Packer

Diego Pacheco

About Me

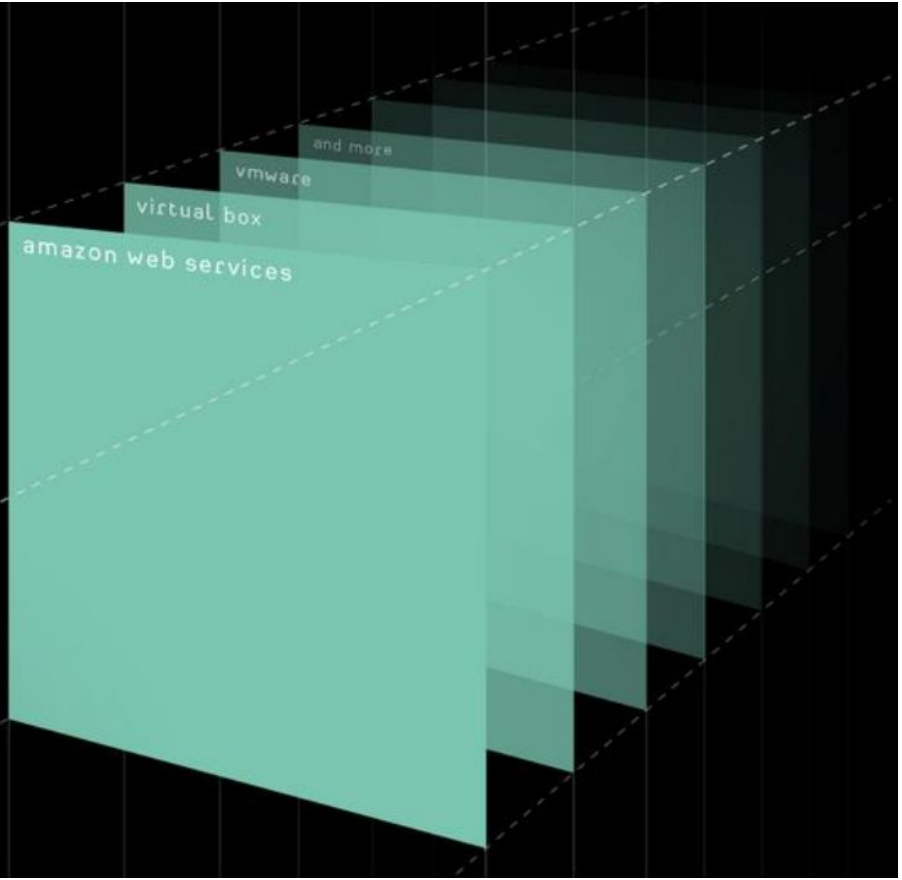
- Cat's Father
- Principal Software Architect
- Agile Coach
- SOA/Microservices Expert
- DevOps Practitioner
- Speaker
- Author

diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

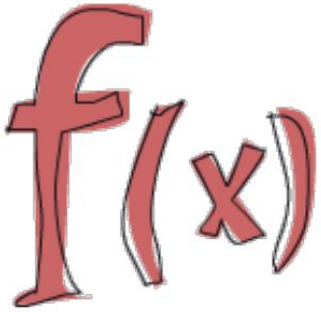
Packer is a tool for creating identical machine images for multiple platforms from a single source configuration.



Packer don’t replcace Automation Engines

Image Bake

Immutable Infrastructure



im·mu·ta·ble
/iˈmyo͞otəbəl/

Adjective
Unchanging over time or unable to be changed: "an immutable fact".

