Session ID #EGA77B878S
Foundational: Ansible / Terraform

Noel Colon
Garage Solution Engineering

John Webb
Garage Solution Engineering

Legal Disclaimer

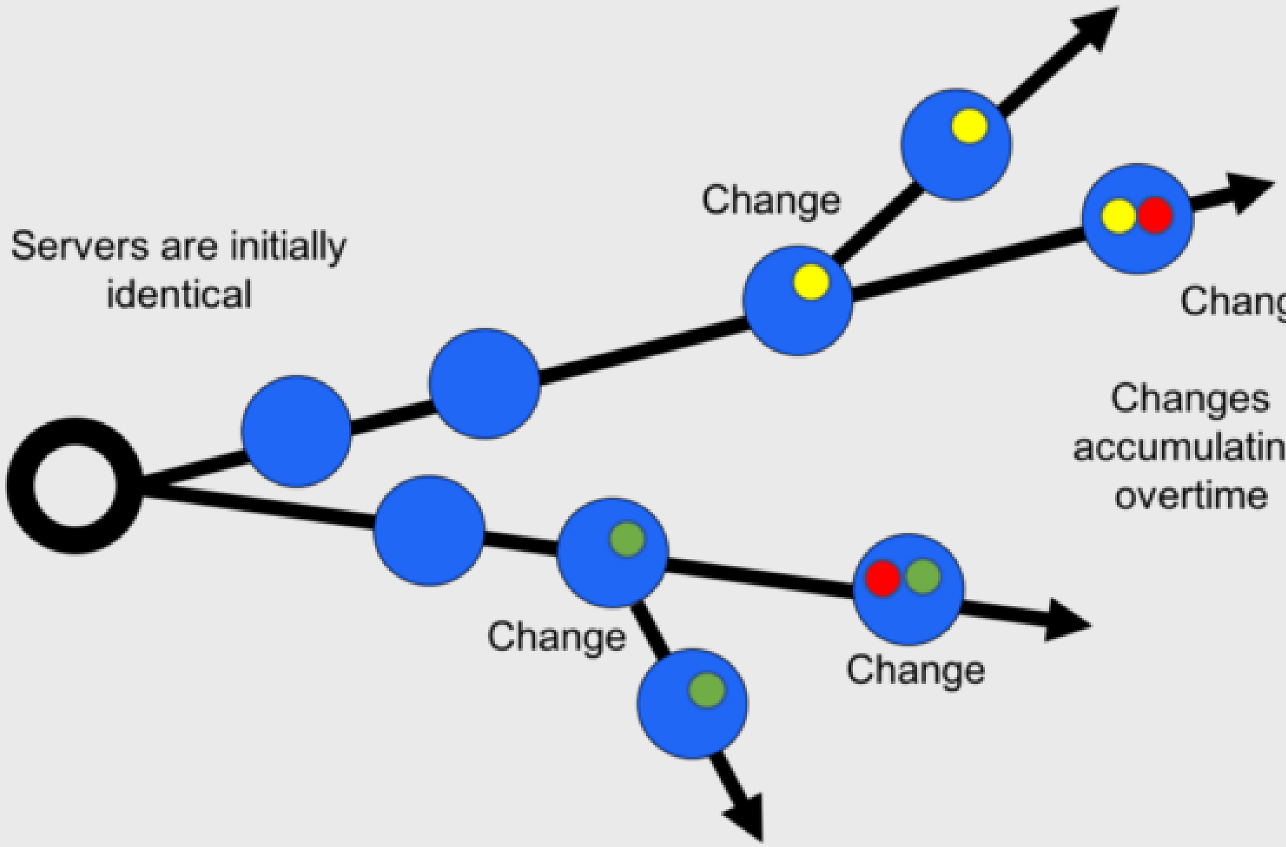
The information contained in this publication is provided for informational purposes only. While efforts were made to verifythe completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

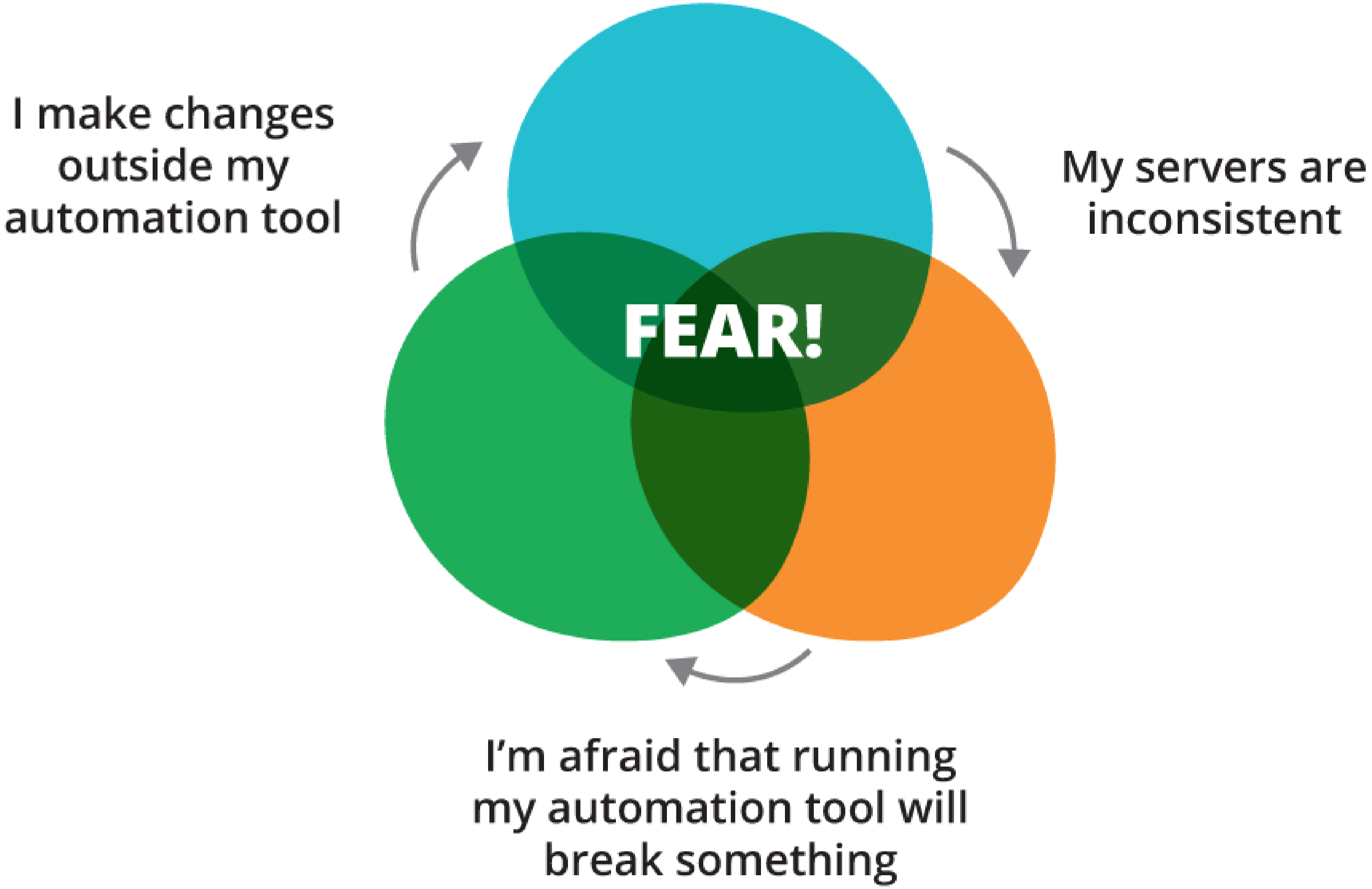
Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

The value of IaC can be broken down into three measurable categories: cost (reduction), speed (faster execution) and risk (remove errors and security violations).[

Configuration Drift


Servers are initially identical
Change
Change
Changes accumulating overtime
Change
Change

The Automation Fear Spiral


I make changes outside my automation tool
My servers are inconsistent
FEAR!
I'm afraid that running my automation tool will break something

What’s infrastructure ?

Traditionally the Cloud services, hosts, virtual machines, Docker containers, networking *(routing, switching, firewalls),* and storage were considered infrastructure. Now infrastructure also includes more complex services or Software-as-a-Service products delivered by third parties such as DNS, Content Delivery Networks *(CDN),* databases, job scheduling, queues, K8s, monitoring…. Terraform can configure components like GitHub organizations and repositories, Grafana monitoring console,…..

What’s Configuration Management ?

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 throughout its life.

CPU | Memory | Disk Drive | Network | Server | DB | …

Make the right choice

HashiCorp

Terraform

Write, Plan, and Create Infrastructure as Code

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently

Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

- *https://www.terraform.io/*
- *https://github.com/hashicorp/terraform*

RED HAT® ANSIBLE® Automation

Ansible is an open-source software provisioning, configuration management, and application-deployment tool

It uses a declarative configuration language: you declare your desired state, and Ansible will manage how to get your systems to that state

➢ *https://www.ansible.com/*

Terraform Primer

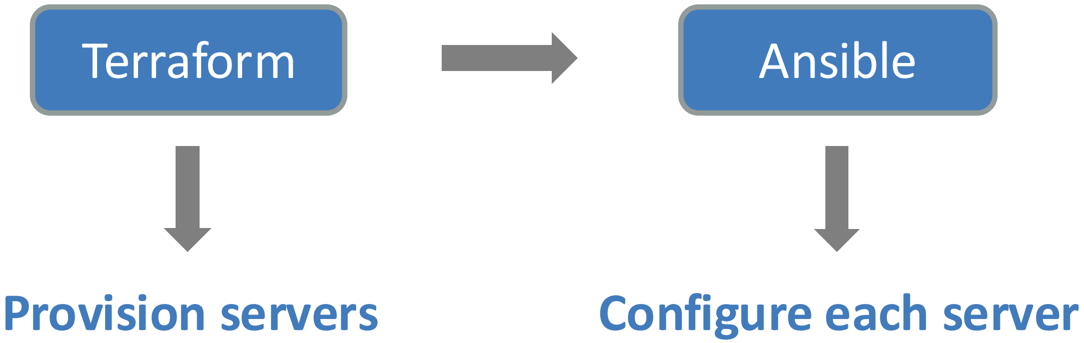
Providers, Data Sources and Resources

Declarative definitions style to define resources to provision

The Terraform and Ansible configuration language is declarative, describing an intended goal rather than the steps to reach that goal.

- ✓ With declarative definitions you specify what should be there and not how to do
- ✓ Only missing parts must be created, existing ones must be in the desired state, and obsolete ones must be destroyed
- ✓ Ensure the correct order of creation
- ✓ Integrate with other tools such as Ansible
- ✓ Platform agnostic
- ✓ Update management
- ✓ Extension capabilities

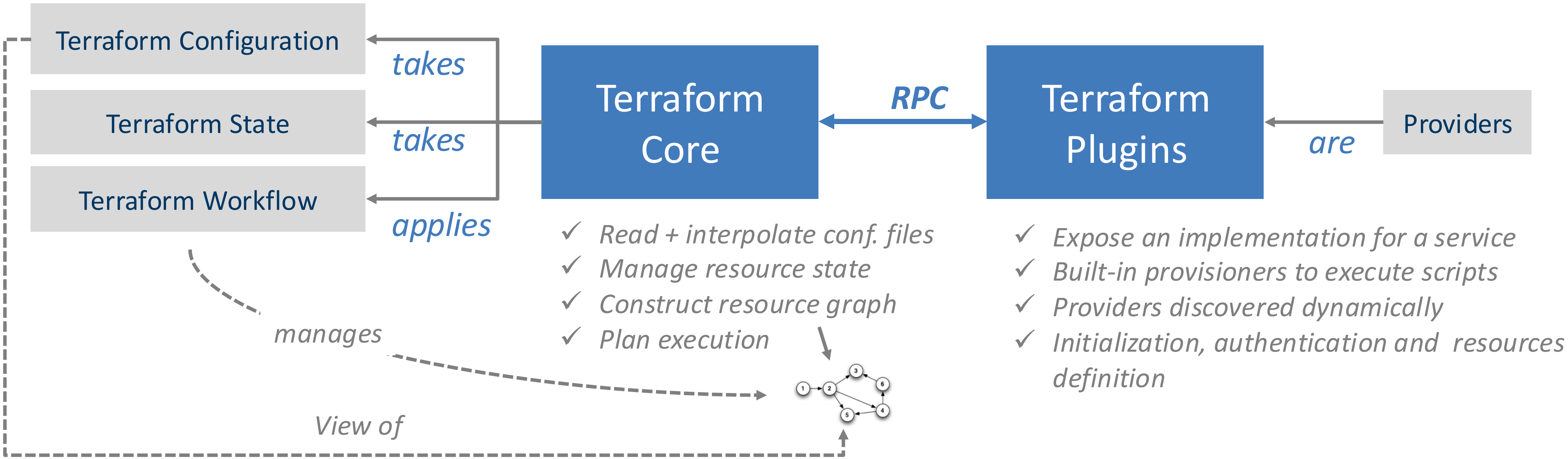
Declarative style



How Terraform operates ?

