



Check Point
SOFTWARE TECHNOLOGIES LTD



Cloud Automation using Terraform (IAC) How-to Guide

By: Shawn Luyke

Contents

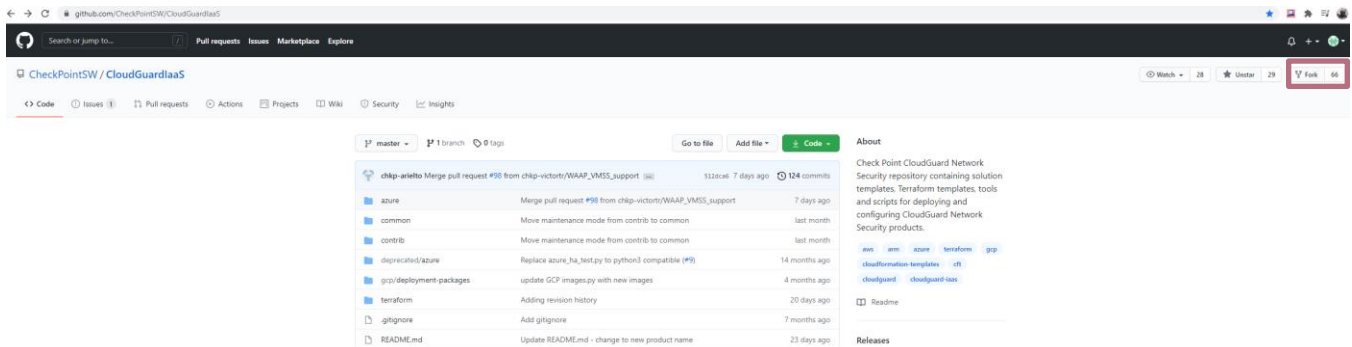
.....	1
Overview.....	3
GitHub Repository	3
GitHub Desktop	4
Visual Studio Code	5
Auto-provision into an existing manager	13
Auto-provisioning working.....	15

Overview

The following document will guide you on how to use Github, Terraform, and Visual Studio Code to deploy Check Point Gateways in AWS, Azure, and GCP. The document will also demonstrate how to auto-provision a Check Point Gateway to a Check Point Manager using the cloud controller “CME”

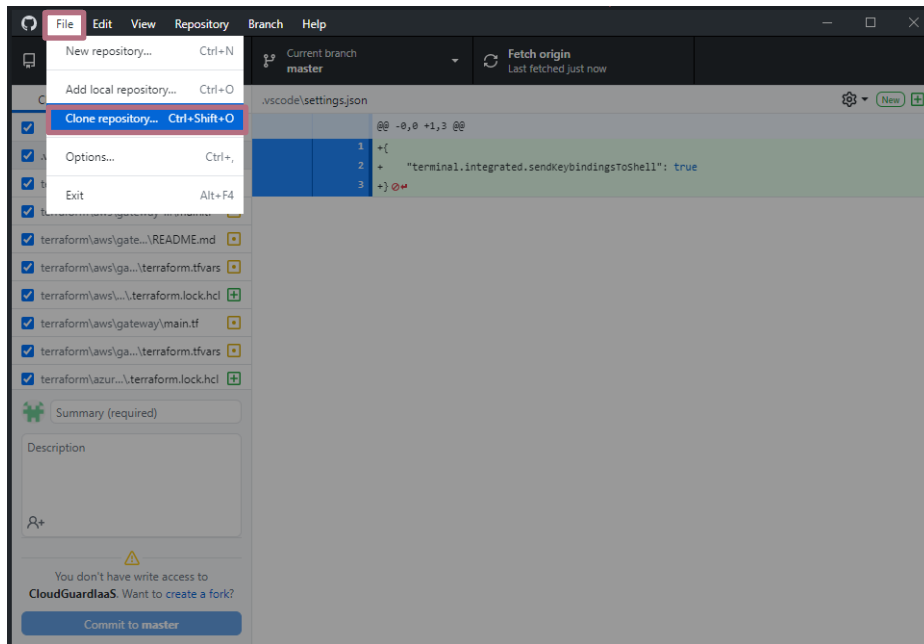
GitHub Repository

- Create yourself a Github account.
- Login and browse to the following URL - <https://github.com/CheckPointSW/CloudGuardIaaS>
- Once at the CheckPointSW/CloudGuardIaaS, fork the repo to your personal repositories library

