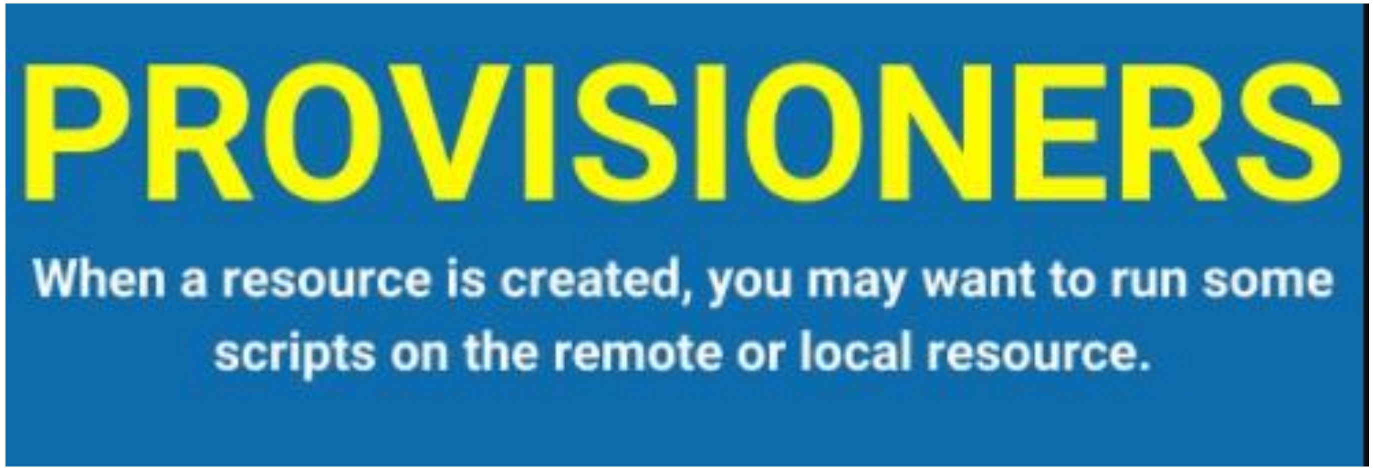
# Terraform Provisioner

Configuring Resources After Creation

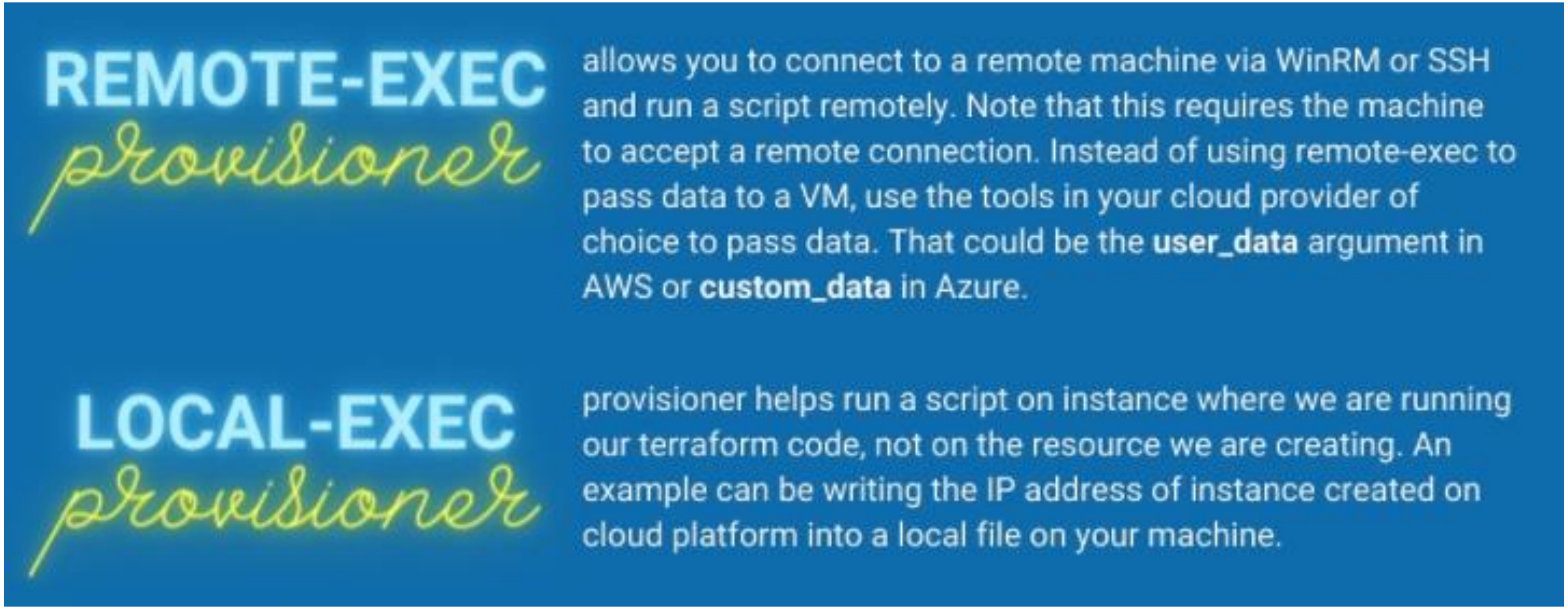
Provisioners can be used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service.


