



Using HCP Packer with Terraform Cloud



Agenda

Packer OSS & HCP Packer Overview

01

Creating a Golden Image

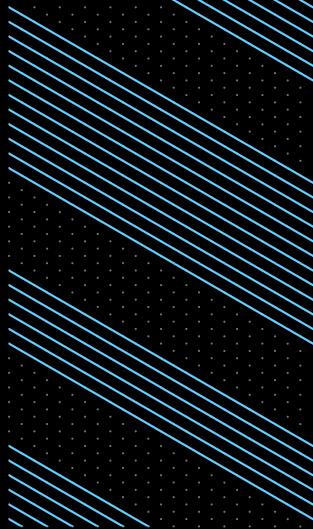
02

Unified Workflow with Terraform Cloud

03



01



Packer OSS & HCP Packer Overview



The standard for infrastructure automation to **build**, **govern** and **manage** any image for any cloud

Images as Code

Customization

Ansible

Bash

Powershell

...

Cloud

AWS

Azure

GCP

...

Private DC

VMware

VirtualBox

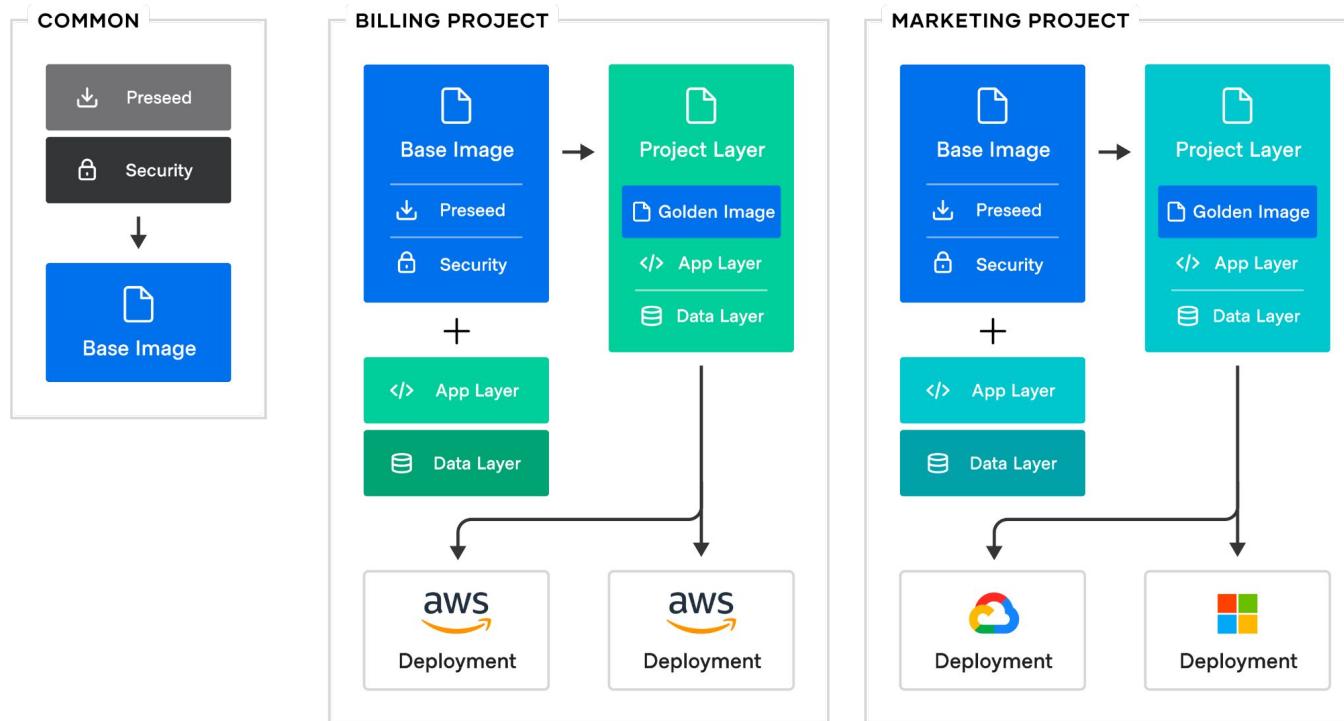
Vagrant

...



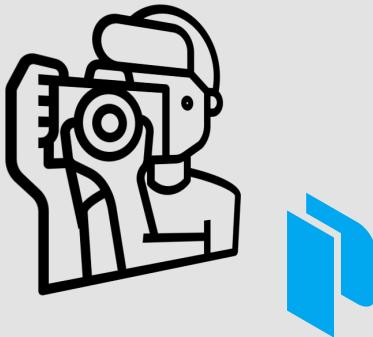
What is HCP Packer?

