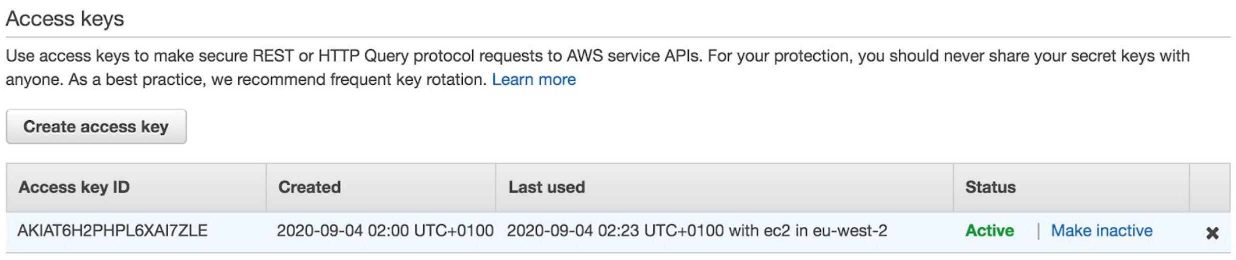
Bink Automation Test

Automation Test

Using a tool of your choice such as Bash, Chef, Ansible or similar automate the installation of a Web Server serving up some static content. Bonus points for the following:

- Using Containers as part of your automation.
- Providing a Vagrantfile to allow us to test your code by simply running `vagrant up`.

1. Installed Vagrant
2. Installed Virtualbox
3. Created user in AWS and downloaded credentials
4. Used Vagrant AWS provider to spin up EC2 instance in AWS



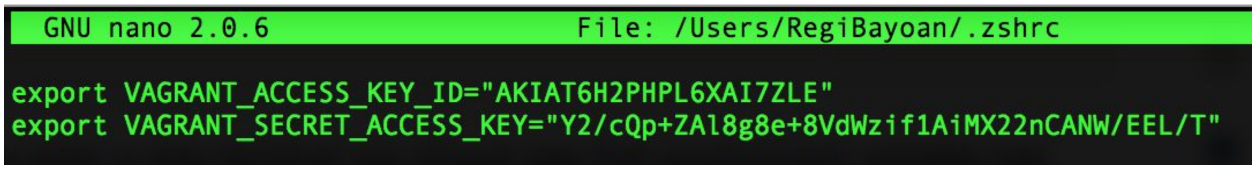
Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. Learn more

Create access key

Access key ID	Created	Last used	Status	
AKIAT6H2PHPL6XAI7ZLE	2020-09-04 02:00 UTC+0100	2020-09-04 02:23 UTC+0100 with ec2 in eu-west-2	Active | Make inactive	✖

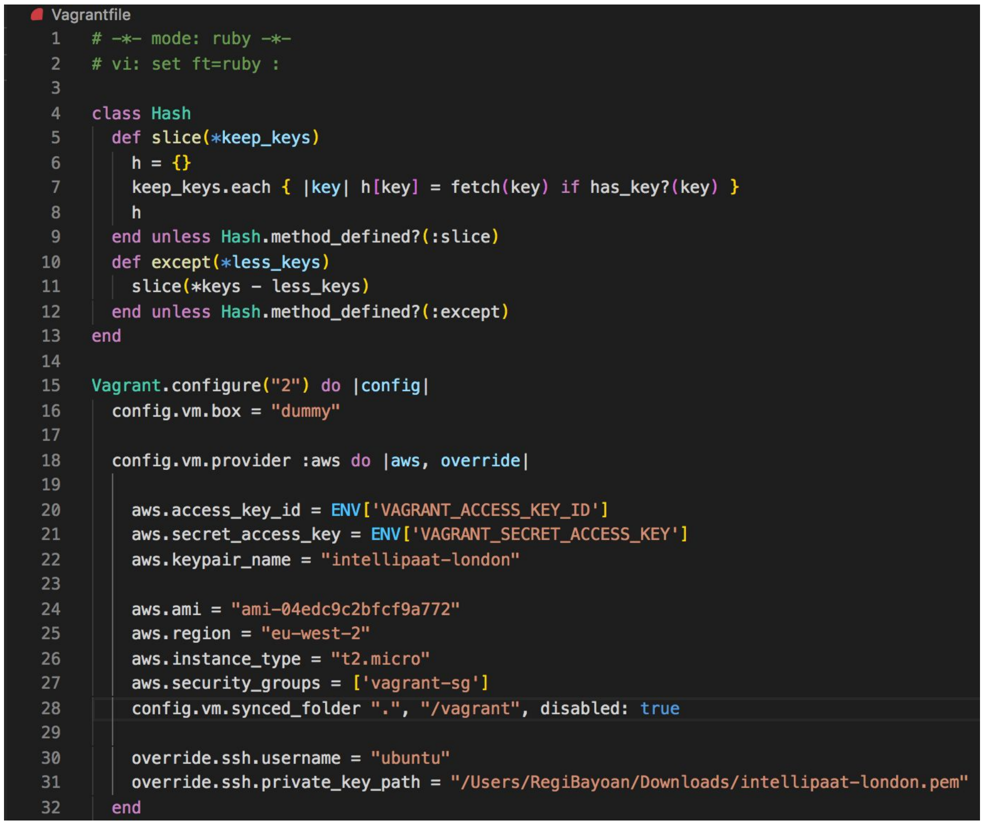
5. Hide AWS user credentials in environment variables. Make sure this AWS user is deleted once exposed



```
  GNU nano 2.0.6                    File: /Users/RegiBayoan/.zshrc

export VAGRANT_ACCESS_KEY_ID="AKIAT6H2PHPL6XAI7ZLE"
export VAGRANT_SECRET_ACCESS_KEY="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"
```

