



VMware Cloud on AWS

Using Terraform with multiple providers to deploy and configure VMware Cloud on AWS



[Gilles Chekroun](#)

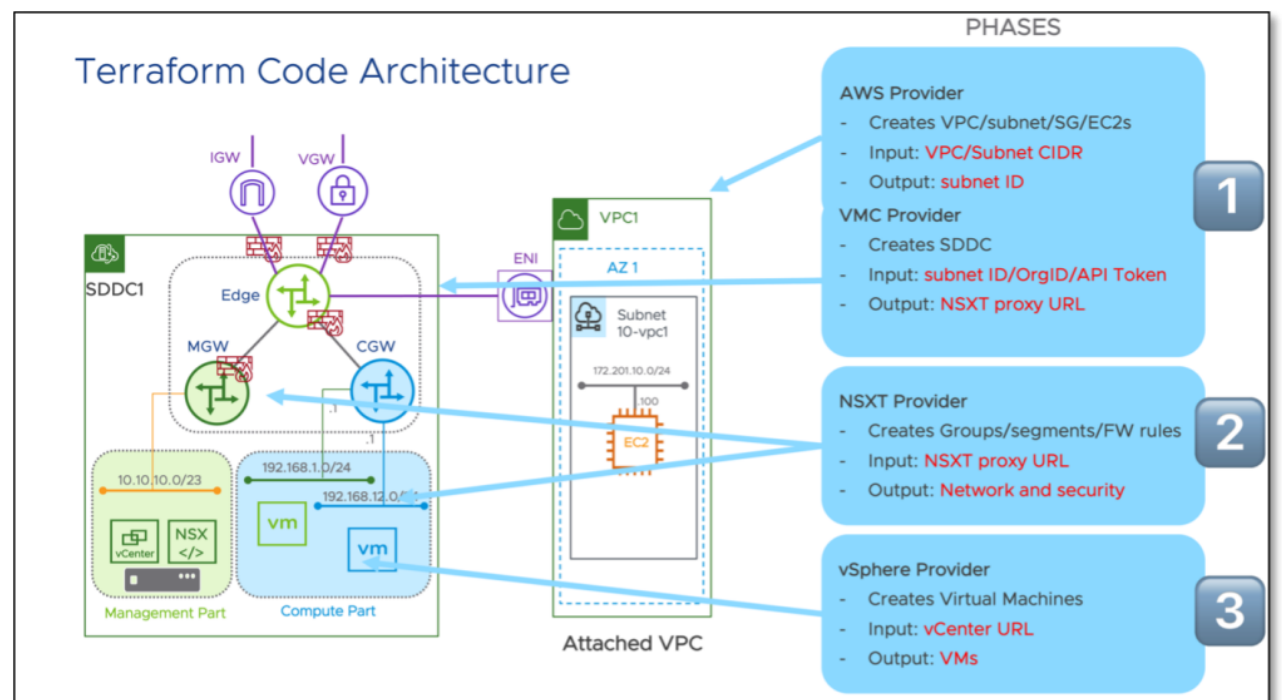
June 30, 2022

Share on:

Terraform is fast emerging as a popular Infrastructure as Code (IaC) tool that lets you define cloud resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle.

This blog post series is intended to be an update to the original [VMware Cloud on AWS with Terraform blog post](#) with the goal of providing an end-to-end guide to standing up a new VMware Cloud on AWS deployment from scratch with Terraform.

With the recent development of VMware Terraform providers for [NSX-T](#) and [VMware Cloud](#) on AWS, we have now the possibility to create a full Infrastructure as Code (IaC) automation and deployment of VMware Cloud on AWS including AWS, VMC, NSX-T and vSphere.



