

# How to Deploy Virtual Test Agents in OpenStack

Published  
2021-01-13

# Table of Contents

**Executive Summary**

**Paragon Active Assurance: Solution Overview**

**Prerequisites**

**Launching the vTA Image in OpenStack**

**Appendix: Description of the vTA VNF and Its Requirements**

# Executive Summary

This guide explains how to deploy Virtual Test Agents (vTAs) from Paragon Active Assurance in OpenStack and how to control these from Paragon Active Assurance Control Center.

In ETSI NFV terminology, these vTAs are likewise referred to as Virtual Test Agents, and the centralized controller is referred to as the combination of Test Controller (TC) and Test Result Analysis Module (TRAM).

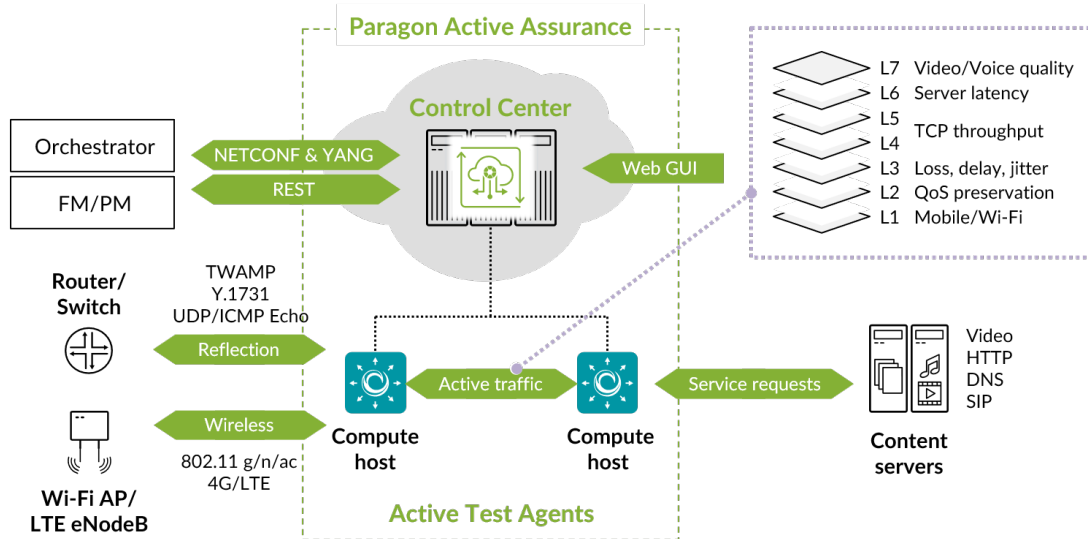
This guide assumes that you have a basic knowledge of OpenStack and Network Function Virtualization (NFV), and that you have your own OpenStack environment in which to launch the vTA images.

## Paragon Active Assurance: Solution Overview

Paragon Active Assurance consists of two parts:

- 1. Test Agents** – software-based active traffic generators. Virtual Test Agents (vTAs) are ones that you upload and boot from your own OpenStack environment. These vTAs will automatically connect to Control Center as part of the deployment process described in this guide. (Juniper Networks also offers non-virtual Test Agents in the form of software that is installed on stand-alone x86 hardware.)
- 2. Control Center** – for centralized control and coordination of Test Agents, including distributed VNF vTAs. This includes initiating test sequences and monitoring sessions, as well as evaluating collected measurement data, SLAs and KPIs.

Paragon Active Assurance vTAs are controlled through Control Center. The interface towards Control Center is either a web GUI or an orchestration API, as illustrated below:



## Prerequisites

### IN THIS SECTION

- [Control Center Account | 2](#)
- [vTA Image | 3](#)

## Control Center Account

You need an account in a Paragon Active Assurance Control Center in order to access it: either the one belonging to the Paragon Active Assurance SaaS solution or one installed on-premise in your organization. If you do not already have a Paragon Active Assurance account, please contact your Juniper partner or your local Juniper account manager or sales representative.

# vTA Image

The VNF vTA image is provided either by Juniper's partners or directly from Juniper.

The vTA image for OpenStack is provided in either raw or QCOW2 format.

