HashiCorp

Automating Image Pipelines with HCP Packer

How we leverage HCP Packer and GitHub Actions to automate our image build, test, and deployment pipeline

Caleb Albers
Infrastructure Engineer
he/him

CI/CD at HashiCorp

2,800

GitHub Repos

70,000

CI Jobs / Month

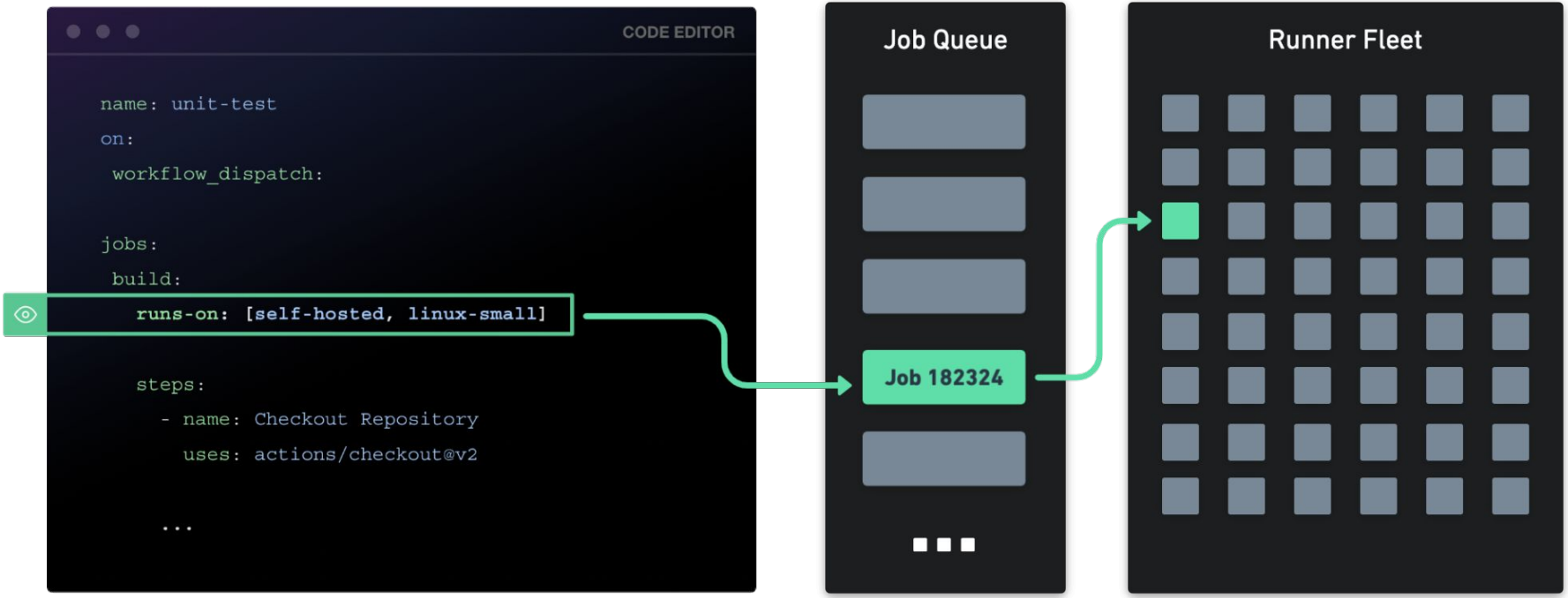
16,000

Compute Hours / Month

+22%

Increase in Jobs / Month

Actions Runner Fleet



Self-Hosted Runner Fleet

Operating Systems

- Ubuntu
- MacOS
- Windows

Instance Sizes

- Small
- Medium
- Large
- Extra-Large

Self-Hosted GitHub Actions Runners

- Support for Ubuntu, MacOS, and Windows
- Strict VM-level isolation and ephemerality
- Frequent updates to get the latest tools and features
- Ability to track images and promote between environments
- Ability to scale as demand evolves
- Confidence in changes through automated testing
- Multi-Region Support
- Future Multi-Cloud Support