Terraform examines each resource and uses a graph-based approach to model and apply the desired state.

Each resource is placed inside the graph, its relationships with other resources are calculated, and then each resource is automatically built in the correct order to produce your infrastructure.

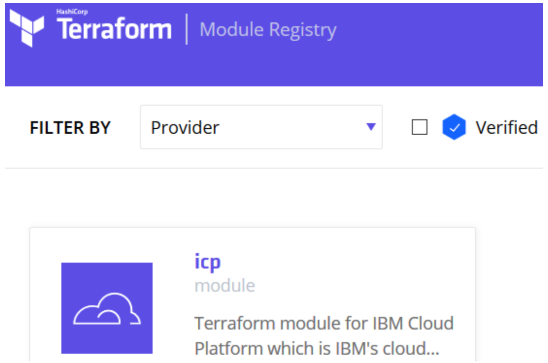


- ✓ *Read + interpolate conf. files*
- ✓ *Manage resource state*
- ✓ *Construct resource graph*
- ✓ *Plan execution*

- ✓ *Expose an implementation for a service*
- ✓ *Built-in provisioners to execute scripts*
- ✓ *Providers discovered dynamically*
- ✓ *Initialization, authentication and resources definition*

Introduction & Concepts

- ➢ You define **resources** as code in Terraform templates
  - ✓ Specify the provider
  - ✓ Specify provisioners
  - ✓ Specify the resources
  - ✓ Parameterize your template using **variables**
- ➢ **Provider**: a source of resources with API endpoint & authentication
- ➢ **Provisioner** execute local or remote scripts during resource creation or destroy time
- ➢ **Data Source**: information read from provider
- ➢ **Modules** allows to reuse the same code in different environments
  - ✓ Modules should be sourced from git tags / branches
  - ✓ Module registry



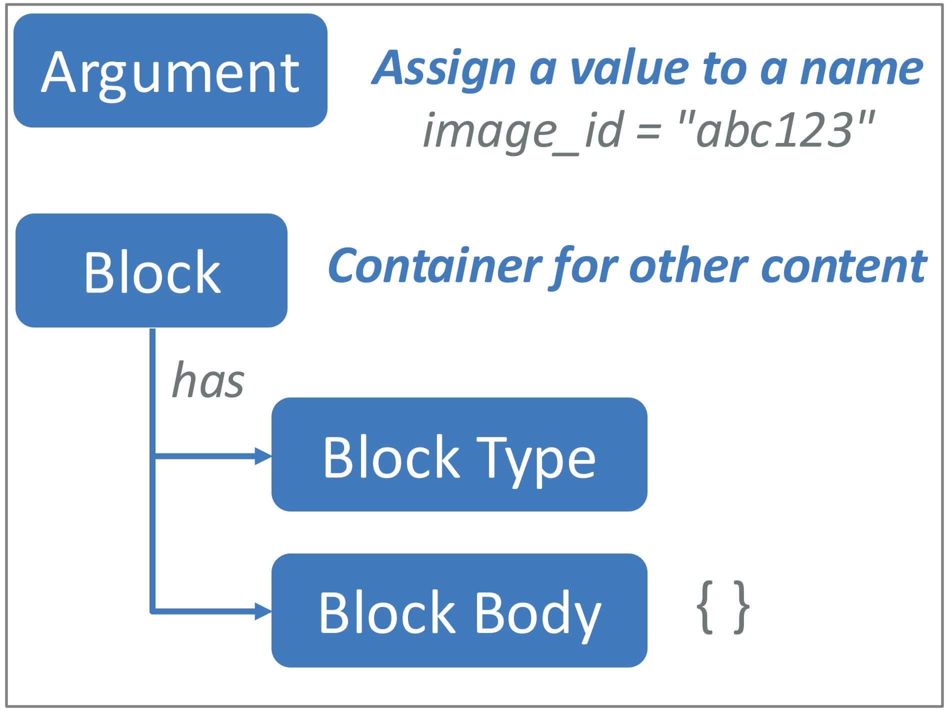
HashiCorp Terraform | Module Registry

FILTER BY Provider ▾ □ Verified

icp
module
Terraform module for IBM Cloud Platform which is IBM's cloud...

- ➢ You follow a **Workflow**:
  - ✓ **Plan** => see what you are about to deploy
  - ✓ **Apply** => to apply the changes
- ➢ Terraform generates a **state file**:
  - ✓ Store information about your managed infrastructure and configuration *(apply execution result)*
  - ✓ Generated during the apply stage, don’t edit manually
- ➢ A **backend** determines how a state is loaded and how operations are executed

Terraform Syntax



Argument

Assign a value to a name

image_id = "abc123"

Block

Container for other content

has

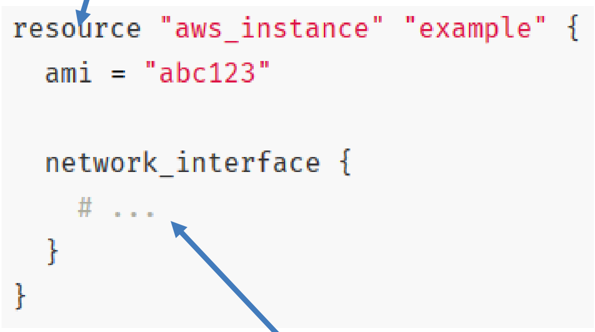
Block Type

Block Body

{ }

type

labels

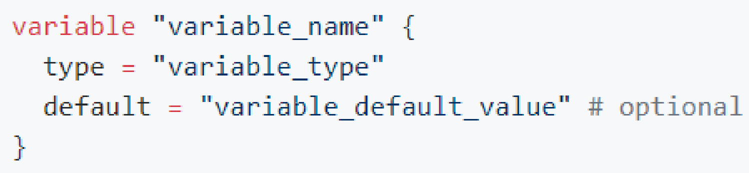


```
resource "aws_instance" "example" {
  ami = "abc123"

  network_interface {
    # ...
  }
}
```

comment

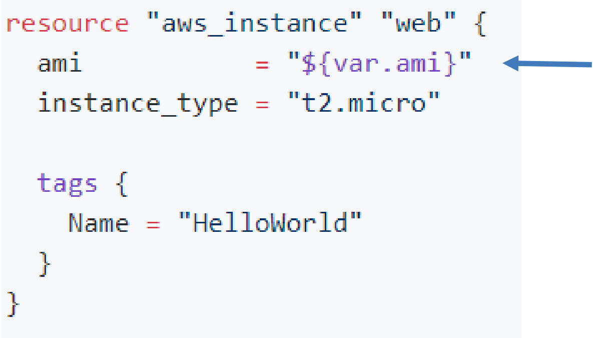
Top level block type

- ✓ *Resource*
- ✓ *Input variable*
- ✓ *Output value*
- ✓ *Data source*



```
variable "variable_name" {
  type = "variable_type"
  default = "variable_default_value" # optional
}
```

- ✓ *string*
- ✓ *map*
- ✓ *list*
- ✓ *boolean*



```
resource "aws_instance" "web" {
  ami           = "${var.ami}"
  instance_type = "t2.micro"

  tags {
    Name = "HelloWorld"
  }
}
```

Variable interpolation

- ✓ *Evaluate the expression*
- ✓ *Convert the value to a string*
- ✓ *Insert it into the string*

Providers

https://www.terraform.io/docs/configuration/providers.html

```
provider "vsphere" {
        version = "~> 1.11.0"
        vsphere_server = "${var.vsphere_server}"

        # if you have a self-signed cert
        allow_unverified_ssl = "${var.allow_unverified_ssl}"
}
```

- The provider block is used to configure the named provider
- A provider is responsible for creating and managing resources
- Multiple provider blocks can exist if a Terraform configuration is composed of multiple providers
- Credential can be passed in different ways (provider dependent)

Data Sources

Used to discover information about existing objects

Can be nested in on other data objects

```
data "vsphere_datacenter" "dc" {
                name = "${var.vsphere_datacenter}"
}

data "vsphere_datastore_cluster" "datastore_cluster" {
                name = "${var.datastore_cluster}"
                datacenter_id = "${data.vsphere_datacenter.dc.id}"
}

data "vsphere_resource_pool" "pool" {
                name =        "${var.vsphere_cluster}/Resources/${var.vsphere_resource_pool}"
                datacenter_id = "${data.vsphere_datacenter.dc.id}"
}

data "vsphere_network" "network" {
                name = "${var.network_label}"
                datacenter_id = "${data.vsphere_datacenter.dc.id}"
}

data "vsphere_virtual_machine" "template" {
                name = "${var.template}"
                datacenter_id = "${data.vsphere_datacenter.dc.id}"
}
```

https://www.terraform.io/docs/configuration/data-sources.html

Resources

Defines a resource that exists within the infrastructure

May be a physical component such as an EC2 instance, or it can be a logical resource such as an SSH key

Resource definition is described by the provider

```
resource "vsphere_virtual_machine" ”camlab" {
        name = "terraform-test"
        resource_pool_id =      "${data.vsphere_resource_pool.pool.id}"
        datastore_id = "${data.vsphere_datastore.datastore.id}"

        num_cpus = 2
        memory = 1024
        guest_id = "other3xLinux64Guest"

        network_interface {
                network_id = "${data.vsphere_network.network.id}"
        }

        disk {
                label = "disk0"
                size = 20
        }
}
```

https://www.terraform.io/docs/configuration/resources.html

Defining Variables

All input variables must be defined in variable blocks

Multiple types of variables string, map, list, etc.

Can define a default or leave default black

Variables that are not defined will need to be defined a run time

```
variable "vsphere_server" {
        description = "vsphere server to connect to"
        default = ”10.0.0.210"
}


variable "camlab" {
        type = "map"

        default = {
                nodes = "1"
                vcpu = "2"
                memory = "4096"
        }
}


variable "dns_servers" {
        description = "DNS Servers to configure on VMs"
        default = ["8.8.8.8", "8.8.4.4"]
}
```

https://www.terraform.io/docs/configuration/variables.html

Referencing Variables

Input variables are referenced in the “${var.<name>}” syntax

Data variables are referenced in the “${data.<name>}” syntax

Local variables are referenced in the “${local.<name>}” syntax

```
vsphere_server = "${var.vsphere_server}"


num_cpus = "${var.camlab["vcpu"]}"


path = "${local.team_folder}"
```

Terraform Primer

Project layout

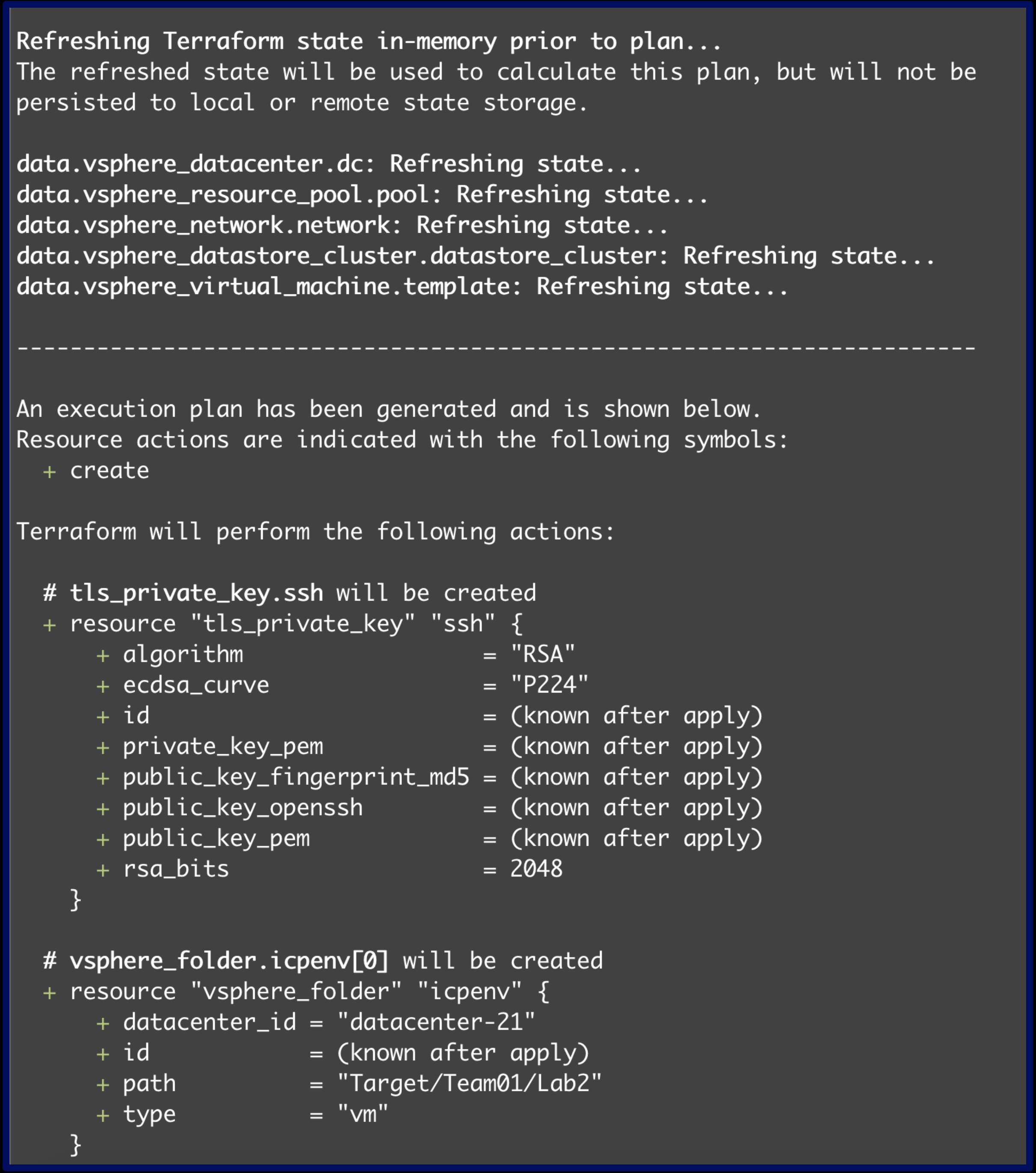
Template planning

terraform plan

- Connects to the hypervisor and performs a dry run to show you what will be created if you apply the template
- Will perform syntax checks and error if you have errors in you code

```
export VSPHERE_USER=administrator@vsphere.local
export VSPHERE_PASSWORD=Passw0rd!
```

