# Packer

Creation of Base AMI's using packer and setup of other prerequisites of project
.

Packer is an open-source tool for creating machine images, such as AMIs, Docker images, and more. It allows you to build consistent, automated, and repeatable images across multiple platforms.

## What is Packer?

### 1-Consistent Image Creation

Packer allows you to define your machine configuration in code, ensuring the same image is built every time.

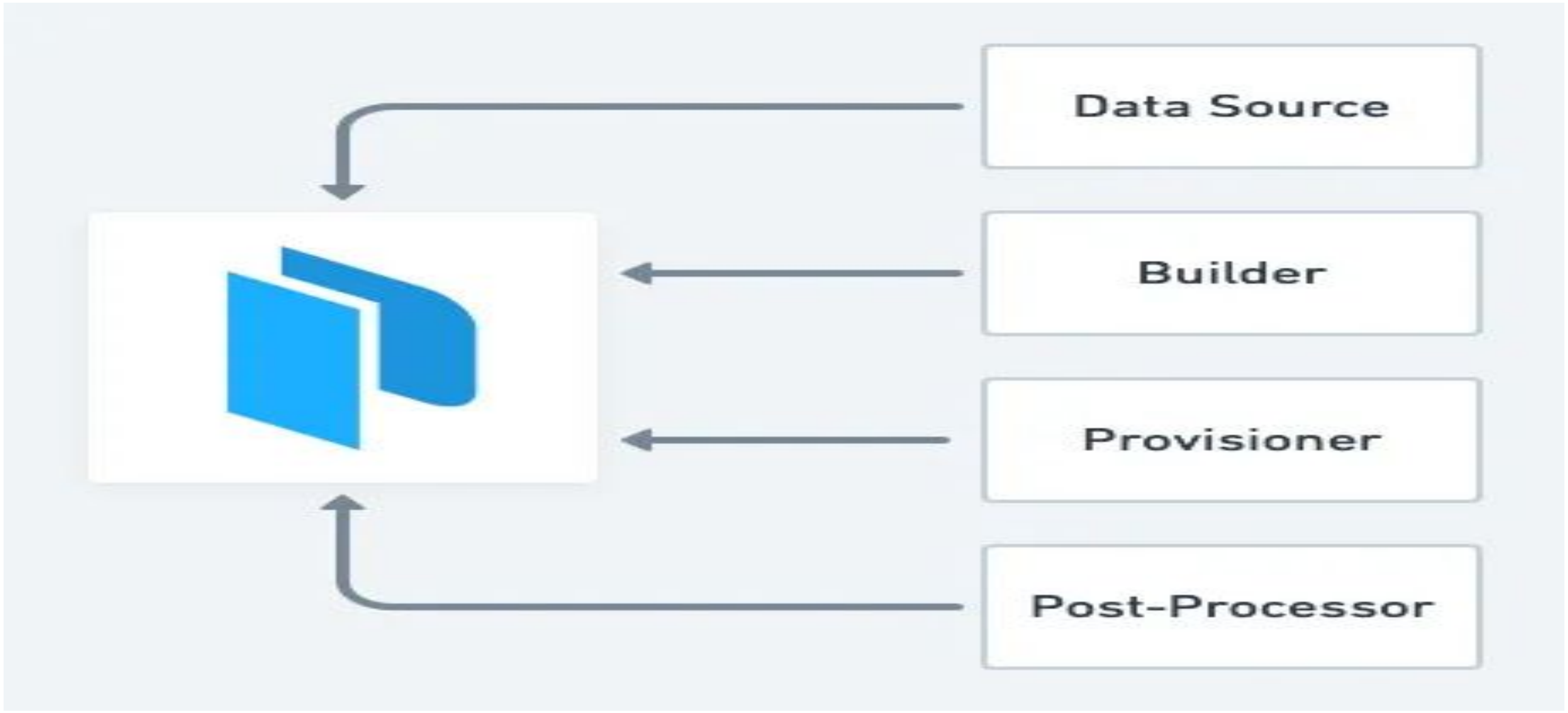
### 2-Automated Builds

Packer can automatically build images for multiple platforms from a single configuration file.

