Ansible Part 2 - Student Guide

Lab Setup and Prerequisites

Environment Setup

Before starting the exercises, you need to set up the lab environment using Docker containers.

Step 1: Clone Required Repositories

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

Step 2: Start Lab Environment

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

Step 3: Access the Lab

- Open your browser and navigate to: `http://localhost:1000/`
- Login credentials:
  - **Username**: `ansible`
  - **Password**: `password`

Step 4: Navigate to Working Directory

```bash
cd diveintoansible/
```

Exercise 1: Ansible Playbooks Introduction

Learning Objectives

- Understand Ansible playbook structure
- Learn about inventory configuration

- Practice running playbooks with multiple host groups
- Troubleshoot connectivity issues

Step 1: Navigate to Exercise Directory

```bash
cd "Ansible Playbooks, Introduction/Ansible Playbooks, Breakdown of Sections/07"
```

Step 2: Examine Configuration Files

Ansible Configuration (`ansible.cfg`)

```ini
[defaults]
inventory = hosts
host_key_checking = False
```

Key Points:

- Sets the default inventory file
- Disables SSH host key checking for lab environment

Inventory File (`hosts`)

```ini
```

```
[control]
ubuntu-c ansible_connection=local

[centos]
centos1 ansible_port=2222
centos[2:3]

[centos:vars]
ansible_user=root

[ubuntu]
ubuntu[1:3]

[ubuntu:vars]
ansible_become=true
ansible_become_pass=password

[linux:children]
centos
ubuntu
```

Key Concepts:

- **Host Groups**: Logical grouping of servers (`centos`, `ubuntu`, `control`)
- **Group Variables**: Common settings for all hosts in a group
- **Host Ranges**: `centos[2:3]` expands to `centos2`, `centos3`
- **Parent Groups**: `linux:children` creates a parent group containing both `centos` and `ubuntu` groups

Step 3: Set Up SSH Keys

Before running playbooks, establish SSH connectivity:

```bash
ssh-copy-id root@centos1
ssh-copy-id root@centos2
ssh-copy-id root@centos3
ssh-copy-id ansible@ubuntu1
ssh-copy-id ansible@ubuntu2
ssh-copy-id ansible@ubuntu3
```

Step 4: Run the MOTD Playbook

```bash
ansible-playbook motd_playbook.yaml
```

Expected Output Analysis

- **UNREACHABLE**: `centos1` connection failed (port 2222 issue)
- **OK**: Successful connections to other hosts
- **CHANGED**: Tasks that modified the system
- **SKIPPED**: Tasks not applicable to certain hosts

Step 5: Fix Connectivity Issue

Problem: `centos1` is configured with port 2222, but should use port 22 **Solution**: Edit the inventory file to change `ansible_port=2222` to `ansible_port=22` for centos1

Understanding the Play Recap

```
PLAY RECAP
centos1   : ok=0  changed=0  unreachable=1  failed=0  skipped=0  rescued=0  ignored=0
centos2   : ok=3  changed=1  unreachable=0  failed=0  skipped=1  rescued=0  ignored=0
```

- **ok**: Successful tasks
- **changed**: Tasks that modified the system
- **unreachable**: Hosts that couldn't be contacted
- **failed**: Tasks that failed
- **skipped**: Tasks that were skipped due to conditions

Exercise 2: Ansible Variables

Learning Objectives

- Explore different types of Ansible variables
- Understand variable precedence
- Practice with variable examples

Step 1: Navigate to Variables Directory

```bash
```

```
cd "../Ansible Playbooks, Variables"
```

Step 2: Explore Variable Examples

```bash
./show_examples.sh
```

Key Variable Types:

- **Host Variables**: Specific to individual hosts
- **Group Variables**: Applied to all hosts in a group
- **Play Variables**: Defined within playbooks
- **Extra Variables**: Passed via command line

Variable Precedence (highest to lowest):

1. Extra vars (command line `-e`)
2. Task vars
3. Block vars
4. Role and include vars
5. Play vars
6. Host facts
7. Host vars
8. Group vars
9. Role defaults

