Comprehensive Requirements and Implementation Specification for Bacula-Based Libvirt Virtual Machine Protection on Red Hat Enterprise Linux 10

The integrity of a research lab's infrastructure is fundamentally tied to the resilience of its core services, particularly those facilitating remote access for a broad engineering base. Within an environment where over 400 engineers rely on virtual private network (VPN) access for daily operations, the virtualization layer must be supported by a data protection strategy that is both robust and transparent. The following document provides an exhaustive technical specification and project plan for utilizing the Bacula Community Edition in conjunction with the Bacularis web interface to protect libvirt-managed virtual machines (VMs) hosted on Red Hat Enterprise Linux (RHEL) 10. This specification addresses the critical needs for non-disruptive live backups, comprehensive metadata preservation for seamless re-instantiation, and the implementation of an automated, idempotent management framework.

Infrastructure Context and Virtualization Baseline

The transition to Red Hat Enterprise Linux 10 as the primary hypervisor platform necessitates a shift in how virtualization resources and containerized services are managed. RHEL 10 removes the legacy runc container runtime in favor of crun and fully transitions the container networking stack from CNI to the netavark backend. In this environment, Podman serves as the primary, native container engine, and all containerized backup components must be compatible with these modern backends.

The lab environment consists of approximately 12 to 20 virtual machines. While these VMs are not characterized by high CPU, memory, or I/O utilization, their availability is essential. Any failure in the backup implementation that causes service interruption could disconnect hundreds of engineers from the lab's infrastructure. Consequently, the backup solution must prioritize "zero-risk" operations, ensuring that the act of backing up data never compromises the stability of the running domain.

Virtual Machine Inventory and Criticality Mapping

The strategy for scheduling and retention is predicated on a granular understanding of the VM roles within the lab. The following table illustrates the classification of these assets.

VM Role	Criticality	Recommended Frequency	Retention Policy
VPN Access Gateway	Tier 0 - Mission Critical	Weekly Full / Daily Inc	30 Days
Authentication/LDAP	Tier 0 - Mission Critical	Weekly Full / Daily Inc	30 Days
Lab Management Tools	Tier 1 - Business Essential	Monthly Full / Weekly Inc	60 Days
Development Workstations	Tier 2 - Supportive	Monthly Full	30 Days
Legacy Infrastructure	Tier 3 - Archival	One-off Full	Indefinite

The total data footprint for these 20 VMs is expected to be manageable within a standard Network File System (NFS) storage target, provided that the Bacula Storage Daemon is optimized for network-attached storage throughput.

Libvirt State Capture and Metadata Requirements

The primary technical requirement for a "full virtual machine backup" in this environment extends beyond simple disk image copying. To properly re-instantiate a virtual machine with its original virtual interfaces and drive attachments, the solution must capture the complete operational context of the domain.

Metadata Extraction and Domain XML Elements

The libvirt domain XML is the definitive source of truth for a VM's hardware configuration. A backup that omits this XML is insufficient for disaster recovery because it fails to preserve the mapping between virtual and physical resources. The metadata capture process must utilize the virsh dumpxml command to export the current state of the domain into a structured format that can be stored alongside the disk images.

XML Element	Importance for Re-instantiation	Risk of Omission
<uuid>	Unique identifier for the domain within libvirt.	Collision with new VMs if not preserved.
<mac address>	Preserves network identity for DHCP and VPN.	Loss of connectivity or IP address change.
<disk source>	Defines the path and format of backing files.	Manual re-mapping of drives required.
<interface type>	Defines bridge, macvtap, or virtual network links.	Loss of network topology and VLAN tagging.
<nvram>	Stores UEFI/BIOS variables for secure boot.	VM may fail to boot if using UEFI.

The automated backup script must ensure that for every VM, the XML metadata is extracted at the exact moment the backup begins. This ensures that any "hot-plugged" devices or configuration changes made just prior to the backup are captured.

Live Backup Mechanics without Service Interruption

The requirement for "zero risk" and "no interruption" dictates a live backup strategy. Modern libvirt versions support sophisticated methods such as external snapshots and the newer virDomainBackupBegin API.

External Snapshot and Active Block-Commit

This method is the most reliable for the Bacula Community Edition. It involves creating a temporary QCOW2 overlay (snapshot) to track live writes while the original backing file is read-only and available for Bacula to copy to the NFS target.

1. **Quiescence**: The QEMU Guest Agent (QGA) must be installed and enabled in the guest OS. When the backup starts, libvirt issues a fs-freeze via the QGA to flush the guest's I/O buffers to disk.
2. **Overlay Creation**: A command such as virsh snapshot-create-as --disk-only --quiesce is executed. This pivots the live VM to write to a new overlay file.
3. **Data Transfer**: Bacula's File Daemon copies the now-static original disk image (the backing file) to the NFS storage.
4. **Live Merge**: Once the transfer is complete, virsh blockcommit is used to merge the data from the temporary overlay back into the original disk image. The VM is then pivoted back to the primary disk, and the overlay is deleted.

This approach satisfies the user's requirement that the VM "not have to shutdown," as the transition is handled at the hypervisor level with minimal latency.

Bacula Architecture and Bacularis Integration

The proposed solution utilizes a multi-daemon architecture managing approximately 20 VMs. A centralized Bacula Director (DIR) manages schedules and the PostgreSQL catalog, while individual File Daemons (FD) run on the RHEL 10 hypervisors to handle data extraction.