Codification = Standardization and Automation

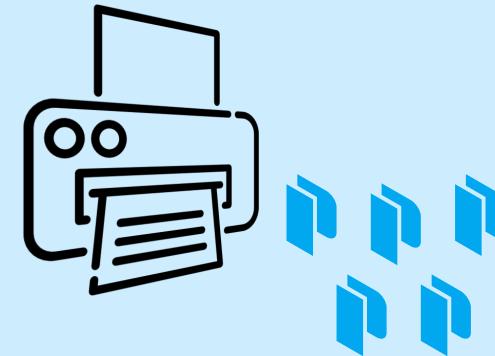


 **Packer**

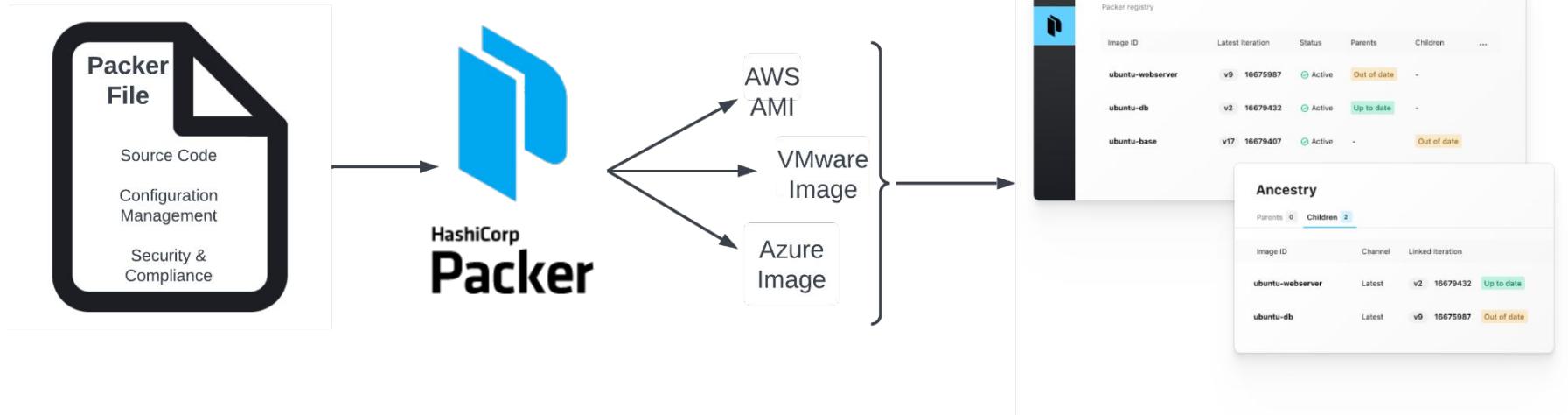
- Create/build your image with all the specific requirements your organization needs to meet
- Maintains the image by making updates when necessary

 **HCP Packer**

- Store the metadata of your image after being built with Packer
- Quickly deploy identical machine images for multiple platforms from a single source template



HCP Packer Workflow



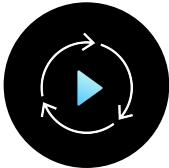
Organizations Must Still Define:



Which are the “golden” images, and what version should they use?



What is the ideal workflow to ensure images are hardened, secure, and compliant and can be replicated across clouds and private data centers?



Can this process be integrated into our current provisioning workflows?

Golden Image Pipeline

- HCP Packer is a multi-cloud image registry
- It uses metadata from your existing images to track iterations
- Builds artifacts across downstream builds and provisioning pipelines
- **It forms the foundation of a multi-cloud golden image pipeline**

The screenshot shows the Packer Artifact Registry interface. On the left, there's a sidebar with the Packer logo and a dark vertical bar. The main area has a header "Packer Artifact Registry". Below it, there's a section titled "Images" with a table:

ID	Latest Version
hashicorp/common	v31 eef9ac1
hashicorp/another	v96 bab1ee3
hashicorp/example	v2 fbb8bc2
hashicorp/core	v142 ccd6ca6
hashicorp/waypoint	v96 bab1ee3
hashicorp/horizon	v47 eee2af0

To the right, there's a detailed view for the "hashicorp/common" image, which is version v31 eef9ac1. This view includes a "Builds" section with entries for AWS, GCP, and Azure:

Cloud Provider	Region	Image ID	Created
aws	AWS-2.south us-east-2	Image ID: aws-a345agh31	Created: 9/17/21
gcp	gcp-east.example gcp-east	Image ID: gcp-at2kadsd29	Created: 8/8/21
azure	azure.basic-example-west us-west-2	Image ID: ami-832kjhd2k0	Created: 8/2/21

Standardizing the Workflow

Bridging the Gap Between Image Factories and Image Deployments

- **Gain visibility** into a base image's downstream builds and associated provisioning pipelines
- **Automate updates** by triggering a build or an apply
- **Create a compliant workflow** for images across all of your infrastructure

The diagram illustrates the standardization of the workflow by comparing two configuration files: `packer-application/application.pkr.hcl` and `main.tf`.

