ACB

Playbook Provisioning Operational Document

Prepared by Ashu Rana On behalf of Rackspace

Table of Contents

1 Introduction

1.1 Purpose of Document

The purpose of this document is to serve as a comprehensive operational guide for developers involved in the deployment and management of a project using Ansible automation. The document aims to provide a clear and detailed understanding of the Ansible playbook, its design, prerequisites, execution workflow, and the resulting resources in an OpenStack environment. Developers can refer to this guide for step-by-step instructions, best practices, and troubleshooting tips to ensure a smooth and successful deployment.

1.2 Project Overview

Rackspace has implemented the Ansible Automation Platform as a pivotal component of ACB's IT infrastructure. The platform comprises key elements, including the Controller, Hub Execution nodes, Database node, and Event Driven node, all of which have been meticulously configured and integrated as per the Architecture discussed and signed off by ACB.

1.2.1 Ansible Playbook

The heart of the automation lies in the Ansible playbook designed for this project. The playbook encapsulates a set of tasks, roles, and configurations essential for deploying and managing the ACB IT infrastructure. It is crafted to align with ACB's specific requirements and leverages the Ansible Automation Platform's capabilities to streamline operational processes.

1.2.2 Playbook Components

Roles and Responsibilities: The playbook is organized into roles, each responsible for a specific aspect of the deployment. These roles collectively contribute to the seamless execution of the automation process.

1.2.3 Automation Workflow

The Ansible playbook follows a well-defined execution workflow, encompassing initialization, environment preparation, playbook execution tasks. This structured approach ensures a methodical and reproducible deployment process.

1.2.4 Integration with Ansible Tower

To facilitate ease of use and centralized management, the Ansible playbook is integrated with Ansible Tower. Developers can utilize Ansible Tower's features to upload the playbook, configure variables, and execute the automation seamlessly. The Tower setup is discussed in detail in the subsequent sections of this operational guide.

1.3 Prerequisites and Environment Overview

Before executing the Ansible playbook for provisioning resources in the OpenStack environment, ensure that the following prerequisites are met:

1.3.1 System Requirements

The system where the Ansible playbook will be executed should meet the following requirements:

- Sufficient CPU and memory resources to handle the Ansible Tower and Docker environment.
- Adequate disk space for storing Ansible playbooks, Docker images, and other temporary files.

1.3.2 Environment Configuration

Configuration	Value	Details
GitLab Repository		
GitLab URL	https://gitlab.acb.vn	
Repository Name	OPENSTACK-VM-WORKFLOW	Name of the GitLab repository
Ansible Tower Setup		
Ansible Tower URL	https://aap.acb.vn	
Tower Credentials	Ansible-Openstack-DEV	Credentials for open stack dev environment created in Ansible Tower.
Proxy Configuration	Ansible Tower should be configured with the necessary proxy settings for downloading dependencies.	Configured in Anisble Tower.
Execution Environment Image	Ensure that the Docker image for the execution environment includes Ansible, OpenStack SDK, and other required dependencies.	Handled in ansible playbook task
CSV File Location		
CSV File in GitLab	Confirm that the CSV file containing provisioning parameters is committed to the specified GitLab repository.	Location of CSV file in the GitLab repository

1.3.3 Configuration Files

- **CSV File:** The user is required to provide a CSV file in a specified format. This CSV file will be read by the Ansible playbook to gather parameters for resource provisioning. The CSV file should be committed to the GitLab repository for version control and playbook execution.

1.3.4 CSV Parameters Sequence

1. OpenStackCluster
2. ProjectEnvironment
3. ProjectName
4. ProjectCPU
5. ProjectMemory
6. ProjectDisk
7. ProjectPermissionGroup
8. ProjectPermissionType
9. NetworkSegmentName
10. NetworkSegmentCIDR
11. DHCPRange
12. DNSServer
13. PortName
14. PortIP
15. IPProvider
16. RouterName
17. DefaultRouteName
18. DefaultRouteIP
19. NetworkGatewayIP
20. AttachNetworkSegment
21. SecurityGroupName
22. ServerGroupName
23. ServerGroupPolicy
24. InstanceName
25. InstanceFlavor
26. InstanceDescription
27. BootFromVolume
28. InstanceBootSize
29. ExtendOSDisk
30. OSImage
31. AttachNetworkPort
32. AttachSecurityGroup
33. KeyPair
34. VolumeName
35. VolumeSize
36. AttachVolume

The CSV file should follow a consistent sequence of parameters, where each parameter represents a distinct aspect of the OpenStack resource provisioning. Developers must adhere to this sequence to ensure the playbook correctly interprets and utilizes the provided information. The sequence is designed to capture various details, such as project information, network configuration, security settings, and resource specifications.

As a best practice, it's essential for users to review and adhere to the specified order when populating the CSV file. Any deviation from the expected sequence might lead to misinterpretation of parameters and potential errors during playbook execution. To facilitate this process, the following are key groups of parameters and their corresponding positions in the CSV file:

1. **Project Information (1-8):** OpenStackCluster, ProjectEnvironment, ProjectName, ProjectCPU, ProjectMemory, ProjectDisk, ProjectPermissionGroup, ProjectPermissionType.
2. **Network Configuration (9-20):** NetworkSegmentName, NetworkSegmentCIDR, DHCPRange, DNSServer, PortName, PortIP, IPProvider, RouterName, DefaultRouteName, DefaultRouteIP, NetworkGatewayIP, AttachNetworkSegment.
3. **Security and Server Groups (21-23):** SecurityGroupName, ServerGroupName, ServerGroupPolicy.
4. **Instance Configuration (24-35):** InstanceName, InstanceFlavor, InstanceDescription, BootFromVolume, InstanceBootSize, ExtendOSDisk, OSImage, AttachNetworkPort, AttachSecurityGroup, KeyPair, VolumeName, VolumeSize, AttachVolume.

By maintaining the correct sequence, users can streamline the provisioning process and ensure that the Ansible playbook accurately interprets and implements the desired configurations within the OpenStack environment. This adherence to sequence promotes consistency and minimizes the risk of errors during resource deployment.

1.3.5 User Input CSV Example

```
OPS-DEV,DEV,ACB-ANSIBLE-DEV,20,30720,800,SURT-Cloud-Admin,admin,ACB-
ANSIBLE-DEV_NETWORK_Internal-OOS,10.26.141.224/27,10.26.141.226-
10.26.141.254,10.56.240.213,DTTANSAPPV001PORT,10.26.141.229,10.24.89.57,ACB
-ANSIBLE-DEV_ROUTER,null,null,10.26.141.225,ACB-ANSIBLE-
DEV_SUBNET_Internal-OOS,ACB-ANSIBLE-DEV_SG,ACB-ANSIBLE-DEV_Svr-
Grp,Affinity,DTTANSAPPV001,S04.008,ACB-ANSIBLE-DEV_NETWORK_Internal-
OOS_APP-01,True,120,yes,ACB-RHEL-8.6_x64_v1.0,DTTANSAPPV001,ACB-ANSIBLE-
DEV_SG,CLOUD-INFRA-KEYPAIR,DTTANSAPPV001_VDB,10,ceph_hdd
```

Ensure that the CSV file is committed to the specified GitLab repository before initiating the Ansible playbook execution. The playbook will read this file to gather configuration parameters for the resource provisioning process.

- Users should provide values for each parameter in a comma-separated format.
- Sequence matters; follow the specified order for entering values.
- For optional parameters, users can enter 'null' if they choose not to provide a value.

