# Provision Infrastructure with Cloud-Init

5 MIN PRODUCTS USED Terraform

This tutorial also appears in: Associate Tutorials.

When you create a generic compute resource in Terraform, your virtual machine (VM) may not have much capability because it is a "fresh" install and needs to be provisioned with the software you want to use. Manually installing the necessary software and its respective dependencies on each VM is time consuming and difficult to maintain at scale.

`cloud-init` is a standard configuration support tool available on most Linux distributions and all major cloud providers. `cloud-init` allows you to pass a shell script to your instance that installs or configures the machine to your specifications.

In this tutorial, you will create a Terraform instance with the `user_data` to deploy a Go web app and SSH key to the newly created device, allowing you to SSH into the machine without a password and start the app with that user.

## Prerequisites

For this tutorial, you will need the following:

- Terraform
- An AWS account

Clone the example repository here.

```
$ git clone -b cloudinit https://github.com/hashicorp/le
```

Copy

Change into your cloned repo directory.

```
$ cd learn-terraform-provisioning
```

Copy

## Create a local SSH key

For this tutorial, create a local SSH key to pair with the new `terraform` user you create on this instance.

Mac or Linux command-line     Windows with PuTTY

Generate a new SSH key in your terminal called `tf-cloud-init`. The argument provided with the `-f` flag creates the key in the current directory and creates two files called `tf-cloud-init` and `tf-cloud-init.pub`. Change the placeholder email address to your email address.

```
$ ssh-keygen -t rsa -C "your_email@example.com" -f ./tf-
```

Copy

When prompted, press enter to leave the passphrase blank on this key.

## Add your public SSH key to your cloud-init script

Open the `scripts/add-ssh-web-app.yaml` file and paste the contents of `tf-cloud-init.pub` into the user data `ssh_authorized_keys` section. You will pass this cloud-init script to your instance resource's `user_data` attribute.

```
##...
users:
  - default
  - name: terraform
    gecos: terraform
    primary_group: hashicorp
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: users, admin
    ssh_import_id:
    lock_passwd: false
    ssh_authorized_keys:
      -  # Paste your created SSH key here
##...
```

For more information on creating a `cloud-init` script, refer to the cloud-init documentation.

## Add the cloud-init script to the Terraform configuration

Open the `main.tf` file. Notice how the `template_file.user_data` data block retrieves the contents of the `add-ssh-web-app.yaml` file. Then, it is passed into `aws_instance.web` as a `user_data` value to be initialized when the instance is created.

```
data "template_file" "user_data" {
  template = file("../scripts/add-ssh-web-app.yaml")
}

resource "aws_instance" "web" {
  ami                         = data.aws_ami.ubuntu.id
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.subnet_public.id
  vpc_security_group_ids      = [aws_security_group.sg_22_80.id]
  associate_public_ip_address = true
  user_data                   = data.template_file.user_data.render
```

```
    tags = {
      Name = "Learn-CloudInit"
    }
  }
```

Create a new file called `terraform.tfvars` then add your AWS region variable definition.

```
region = "us-east-1"
```
Copy

Save this file and then initialize your configuration.

```
$ terraform init
```
Copy

Apply your configuration. Enter `yes` when prompted to create your instance.

```
$ terraform apply
```
Copy

When the apply run completes, your terminal will display your instance's IP address.

You have successfully provisioned your AWS instance with `cloud-init` . This instance should already be configured with the SSH key, allowing you to connect to it. Your instance should also contain the GoLang demo app.

In the next section, you will SSH into this instance with your local key and start the demo app.

## Verify your instance

Connect to your instance via SSH by piping the `aws_instance.web.public_ip` resource attribute to the `terraform` `console` command.

```
$ ssh terraform@$(terraform output -raw public_ip) -i ..
```

Copy

Now you have SSH access to your AWS instances without creating SSH keys in AWS. This is useful if your organization maintains keypairs outside of AWS.

Navigate to the Go directory.

```
$ cd go/src/github.com/hashicorp/learn-go-webapp-demo
```

Copy

Launch the demo webapp.

```
$ go run webapp.go
```

Copy

In your web browser, navigate to the IP address of your instance and port 8080 to see the app you deployed.

## Destroy your image

Avoid unnecessary charges in your AWS account by destroying your instance in Terraform.

```
$ terraform destroy
```

Copy

Type `yes` when you are prompted in your terminal to delete your infrastructure.

## Next Steps

In this tutorial, you deployed a webapp and configured an instance with `cloud-init`.

- To learn about creating images with Packer for Terraform deployments, check out the Provision Infrastructure with Packer tutorial.
- For more information about creating templates with `cloud-init`, visit the data provider documentation.

Was this tutorial helpful?

Yes No

BACK TO COLLECTION