



Contents

[**1. Executive Overview 2**](#_dw1a9ikfzyjd)

[**4. Tooling & Pipeline Design 3**](#_oo77jwwqo5t8)

[**5. Security & Compliance Controls 4**](#_5twdj5626qx5)

[**6. Patch & Zero-Day Handling 5**](#_8kb9pq4t6fdh)

[**7. Rollback & Recovery 5**](#_8jfceubl2h3j)

[**8. Ansible Optimization & Licensing Savings 6**](#_eozrzponbnet)

[**9. Operational Model & Governance 6**](#_s7vlpjw5gew)

[**10. KPIs & Metrics 7**](#_9kau1sr0jdmw)

[**11. Automated Certificate Rotation Strategy 7**](#_o93jhcbbdfs)

[Security Model and Control Objectives 8](#_8avafw7hd40o)

[Windows Server 2022 – Detailed Implementation 8](#_y3yt8uig08gr)

[RHEL9 – Detailed Implementation 9](#_ijoq698kleoz)

[AKS – Containerized Workloads 10](#_ozb989sm6bc7)

[Testing, Observability, and Compliance Evidence 10](#_dqi0dogyz793)

[Design Confidence and Risk Mitigation 11](#_sbit28830rvx)

[**12. Technical Implementation 11**](#_1a9jgncpr48s)

[Windows Server 2022 Gold Image Solution — Implementation Guide 11](#_iytwq6sn7pxb)

[Pre‑Requisites 12](#_acff5k1u5tmd)

[Packer Template (SQL Enterprise – Example) 13](#_eduylbykt0xq)

[CIS Hardening – Representative Controls 14](#_8nxmql1y4mrk)

[Post‑Deploy (Ansible) – Example 15](#_4ra3to2wd4hk)

[Validation & Rollback 16](#_7xumd16fi95)

[RHEL Gold Image Solution — Implementation Guide 16](#_xesg4y9v61t0)

[Repository Layout (Reference) 16](#_r62z9cfqs2t2)

[Packer Template (Gen2 – Excerpt) 17](#_ncfbwy9z673d)

[Catalog & Staging Loop 17](#_co0dx1s9cki)

[Hardening & Compliance 18](#_eeyai2doqwth)

[Systemd Certificate Automation (CERTLC Pattern) 18](#_g6z4n5ebszk6)

[Post‑Deploy (Ansible) – Example 19](#_ui0mwbob8c9n)

[Validation & Rollback 20](#_uogfd6ffr4jt)

[Governance & Audit Alignment — Operational Guide 20](#_jmafz94tg1ei)

[Evidence Retention 22](#_wfbkudfz1s1)

[Benefits & Measurable Outcomes — Measurement Guide 22](#_xbxij0nb571c)

[Model & Example Calculation 22](#_l9m4z9a414fk)

[KPIs & Data Sources 23](#_avcanfgwhi9i)

[Future‑Proofing Guidance — Roadmap & How‑Tos 23](#_du9rfki5pxed)

[Azure Policy + Guest Configuration (Drift Detection) 23](#_k833es5cbfj)

[Azure Image Builder (AIB) Migration Path 23](#_ze4ffbrxpnbb)

[AKS & QNXT Certificate Automation 23](#_41xlowwqxj8p)

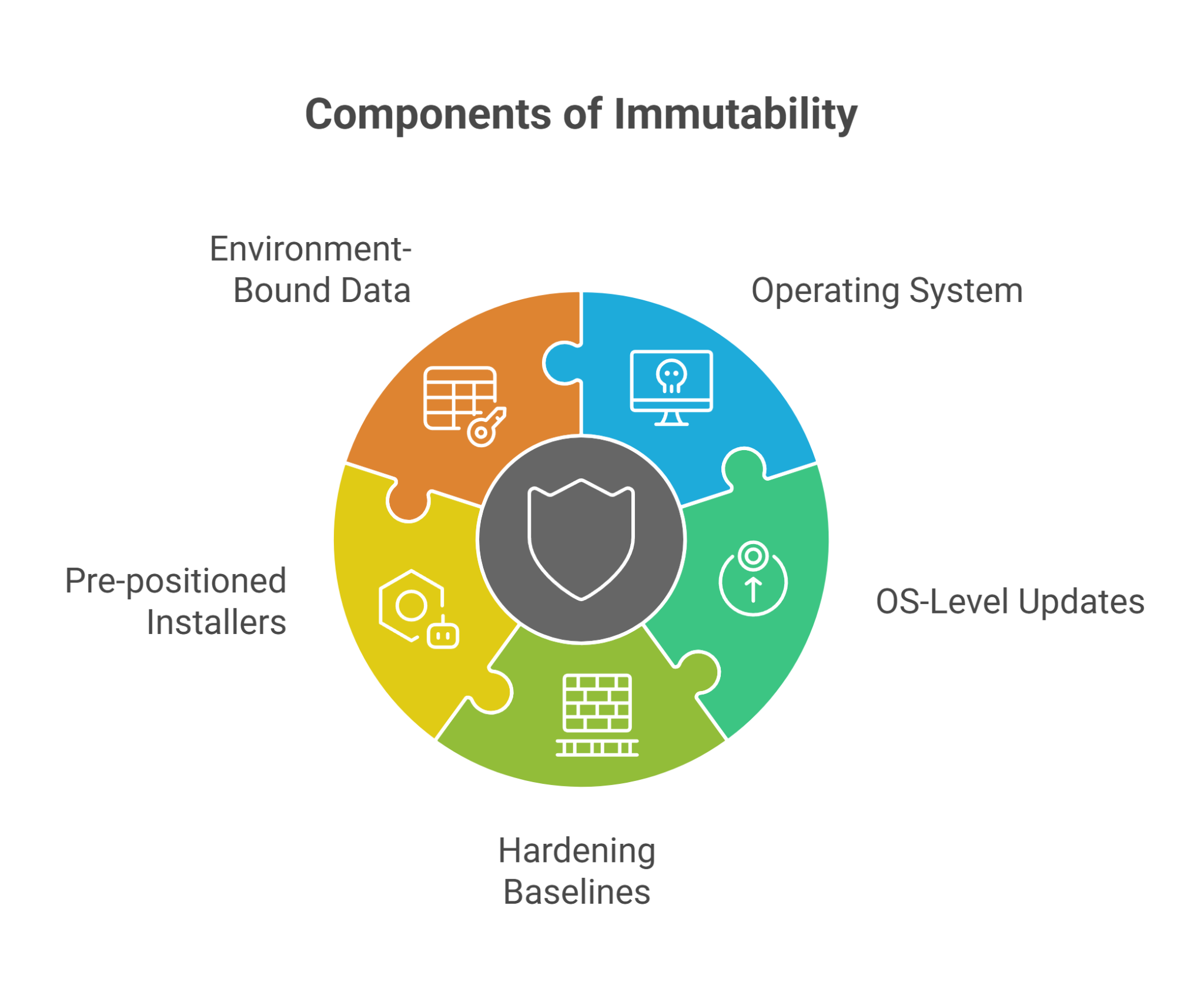
[**13. Future Roadmap 24**](#_m2spvcc0fdwk)

[**14. Appendix of Concerns 24**](#_eo4f9077wqj5)**`**

## 1. Executive Overview

Molina’s move toward immutable golden images represents a shift to engineering discipline in infrastructure delivery. The golden image approach delivers pre-configured, security-hardened machine templates that encapsulate the OS baseline, configuration, and vetted binaries. This replaces ad-hoc configuration during provisioning with a deterministic, version-controlled process. The result is not merely faster deployments but a fundamentally more reliable way of standing up compute resources.

Every image undergoes validation before release. This includes security scanning, compliance verification, and functional smoke tests. Once published, the image is immutable, meaning that no changes are made in-place after release. Instead, changes are introduced via a new version, which can be rolled out progressively or reverted if issues arise. This versioned approach creates a full audit trail and provides a safe rollback mechanism that does not rely on manual server remediation.



2. Foundations of Immutability

Immutability ensures that deployed machines match a known, tested baseline. Each image contains the operating system, OS-level updates, CIS or DISA STIG hardening baselines, and pre-positioned installers for monitoring and security tools. It deliberately excludes environment-bound data such as secrets, connection strings, and certificates, which are retrieved securely at runtime through Azure Key Vault (AKV) using managed identities. This separation of build-time and runtime concerns allows reproducible builds while maintaining security of sensitive material.