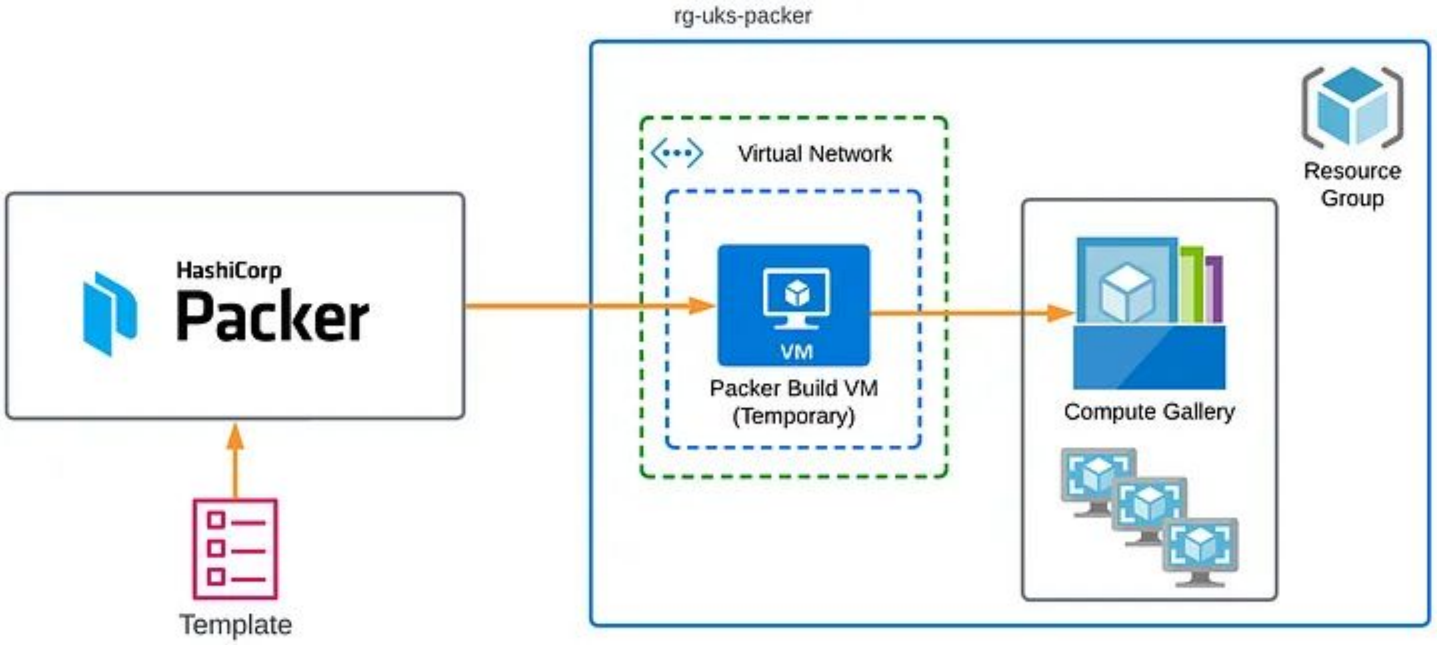
### 3- Reproducible Environments

The images created by Packer can be used to provision identical environments, making deployments reliable and predictable.

