Security Automation with Terraform

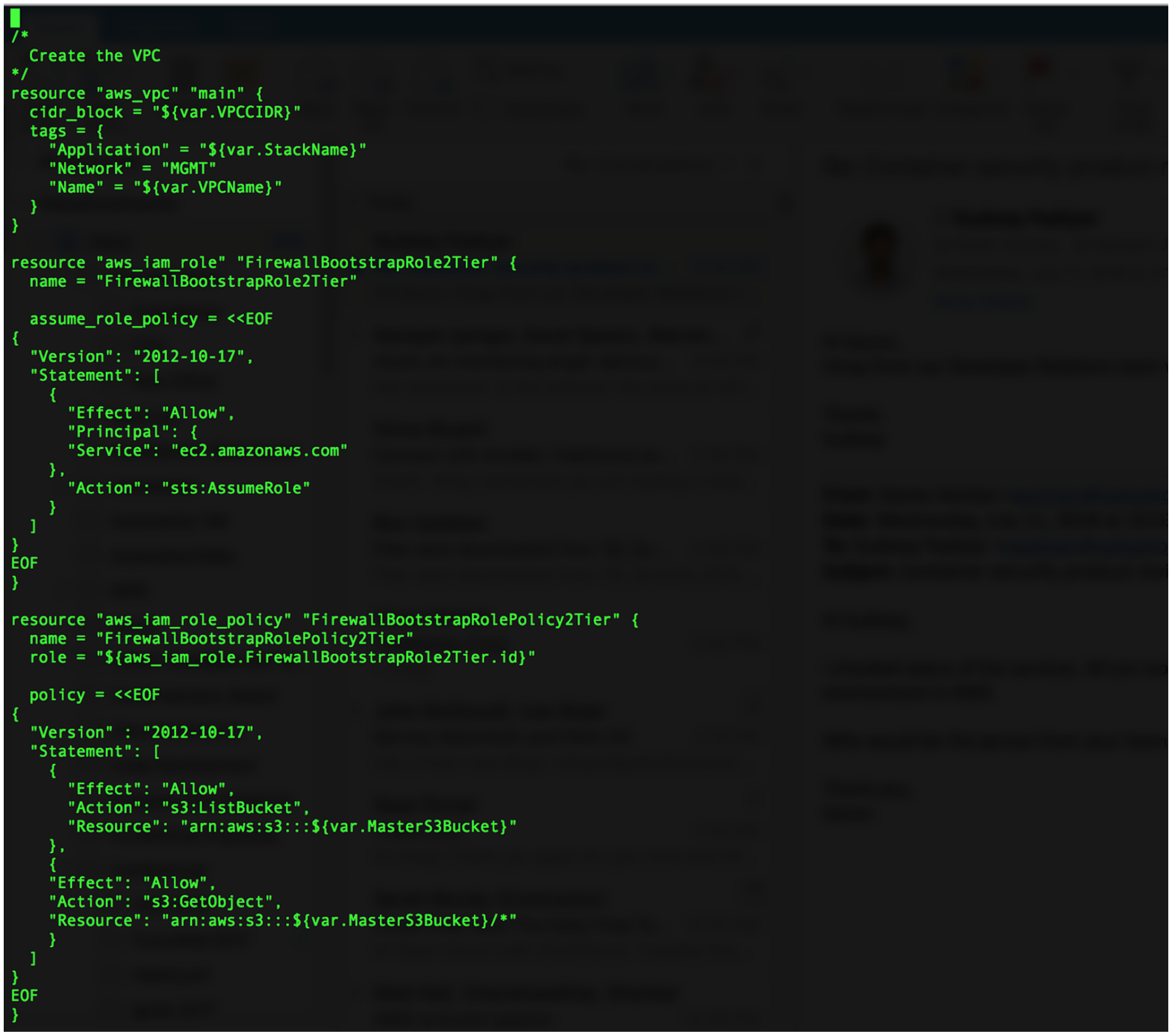
Author: Vinay Venkataraghavan
Date: 12/12/2019

Terraform Technical Overview

What is Terraform?

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

Terraform allows users to use a declarative syntax to interactive with backend / target systems.



```
/*
  Create the VPC
*/
resource "aws_vpc" "main" {
  cidr_block = "${var.VPCCIDR}"
  tags = {
    "Application" = "${var.StackName}"
    "Network" = "MGMT"
    "Name" = "${var.VPCName}"
  }
}

resource "aws_iam_role" "FirewallBootstrapRole2Tier" {
  name = "FirewallBootstrapRole2Tier"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
      "Service": "ec2.amazonaws.com"
    },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
}

resource "aws_iam_role_policy" "FirewallBootstrapRolePolicy2Tier" {
  name = "FirewallBootstrapRolePolicy2Tier"
  role = "${aws_iam_role.FirewallBootstrapRole2Tier.id}"

  policy = <<EOF
{
  "Version" : "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::${var.MasterS3Bucket}"
    },
    {
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::${var.MasterS3Bucket}/*"
    }
  ]
}
EOF
}
```

Terminology and Concepts

Template:

Template refers to the a file with a ".tf" suffix. This contents of this file describes the resources that will be created against the backend platform.

