From Tangled Systems to Paved Roads: Rethinking Architecture for Modern Applications

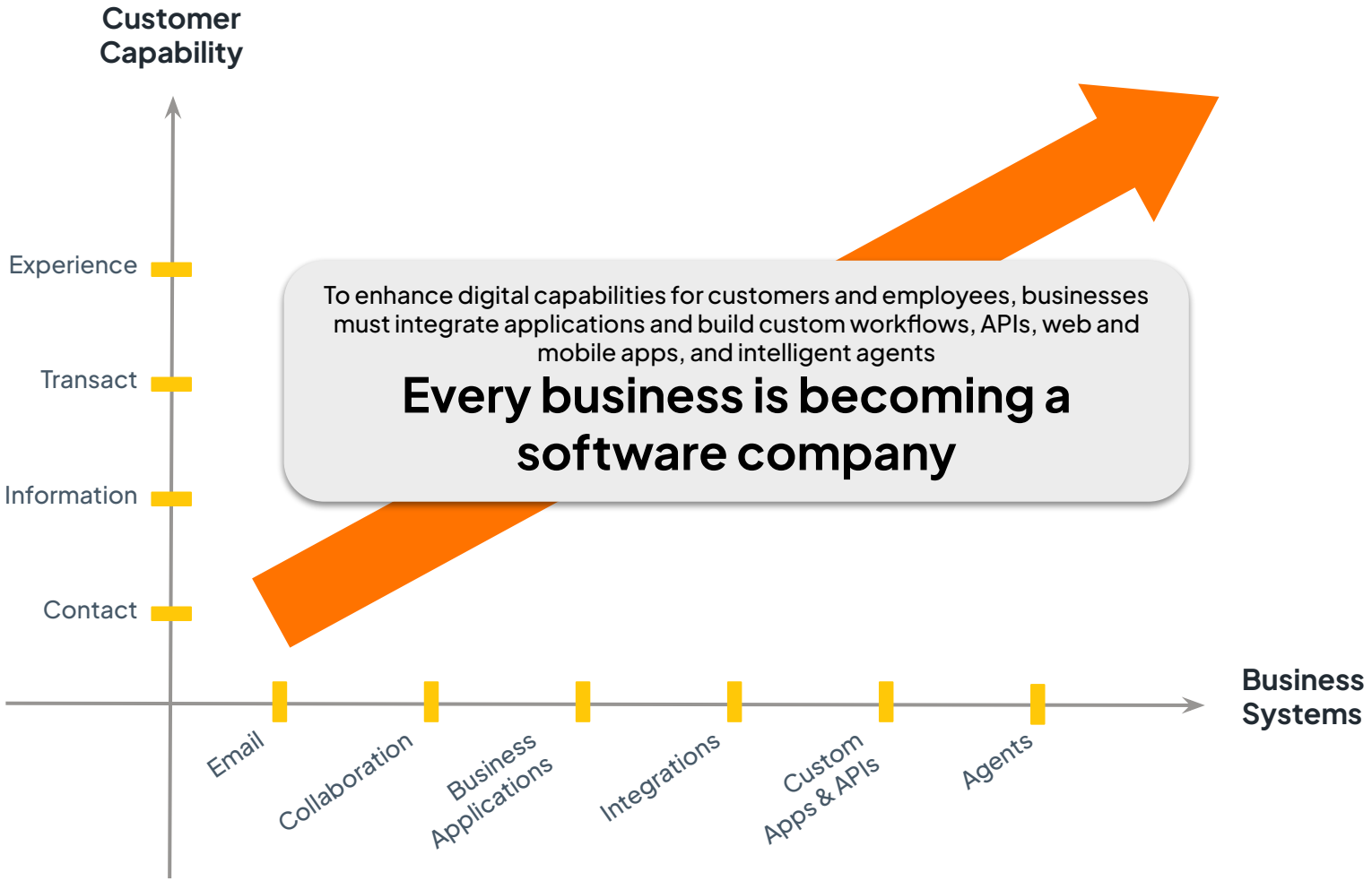


Ramith Jayasinghe
Director, Solutions Architecture
WSO2



Chintana Wilamuna
VP, Solutions Architecture
WSO2




Customer Capability
Experience
Transact
Information
Contact
To enhance digital capabilities for customers and employees, businesses must integrate applications and build custom workflows, APIs, web and mobile apps, and intelligent agents
Every business is becoming a software company
Email
Collaboration
Business Applications
Integrations
Custom Apps & APIs
Agents
Business Systems