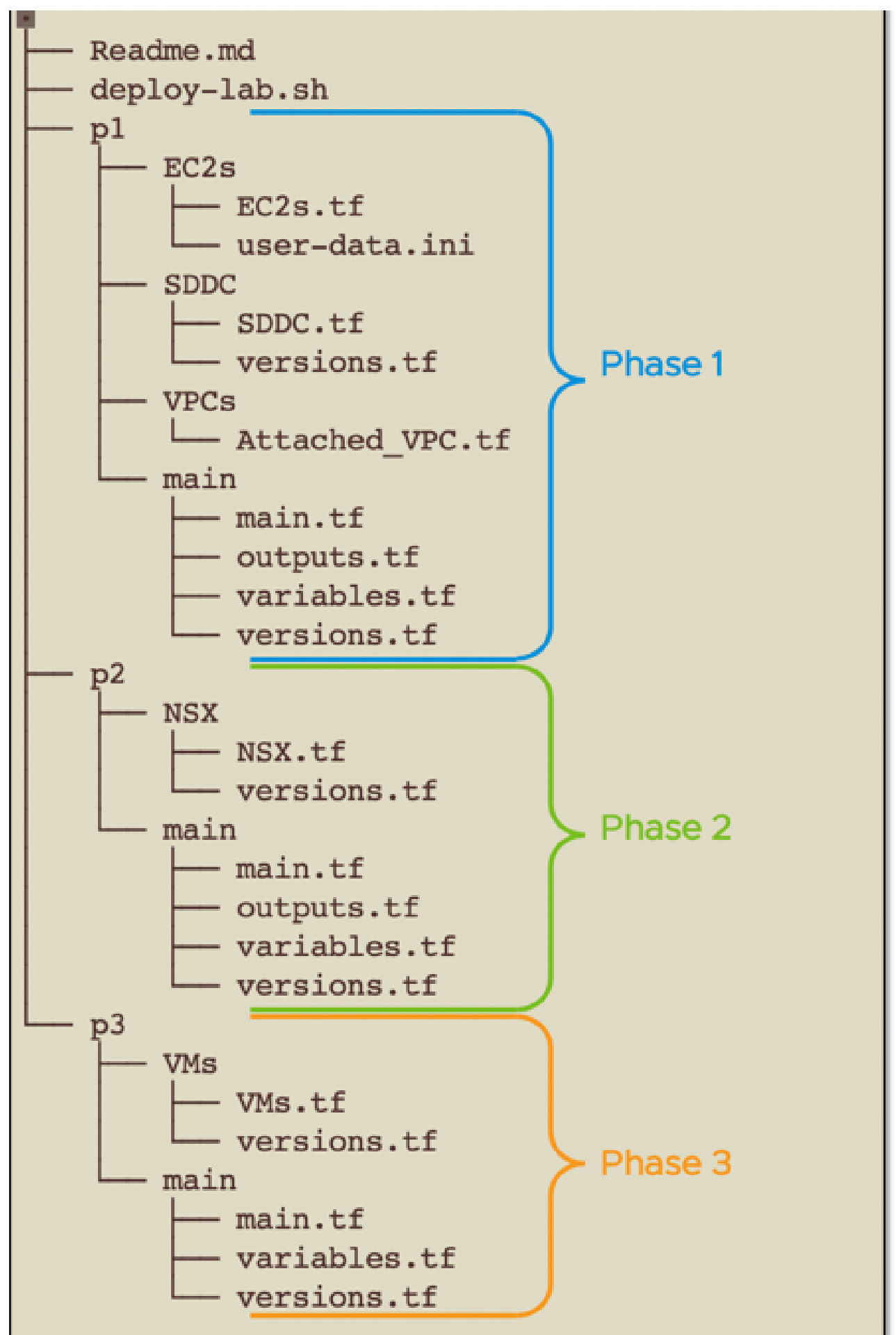
Terraform Code Architecture

This code is architected in three disparate phases as shown in the architecture diagram. The output of one phase will be used as input for another.

- Phase 1: Set up AWS VPC and deploy a VMC SDDC
- Phase 2: Configure network including groups, segments, and firewall rules
- Phase 3: Deploy virtual machines within the VMC SDDC

File structure

Below is a description of the files in the multiple phases of deployment. These files can be downloaded [here](#). The code is written using the [Terraform modules](#) construct.



Phase 1

The first phase will use [AWS provider](#) combined with [VMC provider](#). In this phase we start with the creation of the Connected VPC and various elements inside that VPC like:

- 2 subnets in 2 Availability Zones
- An internet Gateway
- A default route to internet
- Subnet route table association
- A Security Group for ping, SSH, and other ports
- An S3 Gateway endpoint
- 2 EC2s (one Linux and one Windows machine)

Phase 1 Variables

This section explains the variables present in the sample variables file (/p1/main/variables.tf).

For AWS, we will need AWS account, AWS region, EC2 keypair (see [AWS documentation](#) for details on EC2 keypair).

