Desing a **production-grade Terraform + GitHub Actions pipeline** for provisioning a Linux VM on Azure. I'll break it down into **Terraform module design**, **secure SSH key handling**, and **GitHub Actions workflow**.

◆ Terraform Module Structure

We'll use a modular approach so the VM creation logic is reusable and maintainable.

Repo structure:

```
terraform/
  main.tf
  variables.tf
  outputs.tf
  providers.tf
  modules/
    vm/
      main.tf
      variables.tf
      outputs.tf
```

providers.tf

```
terraform {
  required_version = ">= 1.5.0"
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.0"
    }
  }
}

provider "azurerm" {
  features {}
}
```

main.tf

```
module "linux_vm" {
  source              = "./modules/vm"
  resource_group_name = var.resource_group_name
  location            = var.location
  vm_name             = var.vm_name
  admin_username      = var.admin_username
  public_key_path     = var.public_key_path
}
```

variables.tf

```
variable "resource_group_name" {}
variable "location" { default = "eastus" }
variable "vm_name" {}
variable "admin_username" { default = "azureuser" }
```

```
variable "public_key_path" {
  description = "Path to public SSH key"
}
```

◆ VM Module (`modules/vm/main.tf`)

```
resource "azurerm_resource_group" "rg" {
  name     = var.resource_group_name
  location = var.location
}

resource "azurerm_virtual_network" "vnet" {
  name                = "${var.vm_name}-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = var.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_subnet" "subnet" {
  name                 = "${var.vm_name}-subnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.1.0/24"]
}

resource "azurerm_network_interface" "nic" {
  name                = "${var.vm_name}-nic"
  location            = var.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name                          = "internal"
    subnet_id                     = azurerm_subnet.subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.vm_public_ip.id
  }
}

resource "azurerm_public_ip" "vm_public_ip" {
  name                = "${var.vm_name}-pip"
  location            = var.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method   = "Dynamic"
}

resource "azurerm_linux_virtual_machine" "vm" {
  name                = var.vm_name
  resource_group_name = azurerm_resource_group.rg.name
  location            = var.location
  size                = "Standard_B1s"
  admin_username      = var.admin_username
  network_interface_ids = [azurerm_network_interface.nic.id]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }
```

```
  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "20.04-LTS"
    version   = "latest"
  }

  admin_ssh_key {
    username   = var.admin_username
    public_key = file(var.public_key_path)
  }
}
```

◆ Secure SSH Key Handling

- **Do NOT commit `private.pem` or `public.pub` to GitHub.**
- Store keys **locally** and reference them via `file()` in Terraform.
- For CI/CD:
  - Use **GitHub Secrets** to store the **public key** (safe to share).
  - The **private key** stays only on developer machines for SSH access.
  - Example secret: `AZURE_SSH_PUBLIC_KEY`.

◆ GitHub Actions Workflow

`.github/workflows/terraform.yml`

```
name: Terraform Azure VM Deployment

on:
  push:
    branches: [ "main" ]
  pull_request:

jobs:
  terraform:
    runs-on: ubuntu-latest

    env:
      ARM_CLIENT_ID: ${{ secrets.AZURE_CLIENT_ID }}
      ARM_CLIENT_SECRET: ${{ secrets.AZURE_CLIENT_SECRET }}
      ARM_SUBSCRIPTION_ID: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
      ARM_TENANT_ID: ${{ secrets.AZURE_TENANT_ID }}

    steps:
      - name: Checkout repo
        uses: actions/checkout@v3

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2

      - name: Terraform Init
        run: terraform init
```

```
      - name: Terraform Validate
        run: terraform validate

      - name: Terraform Plan
        run: terraform plan -var="public_key_path=/tmp/public.pub"

      - name: Write SSH Public Key
        run: echo "${{ secrets.AZURE_SSH_PUBLIC_KEY }}" > /tmp/public.pub

      - name: Terraform Apply
        if: github.ref == 'refs/heads/main'
        run: terraform apply -auto-approve -
var="public_key_path=/tmp/public.pub"
```

◆ Key Points for Production-Grade Setup

- **Modules**: VM logic isolated in `modules/vm`.
- **Secrets**: Azure credentials + SSH public key stored in GitHub Secrets.
- **Private Key**: Never stored in repo or pipeline; only used locally for SSH.
- **Validation & Plan**: CI runs `terraform validate` and `terraform plan` before apply.
- **Branch Protection**: Apply only runs on `main`.

✅ This setup ensures **secure SSH handling**, **modular Terraform code**, and a **robust GitHub Actions pipeline** for production-grade Azure VM provisioning.

Would you like me to also add **remote state management** (e.g., Azure Storage backend) so multiple team members can collaborate safely?