6

Using Ansible to manage configuration and infrastructure on OCI

Agenda

In this module we will cover the details of running Ansible on OCI to manage both configuration and infrastructure.

- Discuss configuration management and some of the cloud scale challenges
- Describe the key components of Ansible
- Understand the configuration management capabilities of Ansible
- Review the OCI modules for Ansible
- Provision a simple set of OCI resources with Ansible

Configuration Management in Summary

According to the worldwide web, **configuration management** is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information through its life.

In short: Make it easy to go from "works on my box"

to "works on every box."

Configuration Management – Cloud Scale Challenges

Many cloud workloads are highly distributed. Applications are often complex consisting of different software packages, prerequisites, security configurations, and more. Common management challenges include (but are certainly not limited to):

- Configuration drift
- Inconsistent execution of manual changes
- Limited visibility into hardware and software configuration of existing resources.
- Time-consuming deployment of applications or configuration changes

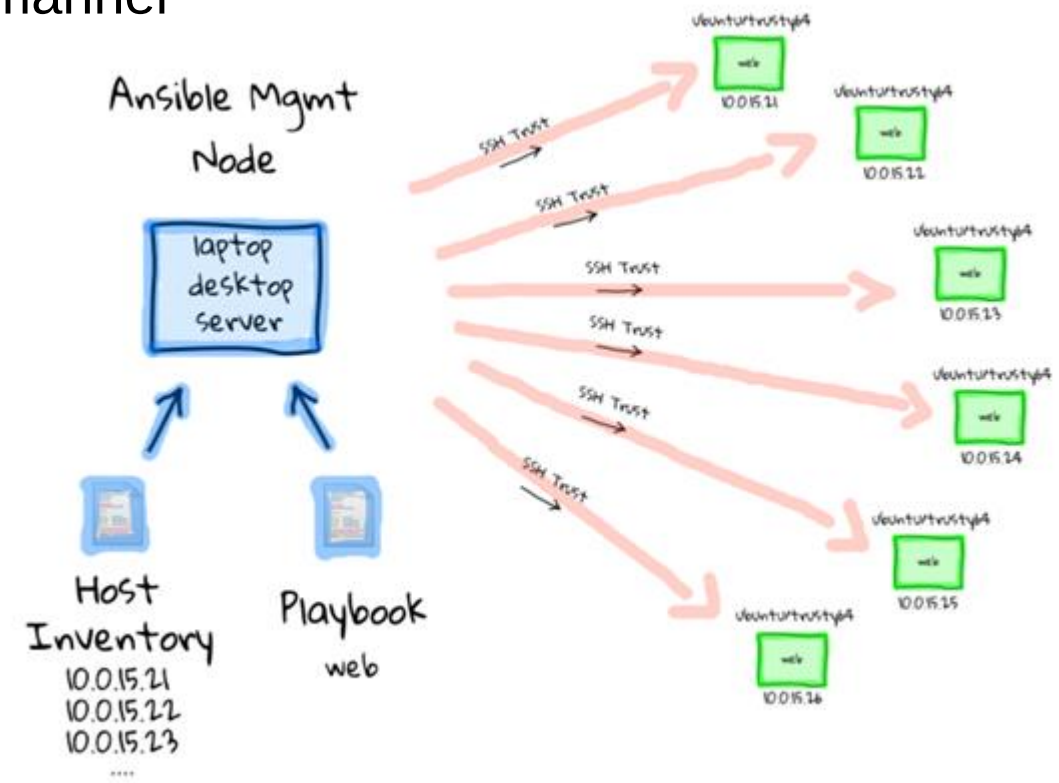
Tracking and managing resources in different parts of the world requires the right toolkit. We should be able to:

- Identify and track resources by name, type, function
- Define and apply configuration in a consistent manner
- Eliminate / overwrite manual changes

Configuration Management – Cloud Scale Solution

Tracking and managing resources in different parts of the world requires the right toolkit. One that should be able to:

- Identify and track resources by name, type, function
- Define and apply configuration in a consistent manner
- Eliminate / overwrite manual changes
- Discover and report hardware / software configurations that existing