PROVISIONERS

When a resource is created, you may want to run some scripts on the remote or local resource.

## List of provisioners

- Chef
- File
- habitat
- puppet
- remote-exec
- salt-masterless
- **local-exec:** Run any script on resource create or delete.

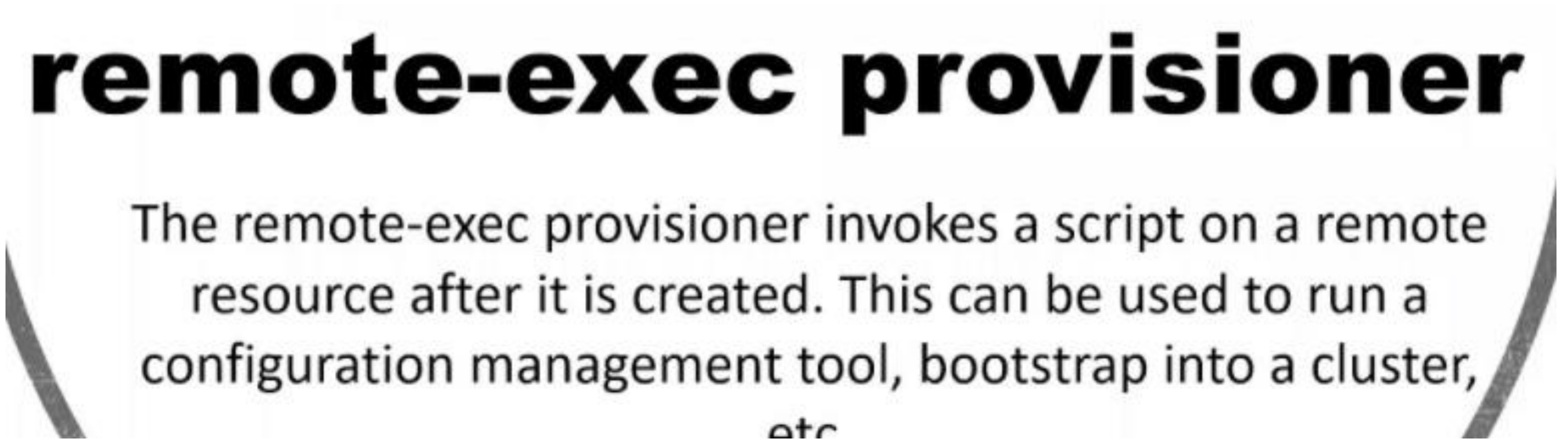


REMOTE-EXEC provisioner

allows you to connect to a remote machine via WinRM or SSH and run a script remotely. Note that this requires the machine to accept a remote connection. Instead of using remote-exec to pass data to a VM, use the tools in your cloud provider of choice to pass data. That could be the **user_data** argument in AWS or **custom_data** in Azure.