Two developer stories

Developer A (Startup)	Developer B (Big corp)
Joins a new team	Joins a new team
Deploys a feature to production	Spend a month trying to run the app locally
On their first day	Gets stuck in a maze of docs, wikis, and tickets





The difference isn’t skill. It’s the path they have to walk

Architectural crossroads



Brownfield reality

- Manage and modernize existing, critical systems
- Manage technical debt
- The present

Greenfield innovation

- Build new, modern, cloud native apps
- Innovate and move faster
- The future

You cannot choose one. You have to master both

Agenda for the lab

By the end of the lab, you will have a strategic blueprint to navigate this dual reality

- Defining the landscape - Greenfield vs brownfield
- Modernization strategies - Paving the path forward
- A unifying solution - Introducing the platform
- Demo of how to deploy an existing app using an Internal Developer Platform
- Setting the stage for next sessions

Defining the landscape

Greenfield development

- It’s like building a house on an empty plot
- No legacy code, policies or constraints
- Freedom to choose best of breed projects/technology, architecture patterns and solutions
- Initial developer velocity is high
- Challenge
  - Without standards, teams will build very different solutions
  - Inconsistent interfaces, dependencies and deployment patterns

Brownfield development

- It’s like renovating a 100 year old building
- Must work around existing structures, old wirings, plumbing, gas lines etc..
- Cannot just knock down walls
- Change is very slow, complex and risky



Taken from https://www.countryliving.com

Real world brownfield problems – “old wiring”

- Monolithic architecture
  - With tightly coupled components
  - A small change requires deploying the entire system
- Technical debt
  - Years of quick fixes, outdated libraries, fragile code base
  - Lack of automated testing slows down pace of new changes or modifications
- High cognitive load
  - A developer needs to understand the entire complex system just to make one fix
- Inconsistent environments
  - “It works on machine” issues

From on-prem to the cloud

- Many of the brownfield systems live in on-premise, virtualized environments
- “Lift & shift” mistake
  - Moving a VM directly to the cloud
  - Problem: Just moved a “monolith” to a more expensive location
  - Don’t gain any cloud-native benefits
    - Scalability
    - Resilience
    - Cost-efficiency
- Simply changing the location doesn’t fix the architecture

Modernization strategies

Spectrum of modernization

- Modernization is not all-or-nothing. There’s a spectrum of options
- Rehost (Lift and shift)
- Replatform (Lift and reshape)
- Repurchase (Drop and shop)
- Retire (Decommission)
- Refactor / Rearchitect (focus of true modernization)

Strategy 1 – Quick fixes

Strategy	What it is	Pros/cons
Rehost (lift & shift)	Move the VM as-is to the cloud	✅ Fast ✅ Low effort ❌ Expensive ❌ No cloud benefits
Replatform	Make minor cloud optimizations like using a managed database (RDS)	✅ Some cloud benefits ❌ Still a monolith ❌ Doesn't fix core issues

These are stop-gaps, not long term solutions

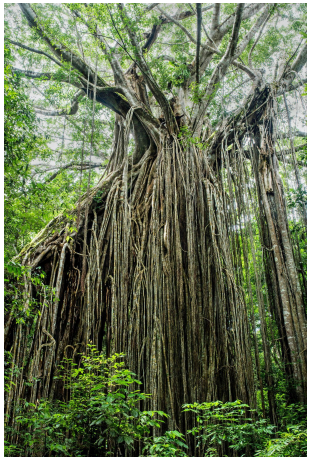
Strategy 2 – True modernization

- Refactor / Rearchitect
  - Changing the application's internal structure to be more modular, resilient and scalable
- The most difficult path
- Delivers the highest value

How do you do this without a 2-year rewrite?

The strangler fig pattern (strangler pattern)





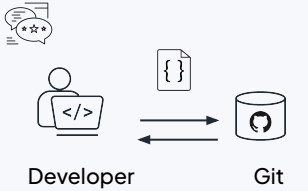
- A new vine grows over an old tree
- It starts small but over time it gets stronger and eventually replaces the old tree entirely
- For software - Incrementally and safely replace parts of a monolith with new, modern services

Photo by Miikka Luotio on Unsplash

Applying the strangler pattern

- Identify a module
  - Choose a single piece of functionality in the monolith (billing, orders etc…)
- Build new services
  - Create new, independent microservices
- Intercept traffic
  - Place a routing layer, API gateway or proxy in front of the monolith
- Redirect
  - Configure the router to send all submodule traffic to the new service
- Repeat
  - Continue carving out functionality piece by piece until the monolith is “strangled” and can be safely and completely retired
- End goal – Low risk. Deliver incremental value. No big bang rewrites.