Self-Hosted GitHub Actions Runners

- Support for Ubuntu, MacOS, and Windows
- Strict VM-level isolation and ephemerality
- Frequent updates to get the latest tools and features
- Ability to track images and promote between environments
- Ability to scale as demand evolves
- **Confidence** in changes through automated testing
- Multi-Region Support
- Future Multi-Cloud Support

Achieving Confidence

- Controlled promotion (dev → staging → prod)
- Tracking image metadata
- Consistent across cloud providers and regions
- High degree of automation
- Image Revocation

  …

- Easy implementation with Terraform and Packer

Build & Deployment Pipeline

- Automated weekly builds
- Infrastructure is managed via Terraform runs
- Integration test launches an instance with the new build
- A Github Actions job is sent to the test instance to verify functionality
- If successful, builds are instantly promoted to dev and staging environments
- Production release 24-hours later

CODE EDITOR

```
variable "image_id_aws_us_west_2" {
   type        = string
   description = "Image for servers in AWS US-WEST-2."
}

variable "image_id_aws_us_east_1" {
   type        = string
   description = "Image for servers in AWS US-EAST-1."
}

# GCP
variable "image_id_gcp_us_west4" {
   type        = string
   description = "Image for servers in GCP US-WEST-4."
}
```

TF Variables

+ Controlled Promotion
+ Granular Implementation
− Custom Tooling
− Variable Sprawl

Most Recent

- ➕ Easy to implement
- ➖ Best Effort Promotion
- ➖ Provider-specific

CODE EDITOR

```
data "aws_ami" "image" {
    owners      = [var.ami_owner]
    most_recent = true

    filter {
        name   = "name"
        values = ["crt-runner-ubuntu-*"]
    }

}
```

Tag Filtering

- ➕ Easy to implement
- ➕ Controlled Promotion
- ➖ Provider-specific

CODE EDITOR

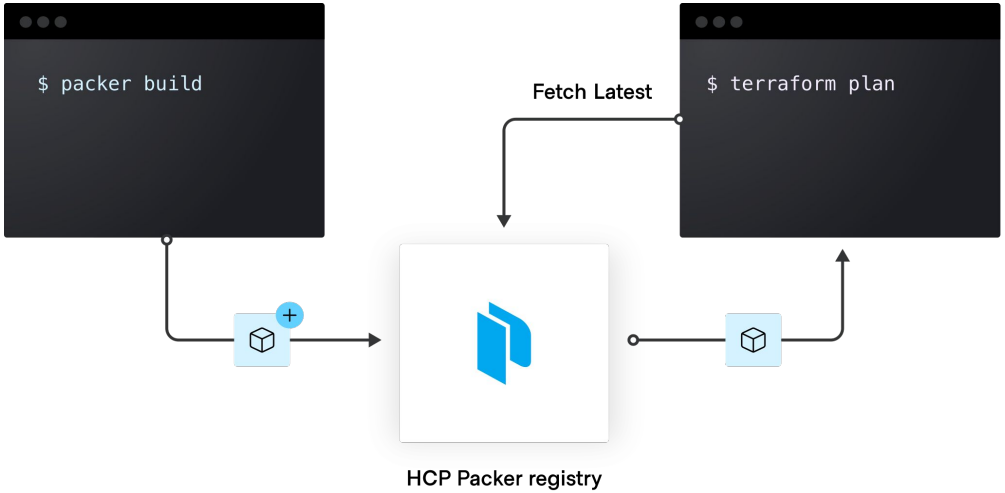
```
data "aws_ami" "image" {
   owners      = [var.ami_owner]
   most_recent = true

   filter {
       name   = "name"
       values = ["crt-runner-ubuntu-*"]
   }

   filter {
       name   = "tag:env"
       values = ["production"]
   }

}
```

What about HCP Packer?