## Core Concepts of Packer

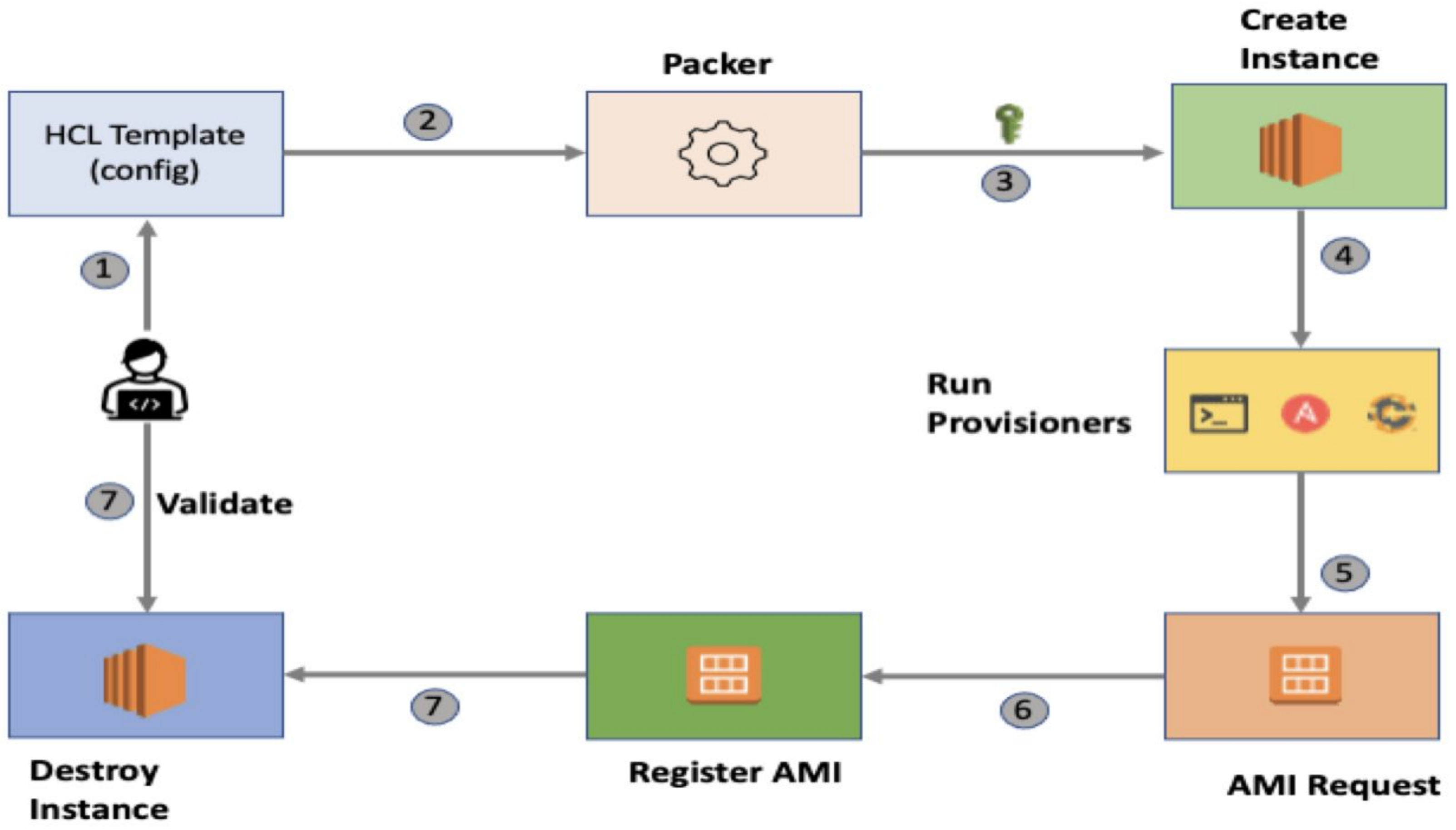


## Builder



rg-uks-packer
HashiCorp Packer
Virtual Network
VM
Packer Build VM (Temporary)
Resource Group
Compute Gallery
Template

