AVI
Networks®

Avi Deployment Guide for Google Cloud Platform (GCP)

Avi Technical Reference (v17.2)

Avi Deployment Guide for Google Cloud Platform (GCP) view online

This article describes the process of provisioning and configuring Avi Vantage with Google Cloud Platform (GCP).

I. Introduction

About Google Cloud Platform (GCP)

Google Cloud Platform is a cloud computing service that offers hosting on the same supporting infrastructure Google uses internally for end-user products such as Google Search and YouTube. Cloud Platform provides developer products to build a range of programs from simple websites to complex applications.

Google Cloud Platform is a part of a suite of enterprise services from Google Cloud and provides a set of modular cloud-based services with a host of development tools, including hosting and computing, cloud storage, data storage, translation APIs and prediction APIs. A sample deployment case would look as below.

Source: https://cloud.google.com/docs/

About Avi Vantage

The Avi Vantage Platform provides enterprise-grade distributed ADC solutions for on-premises as well as public-cloud infrastructure. Avi Vantage also provides built-in analytics to diagnose and improve the end-user application experience, while making operationalizing easier for network administrators.

Avi Vantage is a complete software solution which runs on commodity x86 servers or as a virtual machine and is entirely accessible via REST API calls.

Purpose of this Guide

Avi Vantage serves as an application delivery controller for application workloads running on Google Cloud Platform (GCP). The purpose of this document is to describe the process of provisioning and configuring an Avi Vantage solution version 17.1.x.

Intended Audience

The document is intended for

- GCP system administrators: To provision the Avi Vantage solution
- Network administrators: To configure/operationalize the Avi Vantage solution

We assume familiarity with:

- The basics of load balancing and application delivery
- Basic GCP functionality (find more information here: https://cloud.google.com/docs/)

Scope

Avi Vantage for GCP provides functionality as described below.

- VMs are created using standard Google images (e.g., CentOS 7.5). Note: CentOS 7.5 is supported on Avi Vantage starting with release 17.2.12.

- The Avi Controller and Avi SEs run as Docker containers.
    - The CentOS image for the base VM is available in the Google repository.
- Only Linux Server Cloud is supported.
- There is a single interface on the SE for control and data traffic.
- VIP addresses are manually configured or allocated from an Avi-managed static pool.
    - The VIP address or SNAT addresses cannot be in the same subnet as the interface.
- The service account authentication mechanism is used.
    - Privilege is inherited by virtue of getting spawned by an authenticated entity through API calls.
    - The Controller instance should be spawned with read-write scope, while SEs are spawned with a read-only scope.
- The only Controller interaction with the Google API is to add a ?route? to the VIP via the instance. The Controller uses query API calls as well. The Controller also interacts with the Google Cloud Platform to program the routes.
- For SE high availability, only elastic HA modes are supported for SEs. On members of the Avi Vantage 16.x release family, this is based on L3 ECMP. When upgrading to a member of the 17.x family, if ECMP had been enabled (Advanced Options in Virtual Service Properties), it should be disabled, since Avi Vantage 17.x implements the functionality internally within the code.
- VMs are created with a single interface and are assigned an IP address with a /32 mask by GCE (Google Cloud Environment) from its internal subnet.
- The GCP Avi Controller instance needs to have Internet access for a GCP-based Linux server cloud to work. GCP instances get Internet access only when they have an external IP address attached, or the instance is connected to a network (via VPN) that has Internet access. A GCP VPN screen will appear as shown below:

Limitations: * ?Legacy? networking mode is not supported. * Floating IPs in Google Cloud Platform are not supported as of the date of this writing but are on the Avi Vantage roadmap.

II. Provisioning Avi Vantage in GCP

1. Setting up network, subnet, and instances in Google Cloud

- Browse to the Google Cloud Platform Console via https://console.cloud.google.com.
- Navigate to the respective project to which you have been subscribed.

- Navigate to Google Cloud Platform and click Networking

- Under the Networking tab, click VPC networks and then click CREATE VPC NETWORK

- Provide a name to the VPC network that you want to create, an appropriate region and the IP address range the network is to have. This can be only an IPv4 address range, as GCP only supports IPv4 and not IPv6. Click Create.

