Avi Ansible Integration Documentation

Release 17.1.0

Avi Networks

Apr 25, 2020

Contents

CHAPTER 1

Avi Controller Role

To help automate the deployment of Avi Vantage Controller in your environment we've developed an Ansible Role which can deploy in many environemnts. This includes deploying from Docker Hub, Private Docker Repo, Docker compressed images (tgz), as well as some cloud environments including CSP deployment.

1.1 Prerequisites

To get started you will need the following:

- Ansible 2.2 or higher
- Server or Cloud to install Avi Controller on (Baremetal, CSP, etc.)

1.2 Parameters

Ansible Role can take parameters, these parameters can determine where and how Ansible will execute the installation and configuration of the Avi Vantage Controller.

Due to the fast pace of features and parameters that get added we will not include them here directly, but they can be accessed in two places. In our public GitHub repository (https://github.com/avinetworks/ansible-role-avicontroller) or in the Ansible Galaxy (https://galaxy.ansible.com/avinetworks/avicontroller/). The readme includes all possible parameters.

1.3 Deploying Avi Controller

1.3.1 Bare-metal Avi Deployment

Below are the current methods of deployment supported by this role to deploy the Avi Controller. Please select from the options below to learn how to deploy using each configuration.

Using Docker Hub

Deploying Avi Controller from the Docker Hub is the default action. However it does require the server to have internet access to Docker Hub as it will download the image.

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this.

Note: You can use `con_version` to specify which Avi release you'd like to use. By default it uses whatever version referenced in the README.md file, and the version of the Role.

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an SSH key you will also need to specify `--ask-pass` to Ansible.
- If your current user is different you will need to specify `-u <username>` to Ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Private Docker Repository

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_docker_repo: mydockerrepo.company.com
      con_version: 16.3.9-23923929323923
      con_docker_repo_user: admin
      con_docker_repo_password: adminpassword
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an SSH key you will also need to specify `--ask-pass` to ansible.
- If your current user is different you will need to specify `-u <username>` to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Compressed Docker Image

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_package_deploy: true
      con_package_source: /home/user/Downloads/controller_docker.tgz
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more. We also needed to specify that we want to deploy from package via `con_package_deploy: true` which tells the role we're deploying from package. Then we provided the location of the package file by providing the `con_package_source` parameter.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an SSH key you will also need to specify `--ask-pass` to ansible.
- If your current user is different you will need to specify `-u <username>` to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

Preparing Your Host

Prior to delpoying this role on your servers you may want to also configure the server for your environment. Some tasks you may want to include:

- configure network interfaces
- install Docker, either with another role, or manually from the Docker website
- turn off built-in firewalls (firewalld, ufw, etc.)
- turn off yum-cron if you're using CentOS, Oracle, or RedHat Linux
- subscribe your RedHat host to the Red Hat Subscription Manager
- deploying users, ssh-keys, etc.
- registering the host with another internal tool for tracking or monitoring

There are tons of things I am sure not listed that you may think of, but should be included in preparing a host for deploying the Controller to a baremetal host.

We also want to make sure you don't have multiple items trying to manage this server at the same time. If you must then make sure all entities managing the server aren't managing the same components. You could end up with a conflict where one tool is telling the server to meet one spec, and another is changing it back, and then you have both tools changing each others actions back. (Not good for consistency)

1.3.2 Cisco CSP Avi Deployment

Requirements

- You will need to have available memory and storage for both the image, and the service on your Cisco CSP device.

Using QCOW image

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy install -f avinetworks.avicontroller-csp
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[csp_devices]
10.120.222.56
```

   Replace the IP with the IP of your CSP host you want to deploy to.

3. Make sure you are able to SSH into the CSP device. To verify login, then logout.

