



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.

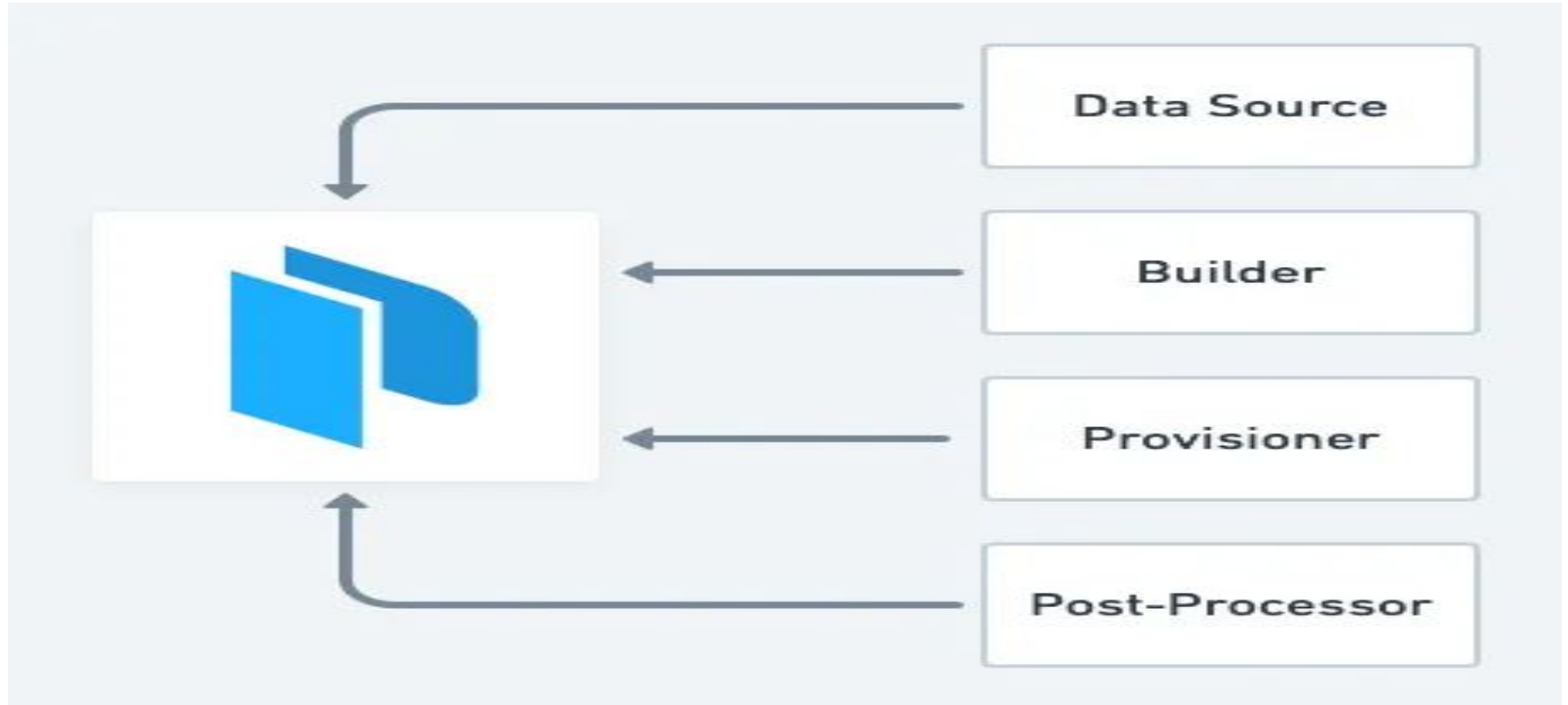
2-Automated Builds

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

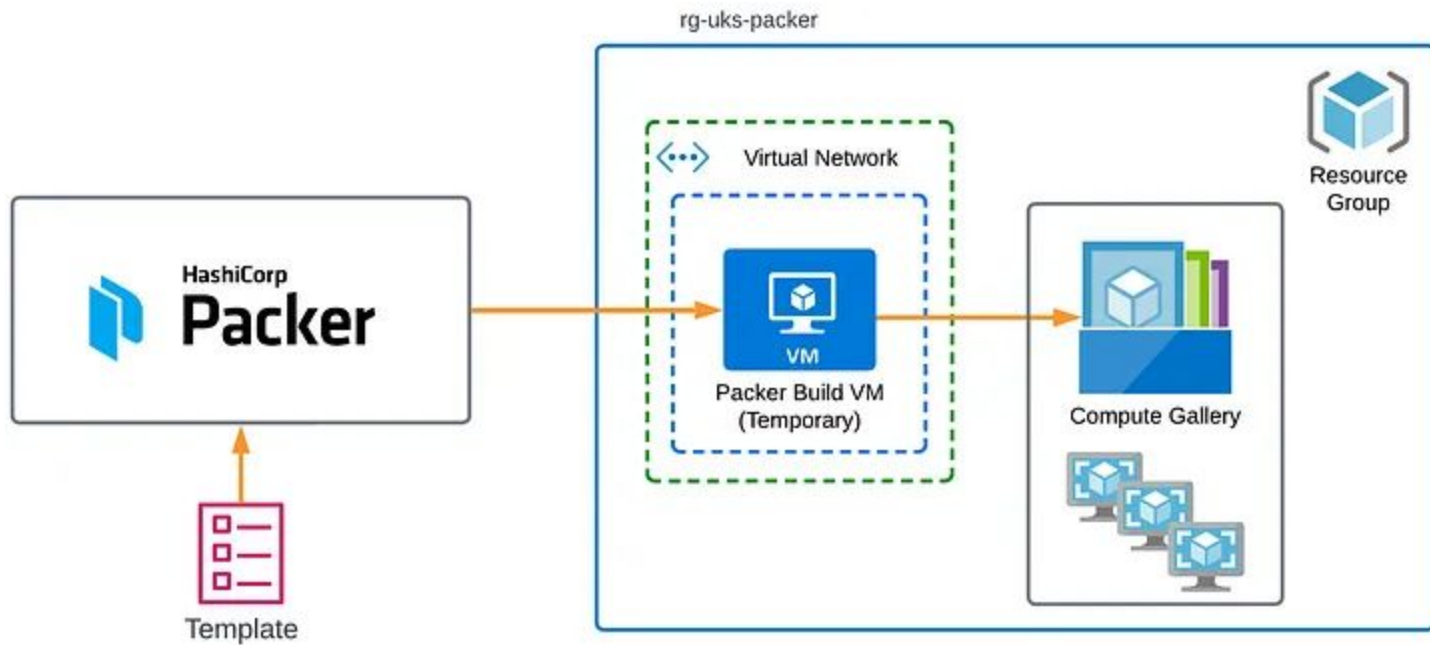
3- Reproducible Environments

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

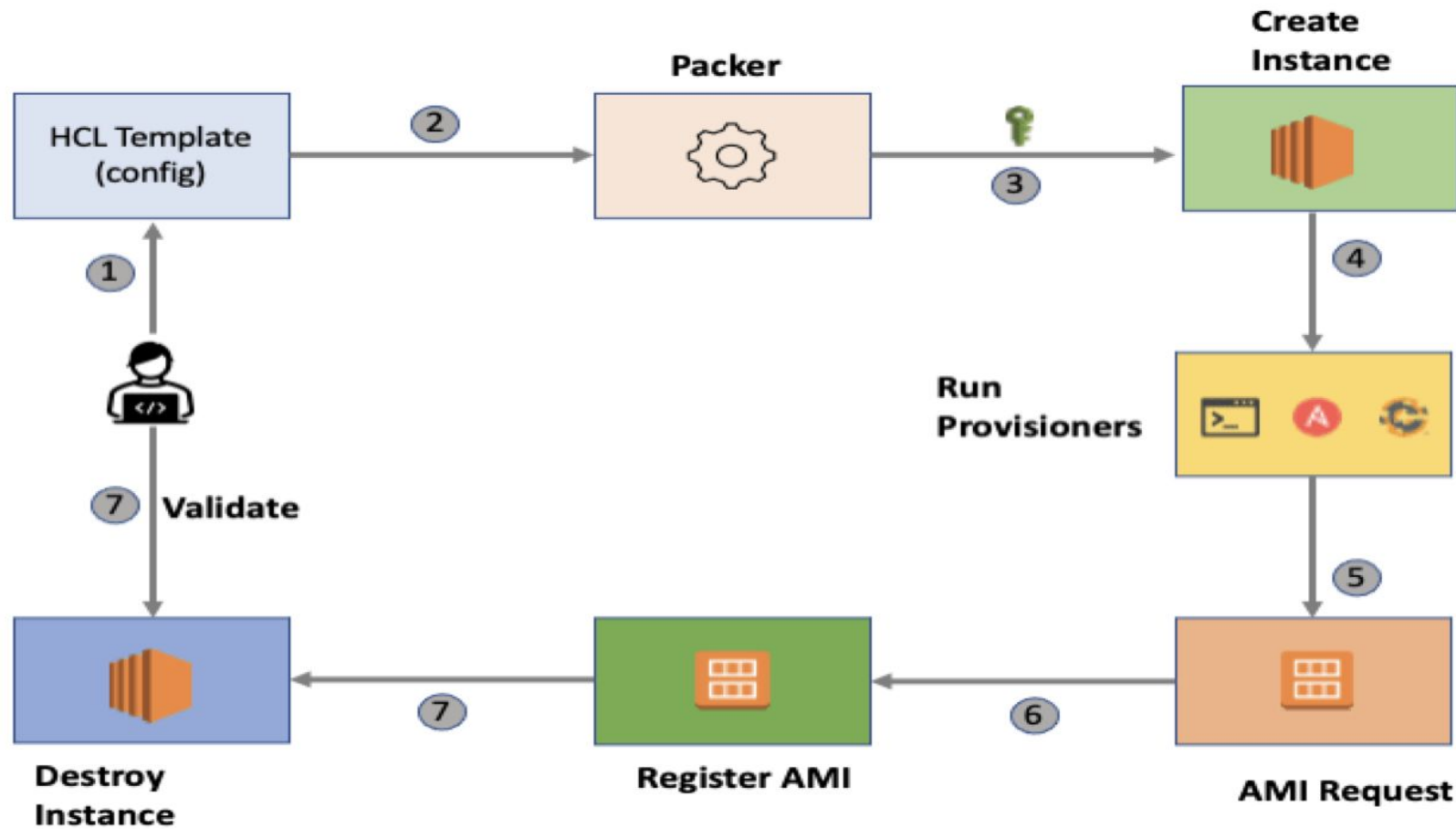
Core Concepts of Packer



Builder



Provisioners

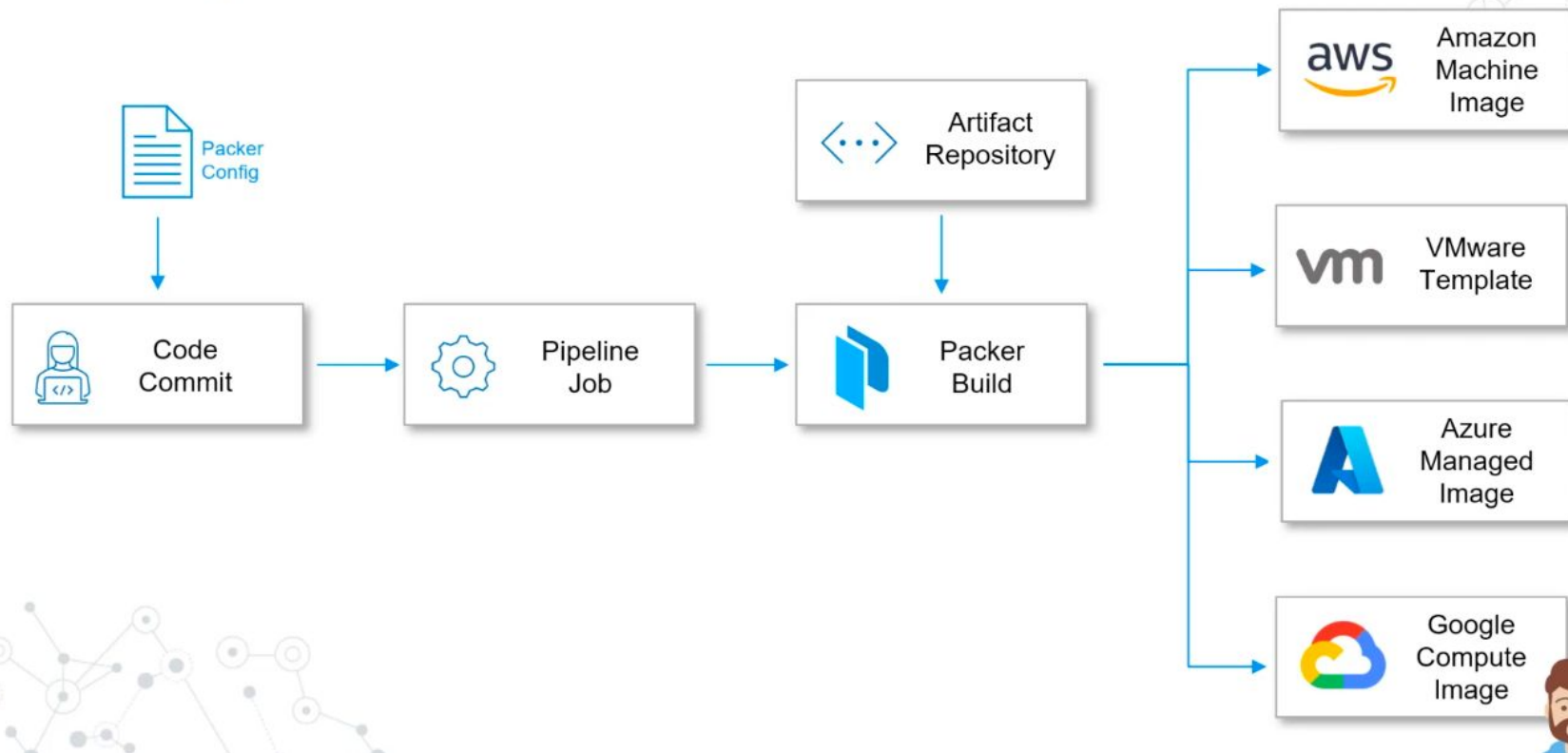


Creation of a Base AMI using Packer

- 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 Packer configuration files use a JSON/HCL format to define the image build process.	Builder Definition The builder section specifies the target platform, such as AWS, Azure, or GCP, and the necessary configuration options.	Provisioner Definition 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	<code>packer init</code>	Create a new Packer configuration file.
02	<code>packer validate</code>	Validate the syntax of a Packer configuration file.
03	<code>packer build</code>	Build one or more machine images from a Packer configuration file.
04	<code>packer inspect</code>	View the components of a Packer configuration file.
05	<code>packer console</code>	Launch an interactive Packer console for testing configurations.

Demo

Creating Custom AMI using packer with Github Action CI/CD

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

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

https://github.com/ankit20000/Custom_AMI_Packer

Step1: Clone/fork this repo to your github Account

← → ↻ 🔍 https://github.com/ankit20000/Custom_AMI_Packer

Gmail YouTube Maps vipin-k/ingress-co... vipin-k/Setup-NGI... New Tab How to use Googl... bnb New email - Ankit...