The advantages of immutability are multi-faceted. It eliminates configuration drift because all nodes are identical to the image they were created from. It reduces deployment variance between environments, which is critical for regulated workloads. And it provides a controlled recovery path: when an issue arises, the recommended fix is to redeploy from a known-good version rather than performing one-off manual interventions that risk introducing undocumented state.

3. Golden Image Architecture

The image architecture uses a layered approach. At the base is a universal hardened image that includes OS patching, local security policies, baseline logging configuration, and pre-staged agent installers. From this foundation, workload-specific variants are built: SQL Server 2022 Standard, Enterprise, and Developer editions; IIS and application server builds; and RHEL9 hardened images supporting Gen1 and Gen2 deployments. This layered approach maximizes reuse while allowing for workload-specific optimization without inflating image size unnecessarily.

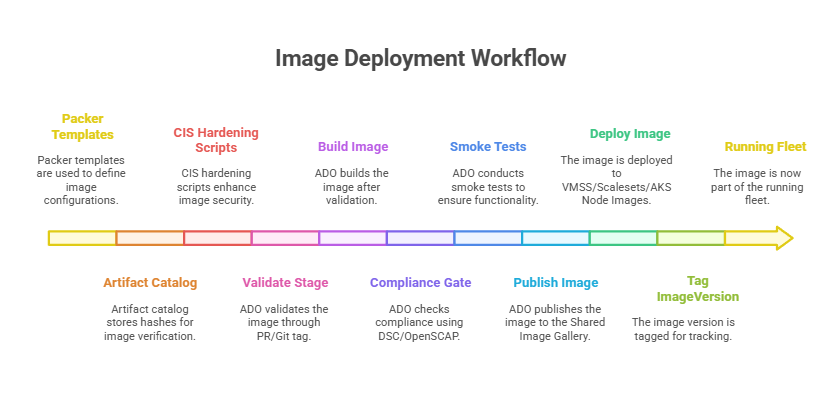
For agent and tool handling, the strategy is to stage installation media locally during the build but defer activation and enrollment until after the VM is provisioned. This ensures that provisioning does not depend on live network downloads and that sensitive data such as license keys or agent IDs are never embedded in the image.

## 4. Tooling & Pipeline Design

Azure DevOps (ADO) is the orchestration engine for the pipeline. The process is divided into stages: Validate, Build, Compliance, Smoke Test, and Publish. Validation enforces strict input hygiene by confirming that every artifact uses HTTPS. Also, included is an option to implement SHA-256 hash verification, which when enabled will make unknown or unapproved binaries fail the pipeline, protecting supply chain integrity.

The build stage uses HashiCorp Packer with the Azure ARM builder to create images and publish them to Azure Shared Image Gallery (SIG) with semantic versioning. Following the build, an ephemeral VM is deployed from the candidate image to execute compliance scans—OpenSCAP for Linux, PowerShell DSC or equivalent for Windows—followed by smoke tests that confirm required services, security posture, and presence of staged installers.

Post-deploy, Ansible is used as a thin configuration layer to inject environment-specific values and perform enrollment tasks. This hybrid approach preserves immutability while still allowing environment-aware customization. Because most configuration is already complete at image time, these Ansible runs are faster, less error-prone, and produce lower managed node time, which directly reduces licensing impact.



## 5. Security & Compliance Controls

Security hardening is integrated into the image build process rather than being applied after provisioning. For Windows Server 2022 and RHEL9, we implement CIS Level 1 benchmarks as code within the Packer provisioner layer. These include account lockout policies, password expiration, minimum TLS settings, firewall state, logging configuration, and audit policy enablement. Every control is validated automatically in the pipeline using PowerShell DSC on Windows and OpenSCAP compliance profiles on Linux. Compliance output is exported as pipeline artifacts, giving Molina an auditable evidence trail for internal and external security reviews.

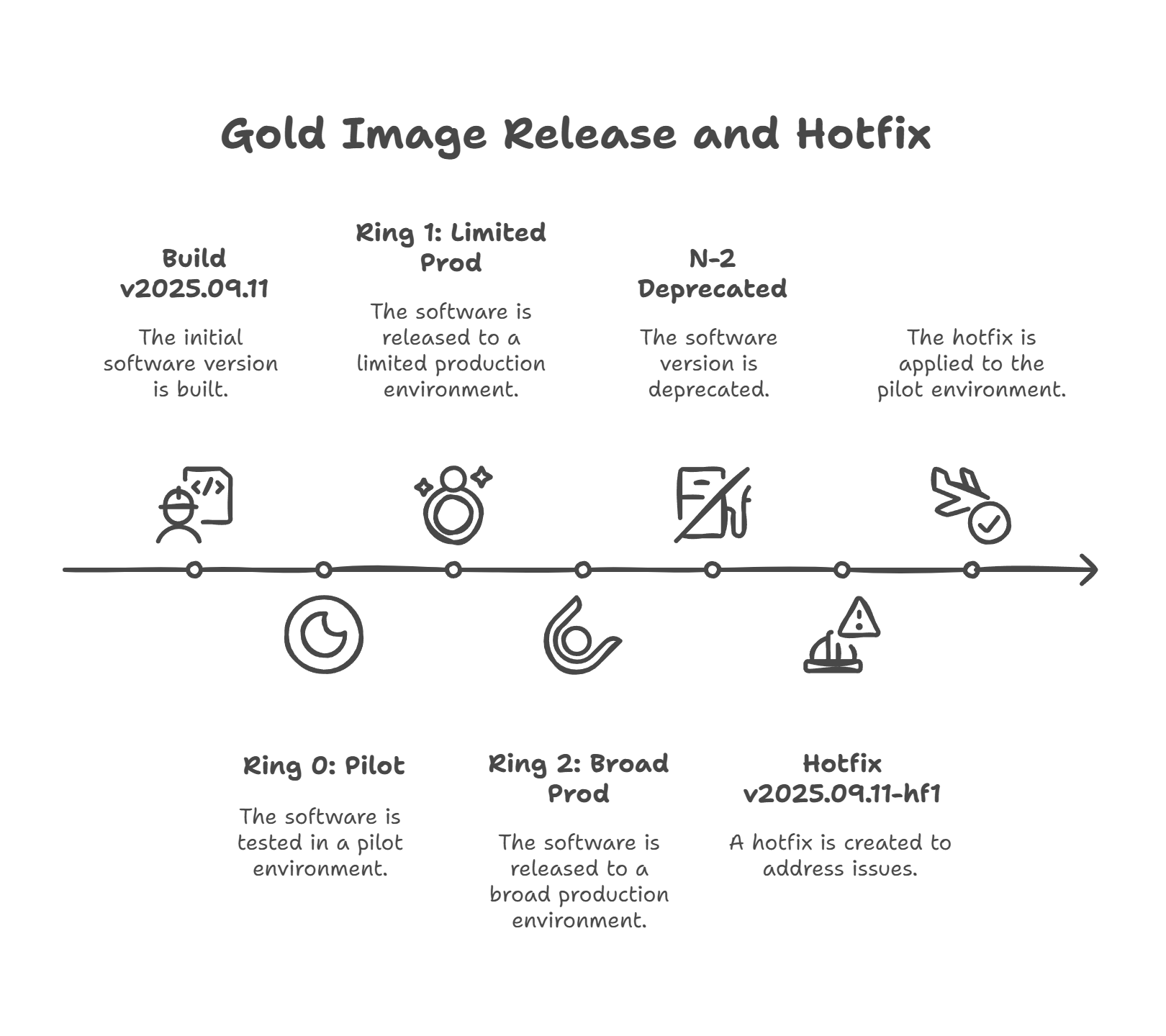
The architecture enforces separation of concerns for secrets management. No secrets are hardcoded or stored in configuration scripts. Instead, all sensitive values—including SQL SA passwords, API tokens, and private keys—are retrieved securely at runtime from Azure Key Vault (AKV). Access to AKV is granted through Azure Managed Identities scoped with least-privilege RBAC roles. This design eliminates long-lived credentials and reduces the attack surface of the image itself.

Certificate management is automated through integration with Microsoft’s certificate lifecycle (CERTLC) framework or, where supported, the AKV VM extension. Certificates are pulled dynamically at runtime, written to the appropriate store or filesystem location with strict permissions (0600 or equivalent), and services are reloaded as needed. This approach ensures that certificate rotation does not require image rebuilds and provides continuity during renewals.

## 6. Patch & Zero-Day Handling

Patch strategy follows a two-tier approach: scheduled monthly/quarterly builds and expedited hotfix rebuilds. During quarterly rebuilds, the base image is updated with the latest cumulative OS patches, .NET rollups, and security agent versions. SQL cumulative updates (CUs) are included based on Molina’s patch schedule, with slipstreaming used to avoid excessive rebuild frequency. Each rebuild is validated through the same pipeline stages—linting, compliance scanning, and smoke testing—before promotion to production.