```
ssh user@10.120.222.56
```

4. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: csp_devices
  gather_facts: false
  roles:
    - role: avinetworks.avicontroller
      con_deploy_type: csp
      con_csp_user: admin
      con_csp_password: password
      con_csp_qcow_image_file: avi-controller.qcow2
      con_csp_mgmt_ip: 10.128.2.20
      con_csp_mgmt_mask: 255.255.255.0
      con_csp_default_gw: 10.128.2.1
      con_csp_service_name: avi-controller
      con_csp_num_cpu: 4
      con_csp_memory_gb: 16
```

con_deploy_type: Sets the type of deployment that should be triggered.

con_csp_user: Username that will be used to connect to the CSP server

con_csp_password: Password required to authenticate the user

con_csp_qcow_image_file: is the relative, or absolute location of the qcow file that will be uploaded to the CSP device.

con_csp_mgmt_ip: IP of the controller on the management network.

con_csp_mgmt_mask: Subnet mask that the controller will require.

con_csp_default_gw: Default gateway for the controller

con_csp_service_name: Name of the service to be created on the CSP

con_csp_num_cpu: Number of CPUs to be allocated to the Controller

con_csp_memory_gb: Amount of memory in GB allocated to the Controller

Note: There are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

In this playbook we are specifying that this "play" should apply to the "csp_devices" hosts. Anything not required mentioned here CSP Deployment Variables can be omitted from your role parameters. Defaults are also noted on that document.

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

1.3.3 OpenShift Avi Deployment

Using Docker Hub

Deploying Avi Controller from the Docker Hub is the default action. However it does require the server to have internet access to Docker Hub as it will download the image.