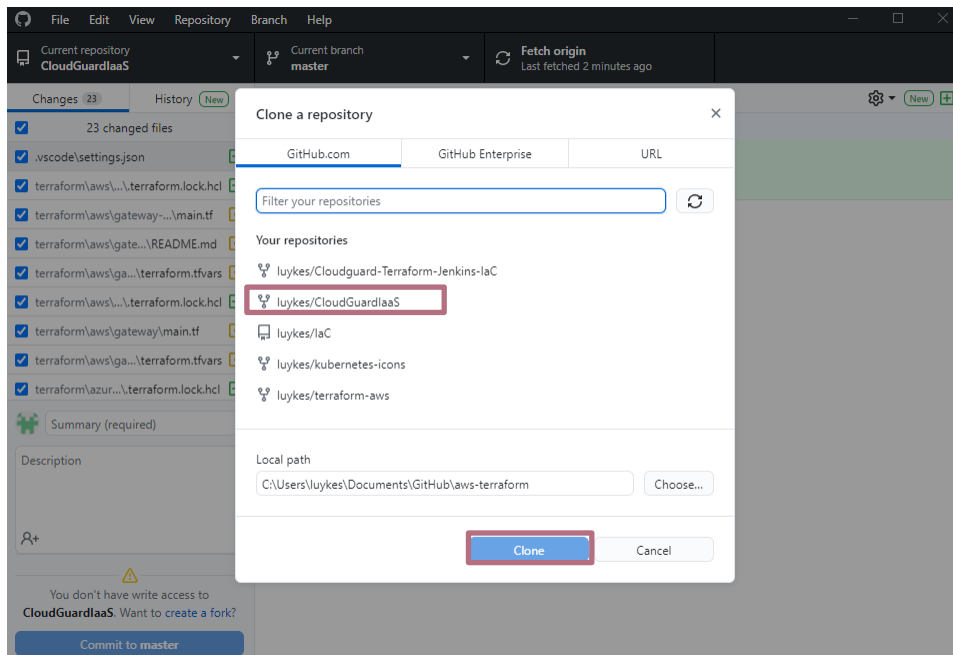


GitHub Desktop

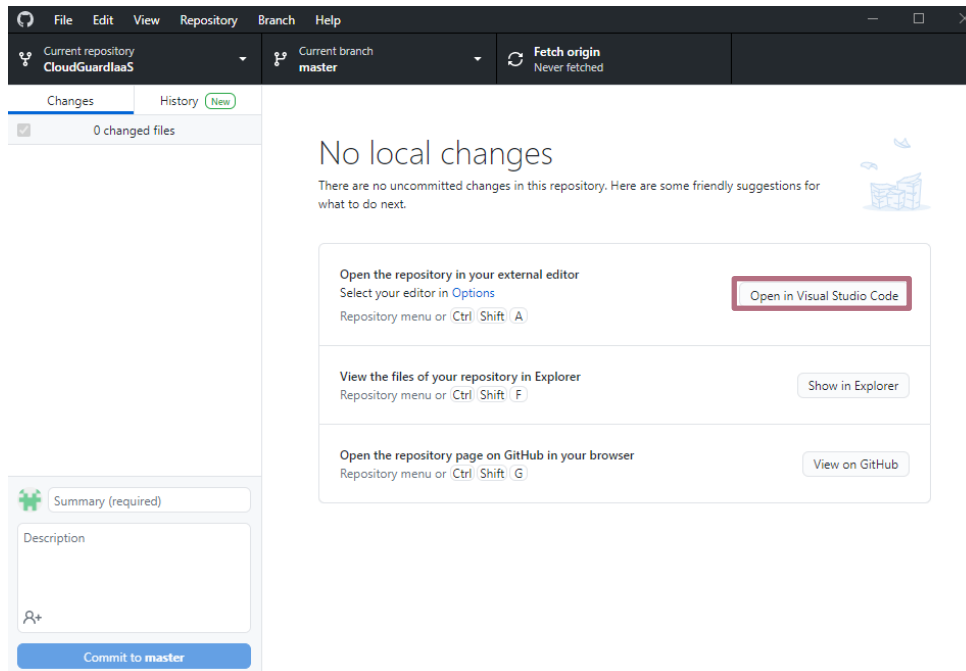
- Download Github Desktop from the following link – <https://desktop.github.com/>
- Once Github Desktop is installed, log in with your newly created Github credentials.
- Clone the repo you forked in the previous step.



- Select the appropriate repo from the list



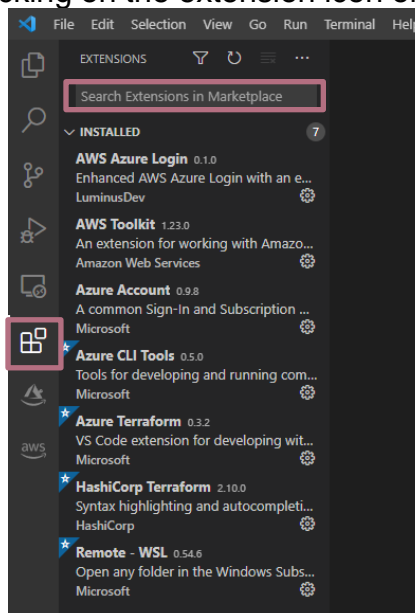
- Once cloning has been completed to your local machine, you can now open GitHub code with Visual Studio Code.



