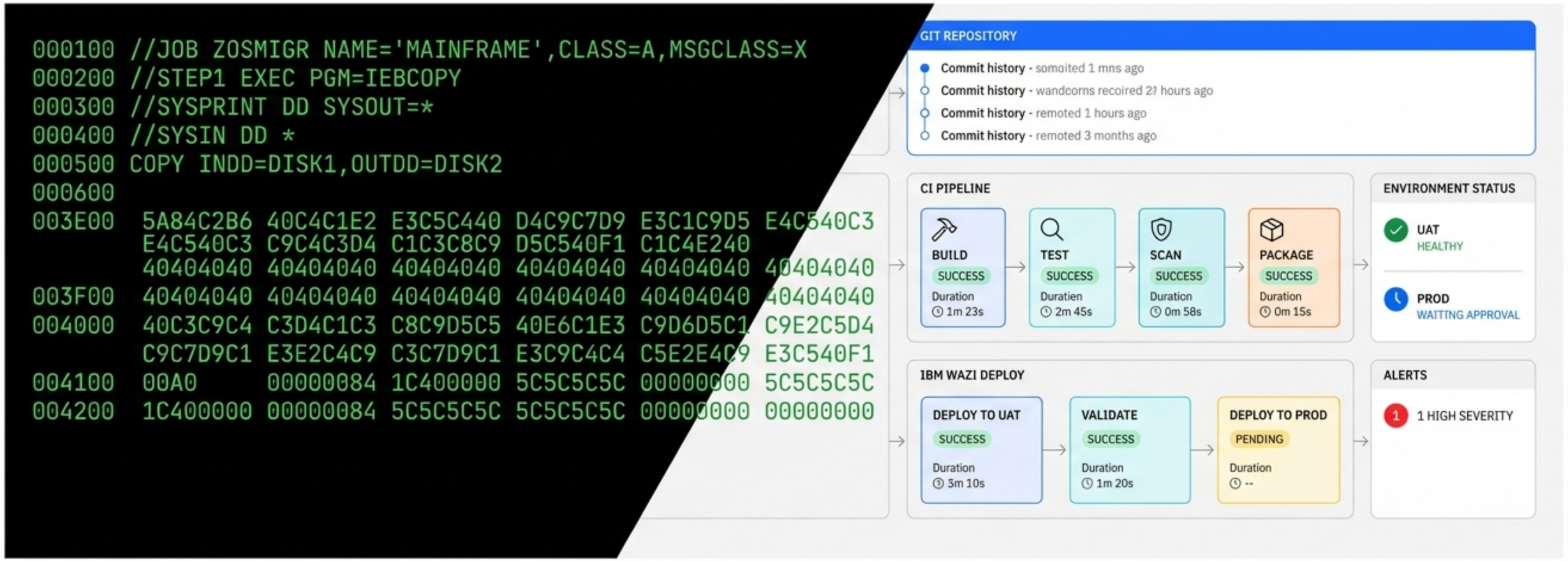
Enterprise Editorial



Modernizing the Mainframe: A Strategic Risk Assessment

Analyzing Vendor, Compliance, and Operational Stability in the migration to Git, IBM Dependency Based Build, and IBM Wazi Deploy.

This document evaluates the specific risks associated with migrating z/OS artifacts to an open toolchain and details the architectural mitigations provided by the solution stack.

The Verdict: Modernization Enhances Control

Core Message

Moving from a monolithic library manager to an open, layered stack (Git + DBB + Wazi) does not compromise stability. Instead, it replaces opaque legacy processes with transparent, auditable automation.

The Shift:

- **From:** Proprietary Legacy SCM & Manual Deployment
- **To:** Open Standards (Git, Python, Groovy, Ansible)

Auditability: Enhanced via granular Git history and immutable Wazi evidence files.

Security: Leverages existing z/OS security (RACF) while adding modern pipeline governance.

Resilience: Automated ‘Impact Builds’ and static deployments reduce human error.