## Provisioners

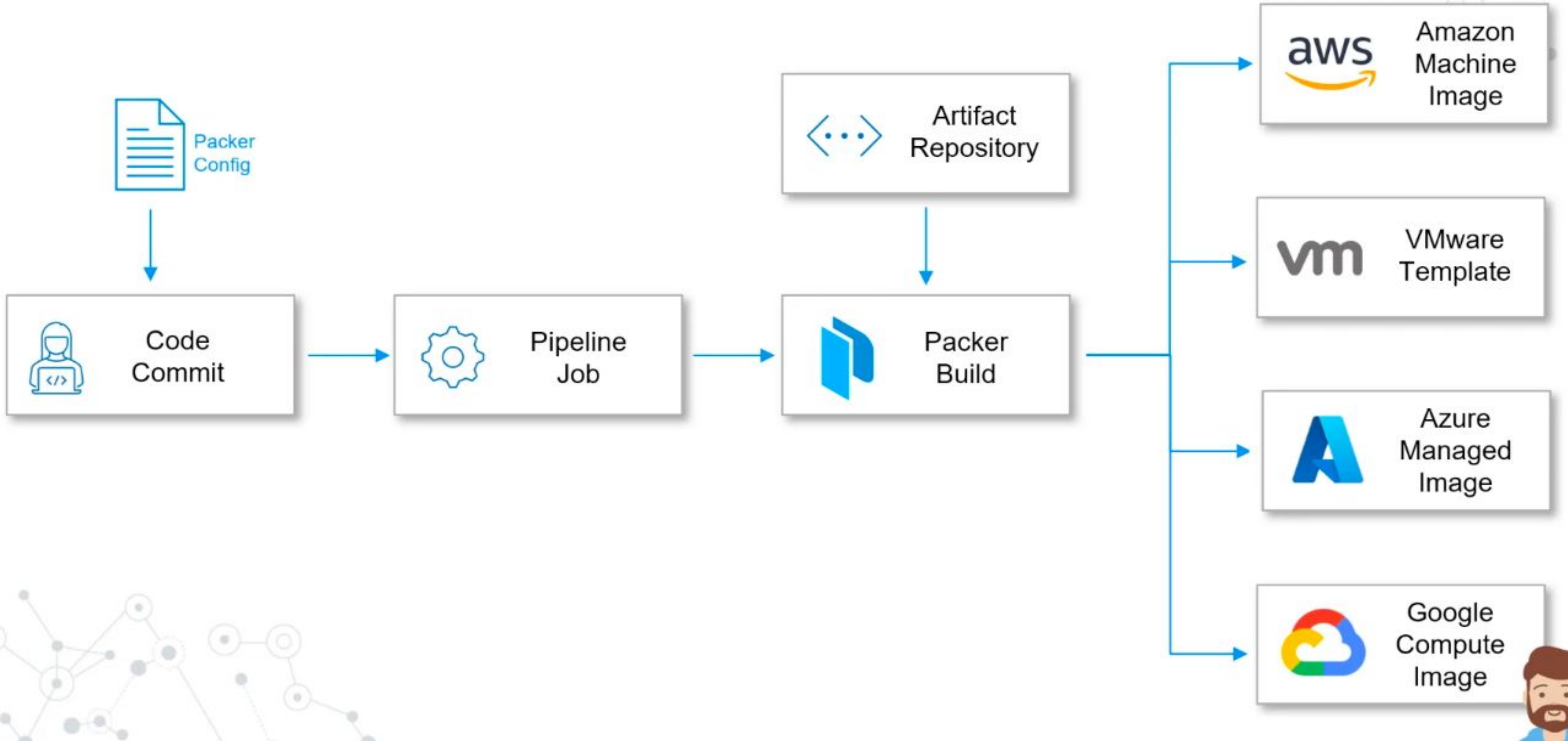


## Creation of a Base AMI using Packer

- You are free to use AWS or Azure for this entire project
- Packer is a differentiating skill. Easy to get started and custom base AMIs are used across the industry and in every company.
- Entry level fresher engineers can use shell script as provisioner.
- If you are expert, then you should try the Ansible based provisioner.
- So, you will build a Packer based pipeline in your choice of CI tool by taking an AWS/Azure provided base image and install your required software using Shell Script or Ansible provisioner.

## Automate Image Builds Across Platforms

Golden Images For All Your Workloads



## Packer Configuration Files

| **JSON/HCL Format** | **Builder Definition** | **Provisioner Definition** |
|---|---|---|
| Packer configuration files use a JSON/HCL format to define the image build process. | The builder section specifies the target platform, such as AWS, Azure, or GCP, and the necessary configuration options. | The provisioner section defines the scripts or commands to be executed during the image build process. |

```
packer {
  required_plugins {
    amazon = {
      source  = "github.com/hashicorp/amazon"
      version = "~> 1"
    }
  }
}

source "amazon-ebs" "ubuntu" {
  ami_name      = "packer-ubuntu-aws-{{timestamp}}"
  instance_type = "t3.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name                = "ubuntu/images/*ubuntu-jammy-22.04-amd64-server-*"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}
```

# Packer Commands

| | | |
|---|---|---|
| 01 | packer init | Create a new Packer configuration file. |
| 02 | packer validate | Validate the syntax of a Packer configuration file. |
| 03 | packer build | Build one or more machine images from a Packer configuration file. |
| 04 | packer inspect | View the components of a Packer configuration file. |
| 05 | packer console | Launch an interactive Packer console for testing configurations. |

## Summary

source block specifies the type (the builder, such as amazon-ebs, googlecompute, etc.) and a unique name for that source.

The source block holds all settings needed to interact with that platform—like region, instance type, base image, credentials, and more

- The source block sets up how to build an image (defining the platform, credentials, etc.).
- The build block tells Packer what to do with that source, including any scripts and post-processing actions.

