Developer Guide: Immutable Custom CIO Image Solution (ADO + AIB)

# 1. Solution Overview

The Immutable Custom CIO Image solution provides a standardized way to build, validate, and publish custom VM images for CIO teams using Azure DevOps (ADO) and Azure Image Builder (AIB).  
  
- Pipelines run in ADO and trigger AIB builds.  
- Code and templates (AIB build template, customization scripts, validator scripts) are stored in a GitHub repo.  
- Secrets are stored securely in Azure Key Vault and referenced dynamically in the AIB build template.  
- Application binaries & software installers are pulled from Artifactory.  
- Logs are pushed to a central Storage Account.  
- Final images are published to a Shared Compute Gallery, making them consumable across subscriptions and tenants.  
  
This approach ensures images are consistent, secure, and fully automated with minimal manual intervention.

# 2. Responsibilities Breakdown

## Automated by the Solution

Once CIO sets up prerequisites, the following are fully automated via pipelines:  
- Image build trigger via ADO pipeline.  
- Retrieval of secrets from Key Vault during build.  
- Download of binaries/software from Artifactory.  
- Execution of customization scripts (software installation, configuration).  
- Execution of validator scripts (health checks, installation verification).  
- Publishing the final image into Compute Gallery.  
- Storage of logs into Storage Account.

## CIO Team Setup (Required Prerequisites)

CIO teams need to provision and configure the following before the pipelines can run:  
  
1. ADO Project – Request a new ADO project in OneDevOps portal or reuse an existing one (recommended: new project for RBAC isolation).  
2. GitHub Repository – Create a new dedicated GitHub repo for Custom Image Solution (preferred).  
3. Azure Resources – Storage Account, Key Vault, Compute Gallery.  
4. Identities & Access – Service Principal (SPN) and User-Assigned Managed Identity (UAMI) with appropriate RBAC permissions.

# 3. Workflow

1. Pipeline Trigger – Developer triggers ADO pipeline manually or via CI/CD schedule.  
2. Image Build – Pipeline reads AIB build template from GitHub repo. Secrets are replaced with Key Vault references. Customization scripts install required software (from Artifactory).  
3. Validation – Validator scripts ensure software installs correctly. Failures are logged and build stops.  
4. Publishing – Successfully built images are published into Compute Gallery. Versioning is automatic.  
5. Logging – Build logs are pushed into Storage Account for review.

# 4. Setup Checklist for CIO Teams

✅ Request/create new ADO project in OneDevOps.  
✅ Create new GitHub repo (recommended).  
✅ Provision Storage Account (with lifecycle policies for log retention).  
✅ Provision Key Vault (store secrets like Artifactory creds).  
✅ Provision Compute Gallery (define image definitions for OS types).  
✅ Create Service Principal (SPN) and grant access to: Resource Group, Key Vault, Storage Account, Compute Gallery.  
✅ Create User Assigned Managed Identity (UAMI) for AIB with same access.  
✅ Share SPN details (App ID, Tenant ID, Secret) securely with ADO pipeline team.

# 5. Best Practices

- RBAC Isolation – Always isolate ADO projects and GitHub repos for image builds.  
- Secret Hygiene – Do not hardcode secrets in templates/scripts. Always reference Key Vault.  
- Version Control – Maintain clear branching strategy in GitHub repo for image templates.  
- Auditability – Review logs in Storage regularly to track failures and usage.  
- Image Lifecycle – Set image version expiry in Compute Gallery to avoid stale images.  
- Automation First – Do not manually publish images; always rely on ADO pipelines.

# 6. Reference Architecture Diagram

Developer (ADO Pipeline Trigger)  
 |  
 v  
ADO Pipeline (YAML) ----> GitHub Repo (Template + Scripts)  
 |  
 v  
Azure Image Builder (AIB)  
 | | |  
 v v v  
Key Vault Artifactory Storage Account (Logs)  
 |  
 v  
Compute Gallery (Final Images)

# 7. Summary

With this solution, CIO teams only need to set up the environment (ADO project, GitHub repo, Storage, Key Vault, Compute Gallery, Identities) once. From then on, ADO + AIB pipelines fully automate image building, validation, and publishing, ensuring compliance, consistency, and security.

# 8. Automation Matrix: Immutable vs CIO Setup vs Recurring

|  |  |  |  |
| --- | --- | --- | --- |
| Area | Automated in Immutable | CIO One-time Setup | Recurring (Ops / Dev) |
| Source Control & Repo Bootstrap | Reference repo contains pipeline YAML, AIB template, customization & validator scripts. Bootstrap process is scripted (zip import). | Provision a dedicated GitHub repo for each image stream. Enable branch protections. | For each new image line: import latest reference zip; follow versioning/branching policy. |
| ADO Pipelines | Pipeline templates and stages (validate → build → publish → notify) are defined as code. Variable groups supported. | Request new ADO project in OneDevOps (recommended) and create a Service Connection bound to SPN. | Trigger builds for updates; update variables; review run results. |
| AIB Staging Resource Groups | Selection of staging RG is handled by pipeline/AIB config. No manual creation required. | — (12 staging RGs pre-created centrally) | Monitor quota/cleanup if a staging RG is stuck. Raise ticket if contention occurs. |
| Identities & RBAC | Pipeline consumes SPN/UAMI once created; role assignment usage is code-driven. | Create SPN and User-Assigned MI. Apply RBAC using provided JSON templates. | Rotate credentials regularly. Review RBAC quarterly. |
| Secrets Management | Key Vault references injected at runtime; no plaintext secrets in code. | Provision Key Vault; store secrets (Artifactory creds, admin passwords, tokens). | Rotate secrets; validate Key Vault access policy and purge protection. |
| Software & Binaries | Download from Artifactory during build via scripts (with KV-provided credentials). | Ensure Artifactory repo + network access (FW rules/private endpoints as needed). | Update installer versions and checksums; deprecate old packages. |
| Build & Publish | AIB performs image build, runs customization/validators, and publishes to Compute Gallery. | Provision Compute Gallery and image definitions (OS SKU variants). | Promote versions across rings; retire old versions; enforce version retention. |
| Logging & Monitoring | Build and validator logs automatically pushed to Storage Account. | Provision Storage Account + lifecycle rules (e.g., 90-day retention). | Review logs per build; alert on failures; track costs. |
| Governance & Compliance | Policies enforced via pipeline checks and validators. | Define naming standards, tags, policy exemptions as required. | Periodic audits; update policies/validators when standards change. |
| Consumption | Published images are discoverable via Compute Gallery. | Document how consuming teams reference gallery images (VMSS, VM). | Communicate new versions; update consumers’ image version/pinning strategy. |