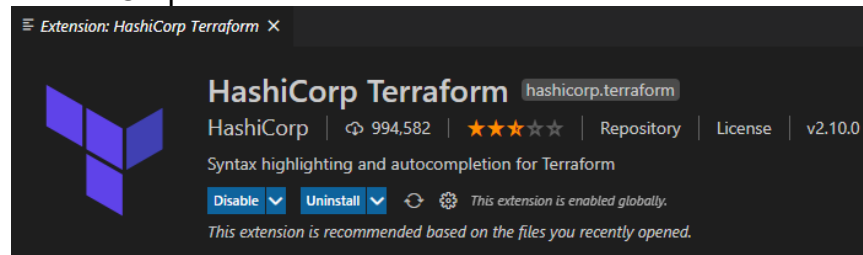
Visual Studio Code

Following packages and add-on needs to be installed on your machine/Visual Studio Code:

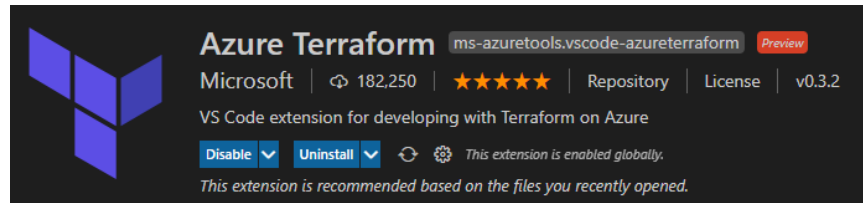
- Terraform will need to be installed on your local machine. Please follow the following link to accomplish this
 - Getting Started with Terraform on Windows: Install, Setup and Demo: https://adamtheautomator.com/terraform-windows/#The_Easy_Way
- Once Terraform is successfully installed the following packages need to be installed within Visual Studio Code.
- Search for an extension by clicking on the extension icon on the left of Visual Studio Code.



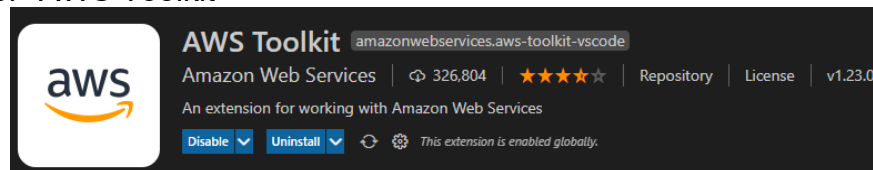
- Search for “HashiCorp Terraform”



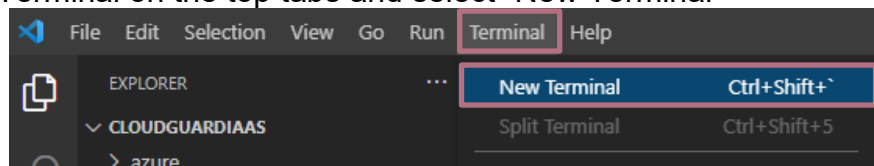
- Search for “Azure Terraform”



- Search for “AWS Toolkit”



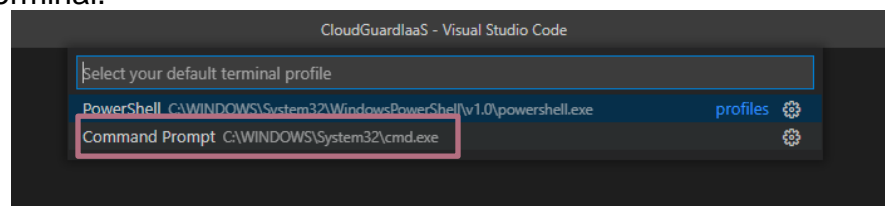
- By default terminal in Terraform use Powershell, this needs to be changed to CMD
 - Click on Terminal on the top tabs and select “New Terminal”



- A new terminal will appear at the bottom. Left of the terminal screen is a drop-down



- Open drop-down and select “Select Default Profile” and then choose “cmd: as the default terminal.



- NOTE: Visual Studio Code will need to be restarted for this change to take effect.

- Reason: Powershell does not understand “&&” signs. When deploying gateways in a new AWS VPC the”&&” is being used to auto-approve the newly created VPC before it starts deploying the rest of the infrastructure.

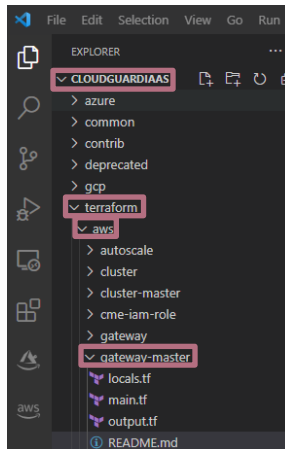
```

- Create or modify the deployment:
- Due to terraform limitation, the apply command is:
...
terraform apply -target=aws_route_table.private_subnet_rtb -auto-approve && terraform apply
...
>Once terraform is updated, we will update accordingly.

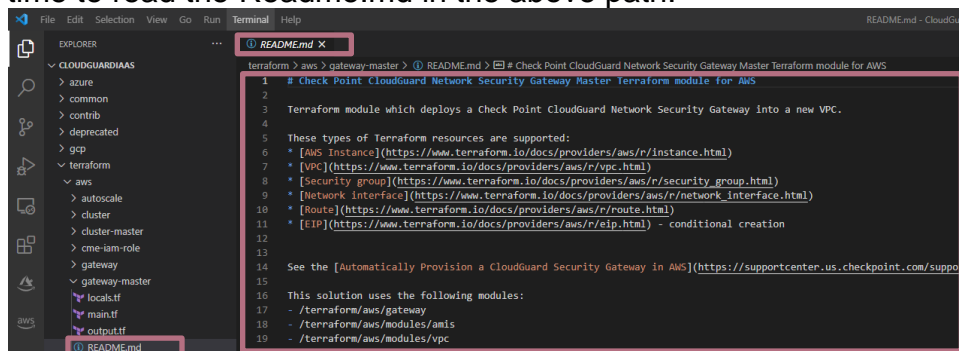
```