## Launching the vTA Image in OpenStack

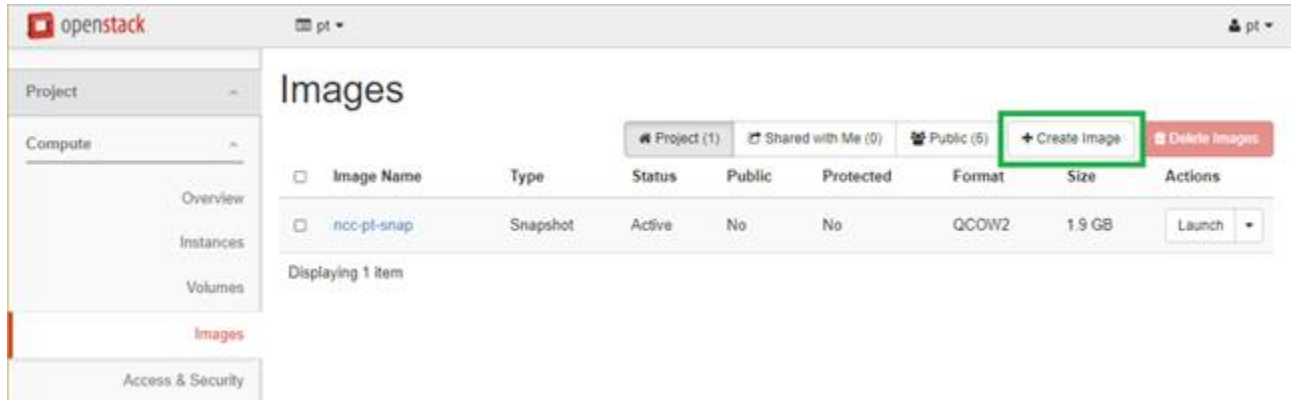
### IN THIS SECTION

- [Uploading the vTA Image to OpenStack | 3](#)
- [Creating a Flavor for the vTA Image | 4](#)
- [Launching an Instance of the vTA Image | 5](#)
- [Launching a vTA Image Manually in Horizon | 6](#)
- [Launching a vTA Image Using a Heat Orchestration Template in Horizon | 8](#)
- [Launching a vTA Image Using the Python OpenStack API | 11](#)

## Uploading the vTA Image to OpenStack

First you need to upload the vTA image to your NFV/OpenStack platform. In OpenStack's Horizon GUI, do as follows:

- On the navigation bar on the left, expand **Project**, and under **Compute**, click **Images**.
- Click the **Create Image** button. Fill out the dialog that appears, and create the image.



## Creating a Flavor for the vTA Image

*Flavors* in OpenStack are virtual hardware templates, defining RAM allocation, block storage, and number of CPU cores. In the Horizon GUI, flavors are shown and created under **Admin** → **System** → **Flavors**.

The minimum recommended flavor for a vTA is:

- 1 vCPU
- 512 MB RAM
- 2 GB block storage

See below for some examples of flavors, including one called “netrounds.small.2GB” which corresponds to the above specifications.

Project

Admin

System

Overview

Resource Usage

Hypervisors

Host Aggregates

Instances

Volumes

Flavors

# Flavors

<input type="checkbox"/>	Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk
<input type="checkbox"/>	m1.large	2	6GB	10GB	4GB	0MB
<input type="checkbox"/>	m1.medium	2	4GB	4GB	4GB	0MB
<input type="checkbox"/>	m1.ncc	1	2GB	10GB	0GB	0MB
<input type="checkbox"/>	netrounds-tiny	1	256MB	1GB	0GB	0MB
<input type="checkbox"/>	netrounds.medium	2	512MB	5GB	0GB	0MB
<input type="checkbox"/>	netrounds.small.2GB	1	512MB	2GB	0GB	0MB

## Launching an Instance of the vTA Image

The final step is to launch an *instance* of the vTA image in OpenStack. Things will look a bit different depending on whether you are using a HEAT Orchestration Template (HOT), which is essentially a virtual machine descriptor, or whether you are specifying the details when launching the vTA. See below for instructions on how to launch the vTA:

- manually – section ["Launching a vTA Image Manually in Horizon" on page 6](#)
- using HOT automation – section ["Launching a vTA Image Using a Heat Orchestration Template in Horizon" on page 8](#)
- using the Python OpenStack API – section ["Launching a vTA Image Using the Python OpenStack API" on page 11.](#)

Once you have launched the vTA image, the vTA will register with Control Center and appear in its web GUI under **Test Agents**. Check for the vTA name in that view to verify that the vTA has registered. You can then initiate tests and monitoring sessions on the vTA from Control Center.