## Demo

## Creating Custom AMI using packer with Github Action CI/CD

https://developer.hashicorp.com/packer/docs/provisioners

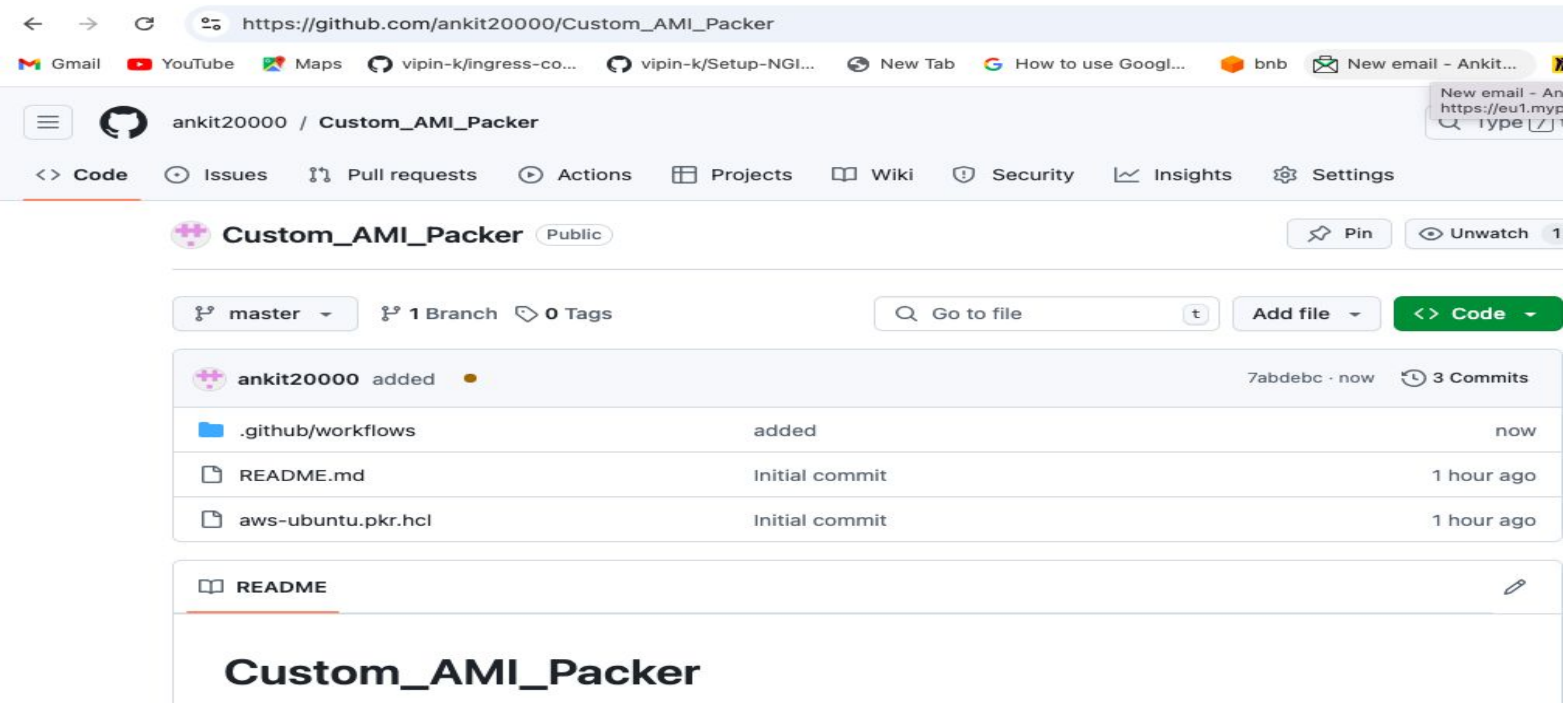
https://developer.hashicorp.com/packer/tutorials/aws-get-started/aws-get-started-build-image

https://github.blog/enterprise-software/ci-cd/build-ci-cd-pipeline-github-actions-four-steps/