Note: You will need to provide credentials to connect your hypervisor

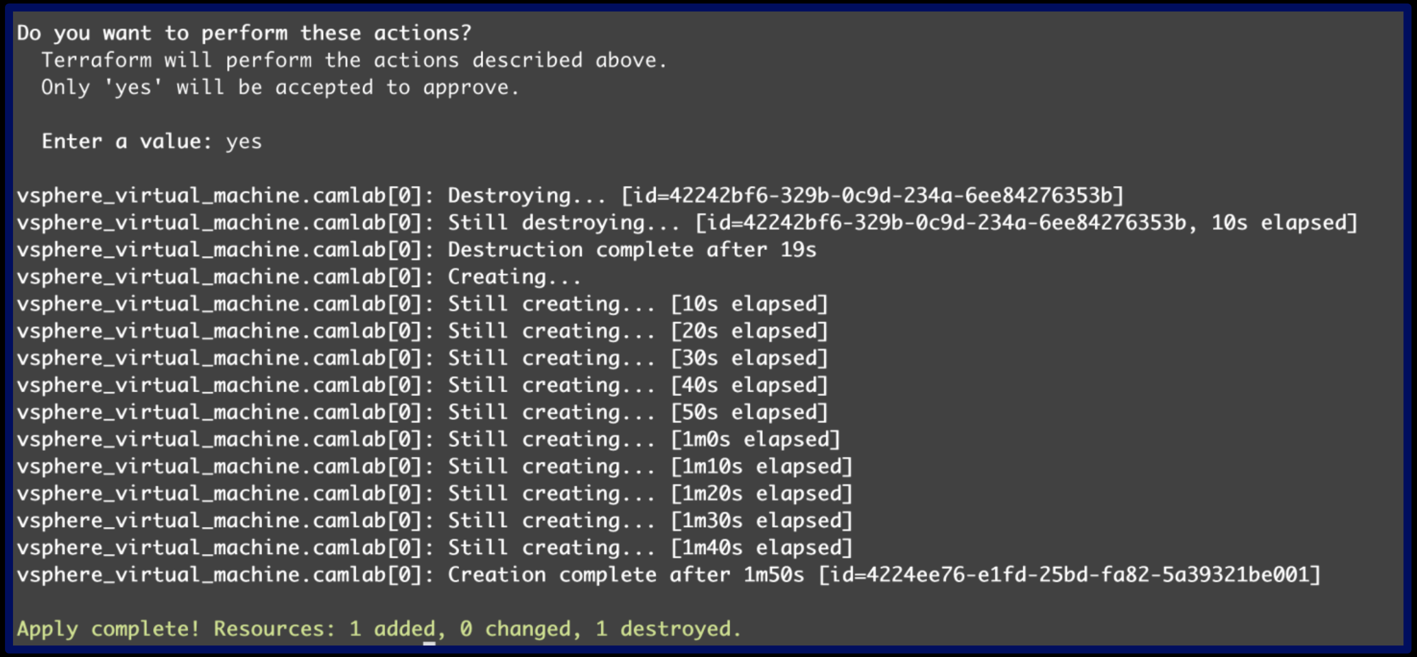


```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

data.vsphere_datacenter.dc: Refreshing state...
data.vsphere_resource_pool.pool: Refreshing state...
data.vsphere_network.network: Refreshing state...
data.vsphere_datastore_cluster.datastore_cluster: Refreshing state...
data.vsphere_virtual_machine.template: Refreshing state...

------------------------------------------------------------------------


An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tls_private_key.ssh will be created
  + resource "tls_private_key" "ssh" {
      + algorithm                  = "RSA"
      + ecdsa_curve                = "P224"
      + id                         = (known after apply)
      + private_key_pem            = (known after apply)
      + public_key_fingerprint_md5 = (known after apply)
      + public_key_openssh         = (known after apply)
      + public_key_pem             = (known after apply)
      + rsa_bits                   = 2048
    }

  # vsphere_folder.icpenv[0] will be created
  + resource "vsphere_folder" "icpenv" {
      + datacenter_id = "datacenter-21"
      + id            = (known after apply)
      + path          = "Target/Team01/Lab2"
      + type          = "vm"
    }
```

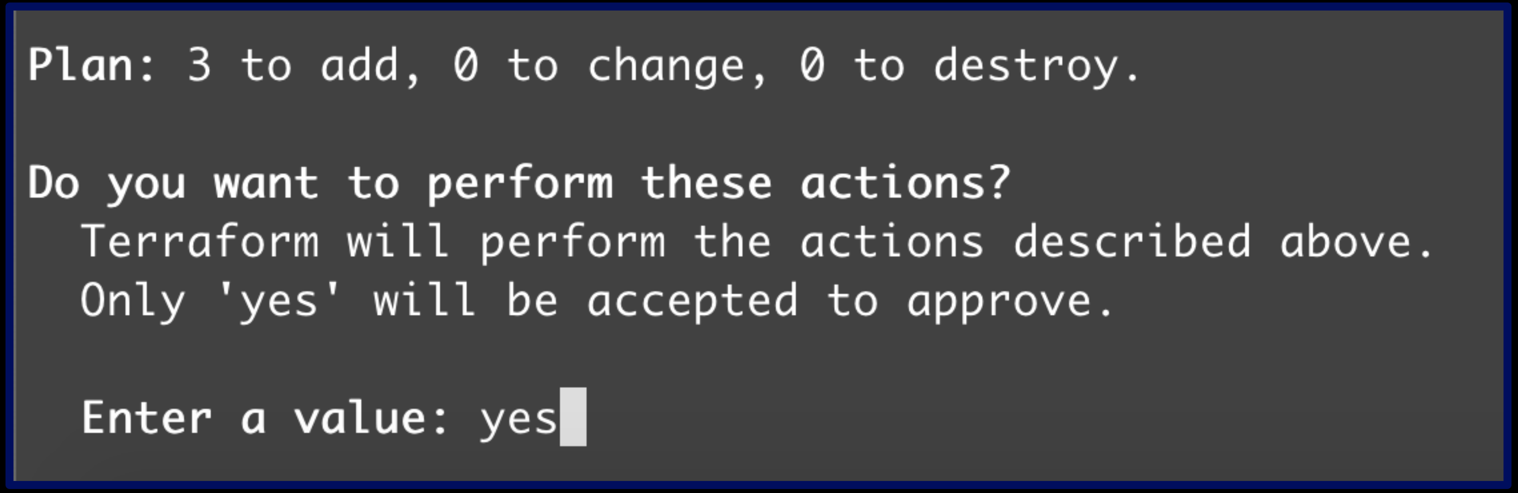
Applying your template

`terraform apply`

- Will first connect to the hypervisor and perform a plan
- Then prompts you to apply the changes that are displayed
- If approved it will apply the changes and create the resources defined



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

vsphere_virtual_machine.camlab[0]: Destroying... [id=42242bf6-329b-0c9d-234a-6ee84276353b]
vsphere_virtual_machine.camlab[0]: Still destroying... [id=42242bf6-329b-0c9d-234a-6ee84276353b, 10s elapsed]
vsphere_virtual_machine.camlab[0]: Destruction complete after 19s
vsphere_virtual_machine.camlab[0]: Creating...
vsphere_virtual_machine.camlab[0]: Still creating... [10s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [20s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [30s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [40s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [50s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [1m0s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [1m10s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [1m20s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [1m30s elapsed]
vsphere_virtual_machine.camlab[0]: Still creating... [1m40s elapsed]
vsphere_virtual_machine.camlab[0]: Creation complete after 1m50s [id=4224ee76-e1fd-25bd-fa82-5a39321be001]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
```

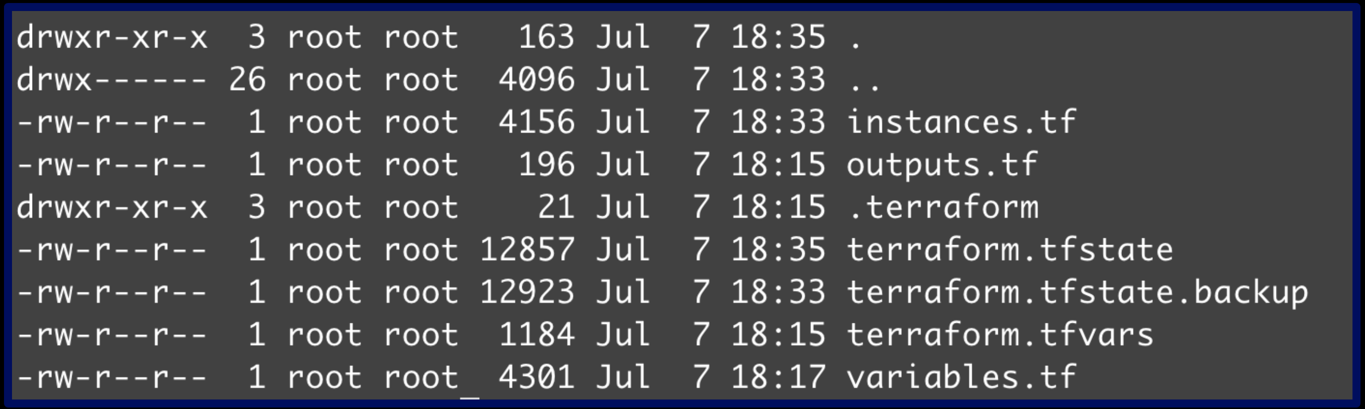


