SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Culture Eats DevOps for Breakfast

(Is the traditional DevOps model a dead end for the mainframe?)

Gatien Dupré

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

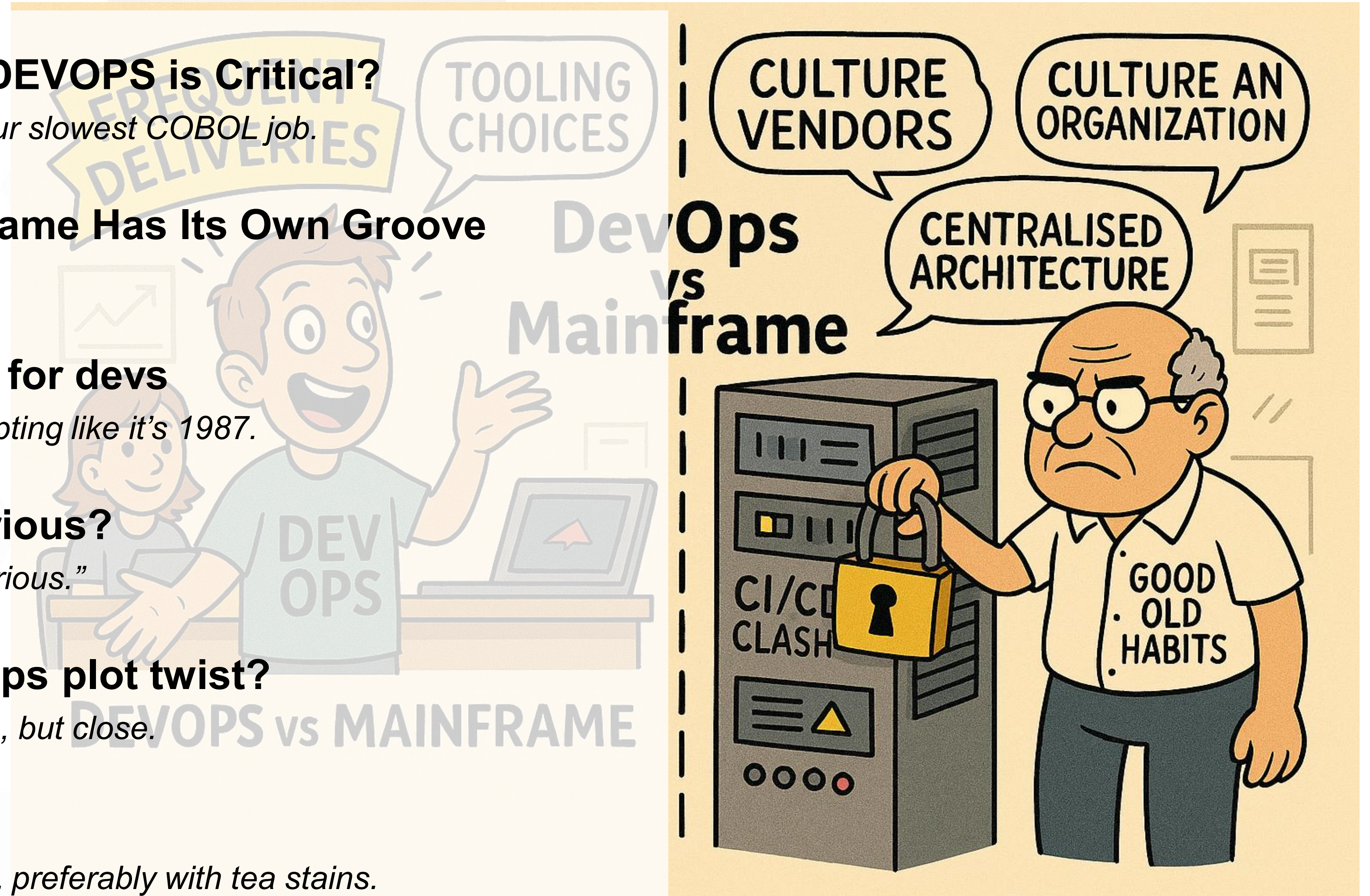
QUICK INTRODUCTION

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

The DevOps Paradox on the Mainframe

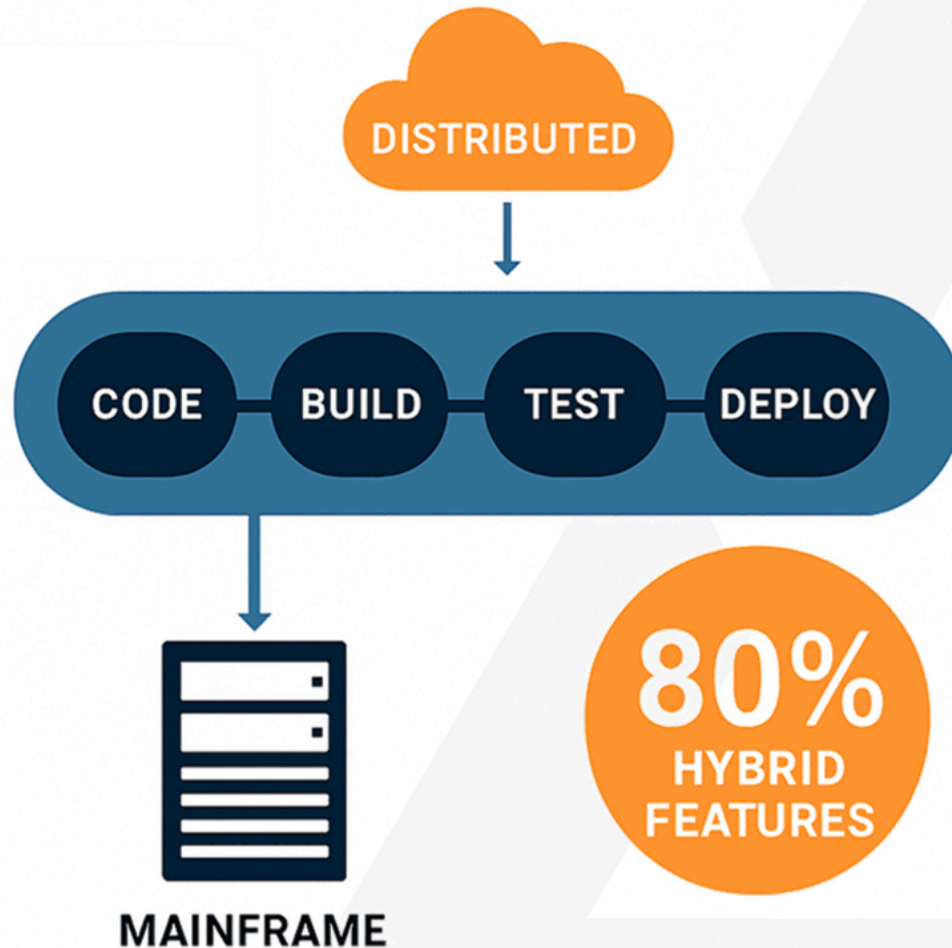
SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

- **Why integrating the Mainframe into DEVOPS is Critical?**
  Your delivery speed is only as fast as your slowest COBOL job.
- **CI/CD ≠ Copy-Paste: Because Mainframe Has Its Own Groove**
  Spoiler : Timeless beats trendy.
- **Tooling Choices: The butterfly effect for devs**
  One wrong tool and suddenly you're scripting like it's 1987.
- **2 Customer Experience: Fast and furious?**
  Well… more like "Fast-ish and politely furious."
- **Major Mainframe Vendors: The DevOps plot twist?**
  Hint: It's not quite Shakespearean drama, but close.
- **My personal CI/CD Magic graph**
  Because every Brit loves a good graph… preferably with tea stains.



Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Why integrating the Mainframe into DEVOPS is Critical ?



- Broken delivery Flow : Distibuted team use Git, Jenkins, Kubernetes…While Z teams rely on legacy tools and manual steps.
- Slower time-to-market : even if cloud apps deploy daily, mainframe dependencies delay production
- Lack of visibility : product teams can't track z components easily.

Your delivery speed is limited by your slowest component !

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

CI/CD ≠ COPY-PASTE: BECAUSE MAINFRAME HAS ITS OWN GROOVE.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

CI/CD ≠ Copy-Paste: Because Mainframe Has Its Own Groove.