## Repo-

**https://github.com/ankit20000/Custom_AMI_Packer**

## Step1: Clone/fork this repo to your github Account

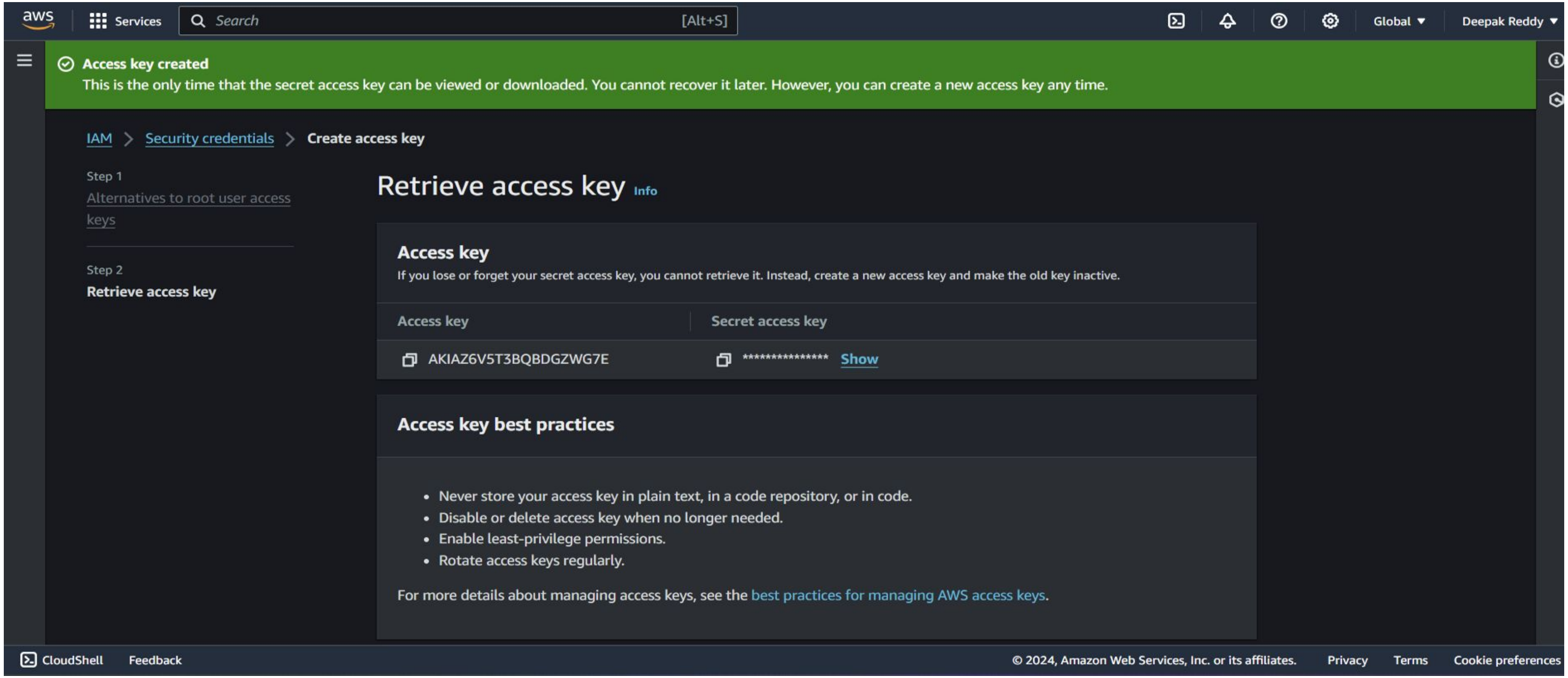


ankit20000 added · 7abdebc · now · 3 Commits

| | | |
|---|---|---|
| .github/workflows | added | now |
| README.md | Initial commit | 1 hour ago |
| aws-ubuntu.pkr.hcl | Initial commit | 1 hour ago |

Custom_AMI_Packer

## **Step2:** Create aws access keys from aws console to create AMI


aws Services Search [Alt+S] Global Deepak Reddy

**Access key created**
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Security credentials > Create access key

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Retrieve access key Info

**Access key**
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

| Access key | Secret access key |
|---|---|
| AKIAZ6V5T3BQBDGZWG7E | *************** Show |

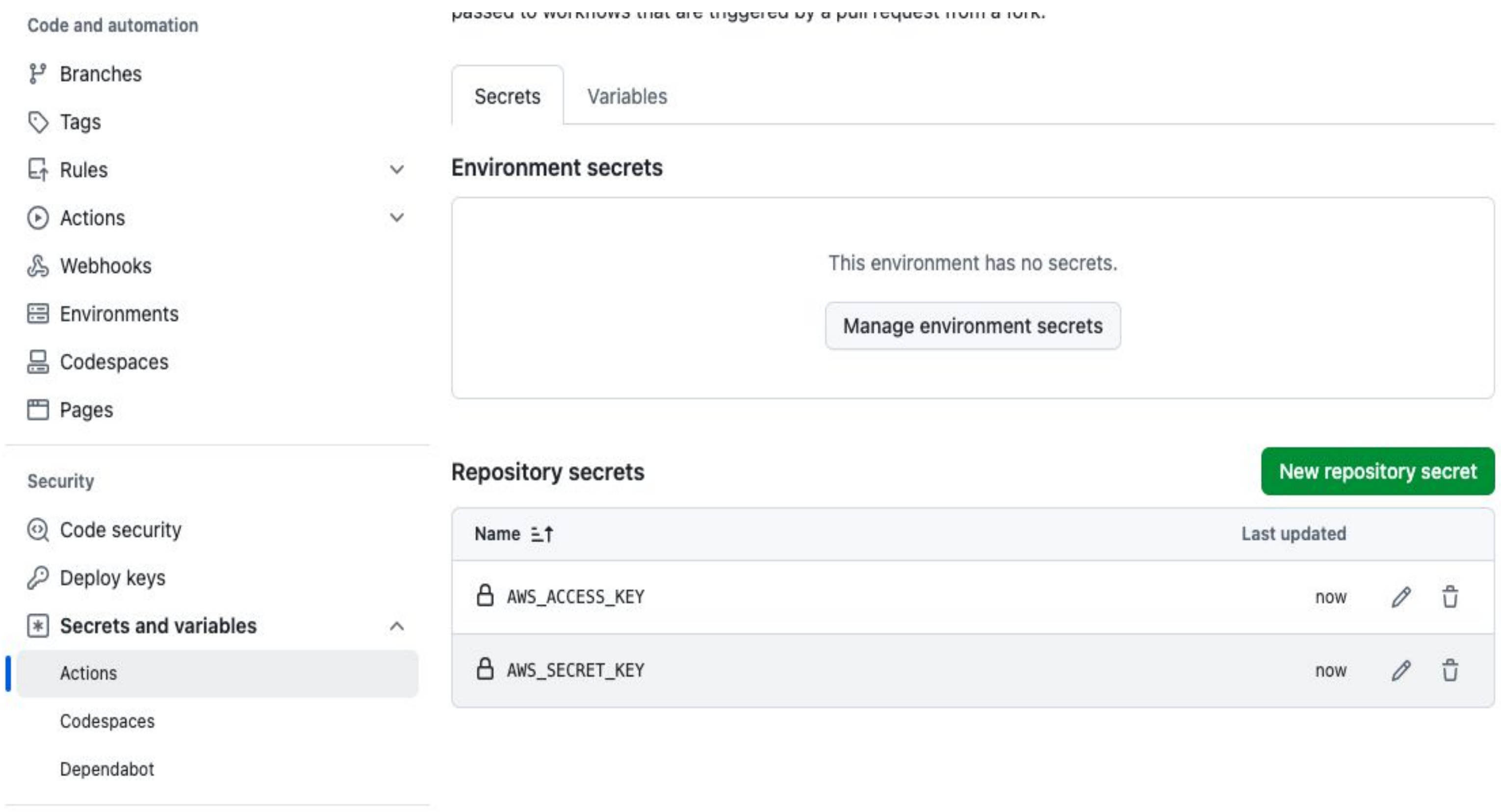
**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step3: now after creating the access keys, Go to setting and add these