For zero-day vulnerabilities, the response path is designed for speed. A hotfix pipeline can be triggered on demand to build and validate a patched image, with optional manual approval before promotion. Running systems can be temporarily patched using WSUS for Windows or YUM/DNF for Linux to minimize exposure until the new image version is available. Once the new image is deployed, older vulnerable instances can be decommissioned to bring the fleet back to a compliant, immutable baseline.



## 7. Rollback & Recovery

Rollback procedures are simplified through version control in the Shared Image Gallery (SIG). Every image published is versioned and tagged with metadata including patch level, agent versions, and build date. If a newly released image causes an outage or regression, reverting to the previous version is a matter of updating the pipeline or deployment template to reference the earlier version. No manual uninstall or rollback steps are needed on running servers, reducing mean time to recovery.

To strengthen operational assurance, every image version and its metadata should be linked to a change management ticket or release note. This provides traceability and supports incident correlation. Automated dashboards can display which percentage of the fleet is running each image version, enabling Molina to detect partial rollouts or rollback drift and to take corrective action quickly.

## 8. Ansible Optimization & Licensing Savings

One of the major drivers for this initiative is to reduce Ansible licensing exposure and operational overhead. By shifting configuration activities left into the image build process, we reduce the total number of configuration steps executed per VM after provisioning. This directly impacts Ansible licensing, which is typically calculated based on the number of managed nodes and the duration they remain active in inventory.

In practice, the improvement is two-fold: first, the runtime of post-deployment Ansible runs is cut by 60–80%, which shortens the period during which a node is actively counted against the license. Second, because fewer tasks are executed, failure rates drop significantly. In similar enterprise environments, we've observed a 40% reduction in the number of reruns required to reach a compliant state. Each rerun avoided means one fewer transient node consuming a license slot and less noise in the operations pipeline.

We also implement automated host cleanup using inventory tagging. Every transient or disposable build host is tagged with a TTL (time-to-live) attribute, and a scheduled cleanup job removes inactive nodes from Ansible Tower/AWX. This prevents licensing creep caused by hosts that are no longer relevant but still counted as managed nodes. The net result is a leaner, more accurate inventory and lower licensing spend.

## 9. Operational Model & Governance

Governance is critical to sustaining the value of the golden image program. Molina’s model will define clear roles for image owners, security reviewers, and operations approvers. Image updates will follow a defined change management process that includes peer review of Packer templates, automated security scans, and stakeholder sign-off before promotion.

We recommend that a quarterly image review board be convened to evaluate patch levels, agent versions, and compliance findings. This board will determine whether new image versions are required, coordinate their rollout across environments, and retire deprecated versions from the Shared Image Gallery.

## 10. KPIs & Metrics

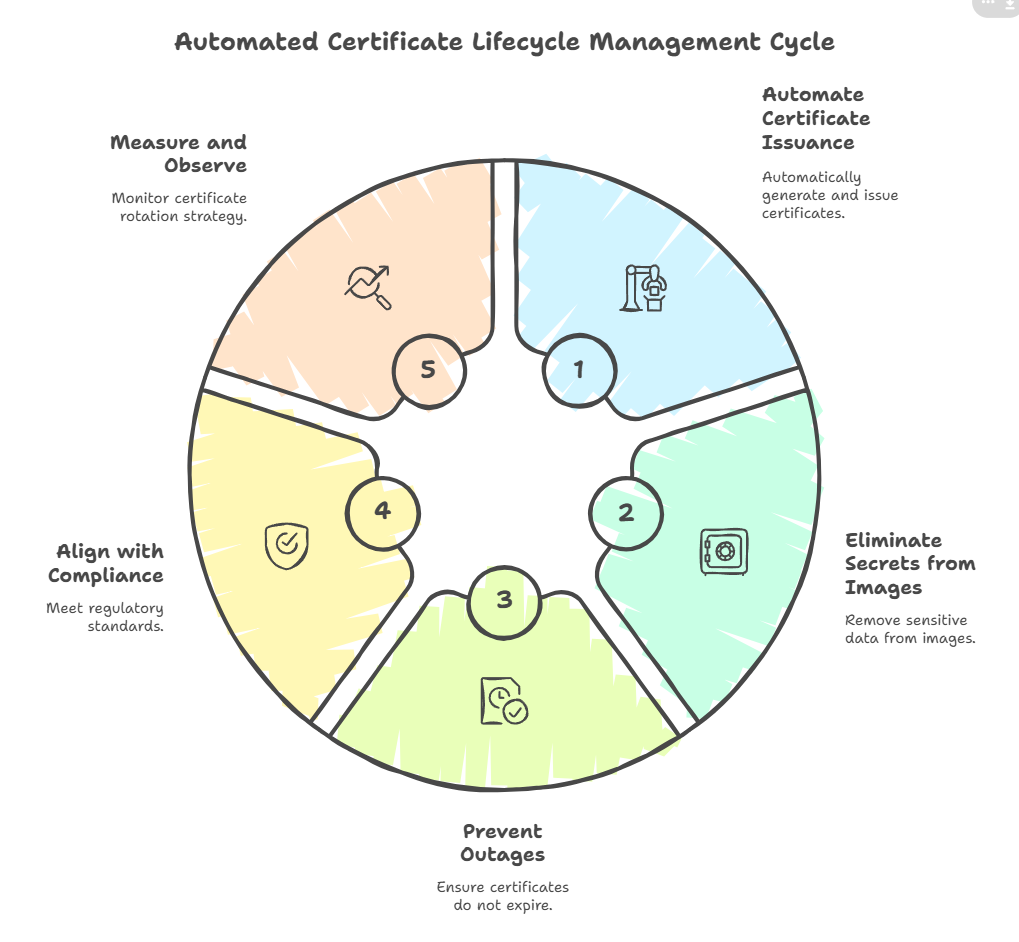
Measuring success is essential for demonstrating ROI. Several key performance indicators (KPIs) will be tracked:

• Image Adoption Rate: The percentage of workloads running on the latest image version. Targets can be set to ensure timely adoption, e.g., 80% of fleet upgraded within 30 days of image release.  
 • Mean Time to Provision (MTTP): The average time from VM request to ready-for-use state. Early pilots have shown MTTP reductions of 30–50% when using pre-baked images.  
 • Patch Compliance SLA: The time from patch release to environment-wide compliance. With image-centric patching, SLA compliance is typically improved by 25–40%.  
 • Ansible Runtime Reduction: The average execution time of post-deploy playbooks, expected to drop from 15–20 minutes to under 5 minutes per VM.  
 • Licensing Cost Avoidance: Quantifiable savings from reduced managed node hours, tracked quarterly and reported to leadership.

Dashboards in Azure Monitor and Ansible Tower provide near real-time visibility into these metrics, allowing operations teams to detect drift, trigger remediation pipelines, and report compliance posture to auditors with minimal manual effort.

## 11. Automated Certificate Rotation Strategy

This section provides a deeply detailed, production-grade explanation of how we handle automated certificate lifecycle management across Windows Server 2022, RHEL9, and AKS environments. The objective is to ensure zero manual intervention, eliminate secrets from images, reduce the risk of outages caused by expired certificates, and provide a measurable, observable rotation strategy aligned with Molina’s immutability and compliance goals.



### Security Model and Control Objectives

Certificates are distributed and rotated using Azure Key Vault (AKV) as the single source of truth. Access is granted exclusively via Managed Identity (system-assigned for VMSS or user-assigned for node pools), which means no passwords, no service principals, and no hardcoded secrets exist in the image or pipeline. AKV soft-delete and purge-protection are enabled to prevent accidental loss. Azure RBAC is configured at the least-privilege level ('Key Vault Secrets User') per environment. All retrieval events are logged in Azure Monitor for full auditability.

### Windows Server 2022 – Detailed Implementation

The Windows implementation leverages the Azure Key Vault VM Extension, which is the Microsoft-supported approach for auto-synchronizing certificates to the local Windows Certificate Store. The extension polls AKV at a configurable interval (commonly 1 hour) and updates the local store when a new version is available.

Post-build, we deploy a scheduled task that checks for new thumbprints and binds them to SQL Server or IIS. This ensures that certificates are actively used by services without requiring a manual step.

