Combining Ansible and Terraform for CI

better together love story based on OVN-CI project

Arie Bregman
abregman@redhat.com

Szymon Datko
sdatko@redhat.com

Red Hat

20th October 2020

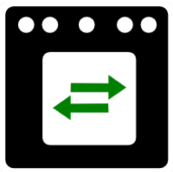
About us

Arie Bregman

- DevOps Engineer
- Gamer with all heart
- Open Source aficionado

Szymon Datko

- DevOps & local Bash wizard
- Open Source software lover
- Computer Graphics enthusiast



Part I

The tools

What is Ansible?

- Ansible is an IT automation tool.
- You can run various tasks with Ansible:
  - configure systems,
  - deploy software,
  - anything you run in your scripts! :-)
- Open Source and actively developed.
- Written in Python, easy to start.
- Widely used by companies and developers!

Red Hat

How Ansible works?

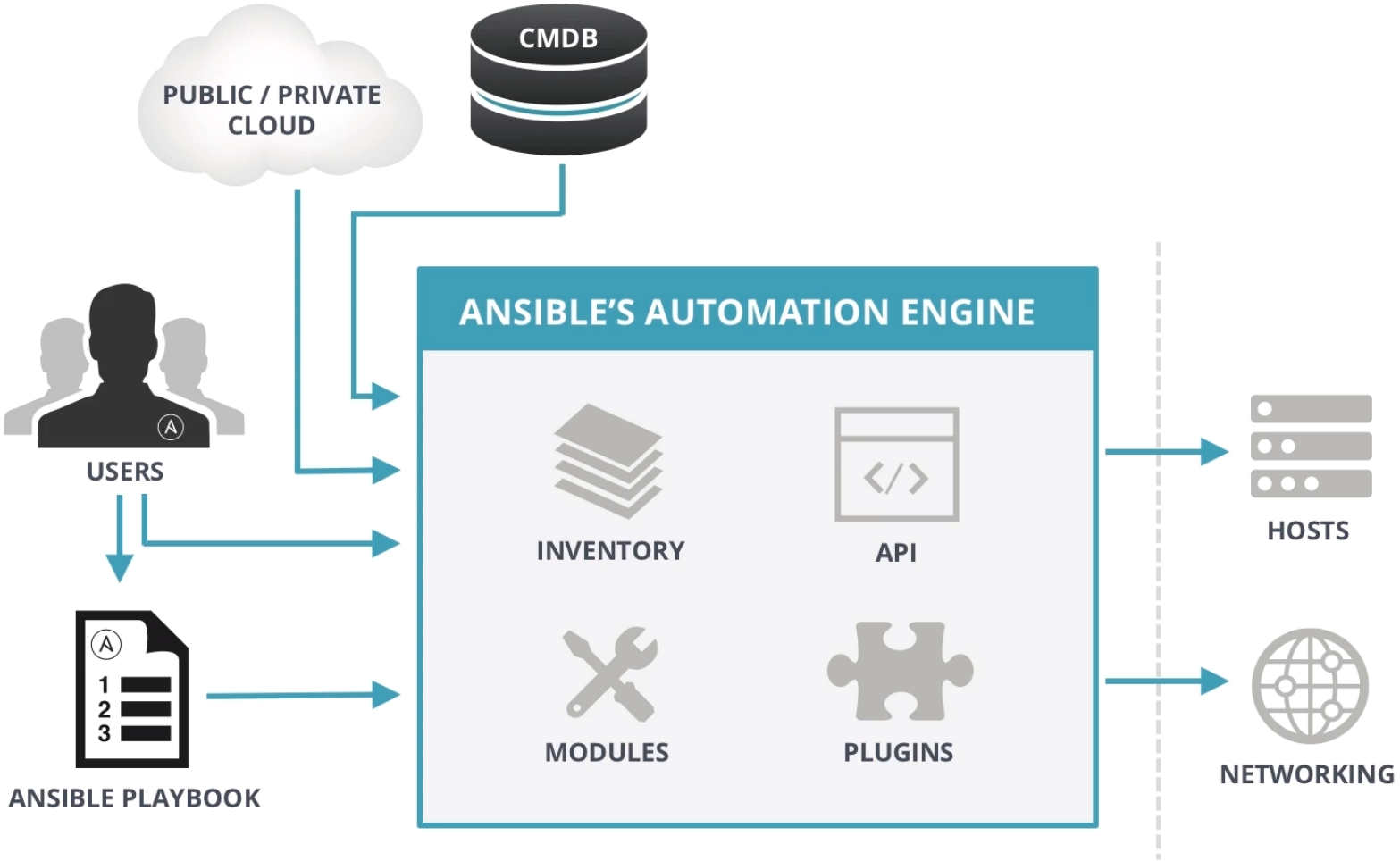


Image source: https://www.ansible.com/resources/videos/quick-start-video.