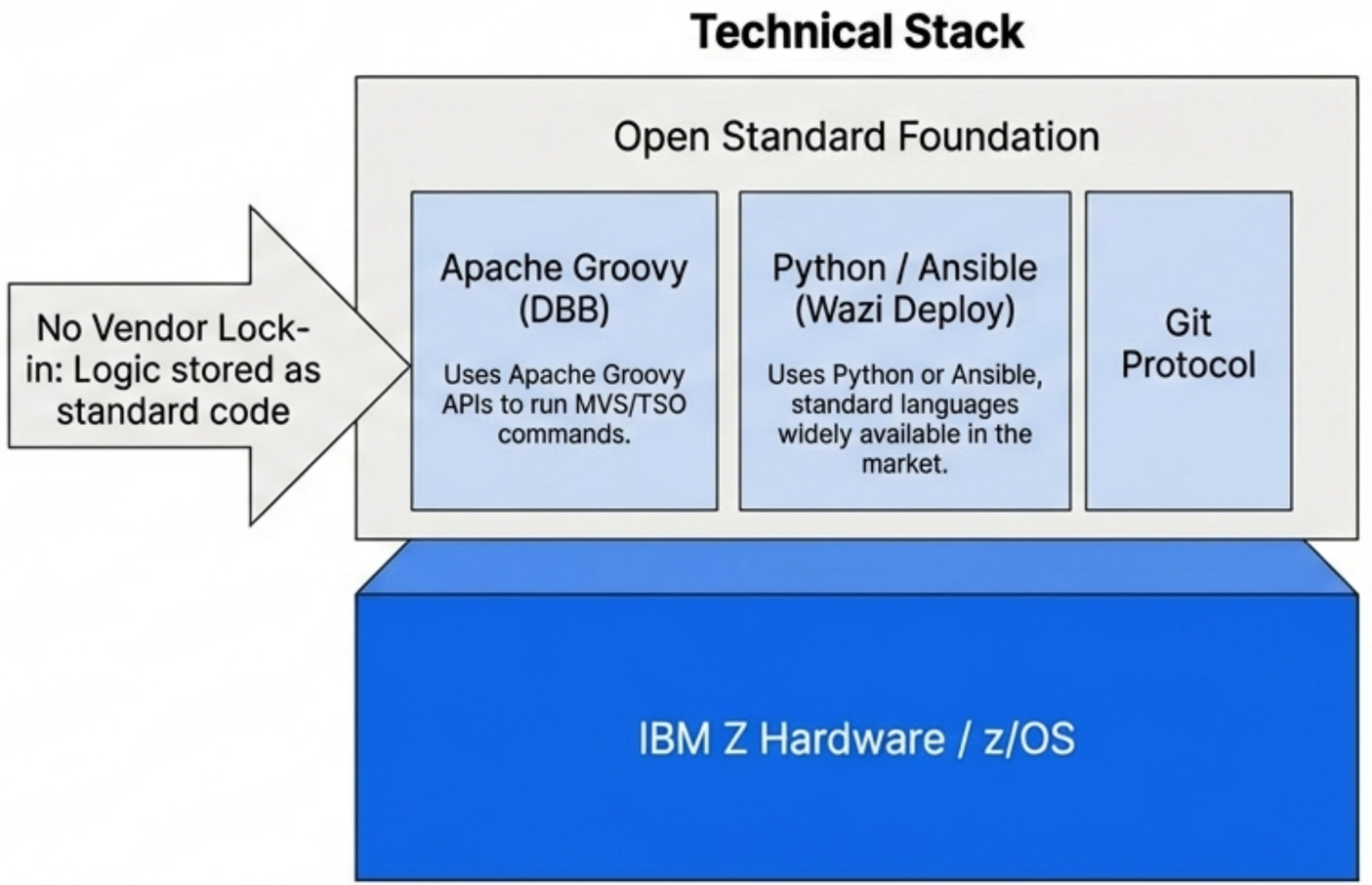
Vendor Stability & Architecture

Risk: Is this new stack stable? Will we be locked into a niche tool?

Mitigation Strategy:

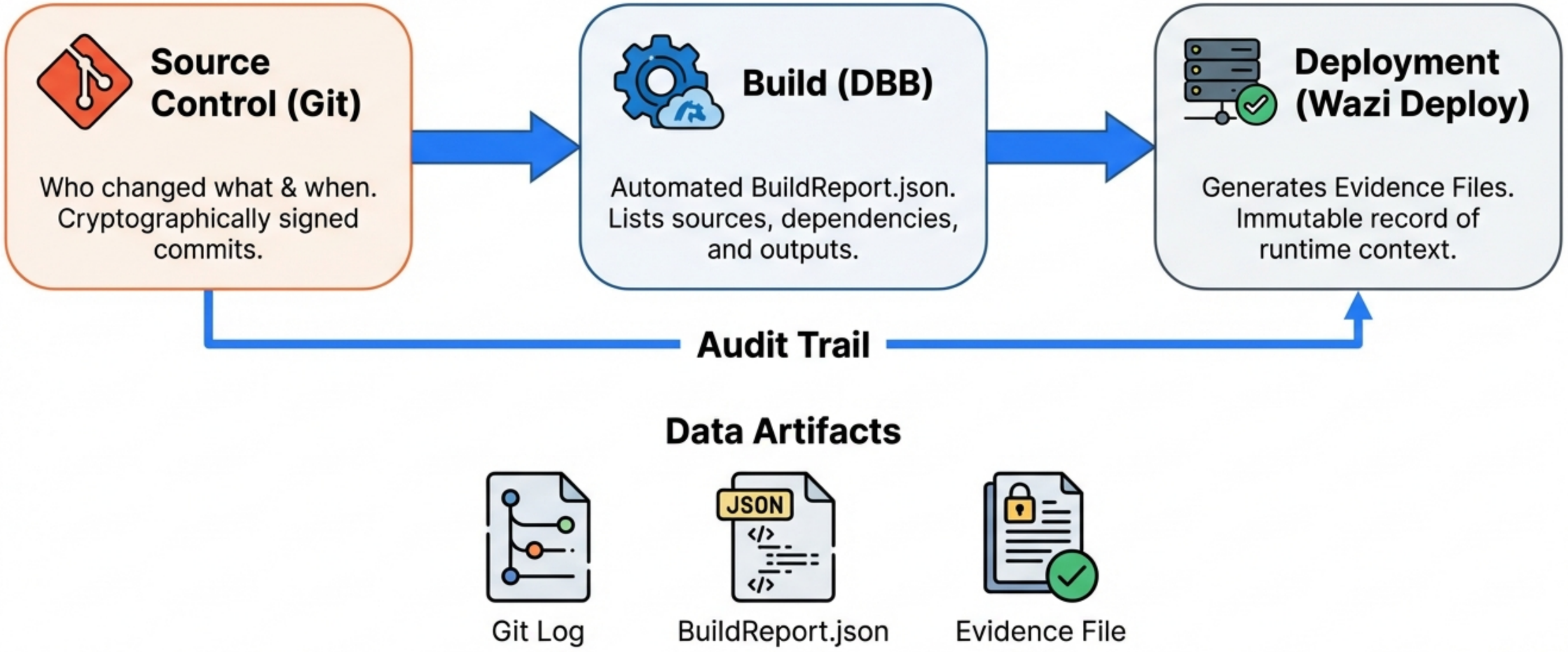
- **Open Standard Foundation:** The solution is not a 'black box'. Build and deploy logic is stored as readable code.
- **IBM Dependency Based Build (DBB):** Uses Apache Groovy APIs to run MVS/TSO commands.
- **IBM Wazi Deploy:** Uses Python or Ansible, standard languages widely available in the market.

IBM Commitment: Supported by IBM (Program Number 5737-K80), ensuring enterprise-grade support and lifecycle management (Java 11/17 support).



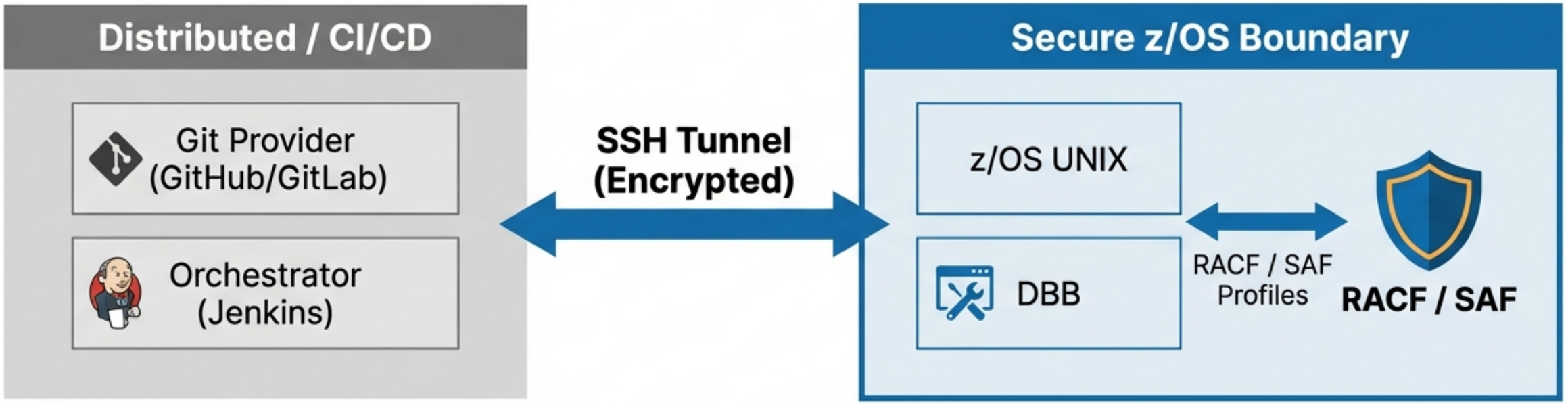
Compliance, Auditability & The Chain of Custody