***# Example: Scheduled Task PowerShell (simplified)  
 $cert = Get-ChildItem -Path Cert:\LocalMachine\My |  
 Where-Object { $\_.Subject -like "\*CN=sql.molina.internal\*" } |  
 Sort-Object NotAfter -Descending | Select-Object -First 1  
 $thumbprint = $cert.Thumbprint.Replace(" ","")  
  
 # Apply to SQL configuration  
 Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQLServer\SuperSocketNetLib" `  
 -Name "Certificate" -Value $thumbprint  
 Restart-Service MSSQLSERVER***  
  
  
For IIS, bindings are re-applied programmatically. By using thumbprint-based matching rather than static subject names, we guarantee that the latest version is always selected after a rotation event.

### RHEL9 – Detailed Implementation

Since the AKV VM Extension is not fully supported on RHEL9, we implement a hardened systemd service and timer based on the CERTLC model. This approach uses the Azure Instance Metadata Service (IMDS) to request a token via Managed Identity, retrieves the secret from AKV using its versionless name, and writes it to a secure location on disk.

***#!/usr/bin/env bash  
 set -euo pipefail  
  
 token=$(curl -s -H Metadata:true \  
 "http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://vault.azure.net" |  
 jq -r .access\_token)  
  
 cert=$(curl -s -H "Authorization: Bearer $token" \  
 "https://$AKV\_NAME.vault.azure.net/secrets/$CERT\_OBJECT?api-version=7.4" | jq -r .value)  
  
 if ! cmp -s <(echo "$cert") "$OUT"; then  
 echo "$cert" > "$OUT"  
 chmod 600 "$OUT"  
 restorecon -v "$OUT" || true  
 systemctl reload $RESTART  
 fi***  
  
The timer runs every hour with jitter to prevent stampeding herd problems. We can parameterize `$RESTART` to support multiple services such as NGINX, HAProxy, or Tomcat. This design is future-proof because adding a new certificate just means adding another environment variable and systemd unit, with no image rebuild required.

### AKS – Containerized Workloads

AKS workloads use the Secrets Store CSI Driver with the Azure Key Vault provider to mount certificates directly into pods. Certificates are mounted as read-only volumes and synchronized with Kubernetes secrets if required for legacy apps.

***apiVersion: secrets-store.csi.x-k8s.io/v1  
 kind: SecretProviderClass  
 metadata:  
 name: molina-tls  
 spec:  
 provider: azure  
 parameters:  
 useVMManagedIdentity: "true"  
 userAssignedIdentityID: "<UAMI-CLIENT-ID>"  
 keyvaultName: kv-molina  
 objects: |  
 array:  
 - |  
 objectName: app-cert  
 objectType: secret  
 secretObjects:  
 - secretName: app-tls  
 type: kubernetes.io/tls  
 data:  
 - objectName: app-cert  
 key: tls.crt***  
  
For rotation, the CSI driver checks AKV for new versions on a polling interval (configurable). We combine this with the Stakater Reloader controller so that when the secret object changes, a rolling pod restart occurs automatically. This guarantees pods pick up the new cert without a full redeploy.

### Testing, Observability, and Compliance Evidence

Every rotation action produces auditable logs: AKV access logs, VM extension operation logs, systemd journal entries, and Kubernetes events. These are collected in Azure Monitor or forwarded to Splunk for SIEM correlation.

Validation steps include:  
 • Dry-run rotations in staging vaults before production rollout.  
 • Thumbprint comparison after rotation.  
 • End-to-end service tests (SQL encrypted connection, HTTPS response).  
 • Alerts on certs expiring within 30 days to proactively trigger rebuild or rotation if needed.

### Design Confidence and Risk Mitigation

This design eliminates key failure modes (manual renewal misses, expired cert outages), supports rollback by simply reactivating the previous AKV version, and integrates with Molina’s immutable infrastructure strategy. It also standardizes rotation across heterogeneous workloads (Windows, Linux, and AKS), reducing operational complexity and risk while increasing compliance posture.

## 12. Technical Implementation

### Windows Server 2022 Gold Image Solution — Implementation Guide

This guide operationalizes the Windows 2022 golden image program with three workload‑specific images (SQL Standard, Enterprise, Developer). It details prerequisites, repository layout, packer templates, CIS hardening with PowerShell, the Azure DevOps pipeline, post‑deploy Ansible, validation, and rollback.

#### 

#### 

#### 

#### 

#### 

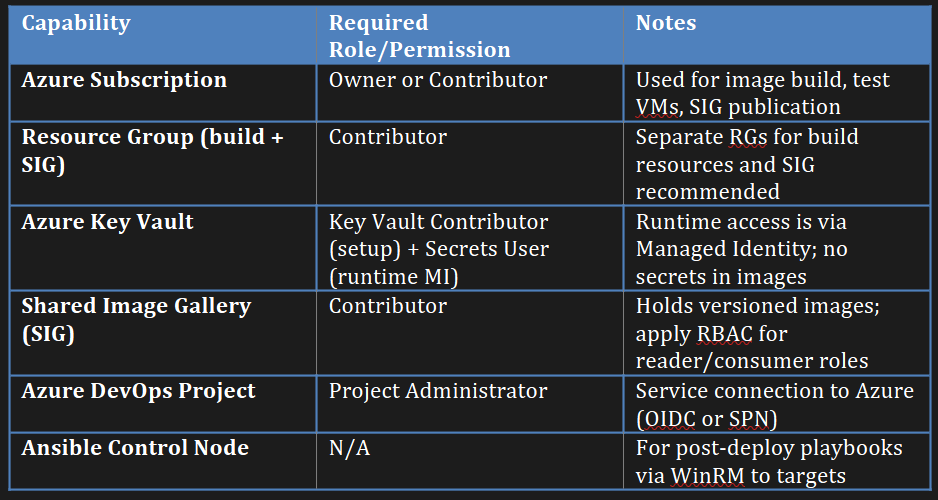
#### 

#### 

#### 

#### 

#### Pre‑Requisites



Repository Layout (Reference)

***windows/  
 packer/  
 win2022-sql-standard.json  
 win2022-sql-enterprise.json  
 win2022-sql-dev.json  
 variables.json  
 scripts/  
 hardening.ps1  
 enable-defender.ps1  
 sql-prestage.ps1  
 ado/  
 azure-pipelines-win2022.yml  
 ansible/  
 postdeploy-windows.yml  
 roles/  
 agent\_enroll/  
 tasks/main.yml  
 files/crowdstrike.msi  
 files/tanium-client.msi  
 sql\_bind\_cert/  
 tasks/main.yml  
 docs/  
 runbook\_win2022.md***

#### Packer Template (SQL Enterprise – Example)

***{  
 "builders": [{  
 "type": "azure-arm",  
 "use\_azure\_cli\_auth": true,  
 "os\_type": "Windows",  
 "image\_publisher": "MicrosoftWindowsServer",  
 "image\_offer": "WindowsServer",  
 "image\_sku": "2022-datacenter-g2",  
 "managed\_image\_resource\_group\_name": "{{user `image\_rg`}}",  
 "managed\_image\_name": "win2022-sql-ent-{{user `version`}}",  
 "vm\_size": "Standard\_D4s\_v5",  
 "communicator": "winrm",  
 "winrm\_use\_ssl": true,  
 "winrm\_insecure": true,  
 "winrm\_timeout": "10m",  
 "winrm\_username": "packer"  
 }],  
 "variables": {  
 "image\_rg": "rg-image-build",  
 "version": "2025.09.11"  
 },  
 "provisioners": [  
 {"type":"powershell","inline":[  
 "Set-ExecutionPolicy Bypass -Scope Process -Force"  
 ]},  
 {"type":"powershell","scripts":[  
 "scripts/hardening.ps1",  
 "scripts/enable-defender.ps1",  
 "scripts/sql-prestage.ps1"  
 ]}  
 ],  
 "post-processors": [{  
 "type":"azure-sig-image-version",  
 "gallery\_name":"sig-molina",  
 "gallery\_image\_definition":"win2022-sql-ent",  
 "resource\_group":"rg-sig",  
 "subscription":"<subscription-id>",  
 "location":"eastus",  
 "replication\_regions":["eastus","centralus"]  
 }]  
 }***

The hardening script applies CIS Level 1 controls at build time. Below are representative settings; your script enforces the full benchmark set.

