**Software Version Monitoring and Update Process for Azure VMs**

**Executive Summary**

This document details a two-phase process for managing software versions across 1,200 Azure virtual machines (600 Windows + 600 Linux) using an Azure DevOps (ADO) CI/CD pipeline for scheduling. In **Phase 1**, a Python script (vm\_software\_checker.py) runs weekly to check software versions on VMs, compare them with versions in an Azure Blob Storage container, and email a report to relevant teams.

In **Phase 2**, the script will be enhanced to automatically update software on VMs if versions differ from the container, based on the report. This leverages ADO for automation, aligning with the workflow where the Systems Engineering (SysEng) team updates the container upon receiving new software from the InfoSec team or other stakeholders. The Golden Image Method, combined with this process, offers time efficiency, cost savings, and standardization over traditional methods.

**Assumptions**

* **VM Distribution**: 600 Windows VMs + 600 Linux VMs.
* **Software**:
  + **Linux VMs**:
    - node\_exporter-1.9.0.linux-amd64.tar.gz (Monitoring).
    - SentinelAgent\_linux\_x86\_64\_v24\_2\_2\_20.deb (Security).
  + **Windows VMs**:
    - windows\_exporter-0.26.1-amd64.msi (Monitoring).
    - SentinelInstaller\_windows\_64bit\_v24\_1\_5\_277.msi (Security).
    - wincollect-10.1.3-24.x64\_6\_3\_(1).msi (Log Collection).
* **Infrastructure**:
  + Azure Blob Storage container for software storage.
  + ADO pipeline hosted on a Microsoft-hosted agent (e.g., ubuntu-latest or windows-latest).
  + SMTP server for email notifications.
* **Scheduling**: ADO pipeline runs weekly (e.g., Sunday at 2:00 AM UTC).

**Process Overview**

**Software Update Workflow**

1. **Software Update Received**: SysEng receives updated software (e.g., new SentinelAgent version) from InfoSec or other teams.
2. **Container Update**: SysEng uploads the updated software to the Azure Blob Storage container, ensuring filenames reflect version numbers.
3. **Pipeline Execution**:
   * **Phase 1**: ADO pipeline runs the script weekly to identify VM software versions, compare them with the container, and send a report.
   * **Phase 2**: The script will update outdated software on VMs based on the report.

**Phase 1: Software Version Monitoring**

**Objective**

Monitor software versions on all VMs weekly via an ADO pipeline, compare them with Azure Blob Storage container versions, and email a report to teams.

**Script Details**

* **Name**: vm\_software\_checker.py
* **Functionality**:
  + Connects to Linux VMs via SSH and Windows VMs via WinRM.
  + Checks installed software versions:
    - Linux: node\_exporter, SentinelAgent.
    - Windows: windows\_exporter, SentinelInstaller, wincollect.
  + Retrieves versions from the Azure Blob Storage container.
  + Generates and emails a comparison report.
* **Location**: Stored in an ADO repository (e.g., scripts/vm\_software\_checker.py).

**ADO Pipeline Configuration**

* **File**: azure-pipelines.yml
* **Variables**:
  + Store sensitive data (e.g., SSH\_PASSWORD, WINRM\_PASSWORD) in an ADO variable group (e.g., SoftwareCheckSecrets) with secure variables.

**Implementation**

* **Setup**:
  + Commit vm\_software\_checker.py to your ADO repository under scripts/.
  + Create azure-pipelines.yml in the repo root.
  + Configure a variable group in ADO:
    - SSH\_USER, SSH\_PASSWORD, WINRM\_USER (dm.fareportal.local.com\devauto), WINRM\_PASSWORD, AZURE\_STORAGE\_CONNECTION\_STRING, SMTP\_USER, SMTP\_PASSWORD.
    - Set TO\_EMAILS and CC\_EMAILS in the script or as pipeline variables.
* **Sample Report**:

**Process Steps**

1. **Pipeline Trigger**: ADO runs the pipeline weekly.
2. **Script Execution**: Checks versions on 1,200 VMs.
3. **Report Generation**: Compares VM versions with container versions.
4. **Email Delivery**: Sends report to TO\_EMAILS (e.g., SysEng, InfoSec) with CC\_EMAILS (e.g., managers).
5. **Review**: Teams assess discrepancies and plan manual updates if needed.

**Phase 2: Automated Software Updates**

**Objective**

Enhance the script and pipeline to automatically update software on VMs if installed versions differ from the container, reducing manual intervention.

**Proposed Enhancements**

* **Update Logic**:
  + **Linux VMs**:
    - Download software from Blob Storage using SAS URLs.
    - Install: dpkg -i for .deb, extract/configure for .tar.gz.
    - Example: wget <sas\_url> -O SentinelAgent.deb && dpkg -i SentinelAgent.deb.
  + **Windows VMs**:
    - Download .msi files.
    - Install: msiexec /i <msi\_file> /quiet /norestart.
    - Example: Invoke-WebRequest -Uri <sas\_url> -OutFile software.msi; msiexec /i software.msi /quiet.
* **Pipeline Changes**:
  + Add steps to generate SAS tokens and pass them to the script.
  + Update script to execute updates and verify post-update versions.
* **Report**: Include update actions (e.g., “Updated SentinelAgent from 24.2.2.20 to 24.2.3.0”).

**Implementation Plan**

* **Development**:
  + Enhance check\_linux\_vm and check\_windows\_vm with update logic.
  + Integrate Azure Blob SAS token generation in the pipeline.
* **Testing**: Pilot on 10 VMs to validate updates.
* **Rollout**: Deploy to all 1,200 VMs after testing (e.g., 3-month timeline).

**Sample Enhanced Report**

## Benefits

* **Phase 1**:
  + Automated monitoring via ADO reduces manual effort.
  + Immediate visibility into software version discrepancies.
* **Phase 2**:
  + Automated updates ensure compliance with minimal intervention.
  + Leverages existing ADO infrastructure for scalability.