Risk: Will we lose traceability compared to our legacy SCM logs?



Z-Centric Security Architecture

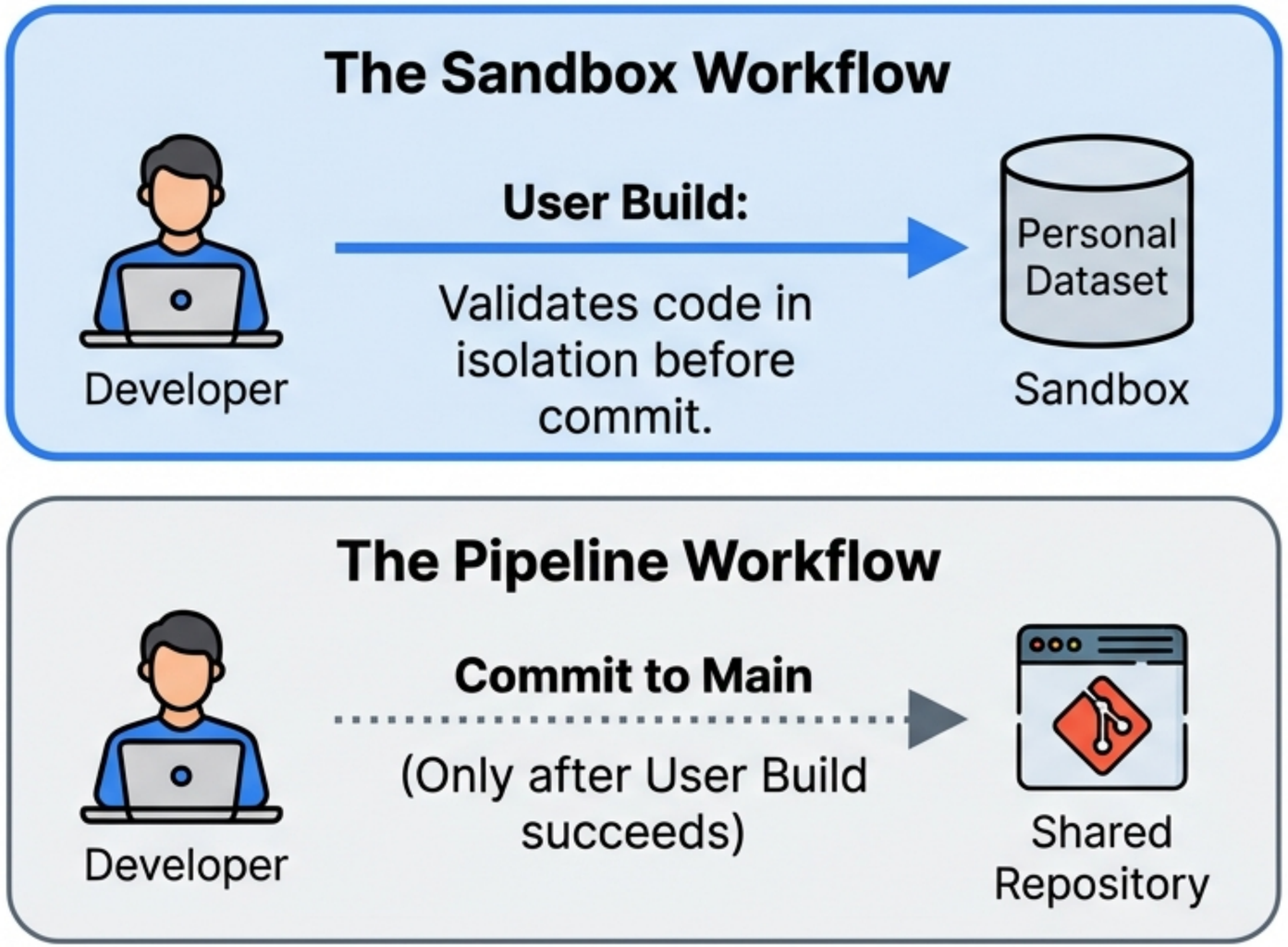
Risk: Does opening the mainframe to distributed tools compromise security?