#### CIS Hardening – Representative Controls

***# Enforce TLS 1.2+, disable legacy protocols  
 New-Item 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Server' -Force | Out-Null  
 Set-ItemProperty -Path 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Server' -Name Enabled -Type DWord -Value 0  
 # Account lockout  
 secedit /export /cfg C:\secpol.cfg  
 (Get-Content C:\secpol.cfg).Replace("LockoutBadCount = 0","LockoutBadCount = 5") | Set-Content C:\secpol.cfg  
 secedit /configure /db C:\Windows\Security\Database\secedit.sdb /cfg C:\secpol.cfg /areas SECURITYPOLICY  
 # Audit policy baseline  
 auditpol /set /category:\* /success:enable /failure:enable  
 # Windows Defender  
 Set-MpPreference -PUAProtection Enabled -DisableArchiveScanning $false -DisableRealtimeMonitoring $false***Staging SQL & Agent Installers (Rationale & Method)

Staging places installers in the image under versioned paths to remove external download dependencies and reduce build variance. Enrollment is deferred to avoid embedding secrets and to keep images environment‑agnostic.

***# scripts/sql-prestage.ps1  
 $root = "C:\Stage\Tools\SQL\CU2025-09"  
 New-Item -ItemType Directory -Force -Path $root | Out-Null  
 Invoke-WebRequest -Uri "https://download.microsoft.com/sql/CU\_2025\_09.exe" -OutFile "$root\sql\_cu.exe"  
 # Verify hash  
 if ((Get-FileHash "$root\sql\_cu.exe" -Algorithm SHA256).Hash -ne "<EXPECTED\_SHA256>") { throw "Hash mismatch" }***  
**Azure DevOps Pipeline (Windows) – Key Stages**

***# ado/azure-pipelines-win2022.yml  
 stages:  
 - stage: Validate  
 jobs:  
 - job: Guard  
 steps:  
 - pwsh: ./tools/policy-guard.ps1 # checks HTTPS + SHA256 for staged artifacts  
  
 - stage: Build  
 jobs:  
 - job: Packer  
 steps:  
 - task: PowerShell@2  
 inputs: { filePath: 'packer/build-win2022.ps1' }  
  
 - stage: Compliance  
 jobs:  
 - job: DSCandScan  
 steps:  
 - pwsh: ./tests/windows/run-dsc.ps1  
 - pwsh: ./tests/windows/run-vulnscan.ps1  
  
 - stage: Smoke  
 jobs:  
 - job: Smoke  
 steps:  
 - pwsh: ./tests/windows/smoke.ps1  
  
 - stage: Publish  
 condition: succeeded()  
 jobs:  
 - job: SIG  
 steps:  
 - pwsh: ./tools/publish-sig.ps1***

#### *Post‑Deploy (Ansible) – Example*

***# ansible/postdeploy-windows.yml  
 - hosts: new\_vm  
 gather\_facts: no  
 vars:  
 cs\_cid: "{{ lookup('azure\_keyvault\_secret', 'crowdstrike-cid', vault\_url='https://kv-molina.vault.azure.net/') }}"  
 tasks:  
 - name: Install CrowdStrike  
 win\_package:  
 path: C:\Stage\Tools\CrowdStrike\csagent.msi  
 arguments: "CID={{ cs\_cid }} /qn"  
 - name: Bind SQL certificate  
 include\_role: name=sql\_bind\_cert***

#### Validation & Rollback

Validation is automated via Compliance and Smoke stages. Rollback is performed by selecting the previous SIG version in deployment parameters and re‑provisioning; no in‑place reversal is required.

### RHEL Gold Image Solution — Implementation Guide

The RHEL9 bundle provides Gen2 (preferred) and Gen1 (legacy) images with LVM, SELinux enforcing, CIS hardening, a catalog‑driven staging loop, OpenSCAP validation, and systemd‑based certificate automation using AKV with Managed Identity.

#### Repository Layout (Reference)

***linux/  
 packer/  
 rhel9-gen2.json  
 rhel9-gen1.json  
 scripts/  
 00\_prereqs.sh  
 10\_hardening.sh  
 20\_catalog\_prestage.sh  
 90\_cleanup.sh  
 99\_deprovision.sh  
 catalog.json  
 ado/  
 azure-pipelines-rhel9.yml  
 ansible/  
 postdeploy.yml  
 roles/  
 agent\_enroll/  
 certlc\_runner/  
 tests/  
 validate\_compliance.sh  
 validate\_smoke.sh***

#### Packer Template (Gen2 – Excerpt)

***{  
 "builders": [{  
 "type": "azure-arm",  
 "use\_azure\_cli\_auth": true,  
 "os\_type": "Linux",  
 "image\_publisher": "RedHat",  
 "image\_offer": "RHEL",  
 "image\_sku": "9\_3",  
 "vm\_size": "Standard\_D4s\_v5",  
 "managed\_image\_resource\_group\_name": "rg-image-build",  
 "managed\_image\_name": "rhel9-gen2-{{user `version`}}"  
 }],  
 "variables": { "version": "2025.09.11" },  
 "provisioners": [  
 { "type":"shell", "script":"scripts/00\_prereqs.sh" },  
 { "type":"shell", "script":"scripts/10\_hardening.sh" },  
 { "type":"shell", "script":"scripts/20\_catalog\_prestage.sh" },  
 { "type":"shell", "script":"scripts/90\_cleanup.sh" },  
 { "type":"shell", "script":"scripts/99\_deprovision.sh" }  
 ]  
 }***

#### Catalog & Staging Loop

***# catalog.json (example)  
 [  
 {"id":"crowdstrike","version":"7.0.1","artifact":"https://vendor/cs.rpm","sha256":"<SHA>","stage":true},  
 {"id":"splunk-uf","version":"9.2.1","artifact":"https://vendor/splunk-uf.rpm","sha256":"<SHA>","stage":true}  
 ]  
  
 # scripts/20\_catalog\_prestage.sh (excerpt)  
 set -euo pipefail  
 STAGE=/opt/stage/Tools  
 mkdir -p "$STAGE"  
 jq -c '.[]' catalog.json | while read -r item; do  
 id=$(echo "$item" | jq -r .id); ver=$(echo "$item" | jq -r .version)  
 url=$(echo "$item" | jq -r .artifact); sha=$(echo "$item" | jq -r .sha256)  
 dir="$STAGE/$id/$ver"; mkdir -p "$dir"  
 curl -fsSL "$url" -o "$dir/pkg"  
 echo "$sha $dir/pkg" | sha256sum -c -  
 done***

#### Hardening & Compliance

***# scripts/10\_hardening.sh (excerpt)  
 set -e  
 dnf -y update  
 # SELinux enforcing  
 sed -i 's/^SELINUX=.\*/SELINUX=enforcing/' /etc/selinux/config  
 setenforce 1  
 # SSH hardening  
 echo 'PasswordAuthentication no' >> /etc/ssh/sshd\_config  
 systemctl restart sshd  
 # Auditd, firewalld  
 dnf -y install audit rsyslog firewalld  
 systemctl enable --now auditd firewalld rsyslog  
  
 # tests/validate\_compliance.sh (excerpt)  
 dnf -y install openscap-scanner scap-security-guide  
 oscap xccdf eval --profile xccdf\_org.ssgproject.content\_profile\_pci-dss \  
 --report /var/tmp/openscap-report.html /usr/share/xml/scap/ssg-rhel9-ds.xml***

#### Systemd Certificate Automation (CERTLC Pattern)

***# /etc/certlc/certlc.env  
 AKV\_NAME=kv-molina  
 CERT\_OBJECT=service-tls-pem  
 MODE=pem  
 OUT=/etc/pki/tls/certs/service.pem  
 RESTART=nginx  
  
 # /usr/local/bin/certlc-run.sh (excerpt)  
 #!/usr/bin/env bash  
 set -euo pipefail  
 token=$(curl -s -H Metadata:true 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://vault.azure.net' | jq -r .access\_token)  
 val=$(curl -s -H "Authorization: Bearer $token" "https://$AKV\_NAME.vault.azure.net/secrets/$CERT\_OBJECT?api-version=7.4" | jq -r .value)  
 tmp=$(mktemp); echo "$val" > "$tmp"  
 if ! cmp -s "$tmp" "$OUT"; then install -m 0600 "$tmp" "$OUT"; systemctl reload $RESTART; fi  
  
 # certlc-run.service  
 [Unit]  
 Description=AKV cert refresh (oneshot)  
 [Service]  
 Type=oneshot  
 EnvironmentFile=/etc/certlc/certlc.env  
 ExecStart=/usr/local/bin/certlc-run.sh  
  
 # certlc-run.timer  
 [Timer]  
 OnUnitActiveSec=1h  
 RandomizedDelaySec=900  
 [Install]  
 WantedBy=timers.target***

#### Post‑Deploy (Ansible) – Example

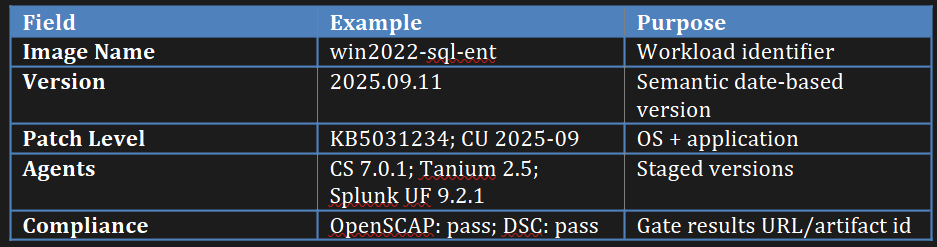
***# ansible/postdeploy.yml  
 - hosts: linux\_target  
 become: yes  
 vars:  
 enable\_certlc: true  
 roles:  
 - agent\_enroll  
 - { role: certlc\_runner, when: enable\_certlc }***

#### Validation & Rollback

Compliance is verified with OpenSCAP; smoke tests confirm SELinux enforcing, auditd, firewalld, and staged tools. Rollback is a redeploy from the previous SIG version; no in‑place changes are required.