This example showcases a user-entered set of parameters for creating an OpenStack environment named "ACB-ANSIBLE-DEV." Customize the values according to specific project requirement

2 Ansible Playbook Structure

2.1 Directory Structure

```
openstack_vm_workflow/
|-- collections/
|   |-- requirements.yml
|
|-- roles/
|   |-- authentication/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- instance_creation /
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- network_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- port_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- prerequisites/
|       |-- tasks/
|       |   |-- main.yml
|
|   |-- project_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- read_and_display_csv_parameters/
|   |   |-- tasks/
|   |   |   |-- main.yml
|   |   |   |-- loop_tasks.yml
|
|   |-- router_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- security_group_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|
|   |-- security_group_rule_creation/
|   |   |-- tasks/
|   |   |   |-- main.yml
|   |   |   |-- create_security_group_rule.yml
|
|-- ansible.csv
|-- README.md
|-- sg-rule.csv
|-- site.yml.
```

2.2 Explanation of Key Files and Their Purposes

1. **Collections/requirements.yml**
   - Lists the required Ansible collection, specifying modules needed for the playbook's OpenStack integration.
2. **Roles**
   - Each role represents a specific task or component in the OpenStack VM workflow.
3. **Authentication/tasks/main.yml**
   - Confirms the authentication with open stack environment.
4. **instance_creation/tasks/main.yml**
   - Manages the creation of OpenStack virtual machines.
5. **network_creation/tasks/main.yml**
   - Manages the creation of OpenStack networks.
6. **port_creation/tasks/main.yml**
   - Handles the creation of OpenStack ports.
7. **prerequisites/tasks/main.yml**
   - Checks and upgrades openstack SDK before proceeding with the playbook.
8. **project_creation/tasks/main.yml**
   - Manages the creation of OpenStack projects.
9. **read_and_display_csv_parameters/tasks/main.yml**
   - Reads values from the CSV file entered by the user for each workflow.
10. **read_and_display_csv_parameters/tasks/loop_tasks.yml**
    - Loops through each line in the CSV file and includes roles for project creation, network creation, router creation, security group creation, security group rule creation, port creation, and VM creation.
11. **router_creation/tasks/main.yml**
    - Manages the creation of OpenStack routers.
12. **security_group_creation/tasks/main.yml**
    - Handles the creation of OpenStack security groups.
13. **security_group_rule_creation/tasks/main.yml**
    - Manages the creation of security group rules.
14. **security_group_rule_creation/tasks/create_security_group_rule.yml:**
    - Specific tasks for creating security group rules.
15. **ansible.csv**
    - CSV file containing user-entered values for each workflow.
16. **README.md**
    - Contains information about the repo and what's inside[to be updated by ACB].
17. **sg-rule.csv**
    - CSV file containing security rules.
18. **site.yml**

- The main playbook file that orchestrates the entire workflow by including relevant roles.

3 Running the Playbook

Follow these steps to trigger the Ansible playbook execution in Ansible Tower after committing the CSV file to GitLab:

1. **Commit the CSV File:**
   - Commit the CSV file containing the provisioning parameters to the specified GitLab repository. Ensure the file is in the correct format and follows the defined parameter sequence.
2. **Re-Sync the Project:**
   - In the Ansible Tower UI, navigate to the "Projects" section.
   - Find the project "**OpenstackVMWorkflow**" and click on it.
   - Click the "Re-Sync Project" button to synchronize the project with the latest changes from the GitLab repository.
3. **Access the Templates Section:**
   - Navigate to the "Templates" section in Ansible Tower.
4. **Find the Job Template:**
   - Locate the job template "**OpenstackVmWorkflowTemplate**".
5. **Launch the Job Template:**
   - Click on the job template to open it.
   - Click on the rocket icon or the "Launch" button to manually trigger the playbook execution.
6. **Monitor Job Execution:**
   - Monitor the progress of the job in the Ansible Tower UI. You can view detailed logs and status updates to track the execution workflow.
7. **Review Results:**
   - Once the job completes, review the results to ensure successful execution. Check for any error messages or warnings.

4 Execution Workflow

The Ansible playbook for creating OpenStack resources follows a structured workflow consisting of several tasks. Each task is designed to perform specific actions, such as validating authentication, reading parameters from a CSV file, and creating various OpenStack resources. The following is the sequence of tasks executed during the playbook:

4.1 1. Check Authentication

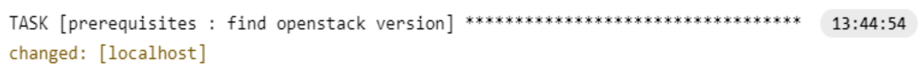
The task in `authentication/main.yml` focuses on confirming authentication by attempting to gather information about OpenStack images. The `openstack.cloud.image_info` module is employed for this purpose. The information retrieved, stored in the `image_info_output` variable, can be useful for validating successful authentication and can be used for further tasks in the Ansible playbook.

1. **Get Image Info to Confirm Authentication:**
   - Utilizes the **`openstack.cloud.image_info`** Ansible module to retrieve information about OpenStack images.
   - Registers the output of the image information retrieval process in a variable named **`image_info_output`**.

4.2 2. Prerequisites

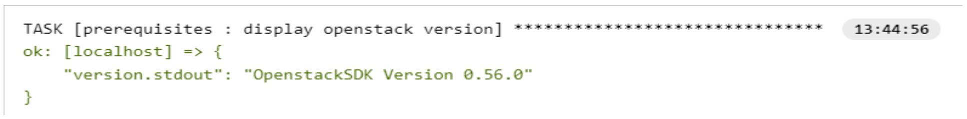
The tasks in `prerequisites/main.yml` aim to check the current version of OpenStack installed on the system. If an outdated version or no version is found, it proceeds to upgrade or install the OpenStack Python library using `pip`. The summary of tasks includes identifying, displaying, and upgrading the OpenStack version, ensuring the required version is available for the subsequent tasks in the Ansible playbook.

1. **Find OpenStack Version:**
   - Executes a command using Python to determine the installed version of OpenStack.
   - Registers the output of the command in a variable named **`version`**.



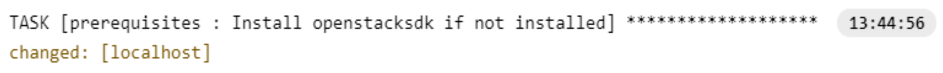
```
TASK [prerequisites : find openstack version] ********************************** 13:44:54
changed: [localhost]
```

2. **Display OpenStack Version:**
   - Outputs the OpenStack version obtained in the previous step for informational purposes.
   - Uses the **`debug`** module to print the version to the console.



```
TASK [prerequisites : display openstack version] ****************************** 13:44:56
ok: [localhost] => {
    "version.stdout": "OpenstackSDK Version 0.56.0"
}
```

3. **Install OpenStack if Not Installed:**
   - Uses the **`pip`** module to ensure that the OpenStack Python library is installed, specifying a version of 1.0.0 or higher.
   - Registers the output of the installation process in a variable named **`upgraded_version`**.



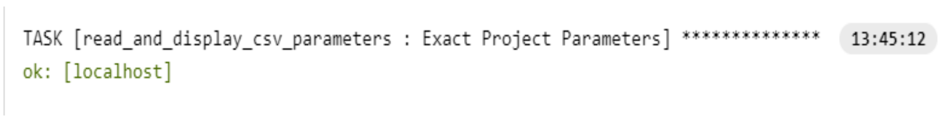
```
TASK [prerequisites : Install openstacksdk if not installed] ****************** 13:44:56
changed: [localhost]
```

4. **Display Upgraded OpenStack Version:**
   - Outputs the upgraded OpenStack version after the installation process, providing visibility into the updated state.
   - Uses the **`debug`** module to print the upgraded version to the console.

