

Vagrant e AWS #2

Adicionando Puppet

Vamos adicionar o Puppet no ambiente `aws_web`. Para instalar o Puppet usaremos o shell provisioner. Para tal existe um arquivo `bootstrap.sh` na pasta `manifests` que executa nada mais do que um `apt-get`:

```
apt-get update && apt-get -y install puppet && puppet module install puppetlabs-apache
```

Além disso, precisamos definir o mecanismo de sincronização dos arquivos locais para a EC2. No exemplo temos `rsync` definido (para o Windows existe SMB).

Segue a definição completa do ambiente `aws_web`. Faça as alterações no seu `Vagrantfile`:

```
#novo ambiente aws_web, agora com puppet
config.vm.define :aws_web do |aws_web_config|

  aws_web_config.vm.box = "dummy"

  #usando rsync
  aws_web_config.vm.synced_folder '.', '/vagrant', type: "rsync"

  aws_web_config.vm.provider :aws do |aws|
    aws.tags = { 'Name' => 'MusicJungle (vagrant)'}
  end

  ## shell e puppet provisioner
  aws_web_config.vm.provision "shell", path: "manifests/bootstrap.sh"
  aws_web_config.vm.provision "puppet" do |puppet|
    puppet.manifest_file = "web.pp"
    #usando rsync
    puppet.synced_folder_type = 'rsync'
  end
end
```

Salve o arquivo.

Testando tudo

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

Novamente, a execução vai demorar um pouco e, além da VM, vamos também provisionar todo o ambiente. Segue uma possível saída.

```
$ 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!
==> aws_web: Rsyncing folder: /Users/nico/Downloads/168-aula6/ => /vagrant

.....

==> aws_web: Running provisioner: shell...

.....

    aws_web: Puppet installed!
==> aws_web: Running provisioner: puppet...
==> aws_web: Running Puppet with web.pp...
==> aws_web: Notice: Compiled catalog for ip-10-153-182-168.ec2.internal in environment product:
==> aws_web: Notice: /Stage[main]/Main/Exec[apt-update]/returns: executed successfully
==> aws_web: Notice: /Stage[main]/Main/Package[openjdk-8-jre]/ensure: created
==> aws_web: Notice: /Stage[main]/Main/Package[tomcat8]/ensure: created
==> aws_web: Notice: /Stage[main]/Main/Package[mariadb-server]/ensure: created
==> aws_web: Notice: /Stage[main]/Main/Exec[musicjungle]/returns: executed successfully
==> aws_web: Notice: /Stage[main]/Main/Exec[user_mariadb]/returns: executed successfully
==> aws_web: Notice: /Stage[main]/Main/File[/var/lib/tomcat8/webapps/vraptor-musicjungle.war]/e
==> aws_web: Notice: /Stage[main]/Main/File_line[production]/ensure: created
==> aws_web: Notice: /Stage[main]/Main/Service[tomcat8]: Triggered 'refresh' from 2 events
==> aws_web: Notice: Applied catalog in 113.38 seconds
```

Testando pelo navegador

Se tudo deu certo, podemos testar a aplicação pelo navegador. Para tal, você precisa descobrir a IP da máquina. Vá o EC2 Dashboard no item *Instances*. Selecione a VM em execução e procure o PUBLIC DNS:

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

Scheduled Instances

Capacity Reservations

IMAGES

AMIs

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

1 to 2 of

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
☐	MusicJungle (vagrant)	i-09307258c828f09f8	m3.medium	us-east-1d	terminated		None	
☒	MusicJungle (vagrant)	i-0d52f51107bf28a3c	m3.medium	us-east-1a	running	2/2 checks ...	None	ec2-54-146-216-83.co...

Instance: i-0d52f51107bf28a3c (MusicJungle (vagrant))

Public DNS: ec2-54-146-216-83.compute-1.amazonaws.com

Description

Status Checks

Monitoring

Tags

Instance ID

i-0d52f51107bf28a3c

Public DNS (IPv4)

ec2-54-146-216-83.compute-1.amazonaws.com

Instance state

running

IPv4 Public IP

54.146.216.83

Depois use esse nome para testar a aplicação:

<http://seu-endereco.amazonaws.com:8080/vraptor-musicjungle> (<http://seu-endereco.amazonaws.com:8080/vraptor-musicjungle>).

Liberando recurso

Depois do seu teste, não esqueça de liberar a VM na AWS:

```
vagrant destroy -f aws_web
```

Também verifique o Dashboard no EC2 para ter certeza que nenhuma maquina está rodando:

☐	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
☐	MusicJungle (vagrant)	i-09307258c828f09f8	m3.medium	us-east-1d	terminated	
☐	MusicJungle (vagrant)	i-0d52f51107bf28a3c	m3.medium	us-east-1a	terminated	