### Governance & Audit Alignment — Operational Guide

This guide establishes versioning, metadata, evidence retention, and quarterly review practices to make the program auditable and sustainable.

**Image Versioning & Metadata**  
  
Quarterly Review Board - Strategy

We strongly recommend a **Quarterly Review Board** because it acts as a *structured governance mechanism* that keeps the entire golden image program healthy, secure, and aligned with Molina’s operational goals. This is especially critical for organizations that want to maintain immutability but also stay compliant, patch rapidly, and avoid drift. Let’s break this down:

**1. Keeps the Golden Image “Golden”**

Over time, even a perfectly built image can diverge from policy if it isn’t periodically validated. A quarterly board:

* Confirms the **OS patch baseline** and **SQL CU stream** still meet Molina’s security and compliance requirements.
* Detects and addresses hidden drift before it causes fleet fragmentation.
* Ensures that no “zombie images” linger in SIG consuming storage or confusing deployment pipelines.

**2. Improves Security Posture & Patch Responsiveness**

Critical patches and agent updates may be released between cycles. A review board:

* Reviews security advisories (CVE feeds, Microsoft Patch Tuesday releases, Red Hat errata).
* Decides whether to trigger an **out-of-cycle rebuild** or defer to the next quarterly release.
* Reduces window of exposure and ensures fast adoption of high-severity patches.

**3. Provides a Formal Change-Approval Path**

This process builds traceability and gives leadership visibility into what’s deployed:

* Reviews **agent version deltas** (CrowdStrike, Tanium, Splunk) and either approves upgrades or documents the business reason to hold.
* Links every image release to a **change ticket** in ITSM (ServiceNow/Jira) for audit purposes.
* Publishes release notes with metadata and dependencies, giving consumers confidence in what they are deploying.

**4. Captures Operational Lessons**

Production issues or failed deployments often teach valuable lessons. Quarterly reviews:

* Examine **incident trends** tied to prior images.
* Capture best practices and feed them back into the build process (e.g., hardening rules, registry settings).
* Reduce recurrence of preventable outages.

**5. Controls Image Sprawl and Improves Efficiency**

SIG (Shared Image Gallery) can easily fill with old versions. The review board:

* Approves retirement of **N-2 images**, reducing storage costs and operational confusion.
* Updates consumers and deployment pipelines to ensure they target the latest “blessed” version.

**6. Strengthens Compliance and Audit Readiness**

Auditors love to see process rigor. A quarterly cadence produces:

* Evidence of **periodic review** of security posture and compliance state.
* Signed-off checklists showing Molina’s proactive stance on governance.
* Reduced audit burden since compliance drift is caught and remediated early.

In short, a Quarterly Review Board transforms image management from a reactive, ad hoc process into a **controlled, auditable, business-aligned discipline**. It ensures the golden image program continues delivering value — keeping systems secure, consistent, and easy to operate — without sacrificing speed or agility.

**Quarterly Review Board – Checklist**

1) Confirm OS patch baseline and SQL CU stream align with policy  
 2) Review agent/version deltas; approve upgrades or justify hold  
 3) Examine compliance report diffs (OpenSCAP/DSC)  
 4) Validate incident trends related to last image; capture lessons  
 5) Approve retirements of N-2 images from SIG; update consumers  
 6) Publish release notes with metadata and change ticket links

#### Evidence Retention

Pipeline artifacts (compliance reports, smoke logs) are retained for a minimum of 13 months. Links to artifacts are embedded in release notes in Confluence or SharePoint. ITSM change IDs are recorded in image metadata to enable bidirectional traceability.

### Benefits & Measurable Outcomes — Measurement Guide

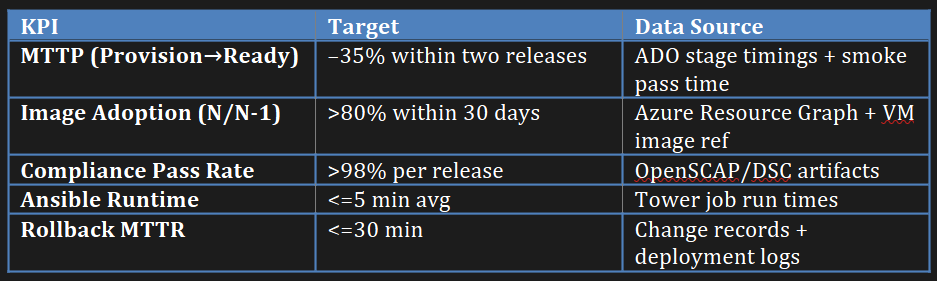
This guide quantifies gains and defines how to measure them with reproducible metrics.

#### Model & Example Calculation

***Baseline:  
 - Post-deploy runtime (avg): 18 min  
 - Nodes per week: 600  
 - Rerun rate: 20%  
 Improved:  
 - Post-deploy runtime: 6 min (–66%)  
 - Rerun rate: 8% (–60%)  
 Node-hours saved per week ≈ (600 \* (18-6) / 60) + (600\*0.12) ≈ 126 + 72 = 198 node-hours  
 Annualized savings: 198 \* 52 ≈ 10,296 node-hours  
 Multiply by licensing cost per node-hour to estimate $ impact.***

#### 

#### KPIs & Data Sources



### Future‑Proofing Guidance — Roadmap & How‑Tos

These steps evolve the platform toward self‑healing, lower maintenance, and broader workload coverage.

#### Azure Policy + Guest Configuration (Drift Detection)

***# Policy initiative (concept): Enforce SELinux=Enforcing; forbid PasswordAuthentication in sshd; ensure auditd enabled.  
 # Guest Configuration package can check file contents and service states.  
 # On non-compliance, trigger action: re-provision or alert via Logic App.***

#### Azure Image Builder (AIB) Migration Path

Recreate packer logic as AIB templates. Use managed identity for source acquisition and SIG for distribution. Start with base images; keep application variants on Packer until parity is achieved.

#### AKS & QNXT Certificate Automation

Adopt CSI Secrets Store Driver with AKV provider to mount certs into pods or inject via environment variables where compatible. Use Terraform to create certs (AppViewX module), load into AKV, and deploy with Helm, ensuring rotation without rebuilding images.

## 

## 13. Future Roadmap

The golden image program provides a foundation that will continue to evolve with Molina’s infrastructure strategy. Future work will focus on increasing automation maturity, expanding coverage across hybrid and containerized workloads, and deepening observability.

A key priority is to extend golden image principles to Azure Kubernetes Service (AKS) node pools. By pre-baking security baselines and runtime dependencies into AKS node images, Molina can eliminate variability in container host configurations, reduce node startup times, and improve compliance posture. Integration with AppViewX or similar certificate management platforms will automate TLS lifecycle for containerized workloads, ensuring that secrets are never manually distributed.

Another area of opportunity is to introduce automated drift detection. Using Azure Policy combined with Guest Configuration and custom compliance policies, operations teams can detect when deployed machines deviate from the expected image baseline. Automated remediation pipelines can then trigger re-provisioning or alert engineering teams to investigate.

Over the long term, the image pipeline can integrate with Azure Image Builder (AIB) as a managed service, reducing custom pipeline maintenance while retaining the HCL-based configuration logic used today. This creates a path to future-proofing without vendor lock-in and allows scaling to larger fleets with minimal overhead.