- After clicking Create, the network is created as below.

---

2. Setting up firewall rules in Google Cloud

- The default behavior of GCP is to drop traffic. Therefore, firewall rules are needed. Protocol ports are used by Avi Vantage for management communication as described in [Protocol Ports Used by Avi Vantage for Management Communication.]()
- Create a firewall rule to allow TCP & UDP & ICMP traffic within the network and HTTP/HTTPS from outside under the respective network created as above. The one shown below is for TCP and UDP. After filling in the form, click Save.

- Firewall rule for TCP on port 80 and 443. After filling in the form, click Save.

- Firewall rule for ICMP. After filling in the form, click Save.

- Firewall rules for internal SE-to-SE communication.

- The firewall rules created are depicted below.

3. Avi Controller, Avi Service Engine, server and client instantiation in Google Cloud

- Navigate to the Google Cloud Platform icon and click Compute Engine.

- Avi Controller: Under Compute Engine, click CREATE INSTANCE.

- Create an Avi Controller instance as below:
  - Name the instance.
  - Provide the zone in which this instance should be created.
  - Select machine type n1-standard-4 for 4 vCPUs and 15 GB of memory. Sizing may vary depending on your scaling requirements.

- Select a boot disk with a CentOS 7 image and with an 80-GB boot disk size.

- Click Identity and API access to make sure that the scope is set to Read Write. Do the same by selecting the Compute Engine parameter to Read Write (click Set access for each API) for GCP route programming.

- Tags for HTTP and HTTPS should be enabled to permit outside connections. Under the Networking tab, make sure that the Network and Subnetwork fields are filled in. For the Avi Controller instance, IP forwarding should be set to Off. Click Create.

- The tag for internal SE-to-SE communication has to be added here. It comes from the one created during firewall rules creation, as described in a previous section of this document.

- Copy the public keys from the machine from which the SSH will be attempted (explained above in detail while creating the Controller instance).

- After the Avi Controller is created you can see that it is up with an external IP address and an internal IP address from the network range that was specified while creating networks.

- Google Cloud Platform does not allow serial console access to the created instance if an external IP is not allocated.   Serial console access is not required for installation or operation of Avi Vantage software, but may be useful for troubleshooting.
- The same steps need to be repeated to create another Service Engine. For high availability purposes, we can create the SEs in different zones, as can be seen below.

The above steps complete the installation of Avi Vantage software on the GCP platform. To run test traffic, create a test server and client instances as described below.

- Server Instance: Under Compute Engine, click the CREATE INSTANCE tab.

- Create one or more test server instances as below.
- Name the instance.
- Provide the zone in which the server will be created.
- Selecting a Machine type of small in the pulldown menu implies 1 vCPU and 1.7 GB of memory

* Select a boot disk with a CentOS 7 image and a 20-GB capacity.

- Click Identity and API access to make sure that the scope is set to Read Only. Do so by setting the Compute Engine parameter to Read Only (click Set access for each API) for GCP route programming.

- Tags for HTTP should be enabled to allow outside connections. Under Network interfaces, ensure the Network and Subnetwork fields are filled in. For a server instance, IP forwarding should be set to On. Click Create.

- Copy the public keys from the machine from which the SSH will be attempted (explained above in detail while creating the Controller instance).

- Client Instance: Under Compute Engine:VM instances, click CREATE INSTANCE.

- Create one or more test client instances as below.
- Select an appropriate name.
- Deploy it in a respective zone.
- Selecting a Machine type of small in the pulldown menu implies 1 vCPU and 1.7 GB of memory.

- Select a boot disk with a CentOS 7 image and 20-GB capacity.

- Click on Identity and API access to ensure that the scope is set to Read Only. Do so by selecting the Compute Engine to Read Only (click on Set access for each API) for GCP route programming.

- Tags for HTTP should be enabled. Under networking ensure the Network and Subnetwork fields are filled in. For a client instance, IP forwarding should be set On. Click Create.

- Copy the public keys from the machine from which the SSH will be attempted (explained above in detail while creating Controller instance).

