# Using Ansible to Deploy LVM

**Andrew Mallett**
LINUX AUTHOR AND TRAINER

@theurbanpenguin   www.theurbanpenguin.com

## Overview

What is Ansible

Installing Ansible on the controller

Delivering SSH keys with Ansible

Configure vim for YAML files

Configuring sudo with Ansible

Ensuring required packages and services are in the desired state

## Ansible

Ansible is a clientless configuration management system. We only need to install on one node to manage others securely using SSH.

```
# On controller
$ sudo apt update; sudo apt install ansible sshpass
$ ansible --version
# On all nodes if required
$ sudo apt update; sudo apt install python
```

## Ansible is Clientless

**It only needs to be installed on one system. Python needs to be installed on all nodes.**

```
# On controller

$ for h in 192.168.56.152 192.168.56.153; do

  ssh-keyscan $h | sudo tee -a /etc/ssh/ssh_known_hosts

done
```

## Gather SSH Keys From Managed Nodes

**Ansible needs to be as automated as possible. The controller will need the public keys from the SSH Servers it will connect to.**

```
# On controller

$ mkdir lvm ; cd lvm

$ sed -E '/^($|#)/d' /etc/ansible/ansible.cfg > ansible.cfg

$ vim ansible.cfg
[defaults]
inventory = ./inventory

$ vim inventory
192.168.56.[152:153]
```

## Create Configuration and Inventory

**It is not good practice to use the central Ansible configuration and inventory. We can create custom setups for each project.**

## Demo

We will install Ansible on the controller node.

Collecting public keys from the managed nodes we can SSH to them.

An initial configuration consists of the ansible.cfg and inventory file.

```
# On controller

$ ansible all -k -m ping
```

## Test the Basics

Using ad-hoc commands we can quickly test the configuration. The ping module in Ansible test for a response from Python on the managed nodes.

```
# On controller

$ ssh-keygen -t rsa
…
$ ansible-doc authorized_key

$ ansible all -k -m authorized_key -a "user=tux state=present \
key={{ lookup('file', '/home/tux/.ssh/id_rsa.pub') }}"
```

## Deploy User Public Keys

**Ideally we don't want to enter passwords for SSH or sudo if we need to elevate privileges. First we look at deploying the user keys.**

## Demo

We will learn to deploy user keys with an Ansible ad-hoc command

## Ansible Playbook

Playbooks contain repeatable steps to meet the desired configuration. They are written in YAML and we can help by making vim work well with .yml files.

## VIM Configuration

~/.vimrc

```
set bg=dark

autocmd FileType yaml setlocal ai ts=2 sw=2 \
et cuc
```

## Sudoers Configuration

The following Playbook deploys a sudoers file so our user is not prompted for the sudo password when escalating privileges.

## Playbook

~/lvm/site.yml

```
---
- name: Name of Play
  hosts: all
  become: true
  tasks:
    - name: Name of Task
      copy:
        dest: /etc/sudoers.d/tux
        content: 'tux ALL=(ALL) NOPASSWD: ALL'
...
```

```
# On controller

$ ansible-playbook sudo.yml --syntax-check

$ ansible-playbook -K sudo.yml
```

## Deploy Playbook

**The ansible-playbook command can both syntax check the file and deploy the Playbook.**

## Demo

We will now configure vim for YAML files.

Then we can create our first Playbook to deploy a sudoers file.

## Demo

Adding to the Playbook we will ensure the package lvm2 is installed and the meta-data service is running.

## Overview

Install Ansible and sshpass on controller

Create ansible.cfg and inventory file

Collect SSH public keys from servers

Deploy user SSH key to servers using ad-hoc command

Create YAML Playbook to deploy sudoers file, ensure lvm2 is installed and meta-data service is both running and enabled

## Configuring Storage for LVM