05

Vagrant e AWS #1

Esse exercício é independente do sistema operacional e foi testado no Windows 10, Linux Ubuntu e MacOS.

Checklist AWS

Só para garantir, você já:

- baixou o projeto atualizado que usa Ubuntu 18.04 e MariaDB
- anotou a **identificação do AMI**.
- sabe da **chave de acesso** (`AWSAccessKeyId` e `AWSSecretKey`).
- guardou o **arquivo PEM**.
- configurou as regras de segurança (**Security Groups**) no EC2.

Obs: Vamos testar primeiro a criação da maquina virtual, sem upload dos arquivos. Uma vez funcionando vamos tentar sincronizar os arquivos e o provisionamento.

Plugin do AWS para Vagrant

No seu computador, na linha de comando instale o plugin do `aws` para o `vagrant` :

```
vagrant plugin install vagrant-aws
```

Isso pode levar alguns minutos.

No Windows, se você recebe um erro acusando a falta do `libxml2` , tente instalar antes:

```
vagrant plugin install fog-aws
vagrant plugin install --plugin-version 1.0.1 fog-ovirt
vagrant plugin expunge --force --reinstall
```

Dummy Box

O Vagrant sempre se baseia em um box que possui a imagem e outras configurações. No nosso caso, quem possui a imagem é a AWS, mas mesmo assim precisamos de um box de modelo ou simulação. Para o Vagrant no AWS funcionar devemos instalar o *dummy box*:

```
vagrant box add --force dummy https://github.com/mitchellh/vagrant-aws/raw/master/dummy.box
```

Provider AWS

No nosso `Vagrantfile` vamos configurar o básico de todas as nossas máquinas da Amazon. Primeiro a box que vamos utilizar mais um provedor, o `aws` . Logo abaixo da configuração do provedor `virtualbox` adicione :

```
    config.vm.provider "aws"
```

Definimos o provedor que foi instalado com o plugin com os dados básicos de todas as máquinas que criaremos na Amazon. Agora você vai precisar os dados da AWS, como a identificação da AMI, arquivo PEM, nome da *security group* e chave de acesso. Logo abaixo da definição do provedor adicione"

```
config.vm.provider :aws do |aws, override|

    #dados do access key
    aws.access_key_id = "sua chave aws access id"
    aws.secret_access_key = "sua chave secret aws access key"

    #a identificacao da AMI, abaixo é Ubuntu 18.04
    aws.ami = "ami-0ac019f4fcb7cb7e6"

    #nome do security group
    aws.security_groups = ['devops-vagrant']

    #nome do arquivo pem
    aws.keypair_name = "devops-key"

    #tipo de instância e região
    aws.instance_type = "t2.micro"
        aws.region = "sa-east-1"

    #nome do usuario, no caso do Ubuntu é ubuntu
    override.ssh.username = "ubuntu"

    #caminho e nome do arquivo pem
    override.ssh.private_key_path = "devops-key.pem"


end
```

Novo ambiente

Ainda no `Vagrantfile` , crie agora um novo ambiente (`aws_web`) para a VM no EC2. Dentro dessa configuração, defina apenas o nome do box (aquele *dummy*), ainda sem sincronização de pasta (`synced_folder`) e o nome de nossa máquina caso o provedor seja o *aws* pelo tag:

```
    #novo ambiente aws_web, ainda sem puppet
    config.vm.define :aws_web do |aws_web_config|
      aws_web_config.vm.box = "dummy"
      aws_web_config.vm.synced_folder '.', '/vagrant', type: "rsync", disabled: true
      aws_web_config.vm.provider :aws do |aws|
          aws.tags = { 'Name' => 'MusicJungle (vagrant)'}
      end
    end
```

Obs: Novamente, vamos primeiro criar apenas a maquina com Vagrant na AWS e depois testar o provisionamento com Puppet.

Levantando a maquina

Para subir a maquina virtual no AWS EC2, rode `vagrant up` especificando o ambiente e o provedor `aws`:

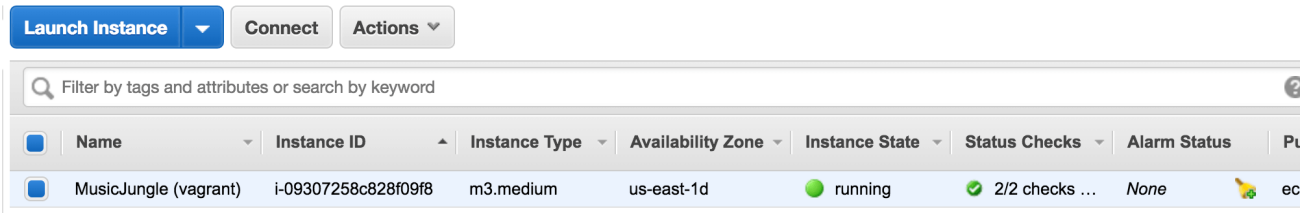
```
vagrant up aws_web --provider=aws
```

A execução no AWS vai demorar mais do que a execução local:

```
$ vagrant up aws_web --provider=aws

Bringing machine 'aws_web' up with 'aws' provider...
==> aws_web: Warning! The AWS provider doesn't support any of the Vagrant
==> aws_web: high-level network configurations (`config.vm.network`). They
==> aws_web: will be silently ignored.
==> aws_web: Launching an instance with the following settings...
==> aws_web:  -- Type: m3.medium
==> aws_web:  -- AMI: ami-0ac019f4fcb7cb7e6
==> aws_web:  -- Region: us-east-1
==> aws_web:  -- Keypair: key-devops-vagrant
==> aws_web:  -- Security Groups: ["devops-vagrant"]
==> aws_web:  -- Block Device Mapping: []
==> aws_web:  -- Terminate On Shutdown: false
==> aws_web:  -- Monitoring: false
==> aws_web:  -- EBS optimized: false
==> aws_web:  -- Source Destination check:
==> aws_web:  -- Assigning a public IP address in a VPC: false
==> aws_web:  -- VPC tenancy specification: default
==> aws_web: Waiting for instance to become "ready"...
==> aws_web: Waiting for SSH to become available...
==> aws_web: Machine is booted and ready for use!
```

Confira também o status da VM no seu EC2 Dashboard:



Teste SSH

Obs: Alguns provedores não liberam a porta 22 (como a Net Virtua). Isso faz que o Vagrant levanta a maquina no AWS mas não consegue se conectar.

Se conseguiu levantar a VM no EC2, tente usar SSH para se conectar com a VM:

```
$ vagrant ssh aws_web

Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-1021-aws x86_64)

 * Documentation:  https://help.ubuntu.com
```

```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Nov  4 11:42:39 UTC 2018

  System load:  0.08              Processes:           89
  Usage of /:   13.4% of 7.69GB   Users logged in:     0
  Memory usage: 3%                IP address for eth0: 10.154.106.221
  Swap usage:   0%


  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.


To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-154-106-221:~$ exit
```

Após execução

Para terminar a maquina no AWS, você pode usar o Vagrant:

```
vagrant destroy -f aws_web
```

Lembre-se de após este capítulo verificar o EC2 Dashboard e dar um `terminate` em sua máquina e apagar qualquer traço de sua conta caso não queira pagar mais nada além do gasto até aqui com a Amazon AWS.