- **Role-Based Access Control (RBAC)**: Security 'Shifted Left'. Git provider enforces separation of duties (e.g., protected branches) before code triggers the pipeline.
- **Host Security Integration**: Tools run on z/OS UNIX but fully respect RACF, CA-ACF2, and CA-Top Secret.
- **Protected Execution**: Pipeline actions run under specific functional IDs, preventing direct developer access to production.

Developer Stability: The “User Build”

Risk: Will this disrupt our developers or break the shared build pipeline?



🔒 **Sandboxed Development:** Developers run builds in private workspaces (IDz/VS Code), isolating errors.

🔍 **Dependency Scanning**: DBB automatically analyzes relationships (Copybooks, sub-modules).

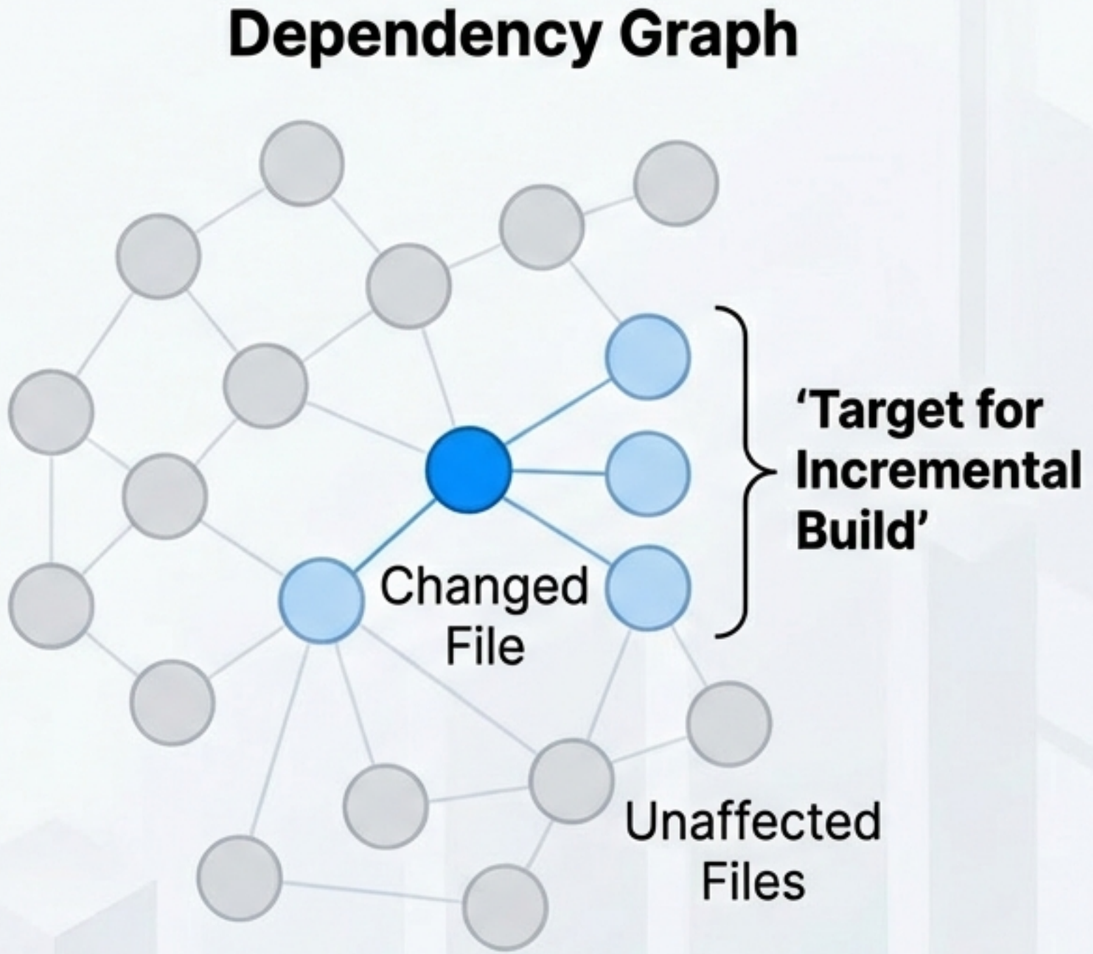
⚡ **Immediate Feedback**: Compilation errors detected instantly in the IDE, not in a nightly batch.