`packer-application/application.pkr.hcl` (top):

```
data "hcp-packer-iteration" "ubuntu" {  
  bucket_name = "learn-packer-ubuntu"  
  channel     = "production"  
}
```

`main.tf` (bottom):

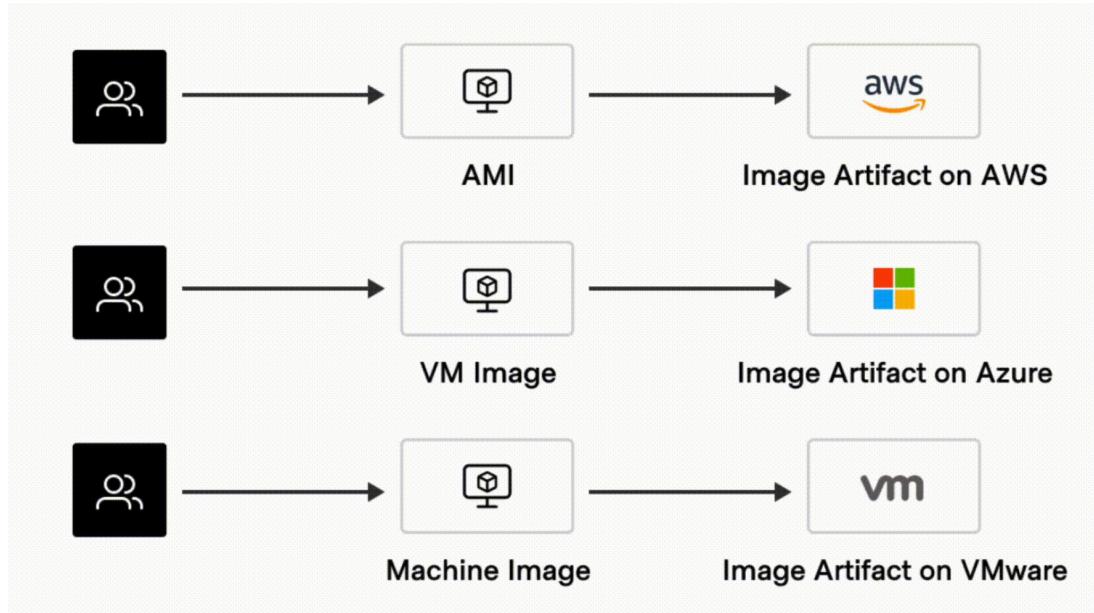
```
provider "hcp" {}  
  
data "hcp_packer_iteration" "ubuntu" {  
  bucket_name = "learn-packer-ubuntu"  
  channel     = "production"  
}
```

A callout box points from the text "Write a golden image once, and reuse it across downstream builds and provisioning pipelines" to the `main.tf` snippet.

Write a golden image once, and reuse it across downstream builds and provisioning pipelines

The Shift to HCP Packer

- **Easy to read and write:**
Leverages HCL2, making it easier to use alongside Terraform
- **Extensible:** Use one language to manage images across clouds and integrate with other configuration tools
- **Open source:** Use existing community templates, or write your own



Use Cases



Automated Machine Images



Golden Image Pipeline



Image Governance & Compliance

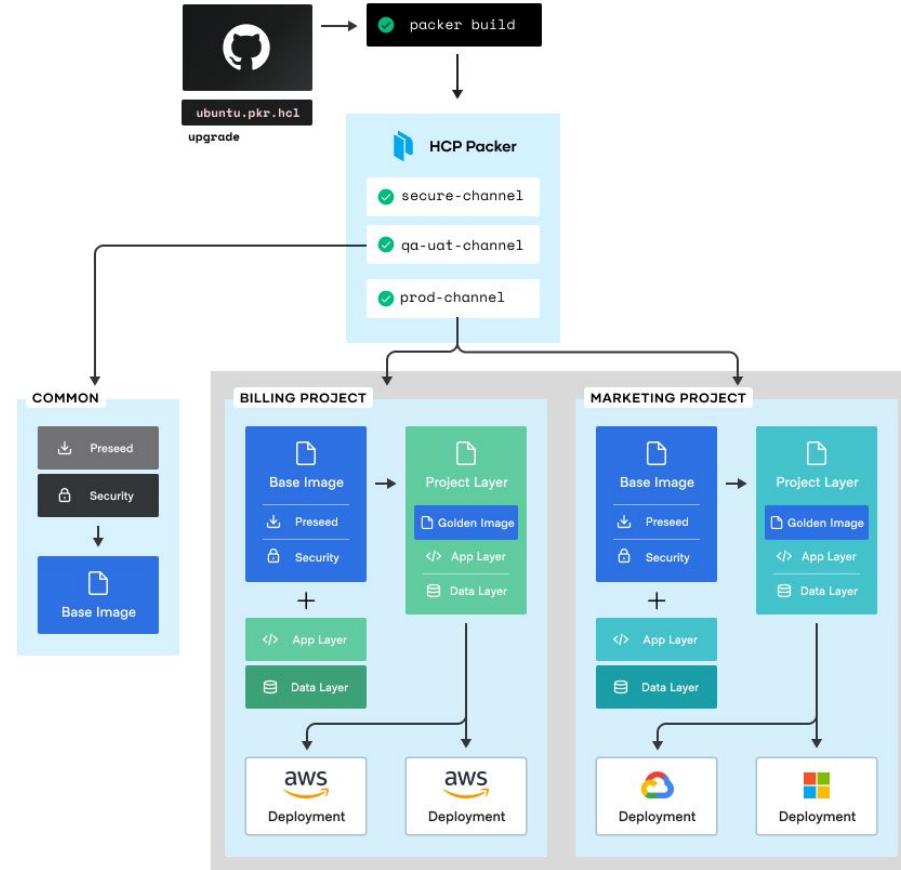


Unified Workflow with Terraform

Feature:

Release Management

- **Image channels** assign iterations to human-readable names that consumers can reference in Packer templates or Terraform configurations
- **Standardize versions** by declaring a preferred iteration of an image
- **Empower other teams** to retrieve the latest image version from their desired channel without changing their configurations



Feature:

Image Compliance Checks

- **Ensure images are compliant** by integrating HCP Packer with Terraform Cloud using run tasks
- **Reduce risks** by scanning all Terraform configurations for revoked images

The screenshot shows the Terraform Cloud interface for a recent run. At the top, there's a header with 'Cost estimated' (orange), 'Triggered via UI' (green), and a 'CURRENT' button. Below the header, the run was triggered by 'mossmartinez' 6 hours ago. The status bar indicates 'Plan finished' 6 hours ago, with 'Resources: 1 to add, 0 to change, 1 to destroy'. A green bar shows '+ 1 to create' and a red bar shows '- 1 to destroy'. A search bar allows filtering by address, and a link to 'Download raw log' is available. The AWS provider is listed with one instance created. Under 'Outputs', two items are planned to change. A link to 'Download Sentinel mocks' is present, with a note that they can be used for testing your Sentinel policies.

! Cost estimated Triggered via UI CURRENT
Triggered via UI

mossmartinez triggered a run from UI 6 hours ago Run Details ▾

Plan finished 6 hours ago Resources: 1 to add, 0 to change, 1 to destroy ^
Started 6 hours ago > Finished 6 hours ago

+ 1 to create - 1 to destroy

Filter resources by address... Terraform 1.1.4 Download raw log

> - + AWS aws_instance.app_server

> Outputs 2 planned to change

Download Sentinel mocks Sentinel mocks can be used for testing your Sentinel policies

Tasks passed 6 hours ago Tasks: 1 passed, 0 failed Beta ^
Running 6 hours ago > Passed 6 hours ago

> HCPPackerV0 passed (mandatory) Details ▾

Advantages of Packer

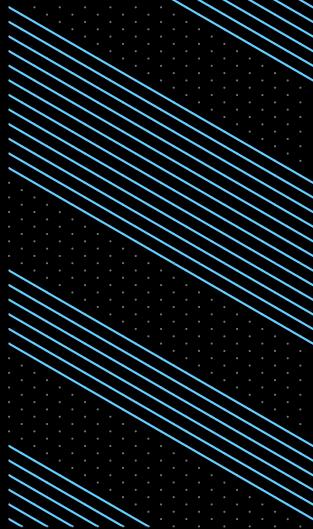
- 1 Rapid infrastructure deployment
- 2 Multi-provider portability
- 3 Improved stability
- 4 Greater testability

Advantages of HCP Packer

- 1 Visibility
- 2 Governance & Controls
- 3 Unified Workflows



02



Creating a Golden Image

Packer Templates

- The `packer {}` block contains Packer settings, including specifying a required Packer version
- The `source` block configures a specific builder plugin, which is then invoked by a `build` block
- The `build` block defines what Packer should do with the image after it launches

```
...
packer {
  required_plugins {
    ...
  }
}

source "<BUILDER_TYPE>" "<UNIQUE_NAME>" {
  ...
}

build {
  ...
}
```

Packer Templates

- [Build an Image](#)

```
    }
}

source "amazon-ebs" "ubuntu" {
  ami_name      = "learn-packer-linux-aws"
  instance_type = "t2.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name          = "ubuntu/images/*ubuntu-xenial-16.04-amd64-server-*"
      root-device-type = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}

build {
  name      = "learn-packer"
  sources  = [
    "source.amazon-ebs.ubuntu"
  ]
}
```



