Lab 02 — Packer Builds: Ubuntu Server & Rocky Linux

Objective

In this lab, you will:

- Use **Packer** to build reproducible VM templates for Ubuntu Server (cloud-init/autoinstall) and Rocky Linux (Kickstart).
- Parameterize build variables and enforce secure defaults (SSH key-only login, SELinux, firewall).
- Provision NGINX during the Packer build.
- Document ISO sourcing and checksum verification.
- Annotate every configuration file with comments.
- Reflect on the history and applications of cloud-init and Kickstart.
- Practice good repo hygiene to avoid oversized artifacts.

Learning Goals

- Understand how **cloud-init (NoCloud)** and **Kickstart** bootstrap OS installation.
- Gain hands-on experience with **parameterized variables vs. hardcoded values**.
- Use **SSH key authentication only** (no passwords).
- Require **SELinux enforcing** and **firewall enabled**.
- Safely handle **sensitive data** in projects.
- Annotate and cite official documentation for every directive and boot sequence.
- Build confidence with the Packer workflow (`init`, `fmt`, `validate`, `build`).

Repository Layout

Your repo should contain the following structure:

```
ubuntu-server/
  ubuntu-server.pkr.hcl
  variables.pkr.hcl
  ubuntu-server.auto.pkrvars.hcl
  http/
    user-data
    meta-data

rocky-server/
  rocky-server.pkr.hcl
  variables.pkr.hcl
  rocky-server.auto.pkrvars.hcl
  kickstart/
    ks.cfg
```

- The `.auto.pkrvars.hcl` files must be committed (exceptions are included in `.gitignore`).

- Secrets are parameterized and must be stored outside the repo.

Add this `.gitignore` at the repo root:

```
# Packer cache and outputs
packer_cache/
output-*/
*.ova
*.ovf
*.mf
*.log

# ISOs and large binaries
iso/
*.iso

# Secrets & local vars
*.pkrvars.hcl
*.auto.pkrvars.hcl
secrets/
*.key
*.pem

# EXCEPTIONS: allow these two auto var files to be committed
ubuntu-server/ubuntu-server.auto.pkrvars.hcl
rocky-server/rocky-server.auto.pkrvars.hcl
```

Step 1 — Verify Packer

1. Install the latest Packer from HashiCorp.
2. Run:

   ```
   packer -v
   ```

3. **Task:** Take a screenshot of the terminal showing the Packer version and include it in your Task Report.

Step 2 — Generate a Dedicated SSH Keypair

1. Create `~/keys` if it does not already exist.
2. Generate your keypair (must use the name `packer`):

   ```
   ssh-keygen -t ed25519 -C "packer@lab02" -f ~/keys/packer
   ```

   - Private key: `~/keys/packer`
   - Public key: `~/keys/packer.pub`
3. Keys must remain outside your repo.
4. **Task:** Take a screenshot of your terminal showing successful key generation and include it in your Task Report.

Step 3 — Variables & Auto Vars Files

1. Each build directory contains a `variables.pkr.hcl` file with variable names only.
2. Create:
   - `ubuntu-server/ubuntu-server.auto.pkrvars.hcl`
   - `rocky-server/rocky-server.auto.pkrvars.hcl`
3. Populate each with:
   - Official ISO URL (official Ubuntu/Rocky sources).
   - ISO SHA256 checksum (verify locally).
   - VM name, CPU, RAM, disk size.
   - Public key contents from `~/keys/packer.pub`.
4. Verify each ISO with `sha256sum` (Linux/macOS) or equivalent.
5. **Task:**
   - Take a screenshot of the terminal showing checksum verification.
   - Take a screenshot of each `.auto.pkrvars.hcl` open in VS Code.
   - Paste the raw checksum output into your Task Report with a short explanation of how you verified it.

Step 4 — Autoinstall & Kickstart Configurations

1. **Requirement:** Every line in `http/user-data`, `http/meta-data`, and `kickstart/ks.cfg` must be documented with a comment explaining its purpose.

   - Place comments inline whenever possible.
   - Some code blocks may fail if comments interrupt the script. In those cases, place your explanatory comment directly **above the block** instead of inline.

2. **Extended explanations (Reflection Essay):**

   - **Ubuntu:** `autoinstall`, `identity`, `late-commands`, boot command sequence (`ds=nocloud-net`).
   - **Rocky:** `install`/`url`, `lang`, `keyboard`, `timezone`, `selinux`, `firewall`, `%packages`, `%post`.

3. **Task:** Take screenshots of each config file open in VS Code with comments visible and include them in your Task Report.

