20TH ANNIVERSARY EDITION

WSO2CON ASIA

PLATFORMLESS MODERNIZATION

Principles of a Good Internal Developer Platform

Nuwan Dias

VP and Deputy CTO

WSO2

Why do you need an Internal Developer Platform?

Why do you need an Internal Developer Platform?

Without an IDP – It's Chaos!

Every team builds infrastructure, pipelines, APIs, security rules, etc. from scratch. Resulting in,

- Fragile snowflake environments
- Inconsistent user experiences
- High onboarding time
- Security loopholes

With an IDP – There's Order

Platform provides shared infrastructure, operating instructions, utilities, etc. Resulting in,

- Faster delivery
- Secure and compliant environments
- Governed and monitored processes
- Happy developers

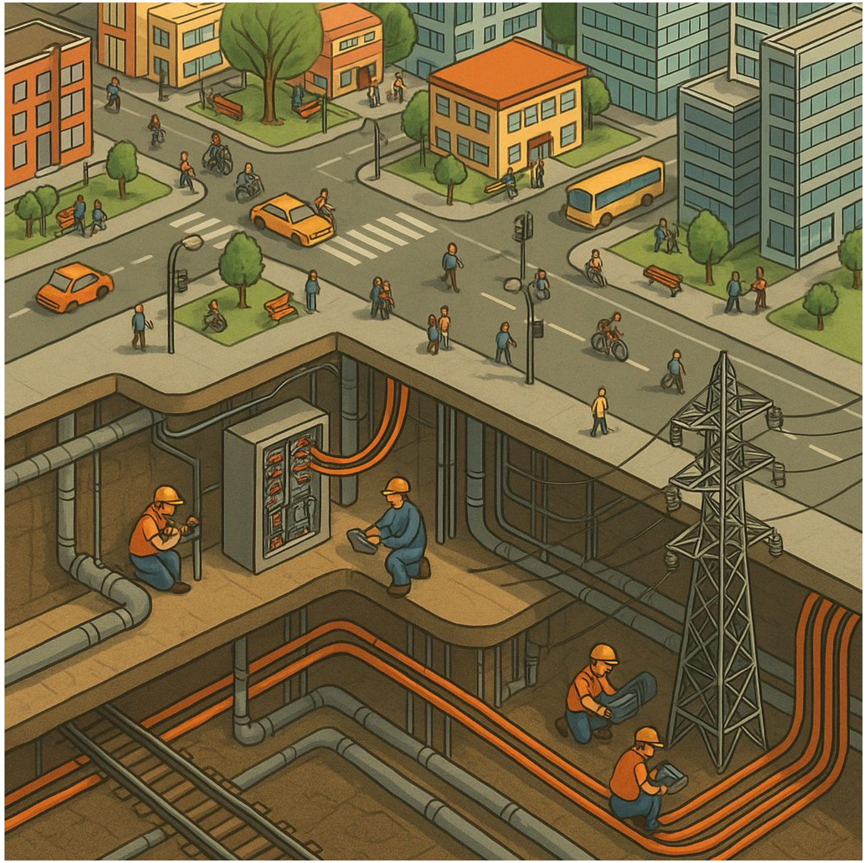
What should an IDP deliver and to Whom?

Developers are the City’s citizens and builders

- Builds homes, shops, schools (Applications).
- Rely on roads, power, water, zoning rules and emergency services.
- Focus on what they build, not how the infrastructure works.

Platform engineers are the city planners and utility operators

- Provide paved roads, electricity, zoning codes, shared spaces (Platform).
- They don’t build the buildings, but makes sure they are safe and connected.
- Uses automation and blueprints to replicate working neighborhoods.



It’s all about the Abstractions!

Quadruple layers of Enterprise Abstractions

1 Domains

A digital representation of the business.
Similar to a city, these are your neighborhoods, with style and purpose.

2 Business

Business APIs, App types, Data entities.
Similar to a city, these are your shops, houses, public transport, etc.

3 Deployment

CI/CD, Observability, Runtime Infra, Networking, Scaling, etc.
Similar to a city, these are your roads, electricity and plumbing.

