Ansible Laboratory Setup and Basic Operations - Student Guide

Overview

This guide will walk you through setting up an Ansible laboratory environment using Docker containers and performing basic Ansible operations including SSH key management, inventory configuration, and running ad-hoc commands.

Prerequisites

- Docker and Docker Compose installed on your system
- Git installed
- Basic knowledge of Linux command line
- SSH client available

Learning Objectives

By the end of this lesson, you will be able to:

- Set up an Ansible lab environment using Docker
- Configure SSH key-based authentication between Ansible control node and managed nodes
- Create and use Ansible inventory files
- Execute basic Ansible ad-hoc commands
- Use common Ansible modules (ping, command, setup, copy)

Part 1: Environment Setup

Step 1: Clone Required Repositories

First, clone the two required repositories:

```bash
git clone git@github.com:spurin/diveintoansible-lab.git
git clone git@github.com:spurin/diveintoansible.git
```

What this does:

- `diveintoansible-lab`: Contains the Docker lab environment
- `diveintoansible`: Contains Ansible learning materials and examples

Step 2: Start the Lab Environment

Navigate to the lab directory and start the containers:

```bash
cd diveintoansible-lab
docker-compose up -d
```

Expected output: Docker will pull and start multiple containers including:

- `ubuntu-c`: The Ansible control node
- `centos1`: A managed node running CentOS
- `portal`: Web interface for accessing the lab

Step 3: Copy Learning Materials

Copy the learning materials into the lab environment:

```bash
cd ..
cp -r diveintoansible diveintoansible-lab/ansible_home/ubuntu-c/ansible
```

This copies all learning materials into the control node's home directory.

Step 4: Access the Lab Interface

Open your web browser and navigate to:

```
http://localhost:1000/
```

Login credentials:

- **Username:** ubuntu-c
- **Password:** ansible

Part 2: SSH Key Configuration

Step 5: Generate SSH Key Pair

Once logged into the ubuntu-c container, generate an SSH key pair:

```bash
ssh-keygen
```

Interactive prompts and responses:

```
Enter file in which to save the key (/home/ansible/.ssh/id_rsa): [Press Enter]
Enter passphrase (empty for no passphrase): [Press Enter]
Enter same passphrase again: [Press Enter]
```

What happens:

- Creates `/home/ansible/.ssh/id_rsa` (private key)
- Creates `/home/ansible/.ssh/id_rsa.pub` (public key)
- Uses RSA 3072-bit encryption by default

Step 6: Copy SSH Key to Managed Node

Copy your public key to the managed node:

```bash
ssh-copy-id ansible@centos1
```

Interactive prompts:

1. **Host authenticity warning:** Type `yes` to continue
2. **Password prompt:** Enter the password for ansible user on centos1

Expected output:

```
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'ansible@centos1'"
```

Step 7: Test SSH Connection

Verify passwordless SSH access:

```bash
ssh ansible@centos1
```

You should be able to log in without entering a password. Type `exit` to return to the control node.

Part 3: Basic Ansible Operations

Step 8: Navigate to Inventory Directory

Navigate to the first inventory example:

```bash
cd ~/ansible/diveintoansible/Ansible\ Architecture\ and\ Design/Inventories/01/
```

Step 9: Examine the Inventory File

View the inventory configuration:

```bash
cat hosts
```

Expected content:

```ini
[all]
centos1
```

This simple inventory defines one host (`centos1`) in the `all` group.

Step 10: Test Connectivity with Ping Module

Test if Ansible can reach the managed node:

```bash
ansible -m ping all
```

Expected output:

```json
```

```
centos1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.9"
    },
    "changed": false,
    "ping": "pong"
}
```

What this tells us:

- Connection successful (`SUCCESS`)
- Python interpreter detected automatically
- No changes made (`"changed": false`)
- Ping response received (`"ping": "pong"`)

Part 4: Running Ad-hoc Commands

Step 11: Execute Commands on Remote Hosts

Run a command on all managed nodes:

```bash
ansible all -m command -a "ls -la"
```