```
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

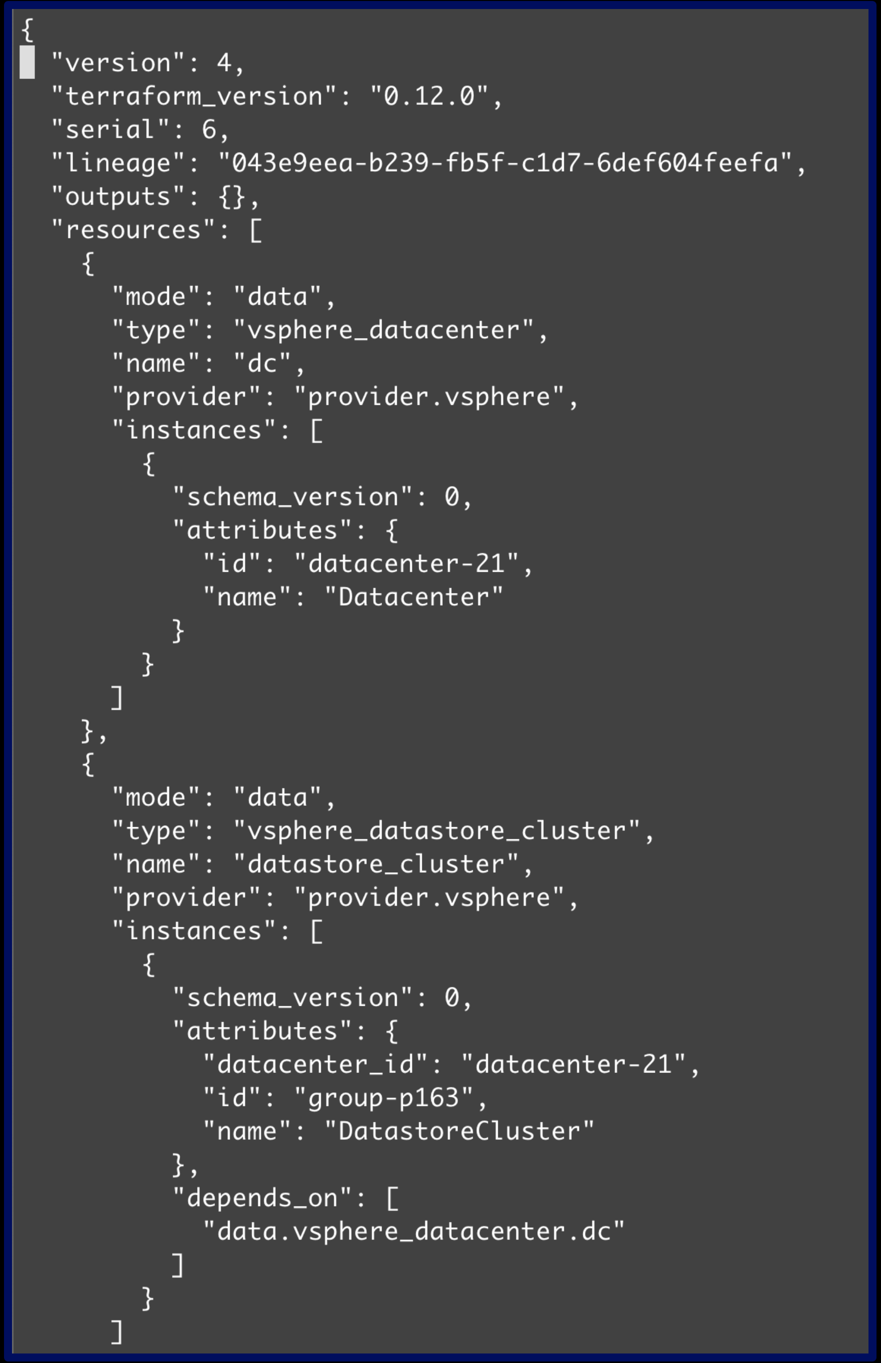
Terraform state files

tfstate

- Tracks the changes that have been made by terraform
- Used to define the declared state and will be compared to actual provider resource to determine configuration change
- JSON format can be queried using **terraform show**



```
drwxr-xr-x  3 root root   163 Jul  7 18:35 .
drwx------ 26 root root  4096 Jul  7 18:33 ..
-rw-r--r--  1 root root  4156 Jul  7 18:33 instances.tf
-rw-r--r--  1 root root   196 Jul  7 18:15 outputs.tf
drwxr-xr-x  3 root root    21 Jul  7 18:15 .terraform
-rw-r--r--  1 root root 12857 Jul  7 18:35 terraform.tfstate
-rw-r--r--  1 root root 12923 Jul  7 18:33 terraform.tfstate.backup
-rw-r--r--  1 root root  1184 Jul  7 18:15 terraform.tfvars
-rw-r--r--  1 root root_ 4301 Jul  7 18:17 variables.tf
```

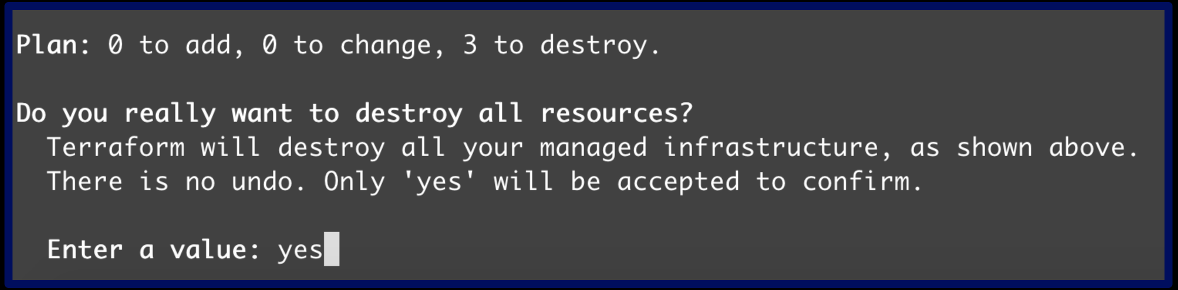


```
{
  "version": 4,
  "terraform_version": "0.12.0",
  "serial": 6,
  "lineage": "043e9eea-b239-fb5f-c1d7-6def604feefa",
  "outputs": {},
  "resources": [
    {
      "mode": "data",
      "type": "vsphere_datacenter",
      "name": "dc",
      "provider": "provider.vsphere",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "id": "datacenter-21",
            "name": "Datacenter"
          }
        }
      ]
    },
    {
      "mode": "data",
      "type": "vsphere_datastore_cluster",
      "name": "datastore_cluster",
      "provider": "provider.vsphere",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "datacenter_id": "datacenter-21",
            "id": "group-p163",
            "name": "DatastoreCluster"
          },
          "depends_on": [
            "data.vsphere_datacenter.dc"
          ]
        }
      ]
```

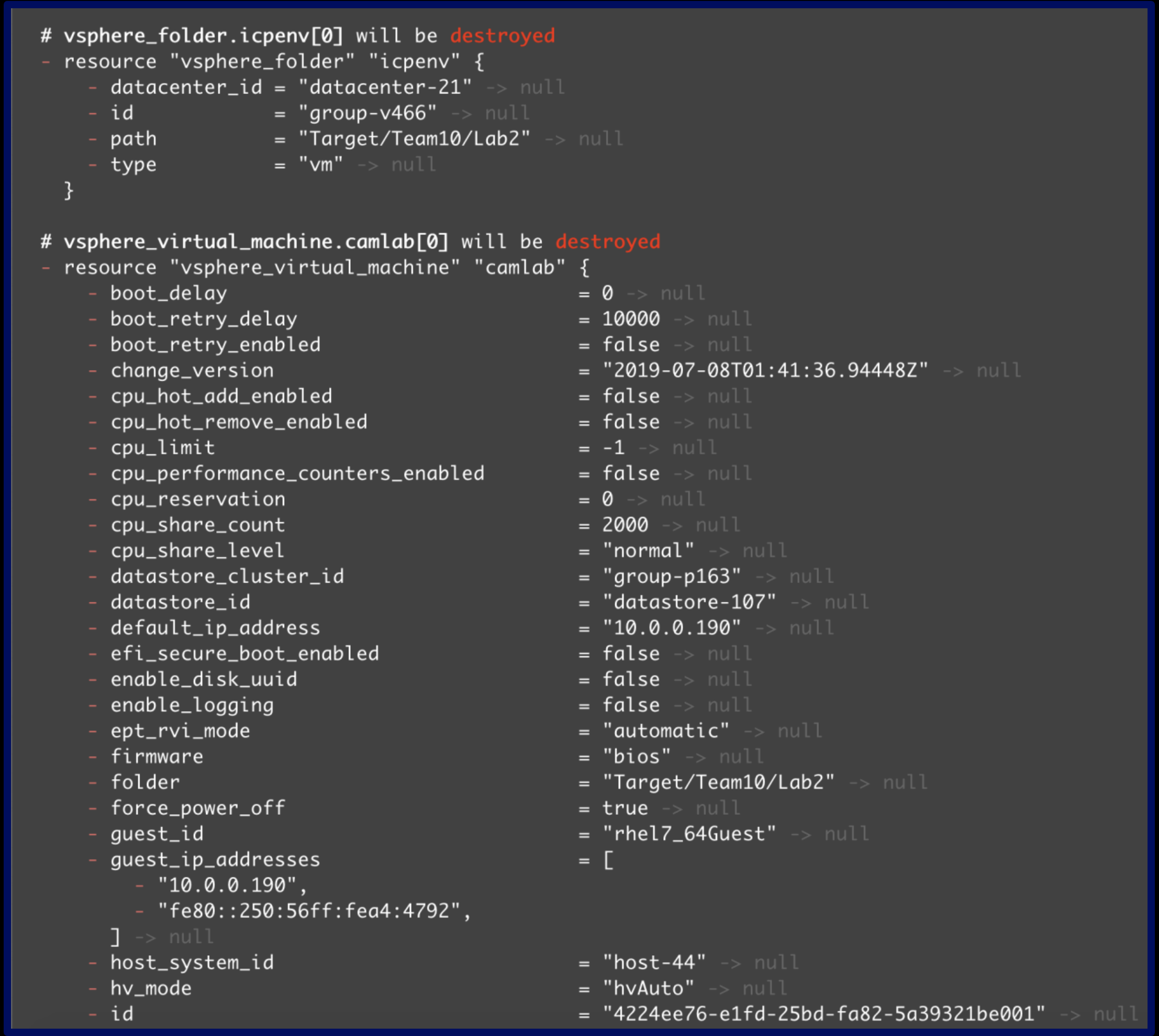
Template Deletion

terraform destroy

- Will first connect to the hypervisor compare what is there to the .tfstate file
- It will show you want is going to be deleted
- Then prompts you to destroy the resources that are displayed
- If approved it will destroy all the resources



```
Plan: 0 to add, 0 to change, 3 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```



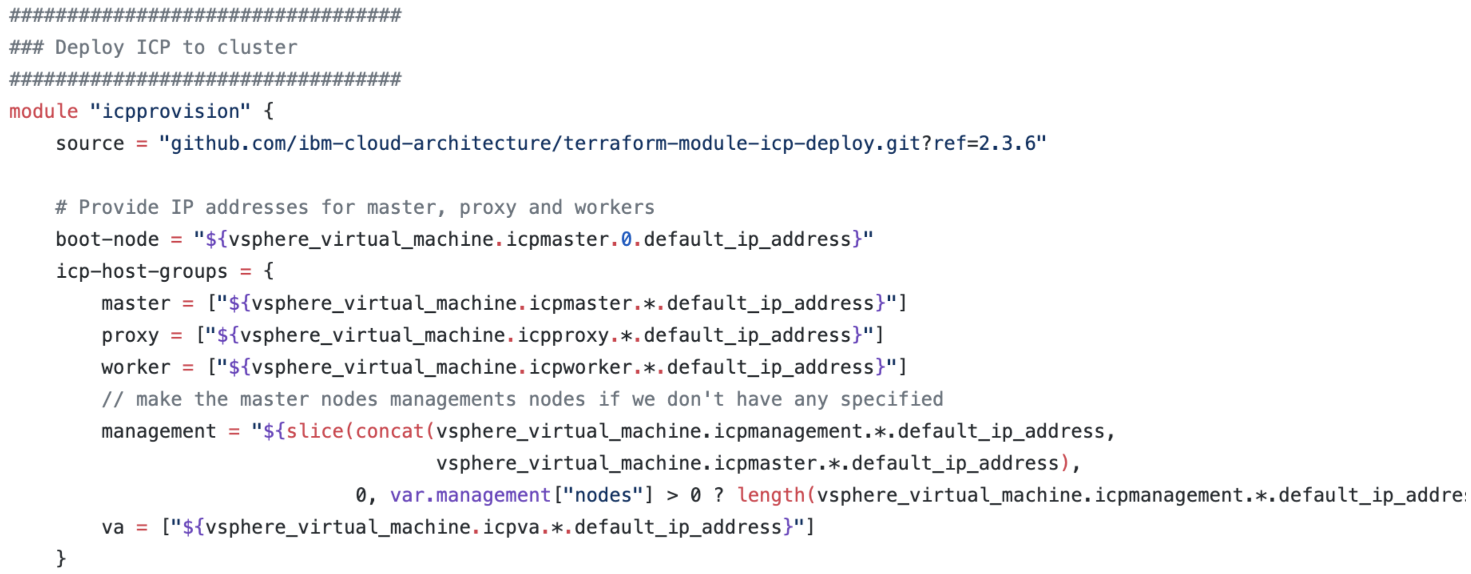
```
# vsphere_folder.icpenv[0] will be destroyed
- resource "vsphere_folder" "icpenv" {
    - datacenter_id = "datacenter-21" -> null
    - id            = "group-v466" -> null
    - path          = "Target/Team10/Lab2" -> null
    - type          = "vm" -> null
  }