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_controller_ip: 10.120.202.24
      con_deploy_type: openshift
      con_cores: 4
      con_memory_gb: 16
```

However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Private Docker Repository

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_docker_repo: mydockerrepo.company.com
      con_version: 16.3.9-23923929323923
      con_docker_repo_user: admin
      con_docker_repo_password: adminpassword
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Compressed Docker Image

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_package_deploy: true
      con_package_source: /home/user/Downloads/controller_docker.tgz
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more. We also needed to specify that we want to deploy from package via `con_package_deploy: true` which tells the role we're deploying from package. Then we provided the location of the package file by providing the `con_package_source` parameter.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Requirements

- CentOS/RHEL/OracleLinux 7.x

- Ubuntu 14.04 or higher
- Docker 1.12 or higher

The main components of the Avi Vantage solution, Avi Controllers and Service Engines (SEs), run as containers on OpenShift/Kubernetes minion nodes. For production deployment, a 3-instance Avi Controller cluster is recommended, with the each of the Avi Controller instances running in containers on separate physical nodes. After configuring the Avi Controller cluster for OpensShift/Kubernetes cloud, it deploys one Avi SE container on OpenShift/Kubernetes nodes.

- The system time on all nodes must be synchronized. Use of a Network Time Protocol (NTP) server is recommended.
- The Avi Controller uses password-less sudo SSH to access all the OpenShift nodes in the cluster and create SEs on those nodes. The SSH user must have password-less sudo access to all three OpenShift nodes hosting the Avi Vantage cluster. The SSH method requires a public-private key pair. You can import an existing private key onto the Avi Controller or generate a new key pair. In either case, the public key must be in the "/home/ssh_user/.ssh/authorized_keys" file, where ssh_user is the SSH username on all OpenShift nodes. The Avi Controller setup wizard automatically stores the private key on the Avi Controller node when you import or generate the key.

1.3.4 AWS Avi Deployment

1.3.5 OpenStack Avi Deployment

CHAPTER 2

Avi SE Role

To help automate the deployment of Avi Vantage Service Engine in your environment we've developed an Ansible Role which can deploy in many environemnts. This includes deploying from Docker Hub, Private Docker Repo, Docker compressed images (tgz), as well as some cloud environments including CSP deployment.

2.1 Prerequisites

To get started you will need the following:

- Ansible 2.2 or higher
- Server or Cloud to install Avi Controller on (Baremetal, CSP, etc.)
- Pre-existing Avi Controller

2.2 Parameters

Ansible Role can take parameters, these parameters can determine where and how Ansible will execute the installation and configuration of the Avi Vantage Service Engine.

Due to the fast pace of features and parameters that get added we will not include them here directly, but they can be accessed in two places. In our public GitHub repository (https://github.com/avinetworks/ansible-role-avise) or in the Ansible Galaxy (https://galaxy.ansible.com/avinetworks/avise/). The readme includes all possible parameters.

2.3 Deploying Avi SE

2.3.1 Bare-metal Avi Deployment

Below are the current methods of deployment supported by this role to deploy the Avi SE. Please select from the options below to learn how to deploy using each configuration.

Using Docker Hub

Deploying Avi SE from the Docker Hub is the default action. However it does require the server to have internet access to Docker Hub as it will download the image.

Steps

1. Install the Avi SE role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avise
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[service_engines]
10.120.202.56
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: service_engines
  become: true
  roles:
    - role: avinetworks.avise
      se_master_ctl_ip: 10.120.202.24
      se_master_ctl_username: admin
      se_master_ctl_password: avi123
      se_disk_gb: 60
      se_cores: 2
      se_memory_gb: 4
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avise). Each one allows customization of the deployment.

   We will automatically use the `se_master_ctl_ip`, `se_master_ctl_username`, and `se_master_ctl_password` to authenticate, and autoregister the service engine to the controllers default cloud. To choose which cloud you want please use the `se_cloud_name` value. If you use multiple tenants please use `se_tenant` to choose the specific tenant the cloud is under.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.56
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Private Docker Repository

Steps

1. Install the Avi SE role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avise
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[service_engines]
10.120.202.56
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: service_engines
  become: true
  roles:
    - role: avinetworks.avise
      se_docker_repo: mydockerrepo.company.com
      se_version: 16.3.9-23923929323923
      se_docker_repo_user: admin
      se_docker_repo_password: adminpassword
      se_master_ctl_ip: 10.120.202.24
      se_master_ctl_username: admin
      se_master_ctl_password: avi123
      se_disk_gb: 60
      se_cores: 2
      se_memory_gb: 4
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avise). Each one allows customization of the deployment.

   We will automatically use the `se_master_ctl_ip`, `se_master_ctl_username`, and `se_master_ctl_password` to authenticate, and autoregister the service engine to the controllers default cloud. To choose which cloud you want please use the `se_cloud_name` value. If you use multiple tenants please use `se_tenant` to choose the specific tenant the cloud is under.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.56
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Compressed Docker Image

Steps

1. Install the Avi SE role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avise
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[service_engines]
10.120.202.56
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: service_engines
  become: true
  roles:
    - role: avinetworks.avise
      se_package_deploy: true
      se_package_source: /home/user/Downloads/se_docker.tgz
      se_master_ctl_ip: 10.120.202.24
      se_master_ctl_username: admin
      se_master_ctl_password: avi123
      se_disk_gb: 60
      se_cores: 2
      se_memory_gb: 4
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avise). Each one allows customization of the deployment.

   We will automatically use the `se_master_ctl_ip`, `se_master_ctl_username`, and `se_master_ctl_password` to authenticate, and autoregister the service engine to the controllers default cloud. To choose which cloud you want please use the `se_cloud_name` value. If you use multiple tenants please use `se_tenant` to choose the specific tenant the cloud is under.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.56
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

2.3.2 Cisco CSP Avi Deployment

Requirements

- You will need to have available memory and storage for both the image, and the service on your Cisco CSP device.

Using QCOW image

Steps

1. Install the Avi SE role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avise
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[csp_devices]
10.120.222.24
```

   Replace the IP with the IP of your CSP host you want to deploy to.

