PERCONA

SOVEREIGN DEPENDENCY AUDIT TEMPLATE

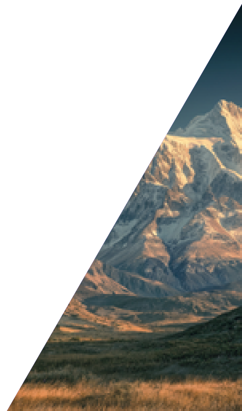
Identifying Technical Cliffs in Database Portability

Use this template to audit your current database environment. This audit identifies where proprietary vendor choices create “technical cliffs” that prevent you from moving workloads between cloud providers, on-premises servers, or Kubernetes clusters.

Part 1: Engine and Binary Sovereignty

Goal: Determine if your database engine is a standard, portable version or a proprietary vendor fork.

- **Upstream Compatibility:** Are you using a standard upstream version of the database (e.g., Percona Distribution for MySQL) or a vendor-specific fork (e.g., Amazon Aurora, Google AlloyDB)?
- **Binary Portability:** Can the exact same database binary run on a standard Linux server, a virtual machine, and a container?
- **Version Parity:** Does the vendor allow you to pin specific versions, or are you forced into “auto-upgrades” that could break compatibility with other environments?
- **Extension Audit:** Are you using proprietary extensions (e.g., specific cloud-only plugins for PostgreSQL) that do not exist in the open source community?



Part 2: Operating System and Environment Standardization

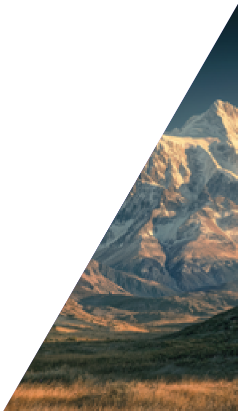
Goal: Ensure the underlying host environment is not a source of lock-in.

- **OS Distribution:** Is the database running on a standard, portable Linux distribution (e.g., Ubuntu, RHEL, AlmaLinux) or a vendor-locked operating system?
- **Kernel Dependencies:** Does the database require specific, proprietary kernel modifications provided only by the cloud vendor?
- **Filesystem and Storage:** Does the database rely on proprietary storage APIs, or does it use standard POSIX-compliant filesystems that can be replicated on any standard compute node?
- **Binary Source:** Can you verify the provenance of the binaries? Are they signed and auditable by your security team?

Part 3: Operational Control Plane Audit

Goal: Evaluate if your automation can move with your data.

- **Provisioning Logic:** Is your scaling and provisioning logic trapped in a vendor-specific console, or is it defined in portable code (e.g., Terraform or Ansible)?
- **Automation Compatibility:** Does your automation support multiple architectures? To achieve true sovereignty, your organization should be able to use operators, use servers, or use both without rewriting your operational playbooks.
- **Secret Management:** Are your database credentials locked in a vendor-specific KMS, or are you using a portable solution like HashiCorp Vault?
- **Backup Portability:** Are backups stored in a proprietary format that requires the vendor's tools to restore, or are they standard physical/logical backups that can be restored on any standard server?



Part 4: Percona Sovereignty Check

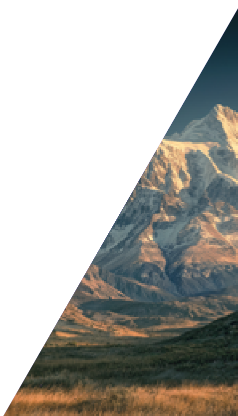
Percona helps you bridge the gap between "Convenience" and "Sovereignty" by providing a consistent software layer.

- **The Percona Distribution:** Our distributions for MySQL, PostgreSQL, and MongoDB are identical across all environments. This ensures that the binary you run on a local server is the same one running in your cloud-native environment.
- **The "Both" Strategy:** We provide the tools to let you use operators, use servers, or use both within the same infrastructure. This flexibility prevents architectural lock-in.
- **PMM Integration:** Percona Monitoring and Management provides a single pane of glass for telemetry that remains under your control, regardless of where the database is hosted.

Audit Summary: The Sovereignty Gap

After completing this audit, list any "Critical Dependencies" found. A dependency is considered critical if moving the workload would require:

1. Changing the database engine version or fork.
2. Rewriting the automation/provisioning scripts.
3. Changing the underlying Operating System or binary source.



READY TO CLOSE THE GAP?

Contact Percona to review your audit results and begin your 90-day transition to a sovereign stack.

CONTACT US