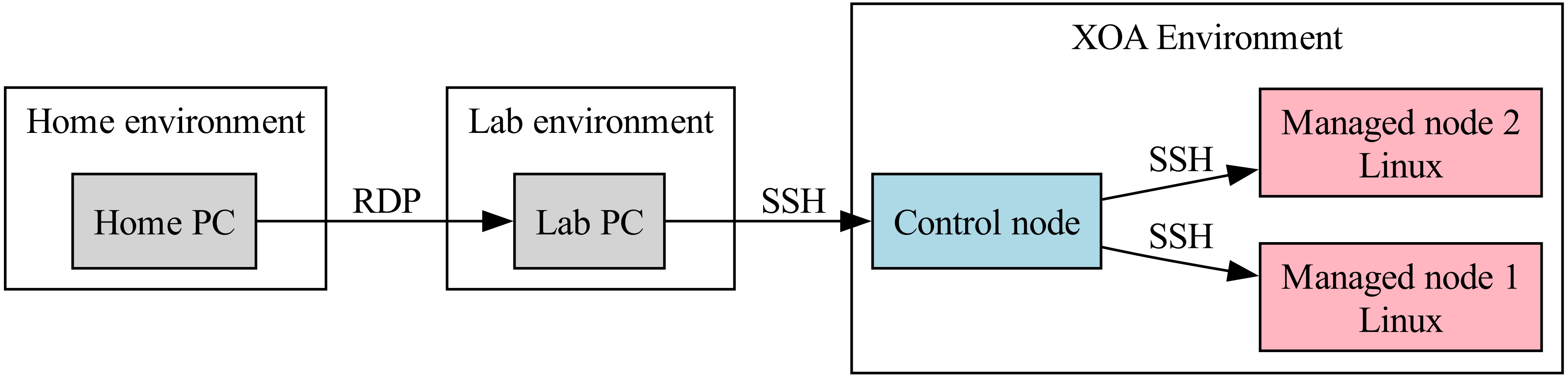
Ansible

Dr Peadar Grant

October 22, 2024

Contents

1 Scenario


XOA Environment
Home environment
Lab environment
Home PC
RDP
Lab PC
SSH
Control node
SSH
Managed node 2
Linux
SSH
Managed node 1
Linux

1.1 Components

Control node : System where we will be automating things frmo.

- So far you could issue SSH commands from here.
- Today we will run Ansible commands such as `ansible` or `ansible-inventory` on the control node.

Managed node : Remote system, or host, that we want to control.

- Commands etc will be run here remotely over SSH.
- May be physical, virtual, cloud VM, SBC like R-Pi etc.
- Commands will be issued today by Ansible system.

Inventory : List of managed nodes that are logically organized.

- Can create inventory on the control node to describe host deployments to Ansible.

1.2 Set-up

To save lab time, let’s start the set up the 3 machines now (Task 1)!

1.3 Common automation tasks

1. Package update
2. Software installation
3. Service configuration (enabled / disabled)
4. Local user account management (creation, deletion)
5. Local group management (creation, deletion, membership)
6. Configuration file maintenance

2 Ansible

Ansible is an open-source automation solution that reduces complexity and runs on a variety of operating systems.

Common use cases for Ansible

- Eliminate repetition and simplify workflows
- Manage and maintain system configuration
- Continuously deploy complex software
- Perform zero-downtime rolling updates

2.1 Design principles

1. **Agent-less architecture:** Low maintenance overhead by avoiding the installation of additional software across IT infrastructure.
2. **Simplicity:** Automation playbooks use straightforward YAML syntax for code that reads like documentation. Ansible is also decentralized, using SSH with existing OS credentials to access to remote machines.
3. **Scalability and flexibility:** Easily and quickly scale the systems you automate through a modular design that supports a large range of operating systems, cloud platforms, and network devices.
4. **Idempotence and predictability:** When the system is in the state your playbook describes Ansible does not change anything, even if the playbook runs multiple times.

2.2 Agentless architecture

Ansible is **agentless**:

- Some automation solutions require an *agent* to be running on the managed node(s).
- Ansible uses SSH to let the control node take actions on the managed node.
- Once you can make an SSH connection from the control to the mangaged node you can run ansible commands on it.

Pre-requisitites

The agentless architecture practically requires the following:

1. We can make an SSH connection from the control to the managed node.
2. The SSH connection can happen without a password (i.e. with keys)
3. The user we connect as can run commands as root without a password
4. The target host is powered on and connected. (Consider Ansible Pull for Desktops)

3 Inventory

Hosts managed by ansible are listed in the *inventory*:

- Simplest inventory is a text file.
    - Default location `/etc/ansible/hosts`
    - Can put elsewhere using the `-i` flag
- Can integrate ansible with other data sources to supply inventory
    - Can even write your own in code

Groups allow us to select multiple hosts at the same time.

3.1 Inventory file example

```
[managed]
10.108.154.105
10.108.156.198
```

4 Playbooks

Ansible automation tasks are defined in **Playbooks**, in YAML format:

Playbook: text file containing multiple *plays*

Play: executes part of the overall goal of the playbook, running one or more *tasks*.

Task: Each task calls an Ansible **module**

Ansible modules are prepackaged functionality in Ansible that avoids us having to script simple operations (e.g. package installation)

Each play must define

1. The managed **nodes** to target
2. At least one **task** to execute

4.1 Idempotency

Most Ansible modules

1. Check whether the desired final state has already been achieved, and
2. Exit without performing any actions if that state has been achieved,
    - So that repeating the task does not change the final state.

Modules that behave this way are often called “idempotent”:

- Whether you run a playbook once, or multiple times, the outcome should be the same.
- However, not all playbooks and not all modules behave this way.