The reality of software manufacturing in the cloud native era



Code reusability limited

Resource discovery delays

Collaborative development hindered

Testing cycles extended

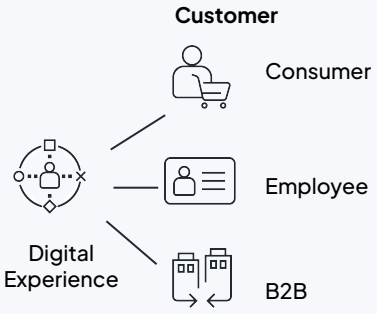
Slow product iteration

Value creation

You write code. You want to get it to the hands of the customer ASAP.

Pain points

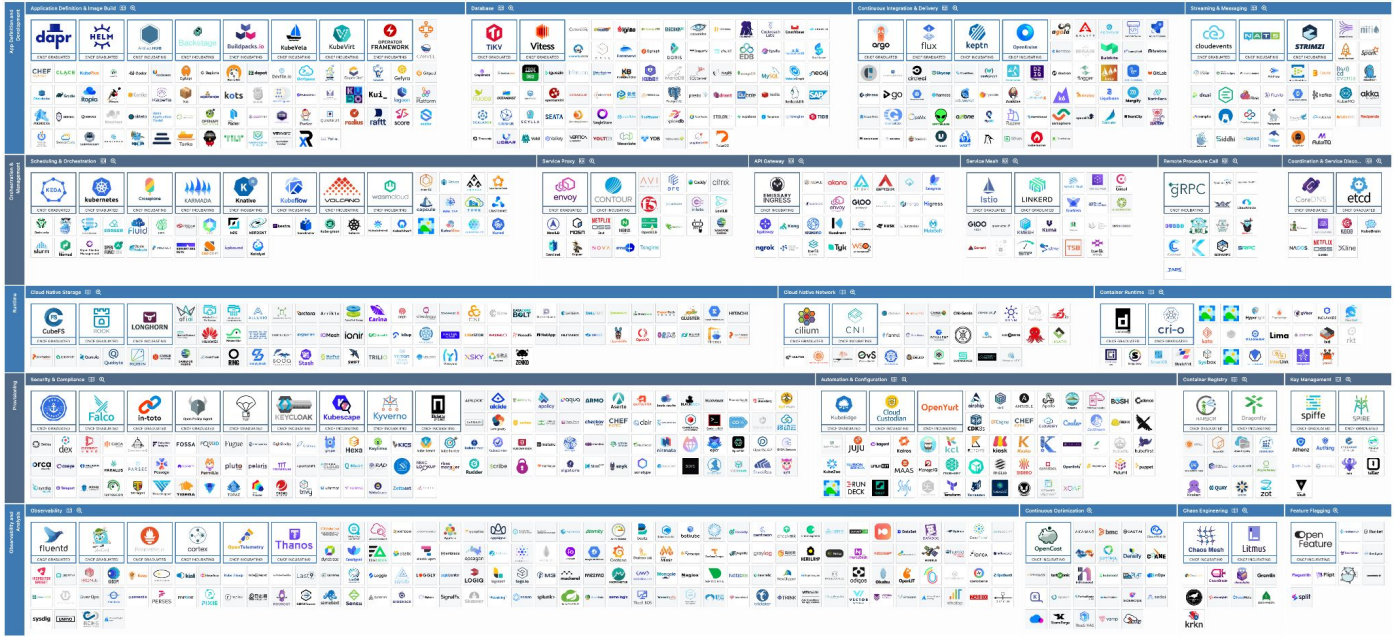
- Runtime architecture, CICD, DevOps, Environments, SecOps, Configuration Management, Version management, Testing, Observability, Analytics, SRE,
- Code writing also needs access to reuse
- DevOps goes to Canary etc.
- Self Service and policy-driven
- Zero Trust environment



Consumption

Problem – Cognitive overload

All technologies/projects in CNCF landscape (as of July 2025)



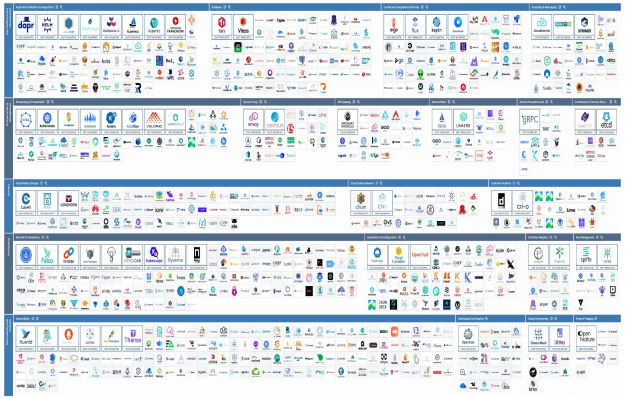
Should every dev team be an expert of all of this, just to deploy an app? NO!