# Launching a vTA Image Manually in Horizon

Here is how to deploy a vTA manually, without using a HEAT Orchestration Template.

- On the navigation bar, select **Project** → **Compute** → **Instances**. Then click **Launch instance**.
- Under **Details**, enter a name for the vTA instance.
- Under **Source**, select the image to boot the vTA instance from. In general, the default volume settings can be kept.
- Under **Flavor**, select a suitable flavor for the vTA. Please refer back to the section ["Creating a Flavor for the vTA Image" on page 4](#).
- Under **Networks**, select whatever is most appropriate in your case.
- The **Network Ports** section can be skipped.
- Under **Security Groups**, the group chosen must satisfy the following:
  - The vTA must be able to establish an outgoing session towards its Control Center: in the cloud server case, to login.netrounds.com using TCP port 443; in the on-premise server case, to the host IP using TCP port 6000 (default).
  - UDP port 123 must be open to permit NTP time sync.
  - Traffic must be allowed on all ports needed for the testing you intend to do with the vTA.
- The **Key Pair** section can be skipped, as SSH is not in use for vTAs.
- Under **Configuration**, in the **Customization Script** box, "cloud-config" metadata must be entered specifying among other things how to connect to Control Center. The format of this metadata is given in the ["Format of "cloud-config" Metadata" on page 6](#) section below.
- The **Server Groups**, **Scheduler Hints**, and **Metadata** sections can be skipped.
- Finish by clicking the **Launch Instance** button at the bottom of the dialog.

A vTA image can also be deployed from the command line, but that method is not explained further in this introductory guide.

## Format of "cloud-config" Metadata

The vTA supports only #cloud-config metadata. Its format is as shown below.



**NOTE:** The `#cloud-config` and `netrounds_test_agent` lines must be present, and that all of the remaining lines must be indented.

Basic configuration:

```
#cloud-config
netrounds_test_agent:
  name: MyvTA                    # vTA name
  email: john.doe@example.com    # NCC user email
  password: secret               # NCC user password
  account: theaccount            # NCC account
  server: login.netrounds.com:443 # Login server
  management_interface: eth0     # Test Agent management interface
  management_address_type: dhcp  # Can be "dhcp" or "static"
```

The following parameters are required only if `management_address_type` is "static":

```
## Set the following if using "static" above:
# management_ip: 192.168.1.2/24      # Management IP address
# management_dns: 8.8.8.8, 8.8.4.4   # DNS server IP address(es)
# management_default_gateway: 192.168.1.1 # Gateway IP address
# management_ntp: ntp.netrounds.com   # NTP server IP address or host
name
```

The following parameters are required only if the vTA is connecting to the server through an HTTP proxy:

```
## Set the following if using an HTTP proxy:
# http_proxy: myproxy.lan           # Proxy server
# http_proxy_port: 80                # Proxy port
# http_proxy_auth_type: none         # Can be "none" or "basic"
```

The following parameters are required only if `http_proxy_auth_type` is "basic":

```
# http_proxy_username: johndoe      # Proxy authorization user name
# http_proxy_password: secret       # Proxy authorization password
```

The following parameters are required if IPv6 is to be used for Test Agent management connections:

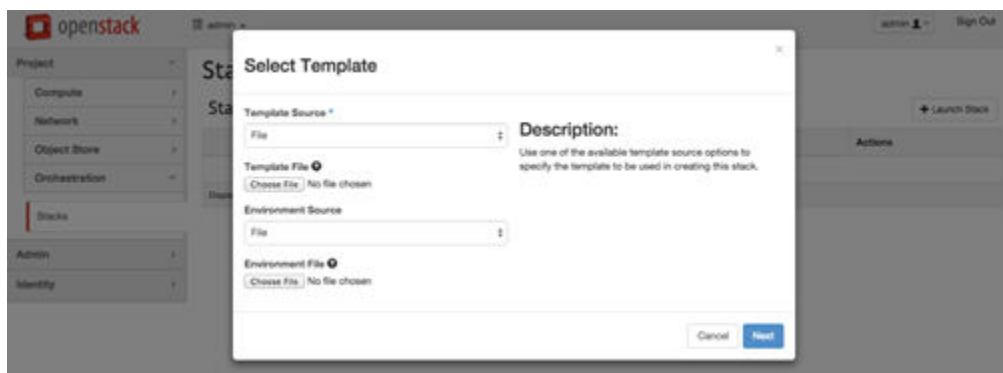
```
## Note: IPv6 management requires special config, see separate documentation
# management_enable_ipv6: False
# management_ntp_allow_ipv6: False
# management_address6_type: none          # Can be "dhcp", "slaac", or
"static"

## Set if "static". Note: Use CIDR format for IP
# management_ip6: 2001:db8:85a3::8a2e:370:7334/64
# management_dns6: 2001:4860:4860::8888, 2001:4860:4860::8844
# management_default_gateway6: <gateway IP address>
```

## Launching a vTA Image Using a Heat Orchestration Template in Horizon

To deploy a vTA image in OpenStack using a Heat Orchestration Template (HOT):

- On the navigation bar, select **Project** → **Orchestration** → **Stacks**.
- Click the **Launch Stack** button on the far right. Select the HOT file to be used. For an example of such a file, see the section ["Example of HOT File" on page 9](#).



The template will ask for input to be specified, as shown in the following screenshot:

Some of these fields have default values, as defined in the HOT file, while others are variables in the template.

After a vTA image has been booted up, it must receive its instance-specific userdata/metadata so that it can properly configure the network interfaces and call back home to Control Center. The vTA supports two ways to provide this instance-specific data:

1. Requesting the data from a service running at <http://169.254.269.254> on the compute host. This is similar to the approach taken by an EC2 instance running in Amazon WS.
2. Reading the data from a special configuration drive that is attached to the vTA instance when it boots. This is referred to as the "config-drive method".

The format of the instance-specific userdata/metadata is given in the section ["Format of "cloud-config" Metadata" on page 6](#)

Using either of the above methods, it is possible to automate connecting the vTA to Netrounds Control Center.

**NOTE:** You need to ensure that the vTA can establish an outgoing session towards its Control Center: in the cloud server case, to [login.netrounds.com](http://login.netrounds.com) using TCP port 443; in the on-premise server case, to the host IP using TCP port 6000 (default). Alternatively, the vTA may connect via an HTTP proxy. In addition, UDP port 123 needs to be open to permit NTP time sync.

## Example of HOT File

An example of a HOT file is provided below. Replace the **boldface text** in square brackets with your input.

```
heat_template_version: 2015-07-15

description: Heat template to deploy a single vTA in OpenStack
parameters:
  account_name:
    type: string
    description: Paragon Active Assurance account name to be used for
autoregistration
  email:
    type: string
    description: User name (email address) for Netrounds account
  account_password:
    type: string
    hidden: true
    description: Paragon Active Assurance account password to be used
for autoreg
  vTA_name:
    type: string
    description: Name to display in Paragon Active Assurance inventory
    default: vTA1
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
    default: [Replace with vTA image name in your OpenStack env]
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used
    default: [Replace with flavor that suits vTA]
  vTA_network:
    type: string
    label: vTA network attach
    default: [Replace with network that vTA attaches to]
    description: vTA test interface attached to this network
resources:
  vTA:
    type: OS::Nova::Server
    properties:
      key_name: Operations
```

```

image: { get_param: image_id }
flavor: { get_param: instance_type }
networks:
- network: management
- network: { get_param: vTA_network }
user_data_format: RAW
user_data:
    str_replace:
        template: |
            email: $email
            password: $password
            account: $account_name
            server: login.netrounds.com:443
            name: $vTA_name
        params:
            $email: { get_param: email }
            $password: { get_param: account_password }
            $account_name: { get_param: account_name }
            $vTA_name: { get_param: vTA_name }

