Introduction to Ansible

Radenko Čitaković
Cisco Systems Engineer

19.03.2019.
Why learn Ansible?
Real-Time Remote Execution of Commands

1. Audit routes on all virtual machines
   ansible -m shell -a “netstat -rn” datacenter-east

2. Updates routes required for consistency
   ansible -m shell -a “route add X.X.X.X” datacenter-east
1. Deploy application change to stage and verify

2. Update load balancer pools to point to stage

Change Control Workflow Orchestration
How does Ansible work?
How does Ansible work?

1. Engineers deploy Ansible playbooks written in YAML to a control station.

2. Ansible copies modules typically written in Python to remote hosts to execute tasks.
Inside the Ansible Control Station

- Linux host with a Python and the Ansible installed
- Support transport to remote hosts
  - Typically SSH but could use an API
- Ansible Components
  - Ansible configuration file
  - Inventory files
  - Ansible modules
  - Playbooks
Ansible Configuration File

- Control operation of Ansible
- Default configuration
  /etc/ansible/ansible.cfg
- Override default settings
  - ANSIBLE_CONFIG ENV
  - ansible.cfg in current directory
  - ..ansible.cfg in home directory
- See Ansible documentation for all options

DevNet$ cat ansible.cfg

# config file for ansible
# override global certain global settings

[defaults]
# default to inventory file of ./hosts
inventory = ./hosts

# disable host checking to automatically add
# hosts to known_hosts
host_key_checking = False

# set the roles path to the local directory
roles_path = ./

http://docs.ansible.com/ansible/latest/intro_configuration.html
Ansible Authentication Basics

• Typically, Ansible uses SSH for authentication and assumes keys are in place

• **Setting up** and **transferring** SSH keys allows playbooks to be run automatically

• Using passwords is possible
  - Network Devices often use passwords

```
DevNet$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key:
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ~/.ssh/id_rsa.
Your public key has been saved in ~/.ssh/id_rsa.pub.

DevNet$ ssh-copy-id root@10.10.20.20
Number of key(s) added: 1

Now try logging into the machine, with:
"ssh 'root@10.10.20.20'"

DevNet$ ssh root@10.10.20.20
Last login: Fri Jul 28 13:33:46 2017 from 10.10.20.7
(python2) [root@localhost sbx_nxos]#
```

Output edited for brevity and clarity
Ansible Inventory File

- Inventory file identifies hosts, and groups of hosts under management
  - Hosts can be IP or FQDN
  - Groups enclosed in []

- Can include host specific parameters as well
  - Example: Instructing Ansible to use the active Python Interpreter when using Python Virtual Environments

```
DevNet$ cat hosts
[dcloud-servers:children]
datacenter-east
datacenter-west

[datacenter-east]
198.18.134.49 ansible_python_interpreter="/usr/bin/env python"

[datacenter-west]
198.18.134.50 ansible_python_interpreter="/usr/bin/env python"
```

*Output edited for brevity and clarity*
# Ansible CLI Tool Overview

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible</td>
<td>Executes modules against targeted hosts without creating playbooks.</td>
</tr>
<tr>
<td>ansible-playbook</td>
<td>Run playbooks against targeted hosts.</td>
</tr>
<tr>
<td>ansible-vault</td>
<td>Encrypt sensitive data into an encrypted YAML file.</td>
</tr>
<tr>
<td>ansible-pull</td>
<td>Reverses the normal “push” model and lets clients &quot;pull&quot; from a centralized server for execution.</td>
</tr>
<tr>
<td>ansible-docs</td>
<td>Parses the docstrings of Ansible modules to see example syntax and the parameters modules require.</td>
</tr>
<tr>
<td>ansible-galaxy</td>
<td>Creates or downloads roles from the Ansible community.</td>
</tr>
</tbody>
</table>
Using Ansible CLI for ad-hoc Commands

- Quickly run a command against a set of hosts
- **Specify the module** with `-m module`
- **Specify the username** to use with `-u user`
- Default is to use local username
- **Specify the server or group** to target
- Provide module arguments with `-a argument`

```
DevNet$ ansible -m setup -u root servers
10.10.20.20 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": ["10.10.20.20", "172.17.0.1"],
    "ansible_all_ipv6_addresses": ["fe80::250:56ff:febb:3a3f"],
    "ansible_apparmor": {"status": "disabled"},
    "ansible_architecture": "x86_64",
  }
}
```

*Output edited for brevity and clarity*
YAML Overview
YAML Overview

• What is YAML?
  • “YAML Ain’t Markup Language”
  • YAML is a human readable data serialization language
    • YAML files are easily parsed into software data structures
  • YAML is a common basis for a number of domain specific languages
    • Ansible
    • Heat
    • Saltstack
    • cloud-init
    • Many more!
YAML Overview

