Systematic Experimentation (with Ansible)

1. Design the Experiment

For our experiment we want two types of node: servers and clients.

The "server" nodes will run an apache web server accessible via the public Internet. The server will also use nmap to port scan the IP space used by the dataplane topology.

The "client" nodes will just respond to the ping generated by the nmap port scanner.

To start we will create a single "server" node and a single "client" node in a way that we like. We can then replicate these two representative nodes to generate larger topologies.

2. Establish the Environment

To run this exercise, you will need two pieces of software. If you haven't already, get or install these now:

i. `omni` installed on your local machine (instructions), and
ii. `ansible` installed on your local machine (find the instructions for your package manager here).

> Windows users should have done the following steps before they reserved their resources.
>
> Before reserving their resources, Windows users should have followed the instructions for setting up a separate GENI node for running Ansible.

Resources:

- Ansible Resources:
  - A third party Getting Started with Ansible walk through
  - Ansible Module Documentation

3. Obtain Resources

3.1 Create Representative "Server" and "Client" Node Types

a. Create a new slice.
b. Click the Add Resources button to open the Add Resources page.
c. Create a representative "server" node. We will copy this node to make other servers.
   i. Drag a "Xen VM" onto the canvas (this is the multicolored icon on the left).
   ii. Click on the node to display the configuration panel on the left. The remaining items can all be set in this panel.
   iii. Set the name of the node to "server". The hostname of all of the servers will start with this string.
   iv. Set the Icon for this node to be a "Server". By setting the icon, it makes it easier to visually identify the node types.
   v. Give the node a "Publicly Routable IP". This will allow users to access the webserver via the public Internet.
   vi. Set the Disk Image to "Ubuntu 12.04 LTS". While there are defaults, it is usually good to configure the OS image so you know what you are getting. In this case, we know how to configure a webserver and nmap on an Ubuntu disk image so we choose the OS appropriately.

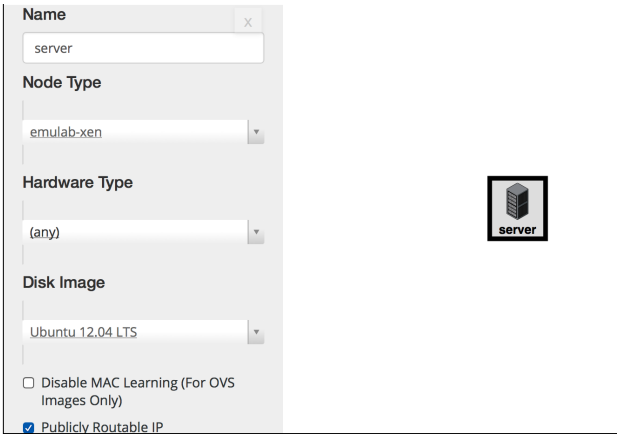

Figure 3-1 *Configuring the server node.*

d. Create a representative "client" node. We will copy this node to make other clients.
    i. Drag a default VM onto the canvas (this is the black icon on the left) and configure the fields described below.
    ii. Set the name of the node to "client".
    iii. Set the Icon for this node to a "Node".
e. Draw a link between the two nodes.
f. Save the RSpec with these two representatives node types to a file on your local machine. For a real experiment, you would commit this file to a version control system for safe keeping.

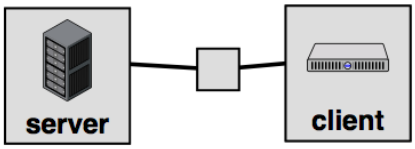

Figure 3-2 *The server and client nodes.*

For a real experiment, we would now define the IP addresses on the dataplane interfaces, reserve the topology, and write and test the automated configuration of the nodes for your experiment. The idea is that by taking the time to properly automate the configuration of these two nodes, we will be able to build larger topologies easily. This step may take some time, but in exchange it will make running the rest of the experiment quicker and easier and save you a great deal of frustration. For today's exercise, this work has already been done and tested for you.

.

Using these well defined node types, you can create a wide array of custom topologies.

3.2 Duplicate Node Types to Create a Custom Topology

Next we will define a simple topology using the representative node types defined above.

a. Duplicate the "server" node once. To copy a node click on the node and select "Duplicate Nodes Only".
b. Duplicate the "client" node twice.
c. Connect each remaining node to the original link (VLAN) to create a multipoint VLAN. To do this, click near each node and drag towards the small unlabeled box on the existing link and then release. You should now have a total of five nodes on a single multipoint VLAN.
d. Set the IPs on the data plane links by pressing the "Auto-IP" button. Click on the unlabeled box defining the link to see what IP addresses were assign. Take note of what IP subnet was assigned as we will need it later.
e. Select a site and reserve your resources.

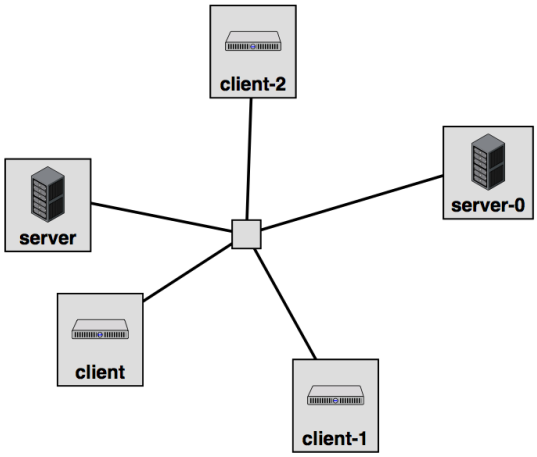

Figure 3-3 *The full topology.*

For a real experiment, we would download this RSpec and save it in a version control system.