Red Hat

Example Ansible playbook

- Ansible playbooks are expressed in YAML format.
- As a user you specify what to run, where and under what conditions.
- Multiple playbooks can be grouped into a role.

```
1  # playbook.yml
2  ---
3  - name: Install DB servers
4    hosts: databases
5    remote_user: root
6
7    tasks:
8    - name: Ensure postgresql is at the latest version
9      package:
10       name: postgresql
11       state: latest
12   - name: Ensure that postgresql service is started
13     service:
14       name: postgresql
15       state: started
```

Red Hat

Running Ansible

- Run playbooks with: `ansible-playbook <playbook_file_path>`.
- Ad-hoc mode is also supported:
  1. `ansible -m package -a "name:postgresql state:present"`,
  2. `ansible -m service -a "name:postgresql state:started"`.
- Offline documentation for modules: `ansible-doc <module_name>`.

```
[user@db-server1 directory]$ ansible-playbook playbook.yml

PLAY [Install DB servers] ******************************************

TASK [Ensure postgresql is at the latest version] ******************
ok: [db-server1]

TASK [Ensure that postgresql service is started] *******************
ok: [db-server1]

PLAY RECAP *********************************************************
db-server1                 : ok=3    changed=0    unreachable=0    failed=0    skipped=0
```

Red Hat

What is Terraform?

- Use Infrastructure as Code to provision and manage any cloud, infrastructure or service.
- Terraform supports multiple providers:
  - clouds: OpenStack, GCP, AWS, Azure, ...;
  - containers: Docker, Kubernetes, ...
- Capable of resolving dependencies in resources.
- Aims to unify a way of resources management.
- Open Source, written in Go language.

Red Hat

How Terraform works?

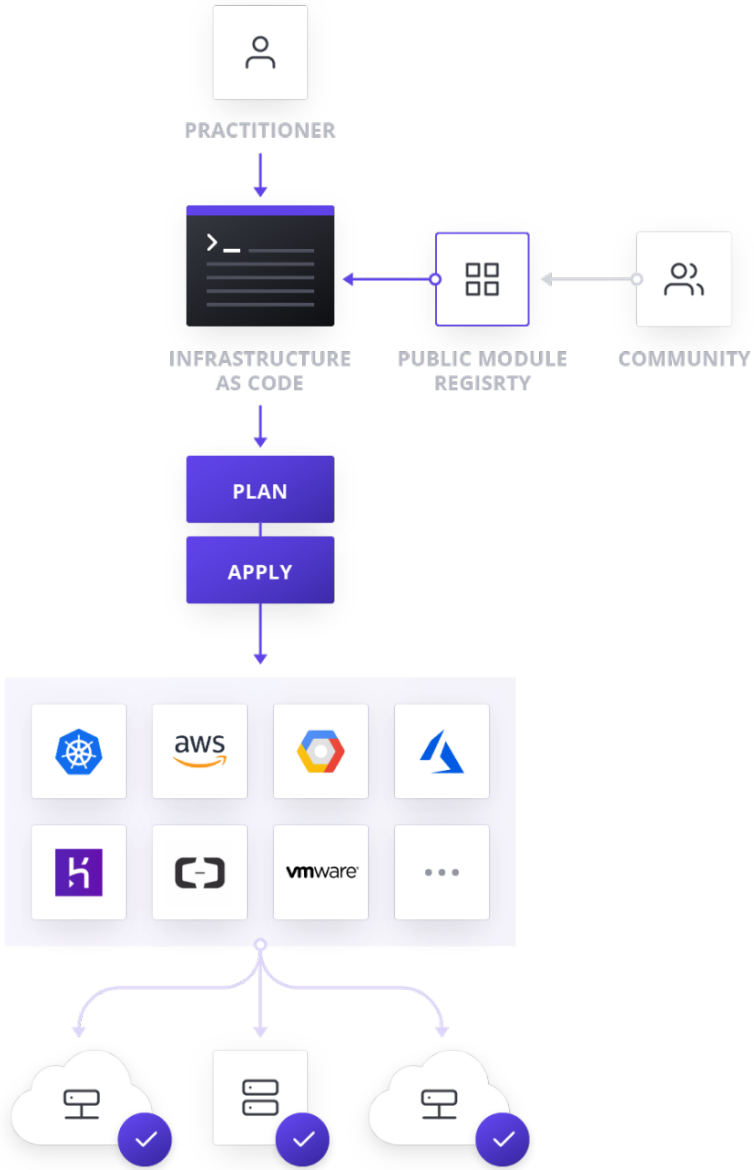


Image source: https://www.terraform.io.