- Acts as glue between Packer and Terraform
- Source of truth for image metadata
- Terraform grabs image IDs from HCP Packer
- API for easy programmatic interaction

Some HCP Packer Terminology:

Bucket

/ˈbəkət/

Collections of artifacts originating from the same Packer file

Iteration

/ˌidəˈrāSH(ə)n/

The output of a given packer build run

Build

/bild/

The output from a single Packer file's builder block in an iteration. Specific per cloud, region, etc.

Channel

/ˈCHanl/

Release labels applied to iterations, e.g. dev, staging, prod

Terraform Data Source

- Grabs current iteration for given release channel
- Allows for multi-cloud and multi-region builds
- Gracefully handles image revocation

CODE EDITOR

```
data "hcp_packer_iteration" "ubuntu" {
  bucket_name = "crt-runner-ubuntu"
  channel     = var.env  # "staging"
}

data "hcp_packer_image" "ubuntu" {
  bucket_name    = "crt-runner-ubuntu"
  cloud_provider = "aws"
  iteration_id   = data.hcp_packer_iteration.ubuntu.id
  region         = "us-west-2"
}

resource "aws_instance" "runner" {
  ami           =  data.hcp_packer_image.ubuntu.cloud_image_id
  instance_type = "t3.small"
}
```

Terraform Data Source

- Grabs current iteration for given release channel
- Allows for multi-cloud and multi-region builds
- Gracefully handles image revocation

CODE EDITOR

```
data "hcp_packer_iteration" "ubuntu" {
  bucket_name = "crt-runner-ubuntu"
  channel     = var.env  # "staging"
}

data "hcp_packer_image" "ubuntu" {
  bucket_name    = "crt-runner-ubuntu"
  cloud_provider = "aws"
  iteration_id   = data.hcp_packer_iteration.ubuntu.id
  region         = "us-west-2"
}

resource "aws_instance" "runner" {
  ami           =  data.hcp_packer_image.ubuntu.cloud_image_id
  instance_type = "t3.small"
}
```

Terraform Data Source

- Grabs current iteration for given release channel
- Allows for multi-cloud and multi-region builds
- Gracefully handles image revocation

CODE EDITOR

```
data "hcp_packer_iteration" "ubuntu" {
  bucket_name = "crt-runner-ubuntu"
  channel     = var.env  # "staging"
}

data "hcp_packer_image" "ubuntu" {
  bucket_name    = "crt-runner-ubuntu"
  cloud_provider = "aws"
  iteration_id   = data.hcp_packer_iteration.ubuntu.id
  region         = "eu-south-1"
}

resource "aws_instance" "runner" {
  ami           =  data.hcp_packer_image.ubuntu.cloud_image_id
  instance_type = "t3.small"
}
```

Terraform Data Source

- Grabs current iteration for given release channel
- Allows for multi-cloud and multi-region builds
- Gracefully handles image revocation

CODE EDITOR

```
data "hcp_packer_iteration" "ubuntu" {
  bucket_name = "crt-runner-ubuntu"
  channel     = var.env  # "staging"
}

data "hcp_packer_image" "ubuntu" {
  bucket_name    = "crt-runner-ubuntu"
  cloud_provider = "gcp"
  iteration_id   = data.hcp_packer_iteration.ubuntu.id
  region         = "us-west2"
}

resource "google_compute_instance" "runner" {
  boot_disk {
    initialize_params {
      image = data.hcp_packer_image.ubuntu.cloud_image_id
      ...
```

API Image Promotion

- Automated as part of build pipeline
- Promotes across all clouds/regions
- Terraform updates during next Plan

CODE EDITOR

```
> curl --request PATCH \
    --url "$base_url/images/$bucket/channels/$channel" \
    --data-raw '{"iteration_id":"'"$iteration_id"'"}' \
    --header "authorization: Bearer $bearer"
```

```
> ./helper.sh channels  set-iteration \
    crt-runner-ubuntu \
    staging \
    <iteration_id>
```

Image Promotion Job

CODE EDITOR

```
packer-promote-image:
 Steps:
  ...
   - name: Determine Iteration ID
     id: hcp
     run: |
       ...
       iteration_id=$(echo "$build" | jq -r '.custom_data.iteration_id')
       echo "::set-output name=iteration_id::$iteration_id"

   - name: Promote Iteration to Dev
     run: ./helper.sh channels set-iteration crt-runner-ubuntu dev \
            ${{ steps.hcp.outputs.iteration_id}}
```

Demo Time!