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Distributed

Legacy

❌ Mistake #1: Copy-pasting distributed CI/CD onto the mainframe

- **In distributed systems:**

You often deploy by **promoting** an environment (e.g., staging becomes production).
Each application typically runs on its **own isolated infrastructure**.

- **On the mainframe:**

There is **only one production environment**, shared by all applications.
You **cannot promote** environments; every deployment affects the **entire system**.

👉 **Conclusion:** CI/CD for the mainframe must be **redesigned** to reflect its **shared, centralized architecture**—not copied from distributed models.

❌ Mistake #2: Confusing package with version as the deployment unit

- **In distributed systems:**

The **version** is the core unit of deployment.
Each release is **versioned**, traceable, and often deployed via containers or artifacts.

- On the mainframe:

The **package** is traditionally used as a **deployment vector**, bundling multiple components.
Packages may include elements from **multiple versions**, obscuring traceability.

👉 **Conclusion:** While the package can remain a **technical vehicle**, the deployment unit must be the **version**, enabling **automation**, **auditability**, and **DevOps compatibility**.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

CI/CD ≠ Copy-Paste: Because Mainframe Has Its Own Groove.

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Distributed

Mainframe CI/CD must be inspired by, but not copied from distributed or legacy models.

Legacy

❌ Mistake #3: Rebuilding the entire application at every deployment

- **In distributed systems:**

Applications are often **stateless** and loosely coupled.
Rebuilding and redeploying the **entire app** is common and relatively safe.

- **On the mainframe:**

Applications are **highly interconnected**, sharing datasets, libraries, and components.
Rebuilding everything is **risky**, **complex**, and often **impractical**.

👉 **Conclusion:** Mainframe deployments must be **incremental**, deploying **only what has changed**, with **strict traceability**.

❌ Mistake #4+5: Treating CI and CD as separate phases and Splitting CI (Dev) and CD (Ops) by responsibility

- **In distributed systems:**

CI (build/test) and CD (deploy) are often decoupled.
CD can involve promoting containers or binaries across environments.

- **On Legacy mainframe:**

QA and Prod environments differ significantly from Dev: Different datasets, security rules, and operational constraints. Deployment requires **Ops involvement** for configuration and validation.
No simple “promotion” like in distributed pipelines.

👉 **Conclusion:** CI and CD are **tightly linked**. Mainframe CI/CD = **Dev + Ops collaboration**.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

TOOLING CHOICES: THE BUTTERFLY EFFECT FOR DEVS

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Tooling Choices: The butterfly effect for devs

❌ **Mistake #6 : Ops choose the tools for Devs without Dev.**

- **Origin:** Traditionally, Operations decided which tools to deploy because they were responsible for availability and compliance.
- **Consequence:** Developers end up with tools that do not meet their productivity needs or integrate well into modern CI/CD pipelines..

👉 **Butterfly effect:** A poor tooling choice for developers creates friction throughout the entire cycle: less automation, more workarounds, and a slower DevOps adoption..

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 licen

Tooling Choices: The butterfly effect for devs

❌ **Mistake #7 : A contract signed for stability can become the prison of innovation.**

- **Origin:** On the mainframe, tools are often selected by Operations and locked in through long-term agreements—five-year contracts or even perpetual licenses.

- **Consequence:** When practices evolve (DevOps, CI/CD, automation), these tools can become obstacles. Yet challenging them is nearly impossible due to financial commitments and lengthy approval processes.

👉 **Butterfly effect:** A decision made for stability today can freeze innovation for years, forcing teams to work around outdated tools.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 licen

Tooling Choices: The butterfly effect for devs

❌ Mistake #8 : Choices are not driven by real priorities.

- **Origin:** Tooling decisions are often made without a clear prioritization framework (Must, Should, Could, Would). Instead of focusing on what truly matters for developers and the delivery pipeline, choices may be influenced by legacy habits or vendor relationships.
- **Consequence:** When practices evolve (DevOps, CI/CD, automation), these tools can become obstacles. Yet challenging them is nearly impossible due to financial commitments and lengthy approval processes.