# vsphere_virtual_machine.camlab[0] will be destroyed
- resource "vsphere_virtual_machine" "camlab" {
    - boot_delay                              = 0 -> null
    - boot_retry_delay                        = 10000 -> null
    - boot_retry_enabled                      = false -> null
    - change_version                          = "2019-07-08T01:41:36.94448Z" -> null
    - cpu_hot_add_enabled                     = false -> null
    - cpu_hot_remove_enabled                  = false -> null
    - cpu_limit                               = -1 -> null
    - cpu_performance_counters_enabled        = false -> null
    - cpu_reservation                         = 0 -> null
    - cpu_share_count                         = 2000 -> null
    - cpu_share_level                         = "normal" -> null
    - datastore_cluster_id                    = "group-p163" -> null
    - datastore_id                            = "datastore-107" -> null
    - default_ip_address                      = "10.0.0.190" -> null
    - efi_secure_boot_enabled                 = false -> null
    - enable_disk_uuid                        = false -> null
    - enable_logging                          = false -> null
    - ept_rvi_mode                            = "automatic" -> null
    - firmware                                = "bios" -> null
    - folder                                  = "Target/Team10/Lab2" -> null
    - force_power_off                         = true -> null
    - guest_id                                = "rhel7_64Guest" -> null
    - guest_ip_addresses                      = [
        - "10.0.0.190",
        - "fe80::250:56ff:fea4:4792",
      ] -> null
    - host_system_id                          = "host-44" -> null
    - hv_mode                                 = "hvAuto" -> null
    - id                                      = "4224ee76-e1fd-25bd-fa82-5a39321be001" -> null
```

Advanced Topics

Interpolation, modules and providers

Modules

- A *module* is a container for multiple resources that are used together.
- Every Terraform configuration has at least one module, known as its *root module*, which consists of the resources defined in the .tf files in the main working directory.
- A module can call other modules, which lets you include the child module's resources into the configuration in a concise way. Modules can also be called multiple times, either within the same configuration or in separate configurations, allowing resource configurations to be packaged and re-used.

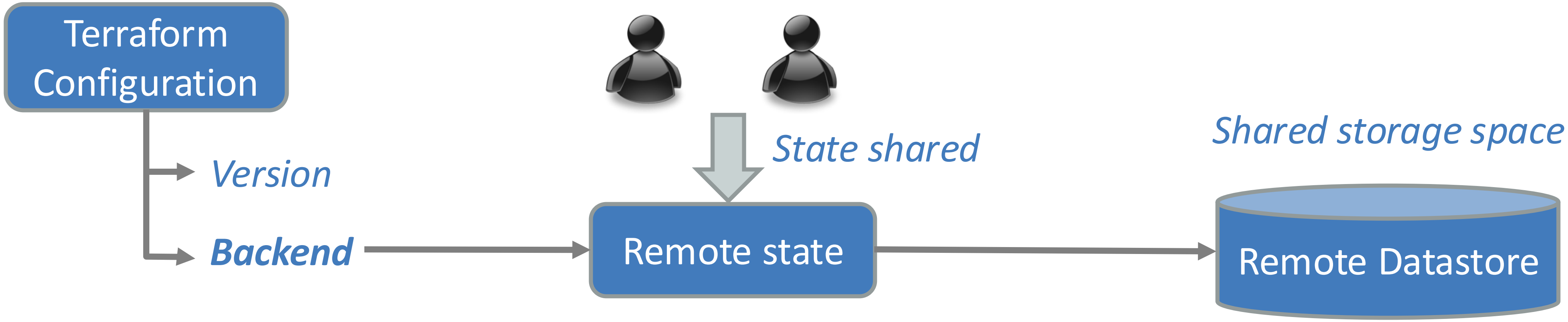


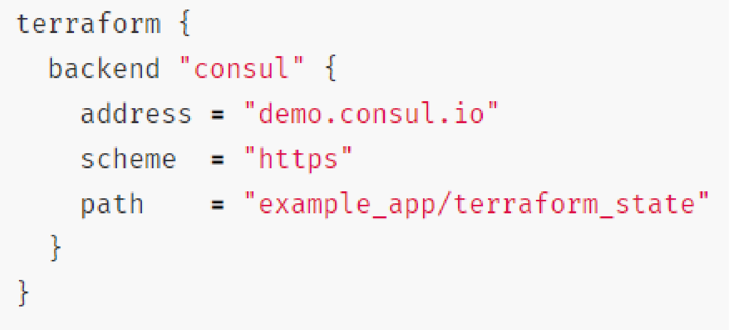
```
##################################
### Deploy ICP to cluster
##################################
module "icpprovision" {
    source = "github.com/ibm-cloud-architecture/terraform-module-icp-deploy.git?ref=2.3.6"

    # Provide IP addresses for master, proxy and workers
    boot-node = "${vsphere_virtual_machine.icpmaster.0.default_ip_address}"
    icp-host-groups = {
        master = ["${vsphere_virtual_machine.icpmaster.*.default_ip_address}"]
        proxy = ["${vsphere_virtual_machine.icpproxy.*.default_ip_address}"]
        worker = ["${vsphere_virtual_machine.icpworker.*.default_ip_address}"]
        // make the master nodes managements nodes if we don't have any specified
        management = "${slice(concat(vsphere_virtual_machine.icpmanagement.*.default_ip_address,
                                     vsphere_virtual_machine.icpmaster.*.default_ip_address),
                              0, var.management["nodes"] > 0 ? length(vsphere_virtual_machine.icpmanagement.*.default_ip_addre
        va = ["${vsphere_virtual_machine.icpva.*.default_ip_address}"]
    }
```

https://www.terraform.io/docs/configuration/modules.html

Store your state file remotely using a terraform backend

- Protect state with locks to prevent corruption
- Init your backend: *terraform init*

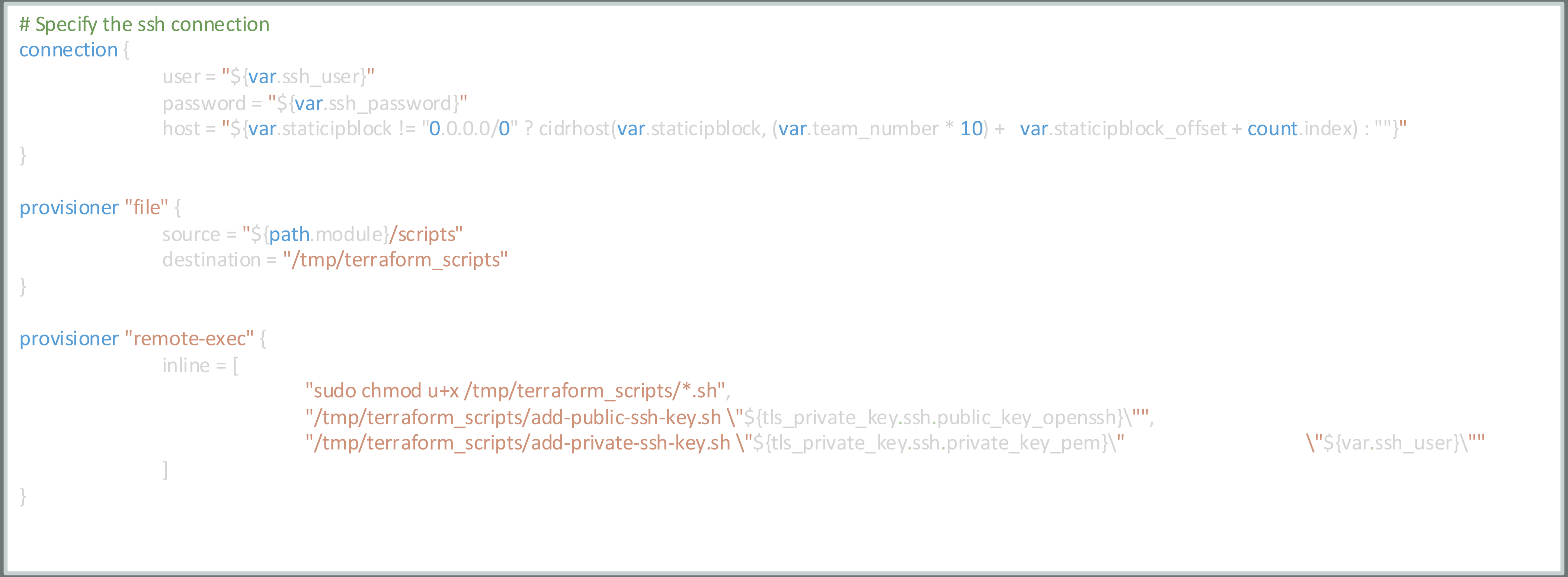




```
terraform {
  backend "consul" {
    address = "demo.consul.io"
    scheme  = "https"
    path    = "example_app/terraform_state"
  }
}
```

- ✓ *Work in a team*
- ✓ *Team work => state locking*
- ✓ *Remote operations*
- ✓ *Keep sensitive information off disk*
- ✓ *Persistent data stored in the backend belongs to a workspace*

Provisioners are used to execute scripts on a local or remote machine as part of resource creation or destruction. Provisioners can be used to bootstrap a resource, cleanup before destroy, run configuration management, etc.



```
# Specify the ssh connection
connection {
            user = "${var.ssh_user}"
            password = "${var.ssh_password}"
            host = "${var.staticipblock != "0.0.0.0/0" ? cidrhost(var.staticipblock, (var.team_number * 10) +  var.staticipblock_offset + count.index) : ""}"
}

provisioner "file" {
            source = "${path.module}/scripts"
            destination = "/tmp/terraform_scripts"
}

provisioner "remote-exec" {
            inline = [
                        "sudo chmod u+x /tmp/terraform_scripts/*.sh",
                        "/tmp/terraform_scripts/add-public-ssh-key.sh \"${tls_private_key.ssh.public_key_openssh}\"",
                        "/tmp/terraform_scripts/add-private-ssh-key.sh \"${tls_private_key.ssh.private_key_pem}\"                \"${var.ssh_user}\""
            ]
}
```

Provisioners

https://www.terraform.io/docs/provisioners/index.html

Ansible Primer

Inventory, Variables, Playbooks, Tasks

Inventory

Used to declare and group hosts together

Default ansible inventory
`/etc/ansible/hosts`

```
mail.example.com

[webservers]
foo.example.com
bar.example.com
web[01:10].example.com

[dbservers]
one.example.com
two.example.com
three.example.com
db-[a:f].example.com
```

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Groups

Collection of hosts that share similar functions

```
mail.example.com

[web]
web01.example.com
web02.example.com

[db]
db01.example.com
db02.example.com

[dev]
web01.example.com
db01.example.com

[prod]
web02.example.com
db02.example.com
```

Special Groups:

all: every host in inventory

unbound: any host not in a group

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Variables

variable values that relate to a specific host or group

in inventory

```
mail.example.com

[web]
web01.example.com
web02.example.com

[db]
db01.example.com
db02.example.com

[web:vars]
web_root=/var/www/html

[db:vars]
db_name=database_name
```

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Playbooks

The basis for a really simple configuration management and multi-machine deployment system, and one that is very well suited to deploying complex applications.

They declare configurations, but they can also orchestrate steps of any manual ordered process, even as different steps must bounce back and forth between sets of machines in particular orders.

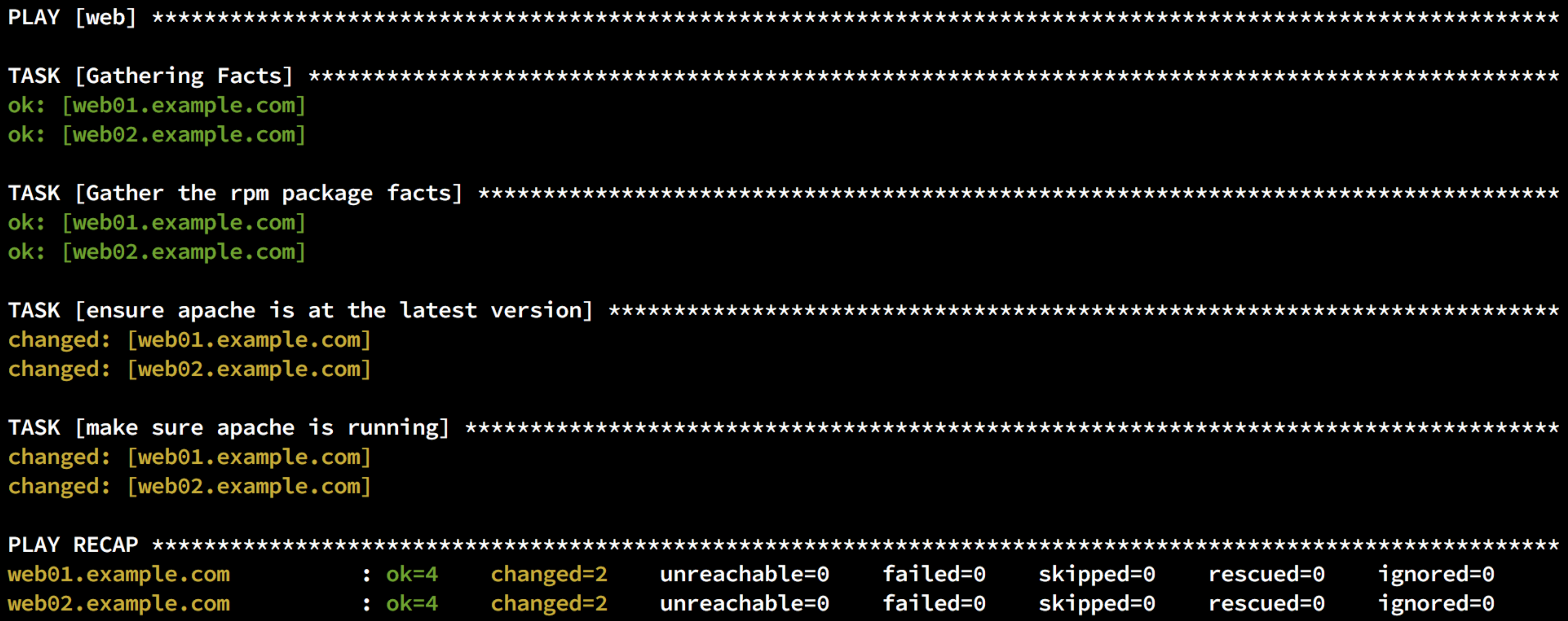
```
---
 - hosts: web
 remote_user: virtuser
 become: yes
 tasks:
 - name: Gather the rpm package facts
  package_facts:
   manager: auto
 - name: ensure apache is at the latest version
  yum:
   name: httpd
   state: latest
  when: "'httpd' not in ansible_facts.packages"
 - name: make sure apache is running
  service:
   name: httpd
   state: started
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