- Verify all the instances are created, as shown below.

  #### 4. Preparing the instances

- Turning off yum-cron For this, the instance needs to be on CentOS 7.3.
  - Note: CentOS 7.3 support comes with Avi Vantage 16.3.4.
- Installing Docker
  - To configure a docker repository, create the file docker.repo under /etc/yum.repos.d/

```
[localhost@avi-controller ~]$ sudo vim docker.repo
[docker-main]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7/
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
```

  - Verify the instances are running on CentOS 7.3.

```
[localhost@avi-controller ~]$ cat /etc/centos-release
CentOS Linux release 7.3.1611 (Core)
```

- Install & start Docker on all 5 instances.

```
sudo yum update -y
sudo yum install -y docker
sudo systemctl enable docker
sudo systemctl start docker
```

To remain on the current release of CentOS/RHEL, lock the Linux System to the specific OS version. To know more, refer to [Locking a Linux System to a Specific OS Version.]()
Note: Doing so on releases lower than 7.2 may require additional effort to install corresponding versions of Docker supported for the particular release.

- It is recommended to use `devicemapper` with thin-pool as the recommended storage driver for production. As shown below, `devicemapper` is configured with loopback. This is suitable for a proof of concept, but not for a production environment.

```
[localhost@avi-controller ~]$ sudo docker info
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 1.10.3
Storage Driver: devicemapper
Pool Name: docker-8:1-67109509-pool
Pool Blocksize: 65.54 kB
Base Device Size: 10.74 GB
Backing Filesystem: xfs
Data file: /dev/loop0
Metadata file: /dev/loop1
Data Space Used: 11.8 MB
Data Space Total: 107.4 GB
Data Space Available: 19.92 GB
Metadata Space Used: 581.6 kB
Metadata Space Total: 2.147 GB
Metadata Space Available: 2.147 GB
Udev Sync Supported: true
Deferred Removal Enabled: false
Deferred Deletion Enabled: false
Deferred Deleted Device Count: 0
Data loop file: /var/lib/docker/devicemapper/devicemapper/data
WARNING: Usage of loopback devices is strongly discouraged for production use. Either use `--storage-opt
Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata
Library Version: 1.02.135-RHEL7 (2016-09-28)
```

```
Execution Driver: native-0.2
Logging Driver: journald
Plugins:
Volume: local
Network: null host bridge
Kernel Version: 3.10.0-514.2.2.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
Number of Docker Hooks: 2
CPUs: 4
Total Memory: 14.69 GiB
Name: test.c.astral-chassis-136417.internal
ID: TOWE:AFZ3:JHJ4:C4A5:PFAI:MF2J:2HKE:ZQLM:LREW:WYNG:UU4C:NBP2
Registries: docker.io (secure)
```

- If the instance is spawned for Ubuntu, `sshguard` must be configured (explained in detail in a section below). sshguard can take whitelists from files when the -w option argument begins with a '.' (dot) or '/' (slash). Below is `/etc/list`, a sample whitelist file.

```
# comment line (a '#' as very first character)
# a single IPv4 and IPv6 address
1.2.3.4
2001:0db8:85a3:08d3:1319:8a2e:0370:7344
#   address blocks in CIDR notation
127.0.0.0/8
10.11.128.0/17
192.168.0.0/24
2002:836b:4179::836b:0000/126
#   hostnames
rome-fw.enterprise.com
hosts.test.com
```

Below we see the `sshguard` referencing the test file.

```
sshguard -w /etc/test
```

5. Test server on the server instance

- Start an NGINX server on the server instance to use as a pool server.

```
sudo docker run -d -p 80:80 avinetworks/server
```

III. Avi Vantage Configuration

1. Installation and configuration of Avi Controller & Avi Service Engines ? Method 1

- Follow instructions outlined in Installing Avi Vantage for a Linux Server Cloud to install/run the Avi Controller on the instance previously created.
- Before proceeding to the next step, in the GUI, ensure the SE status is Green.

- In the cloud configuration, ensure DPDK mode is disabled for the Linux server cloud deployed per the steps documented in the above-mentioned article. In addition, ensure in-band management is enabled.