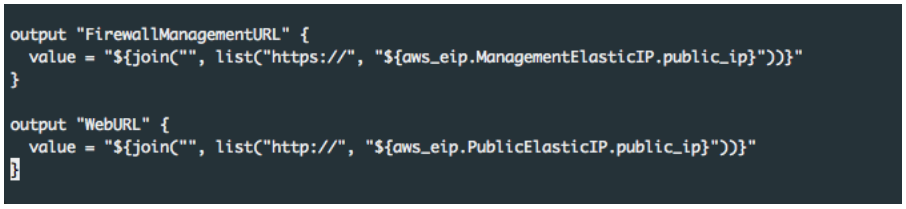
Plan:

A plan represents a representation of the artifacts that will be created or modified.

Output Variables:

It is possible to “return” certain values upon execution of a terraform template file. These can be specified as output variables and the data is outputted once the “apply” command is executed.

For example:



```
output "FirewallManagementURL" {
  value = "${join("", list("https://", "${aws_eip.ManagementElasticIP.public_ip}"))}"
}

output "WebURL" {
  value = "${join("", list("http://", "${aws_eip.PublicElasticIP.public_ip}"))}"
}
```

Modules:

Terraform modules are self contained packages of Terraform configurations that are managed as a group. They are used to create re-useable components, improve code organization and treat pieces of the infrastructure as a black box.

Provider:

The provider is a concept in terraform which specifies the plugin module which will be invoked by Terraform to interact with the target platform as specified by the provider.

For example:

```
provider "aws" {
      access_key = "<access_key>"
      secret_key = "<secret_key>"
      region     = "${var.aws_region}"
    }
```

The specification shown above references the “aws” provider. In this case the “aws” provider requires the specification of the users’ AWS credentials and the region to be able to interact with the platform.

Resources:

A resource is one of the fundamental building blocks in Terraform. It is used to define the entities / artifacts that will be created on the target environment.

For example:

The resource shown below will result in the creation of a VPC on the AWS cloud.
Please also note the name of the resource: `“aws_vpc”`. The `“aws”` prefix indicates that the resource belongs to the `“aws”` provider.

```
/*
  Create the VPC
*/
resource "aws_vpc" "main" {
  cidr_block = "${var.VPCCIDR}"
  tags = {
    "Application" = "${var.StackName}"
    "Network" = "MGMT"
    "Name" = "${var.VPCName}"
  }
}
```

Input Variables:

Variables provide the ability for parametrization by specifying specific values. These variables are in turn used in different resource specifications. This allows for different executions of the terraform template or plan to produce infrastructure with different names / properties.

State File:

Name: terraform.tfstate

Upon the completion of a terraform template execution, a state file will be populated with a record of all the artifacts created.
This file is used by terraform across execution runs in order to determine changes between a template or plan against what is actually deployed. If a change is determined terraform will delete and replace, thus creating a new instance of the resource. If a new resource is detected, the new resource will be created accordingly.

Null resource provider

The “null_resource” is a special type of resource. It is not tied to any specific provider and enables users to run arbitrary commands and operations. An example scenario is to run shell scripts to execute custom operations.

Dependencies

Very often the creation of resources needs to depend upon the prior creation of other resources. This paradigm is supported in one of two ways:

- Using the “depends_on” keyword and explicitly specifying the dependency
- Or by using variable interpolation
    - For example:

```
resource "aws_network_interface" "FWManagementNetworkInterface" {
  subnet_id       = "${aws_subnet.NewPublicSubnet.id}"
  security_groups = ["${aws_security_group.sgWideOpen.id}"]
  source_dest_check = false
  private_ips_count = 1
  private_ips = ["10.0.0.99"]
}
```

In this particular example, the “aws_network_interface” resource named “FWManagementNetworkInterface” will not be created until the value specified in security_groups, which is ["${aws_security_group.sgWideOpen.id}"] is first created.

This is because the interface resource implicitly depends upon the security group resource.

Argument Reference

These describe the arguments that can be specified for a particular resource. Arguments can be mandatory or optional.

Attribute Reference

These describe values exported by the resource.

Terraform Commands

```
vvenkatara :
      ~/Documents/Work/terraform_work/pan_terraform_git/terraform-templates/aw
      s_two_tier >~/Downloads/terraform help
Usage: terraform [--version] [--help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console               Interactive console for Terraform interpolations
    destroy               Destroy Terraform-managed infrastructure
    env                   Workspace management
    fmt                   Rewrites config files to canonical format
    get                   Download and install modules for the configuration
    graph                 Create a visual graph of Terraform resources
    import                Import existing infrastructure into Terraform
    init                  Initialize a Terraform working directory
    output                Read an output from a state file
    plan                  Generate and show an execution plan
    providers             Prints a tree of the providers used in the
      configuration
    push                  Upload this Terraform module to Atlas to run
    refresh               Update local state file against real resources
    show                  Inspect Terraform state or plan
    taint                 Manually mark a resource for recreation
    untaint               Manually unmark a resource as tainted
    validate              Validates the Terraform files
    version               Prints the Terraform version
    workspace             Workspace management

All other commands:
    debug                 Debug output management (experimental)
    force-unlock          Manually unlock the terraform state
    state                 Advanced state management
```

Anatomy of a Terraform Template

The anatomy of a Terraform template involves the specification of at a minimum the following entities:

- Provider: to identify to terraform the name of the provider that will be utilized, as well as the basic configuration (typically authentication parameters) required to use the provider.
- Resources: the resources specify the name of a resource that will be created.