4 Middleware

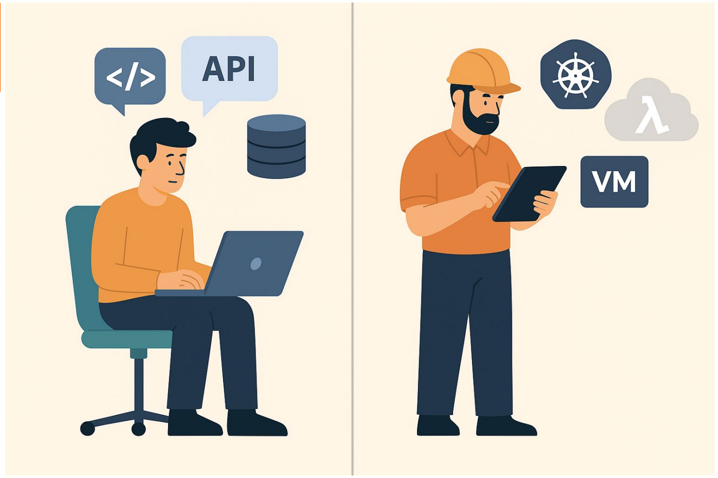
Connecting it all together. API Gateway, Ingress, Firewall, Service Mesh.
Similar to a city, these are your intersections, traffic lights, utilities and control center.

Key principles of a good Internal Developer Platform

1a. Separation of Concerns – *Developers vs Operators*

Design for who's doing what.

Question	Developer	Operator
What do they care about?	Applications, APIs, Databases, etc	Infrastructure, scaling, costs, etc
Need to know what runs the app?	No? Hide it.	Yes? Show it.
Cares about the programming language?	Yes? Show it.	No? Hide it.



"Abstractions aren't just technical, they're intentional boundaries."

1b. Separation of Concerns – *Control Plane vs Data Plane*

One platform. Many runtimes. One control point.

- The control plane is the *unified abstraction* layer for developers and operators.
- The data plane is where the actual application workloads run.
- The IDP must provide a universal control interface to hide the *infrastructure diversity* underneath.

"A great IDP lets teams build and operate apps the same way, no matter what's running underneath or where."

2. Enterprise Portal

The front door of the IDP

Why a portal matters?

- Facilitates *discoverability* → No more “I know a guy who knows a guy”
- Encourages *reuse* → Reduces rework, accelerates delivery
- Enhances *security & compliance* → Governance at entry points

Without a portal → Zombie APIs, Duplication, Underutilized workloads, Wasted spend, Unmanaged risks

With a portal → Golden Paths for the *entire enterprise*, not just individual teams

3. Domain Driven Design

Organizing the City of your Enterprise

- Align application structure with business processes and rules.
- Independence: Each teams owns a "District".
- Shared infra, golden paths, governance.
- Reduces sprawl, duplication and "shadow IT".

"DDD is a critical principal of an IDP to represent the business. Without it, the IDP just reflects the tech stack"

4. API-First Mindset

The Language of the Enterprise

- Treat everything as an API
  - Services, caches, databases, queues, and even infrastructure endpoints
- Shared enterprise assets
  - Not private hidden backdoors. Must be discoverable, governed, reused.
- Shelf-life and Lifecycle Management
  - APIs need versioning, naming standards and depreciation policies.
- Quality enablers
  - Monitoring, observability and proper SDLC built- in

5. Security-First Mindset

Built-in from day one, not Bolted-On later

- Security must be intrinsic
  - Built-in from day one, not bolted-on later
- Every asset needs the right guardrails
  - Services → AuthN, AuthZ
  - Workloads → RBAC, Zero-Trust, Policy-as-Code
  - Data → Encryption at rest and transit
- Golden Paths = Secure by Default
  - Developers inherit secure templates without effort
- Benefits
  - Prevents breachers, security "for free", builds trust

6. Universal Interface

One Dashboard, not Many

- Most IDPs are a patchwork of tools, with many dashboards and many different abstractions.
- Developer and operators waste time hopping between UIs.
- Errors are hard to trace → Do you even know which tool has the answer?
- A good IDP has one single interface, a single set of abstractions, wired across all tools and interfaces.
- Just like an OBD tool → One socket, one screen, full visibility