👉 **Butterfly effect:** Misaligned priorities ripple through the organization—poor adoption, resistance to change, and a perception that DevOps “doesn’t work.”

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 licen

CUSTOMER EXPERIENCE

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Customer experience #1 – Z DevOps Service Initiative

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Context: Initiative from an IT outsourcing subsidiary serving a major French banking group.

Goal: Drive adoption of DevOps practices on the mainframe.

Solution:

- Creation of a **Z DevOps service offering (SaaS)** aligned with DevOps standards.
- Included best-in-class tools and a full 360° scope:
  - Integrated development environment
  - Continuous incremental deployment based on **Ansible**.

Challenge:

- The solution was designed by **Ops without involving developers (end users)**.
- Result: Misaligned with real practices and priorities → poor adoption.

Outcome:

- Despite being a "Rolls-Royce" inspired by distributed DevOps models, the offering was barely adopted by business entities.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Customer experience #2 - Bank chose to stay on the mainframe.

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Context: Major French bank chose to stay on the mainframe.

✅ **REX – Successful Mainframe Transformation**

- **Goal**: Modernize IT delivery and development practices.
- **Solution**: Eclipse-based IDE + virtualized testing platform, co-built with developers.
- **Adoption Strategy**:
  - Voluntary migration (legacy vs. modern platform).
  - **"Bug Hunt"** over 2 months → feedback collection + rewards.
  - **Outcome**: 90% adoption in 2 months.

❌ **Challenge Encountered**

- Git migration blocked by parallel project replacing Library Manager (LCM → Endevor, 4-year contract).
- Resulting in a hybrid model:
  - Agile up to integration
  - Package-based delivery to production.

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

I’M A LITTLE OVER — WANT ME TO LEAVE IT LIKE THAT?

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Major Mainframe Vendors: The DevOps plot twist?

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Why the "twist"? Because what looks like DevOps modernization often hides structural limits:

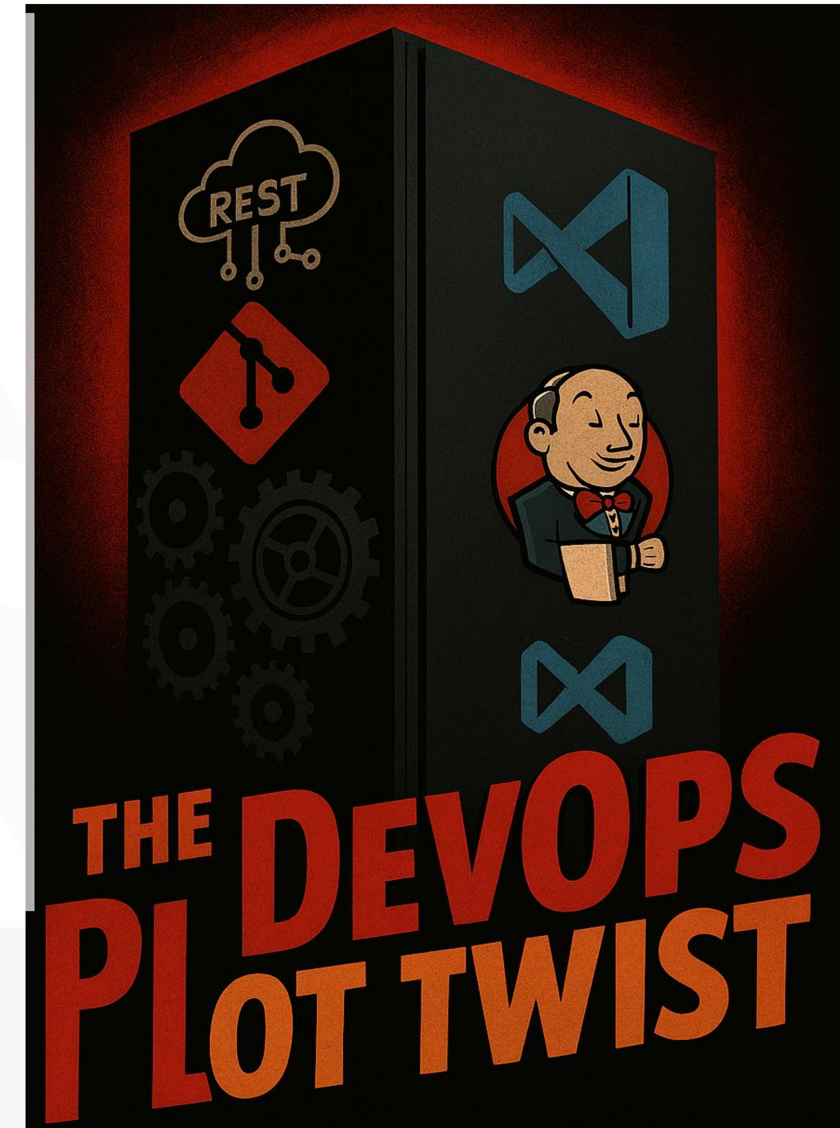
1. Cosmetic Modernization
REST APIs, Git plugins, VS Code connectors… but core logic remains sequential and manual.