Red Hat

Example Terraform manifest

- Terraform has its own declartive, configuration language,
  - HashiCorp Configuration Language (HCL).
- You specify which provider to use and the resources to manage.
- Configuration can be splitted to multiple files in the same directory.

```
provider "openstack" {
  auth_url    = "http://myauthurl:5000/v2.0"
  region      = "RegionOne"
  tenant_name = "admin"
  user_name   = "admin"
  # password  = ""  # If not given here, it will be read from env variable.
}

resource "openstack_compute_instance_v2" "test-server" {
  name = "my test server"
  image_name = "centos8"
  flavor_name = "m1.medium"
  key_pair = "default-access-key"
  network {
    name = "ext-net"
  }
}
```

Red Hat

Running Terraform

- Run the following commands in your shell:
  - `terraform init`
  - `terraform apply`
  - `terraform destroy`
- The **.terraform/** directory and **terraform.tfstate** file will be created.

```
[user@localhost directory]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of terraform-providers/openstack
- Installing terraform-providers/openstack v1.32.0...
- Installed terraform-providers/openstack v1.32.0

Terraform has been successfully initialized!
   (...)
```

```
[user@localhost directory]$ terraform apply
   (...)
Terraform will perform the following actions:

  # openstack_compute_instance_v2.test-server will be created
  + resource "openstack_compute_instance_v2" "test-server" {
      + access_ip_v4        = (known after apply)
      + access_ip_v6        = (known after apply)
      + all_metadata        = (known after apply)
      + all_tags            = (known after apply)
      + availability_zone   = (known after apply)
      + flavor_id           = (known after apply)
      + flavor_name         = "m1.medium"
      + force_delete        = false
      + id                  = (known after apply)
      + image_id            = (known after apply)
      + image_name          = "centos8"
      + key_pair            = "default-access-key"
      + name                = "my test server"
   (...)
```

HashiCorp Terraform

Red Hat

Part II

The combining

Why to combine Ansible and Terraform?

- "One tool to do everything" is not a goal by itself.
  - Though, for simple tasks you may not need to involve new toolkit.
- Recognizing the benefits of each tool will repay in complex setups.
  - Ansible is great for configuration management.
  - Terraform is great for provisioning resources.
- In the CI-related tasks: how do you track resources to clean up?

Image source: https://knowyourmeme.com/photos/882534-swiss-army-knives.

Red Hat

How to combine Ansible and Terraform?

1. In your shell: call Terraform, then call Ansible ;-)
   - Straightforward, yet elegant and comply with Unix philosophy.
   - Terraform can produce the Ansible inventory file for convenience.

```
$ terraform init
$ terraform apply -auto-approve
$ ansible-playbook --inventory terraform-hosts.ini
$ terraform destroy -auto-approve
```

2. Call Ansible and utilize Ansible module that runs Terraform.
   - The facts dictionary from Terraform output may be registered.

3. Call Terraform and use its provisioners mechanism to invoke Ansible.
   - *"Provisioners should only be used as a last resort. For most common situations there are better alternatives."* – Terraform documentation.
   - Details: `https://www.terraform.io/docs/provisioners/`.

Red Hat

Generating Ansible inventory with Terraform – configuration

- The **local_file** resource can be used to generate the inventory file.
- In the example below we create two servers with different names.

```
provider "openstack" {}  // Cloud configuration will be read from env variables.

resource "openstack_compute_instance_v2" "servers" {
  count = 2

  name = "my test server ${count.index + 1}"
  image_name = "centos8"
  flavor_name = "m1.medium"
  key_pair = "default-access-key"
  network {
    name = "ext-net"
  }
}

resource "local_file" "inventory" {
  content = templatefile("terraform-hosts.tpl", {
    servers = openstack_compute_instance_v2.servers[*]
  })
  filename = "terraform-hosts.ini"
  file_permission = "0644"
}
```

Red Hat

Generating Ansible inventory with Terraform – result

- The used template file has a syntax similar to popular **Jinja2**.
- Note that the outcome file is regular Terraform resource, so it will be:
  - created after `terraform apply`,
  - removed after `terraform destroy`.