7. Self-Service

For Developers and Operators

- For Developers
  - No more tickets → Deploy and test directly
  - Golden paths for onboarding workloads
  - Built-in access to logs, metrics, traces
  - Agentic assistance (AI-driven, bots, copilots)
- For Operators
  - Enterprise abstractions to remove repetitive tasks
  - Guardrails for IaC, sensitive data and risky actions
  - Audit trails and compliance checks baked-in
  - Easier governance and scale without manual toil

8. Ops driven, Declarative and Automated

Platform Resilience and Trustworthiness

- Automation is essential
  - Reduces manual toil and errors, scales operations
- Ops driven practices
  - Versioned, auditable, reversible → Builds trust
  - Aligns with Gitops style workflows
- Declarative state management
  - Devs & operators declare what they want, not how
  - IDP reconciles the system to that state
- Benefits
  - Reliable, repeatable, reversible operations
  - Faster recovery and safer experimentation
  - Builds confidence for devs & operators

9. Intelligent and Insightful

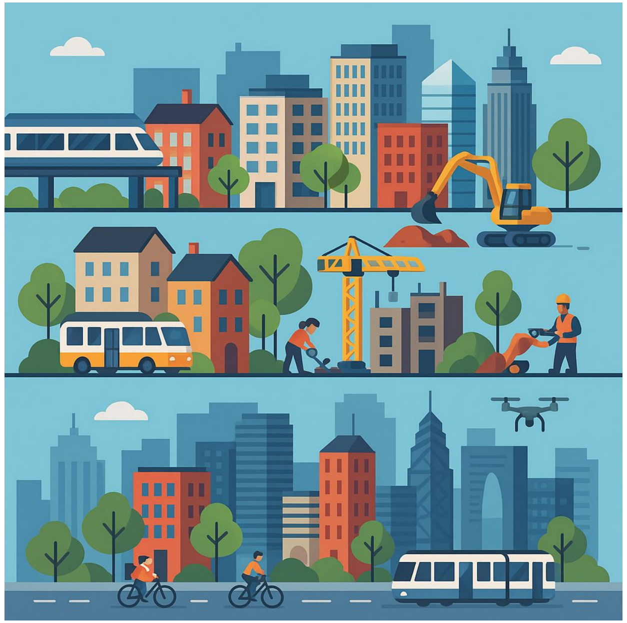
Actionable intelligence for everyone

- Insights for everyone
  - Developers/Operators → Troubleshoot & resolve issues fast
  - Business → Visibility into value of digital assets
  - Engineering → Delivery efficiency and throughput
  - Architects → ROI of assets (value vs spend)
- Intelligence across the board
  - Intelligent insights, not raw reports
  - Operational efficiency (smart scale, anomalies)
  - Engineering quality and consistency
  - Compliance and safety (policy intelligence, risk alerts)

10. Treated as a Product

Reinforces ownership, accountability and continuous improvement

- Roadmap driven
  - Clear vision and evolution plan, not ad-hoc changes
- Regular release cadence
  - New features, fixes, improvements delivered continuously
- Feature lifecycle management
  - Versioned, monitored and deprecated when outdated
- Focus on adoption and effectiveness
  - Measure DevX, usage and business value
  - Increase awareness and enablement
- Continuous adaptation
  - VMs → Containers → Serverless → AI workflows



10 Key principles of a good Internal Developer Platform

1. Separation of concerns
    a. Developers vs Operators
    b. Control Plane vs Data Plane
2. Enterprise Portal
3. Domain Driven Design
4. API-First Mindset
5. Security-First Mindset
6. Universal Interface
7. Self-Service
    a. For Developers
    b. For Operators
8. Ops driven, Declarative and Automated
9. Intelligent and Insightful
10. Treated as a Product

Question Time!

Thank you!

20TH ANNIVERSARY EDITION
WSO2CON ASIA
PLATFORMLESS MODERNIZATION