Introduction to Ansible

Radenko Čitaković
Cisco Systems Engineer

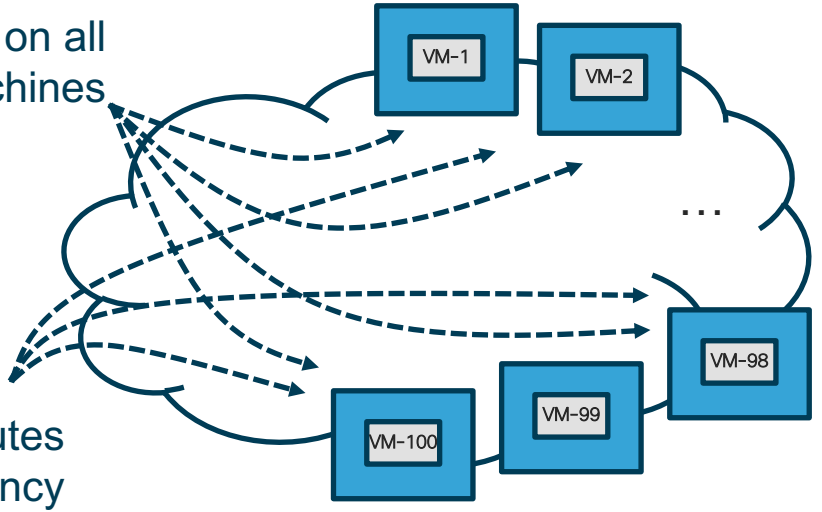
19.03.2019.

Why learn Ansible?

Real-Time Remote Execution of Commands

```
ansible -m shell -a "netstat -rn" datacenter-east
```