Vagrantfile code used for creating an EC2 instance



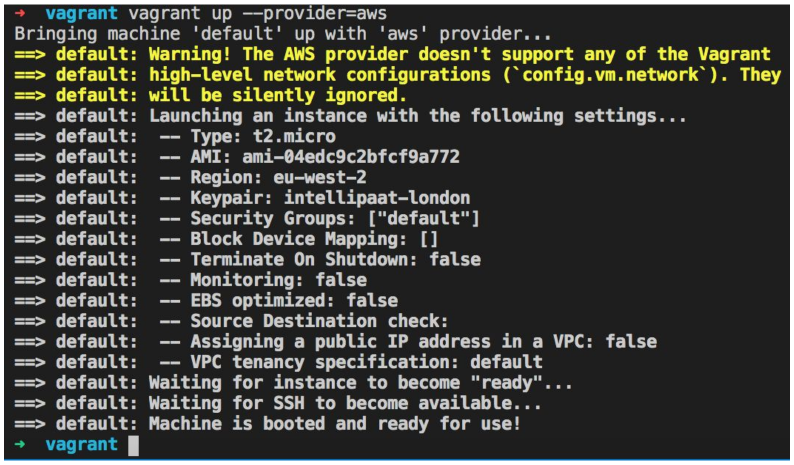
Vagrantfile

```ruby
1   # -*- mode: ruby -*-
2   # vi: set ft=ruby :
3
4   class Hash
5     def slice(*keep_keys)
6       h = {}
7       keep_keys.each { |key| h[key] = fetch(key) if has_key?(key) }
8       h
9     end unless Hash.method_defined?(:slice)
10    def except(*less_keys)
11      slice(*keys - less_keys)
12    end unless Hash.method_defined?(:except)
13  end
14
15  Vagrant.configure("2") do |config|
16    config.vm.box = "dummy"
17
18    config.vm.provider :aws do |aws, override|
19
20      aws.access_key_id = ENV['VAGRANT_ACCESS_KEY_ID']
21      aws.secret_access_key = ENV['VAGRANT_SECRET_ACCESS_KEY']
22      aws.keypair_name = "intellipaat-london"
23
24      aws.ami = "ami-04edc9c2bfcf9a772"
25      aws.region = "eu-west-2"
26      aws.instance_type = "t2.micro"
27      aws.security_groups = ['vagrant-sg']
28      config.vm.synced_folder ".", "/vagrant", disabled: true
29
30      override.ssh.username = "ubuntu"
31      override.ssh.private_key_path = "/Users/RegiBayoan/Downloads/intellipaat-london.pem"
32    end
```

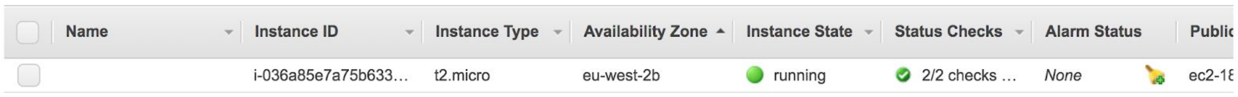
Notes:

- **class Hash** was taken from the internet as a workaround for **MethodNotFound** error
- Used environment variables to hide sensitive aws credentials
- Created a custom security group to allow ALL TCP traffic **(vagrant-sg)**
- **Line 28** was added to bypass SMB credentials prompt when doing **vagrant up**