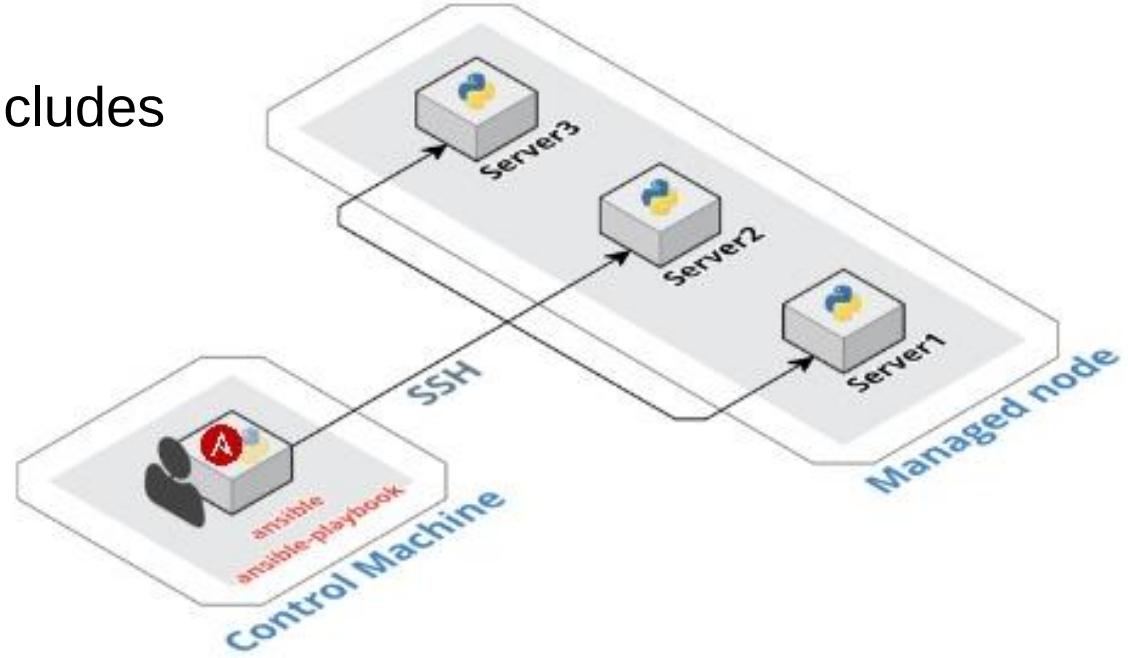
Ansible is…

- Simple
  - Human readable automation
  - No special coding skills needed
  - Tasks are executed in Order
- Powerful
  - App and infrastructure deployment
  - Configuration Management
- Agentless
  - Uses OpenSSH and WinRM
  - No agents to exploit or update

ANSIBLE

How Ansible Works

- Can be used to execute a variety of ad-hock commands initiated from a control point
- Utilizes small modules called “Playbooks” to perform command execution via remote SSH
- Utilizes SSH keypairs for authentication (Kerberos is supported)
- Inventory managed in a simple text file; also includes plugins to read from additional sources



Installing Ansible

Where you choose to run Ansible is referred to as the **Control Machine**.

Current requirements include:

- Linux distro (RHEL, Debian, Centos, OS X, etc)
- Python 2.6 / 2.7 or Python 3.5 or higher
- *Windows is not currently supported*

```
[opc@demo ~]$ cd ansible
[opc@demo ansible]$ sudo yum install -y ansible
```

```
[opc@demo ansible]$ ansible --version
ansible 2.5.2
```

***Note MacOS defaults to 15 file handles. You will need to raise the limit to avoid "Too many open files" errors*

Working with Inventory

Before we begin, you must create a **hosts** file to organize the servers that will be managed by Ansible.

By default, this file is located in /etc/ansible/hosts

```
# Ungrouped hosts, specify before any group headers.

# Jenkins Master
10.0.200.2

# Ex 2: A collection of hosts belonging to the 'webservers' group

[webservers]
10.0.100.7
10.0.110.3
```

Ad-hoc commands

With Ansible, it is easy to execute remote commands against one or more of your hosts as defined in the inventory file. Keep in mind, however:

- The appropriate SSH keys must be available on the Control Machine (consider using ssh-agent)
- Host Key Checking is enabled by default. You must have an entry in the known_hosts file for each server, or disable host key checking
  - Edit /etc/ansible/ansible.cfg
    [defaults]
    host_key_checking = false
  - Set an environment variable for the duration of the session
    export ANSIBLE_HOST_KEY_CHECKING=false

Ad-hoc commands - continued

- To perform an ad-hoc command on a single host, use the hostname or IP address:

```
[opc@demo ~]$ ansible 10.0.200.2 -m ping
10.0.200.2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- To perform an ad-hoc command on a group of hosts defined in your inventory file

- To perform a command on all hosts in your inventory file:
  `$ ansible all -m ping`

```
[opc@demo ~]$ ansible webservers -m ping
10.0.110.3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.100.7 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Ad-hoc commands - Comtinued

Ansible also supports a variety of other ad-hoc commands. Here are a few samples:

- Shell commands:
  $ ansible webservers -m shell -a "touch /home/opc/ansible.test“
- Package management:
  $ ansible webservers -m yum -a “name=httpd state=latest”
- Manage services:
  $ ansible webservers -m service -a “name=httpd state=restarted”

Getting started – Common errors

- Incorrect / missing private SSH key

```
10.0.110.3 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (
pi-with-mic).\r\n",
    "unreachable": true
}
```

- Host Key Checking enabled and no entry in known_hosts file

```
[opc@demo ansible]$ ansible 10.0.200.2 -m ping
The authenticity of host '10.0.200.2 (10.0.200.2)' can't be established.
ECDSA key fingerprint is SHA256:3K0SM5BzQVYuqXSYO/nUHG7vWOajk6KQgjz260qKgCs.
ECDSA key fingerprint is MD5:2b:03:b5:20:7f:56:b8:01:5c:1d:6f:99:0f:cc:3e:2f.
Are you sure you want to continue connecting (yes/no)? yes
```

Ansible Playbooks

- Configuration, deployment, and orchestration “manuals”
- Written in YAML
- Designed to be human readable
- Can declare configuration and orchestrate a series of serial tasks

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
```

Ansible playbooks - continued

In addition to installing packages, you can create files, run tasks, etc.

Ansible uses Jinja2 templating to enable dynamic expressions and access to variables. This is the preferred method over writing static files via command execution.

```
tasks:
  - name: make sure apache is running
    service:
      name: httpd
      state: started
  - name: create simple file
    template:
      src: 404error.j2
      dest: /var/www/html/404error.html
```

OCI Ansible Modules

Oracle Cloud Infrastructure Ansible Modules provide an easy way to create and provision resources in Oracle Cloud Infrastructure (OCI) through Ansible. These modules allow you to author Ansible playbooks that help you automate the provisioning and configuring of Oracle Cloud Infrastructure services and resources, such as Compute, Load Balancing, Database, and other Oracle Cloud Infrastructure services.

Services supported

- Block Volume
- Compute
- IAM
- Load Balancing
- Networking
- Object Storage
- Database

Installing the OCI Ansible Modules

The OCI Ansible modules are available for download from a Github repository.

```
$ git clone https://github.com/oracle/oci-ansible-modules.git
$ cd oci-ansible-modules
$ sudo ./install.py
```

Ansible requires a valid IAM user with API signing key. Default behavior is to look for the OCI CLI Config file in **~/.oci/config**

You may also define environment variables or a different path to the credentials that will be used by Ansible.

Ansible on OCI – Run a simple test

Once the OCI Modules for Ansible are installed and your user credentials are configured, it is time to run a quick test.

After entering your Tenancy ID and compartment ID, run:

```
$ ansible-playbook oci_sample.yml
```

```
---
- name : List summary of existing buckets in OCI object storage
  connection: local
  hosts: localhost
  tasks:
    - name: List bucket facts
      oci_bucket_facts:
          namespace_name: 'mycompartment'
          compartment_id: 'ocid1.compartment.oc1..aaaaaaaaav4xfnc6k7z25pu3q'
      register: result
    - name: Dump result
      debug:
        msg: '{{result}}'
```

OCI Sample – Command Output

By including a task called **Dump result** we will see a detailed output, including information for all of the Object Storage buckets in the designated compartment.

```
- name: Dump result
  debug:
    msg: '{{result}}'
```

```
[opc@demo ansible]$ ansible-playbook oci_sample.yml

PLAY [List summary of existing buckets in OCI object storage] *****

TASK [Gathering Facts] ********************************************
ok: [localhost]

TASK [List bucket facts] ******************************************
ok: [localhost]

TASK [Dump result] ************************************************
ok: [localhost] => {
    "msg": {
        "buckets": [
            {
                "compartment_id": "ocid1.compartment.oc1..aaaaaaaav
qlq3p5xmictf2xfnc6k7z25pu3q",
                "created_by": "ocid1.user.oc1..aaaaaaaais75kkcibznt
3ualqwxy6ofzd7dq",
                "defined_tags": null,
                "etag": "1c0890df-66a1-477b-835b-65605c22494b",
```

OCI Ansible Modules Documentation

In addition to the online documentation, the **ansible-doc** command can be used to view a detailed help file for each module.

The list of modules is too extensive to list here, though it can be viewed here:

https://oracle-cloud-infrastructure-ansible-modules.readthedocs.io/en/latest/modules/list_of_cloud_modules.html

Sample cloud modules

- **oci_compartment_facts**
- **oci_database_facts**
- **oci_image_facts**