Choose your deployment:

- Will use **AWS** as an example: You want to deploy AWS gateways in a New VPC
- Browse to the appropriate file location on the left of Visual Studio Code



- Take some time to read the Readme.md in the above path.



- Follow the instructions closely
- Take note of terraform.tvvars inputs in the readme.md

name	Description	Type	Allowed values	Default	Required
aws_cidr	The CIDR block of the VPC	string	n/a	n/a	yes
public_subnet_map	A map of pairs (availability-zone - subnet-suffix-number), each entry creates a subnet. Minimum 1 pair. (e.g. ["us-east-1a" - 1])	map	n/a	n/a	yes
private_subnet_map	A map of pairs (availability-zone - subnet-suffix-number), each entry creates a subnet. Minimum 1 pair. (e.g. ["us-east-1a" - 2])	map	n/a	n/a	yes
subnet_bit_length	Number of additional bits with which to extend the vpc cidr. For example, if given a vpc_cidr ending in /16 and a subnet_bit_length value of 4, the resulting subnet address will have length /20	number	n/a	n/a	yes
gateway_name	(Optional) The name tag of the Security gateway instance	string	n/a	Check-Point-Gateway-TF	no
gateway_instance_type	The instance type of the Security gateway	string	c5.large	c5.large	no
cidr_large	The EC2 key pair name to allow SSM access to the instance	string	n/a	n/a	yes
allocate_and_associate_eip	If set to true, an elastic IP will be allocated and associated with the launched instance	bool	true/false	true	no
enable_iam_role	four volume size (gb) - additional (gb)	number	n/a	100	no
volume_encryption	AMI or OS key identifier: the key ID, alias or ARN. Key aliases should be prefixed with "alias/" (e.g. for AWS default alias "aws/eks" - insert "alias/aws/eks")	string	n/a	alias/aws/eks	no
enable_instance_connect	Enable SSM connection over aws ssm connect. Supporting regions can be found here: (https://aws.amazon.com/doc/2019/06/20/introducing-aws-ssm-ec2-instance-connect)	bool	true/false	false	no
instance_tags	(Optional) A map of tags as key-value pairs. All tags will be added to the Security gateway EC2 instance	map(string)	n/a	{}	no
gateway_version	Gateway version and license	string	100-40-800	100-40-800	no
enable_ssh	Set the sshd sshd to enable sshd on the instance	string	no	no	no
gateway_lic_key	The Secure Internal Communication key for trusted connection between Check Point components. Choose a random string consisting of at least 8 alphanumeric characters	string	n/a	n/a	yes
gateway_provision_key	(Optional) Admin user's password hash (see command "openssl passwd -1 random") to get the admin's hash	string	n/a	n/a	no
resource_tag_name	(Optional) Security gateway group hostname	string	n/a	n/a	no
allow_update_download	Automatically download R80e contracts and other important data. Improve product experience by sending data to Check Point	bool	true/false	true	no
gateway_management_server	(Optional) IP address of the management server to use on the initial boot	string	n/a	n/a	no
primary_ip	(Optional) The IP address of Network Time Protocol primary server	string	n/a	169.254.169.121	no
secondary_ip	(Optional) The IP address of Network Time Protocol secondary server	string	n/a	4.pool.ntp.org	no
control_gateway_over_public_or_private_address	Determines if the Security gateway is provisioned using its public or private address	string	public	private	no
management_server	The name that represents the Security Management server in the automatic provisioning configuration	string	n/a	n/a	no
configuration_template	(Optional) A name of a Security gateway configuration template in the automatic provisioning configuration	string	n/a	n/a	no

- The root key credentials can be provided in a few different ways within terraform. Please follow the Readme.md for these details.
 - The credentials configs can be located on the main.tf file

```

- ## Configurations

The "main.tf" file includes the following provider configuration block used to configure the credentials for the authentication with AWS, as well as a default region for your resources:

provider "aws" {
  region = var.region
  access_key = var.access_key
  secret_key = var.secret_key
}

The provider credentials can be provided either as static credentials or as [Environment Variables](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/environment-variables).
Static credentials can be provided by adding an access_key and secret_key in /terraform/aws/gateway-master/"terraform.tfvars" file as follows:

...
region = "us-east-1"
access_key = "my-access-key"
secret_key = "my-secret-key"
...

In case the Static credentials are used, perform modifications described below:(br/>
a. the next lines in main.tf file, in the provider aws resource, need to be commented for sub-module /terraform/aws/gateway:
...
provider "aws" {
  // region = var.region
  // access_key = var.access_key
  // secret_key = var.secret_key
}
...

In case the Environment Variables are used, perform modifications described below:(br/>
a. the next lines in main.tf file, in the provider aws resource, need to be commented:
...
provider "aws" {
  // region = var.region
  // access_key = var.access_key
  // secret_key = var.secret_key
}
...

b. the next lines in main.tf file, in the provider aws resource, need to be commented for sub-module /terraform/aws/gateway:
...
provider "aws" {
  // region = var.region
  // access_key = var.access_key
  // secret_key = var.secret_key
}
...