Step 5 — Provisioners (Research Required)

1. Provision **NGINX** during the Packer build for both Ubuntu and Rocky.
2. You must research the commands yourself. At minimum:

    - Update package sources.
    - Install NGINX.
    - Enable and start NGINX at boot.
3. **Task Report requirements:**
    - Provide the exact commands you used for Ubuntu and Rocky.
    - Explain in one sentence what each command does.
    - Note differences between Ubuntu and Rocky.
4. **Task:** Take a screenshot of your provisioner code blocks in VS Code and include it in your Task Report.

Step 6 — Conceptual Toggles

Conceptual toggles are required configuration decisions that affect security and system behavior. You must decide, document, and justify each.

- SSH authentication = key-only.
- Parameterized vs hardcoded values (explain your design).
- SELinux = enforcing (Kickstart).
- Firewall = enabled (Kickstart).
- Update policy = your choice (security-only or manual, but justify).

Step 7 — Build Workflow

1. Navigate into each build directory (`ubuntu-server/` and `rocky-server/`).
2. Run:

```
packer init .
packer fmt .
packer validate .
packer build .
```

3. **Task:** Take screenshots of each command run successfully (Ubuntu and Rocky) and include them in your Task Report.

Step 8 — Import and Verify

1. Import the resulting `.ovf` into VirtualBox (**File → Import Appliance…**).
2. **Task:** Take a screenshot of the VM configuration screen in VirtualBox and include it in your Task Report.
3. Boot the VM and connect via SSH:

```
ssh -i ~/keys/packer packer@<vm-ip>
```

4. In VirtualBox, go to **VM Settings → Network → Advanced → Port Forwarding**.
    - Forward host port `8080` (or another unused port) to guest port `80`.
    - From your host browser, visit:

      ```
      http://127.0.0.1:8080/
      ```

    - This will display the NGINX landing page.
5. **Task:** Take a screenshot of the NGINX landing page for each VM and include it in your Task Report.

Step 9 — Repo Hygiene

Before committing and pushing, move these files out of your repo to a safe location:

- `*.ovf`
- `*.ova`
- `*.vmdk`
- `*.iso`

Warning: Pushing will fail if OVF, VMDK, or ISO files remain in your repo—they are too large for GitLab.

Deliverables

Task Report (one per team)

Embed all required screenshots directly in the Task Report.

Include:

- Screenshots from Steps 1–8.
- ISO URL + checksum verification (raw output + explanation).
- Provisioner commands with explanations.
- Conceptual toggle justifications.
- Security reasoning for SSH keys only.
- Citations:
    - 4 from Ubuntu/cloud-init docs.
    - 4 from Rocky/Kickstart docs.
    - ISO download and checksum sources.
    - At least 1 AI prompt + link to the conversation.

Reflection Essay (4 pages, double-spaced, 12pt Times/Arial)

Address:

- Origins and design goals of cloud-init and Kickstart.
- Ecosystem history (who maintains each, intended environments, how adoption shaped design).
- Practical applications across on-prem, hybrid, and cloud.
- Portability, bootstrap sequencing, day-0 vs day-1:

    - Day-0 = initial provisioning (partitioning, locale, user, keys).
    - Day-1 = customizations post-install (packages, updates, services).
    - Bootstrapping = automated steps that transform a generic installer ISO into a configured VM.
- Extended explanations of directives listed in Step 4.
- Reflection on AI usage: what helped, what misled.

Grading Criteria

- **Build correctness (Ubuntu + Rocky) — 25%**
  Packer builds succeed; OVF imports; SSH works; NGINX reachable.

- **Line-by-line commentary — 30%**
  Every line or section has a meaningful comment.

    - ✅ Example: `# This section defines the default user and applies your SSH public key`
    - ❌ Not meaningful: `# config line`

- **Conceptual decisions & security reasoning — 10%**
  Clear justification for SSH keys only, parameterization, SELinux, firewall, update policy.

- **Research + Reflection (combined) — 25%**
  Citations meet minimums and reflection essay covers all required elements.

- **Professionalism & evidence — 10%**
  Report is grammatically correct, logically structured, and includes all required screenshots. Repo hygiene followed.

Submission Instructions

- Submit **one PDF per team** containing the Task Report and Reflection Essay.
- Verify your repo is shared with the professor and TA.
- Submit the **repo link in Canvas** along with your PDF.
- Repo must include configs, vars, and provisioners, but not OVF/VMDK/ISO artifacts.

Common Pitfalls

- Pushing will fail if OVF, VMDK, or ISO files remain in your repo—they are too large for GitLab.
- Missing comments on config lines or sections.
- Submitting two reports instead of one.
- Forgetting to enable NGINX at boot.
- Omitting an AI prompt and link.