Performance Efficiency & “Impact Builds”

Risk: Will builds take too long? Will we consume too much CPU?

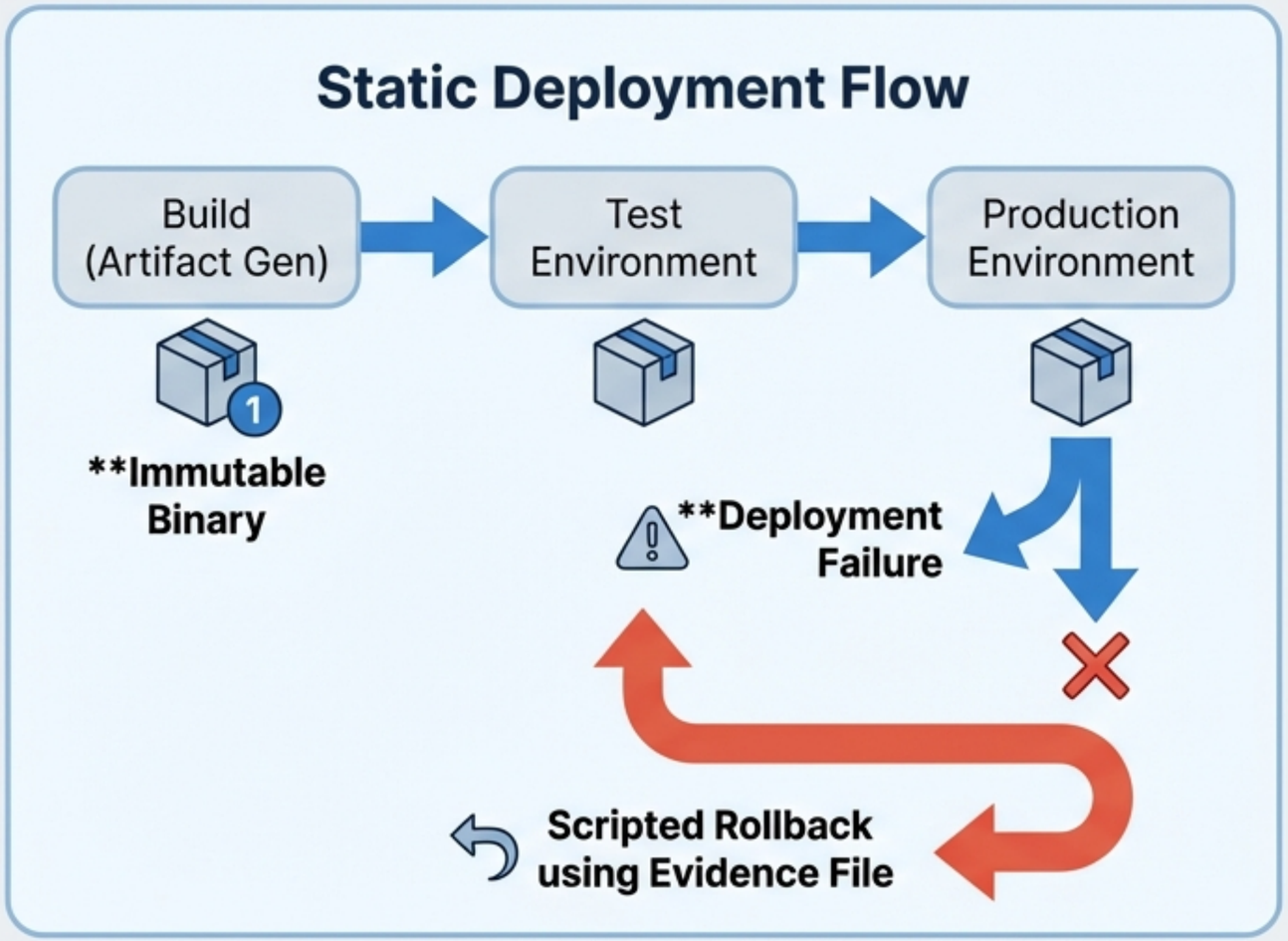
The Intelligent Build

- **Incremental Logic:** DBB utilizes ‘Impact Builds’ to identify changed files and *only* rebuild specific programs impacted by those changes.
- **Smart Resolution:** Automatically resolves dependencies (like Copybooks) and adds referencing programs to the build list.
- **Efficiency:** Drastic reduction in CPU consumption (MIPS) compared to full-system recompilations.