# 9. Pre-created AIB Staging Resource Groups

Twelve (12) AIB staging resource groups are pre-created and managed centrally. These are used transiently by Azure Image Builder during provisioning and cleanup. Developers do NOT need to create or manage these RGs.  
Example naming pattern (adjust to your environment): rg-aib-staging-01 through rg-aib-staging-12.  
If a build appears stuck (resources lingering in a staging RG), open a ticket to the platform team for cleanup.

# 10. RBAC Templates (JSON) for Managed Identity & ADO SPN

Use the ARM templates below to assign the minimum required roles to the User-Assigned Managed Identity (UAMI) used by AIB and to the Azure DevOps Service Principal (SPN). Deploy at the subscription scope.  
  
Built-in role IDs used:  
- Contributor: b24988ac-6180-42a0-ab88-20f7382dd24c  
- Storage Blob Data Contributor: ba92f5b4-2d11-453d-a403-e96b0029c9fe  
- Key Vault Secrets User: 4633458b-17de-408a-b874-0445c86b69e6  
  
Deployment (Azure CLI):  
az deployment sub create --location <region> --template-file rbac-mi.json --parameters principalId=<UAMI\_OBJECT\_ID> keyVaultId=<KV\_RESOURCE\_ID> storageAccountId=<SA\_RESOURCE\_ID> galleryId=<GALLERY\_RESOURCE\_ID>  
az deployment sub create --location <region> --template-file rbac-ado-spn.json --parameters principalId=<SPN\_OBJECT\_ID> keyVaultId=<KV\_RESOURCE\_ID> storageAccountId=<SA\_RESOURCE\_ID> galleryId=<GALLERY\_RESOURCE\_ID>

Files created:  
- rbac-mi.json (UAMI): /mnt/data/rbac-mi.json  
- rbac-ado-spn.json (ADO SPN): /mnt/data/rbac-ado-spn.json

# 11. Developer Step-by-Step (CIO Team)

1. Create or get access to a dedicated GitHub repository for your image stream. Recommendation: use a fresh repo.

2. Bootstrap from reference: Download the ZIP from the reference repository and import it.  
Placeholder (replace with actual): REFERENCE\_REPO\_URL\_GOES\_HERE

3. In your new repo, review the folder structure:  
 /aib/ (AIB template and parameters)  
 /customizations/ (installers, scripts)  
 /validators/ (post-build checks)  
 /pipelines/ (pipeline YAML templates)  
 azure-pipelines.yml (entrypoint)

4. Update configuration variables in azure-pipelines.yml or variable group:  
 subscriptionId, resourceGroup, location, galleryName, imageDefinitionName, stagingRgSelector (if applicable),  
 keyVaultName, storageAccountName, artifactoryUrl, artifactorySecretName, serviceConnectionName.

5. Ensure the ADO Service Connection exists and points to the correct subscription using the SPN provided by the CIO platform team.

6. Store required secrets in Key Vault (e.g., Artifactory credentials) with the exact secret names referenced by the pipeline.

7. Commit and push. In ADO, import the repo or connect GitHub; create a pipeline using azure-pipelines.yml.

8. Run the pipeline: it will Validate → Build → Publish the image via AIB and push logs to Storage Account.

9. Verify in Azure: Confirm a new image version in the Compute Gallery and check validator logs in the Storage Account.

10. Consume the image: update your VM/VMSS definitions to reference the new gallery image version or use latest if your policy allows.

11. Ongoing: bump software versions, update scripts, rotate secrets, and prune old gallery versions per retention policy.

# 12. Repo Bootstrap Notes & Placeholders

When a new GitHub repo is created for a custom image, simply upload the provided ZIP from the reference repo. This includes all workflow steps as code (ADO pipeline YAML, AIB template, customization & validator scripts). Replace the placeholders below with environment-specific values:  
  
<REFERENCE\_REPO\_URL>  
<ADO\_SERVICE\_CONNECTION\_NAME>  
<SUBSCRIPTION\_ID>  
<RESOURCE\_GROUP>  
<LOCATION>  
<GALLERY\_NAME>  
<IMAGE\_DEFINITION\_NAME>  
<KEYVAULT\_NAME>  
<STORAGE\_ACCOUNT\_NAME>  
<ARTIFACTORY\_URL>  
<ARTIFACTORY\_SECRET\_NAME>