6. **vagrant up --provider=aws** to execute



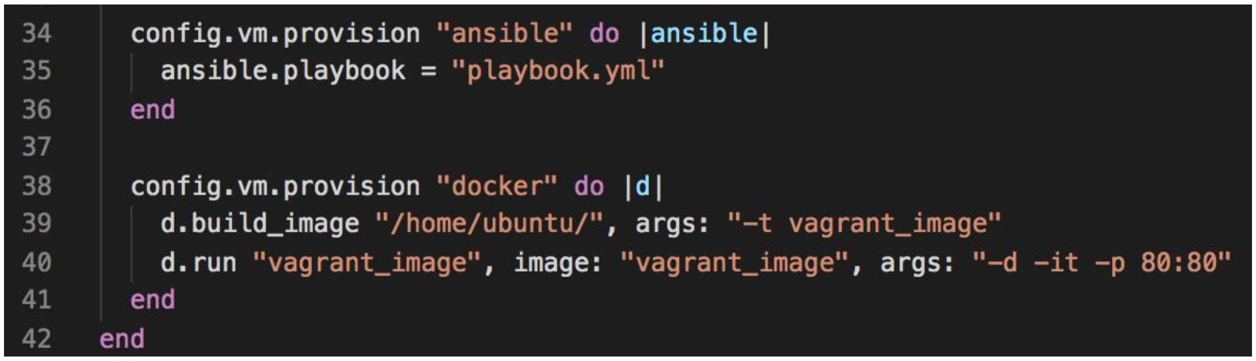
```
➜  vagrant vagrant up --provider=aws
Bringing machine 'default' up with 'aws' provider...
==> default: Warning! The AWS provider doesn't support any of the Vagrant
==> default: high-level network configurations (`config.vm.network`). They
==> default: will be silently ignored.
==> default: Launching an instance with the following settings...
==> default:  -- Type: t2.micro
==> default:  -- AMI: ami-04edc9c2bfcf9a772
==> default:  -- Region: eu-west-2
==> default:  -- Keypair: intellipaat-london
==> default:  -- Security Groups: ["default"]
==> default:  -- Block Device Mapping: []
==> default:  -- Terminate On Shutdown: false
==> default:  -- Monitoring: false
==> default:  -- EBS optimized: false
==> default:  -- Source Destination check:
==> default:  -- Assigning a public IP address in a VPC: false
==> default:  -- VPC tenancy specification: default
==> default: Waiting for instance to become "ready"...
==> default: Waiting for SSH to become available...
==> default: Machine is booted and ready for use!
➜  vagrant
```



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public
		i-036a85e7a75b633...	t2.micro	eu-west-2b	● running	✓ 2/2 checks ...	None	ec2-18

7. Ansible provisioner was used to transfer static content to EC2 instance
8. Docker provisioner was used for creating containers

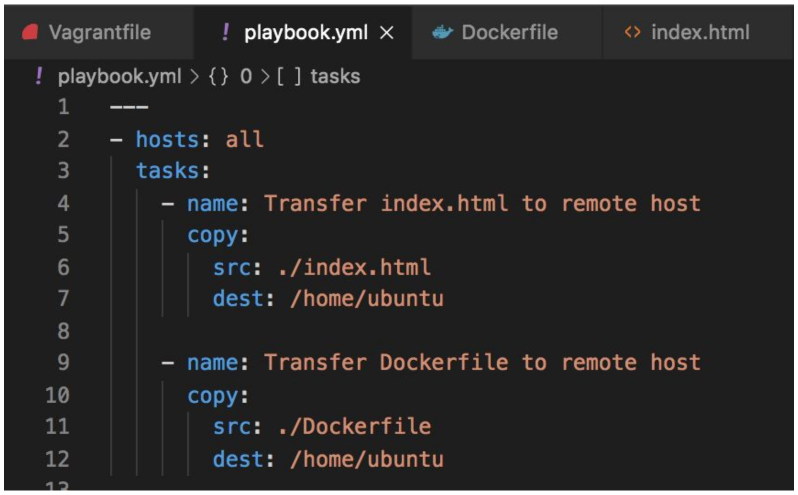
Vagrantfile cont'd



```
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "playbook.yml"
  end

  config.vm.provision "docker" do |d|
    d.build_image "/home/ubuntu/", args: "-t vagrant_image"
    d.run "vagrant_image", image: "vagrant_image", args: "-d -it -p 80:80"
  end
end
```