Synonyms
invariable - unalterable - constant - changeless

docker

Parallels®

Google Compute Engine

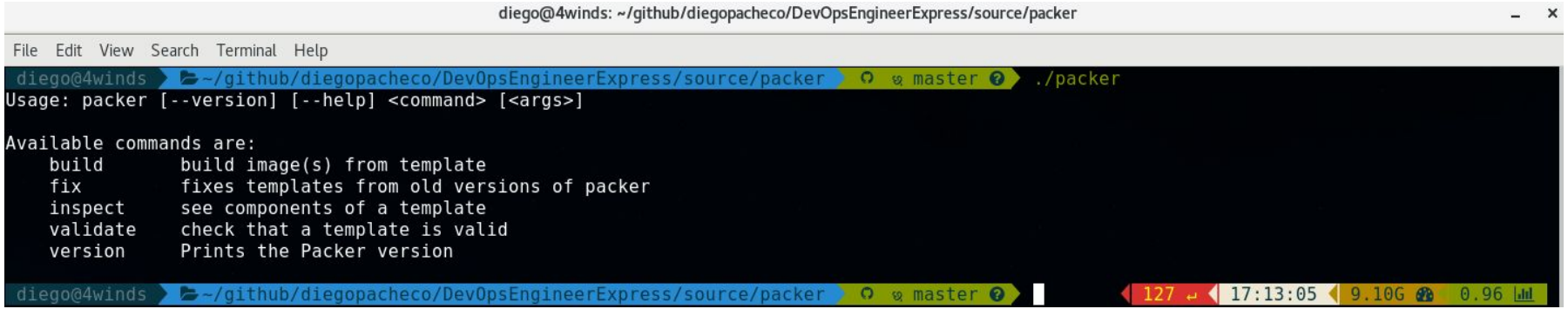
openstack™
CLOUD SOFTWARE

Xen™

Install Packer

install-packher.sh

```
1  #!/bin/bash
2
3  wget https://releases.hashicorp.com/packer/1.3.3/packer_1.3.3_linux_amd64.zip
4  unzip packer_1.3.3_linux_amd64.zip
5  rm -rf packer_1.3.3_linux_amd64.zip
6  ./packer
```



Baking

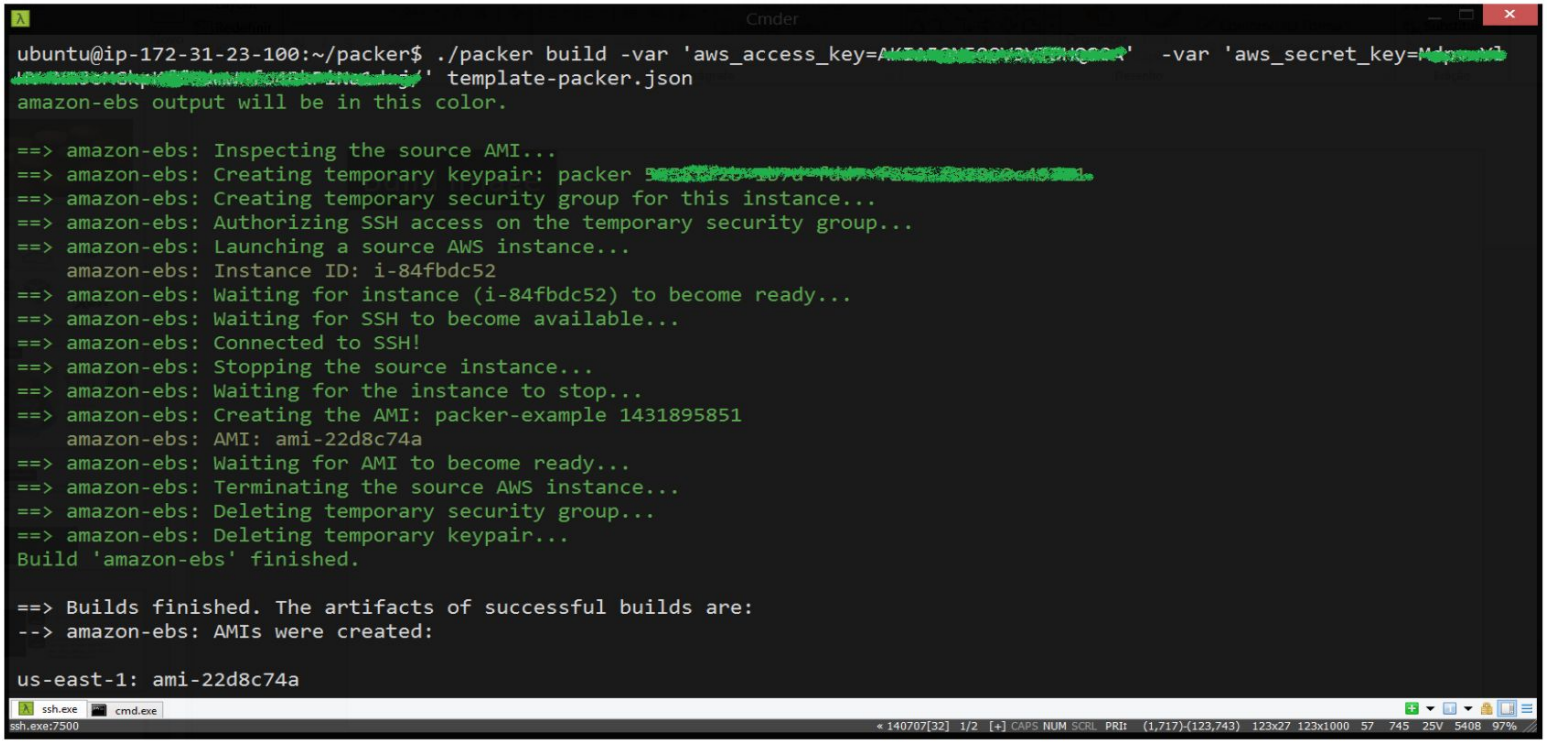
```
{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": ""
  },
  "builders": [{
    "type": "amazon-ebs",
    "access_key": "{{user `aws_access_key`}}",
    "secret_key": "{{user `aws_secret_key`}}",
    "region": "us-east-1",
    "source_ami": "ami-9eaa1cf6",
    "instance_type": "t2.micro",
    "ssh_username": "ubuntu",
    "ami_name": "packer-example {{timestamp}}"
  }]
}
```

