Using Terraform

Aug 2017

Now that we have our simple Terraform script, we can start to use Terraform. First we need to test the script for any errors, then we can launch our EC2 Instance by running the script.

Terraform Plan

The terraform plan command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files. The plan can be saved using -out, and then provided to terraform apply to ensure only the pre-planned actions are executed.

As I only have one configuration file, I can just use the command:

```
  terraform plan
```

For my configuration file, I get the following output.

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

The Terraform execution plan has been generated and is shown below.
Resources are shown in alphabetical order for quick scanning. Green resources
will be created (or destroyed and then created if an existing resource
exists), yellow resources are being changed in-place, and red resources
will be destroyed. Cyan entries are data sources to be read.

Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.

+ aws_instance.example
    ami:                          "ami-40a8bf24"
    associate_public_ip_address:  "<computed>"
    availability_zone:            "<computed>"
    ebs_block_device.#:           "<computed>"
    ephemeral_block_device.#:     "<computed>"
    instance_state:               "<computed>"
    instance_type:                "t2.micro"
    ipv6_address_count:           "<computed>"
    ipv6_addresses.#:             "<computed>"
    key_name:                     "TestSvr"
    network_interface.#:          "<computed>"
    network_interface_id:         "<computed>"
    placement_group:              "<computed>"
    primary_network_interface_id: "<computed>"
    private_dns:                  "<computed>"
    private_ip:                   "<computed>"
    public_dns:                   "<computed>"
    public_ip:                    "<computed>"
    root_block_device.#:          "<computed>"
    security_groups.#:            "1"
    security_groups.2525401260:   "launch-wizard-1"
    source_dest_check:            "true"
    subnet_id:                    "<computed>"
    tags.%:                       "1"
    tags.Name:                    "terraform-instance"
    tenancy:                      "<computed>"
    volume_tags.%:                "<computed>"
```

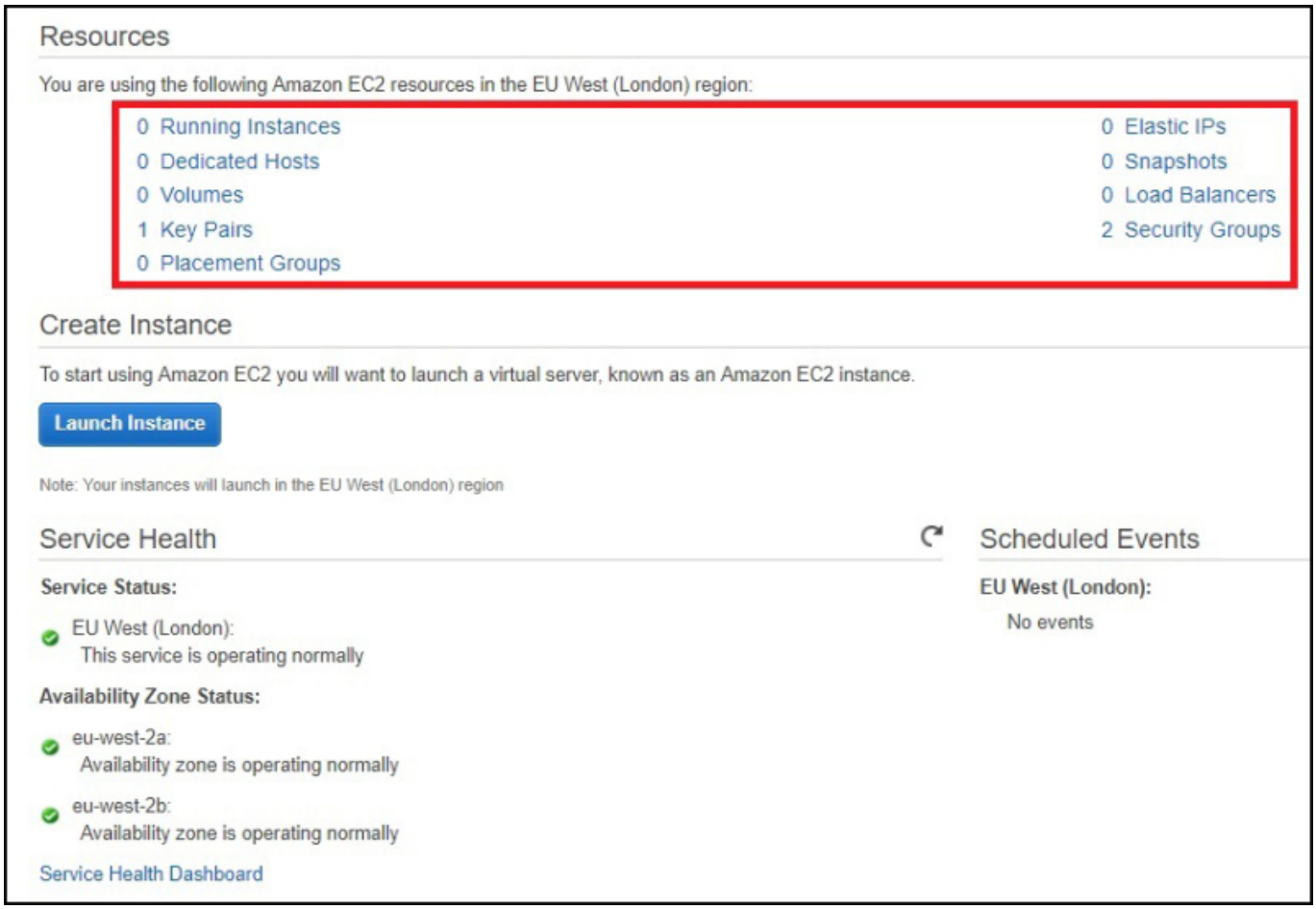
```
    vpc_security_group_ids.#:     "<computed>"


Plan: 1 to add, 0 to change, 0 to destroy.
```

This is the type of output you will get if your script file does not contain any errors.

Terraform Apply

Before starting the Terraform apply option, I suggest you log in to your AWS account so that you can monitor the progress of the script.

Here is my AWS account with no configured EC2 Instances.



In the above example we can see that there is nothing configured on this AWS account. There is a default security group, but this can not be deleted, a new security group will be configured with our Terraform Script, so this should change from a (1) to a (2).