```

- Then go to terraform.tfvars on the left tab
 - Fill in all the fields as per the Readme.md

```

terraform > aws > gateway-master > terraform.tfvars
1 //PLEASE refer to README.md for accepted values FOR THE VARIABLES BELOW
2
3 // --- VPC Network Configuration ---
4 vpc_cidr = "10.0.0.0/16"
5 public_subnets_map = {
6   "us-east-1a" = 1
7 }
8 private_subnets_map = {
9   "us-east-1a" = 2
10 }
11 subnets_bit_length = 8
12
13 // --- EC2 Instance Configuration ---
14 gateway_name = "Check-Point-Gateway-tf"
15 gateway_instance_type = "c5.xlarge"
16 key_name = "privatekey"
17 allocate_and_associate_eip = true
18 volume_size = 100
19 volume_encryption = ""
20 enable_instance_connect = false
21 instance_tags = {
22   key1 = "value1"
23   key2 = "value2"
24 }
25
26 // --- Check Point Settings ---
27 gateway_version = "R80.40-PAYG-NGTP"
28 admin_shell = "/bin/bash"
29 gateway_SICKey = "12345678"
30 gateway_password_hash = "12345678"
31
32 // --- Advanced Settings ---
33 resources_tag_name = "tag-name"
34 gateway_hostname = "gw-hostname"
35 allow_upload_download = true
36 gateway_bootstrap_script = "echo 'this is bootstrap script' > /home/admin/testfile.txt"
37 primary_ntp = ""
38 secondary_ntp = ""
39
40 // --- (Optional) Automatic Provisioning with Security Management Server Settings ---
41 control_gateway_over_public_or_private_address = "private"
42 management_server = ""
43 configuration_template = ""

```

- Take note of the colors
 - Blue = Variable description
 - Orange = Accepted values as per the Readme.md
 - Green = Have been comment out of the code.
 - Some variables are either, "String", "Number" or "Bool".
 - String uses inverted commas ". Numbers and Bool do not use " " .

Deployment:

- Now that all the variables have been filled in, now the fun can start.
- Open a new terminal and go to file path of the deployment package (example: `cd:/user/documents/github/CGIaaS/terraform/aws/gateway`)

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS>cd terraform/aws/gateway-master
C:\Users\Luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master>
```

- When running the directory command all of the following files need to display, especially `main.tf`

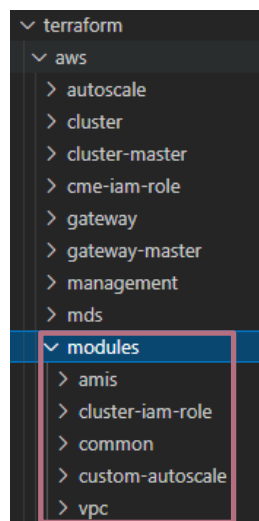
```
C:\Users\Luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master>dir
Volume in drive C is Windows
Volume Serial Number is 4A42-880E

Directory of C:\Users\Luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master

18/04/2021 09:32 pm <DIR>      .
18/04/2021 09:32 pm <DIR>      ..
18/04/2021 09:32 pm             2,322 locals.tf
18/04/2021 09:32 pm             2,146 main.tf
18/04/2021 09:32 pm             959 output.tf
18/04/2021 09:32 pm            10,669 README.md
18/04/2021 09:32 pm             1,195 terraform.tfvars
18/04/2021 09:32 pm             5,566 variables.tf
18/04/2021 09:32 pm             210 versions.tf
                7 File(s)          23,067 bytes
                2 Dir(s)      20,686,696,448 bytes free

C:\Users\Luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master>
```

- These terraform scripts rely on modules. The modules need to be part of the entire file directory. (If cloning was done successfully these modules will be part of the folder directory)



- Now you will need to run “terraform init”. This command will initialise the necessary terraform files and modules that are being used in this script.