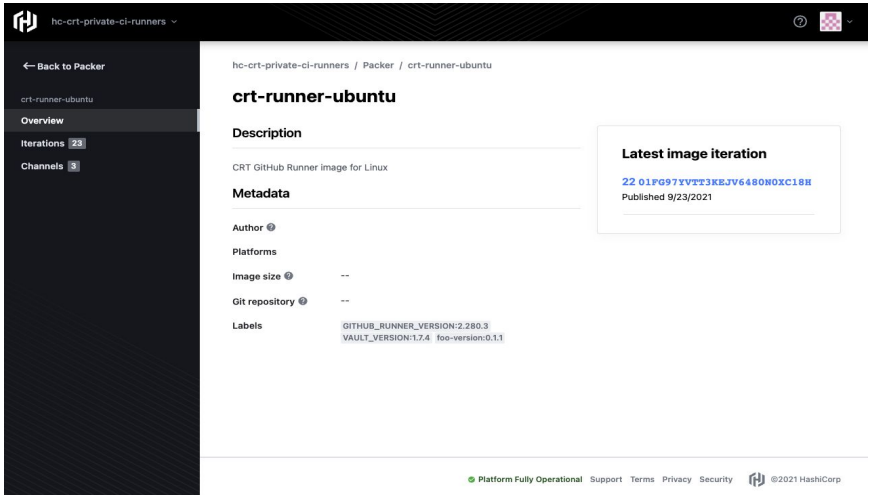
Image Automation

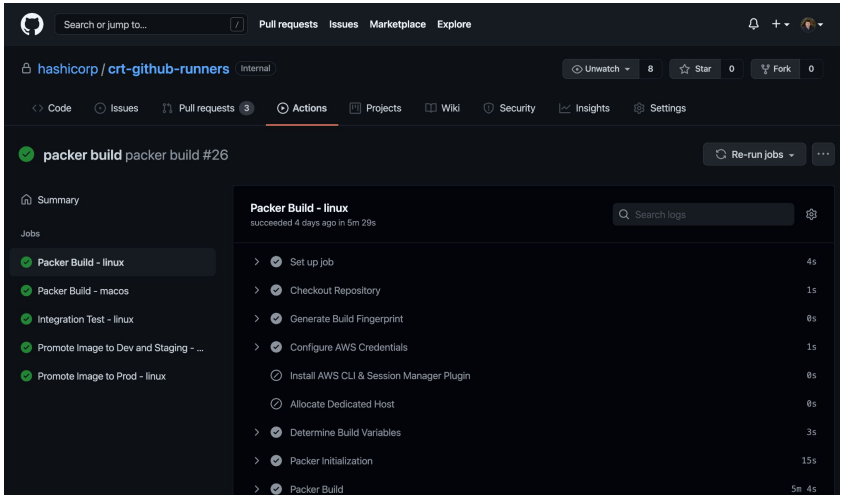
HCP Packer

- Multi-cloud, multi-region image registry
- Stores metadata about each image
- Release Channels for dev, staging, and prod

GitHub Actions

- Packer builds for multiple operating systems
- Automatically manages dedicated hosts
- Promotes image upon successful E2E test





Resources

Code	github.com/calebalbers/hcp-packer-demo
Recording	hashi.co/hashitalks-2022
HCP Packer Docs	cloud.hashicorp.com/docs/packer

LinkedIn	@calebalbers
GitHub	@calebalbers

HashiCorp