template-packer.json


```
~$ ./packer validate template-packer.json
```


```
~$ ./packer build \
-var 'aws_access_key=KEY' \
-var 'aws_secret_key=SECRET' \
template-packer.json
```

Baking


Cmder
ubuntu@ip-172-31-23-100:~/packer$ ./packer build -var 'aws_access_key=[redacted]' -var 'aws_secret_key=[redacted]' template-packer.json
amazon-ebs output will be in this color.
==> amazon-ebs: Inspecting the source AMI...
==> amazon-ebs: Creating temporary keypair: packer [redacted]
==> amazon-ebs: Creating temporary security group for this instance...
==> amazon-ebs: Authorizing SSH access on the temporary security group...
==> amazon-ebs: Launching a source AWS instance...
amazon-ebs: Instance ID: i-84fbdc52
==> amazon-ebs: Waiting for instance (i-84fbdc52) to become ready...
==> amazon-ebs: Waiting for SSH to become available...
==> amazon-ebs: Connected to SSH!
==> amazon-ebs: Stopping the source instance...
==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating the AMI: packer-example 1431895851
amazon-ebs: AMI: ami-22d8c74a
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished.
==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-east-1: ami-22d8c74a
ssh.exe
cmd.exe
ssh.exe:7500

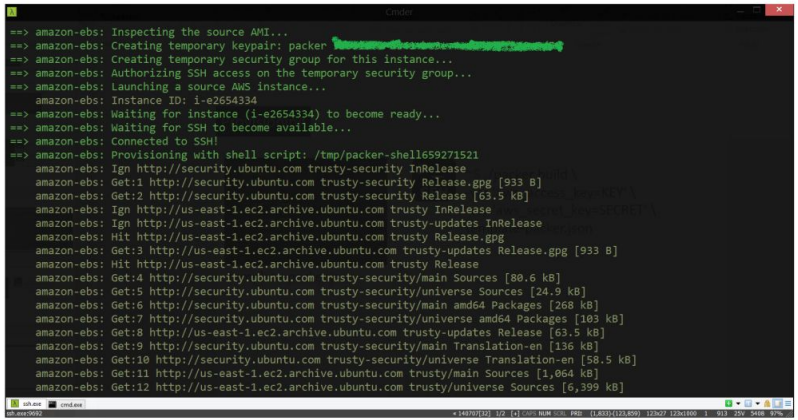
Provision

Provisioning

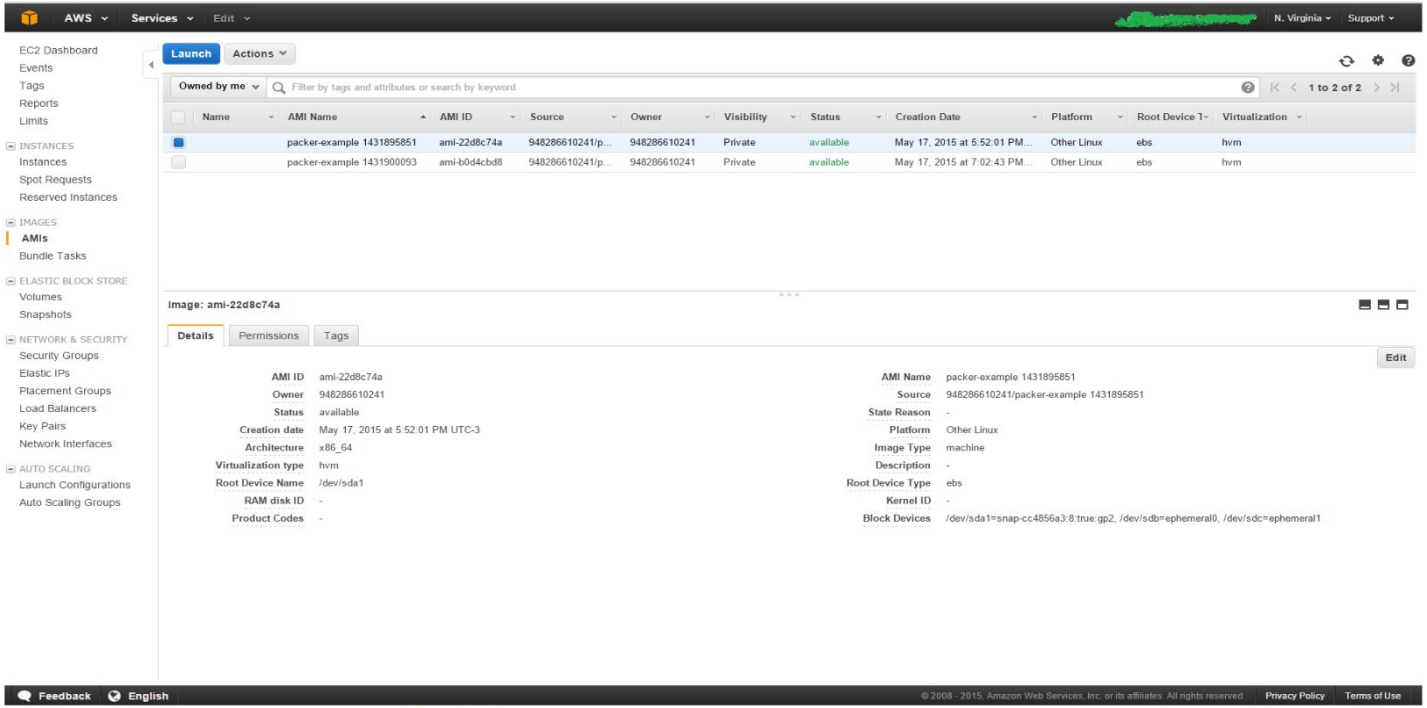
```
{
  "variables": ["..."],
  "builders": ["..."],

  "provisioners": [{
    "type": "shell",
    "inline": [
      "sleep 30",
      "sudo apt-get update",
      "sudo apt-get install -y redis-server"
    ]
  }]
}
```

template-packer.json