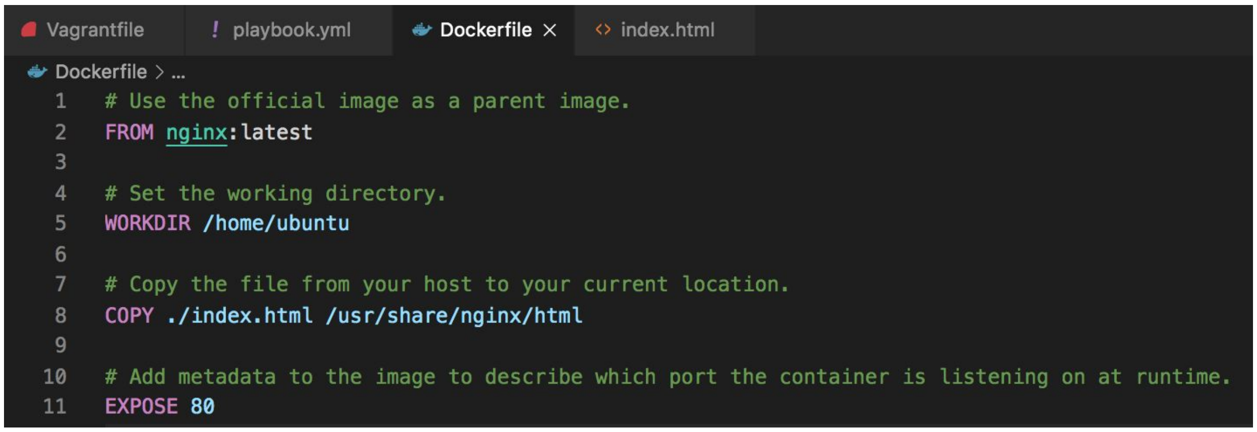
playbook.yml code



```
---
- hosts: all
  tasks:
    - name: Transfer index.html to remote host
      copy:
        src: ./index.html
        dest: /home/ubuntu

    - name: Transfer Dockerfile to remote host
      copy:
        src: ./Dockerfile
        dest: /home/ubuntu
```

Dockerfile code



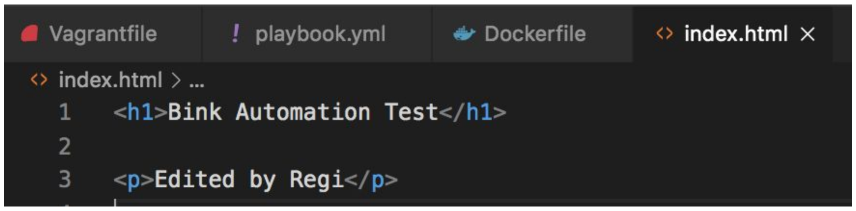
```
# Use the official image as a parent image.
FROM nginx:latest

# Set the working directory.
WORKDIR /home/ubuntu

# Copy the file from your host to your current location.
COPY ./index.html /usr/share/nginx/html

# Add metadata to the image to describe which port the container is listening on at runtime.
EXPOSE 80
```

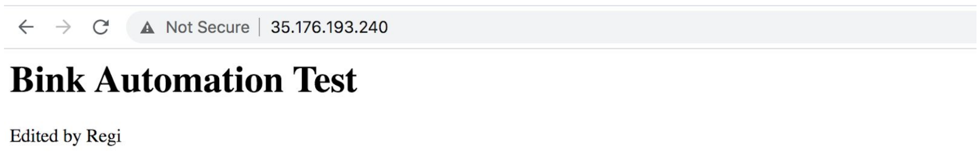
index.html code



```
<h1>Bink Automation Test</h1>

<p>Edited by Regi</p>
```

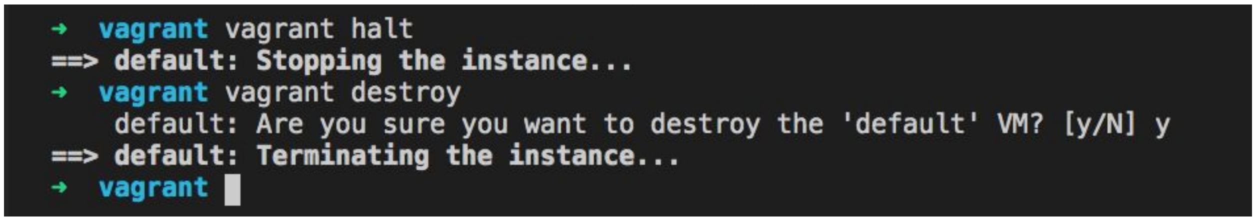
Browser



Bink Automation Test

Edited by Regi

9. Lastly, cleaned up Vagrant environment



```
➜  vagrant vagrant halt
==> default: Stopping the instance...
➜  vagrant vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Terminating the instance...
➜  vagrant 
```