Executing Playbooks

```
ansible-playbook -i /path/to/inventory /path/to/playbook.yaml
```

- You can specify a specific inventory with the -i parameter

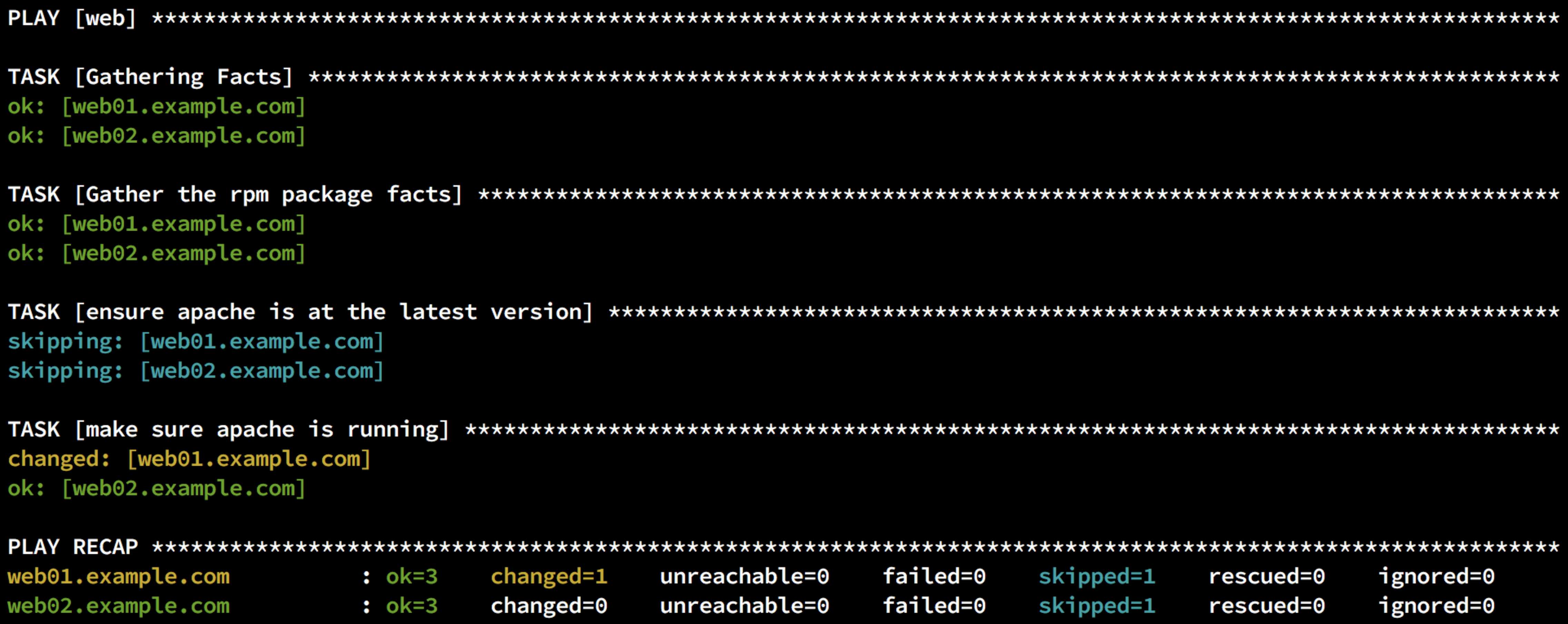


```
PLAY [web] *************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************
ok: [web01.example.com]
ok: [web02.example.com]

TASK [Gather the rpm package facts] ************************************************************************************
ok: [web01.example.com]
ok: [web02.example.com]

TASK [ensure apache is at the latest version] **************************************************************************
changed: [web01.example.com]
changed: [web02.example.com]

TASK [make sure apache is running] *************************************************************************************
changed: [web01.example.com]
changed: [web02.example.com]

PLAY RECAP *************************************************************************************************************
web01.example.com          : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web02.example.com          : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

State Management

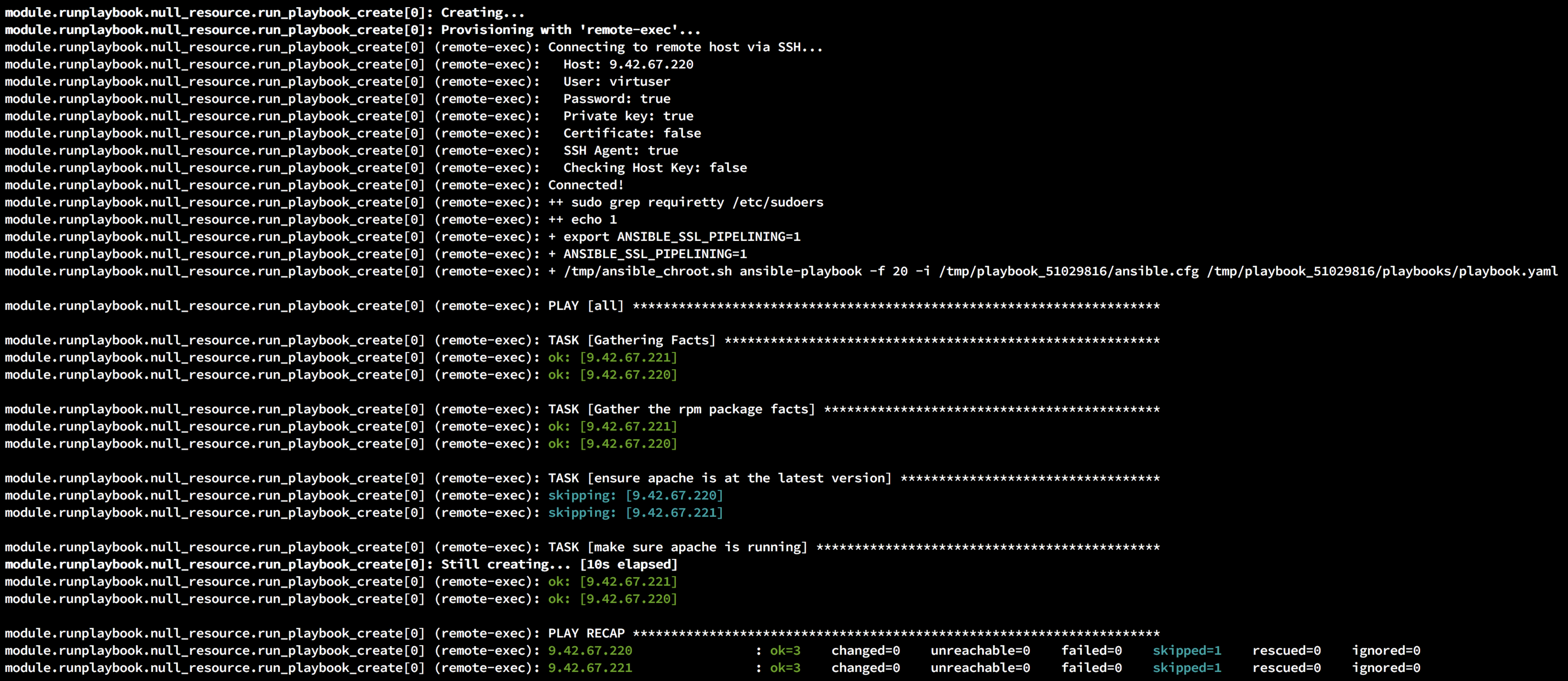
```
ansible-playbook -i /path/to/inventory /path/to/playbook.yaml
```

- What happens if I execute the same playbook again?



```
PLAY [web] *************************************************************************************************************

TASK [Gathering Facts] *************************************************************************************************
ok: [web01.example.com]
ok: [web02.example.com]

TASK [Gather the rpm package facts] ************************************************************************************
ok: [web01.example.com]
ok: [web02.example.com]

TASK [ensure apache is at the latest version] **************************************************************************
skipping: [web01.example.com]
skipping: [web02.example.com]

TASK [make sure apache is running] *************************************************************************************
changed: [web01.example.com]
ok: [web02.example.com]

