Technical Solution Brief

Windows Server 2022 SQL Image Implementation Guide

September 3, 2025

Next Orbit Windows Server 2022 – Image Implementation Guide

1. Overview

This implementation guide describes how to build and manage Windows Server 2022 golden images that support Microsoft SQL Server Enterprise, Standard, and Developer editions. The design ensures hardened, role-specific images that accelerate provisioning, maintain compliance, and support seamless certificate rotation and post-configuration activities.

2. Image Philosophy and Security Baseline

Golden images follow these guiding principles:
 • Hardened OS baseline (LGPO + SCHANNEL) using hardening.ps1.
 • WinRM over HTTPS with certificate validation.
 • Universal prerequisites (.NET 4.8.1, Visual C++ Redistributables, RSAT tools as needed).
 • Agent and client installers (CrowdStrike, Tanium, Splunk, etc.) pre-staged in the image but not enrolled.
 • SQL Server installation media and cumulative update (CU) packages staged and hash-verified.
 • Azure Key Vault (AKV) certificate extension pre-installed for runtime certificate retrieval and rotation.
 • No secrets, product keys, or environment variables stored in the image; runtime bindings handled by post-configuration automation.

3. Packer JSON Templates

Three Packer templates are used: one for SQL Enterprise, one for SQL Standard, and one for SQL Developer. They differ in image naming, tags, and the SQL edition media that is staged. All templates enforce WinRM TLS, stage agents and libraries, and integrate AKV certificate extension.

3.1 Enterprise JSON

```json
 {Enterprise JSON will be provided separately}
```

3.2 Standard JSON

```json
 {Standard JSON will be provided separately}
```

3.3 Developer JSON

```json
 {Developer JSON will be provided separately}
```

4. Azure DevOps Pipeline Design

The pipeline is defined in YAML and parameterized by SQL edition. It follows these stages:

• Validate: Lint Packer templates, check JSON/HCL syntax, enforce policy gates.
• Build: Run Packer builds with environment parameters (edition, manifest).
• Validate Image: Boot a temporary VM from the built image, run Pester/PowerShell tests to confirm hardening, agent staging, and SQL media presence.
• Publish: Push image to Shared Image Gallery (SIG) with semantic versioning, replicate to target regions.
• Notify: Report build results, manifest, and image version metadata.

Pipeline security practices:
• Use Managed Identity for artifact downloads.
• Store secrets in Azure Key Vault and reference securely in pipeline.
• Enforce SHA-256 validation of all downloaded components.
• Never echo SAS tokens or secrets in logs.

Sample Pipeline YAML

```yaml
 {Pipeline YAML will be provided separately}
```

5. Post-Deploy Automation (Ansible)

Environment-specific configuration remains in Ansible. This ensures images are portable and environment bindings are applied securely at runtime. Key tasks include:

- Domain join and OU placement.
- Agent enrollment (e.g., Tanium, CrowdStrike) with tenant keys from AKV.
- SQL instance configuration (license activation for Enterprise/Standard, SA password from AKV, AG orchestration).
- DFS and DNS orchestration.
- Certificate binding using AKV extension.

Tanium post-configuration scripts handle client registration and versioning, ensuring consistency across environments.

6. Operational Best Practices

- Use manifest.json for all artifact downloads, with SHA-256 validation.
- Separate universal base image from SQL role images.
- Maintain semantic versioning in SIG (YYYY.MM.DD.Build).
- Refresh images monthly or when critical CUs are released.
- Retain previous images for rollback.
- Enforce inventory hygiene in Ansible (only active hosts tracked).
- Validate every image via automated post-build tests before publishing.