Build a Golden Image

```
$ packer build aws-ubuntu.pkr.hcl

learn-packer.amazon-ebs.ubuntu: output will be in this color.

=> learn-packer.amazon-ebs.ubuntu: Prevalidating any provided VPC information

=> learn-packer.amazon-ebs.ubuntu: Prevalidating AMI Name: learn-packer-linux-aws

learn-packer.amazon-ebs.ubuntu: Found Image ID: ami-0dd273d94ed0540c0

=> learn-packer.amazon-ebs.ubuntu: Creating temporary keypair:
packer_608a6435-e0b2-c633-95f0-bf168f01e891

=> learn-packer.amazon-ebs.ubuntu: Creating temporary security group for this
instance: packer_608a6437-6b48-288e-6d5e-c085366a5541

=> learn-packer.amazon-ebs.ubuntu: Authorizing access to port 22 from [0.0.0.0/0]
in the temporary security groups...

...
=> Wait completed after 5 minutes 19 seconds

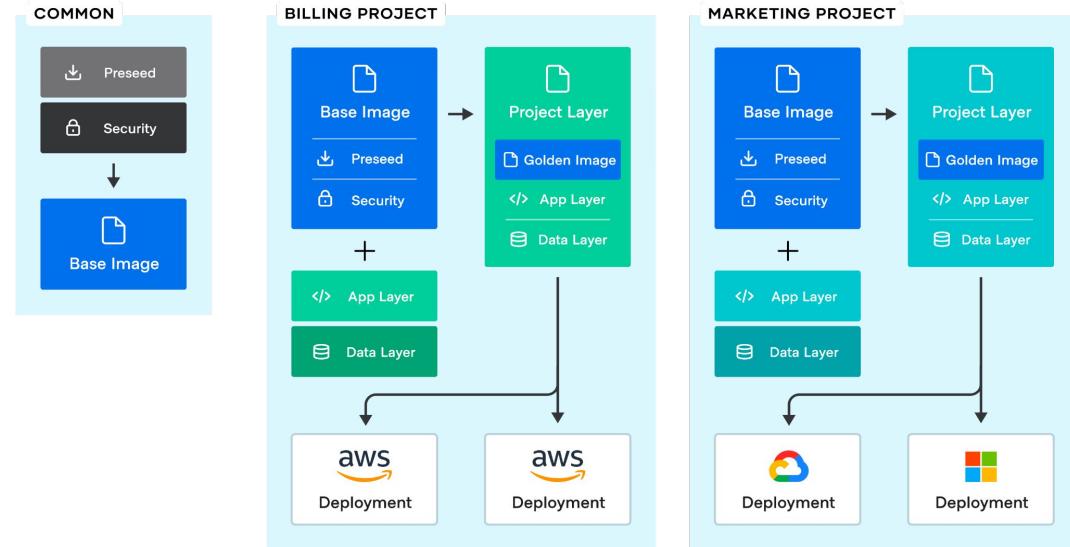
=> Builds finished. The artifacts of the successful builds are:
--> learn-packer.amazon-ebs.ubuntu: AMIs were created:
us-east-2: ami-0d9356dfdbea0d50b

--> learn-packer.amazon-ebs.ubuntu: Published metadata to HCP Packer registry
packer/learn-packer/iterations/01FFJD3GDKRBW4CZ314EZ467VY
```