Unifying solution

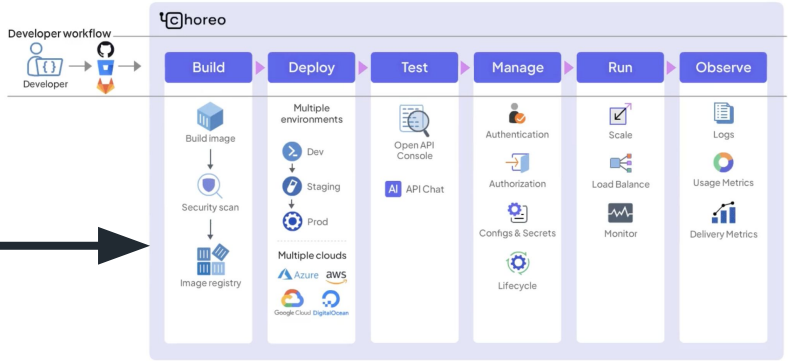
Solution? A paved road to production!

What if we stopped giving developers a box of tools and instead gave them a paved road?

Box of tools



A smooth, standardized, and safe path from their code to the cloud.



This is the core idea of an **Internal Developer Platform (IDP)**

What is an Internal Developer Platform (IDP)?

- An internal developer platform is a **product**, built by a **platform team**, whose customers are the **company's own developers**
- Goal - Radically simplify the developer experience by
  - Abstracting away infrastructure complexity
  - Enabling developer self-service
  - Enforcing standards and best practices **by default**

Core IDP capabilities

- Self-service
  - Developers can provision or reuse environments, databases and other resources via a simple UI or API. No more waiting on tickets.
- Golden paths for developers
  - Pre-configured, reusable templates for creating new apps
  - Built-in CI/CD, security and observability
- Abstraction layer
  - Developers interact with the platform, not with Kubernetes YAML or complex cloud APIs
- Built-in observability
  - Logs, metrics, and traces are automatically configured for every application deployed on the platform.

Core IDP capabilities

- Application config management
  - Unified way to manage properties, endpoints, credentials etc.
- Infrastructure management (for platform engineers)
  - Ability to manage K8s with higher level constructs
  - Environment management
  - Deployment and pipeline management
- Security and governance
  - Ingress/egress policies
  - Security, rate limiting and API best practices policies

Unifying brownfield and greenfield

- Internal Developer Platform is the single strategy that serve both realities
- For brownfield modernization
  - IDP provides the “paved road” for new microservices (using the stranger pattern)
  - Ensure every new service is built and deployed consistently
- For greenfield innovation
  - IDP is the central launchpad
  - Developers use the same “paved road” to go from idea to production in minutes
  - Same governance, security policies ensure consistent environment for teams

Demo – Deploying an existing application to an Internal Developer Platform

Before and after – The impact of an IDP

Before an IDP	After an IDP
High cognitive load	Low cognitive load
Slow, inconsistent deployments	Fast, standardized deployments
Siloed teams	Shared teams with consistent reuse with full autonomy
Mainly ticket-driven	Self-service

Conclusion and what's next

- You must master both brownfield and greenfield
  - They're two sides of the same coin
- True modernization requires strategic refactoring
  - Strangler pattern is key to doing this safely
- An internal developer platform provides the "paved road"
  - Unifying your architecture and accelerate development for all applications

Demo/Tutorial URL



We've defined the "Why". Now it's time for the "How"

- This session
  - We established the architectural problem
  - Modernization strategies
  - Why you need an Internal Developer Platform
- **11:00am** - Accelerate Cloud Native Development with Choreo
  - Detailed walkthrough of how an IDP speed up development
- **1.30pm** - Operating at Scale: Platform Engineering & Strategies with Choreo
  - Detailed information about how platform engineering can be accelerated by an IDP


Thank you!
20TH ANNIVERSARY EDITION
WSO2CONASIA
PLATFORMLESS MODERNIZATION