3. Make sure you are able to SSH into the CSP device. You can logout when done.

```
ssh user@10.120.222.24
```

4. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: csp_devices
  gather_facts: false
  roles:
    - role: avinetworks.avise
```

(continues on next page)

(continued from previous page)

```
    se_deploy_type: csp
    se_csp_user: admin
    se_csp_password: password
    se_master_ctl_ip: 10.128.2.20
    se_master_ctl_username: admin
    se_master_ctl_password: password
    se_csp_qcow_image_file: avi-se.qcow2
    se_csp_mgmt_ip: 10.128.2.20
    se_csp_mgmt_mask: 255.255.255.0
    se_csp_default_gw: 10.128.2.1
    se_csp_service_name: avi-se
    se_csp_disk_size: 10
    se_csp_num_cpu: 2
    se_csp_memory_gb: 4
    se_csp_vnics:
      - nic: "0"
        type: access
        tagged: "false"
        network_name: enp1s0f0
      - nic: 1
        type: passthrough
        passthrough_mode: sriov
        vlan: 200
        network_name: enp7s0f0
      - nic: 2
        type: passthrough
        passthrough_mode: sriov
        vlan: 201
        network_name: enp7s0f1
```

Note: There are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avise). Each one allows customization of the deployment.

In this playbook we are specifying that this "play" should apply to the "csp_devices" hosts. Anything not required mentioned here CSP Deployment Variables can be omitted from your role parameters. Defaults are also noted on that document.

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

2.3.3 OpenShift Avi Deployment

Using Docker Hub

Deploying Avi Controller from the Docker Hub is the default action. However it does require the server to have internet access to Docker Hub as it will download the image.

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Private Docker Repository

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

Steps

1. Install the Avi Controller role on the host you are executing from.

   ```
   sudo ansible-galaxy -f avinetworks.avicontroller
   ```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

   ```
   [avicontrollers]
   10.120.202.24
   ```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

   ```
   # myplaybook.yml
   ---
   - hosts: avicontrollers
     become: true
     roles:
       - role: avinetworks.avicontroller
         con_docker_repo: mydockerrepo.company.com
         con_version: 16.3.9-23923929323923
         con_docker_repo_user: admin
         con_docker_repo_password: adminpassword
         con_controller_ip: 10.120.202.24
         con_cores: 4
         con_memory_gb: 16
   ```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

   In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Using a Compressed Docker Image

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

Steps

1. Install the Avi Controller role on the host you are executing from.

```
sudo ansible-galaxy -f avinetworks.avicontroller
```

2. Identify how we want to define our hosts. Ansible can accept hosts three ways: command-line, by inventory file, or by dynamic inventory file. We will use an inventory in this case. Ansible by default uses /etc/ansible/hosts as it's default inventory. Lets add this segment to the bottom of the page.

```
[avicontrollers]
10.120.202.24
```

   Replace the IP with the IP of your baremetal host you want to deploy to.

3. Create the playbook. It should look similar to this

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.avicontroller
      con_package_deploy: true
      con_package_source: /home/user/Downloads/controller_docker.tgz
      con_controller_ip: 10.120.202.24
      con_cores: 4
      con_memory_gb: 16
```

   However, there are are many possible values which can be found on the roles README file (https://github.com/avinetworks/ansible-role-avicontroller). Each one allows customization of the deployment.

In this playbook we are specifying that this "play" should apply to the "avicontrollers" hosts. Which we have previously defined in the inventory file. We are also specifying that the controllers IP is 10.10.27.101, this is used to properly determine the interface that the controller will use. We are then specifying the core count that the controller will use. This is the amount of cores that docker will be permitted to use for the controller. Then we specify the amount of memory in GB that the controller will be permitted to use. We recommend using 16GB or more. We also needed to specify that we want to deploy from package via `con_package_deploy: true` which tells the role we're deploying from package. Then we provided the location of the package file by providing the `con_package_source` parameter.

4. Verify your local user has access to the hosts you are deploying the controller to. You will need *sudo* access as well. Login using your current user to make sure you can login. Make sure to logout when done to execute your playbook from your hose.

```
ssh 10.120.202.24
```

5. Execute the playbook.

Note:

- If you are not using an ssh-key you will also need to specify *–ask-pass* to ansible.
- If your current user is different you will need to specify *-u <username>* to ansible.

```
ansible-playbook myplaybook.yml -u <username> --ask-pass
```

Requirements

- CentOS/RHEL/OracleLinux 7.x
- Ubuntu 14.04 or higher
- Docker 1.12 or higher

The main components of the Avi Vantage solution, Avi Controllers and Service Engines (SEs), run as containers on OpenShift/Kubernetes minion nodes. For production deployment, a 3-instance Avi Controller cluster is recommended, with the each of the Avi Controller instances running in containers on separate physical nodes. After configuring the Avi Controller cluster for OpensShift/Kubernetes cloud, it deploys one Avi SE container on OpenShift/Kubernetes nodes.