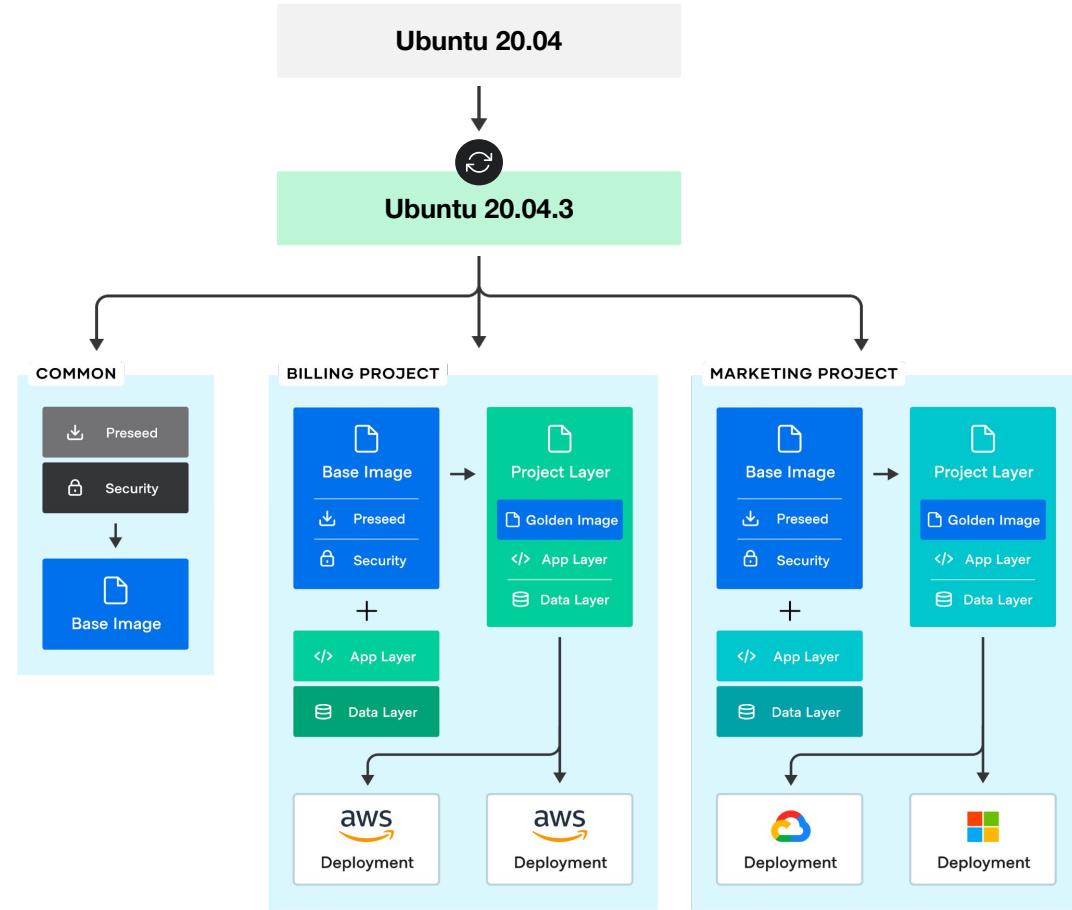
Golden Image Pipeline

Golden Image: an image on top of which developers can build applications, letting them focus on the application itself instead of system dependencies and patches





Updating your Golden Image



Golden Image Ancestry

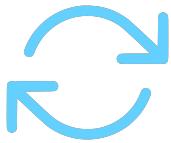
The screenshot displays two main views from the HashiCorp Packer interface:

- Top View (HashiCafe-inc / Packer):** Shows a list of images in the "Images" section. The table includes columns for Image ID, Latest Iteration, Status, Parents, Children, Platforms, and Last updated. Several images are listed, including `ubuntu20-tensorflow`, `ubuntu20-mariadb`, `ubuntu20-base`, `ubuntu20-nginx`, `ubuntu22-base`, and `docker-nginx`. The `ubuntu20-base` image is highlighted with a yellow warning icon.
- Bottom View (ubuntu20-base Overview):** Provides detailed information for the `ubuntu20-base` image. It shows the status as Active, published on Dec 08, 2022, at 4:17:29 PM. A red box highlights the "Ancestry" section, which lists three children images: `ubuntu20-tensorflow`, `ubuntu20-mariadb`, and `ubuntu20-nginx`.
- Integration:** Both views include "Integrate with Terraform Cloud" and "Manage" buttons.

Manage the Lifecycle of a Packer Image



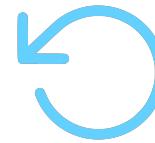
Create multiple
versions of the
same image



Release new
image versions



Scheduling a
revocation of
an image



Use channel
rollback

Best Practices

Variables

- Provide the type in your variable definition
- If no type is set, Packer assumes it will be of type string
- Set a default value so you don't have to at run time
- Do NOT save sensitive information to a VCS

```
...
variable "weekday" {}

variable "sudo_password" {
    type      = string
    default   = "mypassword"
    sensitive = true
}

variable "flavor" {
    type      = string
    default   = "strawberry"
}

variable "exit_codes" {
    type      = list(number)
    default   = [0]
}

locals {
    ice_cream_flavor = "${var.flavor}-ice-cream"
    foo              = "bar"
}
```

Best Practices

Bucket Labels & Build Labels

- Bucket labels help identify characteristics common to a set of images
- HCP Packer applies custom bucket labels to an entire image bucket
- Build labels are a immutable set of labels based on the Packer template at the time of build

```
...
"bucket-labels": {
    "team" = "amazon-development",
    "os"   = "Ubuntu"
}

build_labels = {
    "build-time"  = timestamp(),
    "build-source" = basename(path.cwd)
}
```

Best Practices

Parallel Builds

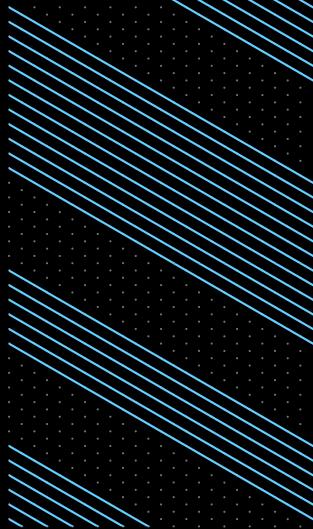
1. Create a source
2. Add the source to the sources array in the build block

```
...
source "docker" "ubuntu" {
  image  = "ubuntu:xenial"
  commit = true
}

source "docker" "ubuntu-bionic" {
  image  = "ubuntu:bionic"
  commit = true
}

build {
  name      = "learn-packer"
  sources = [
    "sources.docker.ubuntu",
    "sources.docker.ubuntu-bionic"
  ]
}
```

03



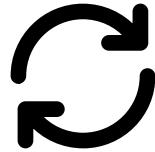
Unified Workflow with Terraform Cloud

Terraform Cloud + HCP Packer

Better together

HCP Packer

- Build images
- Publish metadata
- Release via channels
- Revoke and rollback
- View ancestry & history