2. Architecture Misfit
Tools built for long cycles and centralized governance struggle with modularity and automation.

3. Economic Model of Dependency
High license costs and integration services conflict with DevOps simplicity.

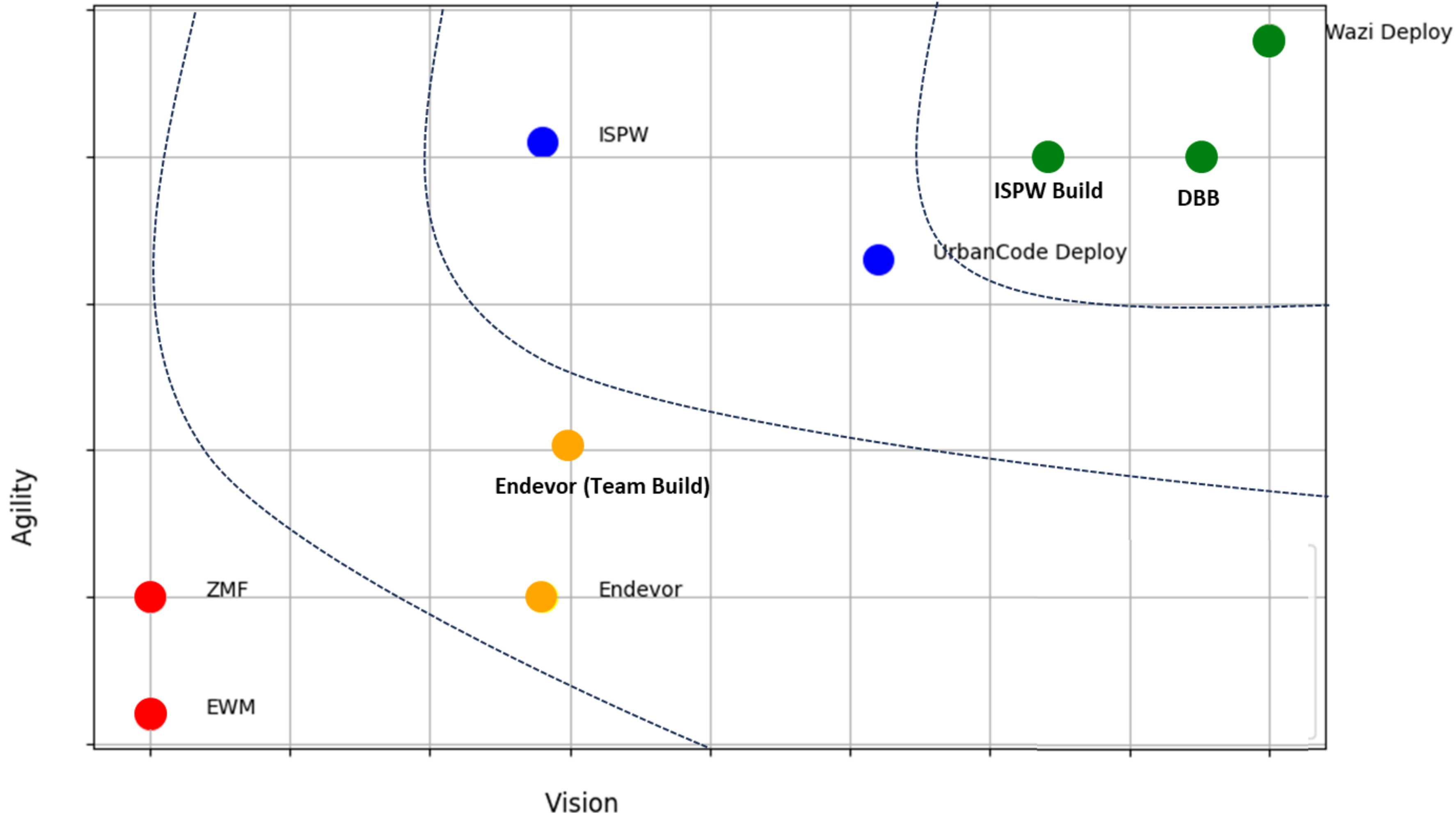
4. Misunderstood DevOps
Vendors equate DevOps with a Git plugin or Jenkins connector—without rethinking workflows.