- The system time on all nodes must be synchronized. Use of a Network Time Protocol (NTP) server is recommended.
- The Avi Controller uses password-less sudo SSH to access all the OpenShift nodes in the cluster and create SEs on those nodes. The SSH user must have password-less sudo access to all three OpenShift nodes hosting the Avi Vantage cluster. The SSH method requires a public-private key pair. You can import an existing private key onto the Avi Controller or generate a new key pair. In either case, the public key must be in the "/home/ssh_user/.ssh/authorized_keys" file, where ssh_user is the SSH username on all OpenShift nodes. The Avi Controller setup wizard automatically stores the private key on the Avi Controller node when you import or generate the key.

CHAPTER 3

Avi Docker Role

Avi's Docker role was created to help deploy and configure the Docker service on your server. The goal was to allow users to not only install Docker, but also configure the service itself. We allow choosing specific storage drivers, and also have automated some of the configuration required when doing so.

Docker recently has split their project into two new projects. Community Edition (CE) and Enterprise Edition (EE).

3.1 Docker Community Edition (CE)

Docker CE is the free version of Docker. It containers the full Docker platform, and is great for developers and operations teams starting to build container applications. CE uses time-based releases with a YY.MM versioning scheme. It can be enhanced by free and paid addons from the Docker Cloud.

Docker CE comes in two different variants:

- **Edge:** for users wanting to get the latest and greatest features
- **Stable:** released quarterly for users that want an easier-to-maintain release cycle

3.2 Docker Enterprise Edition (EE)

Note: At the moment we do not currenlty support deploying this version via this role.

Docker EE is the supported and certified container platform. Docker and its partners provide cooperative support for Certified Containers, and Plugins so customers can confidently use products in production.

Docker EE is available in three tiers:

- **Basic:** The Docker platform for certified infrastructure, with support from Docker Inc. and certified Containers and Plugins from Docker Store
- **Standard:** Adds advanced image and container management, LDAP/AD user integration, and role-based access control (Docker Datacenter)
- **Advanced:** Adds Docker Security Scanning and continuous vulnerability monitoring

It is available as a free trial and for purchase at Docker Store, or from the Docker Sales team. It is also supported by many Docker partners.

3.3 Example Usage

Sample Deployment on Ubuntu

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.docker
```

Sample Deployment on CentOS/RedHat/Oracle

```
# myplaybook.yml
---
- hosts: avicontrollers
  become: true
  roles:
    - role: avinetworks.docker
      docker_storage_driver: devicemapper
      docker_block_device: /dev/sda3
```

CHAPTER 4

Avi Network Interfaces Role

Using the Avi Networks Interfaces role, you are able to automatically configure a hosts network interfaces. This includes bridged interfaces, virtual interfaces, etc.

CHAPTER 5

Avi Ansible Modules

5.1 Modules Examples

5.1.1 Create Pool and Virtual Service

In this example we are going to create a pool, and then a vip. We use two of our Ansible modules: `avi_pool` and `avi_virtualservice`.

The first task we create the pool using the `avi_pool` module. Using the `name` value we are able to name the pool, then we set the `enabled` value to `false`, because at this moment we do not want to enable the pool. `health_monitor_refs` field allows us to link the pools health monitor value to an already present health monitor. Due to the structure of the API, we will need to reference the monitor by it's API name (`/api/healthmonitor?name=System-HTTP`). If the health monitor doesn't exist and you make a reference to it, you will see an API error on execution. So if you are creating a new health monitor, please create it in the task prior to a reference. We also will now list our servers. These are done with the `servers` list, which includes a dictionary for each server. In this example we use an IPv4 address.

