HASHICORP TERRAFORM AND RED HAT ANSIBLE AUTOMATION

Infrastructure as code automation

OVERVIEW

INTRODUCTION

As organizations modernize their application delivery process and adopt new tools to make them more efficient, infrastructure as code (IaC) has become a critical practice. IaC is an umbrella term, usually referring to a broad range of tools, operational frameworks, and a rich ecosystem. In this piece, we describe what is meant by infrastructure as code and how HashiCorp Terraform and Red Hat® Ansible® Automation can be used as a foundation for an IaC initiative.

WHAT IS INFRASTRUCTURE AS CODE?

In the last four decades, most approaches to infrastructure management have relied on graphical and command-line user interfaces that human operators have used to initialize or modify system configurations. These approaches are prone to human error. The process carried out by an operator might be documented by a checklist or a memorized routine without enforced oversight or iterative versioning.

In contrast, the IaC approach promotes formalized, standardized, and automated operational processes—and dictates that these operational processes are documented as configuration files or programming code.

By treating infrastructure as code, IT organizations can automate management tasks while using the best practices of software development, including code review and version control. This approach mitigates management complexity by breaking down a task into smaller, more manageable processes, controlling the execution of code, and effortlessly maintaining up-to-date documentation.

The IaC approach also reduces operational risks by allowing multiple subject matter experts to peer review the code and by saving all the previous revisions of a codified infrastructure, enabling previous versions to be restored in case of mistakes.

Ultimately, the IaC approach mitigates human errors by enforcing an automated execution of the management task performed on the IT infrastructure.

INFRASTRUCTURE AS CODE WITH TERRAFORM AND ANSIBLE AUTOMATION

To address the variety of operational challenges that an IT organization faces in large-scale environments, the industry has developed many tools with specific focus areas.

Two of the most daunting operational challenges in modern environments are infrastructure provisioning and configuration management. As IT environments grow in scale and complexity, human errors multiply across all aspects of infrastructure and application life cycle—from provisioning to configuration, through patching and security enforcement, all the way to final decommissioning.

HashiCorp Terraform is an IaC tool for provisioning and managing IT resources. Configuration files describe to Terraform the components needed to run a single application—or your entire datacenter. Terraform focuses on the higher-level abstraction of the datacenter and associated services without sacrificing the ability to use other tools to perform post-creation tasks.

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

For many resources, there is additional configuration and setup beyond just creating the resource. Terraform can invoke external tools, such as Red Hat Ansible Automation, after creating or before destroying resources to perform additional configuration and other application-specific tasks.

The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as domain name system (DNS) entries and Software-as-a-Service (SaaS) features.

Red Hat Ansible Automation is an IaC solution that can automate a wide range of management tasks across heterogeneous environments. For example, Ansible Automation is designed to orchestrate complex multitier application deployments to automate the human workflow that comprises real-world application stacks.

Configuration files describe to Ansible Automation the desired state of the target IT system—and how to reach that desired state. Ansible Automation also supports a dry run mode to preview needed changes, allowing an understanding of what will be changed without impacting a running system. As the configuration changes over time to adapt to the needs of the business, Ansible Automation is able to incrementally modify the machine to apply the changes.

Modern IT organizations face enormous challenges to deliver increasingly more agile environments. In most cases, no single tool can address all the needs and perform all tasks efficiently. To solve this problem, Ansible Automation provides an automation framework designed to easily integrate with and complement third-party solutions. Ansible Automation can perform automation tasks itself or invoke external tools, such as HashiCorp Terraform, allowing IT organizations to use their preferred tools to automate various aspects of IT operations.

ANSIBLE AUTOMATION AND TERRAFORM INTEGRATION

The challenges of provisioning and configuration management are closely related, as IT organizations usually configure operating systems, middleware, and applications immediately after deploying the infrastructure resources to support them. IT organizations can automate the entire process by using Terraform for infrastructure provisioning and Ansible Automation for infrastructure configuration and application deployment.

Both Ansible Automation and Terraform operate using an agentless architecture. Both also promote a simplified human-readable interface for defining automation, making adoption easier for enterprise and community users alike. Because of these similarities, these products are frequently used together.

While this combination leads to some overlap in capabilities, Terraform and Ansible Automation can effectively build the foundation of an IaC initiative. For example, Terraform can be used for infrastructure provisioning and decommissioning while Ansible Automation can be used for infrastructure configuration and patching, as well as application deployment and maintenance.

