        success_pattern:
        callback:
          method: "POST"
          url: "http://{{
          body: |
            {
              "interface_
              "ip_address
              "status": "
              "timestamp"
            }
```

```
Routing:
  data:
    network:
      type: str
      descripti
    subnet_mask
      type: str
      descripti
    next_hop:
      type: str
      descripti

  add_static_ro
    get_data: [
    method: "ss
    commands:
      - "config
      - "ip rou
      - "exit"
    response:
      format: "
      success_p

  remove_static
    get_data: [
    method: "ss
    commands:
      - "config
      - "no ip
}}"
      - "exit"
```

```
ACL:
  data:
    acl_name:
      type: st
      descript
    sequence_nu
      type: in
      descript
    action:
      type: st
      descript
      enum: ["
    protocol:
      type: st
      descript
    source_ip:
      type: st
      descript
    destinatio
      type: st
      descript

  add_rule:
    get_data: [
p"]
    method: "s
    commands:
      - "confi
      - "ip ac
      - "{{ ge
data: source_ip
      - "exit"
      - "exit"
```

```
BGP:
  data:
    as_number:
      type: inte
      descriptio
    neighbor_ip:
      type: stri
      descriptio
    remote_as:
      type: inte
      descriptio
    network:
      type: stri
      descriptio

  configure_neig
    get_data: ["
    method: "ssh
    commands:
      - "configure terminal"
      - "router bgp {{ get_data: as_number }}"
      - "neighbor {{ get_data: neighbor_ip }} remote-as {{ get_data: remote_as }}"
      - "exit"
      - "exit"

  advertise_network:
    get_data: ["as_number", "network"]
    method: "ssh"
    commands:
      - "configure terminal"
      - "router bgp {{ get_data: as_number }}"
      - "network {{ get_data: network }}"
      - "exit"
      - "exit"
```

```
Monitoring:
  get_interfaces:
    method: "restconf"
    url: "https://{{mgmt_ip}}/restconf/data/Cisco-IOS-XE-interfaces-oper:interfaces"
    response:
      format: "json"
      transform:
        interfaces: "$.Cisco-IOS-XE-interfaces-oper:interfaces.interface"

  get_bgp_summary:
    method: "ssh"
    commands:
      - "show ip bgp summary"
    response:
      format: "cli"
      transform:
        bgp_status: "parse_bgp_summary"
```

Configuration as Code

- Not just ideas
- Under active development (target to ACS 4.23)
- Bring your ideas to CloudStack. Feedbacks are welcome !

Extensions Framework & Orchestrate Anything

Harikrishna Patnala, Marco Sinhoreli

Room: Venere
Time: 15:30 - 16:00 (20st November 2025)

Extension v.s. VNF framework

Use cases	Extension framework	VNF framework
Proxmox	• orchestrate **VM Instances** on Proxmox, Hyper-V	• manage **Proxmox** itself as a network appliance
Ceph	• (could) orchestrate **Volumes** on Ceph	• manage **Ceph monitors** as network appliance (incl. Volumes management, and others)
SDN devices	• (could) orchestrate **Virtual Networks** and manage network rules on SDN	• manage **SDN controller** via API or so.
Summary	• CloudStack **orchestrates** resources on external devices • *CloudStack is the boss*	• CloudStack **configures** the external devices • *CloudStack talks to the boss*

cloud STACK COLLABORATION CONFERENCE

Customizing CloudStack Network with VNF Appliances

Discussion & Feedback

Share your inputs with weizhou@apache.org