# Packer Pipeline Documentation

# Table of Contents

1. Overview  
2. Prerequisites  
 2.1 Required IAM Permissions  
 2.2 Project and Network Configuration  
3. Packer Script Details  
 3.1 Plugins and Dependencies  
 3.2 Source Image Configuration  
 3.3 Provisioners (Hardening & Dynatrace Agent)  
 3.4 Image Encryption and Labels  
4. Wiz.io Agent Installation & Vulnerability Scanning  
5. GitHub Actions Workflow  
 5.1 Workflow Triggers  
 5.2 Authentication Setup  
 5.3 Matrix Strategy for OS Variants  
 5.4 Image Building with Packer  
 5.5 Wiz.io Vulnerability Scanning  
 5.6 Uploading Results to Cloud Storage  
 5.7 Cleanup of Old Images  
6. Validation & Deployment  
7. Screenshots & Logs

# 1. Overview

This document details the Packer pipeline that builds hardened OS images for RHEL, Ubuntu,   
and Windows. The pipeline is automated via GitHub Actions and integrates Wiz.io for vulnerability scanning.

# 2. Prerequisites

## 2.1 Required IAM Permissions

The following IAM permissions are required for the service account executing Packer:  
- Compute Image User (`roles/compute.imageUser`)  
- Compute Instance Admin (`roles/compute.instanceAdmin.v1`)  
- Cloud KMS CryptoKey Encrypter/Decrypter (`roles/cloudkms.cryptoKeyEncrypterDecrypter`)  
- Storage Object Admin (`roles/storage.objectAdmin`) for vulnerability report uploads.

## 2.2 Project and Network Configuration

- \*\*Project ID:\*\* `prj-ospacker-useast-dev-23295`  
- \*\*Subnetwork:\*\* `projects/prj-shrd-ntwk-3/regions/us-east4/subnetworks/sn-ue4-ospac-dev-1`  
- \*\*Zone:\*\* `us-east4-a`  
- \*\*Storage Bucket for Reports:\*\* `packer-images-bucket/vul\_report`

# 3. Packer Script Details

## 3.1 Plugins and Dependencies

The Packer script requires the HashiCorp `googlecompute` plugin for GCP image creation.

## 3.2 Source Image Configuration

- \*\*Base Image:\*\* `rhel-9` (latest non-deprecated image from `rhel-9` family)  
- \*\*Machine Type:\*\* `e2-small`  
- \*\*Internal Networking:\*\* Uses internal IP with IAP for security.

## 3.3 Provisioners (Hardening & Dynatrace Agent)

- \*\*Hardening Script (`rhel-hardening.sh`)\*\*:   
 - Disables root login  
 - Enforces SSH key-based authentication  
 - Applies OS patches  
 - Configures firewall & audit logging  
- \*\*Dynatrace Agent (`install\_dynatrace\_agent.sh`)\*\* (commented out):  
 - Downloads and installs the Dynatrace OneAgent.  
 - Uses `DYNATRACE\_API\_TOKEN` stored in GitHub secrets.

## 3.4 Image Encryption and Labels

- \*\*Encryption\*\*: Uses CMEK from Cloud KMS:  
 - Disk Encryption: `cmdk-key-ring/cmek-key`  
 - Image Encryption: `cmdk-key-ring/cmek-key`  
- \*\*Labels for Organization Policy Compliance\*\*:  
 - `appserviceid`: TBD  
 - `appservicename`: `gcp`  
 - `timestamp`: Auto-generated  
 - `iac`: `packer`  
 - `datatype`: TBD  
 - `costcenter`: TBD  
 - `tierid`: `tier-1`

# 4. Wiz.io Agent Installation & Vulnerability Scanning

- Downloads Wiz CLI and authenticates using `WIZ\_CLIENT\_ID` and `WIZ\_CLIENT\_SECRET`.  
- Scans newly created images for vulnerabilities.  
- Outputs results in SARIF format and uploads them to Google Cloud Storage.

# 5. GitHub Actions Workflow

## 5.1 Workflow Triggers

Pipeline runs on schedule (1st of every month), push events, pull requests, and manual triggers.

## 5.2 Authentication Setup

Uses Google Workload Identity Federation for authentication to avoid storing service account keys.

## 5.3 Matrix Strategy for OS Variants

Runs parallel jobs for RHEL, Ubuntu, and Windows, reducing build time.

## 5.4 Image Building with Packer

- Initializes Packer configurations (`packer init`).  
- Builds OS-specific images (`packer build`).  
- Logs output to `packer-${{ matrix.os }}.log`.

## 5.5 Wiz.io Vulnerability Scanning

- Extracts the created image name from logs.  
- Runs `wizcli vm-image scan` for vulnerability analysis.  
- Generates SARIF report for tracking security issues.

## 5.6 Uploading Results to Cloud Storage

Uploads vulnerability reports (`results-${{ matrix.os }}.sarif`) to `packer-images-bucket/vul\_report`.

## 5.7 Cleanup of Old Images

Runs `cleanup\_old\_images.py`, which removes outdated images to optimize storage costs.

# 6. Validation & Deployment

- \*\*Verification Steps:\*\*  
 - Ensure images boot successfully.  
 - Confirm hardening policies are enforced.  
 - Validate Wiz.io scan results (no critical vulnerabilities).  
- \*\*Deployment Workflow:\*\*  
 - Promote tested images from POC → dev → prod.

# 7. Screenshots & Logs

Include screenshots/logs from:  
- GitHub Actions execution.  
- Wiz.io vulnerability scan reports.  
- Packer build logs.