# Essential Eight on Linux, Part 8 of 8: Regular Backups on Ubuntu 26.04 LTS

May 5, 2026 / Gavin Jackson

essential-eight asd ism ubuntu linux backups restic zfs security

Regular backups are the least glamorous of the Essential Eight mitigations and one of the most important.

They are also where Linux teams sometimes get overconfident. Plenty of Ubuntu environments have backups. Fewer have backups that are protected from the same compromise path, regularly tested, and structured to recover whole services rather than just files.

That difference matters.

## What ASD is trying to achieve

The security purpose of the backup mitigation is not just retention. It is recovery under hostile conditions:

- ransomware
- destructive admin mistakes
- credential compromise
- infrastructure failure

For Ubuntu 26.04 LTS, the question is not "can I copy the files somewhere?" It is "can I restore the system or service safely and predictably when the primary environment is no longer trustworthy?"

## Ubuntu 26.04 LTS reference implementation

### Resolute Raccoon highlights

There is no single headline backup feature in Resolute Raccoon that changes this mitigation the way Livepatch or authd change other parts of the series.

The relevant 26.04 improvements are indirect:

- **TPM-backed full-disk encryption** is generally available in the installer

That helps protect data on lost or stolen endpoints, but it does not replace backup discipline. Backups still need separate credentials, separate storage, tested restores, and protection from the production environment being compromised.

### 1. Back up data, configuration, and rebuild context

For Linux servers, file backup alone is rarely enough.

You usually need:

- application data
- system configuration
- secrets management approach
- package and image definitions
- infrastructure-as-code or deployment manifests
- database-consistent backup procedures

If the restore requires a departed engineer's memory, the control is weaker than it looks.

> ***Rebuild, don't restore***
>
> *Not every Linux server needs a traditional file-level backup. If a host has no unique persistent data and can be rebuilt predictably inside the required recovery window, the stronger pattern may be* ***rebuild, not restore****.*
>
> *This fits well for things like recursive DNS resolvers, secondary DNS servers where the zone source of truth lives elsewhere, jump hosts, static web servers, reverse proxies, load balancers, caching proxies, NTP servers, CI runners, Kubernetes worker nodes, and container hosts. The practical pattern is* ***PXE boot plus Ubuntu autoinstall****, followed by* ***Ansible*** *or another orchestration tool to apply roles, hardening, monitoring, logging, and application configuration.*
>
> *The thing you must protect is the source of truth: Git repositories, Ansible inventory, autoinstall data, package mirrors, secrets management, certificates, DNS/IPAM records, and any content or configuration that is not generated automatically. If that source of truth is gone, the server was not really rebuildable.*
>
> *This lines up with modern container patterns. Stateless workloads should usually be redeployed from known images and configuration, not restored like pets. Persistent volumes, databases, authoritative data, audit logs, and anything users or systems write to over time still need a backup and restore plan.*

## 2. Use native Linux backup tooling that is simple and testable

Ubuntu works well with tools such as:

- `restic`
- `borgbackup`
- filesystem snapshots with ZFS, Btrfs, or LVM where appropriate

For many teams, `restic` or `borgbackup` is a good fit for backing up configuration, user data, small services, and file-based workloads. They are less appropriate as the only backup mechanism for large virtualised server fleets or databases unless you also handle application consistency.

A practical pattern is:

- use database-native dumps, WAL shipping, or application-aware backup hooks for stateful services
- use `restic` or `borgbackup` for files, configuration, and smaller workloads
- use hypervisor-aware backup for VM recovery where the Linux server is virtualised
- use orchestration and rebuild testing for genuinely stateless servers
- document which restore path applies to each service

> ***Hard links and cloud paranoia***
>
> *I still have a soft spot for two older backup tools: **dirvish** and **tarsnap**.*
>
> *Dirvish was clever in a very Unix way. It used `rsync` and hard-linked snapshot trees so each backup could look like a full filesystem image without storing a complete duplicate every time. For file-based backups, that was a beautifully practical trick: simple to browse, simple to restore from, and much more space-efficient than it first appeared.*
>
> *Tarsnap felt ahead of its time when I was working on FreeBSD systems. It offered encrypted, deduplicated, cloud-based backups with a familiar tar-like command line long before "secure cloud backup" felt like a normal infrastructure pattern. It was opinionated, cautious, and very FreeBSD in spirit.*
>
> *I would not necessarily choose either as the default enterprise pattern for Ubuntu 26.04 today, but the design lessons still hold up: make backups space-efficient, keep restores simple, encrypt before data leaves your control, and do not confuse a clever tool with a tested recovery process.*

## 3. Make backup storage hard to tamper with

This is where a lot of backup strategies fail under real attack.

