# Fully Automated Terraform GCP CloudGuard R80.10 Deployment

**CB Currier, SE**
**12/19/2018**

## CHALLENGE:

Use Terraform automation to deploy a CloudGuard IAAS (vSEC) gateway on the Google Cloud Platform (GCP).

## BACKGROUND:

Documentation on accomplishing this task is limited and the process of deploying a “public” template is a bit of smoke and mirrors. These challenges are due to a number of factors including nomenclature and deployment methods and architecture.

Ordinarily a customer would go to the GCP marketplace through a browser, search for Check Point and select one of 2 images offered. Then they would go through a process of filling in fields and launching a deployment process. This is not automated.

## EXPECTATIONS:

At the end of this the reader will understand the automation nuances of terraform, GCP and Check Points Cloud Guard. The biggest details revolve around 3 stages in the deployment process. The first stage is Terraform, setting it up, defining credentials and privileges for accessing the second stage.
The second stage is GCP. Terraform files need to properly reference the templates, networks, subnetworks, drives and other aspects of a deployment. Incorporating these references properly will save time and money.
The third stage is CloudGuard. Communicating with and configuring the gateway are a challenge. This is due to GCP’s cloud-init model and to accessibility with the new CheckPoint gateway. Applying a number of different methods from ssh commands, cloud-init parameters and file template processing allows for success.

## PREREQUISITES:

Automation Server – Linux (My distro of choice is Centos 7)
Terraform – Latest Release
GCP
Check Point - CloudGuard – Latest Image
– this version check-point-r8010-gw-byol-032-338-v20180819
ssh scp vi – tools necessary

 

## CONFIGURATION:

The template (main.tf) will create a gateway with 2 network interfaces. When terraform is completed the gateway will have the FTW (First Time Wizard) run, configuring the instance as a gateway only. The gateway will have sic defined and ready for adding to management and receiving a policy push.
A public IP address will be returned at completion.

### Settings:

1. Run `ssh-keygen` on your automation server, hit enter when asked for a password. A private and a public key will be created.These files should be stored in `~/.ssh/identity` and `~/.ssh/identity.pub`; newer SSH versions will store them in `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`. Copy your SSH user public identity file (e.g identity.pub) to cp_admin_auth_key or redefine the value for the variable ssh_pub_key_file located in the file **vars.tf** to point to the local user identity file.

2. Update in **vars.tf** the service account email address to one that has permissions to access GCP APIS.

3. Update the file **terraform-admin.json** with GCP IAM credentials for the email account in #2.

4. Update **export.sh** with credentials information that will be needed by terraform. This then will need to be executed to push the information into the user env. Alternatively copy the export commands with valid credentials and paste in user file **~/.bashrc**. When logging in this data will be part of the user environment.

5. Update the **vars.tf** file with relevant project name and update other fields as desired.

6. Be sure there is ssh access to public GCP addresses as post image deployment requires this.

## PROCESS:

1. Run ‘terraform init’
   - This command sets up the terraform environment and installs add-ons.

2. Run ‘terraform plan’
   - This command reviews all .tf files. The review process examines syntax of all related files and provides exception and debugging information if relevant.

3. If all is good (no errors) - run ‘terraform apply’ - answer yes
   - This command runs the commands defined in the main.tf file. Sometimes additional errors will be returned. These will be the result of account errors, GCP syntax exceptions or missing or incorrect references. Any changes in main.tf after apply is run can be “applied” simply by running ‘terraform apply’ again. In many instances when dealing with a virtual instance – that instance may be destroyed and recreated.

4. If you need to remove - run ‘terraform destroy’.

 

- This command will remove all instances and configurations created in the ‘apply’ command. However changing the main.tf before running ‘destroy’ may mean an exception if the changed main.tf would have a different result in the apply process.

Once complete the script returns a public IP address of the new gateway.
Further scripting might include a Management API call to add the Gateway to management, set sic and install a policy.