```
> OCI_IMAGE_FACTS    (/usr/lib/python2.7/site-packages

        This module retrieves details about a specific
        images in a specified Compartment in OCI Compu

OPTIONS (= is mandatory):

- api_user
        The OCID of the user, on whose behalf, OCI API
```

Give it a try!

The OCI Ansible Modules Github repository contains a number of sample playbooks. Before you can execute the sample playbooks, you will need to collect some account information and assign the appropriate variables.

- Create a directory (i.e. ansible-oci) and in that directory a file called **env-vars**
- insert the following three lines into the env-vars file:
  - SAMPLE_COMPARTMENT_OCID=<your compartment OCID>
  - SAMPLE_IMAGE_OCID=<desired image OCID>
  - SAMPLE_AD_NAME=<desired AD name>

***note: the AD name will be unique to your tenancy. i.e. nHXp:PHX-AD-2*

Give it a try! - continued

Now source the env-vars contents to your environment variables, clone the Github repo, and execute the playbook.

$ source env-vars

$ git clone https://github.com/oracle/oci-ansible-modules.git

$ cd oci-ansible-modules/samples/compute/nat-instance-configuration

$ ansible-playbook sample.yaml

The last line of the sample.yaml playbook is **- import_tasks: teardown.yaml**; to prevent Ansible from automatically destroying the environment, edit sample.yaml and comment out this line.

Give it a try! - Examples

1

```
@demo demo_compute]$ ansible-playbook sample.yaml

PLAY [Launch a compute instance and connect to it using SSH]

TASK [Gathering Facts] ************************************
ok: [localhost]

TASK [Check pre-requisites] *******************************
skipping: [localhost] => (item=SAMPLE_COMPARTMENT_OCID)
skipping: [localhost] => (item=SAMPLE_IMAGE_OCID)
skipping: [localhost] => (item=SAMPLE_AD_NAME)
```

2

```
SK [Launch an instance in the public subnet to act
changed: [localhost]

TASK [Print instance details] ***********************
ok: [localhost] => {
    "msg": "Launched a new NAT instance in the publi
ycle_state': u'RUNNING', u'availability_domain': u'r
t_instance_for_mytest_nat_vcn', u'compartment_id': u
s757ijx7nwyun773mqqlq3p5xmictf2xfnc6k7z25pu3q', u'de
```

3

```
SK [Print sample SSH invocation to test if
stance] ***
ok: [localhost] => {
    "msg": "ssh -o \"UserKnownHostsFile=/dev/
/ansible.orfPjCcert/private_key.pem -o ProxyC
\" -o \"StrictHostKeyChecking=no\" -i /tmp/an
pc@$129.146.164.86\" opc@$10.0.1.2 ping -c2 o
```

4

```
PLAY RECAP *********************************************************
localhost                  : ok=54   changed=17   unreachable=0    failed=0
```

Knowledge Check

Which three statements about using Ansible on OCI are true?
[Choose 3]

A. You can use Ansible to create and destroy OCI resources, such as compute instances and load balancers.
B. You can use Ansible to execute a shell command on a collection of hosts.
C. You can use Ansible to collect billing and usage data for your OCI tenancy.
D. You can use Ansible to restart Apache on all web servers as defined in inventory.
E. The task sequence in your Ansible Playbook does not matter. Ansible will evaluate dependencies and execute tasks in the most effective sequence.

Knowledge Check

Which three statements about using Ansible on OCI are true?
[Choose 3]

A. You can use Ansible to create and destroy OCI resources, such as compute instances and load balancers.
B. You can use Ansible to execute a shell command on a collection of hosts.
C. You can use Ansible to collect billing and usage data for your OCI tenancy.
D. You can use Ansible to restart Apache on all web servers as defined in inventory.
E. The task sequence in your Ansible Playbook does not matter. Ansible will evaluate dependencies and execute tasks in the most effective sequence.

Summary

In this module we reviewed:

- The basic concepts of configuration management
- How to solve those challenges with Ansible
- How do use Ansible to deploy and manage OCI resources

Ansible and OCI Modules Documentation

- Getting Started with Ansible
  https://docs.ansible.com/ansible/devel/user_guide/intro_getting_started.html
- Getting Started with Ansible for OCI
  https://docs.cloud.oracle.com/iaas/Content/API/SDKDocs/ansiblegetstarted.htm
- Cloud Modules for OCI
  https://oracle-cloud-infrastructure-ansible-modules.readthedocs.io/en/latest/modules/list_of_cloud_modules.html
- OCI Ansible Modules Github Repo
  https://github.com/oracle/oci-ansible-modules

ORACLE®
Cloud Infrastructure

cloud.oracle.com/iaas

cloud.oracle.com/tryit