4.3 3. Read and Display CSV Parameters

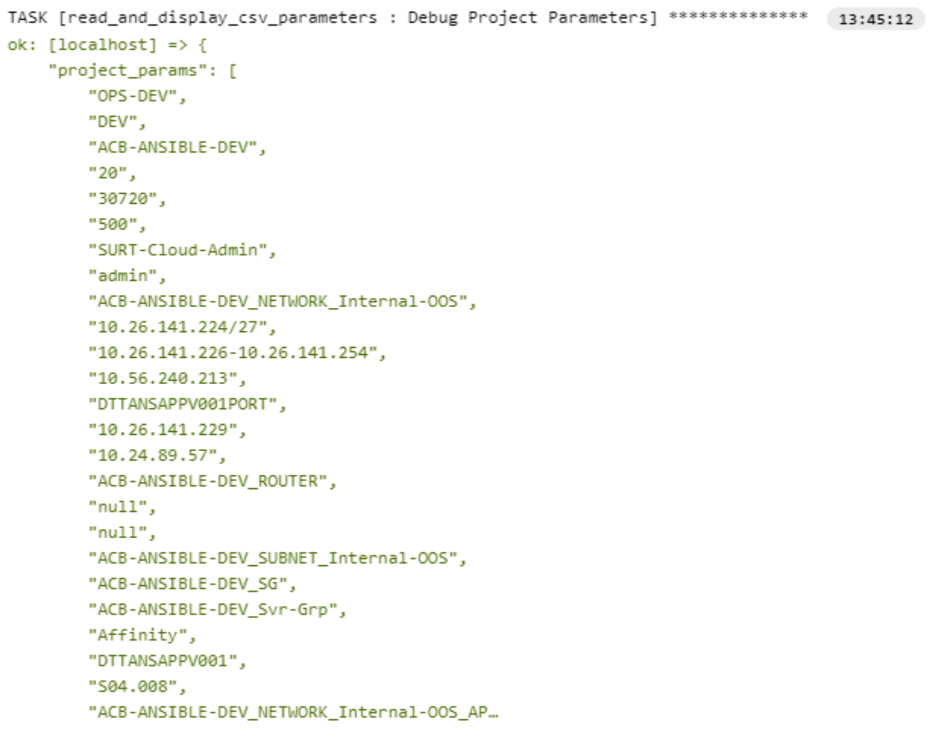
This role orchestrates the reading of CSV file committed in GitLab line-by-line with the execution of individual roles responsible for creating and configuring various components within the OpenStack environment. The roles cover project creation, network setup, router configuration, security group creation, security group rule definition, port creation, and instance provisioning.

1. **Extract Project Parameters:**
   - Reads and splits the CSV input to extract project-related parameters.



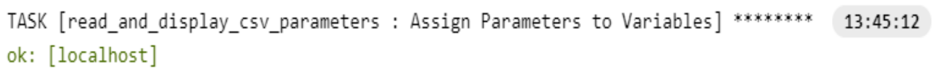
```
TASK [read_and_display_csv_parameters : Exact Project Parameters] **************  13:45:12
ok: [localhost]
```

2. **Debug Project Parameters:**
   - Displays debug information for the extracted project parameters.



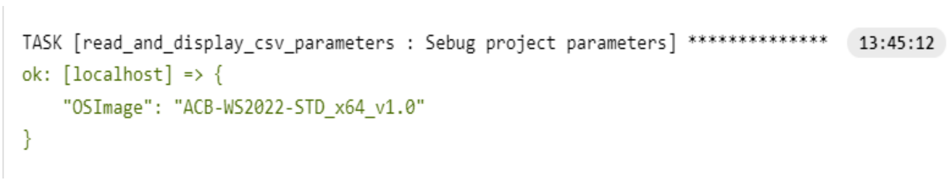
```
TASK [read_and_display_csv_parameters : Debug Project Parameters] **************  13:45:12
ok: [localhost] => {
    "project_params": [
        "OPS-DEV",
        "DEV",
        "ACB-ANSIBLE-DEV",
        "20",
        "30720",
        "500",
        "SURT-Cloud-Admin",
        "admin",
        "ACB-ANSIBLE-DEV_NETWORK_Internal-OOS",
        "10.26.141.224/27",
        "10.26.141.226-10.26.141.254",
        "10.56.240.213",
        "DTTANSAPPV001PORT",
        "10.26.141.229",
        "10.24.89.57",
        "ACB-ANSIBLE-DEV_ROUTER",
        "null",
        "null",
        "ACB-ANSIBLE-DEV_SUBNET_Internal-OOS",
        "ACB-ANSIBLE-DEV_SG",
        "ACB-ANSIBLE-DEV_Svr-Grp",
        "Affinity",
        "DTTANSAPPV001",
        "S04.008",
        "ACB-ANSIBLE-DEV_NETWORK_Internal-OOS_AP…
```

3. **Assign Parameters to Variables:**
   - Maps project parameters to Ansible variables for subsequent role executions.



```
TASK [read_and_display_csv_parameters : Assign Parameters to Variables] ********  13:45:12
ok: [localhost]
```

4. **Debug Project Parameters:**
   - Further debugs specific parameters relevant to the workflow.



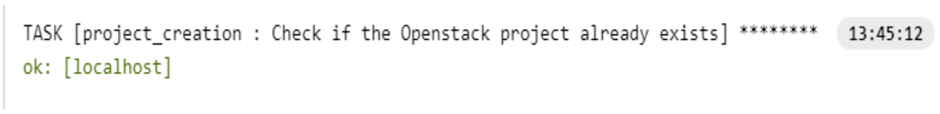
```
TASK [read_and_display_csv_parameters : Sebug project parameters] **************  13:45:12
ok: [localhost] => {
    "OSImage": "ACB-WS2022-STD_x64_v1.0"
}
```

5. **Role Execution: project_creation:**
   - Checks for the existence of the OpenStack project and creates it if not present.
6. **Role Execution: network_creation:**
   - Creates a network and subnet within the specified project.
7. **Role Execution: router_creation:**
   - Creates a router with specified parameters, including the interface to a network segment.
8. **Role Execution: security_group_creation:**
   - Creates a security group within the designated project.
9. **Role Execution: security_group_rule_creation:**
   - Reads, parses, and applies security group rules from a user-provided CSV file.
10. **Role Execution: port_creation:**
    - Creates an OpenStack port associated with the specified network.
11. **Role Execution: instance_creation:**
    - Deploys a virtual machine (VM) within the defined project with specified configurations.

4.4 4. Project Creation

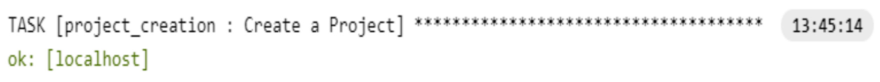
The **project_creation/main.yml** playbook performs a series of tasks to manage the creation and configuration of an OpenStack project. It begins by checking if the project already exists and proceeds to create the project, assign roles for operations, update project quotas. The playbook is designed to ensure the proper setup and configuration of an OpenStack project with flexibility and modularity.

1. **Check if the OpenStack project already exists**
   - Uses the **os_project_info** Ansible module to check if the specified project already exists.