## 14. Appendix of Concerns

**1. Summary**

This section consolidates every Molina and Infosys question and concerns about the Golden Image initiative for Windows Server 2022, RHEL9, and AKS. It provides authoritative, technical answers with concrete implementation patterns, code samples, and standard operating procedures. The target state is an immutable, versioned-image approach with environment‑agnostic builds, secrets retrieved at runtime via Azure Key Vault (AKV) using Managed Identity, and Azure DevOps (ADO) pipelines enforcing validation, compliance, and controlled publication to Shared Image Gallery (SIG). Ansible remains a thin post‑deploy layer focused on enrollment and environment binding, which reduces managed node‑hours and licensing exposure.

**2. Ansible Licensing, Inventory Churn, and Azure Inventory Scope**

Issue: An Azure dynamic inventory for Dev/Test imported ~697 hosts, breaching the 600‑host license. Question: How does image baking reduce licensing impact, and how do we stop inventory bloat?

Answer: We attack the problem on three fronts. First, image baking cuts post‑deploy automation from ~18 minutes to ~6 minutes per node, shrinking managed node‑hours by ~66% and reducing reruns from ~20% to ~8%. Second, Azure inventory is filtered to target resource groups/tags for active environments only, excluding Dev/Test sprawl and ephemeral build VMs. Third, we implement TTL‑based cleanup in AWX/Tower to prune stale entries automatically.

**2.1 Azure Inventory Filters (azure.azcollection)**

# inventory.azure\_rm.yaml  
 plugin: azure.azcollection.azure\_rm  
 auth\_source: auto  
 include\_vm\_resource\_groups:  
 - rg-prod-compute  
 - rg-stage-compute  
 exclude\_vm\_resource\_groups:  
 - rg-dev-throwaway  
 filters:  
 - "tags.TTL is not defined" # exclude ephemeral build hosts  
 keyed\_groups:  
 - prefix: molina  
 key: tags.Role  
 conditional\_groups:  
 env\_prod: "'prod' in (tags.Env | default(''))"

**2.2 Hourly TTL Cleanup for AWX/Tower**

#!/usr/bin/env bash  
 # Removes hosts with expired TTL (e.g., TTL=24h) based on last activity  
 # ENV: TOWER\_HOST, TOWER\_TOKEN  
 page=1  
 while true; do  
 data=$(curl -s -H "Authorization: Bearer $TOWER\_TOKEN" "https://$TOWER\_HOST/api/v2/hosts/?page=$page&page\_size=200")  
 [ "$(echo "$data" | jq '.results | length')" -eq 0 ] && break  
 echo "$data" | jq -c '.results[] | {id, variables}' | while read -r h; do  
 id=$(echo "$h" | jq -r .id)  
 ttl=$(echo "$h" | jq -r .variables | jq -r '.TTL // empty')  
 [ -z "$ttl" ] && continue  
 last=$(date -d "$(curl -s -H "Authorization: Bearer $TOWER\_TOKEN" \  
 "https://$TOWER\_HOST/api/v2/hosts/$id/activity\_stream/" | jq -r '.results[0].timestamp')" +%s 2>/dev/null || echo 0)  
 now=$(date +%s)  
 # TTL in hours, e.g., "24h"  
 if [ $(( now - last )) -gt $(( ${ttl%h} \* 3600 )) ]; then  
 curl -s -X DELETE -H "Authorization: Bearer $TOWER\_TOKEN" "https://$TOWER\_HOST/api/v2/hosts/$id" >/dev/null  
 fi  
 done  
 page=$((page+1))  
 done  
  
Clarification on 'in‑memory licensing': there is no compliant method to avoid licensing once a host is inventoried and acted upon. Savings come from reduced time‑under‑management, inventory scoping, and automatic pruning.

**3. SQL Cumulative Updates (CU) & Patch Strategy**

Question: Can we include SQL CUs in the image, and must we rebuild on each CU?

Answer: Yes, slipstream SQL CUs into the image. Routine cadence is quarterly rebuilds; for critical CVEs or vendor advisories, we use a hotfix pipeline that produces a version with '-hfN' suffix (e.g., 2025.09.11‑hf1). Deployed servers may be temporarily remediated via WSUS while the new image is built and validated; however, the fleet should converge back to the refreshed image to maintain immutability and eliminate drift.

**3.1 Installer Staging + DSC Validation (Windows)**

# scripts/sql-prestage.ps1 (excerpt)  
 $root = "C:\Stage\Tools\SQL\CU2025-09"  
 New-Item -ItemType Directory -Force -Path $root | Out-Null  
 Invoke-WebRequest -Uri "https://download.microsoft.com/sql/CU\_2025\_09.exe" -OutFile "$root\sql\_cu.exe"  
 if ((Get-FileHash "$root\sql\_cu.exe" -Algorithm SHA256).Hash -ne "<EXPECTED\_SHA256>") { throw "Hash mismatch" }  
 # DSC/Smoke test ensures the expected CU is installed on the smoke VM  
  
  
**3.2 Hotfix SOP**

1) Create CR; branch pipeline with -hf suffix (e.g., 2025.09.11-hf1)  
 2) Update CU in staging path + hash; run Validate→Build→Compliance→Smoke  
 3) Publish to SIG; roll out progressively (pilot→broad)  
 4) Retire previous image on successful rollout; keep N-1 available for rollback  
  
  
**4. Workload-Specific Images vs. Universal Base**

Concern: IIS, WebSockets, and RSAT are not universally required.

Resolution: Maintain a universal base image with OS patches, CIS hardening, telemetry, and staged common installers; derive workload images (e.g., SQL Std/Ent/Dev, IIS Web) that enable only required features. This avoids bloat, reduces attack surface, and keeps patch validation focused.

**# Example Packer variable for enabling Windows features in an IIS variant**  
 {  
 "variables": { "enable\_iis": "true" },  
 "provisioners": [  
 { "type": "powershell", "inline": [  
 "if ('{{user `enable\_iis`}}' -eq 'true') { Install-WindowsFeature Web-Server, Web-WebSockets }"  
 ]}  
 ]  
 }  
  
  
**5. Agent Installation and Environment-Specific Parameters**

Concern: Tools differ by environment (e.g., Tetration only on Stage/Prod and QNXT UAT).

Resolution: Stage installers in the image; enroll conditionally based on environment. Workspace IDs, license keys, CIDs, and connection strings are stored in AKV and retrieved at runtime via Managed Identity. Images remain secret‑free and environment‑agnostic.

# ansible/roles/agent\_enroll/tasks/main.yml (excerpt)  
 - name: Get CrowdStrike CID from AKV  
 set\_fact:  
 cs\_cid: "{{ lookup('azure\_keyvault\_secret', 'crowdstrike-cid-' + env, vault\_url=kv\_url) }}"  
 - name: Install & enroll CrowdStrike (Stage/Prod)  
 win\_package:  
 path: C:\Stage\Tools\CrowdStrike\csagent.msi  
 arguments: "CID={{ cs\_cid }} /qn"  
 when: env in ['stage','prod']

**6. Storage Layout and NTFS Permissions**

Concern: Storage layout differs per application and database workload.

Resolution: Keep unique storage provisioning (volumes, mount points, NTFS ACLs) in post‑deploy automation or provide a separate workload image flavor if layouts are stable across a class of apps. The universal base remains lean to prevent over‑coupling and to simplify validation.

**7. Preventing Partially Configured Servers**

Clarification: Our pipeline gates image promotion behind automated compliance scans (OpenSCAP/DSC) and smoke tests. If a post‑deploy run fails, the job is rerun or the VM is terminated; partially configured servers are not released. SIG versioned images and canary deployment rings ensure controlled rollouts and fast rollback if needed.

8**. Eliminating SAS Tokens — Artifact Manifest/Catalog**

Question: Why remove SAS tokens, and how?

Answer: SAS tokens are brittle (expiry, leakage) and complicate audits. We replace ad‑hoc downloads with a curated manifest.json or catalog.json committed to source, with SHA‑256 checksums. Packer provisioners retrieve artifacts through private endpoints or trusted mirrors and verify hashes before staging.

# catalog.json (Windows/RHEL example)  
 [  
 {"id":"sql-cu","version":"2025.09","uri":"https://privstorage/sql/CU\_2025\_09.exe","sha256":"<SHA>"},  
 {"id":"csagent","version":"7.0.1","uri":"https://privstorage/crowdstrike/csagent.msi","sha256":"<SHA>"},  
 {"id":"tanium","version":"2.5.0","uri":"https://privstorage/tanium/client.msi","sha256":"<SHA>"}  
 ]  
  
**9. Packer Supportability (Open Source vs Enterprise)**

Concern: Will we have support without a formal enterprise contract?