```
~$ ./packer build \
-var 'aws_access_key=KEY' \
-var 'aws_secret_key=SECRET' \
template-packer.json
```

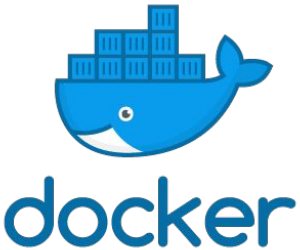


Result on AWS console



https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Images:sort=name

Provisioning Support



- ❏ Great variety of options
- ❏ Install software after VM created
- ❏ Great to test provisioning scripts
- ❏ You can have same provisioning as you have in production
- ❏ More information on docs

https://www.packer.io/docs/provisioners/index.html

It’s Possible to bake docker containers

```
{
  "type": "docker",
  "image": "ubuntu",
  "commit": true,
  "changes": [
    "USER www-data",
    "WORKDIR /var/www",
    "ENV HOSTNAME www.example.com",
    "VOLUME /test1 /test2",
    "EXPOSE 80 443",
    "LABEL version=1.0",
    "ONBUILD RUN date",
    "CMD [\"nginx\", \"-g\", \"daemon off;\"]",
    "ENTRYPOINT /var/www/start.sh"
  ]
}
```

https://www.packer.io/docs/builders/docker.html

Post-Processors

Post-processors run after the image is built by the builder and provisioned by the provisioner(s). Post-processors are optional, and they can be used to upload artifacts, re-package, or more. For more information about post-processors,

- Alicloud Import
- Amazon Import
- Artifice
- Compress
- Checksum
- Docker Import
- Docker Push
- Docker Save
- Docker Tag
- Google Compute Export
- Google Compute Import
- Manifest
- Shell (Local)
- Vagrant
- Vagrant Cloud
- vSphere
- vSphere Template

https://www.packer.io/docs/post-processors/index.html

Packer

Diego Pacheco