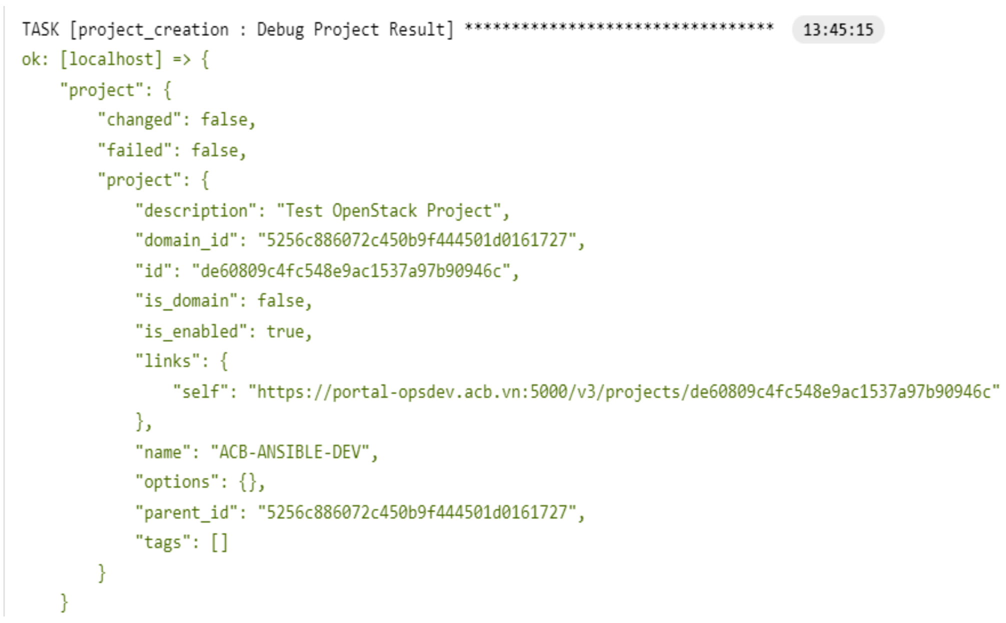
```
TASK [project_creation : Check if the Openstack project already exists] ********  13:45:12
ok: [localhost]
```

2. **Create a Project**
   - Utilizes the **os_project** Ansible module to create an OpenStack project with the specified parameters.



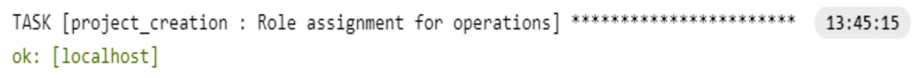
```
TASK [project_creation : Create a Project] ************************************  13:45:14
ok: [localhost]
```

3. **Debug Project Result**
   - Prints debugging information about the result of the project creation.

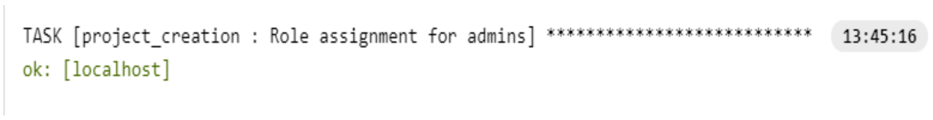


```
TASK [project_creation : Debug Project Result] ********************************* 13:45:15
ok: [localhost] => {
    "project": {
        "changed": false,
        "failed": false,
        "project": {
            "description": "Test OpenStack Project",
            "domain_id": "5256c886072c450b9f444501d0161727",
            "id": "de60809c4fc548e9ac1537a97b90946c",
            "is_domain": false,
            "is_enabled": true,
            "links": {
                "self": "https://portal-opsdev.acb.vn:5000/v3/projects/de60809c4fc548e9ac1537a97b90946c"
            },
            "name": "ACB-ANSIBLE-DEV",
            "options": {},
            "parent_id": "5256c886072c450b9f444501d0161727",
            "tags": []
        }
    }
```

4. **Role assignment for operations**
   - Assigns a role to a specified group for operations within the created project.

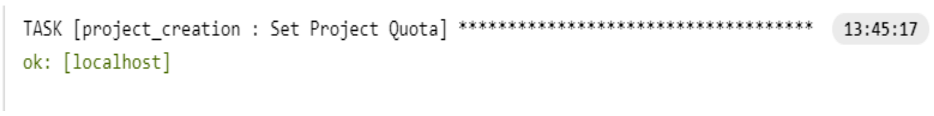


```
TASK [project_creation : Role assignment for operations] ********************** 13:45:15
ok: [localhost]
```

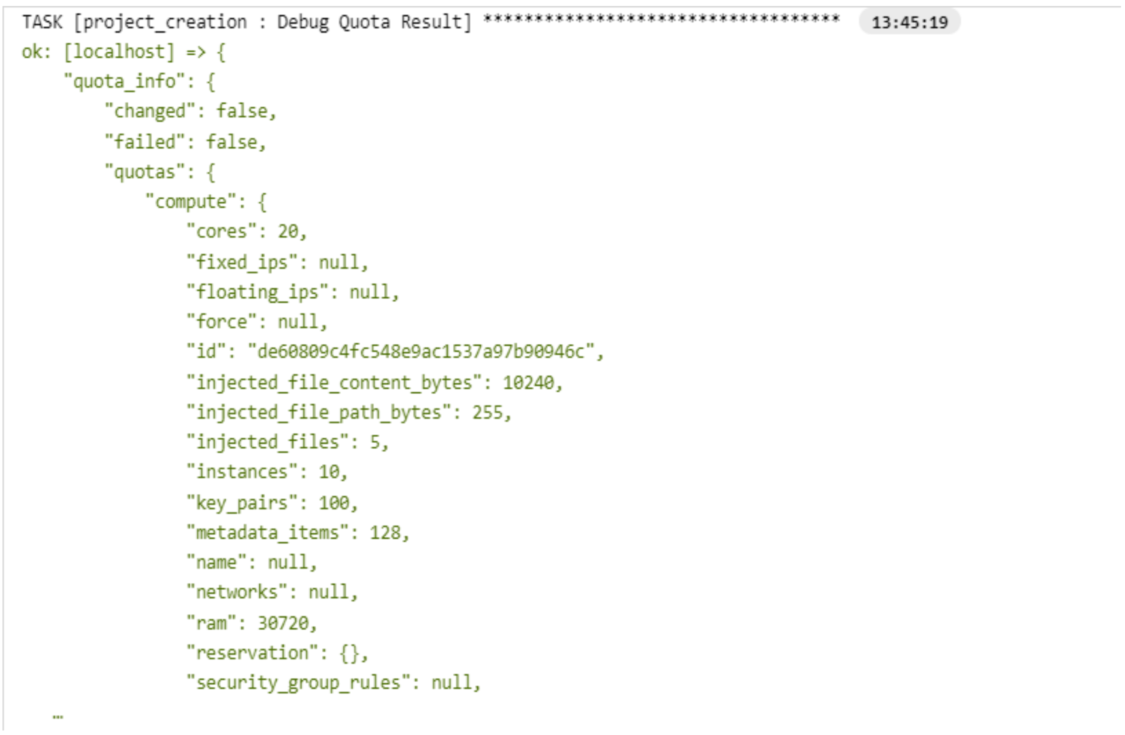


```
TASK [project_creation : Role assignment for admins] ************************** 13:45:16
ok: [localhost]
```

5. **Update Project Quota**
   - Adjusts the resource quotas for the created project based on the provided parameters.



```
TASK [project_creation : Set Project Quota] *********************************** 13:45:17
ok: [localhost]
```



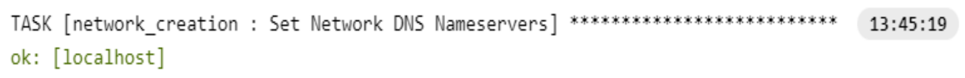
```
TASK [project_creation : Debug Quota Result] ***********************************  13:45:19
ok: [localhost] => {
    "quota_info": {
        "changed": false,
        "failed": false,
        "quotas": {
            "compute": {
                "cores": 20,
                "fixed_ips": null,
                "floating_ips": null,
                "force": null,
                "id": "de60809c4fc548e9ac1537a97b90946c",
                "injected_file_content_bytes": 10240,
                "injected_file_path_bytes": 255,
                "injected_files": 5,
                "instances": 10,
                "key_pairs": 100,
                "metadata_items": 128,
                "name": null,
                "networks": null,
                "ram": 30720,
                "reservation": {},
                "security_group_rules": null,
    …
```

4.5 5. Network Creation

The **network_creation/main.yml** playbook focuses on creating and configuring an OpenStack network and subnet. It begins by setting DNS nameservers for the network, then proceeds to create the network, debug the network creation result, create a subnet within the network, and finally, debug the result of subnet creation.

