Getting started with Ansible

Contents

Be sure you connect via ssh to your MacMini as the nsrc user. Don't become root.

Starting with a fresh Ubuntu 12.04.x install, we will set up ansible to get the workshop server to configure itself.

1 Install packages

Add the ansible PPA to get a more recent version of ansible. Hit Enter when prompted.

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:rquillo/ansible
sudo apt-get update
sudo apt-get install ansible git
```

Make a note of the ansible version installed. These scripts were tested using ansible=1.3.3-precise1. Although the author strives for backwards compatibility, sometimes later versions may change how scripts work.

2 Download the workshop-kit files

This should be done as the non-root (nsrc) user, in your home directory. How you do it depends on whether you have been given full access to the git repository. For purposes of class we will use git over http with passwords.

```
cd
git clone http://trainers@wsnoc.nsrc.org:8000/nsrc/workshop-kit.git
```

When prompted for a password use the one given in class.

3 Configure ansible

`sudo editor ``/etc/ansible/ansible.cfg`` and set:`

Make the following changes to the file:

```
hostfile       = ./hosts.local
...
host_key_checking = False
...
nocows = 1
```

Finally, in your checkout's ansible directory, copy `hosts.sample` to `hosts.local`

```
cd
cd workshop-kit/ansible
cp hosts.sample hosts.local
```

4 First run: network reconfiguration

Still inside the ansible directory, run the following commands:

```
sudo ansible-playbook networking.yml --check --diff     # dummy run
sudo ansible-playbook networking.yml                    # live run
```

With the `--check` flag it will show you what it is going to change; without the `--check` flag it will actually perform the changes. Try both.

The playbook is *idempotent*. This means it is safe to run it repeatedly; anything which is already in the correct state will not be changed. Feel free to run it again.

After the live run, have a look at the following files:

- /etc/hostname
- /etc/hosts
- /etc/network/interfaces
- /etc/iptables/rules.v4

It should have created a more complex configuration with a bridge interfaces for the LAN and WAN connections (br-lan contains eth0, br-wan contains eth1), an IP alias on 10.10.0.254, and NAT rules.

If you are happy with this configuration, reboot your server to activate the new interfaces.

```
sudo shutdown -r now
```

You should not need to re-run `networking.yml` again unless you want ansible to reconfigure your network.

4.1 How does it work?

Have a look at the playbook:

```
cd workshop-kit/ansible
cat networking.yml
```

Notice how it contains:

- The host(s) or group(s) on which to run this playbook

- What tasks or roles to apply
- Tags, to allow parts of the playbook to be run selectively

Have a look at the tasks and handlers contained under each role. Handlers are extra actions which are triggered at the end of the run if a task has changed something.

```
cat roles/update_cache/tasks/main.yml
cat roles/ansible_base/tasks/main.yml
cat roles/networking_ubuntu/tasks/main.yml
cat roles/networking_ubuntu/handlers/main.yml
cat roles/networking_ubuntu/templates/hostname
cat roles/gateway_ubuntu/tasks/main.yml
cat roles/gateway_ubuntu/handlers/main.yml
```

Some of these roles and templates are quite complex - don't worry about the details.

There are also variables which are set per host and group. You can find these in the inventory (hosts) and in files under host_vars/ and group_vars/

```
cat hosts.local
cat host_vars/s1.ws.nsrc.org
cat group_vars/vm_servers
cat group_vars/all
```

In particular, notice that there are variables in `host_vars/s1.ws.nsrc.org` which are used by the interfaces template:

```
gateway_wan_interface: br-wan
interfaces:
  br-lan:
    bridge_ports: [eth0,tap11,tap12,tap13,tap14,tap15,tap16,tap17,tap18,tap19]
    address: 10.10.0.241
    aliases: [10.10.0.254]
  br-wan:
    bridge_ports: [eth1]
    address: dhcp
```

If you change these, you can generate a new `/etc/network/interfaces` with different interfaces.

5 Second run: platform configuration

```
sudo ansible-playbook vm_servers.yml -t platform
```

This installs a few basic packages. If your machine is a Mac Mini then it will install packages specific to that platform, e.g. macfanctld.

`-t platform` means only to run those tasks labelled with tag "platform".

Some steps may take a a while to complete. If you see this step hang for a very long time:

TASK: [install macmini packages] **

You may want to press "ctrl-c" and run the command again. It's possible there is a dialogue we have not accounted for that has hung the process.

6 Third run: full configuration

Now run the vm_servers.yml playbook again, but without restricting to the platform tag.