At a minimum a directory containing terraform template files will contain the following file structure:

Note: required files are shown in red. Filenames do not matter. They only need to end with ".tf" suffix

```
vvenkatara : terraform-templates/aws_two_tier >ls
README.md
provider.tf
aws_vars.tf
aws_creds.tf
pan.log
webserver_config_amzn_ami.sh
aws_two_tier.tf
check_fw.sh
terraform.tfstate
terraform.tfvars
```

Each of these representative files are described in the following sections.

1. A file to represent the provider to use:

```
terraform-templates/aws_two_tier >cat ../pan_guard_duty/aws_creds.tf
provider "aws" {
      access_key = "<access_key>"
      secret_key = "<secret_key>"
      region     = "${var.aws_region}"
    }
```

2. A file to specify the variables:

```
variable "aws_region" {}
variable "WebCIDR_Block" {}
variable "PublicCIDR_Block" {}
variable "MasterS3Bucket" {}
variable "VPCName" {}
variable "VPCCIDR" {}
variable "ServerKeyName" {}
variable "StackName" {}
```

All variables that will be used with the templates will need to defined with the “variable” prefix specification.

There is some special handling to note while dealing with variables:

- It is possible to specify default values to the variables.

```
variable "WebPublicIPName" {
  default = "WebPublicIP"
}
```

- During the execution of the terraform template, the user will be prompted for values for variable definitions which do not have default values.

- It is also possible to specify values for variables that have been defined with or without a default value. These values can be specified in a special file which will need to be named "terraform.tfvars".

This is advantageous for situations wherein there will not be operator intervention.

3. Input values specified in the terraform.tfvars file

As described above, values specified in this file will be picked up and assigned to variables definitions and will also override default values.

101 LAB

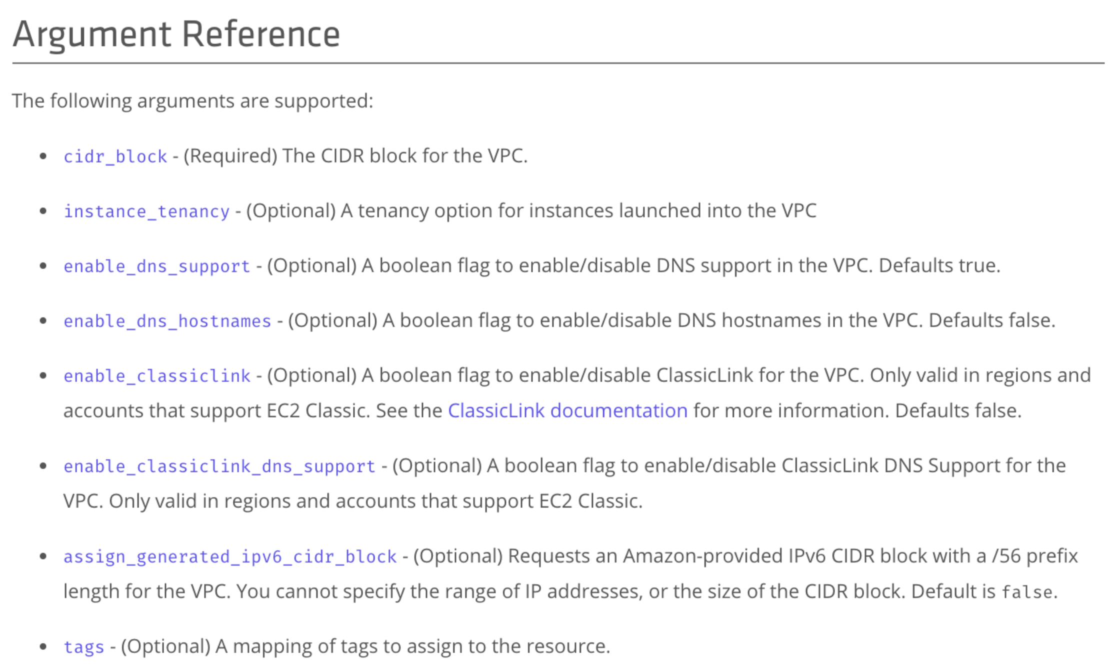
Objectives:

- Introduce the AWS Terraform Provider.
- Understand the details pertaining to a single AWS resource.
- Learn how to specify and utilize a single resource from the AWS provider.
- Develop from scratch numerous resource specified in a terraform template.
- Deploy the resources onto AWS.
- Clean up the resources

AWS Terraform Provider

The details pertaining to all the resources supported by the AWS Terraform provider can be found at: https://www.terraform.io/docs/providers/aws/index.html

Detailed Description of the "aws_vpc" resource.



Argument Reference

The following arguments are supported:

- `cidr_block` - (Required) The CIDR block for the VPC.
- `instance_tenancy` - (Optional) A tenancy option for instances launched into the VPC
- `enable_dns_support` - (Optional) A boolean flag to enable/disable DNS support in the VPC. Defaults true.
- `enable_dns_hostnames` - (Optional) A boolean flag to enable/disable DNS hostnames in the VPC. Defaults false.
- `enable_classiclink` - (Optional) A boolean flag to enable/disable ClassicLink for the VPC. Only valid in regions and accounts that support EC2 Classic. See the ClassicLink documentation for more information. Defaults false.
- `enable_classiclink_dns_support` - (Optional) A boolean flag to enable/disable ClassicLink DNS Support for the VPC. Only valid in regions and accounts that support EC2 Classic.
- `assign_generated_ipv6_cidr_block` - (Optional) Requests an Amazon-provided IPv6 CIDR block with a /56 prefix length for the VPC. You cannot specify the range of IP addresses, or the size of the CIDR block. Default is `false`.
- `tags` - (Optional) A mapping of tags to assign to the resource.