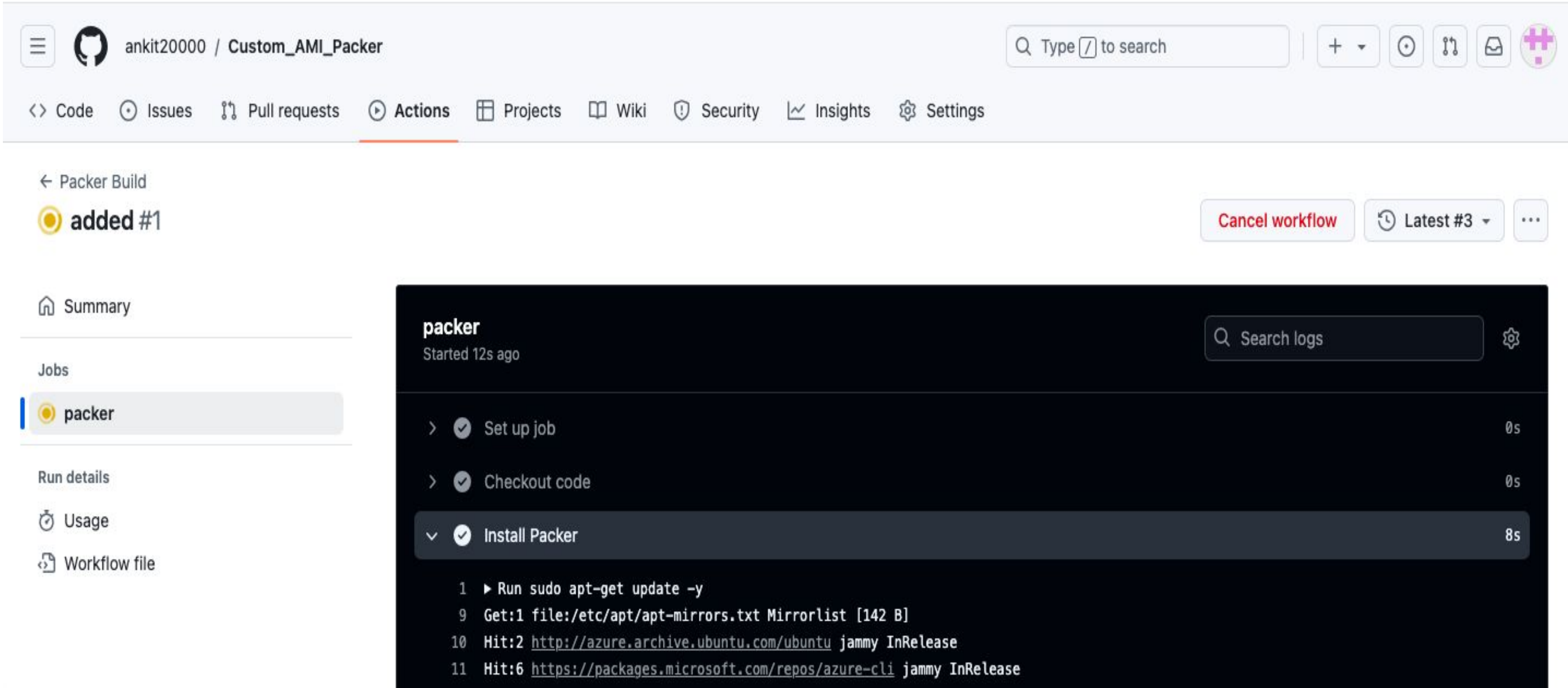
Environment secrets

This environment has no secrets.