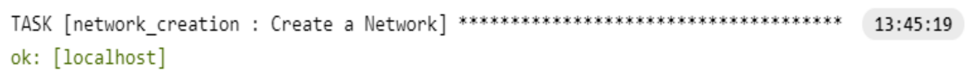
Each task contributes to the overall goal of establishing a network infrastructure in the OpenStack environment. The use of debugging tasks provides visibility into the results of key operations for troubleshooting and verification.

1. **Set network_dns_nameservers:**
   - This task sets the DNS nameservers for the network based on the provided DNSServer parameter.



```
TASK [network_creation : Set Network DNS Nameservers] *************************  13:45:19
ok: [localhost]
```

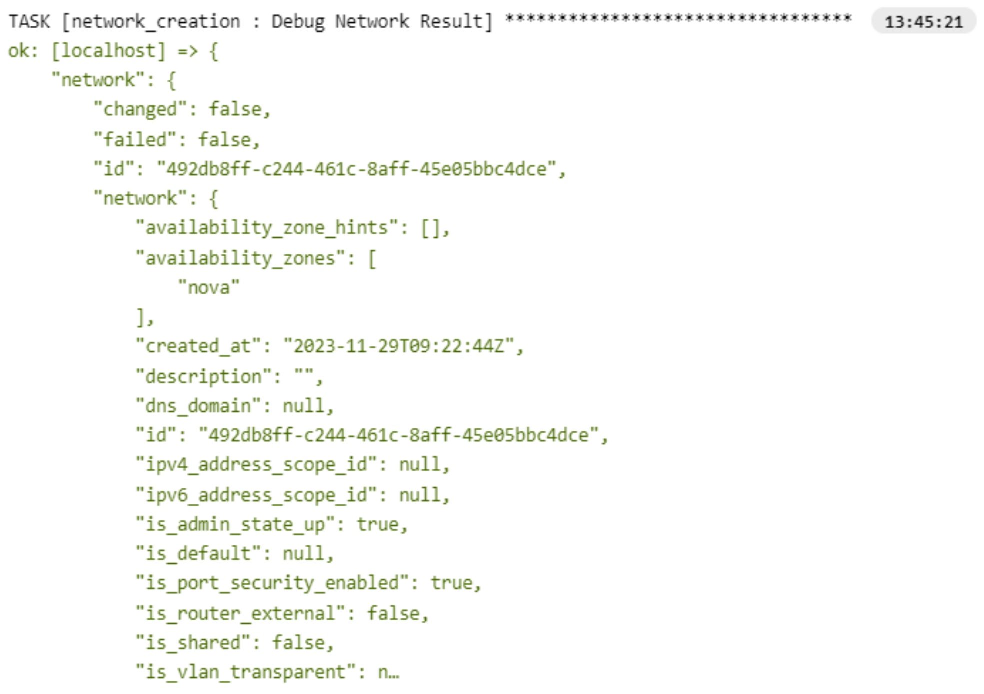
2. **Create a Network:**
   - This task creates an OpenStack network with specified parameters, including the network name and associated project.



```
TASK [network_creation : Create a Network] ************************************  13:45:19
ok: [localhost]
```

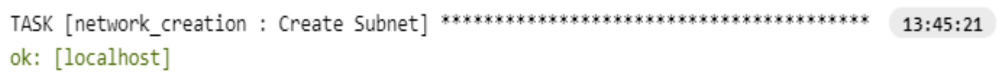
3. **Debug Network Result:**

- This task prints debugging information about the result of the network creation, providing visibility into the network configuration.



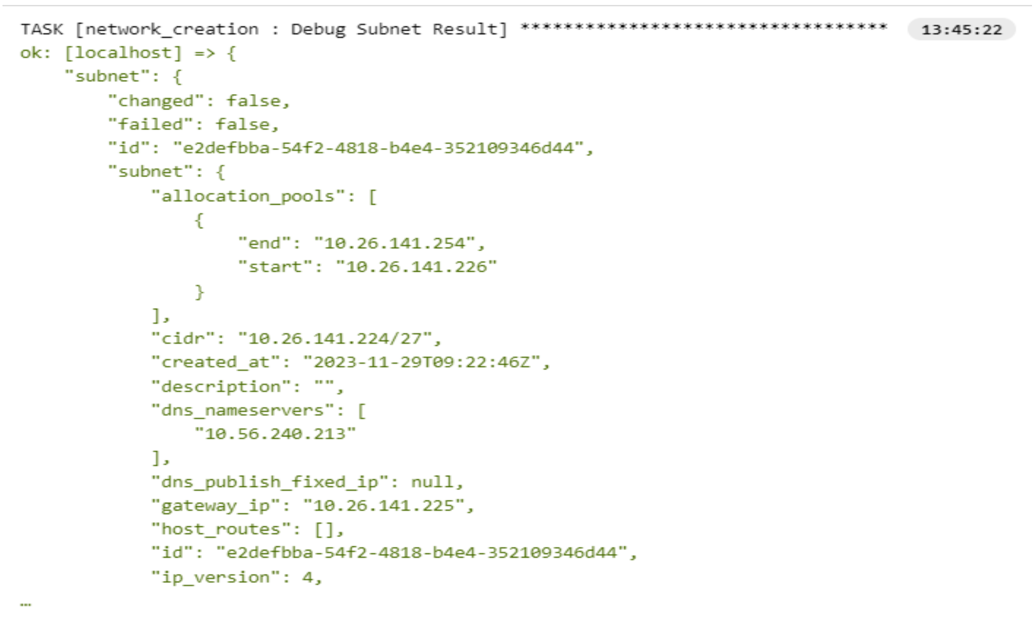
```
TASK [network_creation : Debug Network Result] ********************************  13:45:21
ok: [localhost] => {
    "network": {
        "changed": false,
        "failed": false,
        "id": "492db8ff-c244-461c-8aff-45e05bbc4dce",
        "network": {
            "availability_zone_hints": [],
            "availability_zones": [
                "nova"
            ],
            "created_at": "2023-11-29T09:22:44Z",
            "description": "",
            "dns_domain": null,
            "id": "492db8ff-c244-461c-8aff-45e05bbc4dce",
            "ipv4_address_scope_id": null,
            "ipv6_address_scope_id": null,
            "is_admin_state_up": true,
            "is_default": null,
            "is_port_security_enabled": true,
            "is_router_external": false,
            "is_shared": false,
            "is_vlan_transparent": n…
```