Good Linux backup design should include:

- separate credentials for backup administration
- immutable or write-once protections where possible
- network separation between production systems and backup control paths
- encryption at rest and in transit
- limited direct access from general administrators
- monitoring for failed jobs, missed schedules, and repository verification failures

If an attacker can compromise a Linux server and immediately delete or encrypt its backups with the same credentials, you do not really have a resilient control.

## 4. Snapshot fast, restore deliberately

Snapshots are useful, especially on Ubuntu systems using ZFS or LVM, but do not confuse snapshots with a full backup strategy.

Snapshots help with:

- fast rollback
- point-in-time recovery
- operational mistakes

But they usually still depend on the same platform or storage domain unless replicated elsewhere.

### 5. Test service recovery, not just file retrieval

The meaningful test is not "I restored one file from last Tuesday."

The meaningful test is more like:

- can the application be rebuilt?
- can the database be restored consistently?
- can the host or container platform be brought back in order?
- are credentials and dependencies handled safely during recovery?
- does the restore meet the service's recovery time and recovery point expectations?

That is the difference between backups and recoverability.

### 6. Remember that many Linux servers are really virtual machines

In a lot of enterprise environments, "Linux server backup" is really "virtual infrastructure backup with Linux workloads inside it."

That changes the design conversation a bit.

If your Ubuntu servers are running on **VMware vSphere**, the backup strategy often needs to cover:

- image-level VM backup
- application-consistent guest processing where required
- fast whole-VM recovery
- file-level or item-level recovery for targeted cases
- retention and immutability at the backup platform

For many enterprises, this is where a platform-aware backup product is more realistic than relying purely on in-guest scripts.

The important caveat is that image-level VM backup is not the same thing as application-consistent backup. For databases and directory services, you still need either guest processing that quiesces the workload correctly or an application-native backup path that you have actually restored from.

### 7. vSphere patterns: Bareos and Veeam

If your Linux estate sits mostly on vSphere, two patterns are worth calling out.

**Bareos** is interesting if you want an open-source oriented backup platform and are comfortable investing engineering effort into the backup stack. It is worth a look for teams that want more control over how backup is built and operated, especially in mixed Linux-heavy environments.

**Veeam** is the more mainstream enterprise answer. It is widely used in VMware environments for a reason:

- strong support for vSphere backup and restore
- image-level recovery workflows
- broader enterprise operational maturity
- options such as hardened Linux repositories and immutability

If I were writing this for a typical enterprise with a large vSphere footprint, I would be very comfortable calling **Veeam** the default commercial reference point.

### 8. Proxmox VE and Proxmox Backup Server as an open source path

If the virtualisation stack is **Proxmox VE**, the natural companion is **Proxmox Backup Server (PBS)**.

That pairing is worth mentioning because it is one of the cleaner open-source infrastructure stories available right now:

- VM and container aware backups when integrated with Proxmox VE
- deduplication
- encryption and integrity protections
- replication and sync options
- a design that is purpose-built for the Proxmox ecosystem

I want to be careful here though: I have **not** yet used Proxmox Backup Server directly in anger. It is on my list to evaluate properly in my test environment, because on paper it looks like a very credible open-source option for Proxmox-based virtual estates. I would not position it as a drop-in answer for VMware environments.

That is exactly the sort of thing a home lab is good for. I wrote more about that here: Why the Home Lab Still Matters for IT Professionals.

## ISM control mapping

The October 2024 Essential Eight to ISM mapping links this mitigation to these controls:

| ISM control | E8 requirement in plain English | Ubuntu 26.04 implementation |
|---|---|---|
| ISM-1511 | Backups are performed and retained according to business criticality and continuity requirements. | Define backup scope per service: data, application state, configuration, package/image definitions, secrets handling, and retention. |
| ISM-1810 | Backups are synchronised so restoration can occur to a common point in time. | Coordinate VM snapshots, database dumps, WAL shipping, and application quiescing so restored components line up. |
| ISM-1811 | Backups are retained securely and resiliently. | Use separate backup credentials, separated storage, encryption, immutable retention where available, and monitoring for failed jobs. |
| ISM-1515 | Restoration from backups is tested as part of disaster recovery exercises. | Run restore exercises that rebuild the service, time the restore, and check data consistency. |
| ISM-1812 | Unprivileged users cannot access backups belonging to other users. | Do not expose backup repositories as shared mounts. Use per-service accounts, ACLs, and restore workflows instead of direct user access. |
| ISM-1813 | Unprivileged users cannot access their own backups at higher maturity levels. | Avoid user self-service access to raw repositories. Provide controlled restore requests or delegated restore portals. |
| ISM-1705 | Privileged users, except backup administrators, cannot access backups belonging to other users. | Separate Linux system administration from backup administration. Domain, sudo, or vCenter admin should not automatically grant backup repository access. |
| ISM-1706 | Privileged users, except backup administrators, cannot access their own backups at higher maturity levels. | Keep privileged recovery through the backup platform workflow rather than direct repository browsing. |
| ISM-1814 | Unprivileged users are prevented from modifying or deleting backups. | Use append-only repository modes, object lock, retention lock, or storage-side WORM controls where supported. |
| ISM-1707 | Privileged users, except backup administrators, are prevented from modifying or deleting backups. | Use backup RBAC, separate admin paths, hardened repositories, and storage immutability so ordinary admins cannot destroy backups. |
| ISM-1708 | Backup administrator accounts are prevented from modifying or deleting backups during the retention period. | Use immutability that is enforced by the repository or storage platform, not just by backup-console permissions. |

## Where Linux teams should be careful

Common weak patterns on Ubuntu estates include:

- backups running with overly broad root credentials
- backup repositories mounted directly into production servers
- no immutability or retention lock
- snapshots mistaken for offline recovery

- no documented owner for application-consistent database recovery
- "rebuildable" servers that still contain hand-edited local configuration
- restore testing skipped because "the job says success"

None of those survive a serious ransomware event gracefully.

## Commercial or enterprise-friendly additions

Linux-native backup tooling is good enough for many environments, so this is not a mitigation where you must rush to a commercial product.

Still, if you need enterprise orchestration, policy, or broader platform coverage, products such as **Veeam**, **Commvault**, or **Rubrik** are worth evaluating alongside Linux-native tools. In VMware-heavy environments, I would especially call out **Veeam** as a very common and practical answer.

If open source and platform ownership are stronger priorities, **Bareos** and **Proxmox Backup Server** are also worth looking at, with the important caveat that the fit depends heavily on the hypervisor layer and your willingness to own more of the engineering.

The important thing is not the brand. It is separation, recoverability, and proof.

## A practical Ubuntu backup pattern

For an Ubuntu 26.04 reference design, I would aim for:

- `restic`, `borgbackup`, or another well-understood Linux backup tool for files and configuration
- database-native backup or application-aware hooks for stateful services
- PXE boot, Ubuntu autoinstall, and Ansible for servers that are genuinely stateless and rebuildable
- protected source-of-truth repositories, inventories, secrets, certificates, and IPAM/DNS records
- hypervisor-aware backup for virtualised Linux servers
- Veeam for enterprise vSphere estates, or Bareos where an open-source oriented stack is a better fit
- Proxmox Backup Server where Proxmox VE is the virtualisation platform
- encrypted backups to object storage or a hardened repository
- immutable retention where the storage platform supports it
- snapshot capability for fast operational recovery
- separate admin paths for backup infrastructure
- regular restore exercises for critical services, including timing the restore and checking data consistency

That is a much stronger control than a beautifully green dashboard nobody has ever tried to restore from.

## The bottom line

Regular backups on Ubuntu 26.04 LTS are not hard to implement. Resilient backups are harder.

Use Linux-native tooling where it fits, but do not ignore the hypervisor layer if most of your Linux servers are virtual machines. For many enterprises that means a vSphere-aware platform such as Veeam, and for more open-source oriented environments it may mean evaluating Bareos or Proxmox Backup Server. Isolate the backup trust boundary, test restores like they matter, and design for service recovery instead of checkbox retention. That is the version of this Essential Eight mitigation that actually holds up under pressure.

## References


- ASD Essential Eight maturity model and ISM mapping (October 2024)
- Canonical releases Ubuntu 26.04 LTS Resolute Raccoon
- Setup a ZFS storage pool
- restic
- BorgBackup
- Dirvish
- Tarsnap
- Bareos virtual infrastructure backup
- Veeam VMware vSphere support
- Veeam Linux hardened repository
- Proxmox Backup Server
- Proxmox Backup Server documentation



Downloaded from https://www.gavinj.net/post/essential-eight-linux-regular-backups
Generated November 14, 2026. Copyright Gavin Jackson. All rights reserved.