Example usage:

```
/*
  Create the VPC
*/
resource "aws_vpc" "main" {
  cidr_block = "${var.VPCCIDR}"
  tags = {
    "Application" = "${var.StackName}"
    "Network" = "MGMT"
    "Name" = "${var.VPCName}"
  }
}
```

Points to note:

- The construct “`${var.VPCCIDR}`” specifies that the value of the variable “`VPCCIDR`” should be assigned to the `cidr_block` argument of the “`aws_vpc`” resource.
- Alternatively, it is also possible to assign hard coded values for resource arguments. An example of this is the “Network” argument of the tags argument. The value is set to a fixed hard coded value of “MGMT”.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the VPC
- `cidr_block` - The CIDR block of the VPC
- `instance_tenancy` - Tenancy of instances spin up within VPC.
- `enable_dns_support` - Whether or not the VPC has DNS support
- `enable_dns_hostnames` - Whether or not the VPC has DNS hostname support
- `enable_classiclink` - Whether or not the VPC has Classiclink enabled
- `main_route_table_id` - The ID of the main route table associated with this VPC. Note that you can change a VPC's main route table by using an `aws_main_route_table_association`.
- `default_network_acl_id` - The ID of the network ACL created by default on VPC creation
- `default_security_group_id` - The ID of the security group created by default on VPC creation
- `default_route_table_id` - The ID of the route table created by default on VPC creation
- `ipv6_association_id` - The association ID for the IPv6 CIDR block.
- `ipv6_cidr_block` - The IPv6 CIDR block.

Lab Exercise

Suggested Values for the various resources:

cidr_block (aws_vpc.cidr_block)	`10.0.0.0/16`
cidr_block (aws_subnet)	`10.0.0.0/24`
aws_route_table	`vpc_id = "${aws_vpc.main.id}"`
aws_route_table	`vpc_id = "${aws_vpc.main.id}"`
aws_internet_gateway	`vpc_id = "${aws_vpc.main.id}" tags { Application = "${var.StackName}" Network = "MGMT" Name = "${join("-", list(var.StackName, "InternetGateway"))}" }`

1. Use the Terraform AWS Resource documentation and the concepts learned in the previous sections to create the following entities.

    a. An AWS VPC
    b. Two Subnets: a public subnet and a private subnet
    c. Two route tables
    d. Internet gateway

2. As listed above, you will be creating a total of 6 resources.
3. Please refer to the documentation for each of the resources.
4. For each of the resources, identify the mandatory attributes.
5. Make sure to specify definitions for each of the mandatory values.
6. Identify the optional values and determine if those might be useful for your deployment.
7. Create a "provider.tf" file which will consist of the "AWS" provider credentials and region.
8. Create a "aws_vars.tf" file for any variables you will need to use in the template deployment.
9. Create a "aws_deploy.tf" file which will specify and define all the resources and the associated arguments.
10. Create a "terraform.tfvars" file optionally to hold values for all the variables which do not have default values.
11. Run the terraform template to deploy the resources.

```
Hint: The command "terraform apply" can be used to deploy the resources.
```

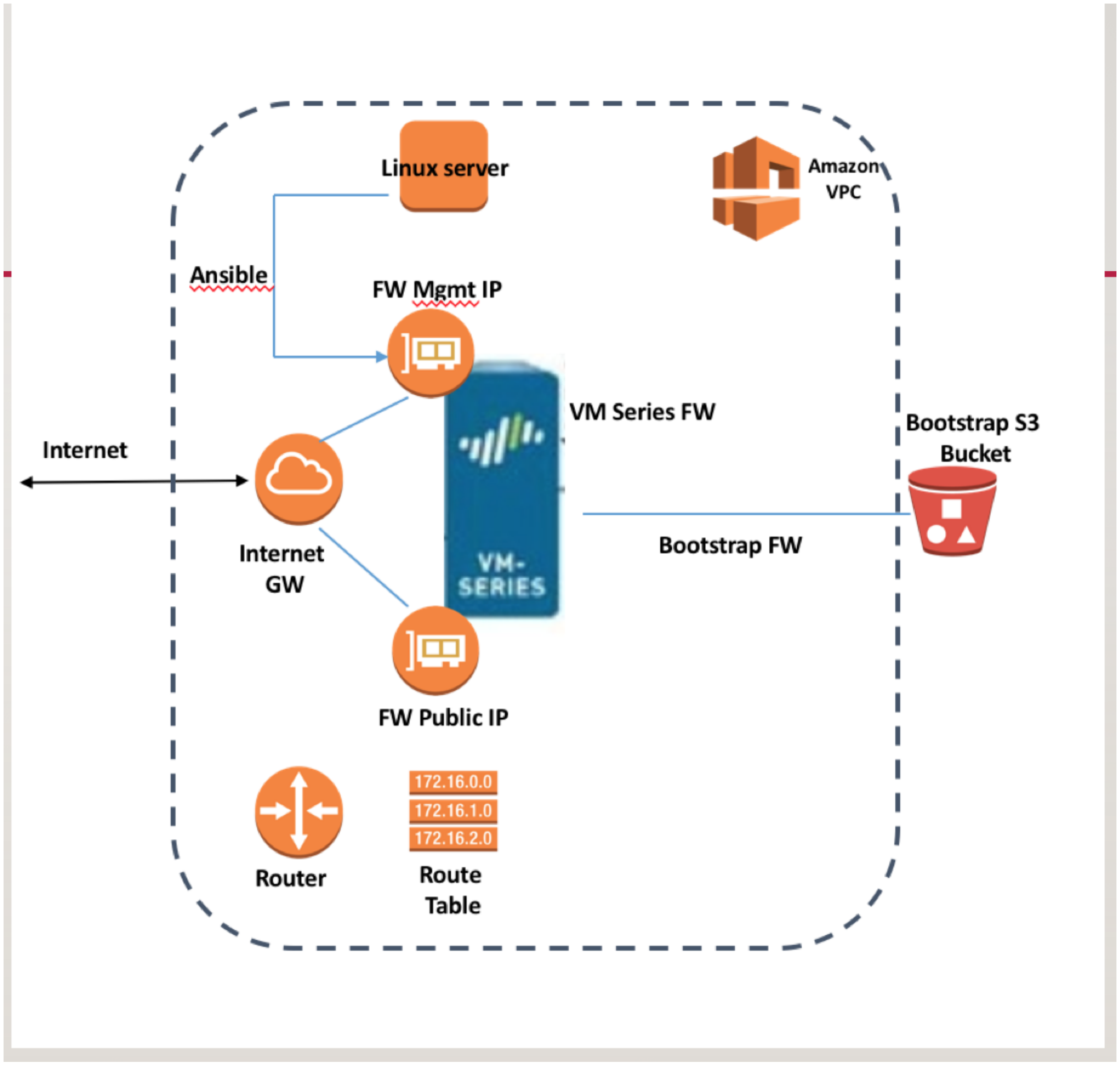
12. Once deployed, open the AWS console to validate that those resources have been created.
13. Delete and clean up the resources once done.

```
Hint: terraform destroy
```