- `ssh` into Avi Controller instance, exec to the Controller and start the Avi `shell`. Type this command to list the container id to be used.

```
sudo docker ps
```

```
sudo docker exec -it [container_id] bash
shell
```

- Create a network with an IP address pool for VIP allocation. In the Avi UI, browse to Infrastructure -> Networks -> Create*[]:

- Create an IPAM Profile for GCP.

- Edit ?Default-Cloud.? Choose the previously created GCP IPAM provider ?gcp? as IPAM provider and configure a Linux server cloud using IP addresses for the two Avi Service Engine instances created above.

Note: For more information on IPAM provider, read IPAM Provider (Google Cloud Platform)

2. Installation and Configuration of Avi Controller & Avi Service Engines ? Method 2

Alternately, if the Controller service file is already created, fresh start a Controller with a `setup.json` file to configure a Linux server cloud with a GCP IPAM profile and a network for VIP allocation.

- Copy the `setup.json` file shown below to `/opt/avi/controller/data` on the host (assuming `/opt/avi/controller/data` is the volume used for the Controller in the service file).
- Modify ssh keys, username, network subnets and network/IPAM names as appropriate.

```
{
   "CloudConnectorUser": [
   {
       "name": "rangar",
       "tenant_ref": "admin",
       "public_key": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC9ZZWDLSl/PJHWA8QuDlfDHJuFh6k55qxRxO28fSRvAEbWCXXgX
Mo0brgqrp+vful2m7hNm0TPv8REbT2luVeWo+G0R1hxzdALzI8VmMBxX2VduKZ5Zrh3C9GKxaUYb4R2hzLaYKUBQnFa2B0YWiAfC3ow71fwwgb7
9Tcb3w9uugv3vXNzyxDssHXtwY60WcVUIK1L+8SqXu/r6YUG8j4IsaYkXJHBE6CHPwDg4uwRG35IkfhsIg0KtKRwpzHbhOx0qRjG9ZaVc0SnfMI
NAmjkix2GIPIi1OISnEngSjnugVb7\n",
       "private_key": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIBAAKCAQEAvWWVgy0pfzyR1gPELg5XwxybhYepOeasUcTtvH0
n4F3Zx/L0lVQxDKNG64Kq6fr37pdpu4TZtEz7/ERG09pblXlqPhtEdYcc3QC8yPFZ\njAcV9lXbimeWa4dwvRisWlGG+Edocy2mClAUJxWtgdGF
3FYcR\nMU8oYRdNYGPU3G98PbroL971zc8sQ7LB17cGOtFnFVCCtS/vEql7v6+mFBvI+CLG\nmJFyRwROghz8A4OLsERt+SJH4bCINCrSkcKcx2
p3zCB5nQBcF\n6Q6YvwClTQJo5IsdhiDyItTiEpxJ4Eo57oFW+wIDAQABAoIBAFu7XeUA9L5ZmdDs\nVhJwg/VOX80W3dHbdc7M8NCAVCsnGSgF
A8NKUoS9ejMMUtvNJ\n7x+ywcF3WE63ze/htKGMF2ZNIJ+yAb3Zl6OIswxynTi131cJbINJ9gBwyExsWZyf\nmXIZQwmDKFxeHLlQ80QeR9qDxF
3JQJMbUX6dmQm0UtOKi5tL8\nzkskZJHnaqwJlem92Zon7S8PIflsPevsAmDrTPbmxIL6Z3KlJkoLzTcWefA6E19N\nw4JmylQokAWiqQ1il+qr