```
# Create pool, 1 VIP, and assign SSL Certificate to Pool
---
- hosts: localhost
  connection: local
  environment:
      AVI_CONTROLLER: 10.10.27.90
      AVI_USERNAME: admin
      AVI_PASSWORD: AviNetworks123!
  roles:
    - role: avinetworks.avisdk
  tasks:
    # Create the pool using the avi_pool api
    - name: Create the testpool1 pool
      avi_pool:
        tenant: admin
        name: testpool1
```

(continues on next page)

(continued from previous page)

```
        state: present
        enabled: false
        health_monitor_refs:
          - '/api/healthmonitor?name=System-HTTP'
        servers:
          - ip:
              addr: 10.90.130.13
              type: V4
          - ip:
              addr: 10.90.130.15
              type: V4
    ...................
```

Next we will create the virtual service. This module will look similar to this. We of course define the name, tenant, state, and other parameters required for the Virtual Service. We also specify the list of `services` to the virtual service. We also call in the `System-Standard` SSL profile, the Application profile, and define the SSL key, and certificate.

Additional values can be found in the Swagger documentation at https://<controller-ip>/swagger/#!/default/get_virtualservice

```
...................
- name: Create the Virtual Service testvs1 and assign testpool1 to it
  avi_virtualservice:
    tenant: admin
    name: testvs1
    state: present
    performance_limits:
      max_concurrent_connections: 1000
    services:
      - port: 443
        enable_ssl: true
      - port: 80
    ssl_profile_ref: '/api/sslprofile?name=System-Standard'
    application_profile_ref: '/api/applicationprofile?name=System-Secure-HTTP'
    ssl_key_and_certificate_refs:
      - '/api/sslkeyandcertificate?name=System-Default-Cert'
    ip_address:
      addr: 10.90.131.105
      type: V4
    pool_ref: '/api/pool?name=testpool1'
```

5.1.2 Create Avi User

This is an example on how to work with the Avi Ansible modules to create a new user within a Tenant. You can definitely loop with the user as well.

Note: Please be aware, this module is not idempotent because of the inability to match the `password` field.

Using the `avi_api_session` module we are able to make API calls freely to Avi. Using this we can control the type of HTTP Method, as well as the data and path of the call.

```
---
- hosts: localhost
  connection: local
  environment:
    AVI_CONTROLLER: ec2-34-250-111-254.eu-west-1.compute.amazonaws.com
    AVI_USERNAME: admin
    AVI_PASSWORD: ************
  roles:
    - role: avinetworks.avisdk
  tasks:
    - name: Check if user exists on Avi
      avi_api_session:
        http_method: get
        path: user?name=testuser
      register: user_exists

    - name: Create User on Avi
      avi_api_session:
        http_method: post
        path: user
        data:
          require_password_confirmation: false
          is_active: true
          access:
            - tenant_ref: '/api/tenant?name=admin'
              role_ref: '/api/role?name=System-Admin'
          default_tenant_ref: '/api/tenant?name=admin'
          full_name: testuser
          username: testuser
          password: AviNetworks123
      when: user_exists.obj.count < 1
```

As you can see because idempotency isn't default, when using `avi_api_session` we created a check to see if the user already exists, and if the user doesn't exist we run the next task which creates the user.

5.2 Modules index

At Avi we understand how important automation is, and in the Ansible world modules are used to give power to your playbooks. With our modules you can easily communicate with the Avi REST API's.

Ansible is an extremely popular tool among operations and network engineers, and allows them to automate many tasks, which are often repeatable. It can configure systems, network devices, handle orchestration, and launch continuous deployments. It's goal is ease of use, and minimum amount of moving parts. It uses OpenSSH for transport instead of proprietary protocols, and communication. It's agentless, and can be ran from just about any machine (of course with network access). It's designed to be used by all types of users. For Ansible there is no need to manage remote daemons. It's also decentralized and relies on your existing credentials to gain access to the remote hosts.

We've built Ansible Roles that can be used to deploy Ansible in your environment, as well as Ansible Modules that can be used to manage and control your Avi Vantage deployment. Due to our vast API, our Ansible integration can do anything our API permits.