Repository secrets

| Name | Last updated |
|---|---|
| AWS_ACCESS_KEY | now |
| AWS_SECRET_KEY | now |

# Step4- Now run the pipeline to build ami



```
▶ Run sudo apt-get update -y
Get:1 file:/etc/apt/apt-mirrors.txt Mirrorlist [142 B]
Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Hit:6 https://packages.microsoft.com/repos/azure-cli jammy InRelease
```

## Step5: check the AMI from the aws console and launch an instance  to check it correctly.

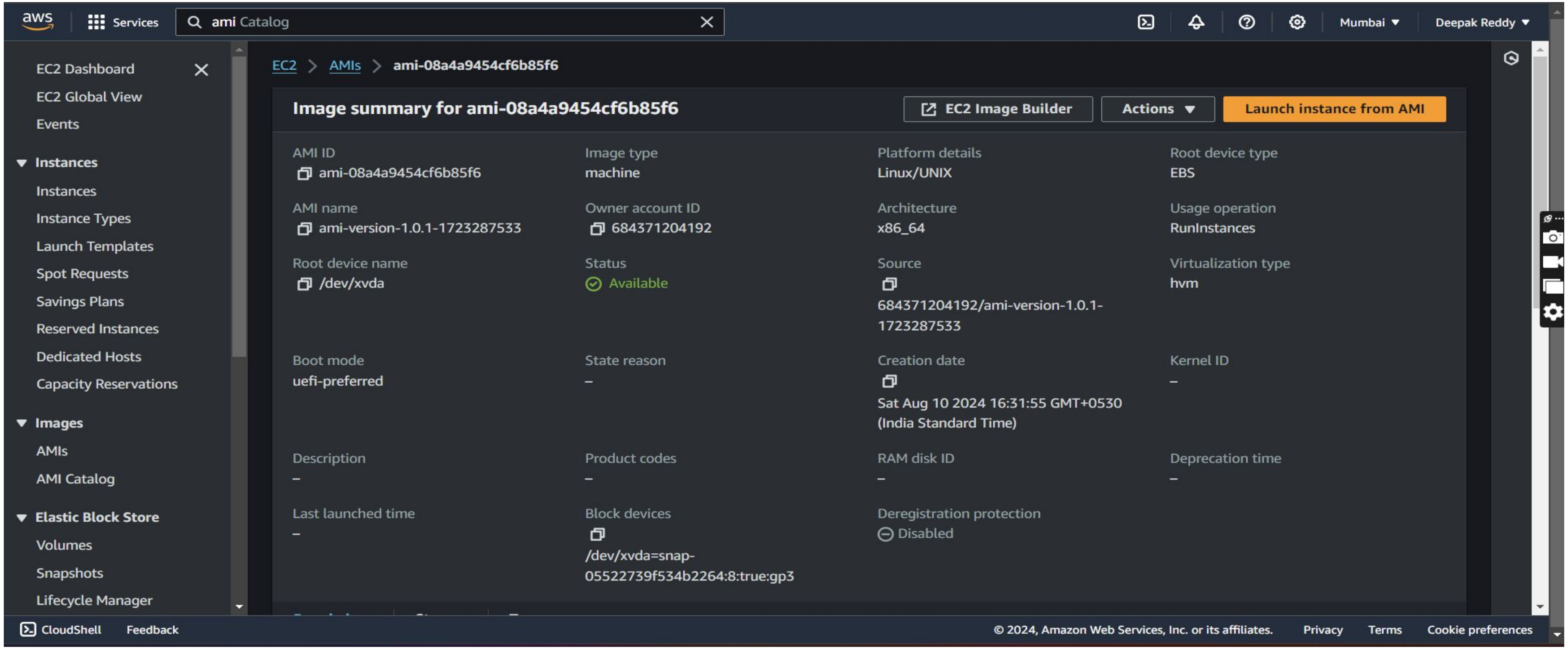


**Image summary for ami-08a4a9454cf6b85f6**

| | | | |
|---|---|---|---|
| AMI ID<br>ami-08a4a9454cf6b85f6 | Image type<br>machine | Platform details<br>Linux/UNIX | Root device type<br>EBS |
| AMI name<br>ami-version-1.0.1-1723287533 | Owner account ID<br>684371204192 | Architecture<br>x86_64 | Usage operation<br>RunInstances |
| Root device name<br>/dev/xvda | Status<br>Available | Source<br>684371204192/ami-version-1.0.1-1723287533 | Virtualization type<br>hvm |
| Boot mode<br>uefi-preferred | State reason<br>– | Creation date<br>Sat Aug 10 2024 16:31:55 GMT+0530 (India Standard Time) | Kernel ID<br>– |
| Description<br>– | Product codes<br>– | RAM disk ID<br>– | Deprecation time<br>– |
| Last launched time<br>– | Block devices<br>/dev/xvda=snap-05522739f534b2264:8:true:gp3 | Deregistration protection<br>Disabled | |