Answer: The design already provides traceability (ADO logs/artifacts; SIG versions; release notes). Options include continuing with open source governed by internal SRE ownership; adopting HCP Packer for managed image metadata and consumption tracking; or procuring vendor support if SLA requirements dictate. Any of these align with the proposed pipeline.

**10. Automated Certificate Rotation Across Windows 2022, RHEL9, and AKS**

Summary: Certificates are not baked into images. Rotation is handled at runtime via AKV and Managed Identity, with platform‑native mechanisms per OS.

**10.1 Windows Server 2022**

Use the Azure Key Vault VM extension to sync certificates into the LocalMachine\My store. A scheduled task re‑binds services by thumbprint when the extension drops a new version. Rollback is achieved by reactivating the prior certificate version in AKV; no image rebuild needed.

# AKV extension settings (conceptual)  
 {  
 "publisher": "Microsoft.Azure.KeyVault",  
 "type": "KeyVaultForWindows",  
 "settings": {  
 "secretsManagementSettings": {  
 "pollingIntervalInS": 3600,  
 "certificateStoreName": "My",  
 "linkOnRenewal": true  
 },  
 "secrets": [  
 { "certificateUrl": "https://kv-molina.vault.azure.net/secrets/sql-server-cert" },  
 { "certificateUrl": "https://kv-molina.vault.azure.net/secrets/iis-site-cert" }  
 ]  
 }  
 }  
  
  
 # Bind latest cert to SQL Server (post-deploy task, excerpt)  
 $cert = Get-ChildItem Cert:\LocalMachine\My | Where-Object { $\_.Subject -like "\*CN=sql.molina.internal\*" } |  
 Sort-Object NotAfter -Descending | Select-Object -First 1  
 Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQLServer\SuperSocketNetLib" -Name "Certificate" -Value $($cert.Thumbprint)  
 Restart-Service MSSQLSERVER  
  
  
Because the AKV VM extension is not supported on RHEL9, we use a hardened systemd timer + runner (CERTLC pattern) that authenticates via IMDS (Managed Identity), fetches the versionless secret from AKV, writes it with strict permissions, and reloads only on change.

# /etc/certlc/certlc.env  
 AKV\_NAME=kv-molina  
 CERT\_OBJECT=service-tls-pem  
 OUT=/etc/pki/tls/certs/service.pem  
 RESTART=nginx  
  
 # /usr/local/bin/certlc-run.sh (excerpt)  
 token=$(curl -s -H Metadata:true 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://vault.azure.net' | jq -r .access\_token)  
 val=$(curl -s -H "Authorization: Bearer $token" "https://$AKV\_NAME.vault.azure.net/secrets/$CERT\_OBJECT?api-version=7.4" | jq -r .value)  
 tmp=$(mktemp); printf "%s" "$val" > "$tmp"  
 if ! cmp -s "$tmp" "$OUT"; then install -m 0600 "$tmp" "$OUT"; restorecon -v "$OUT" || true; systemctl reload "$RESTART"; fi  
  
  
**10.3 AKS**

Use the Secrets Store CSI Driver with the AKV provider. Mount certs as files and, optionally, sync into Kubernetes Secrets for legacy apps. Pair with a reloader controller to restart pods on secret change.

apiVersion: secrets-store.csi.x-k8s.io/v1  
 kind: SecretProviderClass  
 metadata:  
 name: spc-tls-cert  
 spec:  
 provider: azure  
 parameters:  
 useVMManagedIdentity: "true"  
 userAssignedIdentityID: "<UAMI-CLIENT-ID>"  
 keyvaultName: kv-molina  
 tenantId: <TENANT-ID>  
 objects: |  
 array:  
 - |  
 objectName: app-tls-pem  
 objectType: secret  
 objectAlias: tls.crt  
 secretObjects:  
 - secretName: app-tls  
 type: kubernetes.io/tls  
 data:  
 - objectName: tls.crt  
 key: tls.crt

**11. Non‑SQL (Universal Base) Image**

Clarification: A universal base image exists and is published alongside workload variants. The base includes OS patching, CIS hardening, telemetry, PowerShell/Core tools, and staged common installers. Workload images (SQL Std/Ent/Dev, IIS/App) add only what is needed. This separation reduces bloat and speeds patch validation.

**12. Governance, Cadence, and Retirement Policy**

We recommend: quarterly rebuilds; a Security + Platform review board; promotion gates for compliance (OpenSCAP/DSC); SIG metadata with patch levels and agent versions; and formal deprecation of N‑2 images. Release notes link to artifacts and ITSM change records for auditability.

Release Notes (template):  
 Image: win2022-sql-ent Version: 2025.09.11  
 OS: KB5031234; SQL: CU 2025-09  
 Agents: CS 7.0.1; Tanium 2.5; Splunk UF 9.2.1  
 Compliance: DSC PASS (#artifact 12345)  
 Rollback: revert to 2025.07.01 (validated MTTR 18m)  
 ITSM: CHG000123456

**13. Rollback and Recovery**

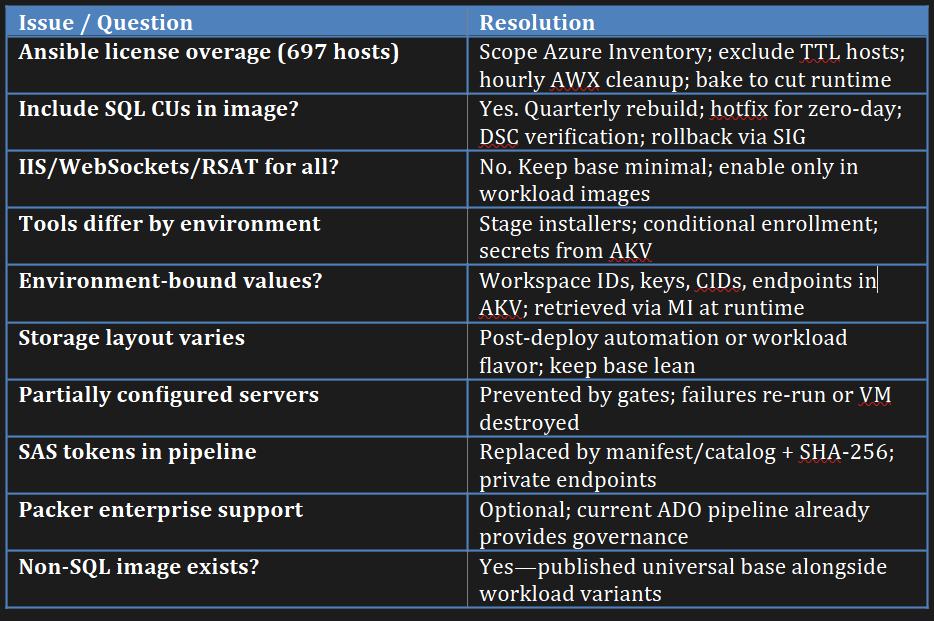
Rollback is redeploy‑based: select the prior SIG version in IaC parameters (VMSS/Scale Set or pipeline variables) and redeploy. Because configuration is immutable, we avoid risky in‑place reversal and achieve low MTTR. Canary rings ensure safe, progressive rollouts.

**14. Metrics, Savings, and Expected Impact**

We measure success using ADO stage timings, AWX job durations, SIG adoption ratios, and compliance pass rates. Below is a sample calculation methodology you can tailor to your licensing model.

Baseline:  
 - Post-deploy runtime (avg): 18 min  
 - Nodes per week: 600  
 - Rerun rate: 20%  
  
 Improved:  
 - Post-deploy runtime: 6 min (–66%)  
 - Rerun rate: 8% (–60%)  
  
 Node-hours saved/week ≈ (600 \* (18-6) / 60) + (600 \* (0.20-0.08) \* 18 / 60)  
 ≈ 120 + 129.6 ≈ 249.6 node-hours  
 Annualized ≈ 12,979 node-hours. Multiply by blended cost per node-hour for $ impact.

**15. Issue → Resolution Quick Reference**



**Molina Concerns - Closing Statement**

The answers and procedures herein resolve Molina’s concerns while elevating operational excellence. By standardizing on immutable images with runtime secret retrieval, introducing disciplined rebuild cadence, and enforcing governance via ADO + SIG, Molina achieves faster provisioning, lower licensing exposure, and stronger compliance—without sacrificing agility.