Ansible Role: eos_config_deploy_cvp

Table of Contents:

Overview

eos_config_deploy_cvp, is a role that deploys the configuration to Arista EOS devices via CloudVision Management platform.

The **eos_config_deploy_cvp** role:

- Designed to configure CloudVision with fabric configlets & topology.
- Deploy intended configlets to devices and execute pending tasks.

Role requirements

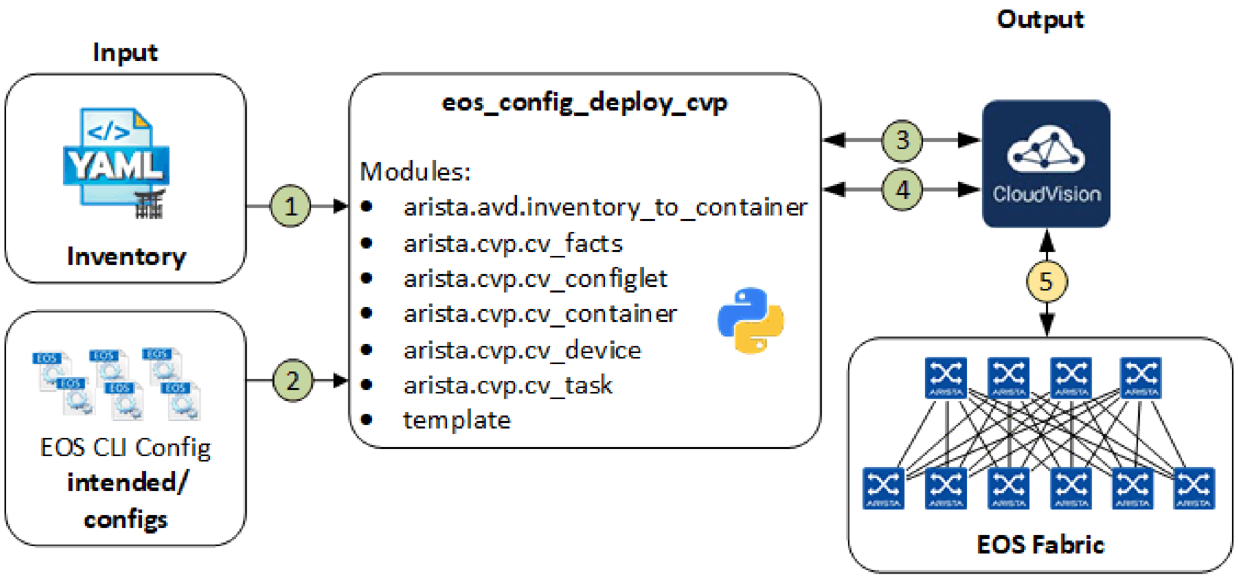
This role requires to install `arista.cvp` collection to support CloudVision interactions.

```
$ ansible-galaxy collection install arista.cvp
```

> **NOTE**: When using ansible-cvp modules, the user that is executing the ansible-playbook has to have access to both CVP and the EOS CLI.

Role Inputs and Outputs

Figure 1 below provides a visualization of the roles inputs, outputs and tasks in order executed by the role.



1. Read inventory file
    1. Build containers topology
2. Role looks for configuration previously generated by `arista.avd.eos_cli_config_gen`
    1. List configuration and build configlets list, one per device.
3. Role looks for additional configlets to attach to either devices or containers.
4. Build CloudVision configuration using `arista.cvp` collection:
    1. Build configlets on CV.
    2. Create containers topology.
    3. Move devices to container.
    4. Bind Configlet to device.
5. Deploy Fabric configuration by running all pending tasks (optional, if execute_tasks == true).

Inputs

Inventory configuration:

An entry must be part of the inventory to describe CloudVision server. `arista.cvp` modules use httpapi approach. Example below provides framework to use in your inventory.

```
all:
  children:
    cloudvision:
      hosts:
        cv_server01:
          ansible_httpapi_host: 10.83.28.164
          ansible_host: 10.83.28.164
          ansible_user: ansible
          ansible_password: ansible
          ansible_connection: httpapi
          ansible_httpapi_use_ssl: True
          ansible_httpapi_validate_certs: False
          ansible_network_os: eos
          ansible_httpapi_port: 443
          # Configuration to get Virtual Env information
          ansible_python_interpreter: $(which python3)
```

For complete list of authentication options available with Cloudvision Ansible collection, you can read dedicated page on arista.cvp collection.