LAB 201

Objectives:

- Apply the concepts and principles learned in the previous lab in order to build out a real-world architectural pattern on AWS.
- Additionally, deploy a VM-Series Firewall on AWS
- Learn how to bootstrap the Firewall on AWS.
- Deploy an application on AWS.
- Learn how to implement and develop a template which will achieve all of the above.



The following table describes the entities / resources that will need to be deployed on AWS along with suggested / sensible values for the parameters in order to make the configuration aspect simpler. In some cases the entire configuration is provided under the “Value” tab with both the argument name and the value for easier consumption again.

Resource Name	Argument Name	Value
aws_vpc	cidr_block	10.0.0.0/16

aws_iam_role	assume_role_policy	`<<EOF`
		`{`
		`  "Version": "2012-10-17",`
		`  "Statement": [`
		`    {`
		`      "Effect": "Allow",`
		`      "Principal": {`
		`      "Service": "ec2.amazonaws.com"`
		`    },`
		`      "Action": "sts:AssumeRole"`
		`    }`
		`  ]`
		`}`
		`EOF`

aws_iam_role_policy	policy	`<<EOF`
		`{`
		`  "Version" : "2012-10-17",`
		`  "Statement": [`
		`    {`
		`      "Effect": "Allow",`
		`      "Action": "s3:ListBucket",`
		`      "Resource":`
		`      "arn:aws:s3:::${var.MasterS3Bucket}"`
		`    },`
		`    {`
		`    "Effect": "Allow",`
		`    "Action": "s3:GetObject",`
		`    "Resource":`
		`      "arn:aws:s3:::${var.MasterS3Bucket}/*"`
		`    }`
		`  ]`
		`}`
		`EOF`

aws_iam_instance_profile	Path	“/”
aws_subnet (Public Subnet)	cidr_block availability_zone	cidr_block = "${var.PublicCIDR_Block}" "${data.aws_availability_zones.available.names[0]}"
aws_subnet (Web Subnet)	cidr_block availability_zone	cidr_block = "${var.WebCIDR_Block}" "${data.aws_availability_zones.available.names[0]}"

aws_network_acl	subnet_ids	`[`
		`"${aws_subnet.NewPublicSubnet.id}",`
		`"${aws_subnet.NewWebSubnet.id}",`
		`]`

aws_network_acl_rule
 	
(ACL #1)
	Param and values combined in the next column	`network_acl_id =`
		`"${aws_network_acl.aclb765d6d2.id}"`
		`rule_number    = 100`
		`egress         = true`
		`protocol       = "-1"`
		`rule_action    = "allow"`
		`cidr_block     = "0.0.0.0/0"`

aws_network_acl_rule
 	
(ACL #2)
	Param and values combined in the next column	`network_acl_id =`
		`"${aws_network_acl.aclb765d6d2.id}"`
		`rule_number    = 100`
		`protocol       = "-1"`
		`rule_action    = "allow"`
		`cidr_block     = "0.0.0.0/0"`

aws_route_table (#1)	vpc_id	"${aws_vpc.main.id}"
aws_route_table (#2)	vpc_id	"${aws_vpc.main.id}"

aws_network_interface
(#1)
		`subnet_id       =`
		`"${aws_subnet.NewPublicSubnet.id}"`
		`security_groups =`
		`["${aws_security_group.sgWideOpen.id}"]`
		`source_dest_check = false`
		`private_ips_count = 1`
		`private_ips = ["10.0.0.99"]`