Bottom line:
Most vendors deliver "DevOps marketing," not true DevOps principles. Enterprises end up hacking pipelines around tools never designed for agility.



Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Personal Insight: Evaluating Mainframe DevOps CI/CD backbone Products



- **Wazi Deploy** → *Cloud-native, YAML, Ansible*
- **DBB** → strong cloud-native build capabilities.
- **Endevor Team Build** → moderate agility.
- **ISPW Build** → good CI/CD integration.
- **UrbanCode Deploy** → *Multi-platform orchestrator*
- **ISPW** → *Git integration, CI/CD pipelines*
- **Endevor** → *Package-based delivery, limited agility*
- **EWM** → *Rigid workflows, ALM heritage*
- **ZMF** → *Classic waterfall model*

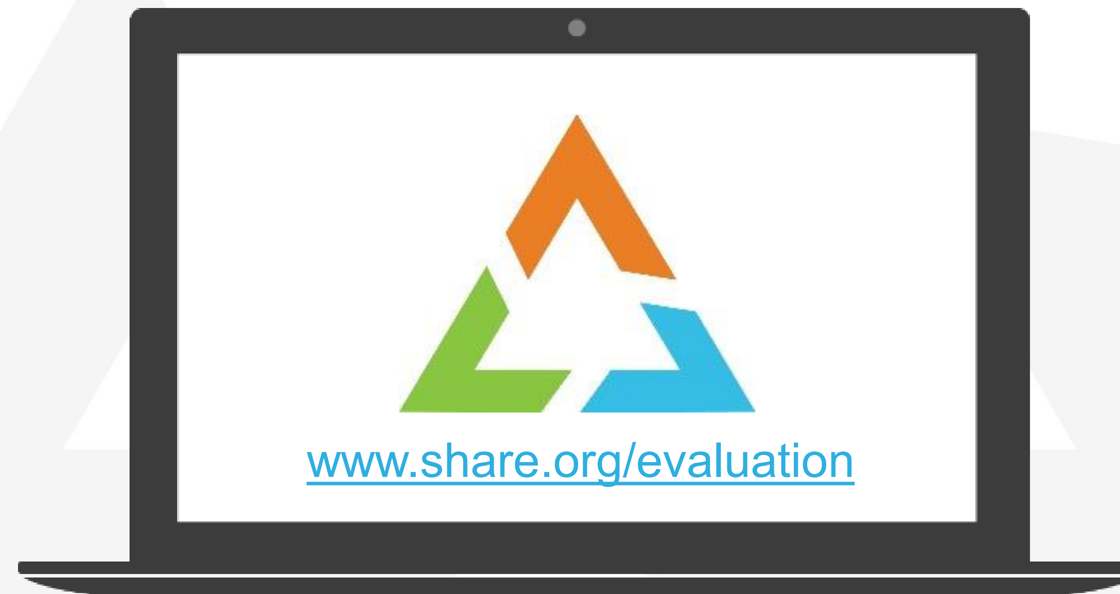
Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

Your feedback is important!

SHARE
EDUCATE ‣ NETWORK ‣ INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation



Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/

MERCI !!

Copyright© by SHARE Association Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/