



HashiCorp

Packer

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.



amazon web services
virtual box
vmware
and more

Packer don't replace Automation Engines



Image Bake



Immutable Infrastructure



$f(x)$



im·mu·ta·ble

/iˈmyootəbəl/ ⓘ

Adjective

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

Synonyms

invariable - unalterable - constant - changeless



Google Compute Engine



Install Packer

install-packher.sh x

```
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
```

diego@4winds: ~/github/diegopacheco/DevOpsEngineerExpress/source/packer

File Edit View Search Terminal Help

diego@4winds > ~/github/diegopacheco/DevOpsEngineerExpress/source/packer master ? ./packer

Usage: packer [--version] [--help] <command> [<args>]

Available commands are:

build	build image(s) from template
fix	fixes templates from old versions of packer
inspect	see components of a template
validate	check that a template is valid
version	Prints the Packer version

diego@4winds > ~/github/diegopacheco/DevOpsEngineerExpress/source/packer master ?

127 17:13:05 9.10G 0.96

Baking

```
{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": ""
  },
  "builders": [{
    "type": "amazon-efs",
    "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

```
ubuntu@ip-172-31-23-100:~/packer$ ./packer build -var 'aws_access_key=AKIA5ON5Q2V3M7H1Q224' -var 'aws_secret_key=Mlp...Ml...Ml...' template-packer.json
amazon-eks output will be in this color.

==> amazon-eks: Inspecting the source AMI...
==> amazon-eks: Creating temporary keypair: packer 9551a22b-8b7d-4d47-9f52-232926c40211
==> amazon-eks: Creating temporary security group for this instance...
==> amazon-eks: Authorizing SSH access on the temporary security group...
==> amazon-eks: Launching a source AWS instance...
    amazon-eks: Instance ID: i-84fdbc52
==> amazon-eks: Waiting for instance (i-84fdbc52) to become ready...
==> amazon-eks: Waiting for SSH to become available...
==> amazon-eks: Connected to SSH!
==> amazon-eks: Stopping the source instance...
==> amazon-eks: Waiting for the instance to stop...
==> amazon-eks: Creating the AMI: packer-example 1431895851
    amazon-eks: AMI: ami-22d8c74a
==> amazon-eks: Waiting for AMI to become ready...
==> amazon-eks: Terminating the source AWS instance...
==> amazon-eks: Deleting temporary security group...
==> amazon-eks: Deleting temporary keypair...
Build 'amazon-eks' finished.

==> Builds finished. The artifacts of successful builds are:
--> amazon-eks: AMIs were created:

us-east-1: ami-22d8c74a
```

ssh.exe cmd.exe
ssh.exe:7500 140707[32] 1/2 [*] CAPS_NUM SCRL PRI: (1,717)-(123,743) 123x27 123x1000 57 745 25V 5408 97%

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
```

```
=> amazon-ebis: Inspecting the source AMI...
=> amazon-ebis: Creating temporary keypair: packer
=> amazon-ebis: Creating temporary security group for this instance...
=> amazon-ebis: Authorizing SSH access on the temporary security group...
=> amazon-ebis: Launching a source AWS Instance...
amazon-ebis: Instance ID: i-e2654334
=> amazon-ebis: Waiting for instance (i-e2654334) to become ready...
=> amazon-ebis: Connected to SSH!
=> amazon-ebis: Provisioning with shell script: /tmp/packer-shell1659271521
amazon-ebis: Ign http://security.ubuntu.com trusty-security InRelease
amazon-ebis: Get:1 http://security.ubuntu.com trusty-security Release.gpg [933 B]
amazon-ebis: Get:2 http://security.ubuntu.com trusty-security Release [63.5 kB]
amazon-ebis: Ign http://us-east-1.ec2.archive.ubuntu.com trusty InRelease
amazon-ebis: Ign http://us-east-1.ec2.archive.ubuntu.com trusty-updates InRelease
amazon-ebis: Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release.gpg
amazon-ebis: Get:3 http://us-east-1.ec2.archive.ubuntu.com trusty-updates Release.gpg [933 B]
amazon-ebis: Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release
amazon-ebis: Get:4 http://security.ubuntu.com trusty-security/main Sources [80.6 kB]
amazon-ebis: Get:5 http://security.ubuntu.com trusty-security/universe Sources [24.9 kB]
amazon-ebis: Get:6 http://security.ubuntu.com trusty-security/main amd64 Packages [268 kB]
amazon-ebis: Get:7 http://security.ubuntu.com trusty-security/universe amd64 Packages [103 kB]
amazon-ebis: Get:8 http://us-east-1.ec2.archive.ubuntu.com trusty-updates Release [63.5 kB]
amazon-ebis: Get:9 http://security.ubuntu.com trusty-security/main Translation-en [136 kB]
amazon-ebis: Get:10 http://security.ubuntu.com trusty-security/universe Translation-en [58.5 kB]
amazon-ebis: Get:11 http://us-east-1.ec2.archive.ubuntu.com trusty/main Sources [1,064 kB]
amazon-ebis: Get:12 http://us-east-1.ec2.archive.ubuntu.com trusty/universe Sources [6,399 kB]
```

Result on AWS console

The screenshot displays the AWS Management Console interface. The left-hand navigation pane shows the 'INSTANCES' section expanded, with 'AMIs' selected. The main content area shows a list of AMIs under the 'Owned by me' filter. Two AMIs are listed: 'packer-example 1431895851' (AMI ID: ami-22d8c74a) and 'packer-example 1431900093' (AMI ID: ami-b0d4c-bd8). The first AMI is selected, and its details are shown below. The details are organized into two columns: 'Details' and 'Permissions'. The 'Details' column lists attributes such as AMI ID, Owner, Status, Creation date, Architecture, Virtualization type, Root Device Name, RAM disk ID, and Product Codes. The 'Permissions' column lists attributes such as AMI Name, Source, State Reason, Platform, Image Type, Description, Root Device Type, Kernel ID, and Block Devices. The 'Edit' button is visible in the top right corner of the details section.

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device 1	Virtualization
packer-example 1431895851	ami-22d8c74a	ami-22d8c74a	948286610241/p...	948286610241	Private	available	May 17, 2015 at 5:52:01 PM...	Other Linux	ebs	hvm
packer-example 1431900093	ami-b0d4c-bd8	ami-b0d4c-bd8	948286610241/p...	948286610241	Private	available	May 17, 2015 at 7:02:43 PM...	Other Linux	ebs	hvm

Image: ami-22d8c74a

Details		Permissions	
AMI ID	ami-22d8c74a	AMI Name	packer-example 1431895851
Owner	948286610241	Source	948286610241/packer-example 1431895851
Status	available	State Reason	-
Creation date	May 17, 2015 at 5:52:01 PM UTC-3	Platform	Other Linux
Architecture	x86_64	Image Type	machine
Virtualization type	hvm	Description	-
Root Device Name	/dev/sda1	Root Device Type	ebs
RAM disk ID	-	Kernel ID	-
Product Codes	-	Block Devices	/dev/sda1=snap-cc4856a3.8:true:gp2, /dev/sdb=ephemeral0, /dev/sdc=ephemeral1

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

Provisioning Support



ANSIBLE



docker



SALTSTACK



- ❑ 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>



HashiCorp

Packer

Packer

Diego Pacheco