wjHkzzA8kdb4EUO0nhy7rzbmS67TN\n08Fe0RECgYEA98WaJR5k/r8VBlKEQTErye29cJmkr0w5ZPX+bwko+ejj2S2vqpJc\nuR0YO3q5zY5a4A
p9QSnBscFvA/AEXGAiAeuCsuB+pw8C3N5C5\ncTzKNFx1c2KXbejRkhvL9gz5tJZpdHIqzbGQmwEiNFqnYy6BPbhTm8UCgYEAw6+2\n5WvAGH9U
DvXA2+G/CBg99KYuXzWWmeVx9652lc4Gv+mxhFiJd\nilMfWljlb+f1G5sJnZ3VMKSf/FF??6Mo8MsnAkvjnVWBoezo2sVzu+9g3qGRXNTtRM\n
D3r8K+iag7cMhrLpGPWk78CgYARatumJlfVLJuOwTg42PsK\nC+NYSgSwqfwS49QJ/CvcPYne135U0EsiXDA65iqvj4VF4Pl8oaS2rpF2yU8dqG
/fYGCoc6idt9ZOm/mwQ64LhzMx38eKF0axdYNnlSdLFZVYolxPSFT\nKltO+ipsYb8IktlU/GMsPQKBgQCeirlqzM64yki11Hcce3Q3qQ3QqGih
L37mnSy9N3MTFAk8hiKks5h6XvRuyC2yTkyXkL2l7jFq39zRp2cBsMzPTSz\nSSpruF2CYL8+6AeOMYi4v3M/2asaR+R6ApNytk90Bs0XQ/V6qc
nOYo67wKBgAcUFRHUX4VwCUZAAIxyTM+efpf5z8dKHh/iJA6rtqcTi4vHddEJinT6\ntOiqXjciZEKqZ08GtImIPtuhIBO0m10fCfcjrGxGz2+N
IXSq8\nU1YOIYvXwWFQLWIUvyOgnyT4bW0OLa8OrJEq1/DaH8gpvvFi8qRK\n-----END RSA PRIVATE KEY-----\n"
   }
   ],
   "IpamDnsProviderProfile": [
   {
       "name": "gcp",
       "type": "IPAMDNS_TYPE_GCP",
       "tenant_ref": "admin",
       "gcp_profile": {
           "usable_network_refs": [
               "/api/network/?name=net1"
           ]
       }
   }
   ],
   "Network": [
   {
       "name": "net1",
       "tenant_ref": "admin",
       "cloud_ref": "admin:Default-Cloud",
       "configured_subnets": [
           {
               "prefix": {
                   "ip_addr": {
                       "type": "V4",
                       "addr": "10.9.0.0"
                   },
                   "mask": 24
               },
               "static_ranges": [
                   {
                       "begin": {
                           "type": "V4",
                           "addr": "10.9.0.2"
                       },
                       "end": {
                           "type": "V4",
                           "addr": "10.9.0.254"
                       }
                   }
               ]
           }
       ]
   }
   ],
   "SeProperties": [
   {
       "se_runtime_properties": {
           "global_mtu": 1400,
           "se_handle_interface_routes": true
       }
   }
   ],
   "Cloud": [
   {
       "name": "Default-Cloud",
       "tenant_ref": "admin",
       "vtype": "CLOUD_LINUXSERVER",
       "ipam_provider_ref": "admin:gcp",
       "linuxserver_configuration": {
           "ssh_attr": {
               "ssh_user": "rangar",
               "host_os": "COREOS"
           },
           "se_sys_disk_path": "/?
       }
   }
   ]
}
```