☰ GitHub ankit20000 / Custom_AMI_Packer

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

🔗 Custom_AMI_Packer Public Pin Unwatch 1

🔗 master 1 Branch 0 Tags 🔍 Go to file t Add file <> Code

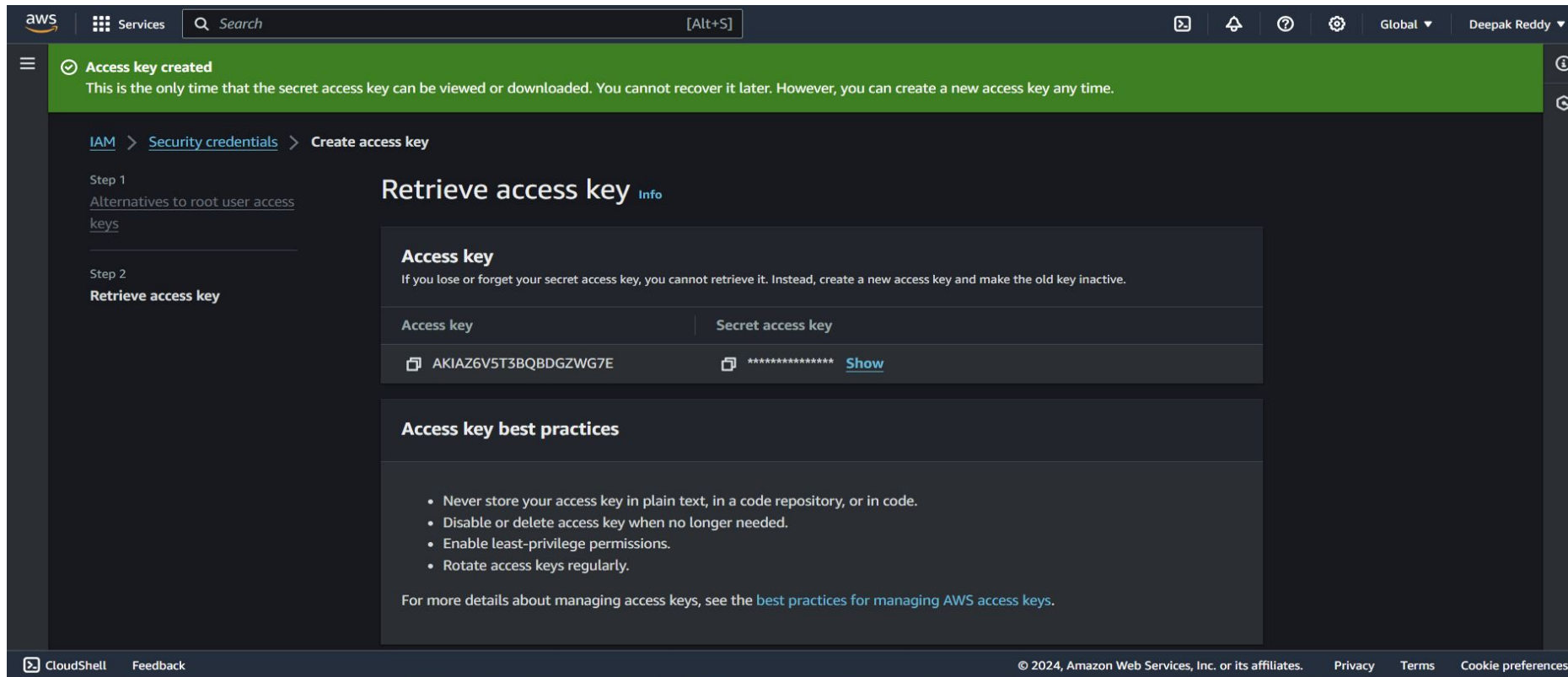
ankit20000 added 7abdebc · now 3 Commits

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

📖 README ✎

Custom_AMI_Packer

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





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	Secret access key
 AKIAZ6VST3BQBDGZWG7E	 ***** 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

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks
- Environments
- Codespaces
- Pages

Security

- Code security
- Deploy keys
- * Secrets and variables**
- Actions
- Codespaces
- Dependabot

passed to workflows that are triggered by a pull request from a fork.

Secrets Variables








Environment secrets

This environment has no secrets.



Manage environment secrets


Repository secrets





New repository secret









Name 	Last updated
 AWS_ACCESS_KEY	now  
 AWS_SECRET_KEY	now  

Step4- Now run the pipeline to build ami

 ankit20000 / Custom_AMI_Packer

Q Type  to search

+ 

<> Code  Issues  Pull requests  **Actions**  Projects  Wiki  Security  Insights  Settings

← Packer Build

 **added #1**

Cancel workflow

 Latest #3 ▾

...

 Summary

Jobs

 **packer**

Run details

 Usage

 Workflow file

packer

Started 12s ago

Q Search logs



- >  Set up job 0s
- >  Checkout code 0s
- ▼  **Install Packer** 8s

```
1 ▶ Run sudo apt-get update -y
9 Get:1 file:/etc/apt/apt-mirrors.txt Mirrorlist [142 B]
10 Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
11 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.