PLAY RECAP *************************************************************************************************************
web01.example.com          : ok=3    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
web02.example.com          : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Ansible + Terraform



```
module.runplaybook.null_resource.run_playbook_create[0]: Creating...
module.runplaybook.null_resource.run_playbook_create[0]: Provisioning with 'remote-exec'...
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): Connecting to remote host via SSH...
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   Host: 9.42.67.220
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   User: virtuser
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   Password: true
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   Private key: true
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   Certificate: false
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   SSH Agent: true
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec):   Checking Host Key: false
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): Connected!
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ++ sudo grep requiretty /etc/sudoers
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ++ echo 1
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): + export ANSIBLE_SSL_PIPELINING=1
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): + ANSIBLE_SSL_PIPELINING=1
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): + /tmp/ansible_chroot.sh ansible-playbook -f 20 -i /tmp/playbook_51029816/ansible.cfg /tmp/playbook_51029816/playbooks/playbook.yaml

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): PLAY [all] *********************************************************************

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): TASK [Gathering Facts] *********************************************************
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.221]
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.220]

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): TASK [Gather the rpm package facts] ********************************************
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.221]
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.220]

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): TASK [ensure apache is at the latest version] **********************************
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): skipping: [9.42.67.220]
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): skipping: [9.42.67.221]

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): TASK [make sure apache is running] *********************************************
module.runplaybook.null_resource.run_playbook_create[0]: Still creating... [10s elapsed]
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.221]
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): ok: [9.42.67.220]

module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): PLAY RECAP *********************************************************************
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): 9.42.67.220                : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
module.runplaybook.null_resource.run_playbook_create[0] (remote-exec): 9.42.67.221                : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

https://github.com/ibm-cloud-architecture/terraform-ansible-runplaybooks

Code walkthrough

Simple to Complex

Code Samples

Basic Example
https://github.ibm.com/john-webb/cam4admins/tree/master/Terraform/Lab2

Advanced Examples (OCP in different Cloud providers)
https://github.com/ibm-cloud-architecture/terraform-openshift4-aws
https://github.com/ibm-cloud-architecture/terraform-openshift4-azure
https://github.com/ibm-cloud-architecture/terraform-openshift4-gcp

Enterprise Environments

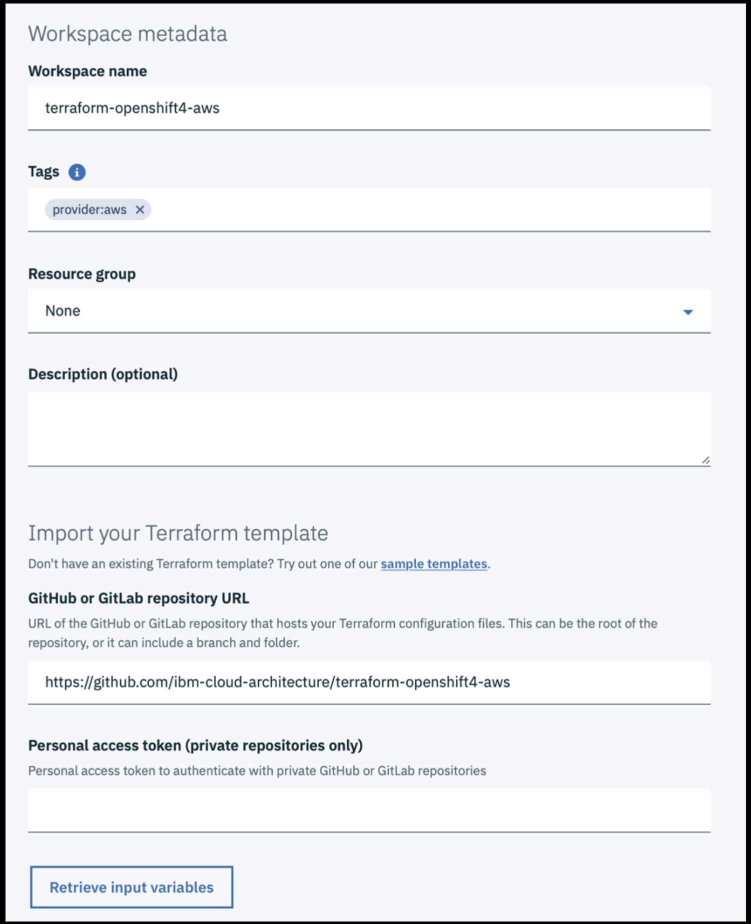
IBM Schematics

Cloud Automation Manager

Ansible Tower

IBM Schematics

IBM Cloud Schematics delivers Terraform-as-a-Service so that you can use a high-level scripting language to model the resources that you want in your IBM Cloud environment, and enable Infrastructure as Code



Workspace metadata

Workspace name

terraform-openshift4-aws

Tags

provider:aws ✕

Resource group

None ▾

Description (optional)

Import your Terraform template

Don't have an existing Terraform template? Try out one of our sample templates.

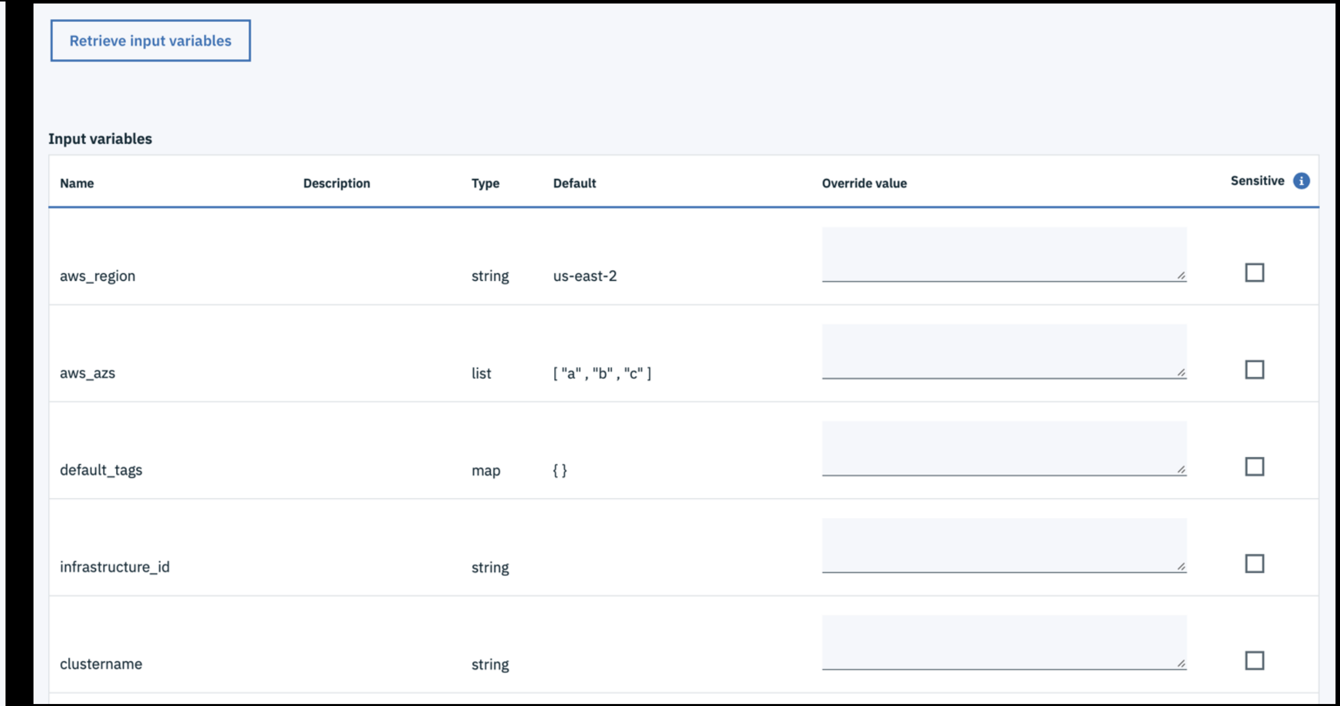
GitHub or GitLab repository URL

URL of the GitHub or GitLab repository that hosts your Terraform configuration files. This can be the root of the repository, or it can include a branch and folder.

https://github.com/ibm-cloud-architecture/terraform-openshift4-aws

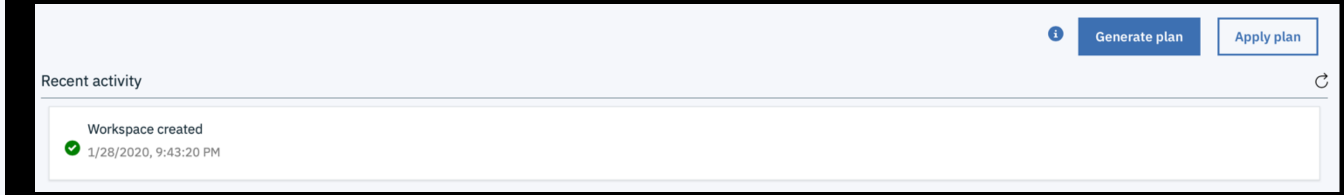
Personal access token (private repositories only)

Personal access token to authenticate with private GitHub or GitLab repositories



Input variables

Name	Description	Type	Default	Override value	Sensitive
aws_region		string	us-east-2		□
aws_azs		list	["a" , "b" , "c"]		□
default_tags		map	{ }		□
infrastructure_id		string			□
clustername		string			□



Recent activity

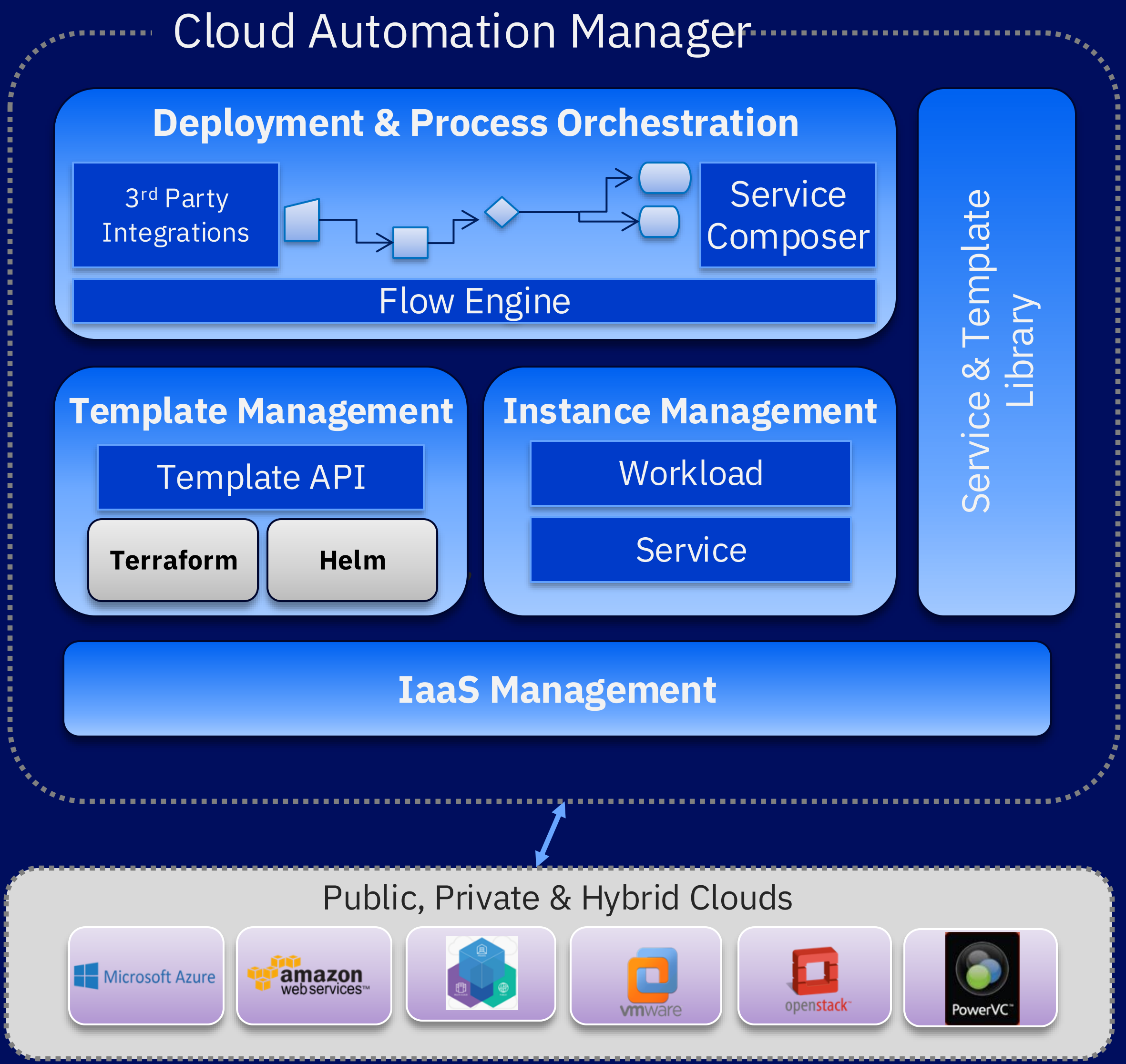
Workspace created
1/28/2020, 9:43:20 PM

https://cloud.ibm.com/docs/schematics

IBM Cloud Automation Manager

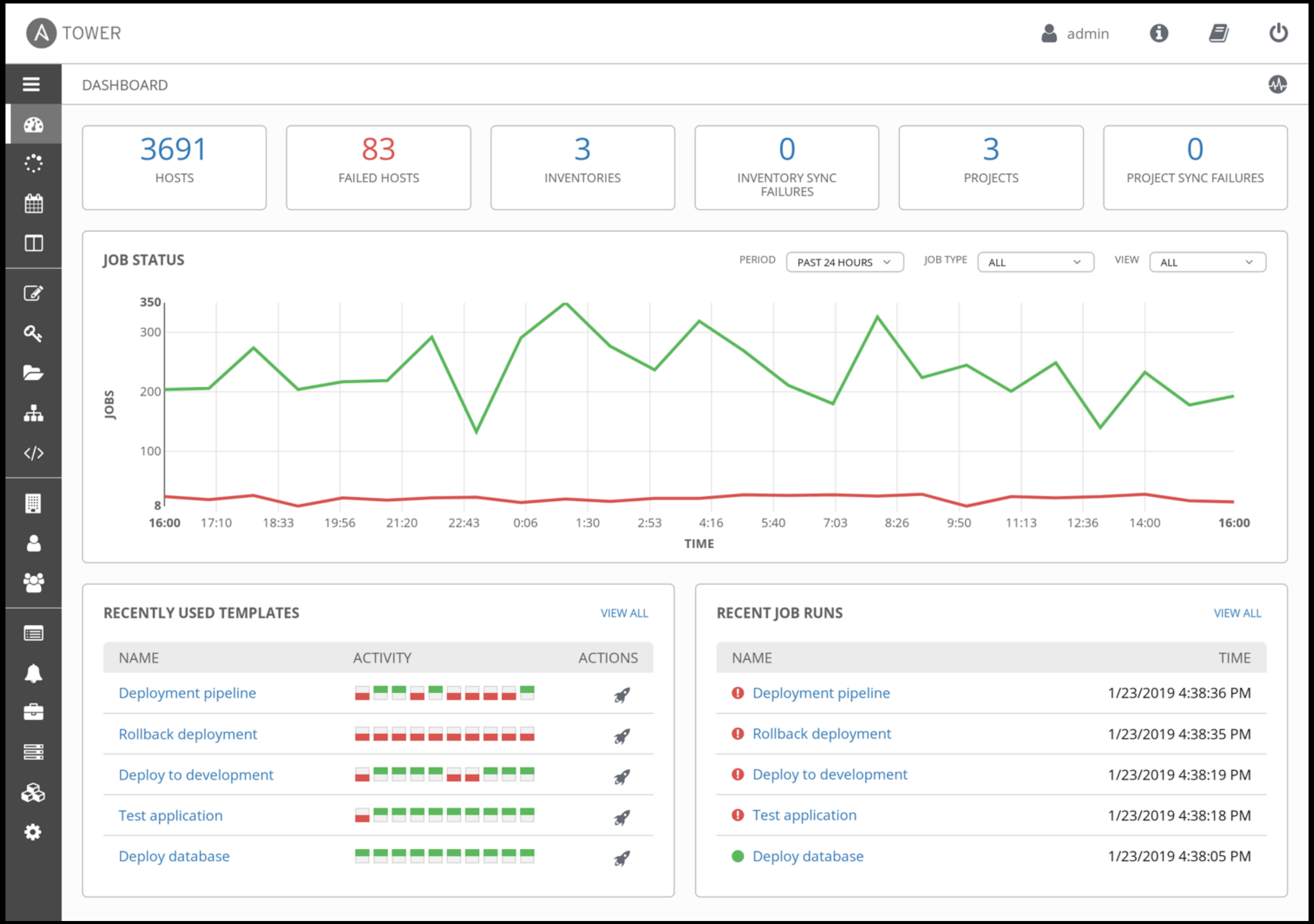
Full stack automation and service orchestration

- ***Automated provisioning –*** *Automated provisioning of infrastructure and applications with workflow orchestration*
- ***Self-service -*** Self-service access to cloud infrastructure and application services
- ***Manage and govern –*** *Manage and govern workloads across multiple and hybrid clouds*
- ***Built with open technology*** *to avoid vendor lock-in*



RedHat Ansible Tower

With Red Hat® Ansible® Tower you can centralize and control your IT infrastructure with a visual dashboard, role-based access control, job scheduling, integrated notifications and graphical inventory management



TOWER admin

DASHBOARD

3691	83	3	0	3	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

JOB STATUS

PERIOD: PAST 24 HOURS | JOB TYPE: ALL | VIEW: ALL

RECENTLY USED TEMPLATES VIEW ALL

NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	🚀
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	🚀
Deploy to development	🟥🟩🟩🟩🟩🟥🟥🟩🟩🟩	🚀
Test application	🟥🟩🟩🟩🟩🟩🟩🟩🟩🟩	🚀
Deploy database	🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩	🚀

RECENT JOB RUNS VIEW ALL

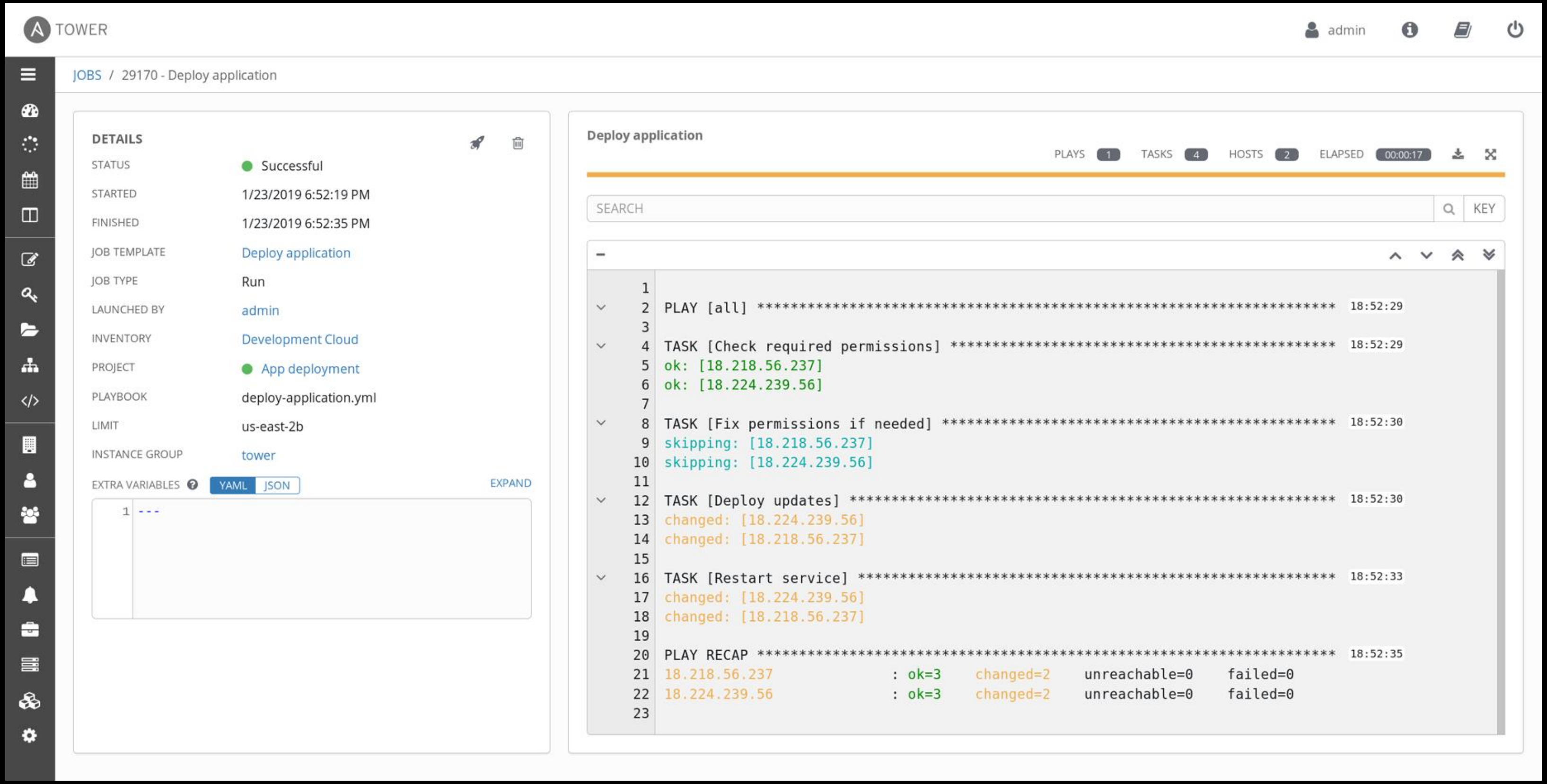
NAME	TIME
● Deployment pipeline	1/23/2019 4:38:36 PM
● Rollback deployment	1/23/2019 4:38:35 PM
● Deploy to development	1/23/2019 4:38:19 PM
● Test application	1/23/2019 4:38:18 PM
● Deploy database	1/23/2019 4:38:05 PM

Dashboard

NOC-style display with access to all your environment data, recent activity, etc

https://www.ansible.com/products/tower

RedHat Ansible Tower



DETAILS

STATUS ● Successful
STARTED 1/23/2019 6:52:19 PM
FINISHED 1/23/2019 6:52:35 PM
JOB TEMPLATE Deploy application
JOB TYPE Run
LAUNCHED BY admin
INVENTORY Development Cloud
PROJECT ● App deployment
PLAYBOOK deploy-application.yml
LIMIT us-east-2b
INSTANCE GROUP tower
EXTRA VARIABLES