```

## Launching a vTA Image Using the Python OpenStack API

It is possible to use the OpenStack Nova and Keystone Python APIs to automatically launch the vTA image from a Python script:

```

def create_instance(net, vta_name, os_password):
    #connect using credentials
    creds = get_nova_creds(os_password)
    nova = nvclient.Client(**creds)

    #get image, flavor, network/nics
    image = nova.images.find(name="vTA_cloudinit_image")
    flavor = nova.flavors.find(name="netrounds.small.2GB")

    network = nova.networks.find(label=net)
    nics = [{'netid': network.id}]

```

```
# Create instances; requires cloudinit on vTA images
instance = nova.servers.create(name=vta_name, image=image,\
    flavor=flavor, key_name="default", nics=nics,\
    userdata=open("./userdata.txt"))
```

The cloud-init file (user-data) in “userdata.txt” above has this content:

```
#cloud-config
netrounds_test_agent:
email: [email you use when logging in to Paragon Active Assurance]
password: [Paragon Active Assurance login password]
account: [Paragon Active Assurance account]
server: [Control Center address; for SaaS: login.netrounds.com:443]
name: [Name of vTA to appear in Control Center inventory]
```

Note again that the **#cloud-config** and **netrounds\_test\_agent** lines must be present.

Please contact Juniper Network support at <https://support.juniper.net/support/requesting-support> to get example Python code.

## Appendix: Description of the vTA VNF and Its Requirements

1. The vTA VNF is capable of running in a plain, “vanilla” environment using a standard cloud configuration and orchestration based on OpenStack. There might be some limitations in terms of performance and also some minor limitations in terms of jitter and delay accuracy, depending on your OpenStack infrastructure and how heavily loaded it is. However, for early proof-of-concepts and evaluations, this should not be a major issue. To obtain line rate packet generation and optimal usage of your specific hypervisor environment, an integration project would be required.
2. The vTA VNF consists of a single stand-alone VNF. However, the VNF must be able to connect and communicate securely with Control Center, which is not a VNF. Control Center is readily available in the public cloud (in addition to private cloud installations), something which simplifies test and evaluation projects.
3. Interfaces trust the natural OS bootstrap order in terms of how they are identified.

4. The performance is dependent on the underlying hardware. The more powerful the hardware, the higher the performance. For a 3 GHz quad-core processor, achievable performance is up to 10 Gbit/s using five concurrent unidirectional TCP streams.
5. The minimum recommended specification is: 1 vCPU, 512 MB RAM, and 2 GB of block storage.
6. It is assumed that a generic VNF manager which is not part of the Paragon Active Assurance solution does the instantiation, scaling, and termination of the vTA VNF.
7. The vTA VNF needs to register with Control Center to receive commands. For public cloud Control Center scenarios, the VNF needs connectivity to the Internet from the eth0 interface. For plug-and-play configuration of the VNF, DHCP should be used for IP addressing of the vTA's interfaces, as well as for assignment of an available DNS server.
8. The VNF will resolve the Control Center address and initiate an outbound connection using TCP. To successfully connect and authenticate itself to the correct Paragon Active Assurance account, the VNF needs to have credentials provided to it during initialization. The VNF supports cloud-init and config-drive for this purpose for its day-zero configuration. Once the VNF has connected to Control Center, it can be controlled either via a web browser or through the Paragon Active Assurance cloud API to start monitoring user experience KPIs, conduct a service turn-up test, or perform on-demand troubleshooting tests. The connection is an encrypted OpenVPN connection.
9. The vTA VNF also requires synchronization to an NTP server in order to achieve accurate delay and jitter measurements. By default, Test Agents will synchronize their internal clock to time.google.com, a service provided by Google; however, any NTP server (internal or external) can be used.
10. Rescaling of the VNF again needs to be handled by a generic VNF manager (compare paragraph "6" on [page 13](#)). For example, if the available connection bandwidth is increased, the VNF might need to be scaled up to be able to push enough bandwidth through the link for testing purposes.