## FILES:

**exports.sh**

```
---
#!/bin/bash

#export TF_VAR_org_id=YOUR_ORG_ID
export TF_VAR_billing_account=000000-000000-00000
export TF_ADMIN=${USER}
export TF_CREDS=terraform-admin.json
```

---

**terraform-admin.json**

```
---
{
  "type": "service_account",
  "project_id": "check-point-project"
  "private_key_id": "1111111111111111111111111111111111111111",
  "private_key": "-----BEGIN PRIVATE KEY-----\xxxxxxxxx\n-----END PRIVATE KEY-----\n",
  "client_email": "111111111111-compute@developer.gserviceaccount.com",
  "client_id": "111111111111111111111",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/111111111111-
compute%40developer.gserviceaccount.com"
}
```

---

**vars.tf**

```
---
variable "project" {
  default = "check-point-project"
}
variable "templName" {
  default = "check-point-r8010-gw-byol"
}

variable "templVer" {
  default = "032-338-v20180819"
```

 

```
}

variable "instPref" {
  default = "gw"
}
variable "Ginstype"{
  type = "map"
  default = {
    gw = "Gateway only"
    mgmt = "Management only"
    standalone = "Gateway and Management (Standalone)"
    manual = "Manual Configuration"
  }
}
variable "proj_region" {
  default = "us-central1"
}
variable "proj_zone" {
  default = "us-central1-a"
}
variable "machinetype" {
  default = "n1-standard-2"
}
variable "canipfwd" {
  default = true
}
variable "gtags" {
  default = ["GATEWAY","UID"]
}
variable "gwname" {
  default = "cpgateway"
}
variable "adminpwd" {
  default = "CPr0ck$1!"
}
variable "ssh_pub_key_file" {
  default = "cp_admin_auth_key"
}
variable "sickey" {
  default = "qwe123"
}
variable "allowupdown" {
  default = true
}
variable "eth2_nw" {
  default = "10.10.0.0/24"
}
variable "eth3_nw" {
  default = "10.11.0.0/24"
}
```

 

```
variable "enablMon" {
  default = false
}
variable "svc_accnt_email" {
  default = "111111111111-compute@developer.gserviceaccount.com"
}
variable "genpasswd" {
  default = false
}
```

---

**config.tpl**

```
---
#!/bin/bash
config_system.orig --config-string
"hostname=${gwname}&ftw_sic_key=${sickey}&timezone='America/New_York'&install_sec
urity_managment=false&install_mgmt_primary=false&install_security_gw=true&gateway_d
aip=false&install_ppak=true&gateway_cluster_member=false&download_info=true&reboot_
if_required=true" >> ftw.output &
```

---

**main.tf**

```
---
// Terraform Main file
// v1.0 12/19/2018
// To launch with Terraform on GCP a Check Point R80.10 Cloudguard Instance
// By CB Currier <ccurrier@checkpoint.com>

provider "google" {
  credentials                 = "${file("terraform-admin.json")}"
  project                     = "${var.project}"
  region                      = "${var.proj_region}"
}

// Create default
resource "google_compute_network" "default" {
 name                  = "default"
}

// Create VPC1
resource "google_compute_network" "chkp-vpc1" {
 name                  = "chkp-vpc1"
 auto_create_subnetworks = "false"
}

// Create VPC1-Subnet1
resource "google_compute_subnetwork" "chkp-vpc1-subnet" {
 name          = "chkp-vpc1-subnet"
 ip_cidr_range = "10.0.0.0/24"
 network       = "${google_compute_network.chkp-vpc1.name}"
```