1. Audit routes on all virtual machines

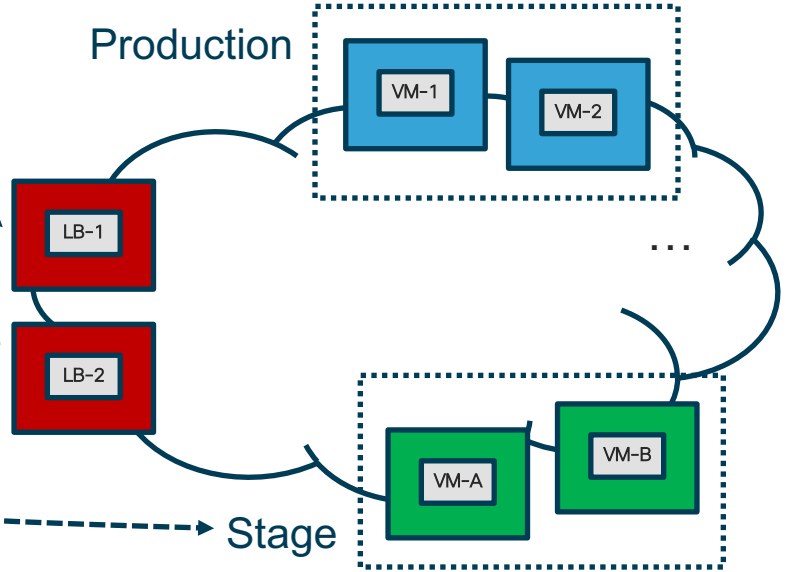


2. Updates routes required for consistency

```
ansible -m shell -a "route add X.X.X.X" datacenter-east
```

Change Control Workflow Orchestration

2. Update load balancer pools to point to stage

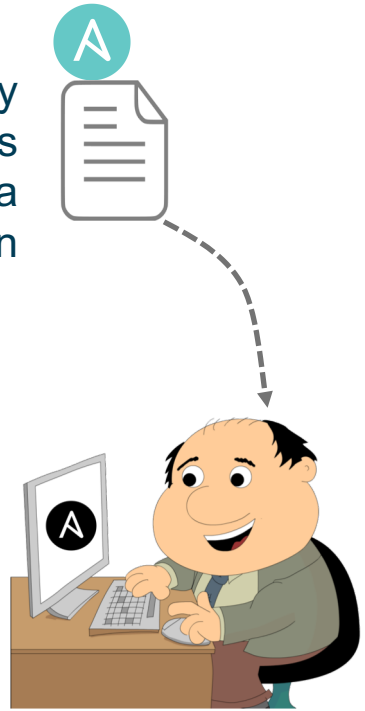


1. Deploy application change to stage and verify

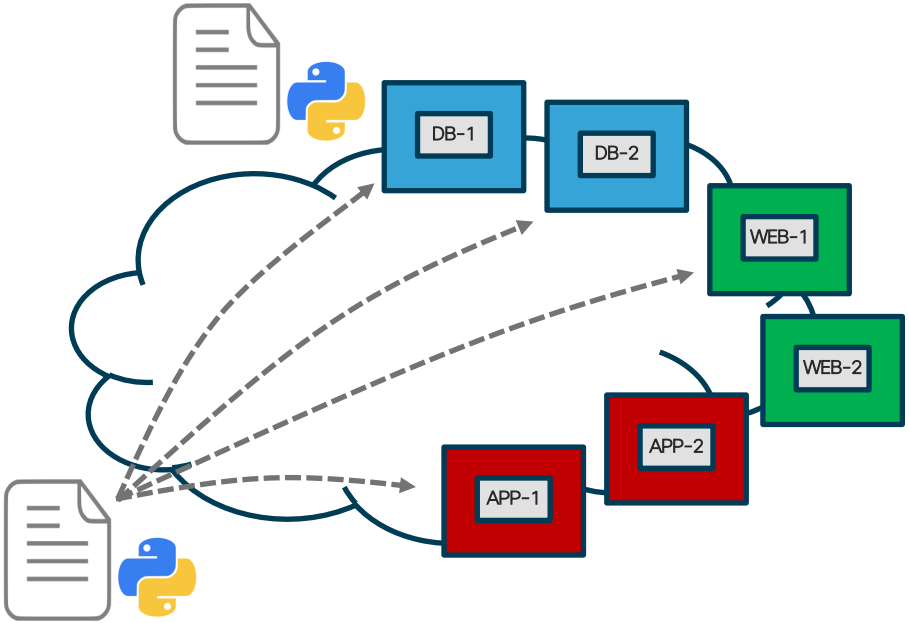
How does Ansible work?

How does Ansible work?

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



Ansible 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

Tool	Description
`ansible`	Executes modules against targeted hosts without creating playbooks.
`ansible-playbook`	Run playbooks against targeted hosts.
`ansible-vault`	Encrypt sensitive data into an encrypted YAML file.
`ansible-pull`	Reverses the normal “push” model and lets clients "pull" from a centralized server for execution.
`ansible-docs`	Parses the docstrings of Ansible modules to see example syntax and the parameters modules require.
`ansible-galaxy`	Creates or downloads roles from the Ansible community.

Using Ansible CLI for ad-hoc Commands

- Quickly run a command against a set of hosts
- Specify the module with `-m module`
- Specfiy 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

```
---
  - item1
  - item2
  - item3
  - item4
```

YAML sequences become Python lists

```
---
  - item1
  - item2
  - item3
  - item4
---
  - item5
  - item6
  - item7
  - item8
```

Multiple YAML documents separates by a ---

```
---
  key-1: value1
  key-2: value2
  key-3: value3
  key-4: value4
```

YAML mappings become Python dictionaries

```
---
  key-1:
    key-a: value-a
    key-b: value-b
    key-c: value-c
    key-d: value-d
  key-2:
    key-e: value-e
    key-f: value-f
    key-g: value-g
    key-h: value-h
```

YAML uses spacing to nest data structures

Ansible Playbooks

Ansible Terms

Tool	Description
`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. A play could leverage a role rather than tasks. *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

```
---
- name: Report Hostname and Operating System Details
  hosts: servers

  tasks:
    - name: "Get hostname from server"
      debug:
        msg: "{{ansible_hostname}}"

    - name: "Operating System"
      debug: msg="{{ansible_distribution}}"

- name: Report Network Details of Servers
  hosts: servers

  tasks:
    - name: "Default IPv4 Interface"
      debug: msg="{{ansible_default_ipv4.interface}}"

    - name: "Retrieve network routes"
      command: "netstat -rn"
      register: routes

    - name: "Network routes installed"
      debug: msg="{{routes}}"
```

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`

example2_vars.yaml

```
---
company_name: "DevNet"
quotes:
  - "DevNet Rocks!"
  - "Programmability is amazing"
  - "Ansible is easy to use"
  - "Lists are fun!"
```

```
---
- name: Illustrate Variables
  hosts: servers
  gather_facts: false
  vars_files:
    - example2_vars.yaml

  tasks:
    - name: "Print Company Name from Variable"
      debug: msg="Hello {{company_name}}"

    - name: "Loop over a List"
      with_items: "{{quotes}}"
      debug: msg="{{item}}"
```

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

example3.j2

```
feature {{feature}}
router bgp {{asn}}
  router-id {{router_id}}
```

```
---
- name: Generate Configuration from Template
  hosts: localhost
  gather_facts: false
  vars:
    feature: bgp
    asn: 65001
    router_id: 10.10.10.1

  tasks:
    - name: "Generate config"
      template:
        src: "example3.j2"
        dest: "./example3.conf"
```

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

```
├── group_vars
│   └── all.yaml
│   └── switches.yaml
├── host_vars
│   ├── 172.16.30.101.yaml
│   ├── 172.16.30.102.yaml
│   ├── 172.16.30.103.yaml
│   └── 172.16.30.104.yaml
```

Using Ansible Roles

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

```
- hosts: dcloud-servers
  roles:
    - { role: geerlingguy.apache }
```

Roles promote playbook reuse

```
$ cd geerlingguy.apache/
$ ls
LICENSE   README.md defaults  handlers  meta      tasks     templates tests     vars
```

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
- Ansible's domain specific language is powerful
  - Loops
  - Conditionals
  - Many more!
  - http://docs.ansible.com/ansible/playbooks.html
- Ansible galaxy contains community supported roles for re-use
  - 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


CISCO