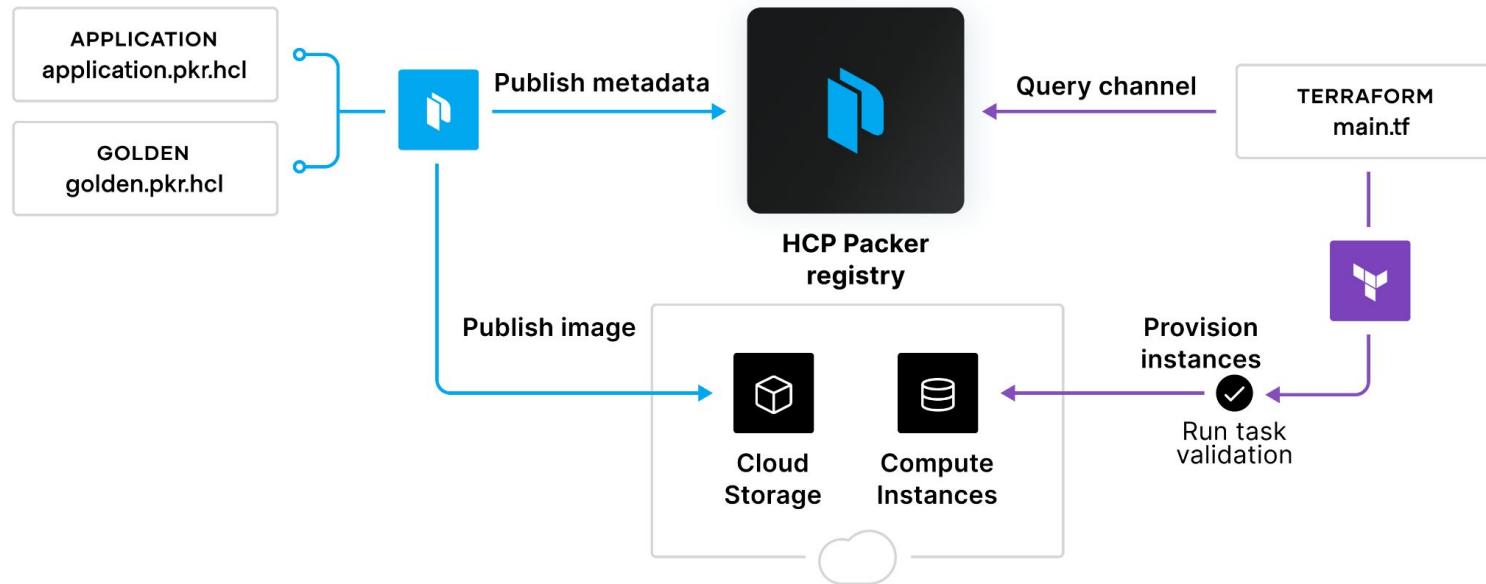


Terraform Cloud

- Fetch images
Data source
- Provision infrastructure
- Validate image status
Run tasks, continuous validation