4. **Create Subnet:**
    - This task creates a subnet within the previously created network, specifying parameters such as CIDR, DHCP settings, and DNS nameservers.



```
TASK [network_creation : Create Subnet] ****************************************  13:45:21
ok: [localhost]
```

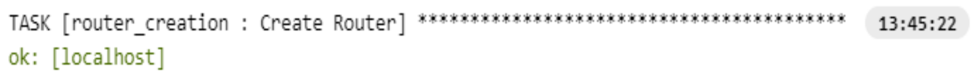
5. **Debug Subnet Result:**
    - This task prints debugging information about the result of the subnet creation, helping to verify the subnet configuration.


```
TASK [network_creation : Debug Subnet Result] **********************************  13:45:22
ok: [localhost] => {
    "subnet": {
        "changed": false,
        "failed": false,
        "id": "e2defbba-54f2-4818-b4e4-352109346d44",
        "subnet": {
            "allocation_pools": [
                {
                    "end": "10.26.141.254",
                    "start": "10.26.141.226"
                }
            ],
            "cidr": "10.26.141.224/27",
            "created_at": "2023-11-29T09:22:46Z",
            "description": "",
            "dns_nameservers": [
                "10.56.240.213"
            ],
            "dns_publish_fixed_ip": null,
            "gateway_ip": "10.26.141.225",
            "host_routes": [],
            "id": "e2defbba-54f2-4818-b4e4-352109346d44",
            "ip_version": 4,
…
```

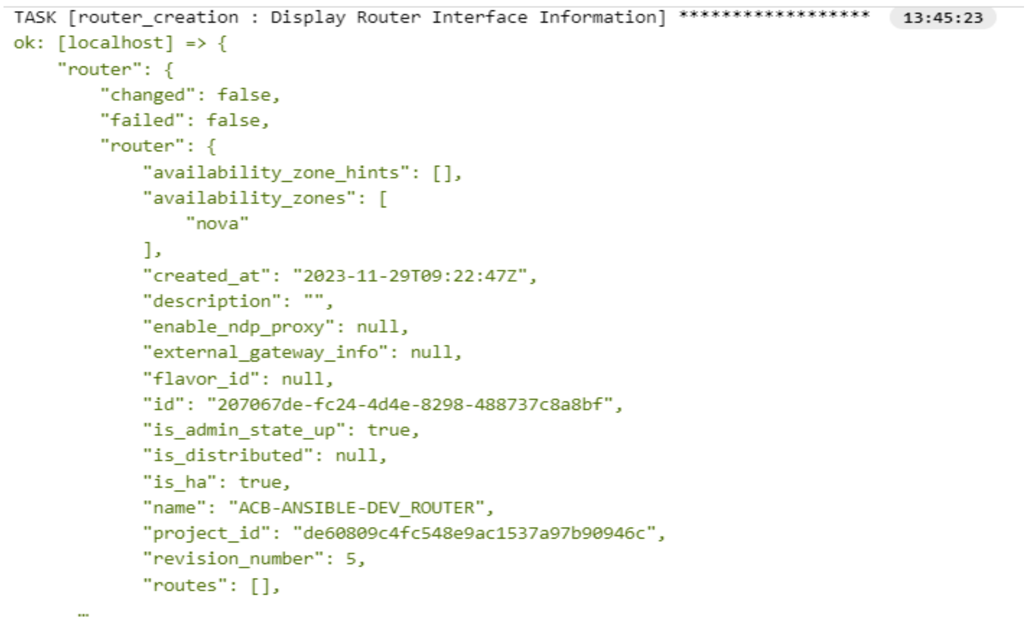
4.6 6. Router Creation

The `router_creation/main.yml` playbook creates an OpenStack router with parameters taken from the CSV input, including the project name, router name (dynamic based on CSV input), and network interfaces. The router is associated with a specified subnet, port IP, and network. The debugging task provides visibility into the result of the router creation for verification and troubleshooting. This playbook allows flexibility in specifying router names through the CSV input.

1. **Create a Router:**
   - This task creates an OpenStack router with specified parameters.
     - **`state: present`**: Ensures the router is created.
     - **`auth`**: Provides authentication details for connecting to the OpenStack environment.
     - **`name: "{{ RouterName }}"`**: Input from the csv.
     - **`project: "{{ ProjectName }}"`**: Associates the router with the specified project.
     - **`interfaces`**: Defines network interfaces for the router, including subnet, port IP, and network name.


```
TASK [router_creation : Create Router] *****************************************  13:45:22
ok: [localhost]
```

2. **Debug Router Result:**
   - This task prints debugging information about the result of the router creation, providing visibility into the router configuration.

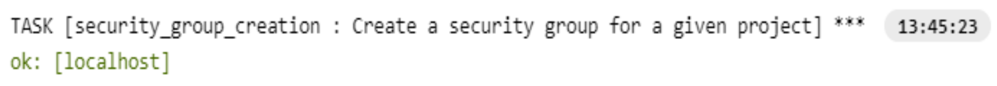


```
TASK [router_creation : Display Router Interface Information] ******************   13:45:23
ok: [localhost] => {
    "router": {
        "changed": false,
        "failed": false,
        "router": {
            "availability_zone_hints": [],
            "availability_zones": [
                "nova"
            ],
            "created_at": "2023-11-29T09:22:47Z",
            "description": "",
            "enable_ndp_proxy": null,
            "external_gateway_info": null,
            "flavor_id": null,
            "id": "207067de-fc24-4d4e-8298-488737c8a8bf",
            "is_admin_state_up": true,
            "is_distributed": null,
            "is_ha": true,
            "name": "ACB-ANSIBLE-DEV_ROUTER",
            "project_id": "de60809c4fc548e9ac1537a97b90946c",
            "revision_number": 5,
            "routes": [],
      …
```

4.7 7. Security Group Creation

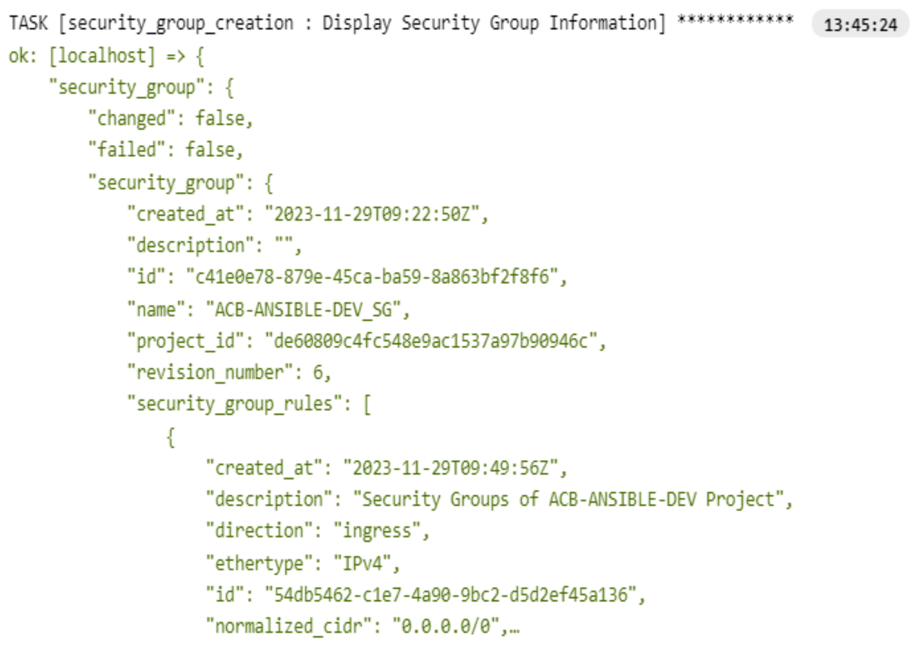
The `security_group_creation/main.yml` playbook ensures the creation of a security group in the specified project, and the debugging task provides visibility into the result of the security group creation for verification and troubleshooting purposes. Adjustments to the security group parameters can be made based on specific project requirements.