What other topologies could you create using these two node types? What other node types would be helpful in creating more complex topologies?

Next we will install and configure software on our nodes.

Introduction

Next: Execute

Systematic Experimentation (with Ansible)

4. Configure and Initialize

4.1. Download the Ansible playbook

a. Download the Ansible playbook, webpages, etc needed to configure the nodes.

Use `wget` to download the tarball of files onto your local machine and use `tar` to uncompress it:

```
mkdir ansible
cd ansible
wget http://www.gpolab.bbn.com/exp/scalingup/ansible/ansible.tar.gz
tar zxvf ansible.tar.gz
```

4.2. Create the Ansible inventory file

`omni` comes with a script, `readyToLogin` which finds the login information for nodes in your slice. As of `omni` version 2.8, `readyToLogin` has an `--ansible-inventory` flag which generates the Ansible inventory, which is a flat file which tells Ansible the name and login information for your nodes.

a. Create your Ansible inventory file:

On your local machine:

```
$ readyToLogin MYSLICE --useSliceAggregates --ansible-inventory -o
$ cat inventory
```

Example output of running these commands:

```
$ readyToLogin MYSLICE --useSliceAggregates --ansible-inventory -o
Host info saved in inventory file: /Users/jdoe/projects/GENI/hellogeni/inventory

$ cat inventory
host-2  ansible_ssh_host=pc2.instageni.stanford.edu  ansible_ssh_port=31291
host-1  ansible_ssh_host=pc2.instageni.stanford.edu  ansible_ssh_port=31290
server-1  ansible_ssh_host=pcvm2-33.instageni.stanford.edu
rt-1  ansible_ssh_host=pc2.instageni.stanford.edu  ansible_ssh_port=31292
```

Windows users should copy their 'inventory' file onto their node running the ansible client.

b. Be sure your private key has been added to your SSH agent:

```
ssh-add /path/to/your/private/key
```

c. Check to see if your nodes are up and ready.

This command uses the `ping` module to ping the specified nodes (in this case `all`) listed in the inventory file:

```
$ ansible -i inventory all -m ping
```

Example output showing all of the nodes responding to ping:

```
$ ansible -i inventory all -m ping
server-1 | success >> {
    "changed": false,
    "ping": "pong"
}

host-1 | success >> {
    "changed": false,
```

```
    "ping": "pong"
}

rt-1 | success >> {
    "changed": false,
    "ping": "pong"
}

host-2 | success >> {
    "changed": false,
    "ping": "pong"
}
```

d. Try using the ping module in Ansible to only ping `server-1` or `host-1` by replacing `all` in the above with `server-1` or `host-1`.

4.3. Configure the nodes

Ansible commands can be collected into files called *Playbooks*. Playbooks are in a configuration file format called *YAML* which is very straightforward. In particular, Ansible Ad Hoc commands easily map to commands used in an Ansible Playbook.

The Playbook to configure the `server` node is in `server.yml`. It links to other files. For example, the code to tell the `server` node to run an `nmap` scan and post the results is in `roles/nmap/tasks/map.yml` and looks as follows:

```
---
   - name: map network using nmap
     command: nmap -sP -oX {{ nmap_xml_file }} {{ address_range }}
   - name: convert nmap xml to html
     shell:  xsltproc /usr/share/nmap/nmap.xsl {{ nmap_xml_file }} > {{ nmap_html_file }}
   - name: create directory for nmap logs in WEB_ROOT/nmaplogs with permissions of 755
     file: >
        dest={{ WEB_ROOT }}/{{ nmap_dir }}
        state=directory
        mode=755
   - name: copy nmap html file to a public place
     command: mv {{ nmap_html_file }} {{ WEB_ROOT }}/{{ nmap_dir }}/nmap.html removes={{ nmap_ht
```

a. Edit the file `group_vars/all.yml` so that the `address_range` variable uses the IP subnet from your topology. It should look like this:

```
address_range: 10.10.1.1-10
```

b. Run the playbook to configure the `server` with the following command on the local machine:

```
ansible-playbook server.yml -i inventory --limit server
```

c. Browse to hostname of the server node in your browser. Click on the `nmap` link.
d. If this looks ok, run the following to install the code on both of your servers:

```
ansible-playbook server.yml -i inventory
```

e. Browse to the hostname of the `server-0` node in your browser.

4.4. Update a portion of the configuration

a. Run the following command to only update the `nmap` portion of the `server` configuration:

```
ansible-playbook update-map.yml -i inventory
```

b. Feel free to change the value of `address_range` in `groups_vars/all.yml` and rerun `update-map.yml` to search for different nodes.

5. Execute Experiment

For a real experiment, you would now run your procedure.

Setup

Next: Finish

Systematic Experimentation (with Ansible)

7. Teardown Experiment

As always, please delete your resources when you are done with them.

Feel free to explore other topologies using these node types.

Introduction