 

```
 region      = "${var.proj_region}"
}

data "template_file" "config_tpl" {
  template = "${file("${path.cwd}/config.tpl")}"

  vars {
    gwname = "${var.gwname}"
    sickey = "${var.sickey}"
  }
}

data "google_compute_image" "cpgateway" {
  name    = "${var.templName}-${var.templVer}"
  project = "checkpoint-public"
}

resource "google_compute_instance" "default" {
  name           = "${var.gwname}"
  machine_type   = "${var.machinetype}"
  zone           = "${var.proj_zone}"
  can_ip_forward = "${var.canipfwd}"

   boot_disk {
    initialize_params {
      image = "${data.google_compute_image.cpgateway.self_link}"
    }
  }
  network_interface {
    network = "default"
    access_config {
      // Ephemeral IP
    }
  }
  network_interface {
    subnetwork = "${google_compute_subnetwork.chkp-vpc1-subnet.self_link}"
    access_config = {}
  }
  metadata {
    adminPasswordSourceMetadata = "${var.adminpwd}"
    enable-oslogin = "true"
    sshKeys = "admin:${file(var.ssh_pub_key_file)}"
    additionalNetwork1 = "netw2"
    additionalSubnetwork1 = "net2sub1"
    allowUploadDownload = "${var.genpasswd}"
    cloudguardVersion = "R80.10-GW"
    computed_sic_key = "${var.sickey}"
    config_path = "projects/${var.project}/configs/${var.gwname}-config"
```

 

```
    config_url =
"https://runtimeconfig.googleapis.com/v1beta1/projects/${var.project}/configs/${var.gwname
}-config"
    deployment_config = "${var.gwname}-config"
    enableMonitoring = "${var.enablMon}"
    externalIP = "Static"
    externalIP1 = "Static"
    gatewayNetwork = "0.0.0.0/0"
    generatePassword = "${var.genpasswd}"
    hasInternet = "true"
    installationType = "${var.Ginstype["${var.instPref}"]}"
    machineType = "${var.machinetype}"
    managementGUIClientNetwork = "0.0.0.0/0"
    numAdditionalNICs = "1"
    shell = "/etc/cli.sh"
    sicKey = "${var.sickey}"
    templateName = "${var.templName}"
    templateVersion = "${var.templVer}"
  }
//  metadata_startup_script = "${data.template_file.gcp_tpl.rendered}"

  depends_on = ["google_compute_firewall.test-firewall",
    "google_compute_network.default",
    "google_compute_network.chkp-vpc1",
  ]

  service_account {
    email  = "${var.svc_accnt_email}"
    scopes = ["https://www.googleapis.com/auth/cloud-
platform","https://www.googleapis.com/auth/devstorage.read_only","https://www.googleapis
.com/auth/logging.write","https://www.googleapis.com/auth/monitoring.write","https://www.g
oogleapis.com/auth/servicecontrol","https://www.googleapis.com/auth/service.management
.readonly","https://www.googleapis.com/auth/trace.append","https://www.googleapis.com/a
uth/source.full_control", "https://www.googleapis.com/auth/cloudruntimeconfig" ]
  }
  provisioner "local-exec" {
   command = "sleep 120; sshpass -p 'admin' ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null
admin@${google_compute_instance.default.network_interface.0.access_config.0.nat_ip}
'set user admin shell /bin/bash'"
  }
  provisioner "local-exec" {
   command = "sshpass -p 'admin' ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null
admin@${google_compute_instance.default.network_interface.0.access_config.0.nat_ip}
'echo ${data.template_file.config_tpl.rendered} > /home/admin/ftwstart'"
  }
  provisioner "local-exec" {
    command = "sshpass -p 'admin' ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null
```

 

```
admin@${google_compute_instance.default.network_interface.0.access_config.0.nat_ip}
'chmod 755 /home/admin/ftwstart;nohup /bin/bash /home/admin/ftwstart'"
  }
}

resource "google_compute_firewall" "test-firewall" {
   name    = "test-firewall"
   network = "${google_compute_network.default.name}"

   allow {
     protocol = "icmp"
   }
   allow {
     protocol = "tcp"
   }
   allow {
     protocol = "icmp"
   }
   allow {
     protocol = "udp"
   }
   allow {
     protocol = "sctp"
   }
   allow {
     protocol = "esp"
   }

   source_ranges = ["0.0.0.0/0"]
}


// Output
output "gw-public-default-ip" {
  value = "${google_compute_instance.default.network_interface.0.access_config.0.nat_ip
}"
}
```

 