1. **Create a security group for a given project:**
    - This task uses the **`openstack.cloud.security_group`** Ansible module to create an OpenStack security group.
    - It specifies the desired state (**`state: present`**) and provides parameters such as the name of the security group (**`name: "{{ SecurityGroupName }}"`**) and the project to which it belongs (**`project: "{{ ProjectName }}"`**).



```
TASK [security_group_creation : Create a security group for a given project] ***   13:45:23
ok: [localhost]
```

2. **Display Security Group Information:**
    - This task prints debugging information about the result of the security group creation.
    - The **`debug`** module is used to display the content of the **`security_group`** variable, which contains information about the created security group.



```
TASK [security_group_creation : Display Security Group Information] ************  13:45:24
ok: [localhost] => {
    "security_group": {
        "changed": false,
        "failed": false,
        "security_group": {
            "created_at": "2023-11-29T09:22:50Z",
            "description": "",
            "id": "c41e0e78-879e-45ca-ba59-8a863bf2f8f6",
            "name": "ACB-ANSIBLE-DEV_SG",
            "project_id": "de60809c4fc548e9ac1537a97b90946c",
            "revision_number": 6,
            "security_group_rules": [
                {
                    "created_at": "2023-11-29T09:49:56Z",
                    "description": "Security Groups of ACB-ANSIBLE-DEV Project",
                    "direction": "ingress",
                    "ethertype": "IPv4",
                    "id": "54db5462-c1e7-4a90-9bc2-d5d2ef45a136",
                    "normalized_cidr": "0.0.0.0/0",…
```

4.8 8. Security Group Rule Creation

The role reads a CSV file named as ***sg-rule.csv*** from GitLab containing security group rule parameters, where each rule is entered on a separate line. It then parses the content and creates or checks the existence of security group rules accordingly.

User Input:

- Users are required to provide security group rules information in a separate CSV file named ***sg-rule.csv***
- Each rule should be entered on a separate line in the CSV file with the following format
- CSV Parameters Sequence:

    1. SecurityGroupName
    2. SecurityGroupDirection
    3. SecurityGroupProtocol
    4. SecurityGroupPort
    5. SecurityGroupSegment
    6. SecurityGroupDescription

- User Input CSV Example

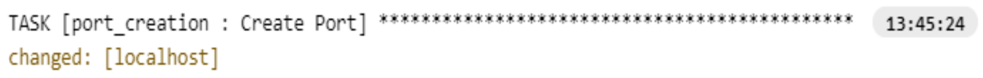
```
ACB-CLOUD-UAT_SG,Egress/Ingress,ICMP/TCP/UDP,SSH,0.0.0.0/0,Security
Groups of ACB-CLOUD-UAT Project.
```

1. **Read CSV File:**
   - Reads the content of a CSV file and registers it as binary data.
2. **Convert Binary Content to String:**
   - Decodes binary content to string format.
3. **Parse CSV Content:**
   - Splits the CSV content into lines and stores them as a list.
4. **Loop through CSV Lines:**
   - Includes tasks from create_security_group_rule.yml for each CSV line.
5. **Extract Project Parameters:**
   - Splits each CSV line into parameters and stores them in sg_rule_params.
6. **Debug Project Parameters:**
   - Displays debug information about the extracted parameters.
7. **Assign Parameters to Variables:**
   - Assigns extracted parameters to variables for security group rule creation.
8. **Check if Rule Exists:**
   - Checks if the security group rule already exists.
9. **Create Security Group Rule:**
   - Creates a security group rule if it doesn't already exist.

4.9 9. Port Creation

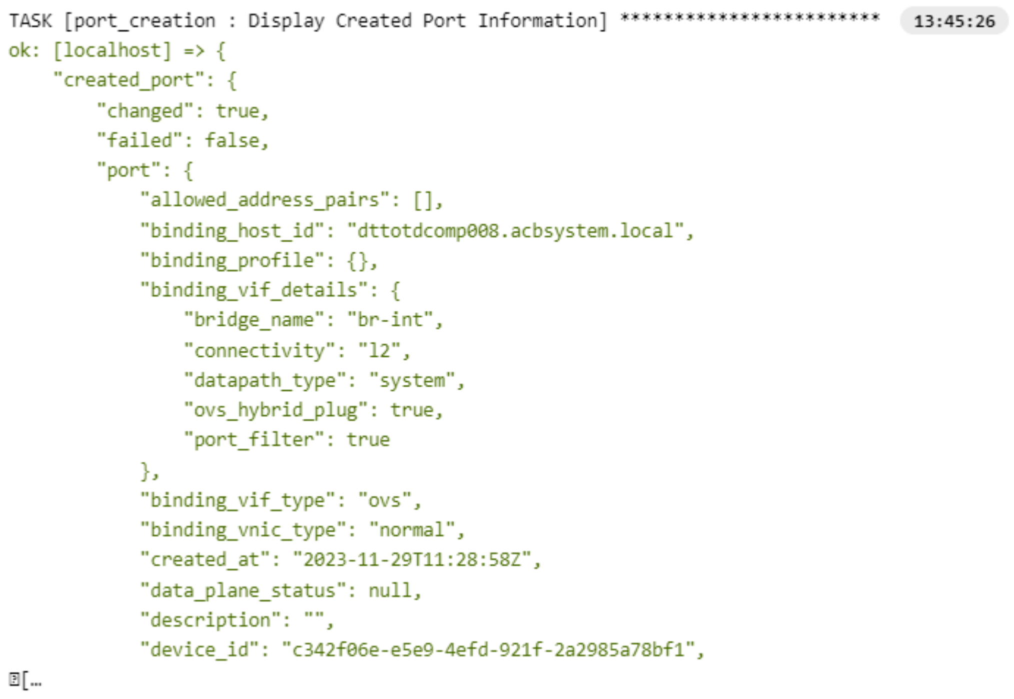
The **port_creation/main.yml** playbook creates an OpenStack port with specified parameters, including the network association, optional port name, fixed IP configurations, and assignment to a security group.

1. **Create OpenStack Port:**
   - This task creates an OpenStack port with specified parameters.
     - **state: present**: Ensures the port is created.
     - **network: "{{ NetworkSegmentName }}"**: Associates the port with the specified network.
     - **name:** Specifies the port name.
     - **fixed_ips**: Defines fixed IP configurations for the port, including subnet and IP address.
     - **security_groups**: Assigns the port to the specified security group.



```
TASK [port_creation : Create Port] ********************************************* 13:45:24
changed: [localhost]
```

2. **Display Created Port Information:**
   - This task prints debugging information about the result of the port creation, providing visibility into the port's configuration.