```
sudo ansible-playbook vm_servers.yml
```

This will run through a full set of configuration including:

- dns server
- ntp server
- snmp agent
- apt-cacher
- dhcp server
- kvm
- vmbuilder
- dynamips

This may take a fair amount of time to complete.

At this point you should have functioning DHCP on your LAN network. If you had configured a static IP on your laptop, you no longer need it.

You can look around and see how your MacMini has changed. For instance, you are now running an snmp server:

```
snmpwalk -v2c -c NetManage localhost
```

DNS for your private network that you will be using has been configured:

```
dig pc10.ws.nsrc.org
dig sw.ws.nsrc.org
```

You might to consider looking at current running processes, use netstat to see your current routes or open ports, typing ifconig to see all your interfaces, etc... to get a feel for your new MacMini environment.

Note that you can now use your wireless access point to connect to your MacMini. On your laptop connect to the SSID for your group (KITX-24 or KITX-5 where X={1..6}). You will receive an address on the 10.10.0.0/24 network with a gateway of 10.10.0.254, which is your MacMini, a DNS server of 10.10.0.241 (again, your MacMini) and you will have full access to the public internet.

7 Appendix

7.1 Pointing your apt to your local apt-cacher instance

Your box is now acting as a cache for installation of software using apt and is available for use for any machine on your private network (10.0.0.0/8). However, we have not set your own MacMini to use the cache that it is running. You can do this if you wish, but remember, if you must troubleshoot issues with apt, then you will probably need to disable this by removing/moving the file we will create below.

If you wish to point to your local apt-cacher program do the following:

```
sudo editor /etc/apt/apt.conf.d/01proxy
```

and add the line:

```
Acquire::http::Proxy "http://127.0.0.1:3142";
```

Now update your local apt database:

```
sudo apt update
```

The next time you use apt to install a package it will first see if it is available locally and use that copy if it's available. Otherwise, the package will still be downloaded over the network, but it will now be available for all other users who may wish to install the software.

7.2 Using git

Detailed instructions for using git with ssh and configuring it for use for interactive work flow.

This assumes you have an account on git.nsrc.org and your ssh public key has been installed there.

```
cd
git clone ssh://YOURUSERNAME@git.nsrc.org/usr/local/repositories/workshop-kit.git
cd workshop-kit
```

If authentication is rejected, you should NOT copy your ssh private key onto your workshop server! Rather you should:

1. Disconnect your ssh session
2. Log back in using ssh and agent forwarding
   - For Linux and OSX:
     - ssh-add
       - enter your passphrase when prompted
     - ssh -oForwardAgent=yes nsrc@x.x.x.x
   - For Windows/putty:
     - run pageant
     - point it to your private key and enter your passphrase
     - connect with agent forwarding enabled

7.2.1 Configure git

Create a file `~/.gitconfig` containing the following. This ensures that any commits you make are labelled with your correct details.

```
[user]
        name = Your Fullname
        email = yourname@yourdomain
[core]
        excludesfile = ~/.gitignore
        #editor = /usr/bin/joe   << or whatever you prefer
[push]
        default = tracking
```

And create `~/.gitignore` as follows: this is to minimise the junk which is picked up.

```
*~
```

8 Obtaining workshop kit files with tarballs

8.1 From a tarball

Alternatively, you may be given a tarball or zipfile containing a snapshot of the repository. Download it using wget, and extract it as appropriate:

```
cd
wget http://...../workshop-kit.tgz
tar -xvzf workshop-kit.tgz

or:
wget http://...../workshop-kit.zip
unzip workshop-kit.zip
```

Either way, you should have a directory called "workshop-kit" which you can cd into.