Template file:

```
localhost  ansible_connection=local ansible_python_interpreter=auto_silent

[nodes]
%{ for index, server in servers ~}
node${index + 1} ansible_host=${server.access_ip_v4}
%{ endfor ~}
```

Outcome inventory file:

```
localhost  ansible_connection=local ansible_python_interpreter=auto_silent

[nodes]
node1 ansible_host=192.168.1.2
node2 ansible_host=192.168.1.3
```

Red Hat

Using Ansible module for running Terraform

Add to the Terraform configuration:

```
output "addresses" {
  value = openstack_compute_instance_v2.servers[*].access_ip_v4
}
```

Example Ansible playbook:

```
- name: Terraform example
  hosts: localhost
  tasks:
    - name: Create resources
      terraform:
        project_path: "{{ playbook_dir }}"
        force_init: true
        state: present
      register: terraform_run
    - debug:
        msg: "{{ terraform_run.outputs.addresses.value }}"
        # ok: [localhost] => {
        #     "msg": [
        #         "192.168.1.2",
        #         "192.168.1.3"
        #     ]
        #}
    - name: Cleanup resources
      terraform:
        project_path: "{{ playbook_dir }}"
        state: absent
```

Red Hat

Part III

Tips and pitfalls to avoid

Wait for connection and facts gathering

- Facts gathering fetches the useful information about the target host,
  - e.g. operating system details, file system layout, networking setup, etc.
- When used in the play level, it is performed before the tasks.
  - It will immediately try to connect and fail if host is not up and ready.

This is bad

```
1| - name: Incorrect play
2|   gather_facts: true
3|   hosts: my_nodes
4|   tasks:
5|     - name: Ensure system is reachable
6|       wait_for_connection:
7|         delay: 5
8|         sleep: 5
9|         timeout: 300
10|
11|     - name: Install Java
12|       include_role:
13|         name: geerlingguy.java
```

This is good

```
1| - name: Correct play
2|   gather_facts: false
3|   hosts: my_nodes
4|   tasks:
5|     - name: Ensure system is reachable
6|       wait_for_connection:
7|         delay: 5
8|         sleep: 5
9|         timeout: 300
10|
11|     - name: Gather facts
12|       setup:
13|
14|     - name: Install Java
15|       include_role:
16|         name: geerlingguy.java
```

Red Hat

Real-time output streaming from Ansible

- It may be useful for long-running tasks, like tests execution.
- No built-in solution so far, though there are proposals in community.
  - Details: `https://github.com/ansible/proposals/issues/92`.

Image source: `https://br.pinterest.com/pin/331366485055644550/`.

Real-time output streaming from Ansible – solution

Ideas:

- Patch Ansible, utilizing the callback plugin mechanism or so.
  - Example concept: `https://stackoverflow.com/a/54448411`.
- Do some Bash magic and SSH tunneling for this.
  - Will be really useful for **shell** tasks only.
  - The idea is to start a local process that will listen and display output.
  - Use the `--ssh-extra-args` Ansible argument to open the tunnel.

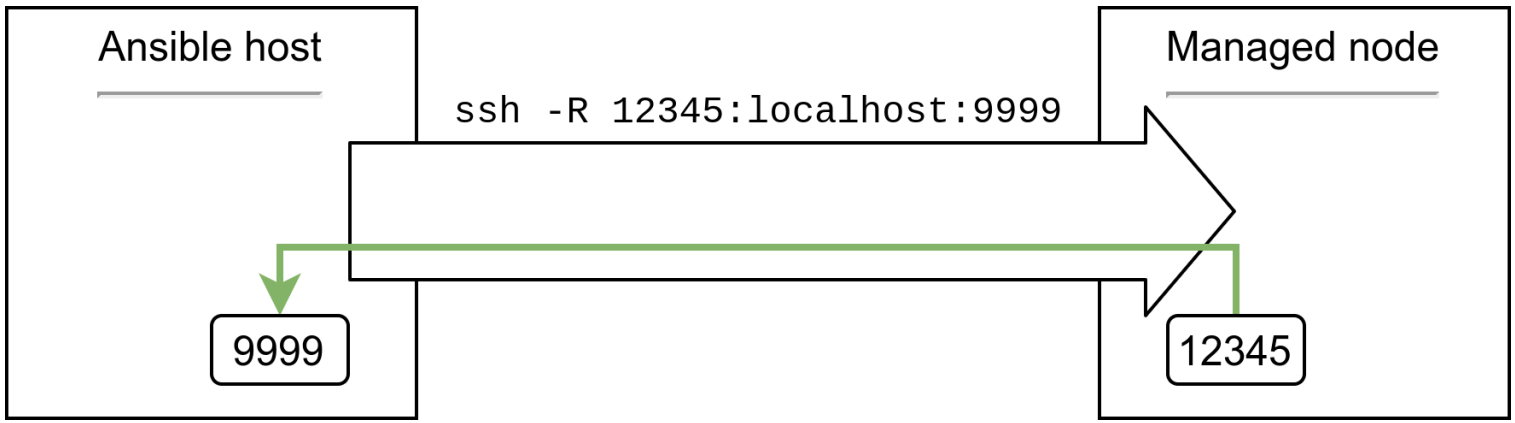


Diagram made with `https://draw.io/`.