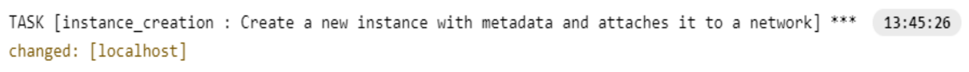


```
TASK [port_creation : Display Created Port Information] ************************  13:45:26
ok: [localhost] => {
    "created_port": {
        "changed": true,
        "failed": false,
        "port": {
            "allowed_address_pairs": [],
            "binding_host_id": "dttotdcomp008.acbsystem.local",
            "binding_profile": {},
            "binding_vif_details": {
                "bridge_name": "br-int",
                "connectivity": "l2",
                "datapath_type": "system",
                "ovs_hybrid_plug": true,
                "port_filter": true
            },
            "binding_vif_type": "ovs",
            "binding_vnic_type": "normal",
            "created_at": "2023-11-29T11:28:58Z",
            "data_plane_status": null,
            "description": "",
            "device_id": "c342f06e-e5e9-4efd-921f-2a2985a78bf1",
[…
```

4.10 10. Instance Creation

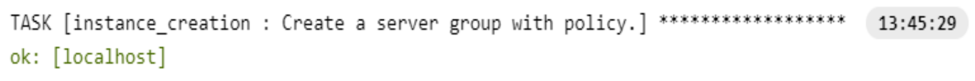
The **instance_creation/main.yml** playbook orchestrates the creation of an OpenStack virtual machine, allowing for flexibility in configurations based on the provided CSV input. It also handles the creation of a server group with an 'affinity' policy, volume creation, and volume attachment based on specific conditions. The debugging task provides visibility into the result of the virtual machine creation for verification and troubleshooting purposes.

1. **Create OpenStack VM:**
   - This task creates an OpenStack virtual machine with specified parameters, including name, description, flavor, image, key pair, network or port, security groups, and volume settings.



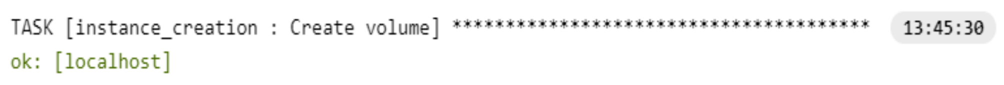
```
TASK [instance_creation : Create a new instance with metadata and attaches it to a network] ***  13:45:26
changed: [localhost]
```

2. **Create a server group policy:**
   - This task creates a server group with policy if specified in the CSV input.



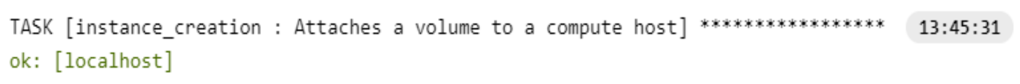
```
TASK [instance_creation : Create a server group with policy.] *****************  13:45:29
ok: [localhost]
```

3. **Create Volume:**
   - This task creates an OpenStack volume with specified parameters if ExtendOSDisk is defined in the CSV input.



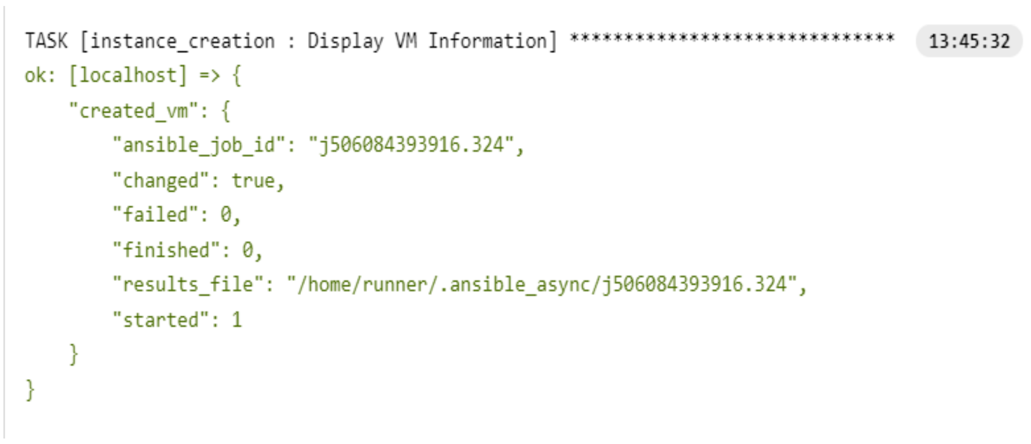
```
TASK [instance_creation : Create volume] ***************************************  13:45:30
ok: [localhost]
```

4. **Attaches a volume to a compute host:**
   - This task attaches a volume to the created virtual machine if ExtendOSDisk is defined in the CSV input.



```
TASK [instance_creation : Attaches a volume to a compute host] ****************  13:45:31
ok: [localhost]
```

5. **Display Server Creation Result:**
   - This task prints debugging information about the result of the virtual machine creation, providing visibility into the VM's configuration for verification and troubleshooting.



```
TASK [instance_creation : Display VM Information] ******************************  13:45:32
ok: [localhost] => {
    "created_vm": {
        "ansible_job_id": "j506084393916.324",
        "changed": true,
        "failed": 0,
        "finished": 0,
        "results_file": "/home/runner/.ansible_async/j506084393916.324",
        "started": 1
    }
}
```

5 Exceptions and Limitations

#	Exception	Reason	Workaround	Required
1	Project / Instance Permission	Error when adding service account to project role	Manually adding roles of service account but requires fixing for end-to-end automation	Mandatory
2	Edit Interface Name - To be discussed	Not supported by Ansible OpenStack modules	OpenStack CLI can be explored as a workaround	Optional
3	Static Routes - To be discussed	Not supported by Ansible OpenStack modules	OpenStack CLI can be explored as a workaround	Mandatory
4	Add Interface to Subnet in Different Project	Resources are project-scoped, not supported	N/A	Mandatory

#	Exception	Reason	Workaround	Required
5	Changing the Flavor of Existing Instance	Ansible operates in a stateless manner, meaning it doesn't keep track of the current state of managed systems.	N/A	Mandatory

6 Promote to Prod

Preparing for Production Execution:

1. **Ansible Credentials for OpenStack Prod:**
   - Create specific Ansible credentials to authenticate with the production OpenStack environment during playbook execution.
2. **Open Firewall Ports:**
   - Ensure all required firewall ports are open to facilitate communication between Ansible Tower and the OpenStack production environment.
3. **Ansible Tower Environment Variables:**
   - Set up environment variables in Ansible Tower to mirror the production environment. Go to settings -> job settings -> environment variables. Mirror the dev credentials for production accordingly*[this is done because we need to enable playbook to create resources in the created project by the playbook and not in the project configured in tower credentials with service account]*.

     Replace these variables in the playbook accordingly for seamless execution in the production context.

7 Debug/Troubleshooting

In the event of errors during the execution of the Ansible playbook, follow these steps to troubleshoot and trace down the issue to a specific role

1. **Review Ansible Tower Job Output:**
   - Access the Ansible Tower web interface and navigate to the specific job that encountered an error. Examine the detailed output, including task logs and error messages.
2. **Enable Verbose Mode in Ansible Tower:**
   - Adjust the verbosity level for Ansible Tower job runs. This can be configured in the job template settings or when launching a job.
3. **Debugging Statements:**
   - Utilize Ansible's `debug` module strategically within tasks to output variable values and debug information.
4. **Check Ansible Facts:**
   - Ensure that Ansible facts are correctly retrieved and stored. Debug facts using the `debug` module.
5. **Review Role Outputs:**

    - Inspect the output of each role by adding **`debug`** tasks within the roles to display relevant information.
6. **Isolate Roles:**
    - Temporarily comment out roles in the playbook to isolate and identify the specific role causing the issue. Run the playbook with a reduced set of roles.

By following these steps, you can systematically troubleshoot and debug issues in both Ansible Tower and Ansible playbooks, isolating errors and identifying the root cause.