Module variables

- **`container_root`**: Inventory group name where Fabric devices are located. Default: `all`.
- **`configlets_prefix`**: Prefix to use for configlet on CV side. Default: ``.
- **`device_filter`**: Filter to target a specific set of devices on CV side. Default: `AVD--`. It can be either a string or a list of string.
- **`state`**: `present` / `absent`. Support creation or cleanup topology on CV server. Default: `present`.
- **`execute_tasks`**: `true` / `false`. Support automatically excuting pending tasks. Default: `false`.
- **`cvp_configlets`**: Structure to add additional configlets to those automatically generated by AVD roles.
- **`cv_collection`**: Version of Cloudvision collection to use. Can be `v1` or `v3`. Default is `v1`.

!!! warning Use of arista.cvp in version 3 is only supported for testing purpose. Please use it carrefully and for lab only

Getting Started

Below is an example of how to use role with a single string as `device_filter`:

```yaml
tasks:
  - name: run CVP provisioning
    import_role:
        name: eos_config_deploy_cvp
    vars:
      container_root: 'DC1_FABRIC'
      configlets_prefix: 'DC1-AVD'
      device_filter: 'DC1'
      state: present
      execute_tasks: false
```

Next code is an example of how to use role with a list of strings to create `device_filter` entry:

```yaml
tasks:
  - name: run CVP provisioning
    import_role:
        name: eos_config_deploy_cvp
    vars:
      container_root: 'DC1_FABRIC'
      configlets_prefix: 'DC1-AVD'
      device_filter:
        - 'DC1'
        - 'DC2'
      state: present
      execute_tasks: false
```

Ignore devices not provisioned in Cloudvision

When you want to provision a complete topoplogy and devices are not already in Cloudvision, you can configure inventory to ignore these devices by using a host variable: `is_deployed`

- `is_deployed: true` or `is_deployed is not defined`: An entry in **cv_device** is generated and AVD will configure device on Cloudvision. If device is undefined, an error is raised.
- `is_deployed: false`: Device is not configured in **cv_device** topology and only its configlet is uploaded on Cloudvision.

Here is an overview with the key configured in the YAML inventory:

```
  DC1_BL1:
    hosts:
      DC1-BL1A:
        ansible_port: 8012
  DC1_BL2:
    hosts:
      DC1-BL2A:
        ansible_port: 8012
        # Device configuration is generated by AVD
        # Device is not configured on Cloudvision (configlet is
uploaded)
        is_deployed: false
```

Add additional configlets

This structure MUST be part of `group_vars` targeting `container_root`. Below is an example applied to `eos_l3_evpn`:

```
# group_vars/DC1_FABRIC.yml

# List of additional CVP configlets to bind to devices and containers
# Configlets MUST be configured on CVP before running AVD playbooks.
cv_configlets:
  containers:
    <name of container>:
      - <First configlet to attach>
      - <Second configlet to attach>
      - <...>
  devices:
    <inventory_hostname>:
      - <First configlet to attach>
      - <Second configlet to attach>
      - <...>
    <inventory_hostname>:
      - <First configlet to attach>
      - <Second configlet to attach>
      - <...>
```

Full example:

```
# group_vars/DC1_FABRIC.yml

# List of additional CVP configlets to bind to devices and containers
```

```yaml
# Configlets MUST be configured on CVP before running AVD playbooks.
cv_configlets:
  containers:
    DC1_L3LEAFS:
      - GLOBAL-ALIASES
  devices:
    DC1-L2LEAF2A:
      - GLOBAL-ALIASES
    DC1-L2LEAF2B:
      - GLOBAL-ALIASES
```

Notes:

- These configlets **MUST** be created previously on CloudVision server and won't be managed by AVD roles.
- Current version **does not support** configlets unbound from container for safety reason. In such case, configlets should be removed from variables and manually unbind from containers on Cloudvision.

Run module with different tags

This module also supports tags to run a subset of ansible tasks:

- **`build`**: Generate Arista Validated Design configuration for EOS devices (structure_configs / configs / documentation) and CloudVision inputs.
- **`provision`**: Run `build` tags + configure Cloudvision with information generated in previous tasks

```shell
$ ansible-playbook playbook.to.deploy.with.cvp.yml --tags "provision"
```

Other option to run a subset of ansible tasks is to use **`--skip-tags <tag>`**:

- in order to run module to update existing configlets only, following command can be used:

```shell
$ ansible-playbook playbook.to.deploy.with.cvp.yml --skip-tags
"containers"
```

- Skipping multiple tags could make playbook even more lightweight. For example, above command with avoiding CVP task execution

```shell
$ ansible-playbook playbook.to.deploy.with.cvp.yml --skip-tags
"containers,apply"
```

Outputs

- None.

Tasks

1. Copy generated configuration to CloudVision static configlets.
2. Create container topology and attach devices to correct container
3. Bind configlet to devices.

4. Apply generated tasks to deploy configuration to devices.

Requirements

Requirements are located here: avd-requirements

License

Project is published under Apache 2.0 License