Terraform and Ansible Automation can be integrated in different ways, depending on the operating model of the IT organization:

- Terraform invoking Ansible Automation
- Ansible Automation invoking Terraform

Where you initiate your automation depends on the approach taken to resource management and the scope at which your automation is being addressed. There is no right or wrong answer—the decision largely depends on preference.

From an infrastructure perspective, initiation with Terraform makes sense if the goal is to create infrastructure resources and if you use an IaC approach to gain a representation of configuration and images to be deployed using Ansible Automation. These resources can then be delivered to the application team for further action.

Example: A cloud operations team creates a selection of virtual machines (VMs) at the request of the application team. The VMs may have standard packages installed, but they do not have a business application identity.

From an application or broader system perspective, initiation with Ansible Automation makes sense. The end-to-end definition of the automation required for the application stack includes a step to provision infrastructure, and in this case, Ansible Automation would call Terraform for provisioning activities before continuing with its workflow.

Example: An application team kicks off a build pipeline to create, configure, test, and promote to production a new version of an application.

Ansible Automation users may want to use HashiCorp Packer to prebuild machine images and containers to reduce provisioning times. This approach can be used with Ansible Automation local and remote provisioners. Users of HashiCorp Vault can also integrate with Ansible Automation using the hashi_vault plugin to securely fetch secrets and deliver them to applications.

TERRAFORM INVOKING ANSIBLE AUTOMATION

The mechanism for invoking external tools in Terraform is called a provisioner, and it includes several built-in options. The two applicable to Ansible Automation are local-exec and remote-exec.

The local-exec provisioner, which allows any locally installed tool to be executed, can be used to invoke Ansible Automation locally on the same machine as Terraform. This provisioner is used when Ansible Automation is configuring a machine over the network.

The remote-exec provisioner, which allows Terraform to execute commands against a remote resource, can be used to invoke Ansible Playbooks on remote resources after creation. This provisioner is used when Ansible Automation is running on the machine being configured.

To invoke locally using local-exec requires Terraform to invoke ansible-playbook to start running a playbook that can be specified using the command argument. Below is an example of a Terraform configuration that uses local-exec to provision a VM.

```
hcl

resource “aws_instance” “web” {

  # ...

  provisioner “local-exec” {

    command = “ansible-playbook -u ubuntu -i ‘${aws_instance.web.public_
dns},’ main.yml”

  }

}
```

A Terraform plan can be generated using the following command:

```
$ terraform plan
```

After the plan is reviewed, a Terraform apply can be run using the following command:

```
$ terraform apply
```

Figure 1 details how this process works.

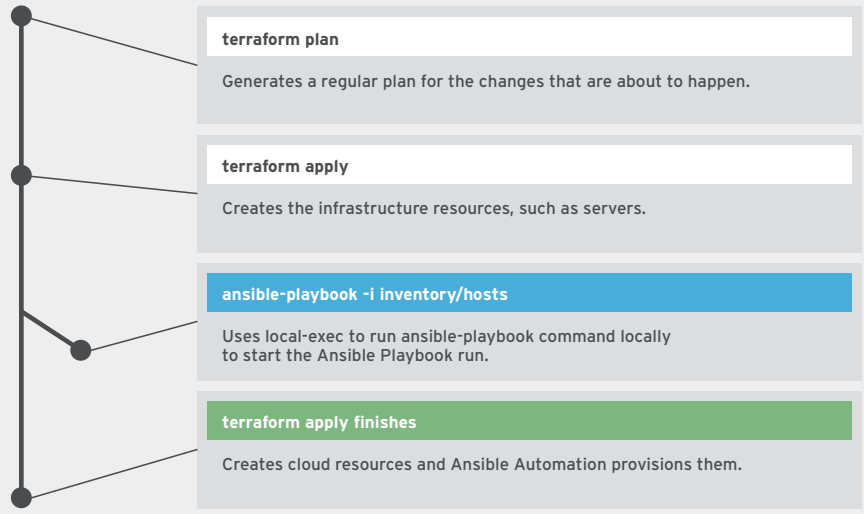


Figure 1. Terraform using local-exec

This approach works well when Terraform is responsible for creating resources and Ansible Automation is installed on the same machine with access to the newly created resources—i.e., Terraform and Ansible Automation are both installed on the same control node used to execute resource creation commands, like a bastion host or management proxy.