aws_network_interface
(#2)
		`subnet_id       =`
		`"${aws_subnet.NewPublicSubnet.id}"`
		`security_groups =`
		`["${aws_security_group.sgWideOpen.id}"]`
		`source_dest_check = false`
		`private_ips_count = 1`
		`private_ips = ["10.0.0.100"]`

aws_network_interface
(#3)
		`subnet_id       = "${aws_subnet.NewWebSubnet.id}"`
		`security_groups =`
		`["${aws_security_group.sgWideOpen.id}"]`
		`source_dest_check = false`
		`private_ips_count = 1`
		`private_ips = ["10.0.1.11"]`

aws_network_interface
(For the web instance)
		subnet_id = "${aws_subnet.NewWebSubnet.id}"
		  security_groups =
		      ["${aws_security_group.sgWideOpen.id}"]
		  source_dest_check = false
		  private_ips_count = 1
		  private_ips = ["10.0.1.101"]

aws_eip (Public IP)		vpc = true depends_on = ["aws_vpc.main", "aws_internet_gateway.InternetGateway"]
aws_eip (Mgmt IP)		vpc = true depends_on = ["aws_vpc.main", "aws_internet_gateway.InternetGateway"]

aws_internet_gateway		vpc_id = "${aws_vpc.main.id}"
		  tags {
		    Application = "${var.StackName}"
		    Network = "MGMT"
		    Name = "${join("-", list(var.StackName,
		      "InternetGateway"))}"
		  }

aws_eip_association (Mgmt IP)		network_interface_id =
		      "${aws_network_interface.FWManagementNetworkInterface.id}"
		  allocation_id =
		      "${aws_eip.ManagementElasticIP.id}"

aws_eip_association (Public Interface IP)		network_interface_id =
		      "${aws_network_interface.FWPublicNetworkInterface.id}"
		  allocation_id = "${aws_eip.PublicElasticIP.id}"

aws_route_table_association		subnet_id = "${aws_subnet.NewPublicSubnet.id}"
		  route_table_id =
		      "${aws_route_table.rtb049a2461.id}"

aws_route
 	
(Route 1)
		route_table_id =
		      "${aws_route_table.rtb059a2460.id}"
		  destination_cidr_block = "0.0.0.0/0"
		  gateway_id =
		      "${aws_internet_gateway.InternetGateway.id}"

aws_route (Route 2)		route_table_id = "${aws_route_table.rtb049a2461.id}" destination_cidr_block = "0.0.0.0/0" gateway_id = "${aws_internet_gateway.InternetGateway.id}"

aws_security_group		name = "sgWideOpen"
		  description = "Wide open security group"
		  vpc_id = "${aws_vpc.main.id}"
		
		  ingress {
		    from_port = "0"

		```
		    to_port      = "0"
		    protocol     = "-1"
		    cidr_blocks = ["0.0.0.0/0"]
		  }

		  egress {
		    from_port        = "0"
		    to_port          = "0"
		    protocol         = "-1"
		    cidr_blocks      = ["0.0.0.0/0"]
		```

aws_instance (VM-Series FW instance)		
		```
		disable_api_termination = false
		  iam_instance_profile =
		      "${aws_iam_instance_profile.FirewallBootstrap
		      InstanceProfile2Tier.name}"
		  instance_initiated_shutdown_behavior = "stop"
		  ebs_optimized = true
		  ami = "${var.PANFWRegionMap[var.aws_region]}"
		  instance_type = "m4.xlarge"

		  ebs_block_device {
		    device_name = "/dev/xvda"
		    volume_type = "gp2"
		    delete_on_termination = true
		    volume_size = 60
		  }

		  key_name = "${var.ServerKeyName}"
		  monitoring = false

		  network_interface {
		    device_index = 0
		    network_interface_id =
		      "${aws_network_interface.FWManagementNetworkI
		      nterface.id}"
		  }

		  network_interface {
		    device_index = 1
		    network_interface_id =
		      "${aws_network_interface.FWPublicNetworkInter
		      face.id}"
		  }

		  network_interface {
		    device_index = 2
		```

```
    network_interface_id =
      "${aws_network_interface.FWPrivate12NetworkIn
      terface.id}"
  }


  user_data = "${base64encode(join("",
      list("vmseries-bootstrap-aws-s3bucket=",
      var.MasterS3Bucket)))}"
```

aws_instance

```
disable_api_termination = false
  instance_initiated_shutdown_behavior = "stop"
  ami = "${var.UbuntuRegionMap[var.aws_region]}"
  instance_type = "t2.micro"

  key_name = "${var.ServerKeyName}"
  monitoring = false

  network_interface {
    #delete_on_termination = true
    device_index = 0
    network_interface_id =
      "${aws_network_interface.WPNetworkInterface.i
      d}"
  }

  user_data = "${base64encode(join("", list(
  "#! /bin/bash\n",

          "exec > >(tee
      /var/log/user-data.log|logger -t user-data -s
      2>/dev/console) 2>&1\n",
          "echo \"export
      new_routers='${aws_network_interface.FWPrivat
      e12NetworkInterface.private_ips[0]}'\" >>
      /etc/dhcp/dhclient-enter-hooks.d/aws-default-
      route\n",
          "ifdown eth0\n",
          "ifup eth0\n",

          "while true\n",
          "  do\n",
          "   resp=$(curl -s -S -g --insecure
      \"https://${aws_eip.ManagementElasticIP.publi
      c_ip}/api/?type=op&cmd=<show><chassis-ready><
      /chassis-ready></show>&key=LUFRPT10VGJKTEV6a0
      R4L1JXd0ZmbmNvdUEwa25wMlU9d0N5d292d2FXNXBBeEF
      BUW5pV2xoZz09\")\n",
```