For VMC we will need API token (see Authentication and Authorization section in [the VMC API reference](#) for instructions to generate an API token) and Org ID.

```
variable "AWS_account"      {}
variable "vmc_token"         {}
variable "my_org_id"         {}

variable "AWS_region"        { default = "us-west-2" }
variable "key_pair"          { default = "my-oregon-key" }
```

We will also define the SDDC management subnet and default NSX segment together with AWS Connected VPC CIDR and subnets.

```
/*=====
Subnets IP ranges
=====*/
variable "My_subnets" {
  default = {

    SDDC_Mngt      = "10.10.0.0/23"
    SDDC_default   = "192.168.1.0/24"

    VPC_Attached   = "172.200.0.0/16"
    Subnet10-Att_vpc = "172.200.10.0/24"
    Subnet20-Att_vpc = "172.200.20.0/24"

  }
}
```

We will point to AWS AMI IDs in our region for Linux and Windows

```
/*=====
VM AMIs
=====*/
variable "VM_AMI"          { default = "ami-0528a5175983e7f28" } # Amazon Linux 2 AMI (HVM),
SSD Volume Type - Oregon
variable "Win_AMI"         { default = "ami-0f18e475ccdc50e07" } # Microsoft Windows Server
2019 Base - Oregon
```

Phase 1 “main.tf”

This section explains the content from the sample main.tf file (p1/main/variables.tf). Here we will use 2 providers (AWS and VMC) and set the backend for the state file (local).

```
provider "aws" {
  region = var.AWS_region
}
```

```
provider "vmc" {
  refresh_token = var.vmc_token
  org_id        = var.my_org_id
}

terraform {
  backend "local" {
    path = "../../phase1.tfstate"
  }
}
```

Next, we use separate modules for VPC, EC2s, and SDDC creation. The source of the modules are subdirectories in phase1 (p1) folder.

```
/*=====
Create AWS VPC
The VPC and subnets CIDR are set in "variables.tf" file
=====*/
module "VPCs" {
  source = "../VPCs"

  VPC_Att_cidr      = var.My_subnets["VPC_Attached"]
  Subnet10-Att_vpc  = var.My_subnets["Subnet10-Att_vpc"]
  Subnet20-Att_vpc  = var.My_subnets["Subnet20-Att_vpc"]

  region            = var.AWS_region
}
```

```
/*=====
Create EC2s
=====*/
module "EC2s" {
  source = "../EC2s"

  VM-AMI      = var.VM_AMI
  WIN-AMI      = var.Win_AMI
  key_pair     = var.key_pair

  // Attached VPC
  Subnet10-Att_vpc      = module.VPCs.Subnet10-Att_vpc_id
  Subnet10-Att_vpc-base = var.My_subnets["Subnet10-Att_vpc"]
  SG-Att_vpc           = module.VPCs.SG-Att_vpc_id
}
```

```
Create SDDC
=====*/
module "SDDC" {
  source = "../SDDC"

  my_org_id      = var.my_org_id      # ORG ID
  SDDC_Mngt      = var.My_subnets["SDDC_Mngt"] # Management IP range
  SDDC_default    = var.My_subnets["SDDC_default"] # Default SDDC Segment
  Att_vpc_subnet_id = module.VPCs.Subnet10-Att_vpc_id # VPC attached subnet
  region         = var.AWS_region      # AWS region
  AWS_account     = var.AWS_account     # Your AWS account
}
```

We will use environment variables set by the shell where Terraform runs with TF_VAR_prefix as noted in the provided shell script (/deploy-lab.sh). To deploy the complete environment, execute this shell script that will code all the secret parameters like:

- VMC Org ID
- VMC API Token
- AWS Account
- AWS Access key
- AWS Secret key
- AWS Token (if used)

Run the shell script by executing the following command.

source deploy-lab.sh

```
$source deploy-lab.sh

=====
      VMC deployment
=====
===== Set Credentials =====

Enter your ORG ID (long format) [default=3dlc2abe-.....]:
.....Exporting 3dlc2abe-.....

Enter your VMC API token [default=qpQ3vZXB1.....]:
.....Exporting qpQ3vZXB1.....

Enter your AWS Account [default=063.....]:
.....Exporting 063.....

Enter your AWS Access Key [default=ASIA.....]:
.....Exporting ASIA.....

Enter your AWS Secret Key [default=HNLs+.....]:
.....Exporting HNLs+.....

Enter your AWS Session Token [default=IQoJb3JpZ.....]:
.....Exporting IQoJb3JpZ.....

Press enter to continue (^C to stop)...
```

And finally, run “terraform init” and “terraform apply”.