A potential challenge with the local-exec provisioner is that it expects Ansible Automation to be installed and executed locally on the machine running Terraform. For users who want to run Ansible Automation on the newly created resource, ensure that Ansible Automation is installed on the machine image and then the remote-exec provisioner can be used to execute Ansible Playbooks on the newly created remote resource.

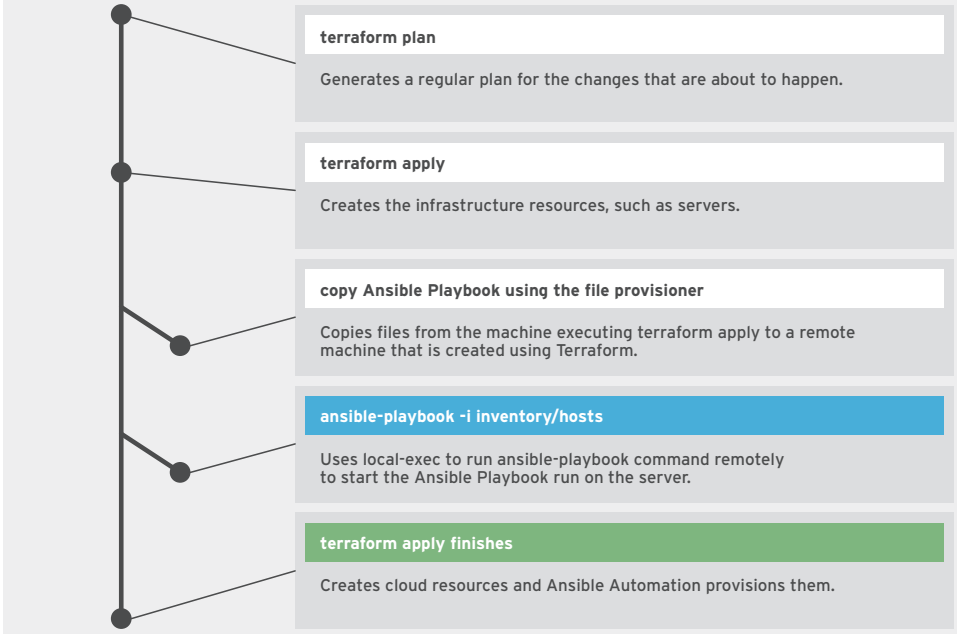


Figure 2. Terraform using remote-exec

This approach works well when Terraform is responsible for creating the resources, and Ansible Automation is installed and invoked on the remote machine.

ANSIBLE AUTOMATION INVOKING TERRAFORM

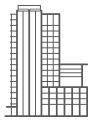
For users who want to use Ansible Automation to invoke Terraform, the Ansible module for Terraform can be used to run terraform plan, terraform apply, and terraform destroy commands right from the Ansible Playbook. This approach requires Terraform to be installed and available in the system path of the Ansible control node. Ansible Automation uses the locally installed Terraform binary to execute commands. Below is an example of an Ansible Playbook that runs terraform plan, terraform apply, and terraform destroy commands.

```
---
- name: main
  hosts: all
  gather_facts: false
  connection: local
  tasks:
  - name: plan
    terraform:
      project_path: 'terraform/'
      plan_file: "{{playbook_dir}}/tfplan"
      lock: true
      state: planned

  - name: apply
    terraform:
      project_path: 'terraform/'
      lock: true
      state: present

  - name: destroy
    terraform:
      project_path: 'terraform/'
      lock: true
      state: absent
```

The project_path variable defines where the Terraform configuration is stored. To learn more about the Ansible Terraform module, visit https://docs.ansible.com/ansible/latest/modules/terraform_module.html.

SUMMARY

HashiCorp Terraform and Red Hat Ansible Automation are solutions that support an infrastructure as code initiative. Both products offer strengths and areas of focus and support integrations with third-party solutions. As technology partners, HashiCorp and Red Hat are committed to supporting scenarios where Terraform and Ansible Automation can complement each other. This commitment allows joint customers to automate end-to-end infrastructure and application life-cycle management—from provisioning and retirement of computing resources to application deployment and configuration of operating systems, middleware, and applications.



ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
f14774_1118

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com