```
C:\Users\luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master>terraform init
Initializing modules...
- launch_gateway_into_vpc in ..\gateway
- launch_gateway_into_vpc.amis in ..\modules\amis
- launch_gateway_into_vpc.common_eip in ..\modules\common\elastic_ip
- launch_gateway_into_vpc.common_gateway_instance in ..\modules\common\gateway_instance
- launch_gateway_into_vpc.common_gateway_instance.validate_instance_type in ..\modules\common\instance_type
- launch_gateway_into_vpc.common_internal_default_route in ..\modules\common\internal_default_route
- launch_gateway_into_vpc.common_permissive_sg in ..\modules\common\permissive_sg
- launch_gateway_into_vpc.validate_gateway_version in ..\modules\common\version_license
- launch_gateway_into_vpc.validate_instance_type in ..\modules\common\instance_type
- launch_vpc in ..\modules\vpc
- validate_gateway_version in ..\modules\common\version_license
- validate_instance_type in ..\modules\common\instance_type

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.24.1"...
- Finding hashicorp/http versions matching "~> 2.0.0"...
- Finding latest version of hashicorp/null...
- Installing hashicorp/aws v3.24.1...
- Installed hashicorp/aws v3.24.1 (signed by HashiCorp)
- Installing hashicorp/http v2.0.0...
- Installed hashicorp/http v2.0.0 (signed by HashiCorp)
- Installing hashicorp/null v3.1.0...
- Installed hashicorp/null v3.1.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

C:\Users\luykes\Documents\GitHub\aws-terraform\CloudGuardIaaS\terraform\aws\gateway-master>
```

- If all variables have been filled in correctly the “terraform apply” needs to be run. Terraform will now run checks to see if there are any errors within the script.
 - Note: “terraform apply” did not work. This is because of the terraform limitation in being used in this example. Under normal circumstance “terraform plan” will work

```
C:\Users\luykes\Documents\GitHub\aws-terraform\CGIaaS\terraform\aws\gateway-master>terraform apply
Error: Invalid count argument

on ..\modules\common\internal_default_route\main.tf line 2, in resource "aws_route" "internal_default_route":
  2:   count = local.internal_route_table_condition

The "count" value depends on resource attributes that cannot be determined
until apply, so Terraform cannot predict how many instances will be created.
To work around this, use the -target argument to first apply only the
resources that the count depends on.

C:\Users\luykes\Documents\GitHub\aws-terraform\CGIaaS\terraform\aws\gateway-master>terraform apply -target=aws.route.table.private.subnet.rtb -auto-approve && terraform apply
```

- With the “terraform apply” being run. It will start to deploy the VPC before hand. Once that has been done. Terraform will supply you with an output on whats going to be deployed in the AWS environment.
 - The rest of the modules are depended on the newly created VPC:

```

C:\Users\llokes\Documents\git\194\aws-terraform\GitLab\terraform\aws\gateway-master\terraform apply -target=aws_route_table.private_subnet_rtb -auto-approve && terraform apply
module.launch_vpc.aws_vpc.vpc: Creating...
module.launch_vpc.aws_vpc.vpc: Creation complete after 10s [id=vpc-0ed2e73f8e0a7dfed]
module.launch_vpc.aws_internet_gateway.igw: Creating...
module.launch_vpc.aws_route_table.public_subnet_rtb: Creating...
module.launch_vpc.aws_subnet.private_subnets["us-east-1a"]: Creating...
module.launch_vpc.aws_subnet.private_subnets["us-east-1a"]: Creation complete after 25 [id=rtb-0f3a108c5b84e537]
module.launch_vpc.aws_subnet.private_subnets["us-east-1a"]: Creation complete after 3s [id=subnet-09ac0984dbdfde4d]
module.launch_vpc.aws_subnet.private_subnets["us-east-1a"]: Creation complete after 4s [id=subnet-0085548b7f33590b09]
module.launch_vpc.aws_route_table.association_public_rtb_to_public_subnets["rtb-0-1-0/24"]: Creating...
module.launch_vpc.aws_internet_gateway.igw: Creation complete after 4s [id=igw-0da1ffa001e6b32]
module.launch_vpc.aws_route_vpc_internet_access: Creating...
module.launch_vpc.aws_route_table.association_public_rtb_to_public_subnets["10.0.1.0/24"]: Creation complete after 1s [id=rtbassoc-045d0524ac04c730]
module.launch_vpc.aws_route_vpc_internet_access: Creation complete after 3s [id=r-rtb-0f3a108c5b84e537100028904]
aws_route_table.private_subnet_rtb: Creating...
aws_route_table.private_subnet_rtb: Creation complete after 3s [id=rtb-0ca090e247ee0111]

Warning: Resource targeting is in effect

You are creating a plan with the -target option, which means that the result
of this plan may not represent all of the changes requested by the current
configuration.

The -target option is not for routine use, and is provided only for
exceptional situations such as recovering from errors or mistakes, or when
Terraform specifically suggests to use it as part of an error message.

Warning: Applied changes may be incomplete

The plan was created with the -target option in effect, so some changes
requested in the configuration may have been ignored and the output values may
not be fully updated. Run the following command to verify that no other
changes are pending:
  terraform plan

Note that the -target option is not suitable for routine use, and is provided
only for exceptional situations such as recovering from errors or mistakes, or
when Terraform specifically suggests to use it as part of an error message.