Component Deployment and Podman Topology

The preference for containerized deployment will be realized using Podman. Bacularis versions 5.10.0 and later specifically include support for Podman and "minimal OS" environments that lack standard system tools like systemd.

Bacula Component	Deployment Method	Connection Port
Director (DIR)	Podman Container (rootless)	9101 (Inbound)
Storage Daemon (SD)	Podman Container with NFS mount	9103 (Inbound)

File Daemon (FD)	Native RHEL 10 Service	9102 (Inbound)
PostgreSQL DB	Podman Container (Persistent Vol)	5432 (Inbound)
Bacularis Web	Podman Container	80/443 (Inbound)
Bacularis API	Podman Container	9097 (Inbound)

Containers will utilize the netavark network backend. For persistent storage, Podman named volumes or bind mounts will be used, with care taken to ensure proper UID/GID mapping for rootless operation. Bacularis will connect to the local API instance, providing a unified management interface for both scheduled and one-off backup tasks.

Bacularis Web Interface and User Access

Bacularis provides the user-facing portal for managing the lab's backups.

1. **Authentication**: Bacularis will be configured with simple password authentication for local user accounts. For enhanced security, TOTP 2FA or FIDO U2F can be enabled.
2. **API Management**: The web interface communicates with the Bacularis API via HTTP Basic auth or OAuth2 secure tokens.
3. **One-Off Backups**: Users can initiate immediate backups through the "Run" wizard, selecting the specific VM and backup level (e.g., Full).

Storage Engineering: NFS Reliability on RHEL 10

The target for backup data is an NFS-mounted disk. Backing up large VM images to network storage requires tuning to avoid timeouts.

NFS Mount Strategy for the Storage Daemon

The Storage Daemon container must have persistent, reliable access to the NFS mount.

- **Mount Options**: Utilizing "soft" mounts with appropriate timeo and retrans parameters ensures that transient network issues do not cause permanent hangs in the Bacula process.

- **Sparse File Support**: Bacula will be configured to maintain the sparseness of QCOW2 images to save network bandwidth and storage space.
- **Volume Sizing**: The system will use multiple smaller File volumes (e.g., 50GB) within the NFS mount to limit the blast radius of potential file corruption.

Volume Retention and Media Management

Bacula's Pool resources will define retention periods for weekly and monthly cycles. For 20 VMs (~100GB each) with a 30-day retention policy, approximately 6TB of usable NFS storage is required.

Automation and Scripting Framework

The automation framework ensures idempotent installation and self-healing operations.

Idempotent Podman-Based Installation

The installation script will automate the setup of the Bacula/Bacularis stack on RHEL 10.

1. **Podman Provisioning**: Ensure podman and netavark are active.
2. **Container Deployment**: Pull and run the Bacularis and Bacula containers using podman run or podman-compose, passing environment variables for configuration.
3. **Service Integration**: Use podman generate systemd to create unit files that allow systemd on the host to manage the lifecycle of the backup containers.
4. **Security**: Apply proper SELinux context (:Z) to volumes to allow container access on RHEL 10.

Health-Check and Fix-Bacula Logic

The health-check script (check-bacula.sh) will validate the stack:

- **Podman Status**: Verifies that all 5 core containers are in the running state.
- **Process Verification**: Checks internal daemons (DIR, SD) using bconsole status.
- **NFS Connectivity**: Confirms the NFS volume is writable within the SD container.

The fix-bacula.sh script will:

1. Restart specific containers if they are unhealthy.
2. Perform podman system migrate --new-runtime=crun if runtime issues are detected after a system upgrade.
3. Kill any "zombie" processes that are holding locks on the catalog or volumes.

End-User CLI Helper Scripts

A wrapper script (lab-backup) will simplify operations for engineers:

- lab-backup run <vm>: Triggers a job via bconsole.
- lab-backup status: Summarizes recent jobs and storage health.
- lab-backup restore <vm>: Automates the multi-step virsh define and cp process.

Operational Requirements: Alerting, Logging, and Documentation

Logging and Verbosity

Bacula's Messages resource will be configured to log to both the catalog and syslog. Bacularis will provide a web-based view of these logs for easy troubleshooting.

Alerting and Notifications

- **Email Alerts**: Using bsmtp to notify administrators of job success, failure, or warnings.
- **Critical Escalation**: The health-check script will use webhooks to alert engineers of systemic failures (e.g., NFS disconnect).

Documentation Standards

Deliverables include an Architecture Specification, SOPs for restores, a Disaster Recovery Runbook, and documentation for all custom automation scripts.

Deployment Roadmap

1. **Phase 1: Staging**: Prepare NFS storage and benchmark I/O on RHEL 10 hypervisors.
2. **Phase 2: Core Deployment**: Use Podman to deploy the containerized Director, API, and Web UI.
3. **Phase 3: Pilot**: Move non-critical VMs to the system and verify metadata-consistent restores.
4. **Phase 4: Production**: Enroll VPN gateways and critical lab infrastructure into the schedule.

Risk Assessment and Mitigation

Risk	Impact	Mitigation Strategy

Snapshot Failure	VM I/O stalls.	Quiesce via QEMU Guest Agent.
NFS Congestion	Network latency for lab engineers.	Use dedicated backup VLAN or traffic shaping.
Container Runtime Issue	Backup services fail.	Ensure crun is configured as the default runtime on RHEL 10.
Improper XML Capture	Incomplete VM re-instantiation.	Use virsh dumpxml to verify all metadata is captured.