		`"   echo $resp >> /tmp/pan.log\n",` `"   if [[ $resp == *\"[CDATA[yes\"* ]] ; then\n",` `"     break\n",` `"   fi\n",` `"  sleep 10s\n",` `"done\n",` `"apt-get update\n",` `"apt-get install -y apache2 wordpress\n"` `)))` `}"`
null_resource		`triggers {` `key = "${aws_instance.FWInstance.id}"` `}` `provisioner "local-exec" {` `command = "./check_fw.sh ${aws_eip.ManagementElasticIP.public_ip}"` `}`
		`output "FirewallManagementURL" {` `value = "${join("", list("https://", "${aws_eip.ManagementElasticIP.public_ip}")))}"` `}`
		`output "WebURL" {` `value = "${join("", list("http://", "${aws_eip.PublicElasticIP.public_ip}")))}"` `}`
aws_vpc_dhcp_options		`domain_name          = "us-west-2.compute.internal"` `domain_name_servers  = ["AmazonProvidedDNS"]`
aws_vpc_dhcp_options_association		`vpc_id          = "${aws_vpc.main.id}"` `dhcp_options_id = "${aws_vpc_dhcp_options.dopt21c7d043.id}"`

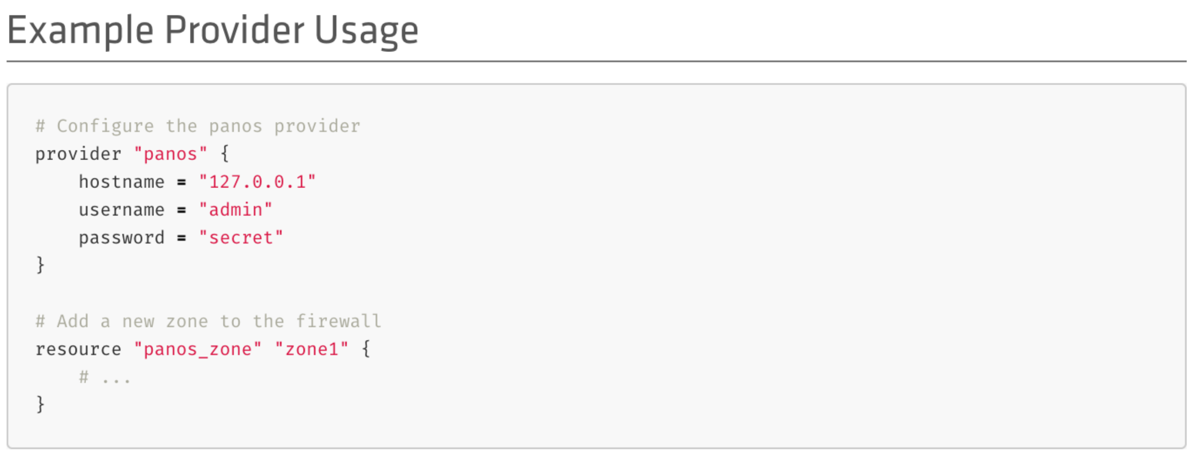
LAB 301

Objectives:

- Learn the various aspects and features of the ```panos``` terraform provider.
- Learn the how to configure the provider
- Learn how to use the provider to:

  - Configure interfaces
  - Create zones
  - Create NAT policies
  - Create security policies
  - Create objects
- Utilize all theses artifacts in a terraform template file to configure the VM-Series FW.

```panos``` provider:



Example Provider Usage

```
# Configure the panos provider
provider "panos" {
    hostname = "127.0.0.1"
    username = "admin"
    password = "secret"
}

# Add a new zone to the firewall
resource "panos_zone" "zone1" {
    # ...
}
```

Please use the following link as a reference to create the necessary resources on the VM-Series firewall using the ```panos``` provider: https://www.terraform.io/docs/providers/panos/

Create the following resources with the associated values in a terraform template to deploy onto a firewall:

Resource Name	Value / Argument Parameters
panos_ethernet_interface (ethernet_1_1)	`name = "ethernet1/1"` `mode = "layer3"` `vsys = "vsys1"` `enable_dhcp = true` `create_dhcp_default_route = true`
panos_ethernet_interface (ethernet_1_1)	`name = "ethernet1/2"` `mode = "layer3"` `vsys = "vsys1"` `enable_dhcp = true` `create_dhcp_default_route = false`
panos_virtual_router	`name = "default"` `interfaces = ["ethernet1/1", "ethernet1/2"]` `depends_on = ["panos_ethernet_interface.ethernet_1_1", "panos_ethernet_interface.ethernet_1_2"]`

panos_zone

(external zone)

```
name        = "external"
  mode        = "layer3"
  interfaces =
      ["${panos_ethernet_interface.ethernet_1_1.name}"]
```

panos_zone

(external zone)

```
name        = "web"
  mode        = "layer3"
  interfaces =
      ["${panos_ethernet_interface.ethernet_1_2.name}"]
```

panos_service_object

```
  name             = "service-tcp-221"
  vsys             = "vsys1"
  protocol         = "tcp"
  description      = "Service object to map port 22 to
      221"
  destination_port = "221"
```

panos_nat_policy

(nat_rule_for_web_ssh)