YAML sequences become Python lists

Multiple YAML documents separates by a ---

YAML mappings become Python dictionaries

YAML uses spacing to nest data structures
Ansible Playbooks
# Ansible Terms

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
</table>
| module | Code, typically written in Python, that will perform some action on a host.  
*Example: yum - Manages packages with the yum package manager*                           |
| task   | A single action that references a module to run along with any input arguments and actions                                               |
| play   | Matching a set of tasks to a host or group of hosts                                                                                       |
| playbook | A YAML file that includes one or more play                                                                                               |
| role   | A pre-built set of playbooks designed to perform some standard configuration in a repeatable fashion.  
*Example: A role to configure a web server would install Apache, configure the firewall, and copy application files.* |

**http://docs.ansible.com/ansible/latest/list_of_all_modules.html**  
**http://docs.ansible.com/ansible/latest/playbooks.html**
Ansible Playbooks

- Written in YAML
- One or more plays that contain hosts and tasks
- Tasks have a name & module keys.
- Modules have parameters
- Variables referenced with `{name}`
  - Ansible gathers “facts”
  - Create your own by register-ing output from another task

http://docs.ansible.com/ansible/latest/YAMLSyntax.html
Ansible Playbooks

DevNet$ ansible-playbook -u root example1.yaml

PLAY [Report Hostname and Operating System Details]
***********************************************************************************************
TASK [Gathering Facts]
*****************************************************************************************************
ok: [10.10.20.20]

TASK [Get hostname from server]
*****************************************************************************************************
ok: [10.10.20.20] => {
  "msg": "localhost"
}

PLAY [Report Network Details of Servers]
*****************************************************************************************************
TASK [Network routes installed]
*****************************************************************************************************
ok: [10.10.20.20] => {
  "stdout_lines": [
    "Kernel IP routing table",
    "Destination     Gateway         Genmask         Flags MSS Window irtt Iface",
    "0.0.0.0         10.10.20.254    0.0.0.0         UG        0 0          0 ens160",
    "10.10.20.0      0.0.0.0         255.255.255.0   U         0 0          0 ens160",
    "172.16.30.0     10.10.20.160    255.255.255.0   UG        0 0          0 ens160",
  ]
}

PLAY RECAP
********************************************************************************************************************
*******************
10.10.20.20                : ok=7    changed=1    unreachable=0    failed=0

Output edited for brevity and clarity
Using Variable Files and Loops with Ansible

- Include external variable files using vars_files: `filename.yaml`
- Reference variables with `{{name}}`
- YAML supports lists and hashes (ie key/value)
- Loop to repeat actions with with_items: `variable`
Using Variable Files and Loops with Ansible

DevNet$ ansible-playbook -u root example2.yaml

PLAY [Illustrate Variables] ***********************************************

TASK [Print Company Name from Variable] ***************
ok: [10.10.20.20] => {
  "msg": "Hello DevNet"
}

TASK [Loop over a List] ***********************************************
ok: [10.10.20.20] => (item=DevNet Rocks!) => {
  "item": "DevNet Rocks!",
  "msg": "DevNet Rocks!"
}
ok: [10.10.20.20] => (item=Programmability is amazing) => {
  "item": "Programmability is amazing",
  "msg": "Programmability is amazing"
}
ok: [10.10.20.20] => (item=Ansible is easy to use) => {
  "item": "Ansible is easy to use",
  "msg": "Ansible is easy to use"
}
ok: [10.10.20.20] => (item=Lists are fun!) => {
  "item": "Lists are fun!",
  "msg": "Lists are fun!"}
Jinja2 Templating – Variables to the Max!

- Not just for Ansible templates
- Powerful templating language
  - Loops, conditionals and more supported
- Leverage template module
  - Attributes
    - `src`: The template file
    - `dest`: Where to save generated template

http://docs.ansible.com/ansible/latest/playbooks_templating.html
Jinja2 Templating – Variables to the Max!

DevNet$ ansible-playbook -u root example3.yaml

PLAY [Generate Configuration from Template] ****************************************

TASK [Generate config] ************************************************************
changed: [localhost]

PLAY RECAP **************************************************
localhost : ok=1    changed=1    unreachable=0    failed=0

DevNet$ cat example3.conf
feature bgp
router bgp 65001
  router-id 10.10.10.1
Host and Group Variables

- Ansible allows for Group and Host specific variables
  - `group_vars/groupname.yaml`
  - `host_vars/host.yaml`
- Variables automatically available
Using Ansible Roles

roles
declares any
playbooks
defined within
a role must be
executed
against hosts

Roles promote
playbook reuse

Roles contain playbooks,
templates, and variables
to complete a workflow
(e.g. installing Apache)
Learning More About Ansible

- Ansible has an extensive module library capable of operating compute, storage and networking devices
  - [http://docs.ansible.com/ansible/modules_by_category.html](http://docs.ansible.com/ansible/modules_by_category.html)
- Ansible’s domain specific language is powerful
  - Loops
  - Conditionals
  - Many more!
  - [http://docs.ansible.com/ansible/playbooks.html](http://docs.ansible.com/ansible/playbooks.html)
- Ansible galaxy contains community supported roles for re-use
  - [https://galaxy.ansible.com/](https://galaxy.ansible.com/)
Wrap-Up
What you learned in this session...

- Ansible use cases
- Setting up Ansible infrastructure
- Using the Ansible ad-hoc CLI
- Creating and running Ansible playbooks