LOCAL-EXEC provisioner

provisioner helps run a script on instance where we are running our terraform code, not on the resource we are creating. An example can be writing the IP address of instance created on cloud platform into a local file on your machine.


## remote-exec provisioner

The remote-exec provisioner invokes a script on a remote resource after it is created. This can be used to run a configuration management tool, bootstrap into a cluster, etc

## file provisioner

The file provisioner is used to copy files or directories from the machine executing Terraform to the newly created resource. The file provisioner supports both ssh and winrm type connections.

## chef provisioner

The chef provisioner installs, configures and runs the Chef Client on a remote resource. The chef provisioner supports both ssh and winrm type connections.

## The habitat provisioner

The habitat provisioner installs the Habitat supervisor and loads configured services. This provisioner only supports Linux targets using the ssh connection type at this time.

**The puppet provisioner**

The puppet provisioner installs, configures and runs the Puppet agent on a remote resource. The puppet provisioner supports both ssh and winrm type connections.

The salt-masterless

The salt-masterless Terraform provisioner provisions machines built by Terraform using Salt states, without connecting to a Salt master. The salt-masterless provisioner supports ssh connections.

## Terraform Provisioners

- Run code locally or remotely on resource creation
- Resource is tainted if provisioning failed. (next apply it will be re-created
- You can run code on deletion. If it fails - resources are not removed

SALTSTACK

masterless

null_resource +

- file
- local-exec
- remote-exec

## Provisioner Example

```
provisioner “local-exec” {
  command = “local command here”
}

provisioner “remote-exec” {
  scripts = “[list, of, local, scripts]”
}
```

## Provisioner Example

```
connection {
  user = “username”
  private_key = “privatekey”
}

provisioner “file” {
  content = <<EOF
your file content here
EOF
  destination = “/path/to/file.txt”
}
```

```
# Copies the file as the root user using SSH
provisioner "file" {
  source      = "conf/myapp.conf"
  destination = "/etc/myapp.conf"

  connection {
    type     = "ssh"
    user     = "root"
    password = "${var.root_password}"
    host     = "${var.host}"
  }
}
```

```
# Copies the file as the Administrator user using WinRM
provisioner "file" {
  source      = "conf/myapp.conf"
  destination = "C:/App/myapp.conf"

  connection {
    type     = "winrm"
    user     = "Administrator"
    password = "${var.admin_password}"
    host     = "${var.host}"
  }
}
```

```
resource "null_resource" "example1" {
  provisioner "local-exec" {
    command = "open WFH, '>completed.txt' and print WFH scalar localtime"
    interpreter = ["perl", "-e"]
  }
}
```

```
resource "null_resource" "example2" {
  provisioner "local-exec" {
    command = "Get-Date > completed.txt"
    interpreter = ["PowerShell", "-Command"]
  }
}
```

```
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "echo $FOO $BAR $BAZ >> env_vars.txt"

    environment = {
      FOO = "bar"
      BAR = 1
      BAZ = "true"
    }
  }
}
```

## run code after resources is created using using remote-exec provisioners

Using null_resource

https://www.devopsschool.com/blog/how-to-run-provisioners-code-after-resources-is-created-in-terraform/

https://www.devopsschool.com/blog/terraform-example-code-for-create-azure-linux-windows-vm-with-file-remote-exec-local-exec-provisioner/

https://www.devopsschool.com/blog/terraform-example-code-for-create-azure-linux-windows-vm-with-file-remote-exec-local-exec-provisioner/

https://www.devopsschool.com/blog/terrafrom-example-code-for-aws-ec2-instance-and-remote-exec-provisioner/

https://www.devopsschool.com/blog/terraform-provisioners-tutorials-and-complete-guide/

https://www.devopsschool.com/blog/terraform-example-program-for-aws_security_group-aws_instance-and-provisioner/

https://www.devopsschool.com/blog/how-to-run-provisioners-code-after-resources-is-created-in-terraform/

https://www.devopsschool.com/blog/understanding-local-exec-provisioner-in-terraform/