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:
internal_rtb_id = "rtb-0ca090e247ee0111"
vpc_id = "vpc-0ed2e73f8e0a7dfed"
vpc_private_subnets_ids_list = [
  "subnet-09ac0984dbdfde4d",
]
vpc_public_subnets_ids_list = [
  "subnet-0085548b7f33590b09",
]

```

- Terraform will supply you with the following output, when ready and happy with what is going to happen, enter the value “yes” to confirm your changes.

```

}
+ vpc_id = "vpc-0ed2e73f8e0a7dfed"
}

Plan: 8 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ ami_id = "ami-08c11febea074b3d3"
+ gateway_instance_id = (known after apply)
+ gateway_instance_name = "Check-Point-Gateway-tf"
+ gateway_public_ip = [
+ (known after apply),
]
+ gateway_url = (known after apply)
+ permissive_sg_id = (known after apply)
+ permissive_sg_name = (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: yes

```

- Now terraform will be deploying and creating the resources into the AWS account as per the credentials specified in the main.tf file.

```

Enter a value: yes
aws_route_table_association.private_rt_to_private_subnets: Creating...
module.launch_gateway_into_vpc.module.common_permissive_sg.aws_security_group.permissive_sg: Creating...
aws_route_table_association.private_rt_to_private_subnets: Creation complete after 1s [id=rtbassoc-8c2a20b7365b79c24]
module.launch_gateway_into_vpc.module.common_permissive_sg.aws_security_group.permissive_sg: Creation complete after 8s [id=sg-053794571034f200c]
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Creating...
module.launch_gateway_into_vpc.aws_network_interface.public_eni: Creating...
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Still creating... [18s elapsed]
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Still creating... [18s elapsed]
module.launch_gateway_into_vpc.aws_network_interface.public_eni: Still creating... [20s elapsed]
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Still creating... [20s elapsed]
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Still creating... [30s elapsed]
module.launch_gateway_into_vpc.aws_network_interface.private_eni: Creation complete after 34s [id=eni-048fa8bb64104444]
module.launch_gateway_into_vpc.module.common_internal_default_route.aws_route.internal_default_route[0]: Creating...
module.launch_gateway_into_vpc.module.common_gateway_instance.aws_instance.gateway_instance: Still creating... [18s elapsed]
module.launch_gateway_into_vpc.module.common_gateway_instance.aws_instance.gateway_instance: Creation complete after 19s [id=i-0d2557804f0c6e6a4]
module.launch_gateway_into_vpc.module.common_eip.aws_eip.gateway_eip[0]: Creating...
module.launch_gateway_into_vpc.module.common_eip.aws_eip.gateway_eip[0]: Creation complete after 4s [id=eipalloc-0fa22006c6df3898c]
module.launch_gateway_into_vpc.module.common_eip.aws_eip_association.address_assoc[0]: Creating...
module.launch_gateway_into_vpc.module.common_eip.aws_eip_association.address_assoc[0]: Creation complete after 3s [id=eipassoc-0d89735897abe38fe]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:
ami_id = "ami-0bc11feba074b3d3"
gateway_instance_id = "i-0d2557804f0c6e6a4"
gateway_instance_name = "Check-Point-Gateway.tf"
gateway_public_ip = [
  "52.200.18.47",
]
gateway_url = "https://52.200.18.47"
internal_rt_id = "rtb-0cae50a4247ea0111"
permissive_sg_id = "sg-053794571034f200c"
permissive_sg_name = "Tag-Name-PermissiveSecurityGroup20221041011103064430080001"
vpc_id = "vpc-0ed2b73f8bba29fad"
vpc_private_subnets_ids_list = [
  "subnet-0fac0904b2df6e6d",
]
vpc_public_subnets_ids_list = [
  "subnet-0d8f50b7ff33596b9",
]

```

- Login in to your AWS account to view the newly created Check Point GW's

Auto-provision into an existing manager

On existing manager:

- Install CME package on the manager following SK157492
 - https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoview_solutiondetails=&solutionid=sk157492&partition=Advanced&product=CloudGuard
 - Admin Guide - https://sc1.checkpoint.com/documents/laaS/WebAdminGuides/EN/CP_CME/Content/Topics-Cloud_Management_Extension_CME/Installing_and_Updating_CME.htm?tocpath=____3
 - AWS Manager - https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoview_solutiondetails=&solutionid=sk130372#Setting%20up%20Automatic%20Provisioning
- Cloud Controller has been to be configured on the manager.
 - The below command will initialise a controller and template within the manager.
 - `“autoprov_cfg init AWS -mn <MANAGEMENT-NAME> -tn <CONFIGURATION-TEMPLATE-NAME> -otp <SIC-KEY> -ver <VERSION> -po <POLICY-NAME> -cn <CONTROLLER-NAME> -r <AWS-REGIONS> -ak <AKIA *****> -sk <AoG90*****>”`
 - The template will need to be configured. Here we specify which blades need to be enabled for the gateway at hand.
 - `autoprov_cfg set template -tn < CONFIGURATION-TEMPLATE-NAME> -appi -uf -pp 8080 -hi -ips -ab -av -ia`
 - If the CME has been installed and the controller and template have been configured. You should see something similar.