```
---
```

Deploy application

PLAYS 1 TASKS 4 HOSTS 2 ELAPSED 00:00:17

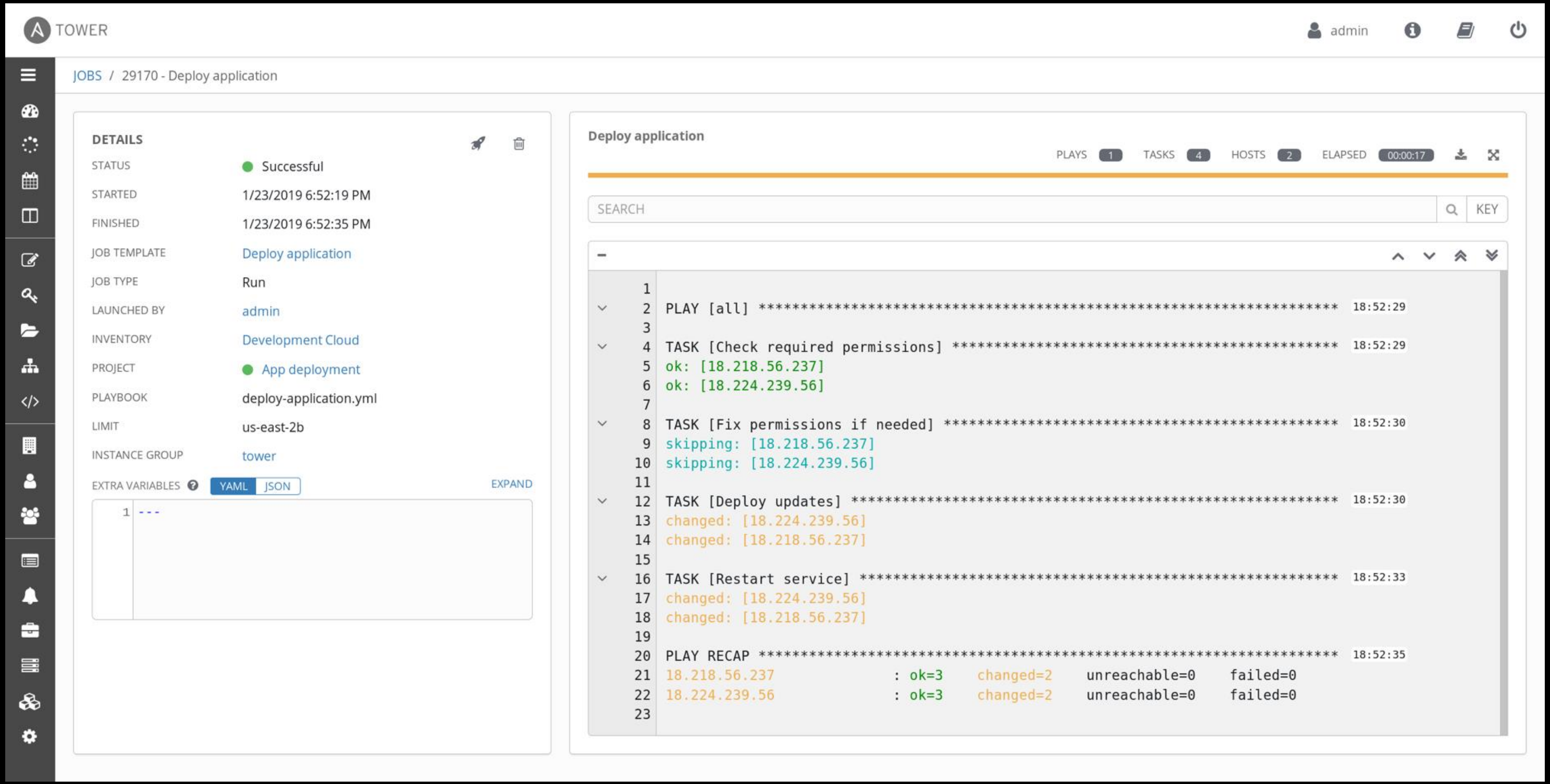
```

PLAY [all] ********************************************************************  18:52:29

TASK [Check required permissions] *********************************************  18:52:29
ok: [18.218.56.237]
ok: [18.224.239.56]

TASK [Fix permissions if needed] **********************************************  18:52:30
skipping: [18.218.56.237]
skipping: [18.224.239.56]

TASK [Deploy updates] *********************************************************  18:52:30
changed: [18.224.239.56]
changed: [18.218.56.237]

TASK [Restart service] ********************************************************  18:52:33
changed: [18.224.239.56]
changed: [18.218.56.237]

PLAY RECAP ********************************************************************  18:52:35
18.218.56.237              : ok=3    changed=2    unreachable=0    failed=0
18.224.239.56              : ok=3    changed=2    unreachable=0    failed=0

```

Real-time status updates

Playbook streams in real-time so you can see success and failures of tasks and jobs.

https://www.ansible.com/products/tower

RedHat Ansible Tower



DETAILS

STATUS	● Successful
STARTED	1/23/2019 6:52:19 PM
FINISHED	1/23/2019 6:52:35 PM
JOB TEMPLATE	Deploy application
JOB TYPE	Run
LAUNCHED BY	admin
INVENTORY	Development Cloud
PROJECT	● App deployment
PLAYBOOK	deploy-application.yml
LIMIT	us-east-2b
INSTANCE GROUP	tower
EXTRA VARIABLES	YAML JSON

```
---
```

Deploy application

```

PLAY [all] *********************************************************************  18:52:29

TASK [Check required permissions] **********************************************  18:52:29
ok: [18.218.56.237]
ok: [18.224.239.56]

TASK [Fix permissions if needed] ***********************************************  18:52:30
skipping: [18.218.56.237]
skipping: [18.224.239.56]

TASK [Deploy updates] **********************************************************  18:52:30
changed: [18.224.239.56]
changed: [18.218.56.237]

TASK [Restart service] *********************************************************  18:52:33
changed: [18.224.239.56]
changed: [18.218.56.237]

PLAY RECAP *********************************************************************  18:52:35
18.218.56.237              : ok=3    changed=2    unreachable=0    failed=0
18.224.239.56              : ok=3    changed=2    unreachable=0    failed=0

```

Activity Stream

See **who** ran **what** job **when**

Complete audit trail of all changes made to Ansible Tower itself - job creation, inventory changes, credential storage, all securely tracked.

All audit and log information can be sent to your external logging and analytics provider

https://www.ansible.com/products/tower

CONCLUSION

- Use the right tool for the right job
- Terraform for Infrastructure
- Ansible for Configuration Management
- Leverage Enterprise Solutions