Expected output:

```
centos1 | CHANGED | rc=0 >>
total 32
drwxr-xr-x 4 ansible ansible 4096 Jul 27 18:35 .
drwxr-xr-x 1 root    root    4096 Jul 27 17:39 ..
drwx------ 3 ansible ansible 4096 Jul 27 18:35 .ansible
-rw------- 1 ansible ansible    5 Jul 27 18:10 .bash_history
-rw-r--r-- 1 ansible ansible   18 Feb 15  2024 .bash_logout
-rw-r--r-- 1 ansible ansible  141 Feb 15  2024 .bash_profile
-rw-r--r-- 1 ansible ansible  492 Feb 15  2024 .bashrc
drwx------ 2 ansible ansible 4096 Jul 27 18:09 .ssh
```

Command breakdown:

- `ansible all`: Target all hosts in inventory

- `-m command`: Use the command module
- `-a "ls -la"`: Arguments to pass to the module

Part 5: Additional Ansible Modules

Step 12: Gather System Facts

Collect detailed information about managed nodes:

```bash
ansible all -m setup
```

This command returns extensive JSON data about the system including:

- Operating system details
- Hardware information
- Network configuration
- Environment variables

Step 13: Copy Files to Remote Hosts

Create a test file and copy it to managed nodes:

```bash
echo "Hello from Ansible" > 1.txt
ansible all -m copy -a "src=./1.txt dest=~/1.txt"
```

Expected output:

```json
```

```
centos1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.9"
    },
    "changed": true,
    "checksum": "...",
    "dest": "/home/ansible/1.txt",
    "gid": 1000,
    "group": "ansible",
    "md5sum": "...",
    "mode": "0664",
    "owner": "ansible",
    "size": 18,
    "src": "/home/ansible/.ansible/tmp/.../source",
    "state": "file",
    "uid": 1000
}
```

Part 6: Cleanup

Step 14: Stop the Lab Environment

When finished with the lab, stop the containers:

```bash
cd ~/diveintoansible-lab/
docker-compose down
```

Key Concepts Review

Ansible Architecture Components

1. **Control Node** (`ubuntu-c`): Where Ansible is installed and commands are executed
2. **Managed Nodes** (`centos1`): Target systems that Ansible manages
3. **Inventory**: File defining which hosts Ansible manages
4. **Modules**: Reusable units of work (ping, command, setup, copy)

Important Files and Directories

- `/home/ansible/.ssh/`: SSH keys and configuration

- `~/ansible/`: Learning materials and examples
- `hosts`: Inventory file defining managed nodes

Command Patterns

```bash
# Basic syntax
ansible <target> -m <module> -a "<arguments>"

# Examples
ansible all -m ping                    # Test connectivity
ansible all -m command -a "uptime"     # Run commands
ansible all -m setup                   # Gather facts
ansible all -m copy -a "src=file dest=path"  # Copy files
```

Troubleshooting

Common Issues

SSH Connection Problems:

- Verify SSH key was copied correctly: `ssh ansible@centos1`
- Check if containers are running: `docker-compose ps`
- Ensure inventory file exists and contains correct hostnames

Permission Denied:

- Verify you're running commands as the `ansible` user
- Check SSH key permissions: `ls -la ~/.ssh/`

Module Not Found:

- Ensure you're in the correct directory with inventory file
- Verify Ansible is installed: `ansible --version`

Next Steps

After completing this lab, you should explore:

1. Creating more complex inventory files with groups
2. Writing Ansible playbooks instead of ad-hoc commands

3. Using variables and templates
4. Implementing proper error handling and conditionals
5. Working with Ansible roles for code organization

Summary

You have successfully:

- ✅ Set up an Ansible lab environment with Docker
- ✅ Configured SSH key-based authentication
- ✅ Created and used basic inventory files
- ✅ Executed Ansible ad-hoc commands
- ✅ Used common Ansible modules (ping, command, setup, copy)
- ✅ Understood basic Ansible architecture and workflow

This foundation prepares you for more advanced Ansible topics including playbooks, roles, and automation workflows.