After the credentials, phase 1 is deployed and Terraform providers are initialized.

```
===== PHASE 1: Creating SDDC =====

Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of terraform-providers/vmc from the dependency lock file
- Using previously-installed hashicorp/aws v4.9.0
- Using previously-installed terraform-providers/vmc v1.9.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
+ create
```



```
terraform will perform the following actions: (please remove,
# module.EC2s.aws_instance.VM1 will be created
+ resource "aws_instance" "VM1"
# module.EC2s.aws_instance.Windows will be created
+ resource "aws_instance" "Windows"
# module.EC2s.aws_network_interface.VM1-Eth0 will be created
+ resource "aws_network_interface" "VM1-Eth0"
# module.EC2s.aws_network_interface.WIN-Eth0 will be created
+ resource "aws_network_interface" "WIN-Eth0"
# module.SDDC.vmc_sddc.Terraform_SDDC1 will be created
+ resource "vmc_sddc" "Terraform_SDDC1"
# module.VPCs.aws_default_route_table.Att_vpc-RT will be created
+ resource "aws_default_route_table" "Att_vpc-RT"
# module.VPCs.aws_default_security_group.default will be created
+ resource "aws_default_security_group" "default"
# module.VPCs.aws_internet_gateway.Att_vpc-IGW will be created
+ resource "aws_internet_gateway" "Att_vpc-IGW"
# module.VPCs.aws_route_table_association.Att_vpc_10 will be created
+ resource "aws_route_table_association" "Att_vpc_10"
# module.VPCs.aws_security_group.SG-Att_vpc will be created
+ resource "aws_security_group" "SG-Att_vpc"
# module.VPCs.aws_subnet.Subnet10-Att_vpc will be created
+ resource "aws_subnet" "Subnet10-Att_vpc"
# module.VPCs.aws_subnet.Subnet20-Att_vpc will be created
+ resource "aws_subnet" "Subnet20-Att_vpc"
# module.VPCs.aws_vpc.Att_vpc will be created
+ resource "aws_vpc" "Att_vpc"
# module.VPCs.aws_vpc_endpoint.s3 will be created
+ resource "aws_vpc_endpoint" "s3"
```

Plan: 14 to add, 0 to change, 0 to destroy.

```
Changes to Outputs:
+ Windows_IP           = (known after apply)
+ cloud_password       = (sensitive value)
+ cloud_username       = (known after apply)
+ nsxt_cloudadmin      = (known after apply)
+ nsxt_cloudadmin_password = (sensitive value)
+ nsxt_private_IP     = (known after apply)
+ proxy_url           = (known after apply)
+ sddc_subnet         = (known after apply)
+ vc_public_IP        = (known after apply)
+ vc_url              = (known after apply)
```

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

Enter a value:

At this stage, 14 resources are ready. Enter YES to continue.

Next- Check out the steps for phase 2 of the deployment in [Part 2](#) of this blog series.




[Gilles Chekroun](#)
Gilles Chekroun is Lead VMware Cloud on AWS Solutions Architect in the European team. He joined VMware in 2015 after spending 20 years at Cisco in the Data Centre Network...

Related Articles




[VMware Cloud on AWS](#)
[VMware Cloud on AWS Free Trial is available now! Start your hybrid cloud journey today!](#)

 [Nancy Cheng](#)
July 11, 2023




[VMware Cloud on AWS](#)
[VMware Lab Platform with VMware Cloud on AWS- Delivering on-demand labs anytime and anywhere](#)

 [noeldnerb](#)
July 6, 2023



[VMware Cloud on AWS](#)
[Introducing the New and Improved Networking Experience for VMware Cloud Console](#)

 [Nancy Cheng](#)
June 9, 2023



[VMware Cloud on AWS](#)

Now Live – vCenter Federation for VMware Cloud on AWS



[Aakash Chandhoke](#)
June 8, 2023



[VMware Cloud on AWS](#)

Custom Managed Elastic DRS Policy: Improving Cost-Efficiency and Performance in VMware Cloud on AWS



[Amitha Shetty](#)
May 22, 2023



Company

[About Us](#)

[Executive Leadership](#)

[News & Stories](#)

[Investor Relations](#)

[Customer Stories](#)

[Diversity, Equity & Inclusion](#)

[Environment, Social & Governance](#)

[Careers](#)

[Blogs](#)

[Communities](#)

[Acquisitions](#)

[Office Locations](#)

[VMware Cloud Trust Center](#)

[COVID-19 Resources](#)

Support

[VMware Customer Connect](#)

[Support Policies](#)

[Product Documentation](#)

[Compatibility Guide](#)

[Terms & Conditions](#)

[California Transparency Act Statement](#)



[Twitter](#)



[YouTube](#)



[Facebook](#)



[LinkedIn](#)



[Contact Sales](#)

© 2023 VMware, Inc.

[Terms of Use](#)

[Your California Privacy Rights](#)

[Privacy](#)

[Accessibility](#)

[Trademarks](#)

[Glossary](#)

[Help](#)

[Feedback](#)