Deployment Reliability & Rollback

Risk: What if a deployment fails in Production?



- **Static Deployment**: Wazi Deploy uses a 'Manifest' to deploy a frozen, consistent package across all environments.
- **Scripted Consistency**: Deployments orchestrated via Python/Ansible. No manual copying.
- **Validation**: Pre-flight controls check environment readiness (e.g., disk space) before execution.
- **Traceability**: Every deployment generates an **Evidence File** for root-cause analysis.

Acceptance Risk: The Cultural Transformation

Bridging the gap between veteran expertise and new talent.

The Mainframe Veteran

- Gains automated Impact Analysis.
- Removes manual dependency tracking drudgery.
- Modern editors available.

Unified Pipeline

Silo Busting: Mainframe developers join the enterprise-wide DevOps ecosystem.

Talent Attraction: Modern tooling is essential to attract and retain new developer talent (IDC Research).

The Next-Gen Developer

- Uses familiar tools (Git, VS Code, Python).
- Lowers barrier to entry.
- Attracted by modern DevOps stack.

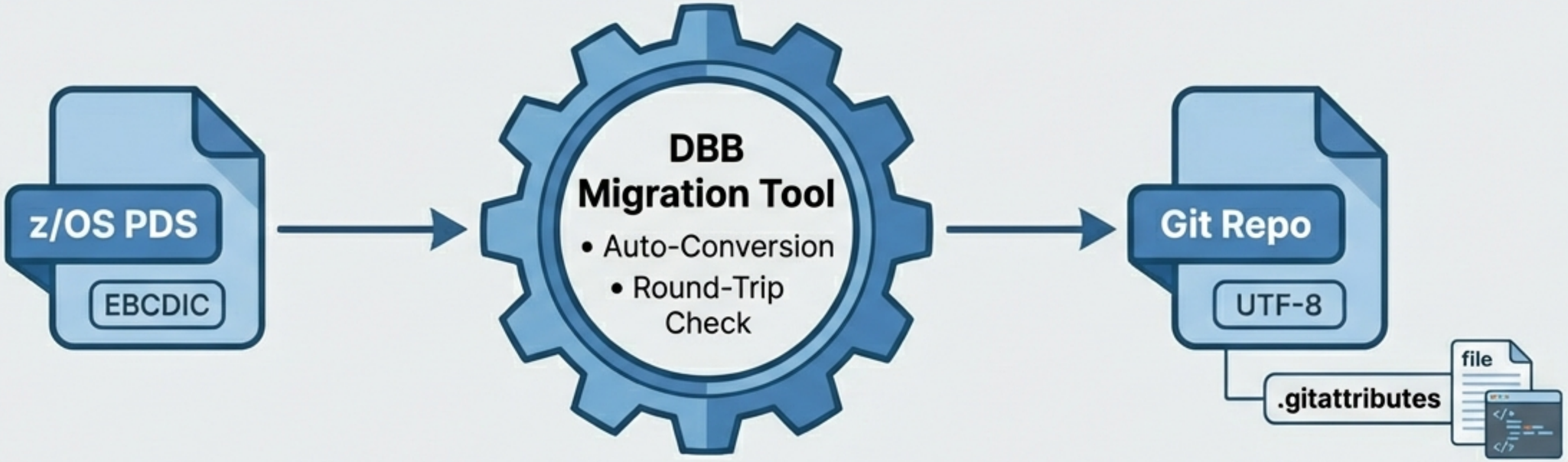
Transformation Strategy: Phased vs. Big Bang

Risk: Changing everything at once will cause chaos.

- **The Iterative Approach**: Avoid the ‘Big Bang’. Migrate one application pilot team at a time.
- **Co-existence:** The DBB Migration Tool supports a hybrid state where legacy PDS members coexist with Git-migrated code.

Technical Migration Risks (Data Integrity)

Risk: Data corruption moving from EBCDIC (Mainframe) to UTF-8 (Git).