```
name                     = "web_ssh"
  source_zones           = ["external"]
  destination_zone       = "external"
  source_addresses       = ["any"]
  destination_addresses = ["10.0.0.100"]
  service                = "service-tcp-221"
  sat_type               = "dynamic-ip-and-port"
  sat_address_type       = "interface-address"
  sat_interface          = "ethernet1/2"
  dat_address            = "10.0.1.101"
  dat_port               = "22"

  depends_on = ["panos_service_object.service_tcp_221",
      "panos_zone.external",
    "panos_zone.web",
    "panos_ethernet_interface.ethernet_1_2",
  ]
```

panos_nat_policy

(nat_rule_for_web_http)

```
  name                   = "web_http"
  source_zones           = ["external"]
  destination_zone       = "external"
  source_addresses       = ["any"]
  destination_addresses = ["10.0.0.100"]
  service                = "service-http"
  sat_type               = "dynamic-ip-and-port"
  sat_address_type       = "interface-address"
  sat_interface          = "ethernet1/2"
  dat_address            = "10.0.1.101"
  dat_port               = "80"
  depends_on             = ["panos_zone.external",
      "panos_zone.web",
      "panos_ethernet_interface.ethernet_1_2"]
```

panos_nat_policy
 	
(outbound NAT)
	```
	name                     = "NATAllOut"
	  source_zones           = ["web"]
	  destination_zone       = "external"
	  source_addresses       = ["any"]
	  destination_addresses  = ["any"]
	  sat_type               = "dynamic-ip-and-port"
	  sat_address_type       = "interface-address"
	  sat_interface          = "ethernet1/1"
	  depends_on             = ["panos_zone.external",
	      "panos_zone.web",
	      "panos_ethernet_interface.ethernet_1_1"]
	```

panos_security_policies	
	```
	rule {
	    name                     = "web traffic"
	    source_zones             = ["external"]
	    source_addresses         = ["any"]
	    source_users             = ["any"]
	    hip_profiles             = ["any"]
	    destination_zones        = ["web"]
	    destination_addresses    = ["any"]
	    applications             = ["web-browsing"]
	    services                 = ["application-default"]
	    categories               = ["any"]
	    action                   = "allow"
	  }

	  rule {
	    name                     = "ssh traffic"
	    source_zones             = ["external"]
	    source_addresses         = ["any"]
	    source_users             = ["any"]
	    hip_profiles             = ["any"]
	    destination_zones        = ["web"]
	    destination_addresses    = ["any"]
	    applications             = ["any"]
	    services                 = ["service-tcp-221"]
	    categories               = ["any"]
	    action                   = "allow"
	  }

	  rule {
	    name                     = "allow all out"
	    source_zones             = ["web"]
	    source_addresses         = ["any"]
	    source_users             = ["any"]
	    hip_profiles             = ["any"]
	```

```
    destination_zones     = ["external"]
    destination_addresses = ["any"]
    applications          = ["any"]
    services              = ["any"]
    categories            = ["any"]
    action                = "allow"
  }

   rule {
    name                  = "web traffic 2"
    source_zones          = ["external"]
    source_addresses      = ["any"]
    source_users          = ["any"]
    hip_profiles          = ["any"]
    destination_zones     = ["web"]
    destination_addresses = ["any"]
    applications          = ["web-browsing"]
    services              = ["http-81"]
    categories            = ["any"]
    action                = "allow"
  }

  rule {
    name                  = "ssh traffic2"
    source_zones          = ["external"]
    source_addresses      = ["any"]
    source_users          = ["any"]
    hip_profiles          = ["any"]
    destination_zones     = ["web"]
    destination_addresses = ["any"]
    applications          = ["any"]
    services              = ["service-tcp-222"]
    categories            = ["any"]
    action                = "allow"
  }

  rule {
    name                  = "log default deny"
    source_zones          = ["external"]
    source_addresses      = ["any"]
    source_users          = ["any"]
    hip_profiles          = ["any"]
    destination_zones     = ["web"]
    destination_addresses = ["any"]
    applications          = ["any"]
```

	`    services                = ["any"]`
	`    categories              = ["any"]`
	`    log_start               = true`
	`    log_end                 = true`
	`    action                  = "deny"`
	`  }`
	
	`  depends_on = ["panos_zone.external", "panos_zone.web",`
	`      "panos_nat_policy.outbound_nat","panos_nat_policy.na`
	`      t_rule_for_web_http",`
	`    "panos_nat_policy.nat_rule_for_web_ssh",`
	`    "panos_virtual_router.default_vr",`
	`  ]`

null_resource	`triggers {`
	`    version = "${timestamp()}"`
	`  }`
	
	`  provisioner "local-exec" {`
	`    command = "./commit.sh ${var.fw_ip}"`
	`  }`

panos_service_object
 	
(service_tcp_222)
	`  name             = "service-tcp-222"`
	`  vsys             = "vsys1"`
	`  protocol         = "tcp"`
	`  description      = "Service object to map port 22 to`
	`      222"`
	`  destination_port = "222"`

panos_service_object
 	
(http-81)
	`name             = "http-81"`
	`  vsys             = "vsys1"`
	`  protocol         = "tcp"`
	`  description      = "Service object to map port 22 to`
	`      222"`
	`  destination_port = "81"`

References

[1] Terraform Templates: https://github.com/PaloAltoNetworks/terraform-templates/
[2] PANOS Provider: https://www.terraform.io/docs/providers/panos/