Red Hat

Real-time output streaming from Ansible – implementation

In Bash script:

```
1  SOCAT_PID=-1
2  SOCAT_PORT=-1
3
4  # selection of value for SOCAT_PORT goes here -- see next slide
5
6  socat -t 3600 - "TCP-LISTEN:${SOCAT_PORT},fork,reuseaddr" &
7  SOCAT_PID=$!
8
9  ansible-playbook --inventory hosts.ini \
10     --ssh-extra-args="-R 12345:localhost:${SOCAT_PORT}" \
11     "${BASE_DIR}/launch-unit-tests.yml"
```

In Ansible playbook:

```
1  - name: Example play
2    hosts: node1
3    tasks:
4      ...
5      - name: Long-running task
6        shell:
7          chdir: ovn
8          cmd: |
9            exec > /dev/tcp/localhost/12345 2>&1
10           make
11     ...
```

Red Hat

Real-time output streaming from Ansible – helpful addition

As a bonus, two helpful snippets for process cleanup and port selection.

In Bash script:

```
#
# Cleanup function for the script, trap ensures to execute it
#
function cleanup() {
    if [ "${SOCAT_PID}" -gt 0 ]; then
        kill "${SOCAT_PID}"
    fi
}
trap cleanup EXIT

#
# Find random unused port
#
limit=20
for trial in $(seq 1 "${limit}"); do
    SOCAT_PORT=$(( ( ${RANDOM} % 57535 ) + 8000 ))
    if ! ( echo '' > "/dev/tcp/127.0.0.1/${SOCAT_PORT}" >/dev/null 2>&1 ); then
        break
    fi
    if [ "${trial}" -ge "${limit}" ]; then
        exit 1
    fi
done
```

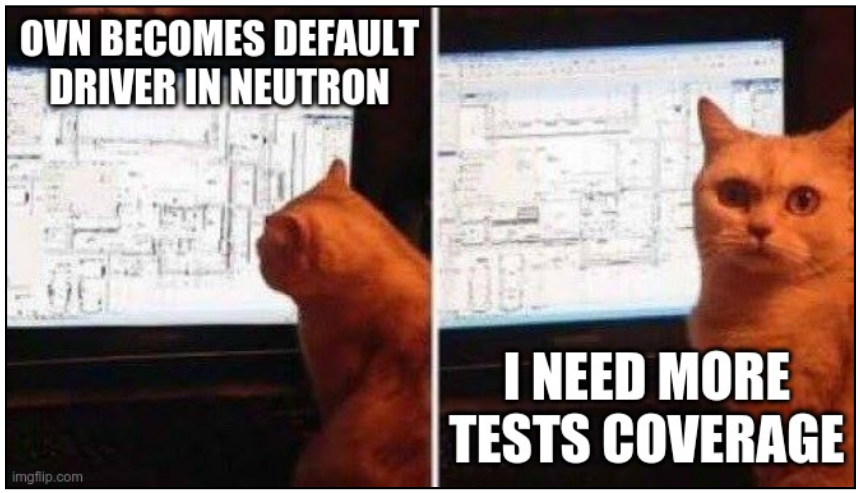
Red Hat

Part IV

Final words

Summary – how we use Ansible and Terraform in OVN CI

- Project started to make OVN rock solid and comprehensively tested.
  - Recently OVN was announced to be default network driver for Neutron.
- Why Ansible and Terraform for CI?
  - Provides convenient managing and configuring environments.
  - Out-of-box allows to track usage and status of resources.
  - Less dependency on CI/automation system = easy to reuse by hand.



Red Hat

Thank you for your attention!

The slides are available: `http://datko.pl/ansible-terraform.pdf`

Red Hat

Combining Ansible and Terraform for CI

better together love story based on OVN-CI project

Arie Bregman
abregman@redhat.com

Szymon Datko
sdatko@redhat.com

Red Hat

20th October 2020