- **Automated Conversion:** Handles code page conversion from z/OS PDS to z/OS UNIX (zFS).
- **Round-tripping Safety:** Automatically scans for ‘non-roundtrippable’ characters before migration to prevent corruption.
- **.gitattributes:** Automatically generates attributes files to strictly define text vs. binary handling.

Summary Risk Matrix

Risk Area	Specific Concern	Primary Mitigation
Vendor	Lock-in to niche tools	**Open Standards** (Python/Groovy/Ansible)
Compliance	Loss of audit logs	**DBB Build Reports & Wazi Evidence Files**
Security	Distributed vulnerabilities	**RACF Integration & SSH Tunneling**
Performance	High MIPS consumption	**DBB Impact Build Analysis**
Stability	Broken shared builds	**User Builds** (Sandboxed Development)
Data	Character corruption	**Automated Round-trip Integrity Checks**

The Greater Risk: Inaction

The cost of maintaining the status quo is rising.

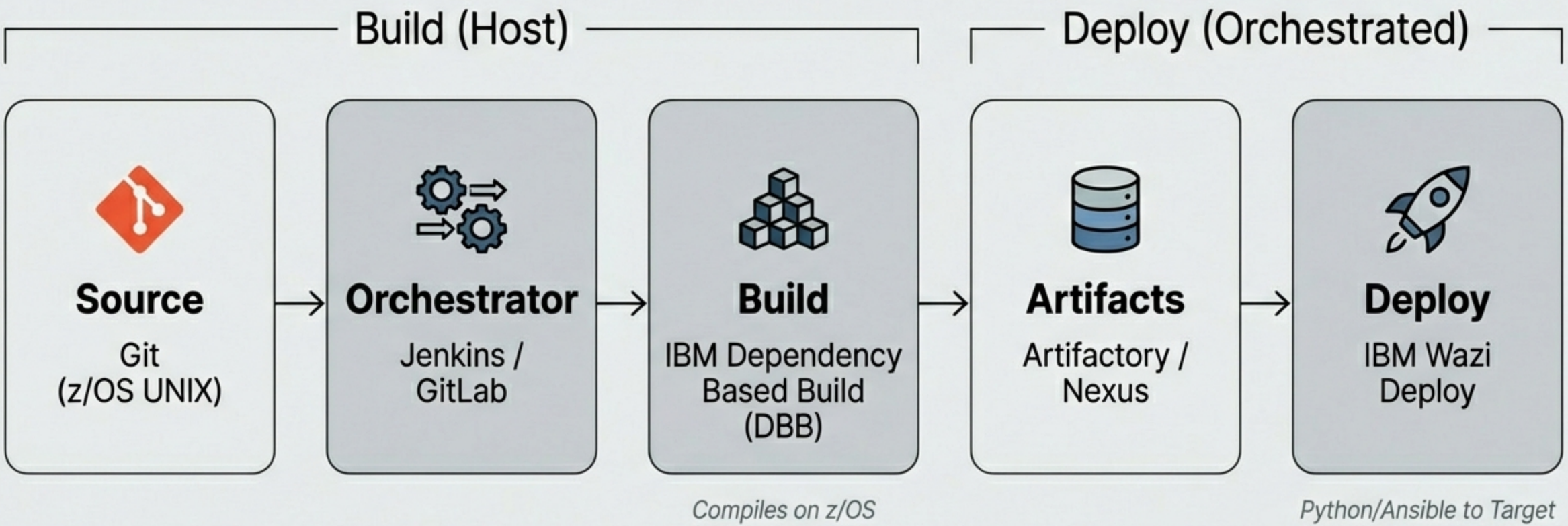
Talent Risk: The ‘graying hair’ crisis. New talent will not adopt ISPF/Green Screen tools.

Velocity Risk: Manual processes and nightly batch windows cannot match digital business speed.

Integration Risk: Legacy silos isolate the mainframe from the hybrid cloud ecosystem.

Modernization is not just an upgrade; it is a survival strategy for the platform.

Appendix: The Modern Technical Stack



The Path Forward

Start with Discovery, Not Commitment.

Next Steps

1. **‘Value Stream Assessment:** Map your current development process efficiency (Milestone 1).
2. **‘Pilot Program:** Select one non-critical application for a DBB/Git pilot.
3. **‘Leverage the DAP:** Use the IBM Z DevOps Acceleration Program for no-charge training and assistance.

Align your tools with your talent.
Secure your platform for the future.