- Perform first setup on the Controller & specify a username/password.
- Edit ?Default-Cloud,? choosing the previously created GCP IPAM provider ?gcp? as IPAM provider and configure a Linux server cloud using IP addresses for the two Avi Service Engine instances created above.

IV. VS Creation and Traffic Verification

- Create a pool, e.g., GCP-Perf-Test-VS-Pool.
- Add server instance IP as pool server.
- Create an internal virtual service called ?GCP-Perf-Test-VS? by clicking to the Advanced tab.
- The VIP is auto-allocated from the VIP/IPAM subnet 10.y.y.y./24
- Use placement subnet as net1-subnet4 - 10.x.x.x/24
  Note: IP subnet *10.x.x.x* is mentioned only for reference purpose. Placement subnet should be set to the major subnet in the VPC used by Avi Controller and Service Engines.
- After the VS is created (and a VIP is allocated), use VIP as SNAT IP, if desired.

1. Pool Creation

---

2. Virtual Service Creation

Below are screenshots taken after creating the new virtual service:

---

3. ICMP Traffic

Send ICMP traffic to the VIP IP 10.10.0.1 in this case and make sure that it gets programmed.

```
[localhost@avi-test-server ~]$ ping 10.10.0.1

PING 10.10.0.1 (10.10.0.1) 56(84) bytes of data.
64 bytes from 10.10.0.1: icmp_seq=1 ttl=64 time=0.889 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=64 time=0.278 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=64 time=0.291 ms
64 bytes from 10.10.0.1: icmp_seq=4 ttl=64 time=0.288 ms
64 bytes from 10.10.0.1: icmp_seq=5 ttl=64 time=0.294 ms
64 bytes from 10.10.0.: icmp_seq=6 ttl=64 time=0.302 ms
64 bytes from 10.10.0.1: icmp_seq=7 ttl=64 time=0.320 ms
64 bytes from 10.10.0.1: icmp_seq=8 ttl=64 time=0.257 ms
64 bytes from 10.10.0.1: icmp_seq=9 ttl=64 time=0.315 ms
```

- Verify that a route for the VIP/32 is programmed in GCP with nextHop as Service Engine 1 with IP 10.8.2.3, as can be seen below with the notation.

```
<avi-route-CCID ( Cloud Connector ID)-target-next hop/32>
<avi-route-aa03f67634a6-10-8-2-3-10-10-0-2-32>
```

---

4. API for Configuration of Virtual Service and Pool

Copy the `setup.json` file shown below to `/opt/avi/controller/data` on the host (assuming `/opt/avi/controller/data` is the volume used for the Controller in the service file).

```
{
    "name": "vs1",
    "pool_ref": "pool_ref",
    "services": [
        {
            "port": 80
        }
    ],
    "vip": [
        {
            "auto_allocate_ip": true,
            "ipam_network_subnet": {
                "network_ref": "network_ref",
                "subnet": {
                    "ip_addr": {
                        "addr": "6.2.0.0",--> IPAM subnet.
                        "type": "V4"
                    },
                    "mask": 16
                }
            },
            "subnet": {
                "ip_addr": {
                    "addr": "10.146.11.0", --> placement subnet, subnet having reachability to client facing VIP
                    "type": "V4"
                },
                "mask": 24
            }
        }
    ]
}
```

V. Functional Troubleshooting

1. Issue

Service Engine fails to connect to the Controller or frequently loses connectivity on Ubuntu.

2. Root Cause

- This is due to `sshguard`, which is documented here: http://www.sshguard.net/docs/
- `sshguard` protects hosts from brute-force attacks against SSH and other services. It aggregates system logs and blocks repeat offenders using one of several firewall backends, including iptables, ipfw, and pf.
- `sshguard` can read log messages from standard input (suitable for piping from `syslog`) or monitor one or more log files. Log messages are parsed, line-by-line, for recognized patterns. If an attack, such as several login failures within a few seconds, is detected, the offending IP is blocked. Offenders are unblocked after a set interval, but can be semi-permanently banned using the blacklist option.
- `sshguard` supports address whitelisting. Whitelisted addresses are not blocked, even if they appear to generate attacks. This is useful for protecting lame LAN users (or external friendly users) from being incidentally blocked.
- When longer lists are needed for whitelisting, they can be wrapped into a plain text file, one address/hostname/block per line.

3. Mitigation

Configure the Controller IP (each of the 3 if clustered) in the whitelist file used by `sshguard`.

`sshguard` can take whitelists from files when the -w option argument begins with a '.' (dot) or '/' (slash). Below is a sample whitelist file (/etc/test), with comment lines denoted by a '#' as the very first character.

```
#   a single IPv4 and IPv6 address
      1.2.3.4
      2001:0db8:85a3:08d3:1319:8a2e:0370:7344
      #   address blocks in CIDR notation
      127.0.0.0/8
      10.11.128.0/17
      192.168.0.0/24
      2002:836b:4179::836b:0000/126
      #   hostnames
      rome-fw.enterprise.com
      hosts.test.com
```

`sshguard` is told to make a whitelist up from the /etc/test file as follows:

```
sshguard -w /etc/test
```