Exercise 3: Blocks and Error Handling

Learning Objectives

- Understand Ansible blocks for task organization
- Learn error handling with rescue and always sections
- Practice with group_vars and host_vars directories

Step 1: Navigate to Blocks Directory

```bash
```

```
cd ~/diveintoansible/"Ansible Playbooks, Deep Dive"/Blocks/03
```

Step 2: Examine Configuration Structure

Enhanced Ansible Configuration

```ini
[defaults]
inventory = hosts
host_key_checking = False
forks=6
```

New Setting: `forks=6` - Allows Ansible to run tasks on up to 6 hosts simultaneously

Simplified Inventory

```ini
[control]
ubuntu-c

[centos]
centos[1:3]

[ubuntu]
ubuntu[1:3]

[linux:children]
centos
ubuntu
```

Step 3: Understand Variable File Structure

Group Variables

- `group_vars/centos`: Variables for all CentOS hosts
- `group_vars/ubuntu`: Variables for all Ubuntu hosts

Host Variables

- `host_vars/centos1`: Variables specific to centos1
- `host_vars/ubuntu-c`: Variables specific to ubuntu-c

Step 4: Modify Host Variables

Remove the problematic port configuration:

```bash
rm host_vars/centos1
```

Step 5: Run the Blocks Playbook

```bash
ansible-playbook blocks_playbook.yaml
```

Understanding Block Structure

A typical block structure includes:

- **block**: Main tasks to execute
- **rescue**: Tasks to run if block tasks fail
- **always**: Tasks that always run, regardless of success/failure

Output Analysis

- CentOS hosts fail on `python3-dnspython` installation
- Rescue tasks execute automatically
- Always tasks run on all hosts regardless of previous task results

Exercise 4: Looping

Learning Objectives

- Understand Ansible looping mechanisms
- Practice with until loops for conditional execution

Step 1: Navigate to Looping Directory

```bash
cd ~/diveintoansible/"Ansible Playbooks, Deep Dive"/Looping/21
```

Step 2: Run Until Loop Playbook

```bash
ansible-playbook until_playbook.yaml
```

Common Loop Types

- **loop**: Simple iteration over a list
- **with_items**: Legacy loop method (still supported)
- **until**: Retry tasks until a condition is met
- **with_dict**: Loop over dictionary key-value pairs
- **with_fileglob**: Loop over files matching a pattern

Key Concepts Summary

Ansible Playbook Structure

```yaml
---
- name: Playbook Description
  hosts: target_group
  become: yes
  vars:
    variable_name: value
  tasks:
    - name: Task Description
      module_name:
        parameter: value
      when: condition
```

Best Practices

1. **Use descriptive names** for plays and tasks
2. **Organize variables** using group_vars and host_vars
3. **Handle errors gracefully** with blocks, rescue, and always
4. **Use loops efficiently** to avoid repetitive tasks
5. **Test connectivity** before running complex playbooks

Troubleshooting Tips

1. **Check connectivity**: Use `ansible all -m ping` to test host reachability

2. **Verify inventory**: Ensure host groups and variables are correctly defined
3. **Review logs**: Ansible provides detailed output for debugging
4. **Use verbose mode**: Add `-v`, `-vv`, or `-vvv` for more detailed output
5. **Test incrementally**: Run tasks step by step when developing playbooks

Common Inventory Patterns

```ini
# Range notation
webservers[1:5]  # webservers1, webservers2, webservers3, webservers4, webservers5

# Alphabetic ranges
db[a:c]  # dba, dbb, dbc

# Group variables
[webservers:vars]
http_port=80
ssl_port=443

# Parent/child groups
[production:children]
webservers
databases
```

Next Steps

After completing these exercises, you should be comfortable with:

- Writing and executing Ansible playbooks
- Managing inventory and variables
- Implementing error handling with blocks
- Using loops for repetitive tasks

Continue practicing by creating your own playbooks and experimenting with different modules and configurations.