```
[Expert@mgmt-aws:0]# service cme test
Successfully initialized logger for CME SERVICE
run_local Executing: ['/opt/CPcme/menu/additions/autoprov_cfg.py', '--upgrade'] with:
  data=None
  env=None
Return code= 0
  Output=
Testing the configuration file loads...
run_local Executing: ['/opt/CPcme/menu/additions/autoprov_cfg.py', '--dump'] with:
  data=None
  env=None
Return code= 0
  Output={
    "controllers": {
      "AWS-US": {
        *****,
        "class": "AWS",
        "regions": [
          "us-east-1"
        ],
        *****,
      },
      "Azure-CP": {
        "class": "Azure",
        "credentials": {
          *****,
          *****,
          "grant_type": "client_credentials",
          "tenant": "23d95250-5e04-4996-a7ef-44c2840a5a90"
        },
        "environment": "AzureCloud",
        "subscription": "bc83fdfa-bld9-4c54-86e2-b8239d06060c"
      },
    },
    "delay": 30,
    "management": {
      "host": "localhost",
      "name": "R80"
    },
    "templates": {
      "template": {
        "anti-bot": true,
        "anti-virus": true,
        "application-control": true,
        "https-inspection": true,
        "identity-awareness": true,
        "ips": true,
        *****,
        "policy": "Standard",
        "proxy-ports": [
          "8080"
        ],
        "url-filtering": true,
        "version": "R80.40"
      }
    }
  }
}
```

- Now that the controller and template have been configured. You can run “tail -f /var/log/CPcme/cme.log” on the manager. This will display if the manager has seen any new gateways’ in the cloud account.

```
[Expert@mgmt-aws:0]# tail -f /var/log/CPcme/cme.log
2021-04-18 11:40:19,209 CME_SERVICE INFO No gateways were found in controller Azure-CP
2021-04-18 11:40:19,209 CME_SERVICE INFO
2021-04-18 11:40:19,209 CME_SERVICE INFO
2021-04-18 11:40:19,419 CME_SERVICE INFO There are no gateways known by the management at the end of the iteration
2021-04-18 11:40:19,419 CME_SERVICE INFO ***** End of the iteration number 13 for gateway instances. Iteration time: 0:00:03.427191 *****
2021-04-18 11:40:19,419 CME_SERVICE INFO
2021-04-18 11:40:49,444 CME_SERVICE INFO ***** Starting loop iteration number 14 for gateway instances *****
2021-04-18 11:40:50,039 CME_SERVICE INFO There are no gateways known by the management at the beginning of the iteration
2021-04-18 11:40:51,188 CME_SERVICE INFO No gateways were found in controller AWS-US
2021-04-18 11:40:51,188 CME_SERVICE INFO
2021-04-18 11:40:52,539 CME_SERVICE INFO No gateways were found in controller Azure-CP
2021-04-18 11:40:52,539 CME_SERVICE INFO
2021-04-18 11:40:52,539 CME_SERVICE INFO
2021-04-18 11:40:52,733 CME_SERVICE INFO There are no gateways known by the management at the end of the iteration
2021-04-18 11:40:52,733 CME_SERVICE INFO ***** End of the iteration number 14 for gateway instances. Iteration time: 0:00:03.289056 *****
2021-04-18 11:40:52,734 CME_SERVICE INFO
^C
[Expert@mgmt-aws:0]#
```

Variables in Terraform for Gateways to Auto-deploy:

- For the manager to automate this process, you will need to specify a couple of variables on the terraform.tfvars file in your Visual Studio Code.

```
// --- (Optional) Automatic Provisioning with Security Management Server Settings ---  
control_gateway_over_public_or_private_address = "public"  
management_server = "<MANAGEMENT-NAME>"  
configuration_template = "<CONFIGURATION-TEMPLATE-NAME>"
```

```
"autoprov_cfg init AWS -mn <MANAGEMENT-NAME> -tn <CONFIGURATION-  
TEMPLATE-NAME> -otp <SIC-KEY> -ver <VERSION> -po <POLICY-NAME> -cn  
<CONTROLLER-NAME> -r <AWS-REGIONS> -ak <AKIA *****> -sk <  
AoG90*****>"
```

- Note: The SIC key in the terraform.tfvars need to be the same on the controller configuration of the manager.

```
// --- Check Point Settings ---  
gateway_version = "R80.40-PAYG-NGTP"  
admin_shell = "/bin/bash"  
gateway_SICKey = "12345678"  
gateway_password_hash = "12345678"
```

```
"autoprov_cfg init AWS -mn <MANAGEMENT-NAME> -tn <CONFIGURATION-  
TEMPLATE-NAME> -otp <SIC-KEY> -ver <VERSION> -po <POLICY-NAME> -cn  
<CONTROLLER-NAME> -r <AWS-REGIONS> -ak <AKIA *****> -sk <  
AoG90*****>"
```

Auto-provisioning working

- Small clip of how an AWS gateway being deployed in a NEW VPC and auto-provisioned on an existing manager in AWS
 - <https://youtu.be/6c78dENuZAM>