Terraform Cloud + HCP Packer

Integrated build & provisioning workflow



Automate Image Updates

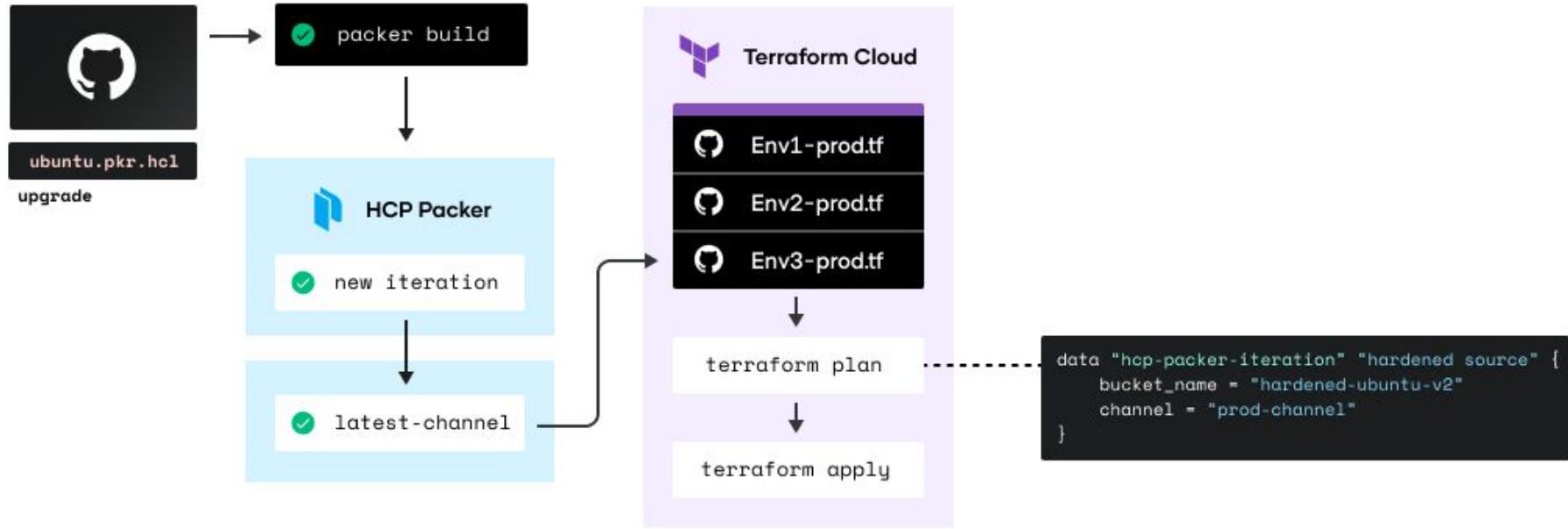


Image Lifecycle Management

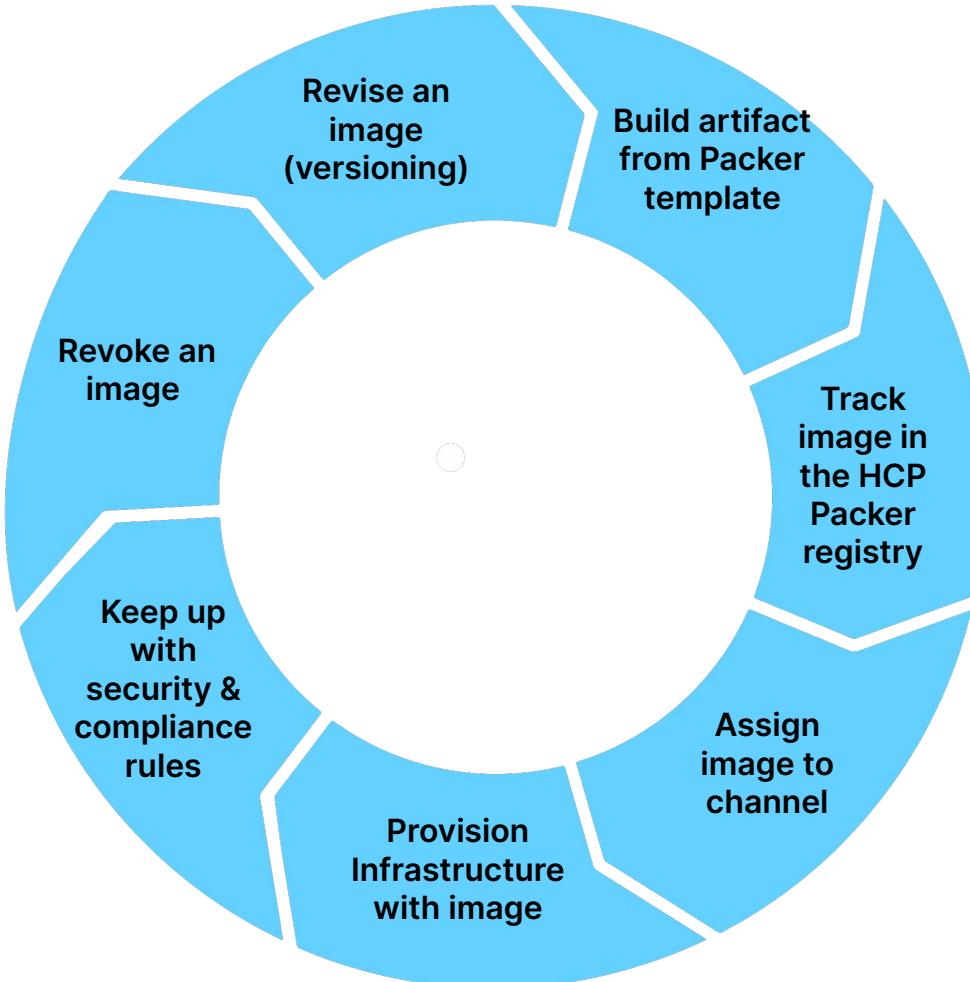


Image Revocation

Revoke iteration v2 c2ec6f1a

This action cannot be undone.

Reason (Optional)

Channel rollback

Rollback channels? Required

This iteration is assigned to multiple channels. Would you like to roll all channels back to their most

Choose an option

Yes, rollback channel

No, do not rollback channel

Revoke

Cancel

⚠ Revoke iteration v1 810d1272

X

Revoke iteration v1 810d1272 and detach it from all channels?

This action cannot be undone.

When would you like to revoke this iteration?

Revoke immediately

Revoke at a future date

You can cancel this scheduled revocation at any time before the selected date.

Choose date to revoke this iteration

11/16/2022



Choose time to revoke this iteration

All times are represented in UTC

09:40 PM



Reason (Optional)

Learning about scheduling revocation

Revoke

Cancel



Resources

- [Intro to HCP Packer](#)
- [Mutable vs. Immutable Infrastructure](#)
- [Packer with AWS](#)
- [Getting Started with HCP Packer](#)
- [Fundamentals of HCP Packer](#)
- [Terraform Cloud Run Tasks for HCP Packer](#)
- [Channel Assignment History and Automated Rollback](#)



Thank you

customer.success@hashicorp.com

www.hashicorp.com/customer-success