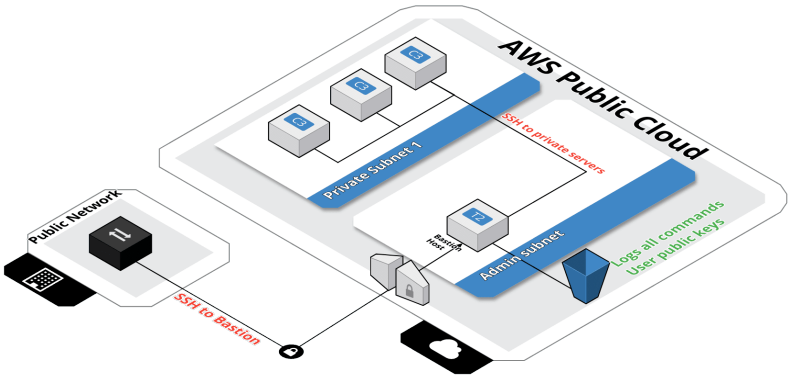
AWS Bastion Terraform module

Terraform module which creates a secure SSH bastion on AWS.

Mainly inspired by Securely Connect to Linux Instances Running in a Private Amazon VPC

Features

This module will create an SSH bastion to securely connect in SSH to your pri-



vate instances.
All SSH commands are logged on an S3 bucket for security compliance, in the /logs path.

SSH users are managed by their public key, simply drop the SSH key of the user in the /public-keys path of the bucket. Keys should be named like 'username.pub', this will create the user 'username' on the bastion server.

Then after you'll be able to connect to the server with :

```
ssh [-i path_to_the_private_key] username@bastion-dns-name
```

From this bastion server, you'll able to connect to all instances on the private subnet.

If there is a missing feature or a bug - open an issue.

Usage

```
module "bastion" {
  "source" = "Guimove/bastion/aws"
```

```
  "bucket_name" = "my_famous_bucket_name"
  "region" = "eu-west-1"
  "vpc_id" = "my_vpc_id"
  "is_lb_private" = "true|false"
  "bastion_host_key_pair" = "my_key_pair"
  "create_dns_record" = "true|false"
  "hosted_zone_id" = "my.hosted.zone.name."
  "bastion_record_name" = "bastion.my.hosted.zone.name."
  "bastion_iam_policy_name" = "myBastionHostPolicy"
  "elb_subnets" = [
    "subnet-id1a",
    "subnet-id1b"
  ]
  "auto_scaling_group_subnets" = [
    "subnet-id1a",
    "subnet-id1b"
  ]
  tags = {
    "name" = "my_bastion_name",
    "description" = "my_bastion_description"
  }
}
```

Requirements

- Terraform >= 0.12

Providers

Name	Version
aws	~> 3.0
null	~> 3.0

Inputs

Name	Description	Type	Default	Required
allow_ssh_commands	Allows the SSH user to execute one-off commands. Pass 'True' to enable. Warning: These commands are not logged and increase the vulnerability of the system. Use at your own discretion.	`string`	`""`	no
associate_public_ip_address	Whether or not to associate a public ip	`bool`	`true`	no
auto_scaling_group_subnets	List of subnets where the Auto Scalling Group will deploy the instances	`list(string)`	n/a	yes
bastion_ami	The AMI that the Bastion Host will use.	`string`	`""`	no
bastion_host_key_pair	Select the key pair to use to launch the bastion host	`any`	n/a	yes
bastion_iam_policy_name	Name to apply to the IAM policy to create for granting the instance role access to the bucket	`string`	`"BastionHost"`	no
bastion_instance_count	Number of instances to create	`number`	`1`	no
bastion_launch_template_name	Bastion Launch template name, will also be used for the ASG	`string`	`"bastion-lt"`	no
bastion_record_dns_name	DNS record name to use for the bastion	`string`	`""`	no
bucket_force_destroy	The bucket and all objects should be destroyed when using true	`bool`	`false`	no

Name	Description	Type	Default	Required
bucket_name	Bucket name were the bastion will store the logs	`any`	n/a	yes
bucket_versioning	Enable or disable bucket versioning or not	`bool`	`true`	no

| cidrs | List of CIDRs than can access to the bastion. Default : 0.0.0.0/0 | `list(string)` |

| no | | create_dns_record | Choose if you want to create a record name for the bastion (LB). If true 'hosted_zone_id' and 'bastion_record_name' are mandatory | `any` | n/a | yes | | disk_encrypt | EBS encryption of instance | `bool` | `true` | no | | disk_size | Root device disk size | `number` | `8` | no | | elb_subnets | List of subnet were the ELB will be deployed | `list(string)` | n/a | yes | | extra_user_data_content | Additional scripting to pass to the bastion host. For example, this can include installing postgresql for the `psql` command. | `string` | `""` | no | | hosted_zone_id | Name of the hosted zone were we'll register the bastion DNS name | `string` | `""` | no | | instance_type | Instance size of the bastion | `string` | `"t3.nano"` | no | | is_lb_private | If TRUE the load balancer scheme will be "internal" else "internet-facing" | `any` | n/a | yes | | enable_logs_s3_sync | Enable cron job to copy logs to S3 | `bool` | `true` | yes | | log_auto_clean | Enable or not the lifecycle | `bool` | `false` | no | | log_expiry_days | Number of days before logs expiration | `number` | `90` | no | | log_glacier_days | Number of days before moving logs to Glacier | `number` | `60` | no | | log_standard_ia_days | Number of days before moving logs to IA Storage | `number` | `30` | no | | private_ssh_port | Set the SSH port to use between the bastion and private instance | `number` | `22` | no | | public_ssh_port | Set the SSH port to use from desktop to the bastion | `number` | `22` | no | | region | AWS Region | `any` | n/a | yes | | tags | A mapping of tags to assign | `map(string)` | `{}` | no | | vpc_id | VPC id were we'll deploy the bastion | `any` | n/a | yes |

Outputs

Name	Description
bastion_host_security_group	The security group ID of the Bastion Host
bucket_kms_key_alias	The alias of the buckets kms key
bucket_kms_key_arn	The arn of the buckets kms key
bucket_name	The name of the bucket where logs are sent
elb_ip	The ELB DNS Name for the Bastion Host instances
private_instances_security_group	The security group ID of the the private instances that allow Bastion SSH ingress

Known issues

Tags are not applied to the instances generated by the auto scaling group do to known terraform issue : terraform-providers/terraform-provider-aws#290

Change of disk encryption isn't propagate immediately. Change have to trigger manually from AWS CLI: Auto Scaling Groups -> Instance refresh . Keep in mind all data from instance will be lost in case there are temporary or custom data.

Authors

Module managed by Guimove.

License

Apache 2 Licensed. See LICENSE for full details.