WSO2CON 2024

Software Delivery in Hybrid (Cloud) Environments



Binura Gunasekara
Associate Director/Architect
WSO2



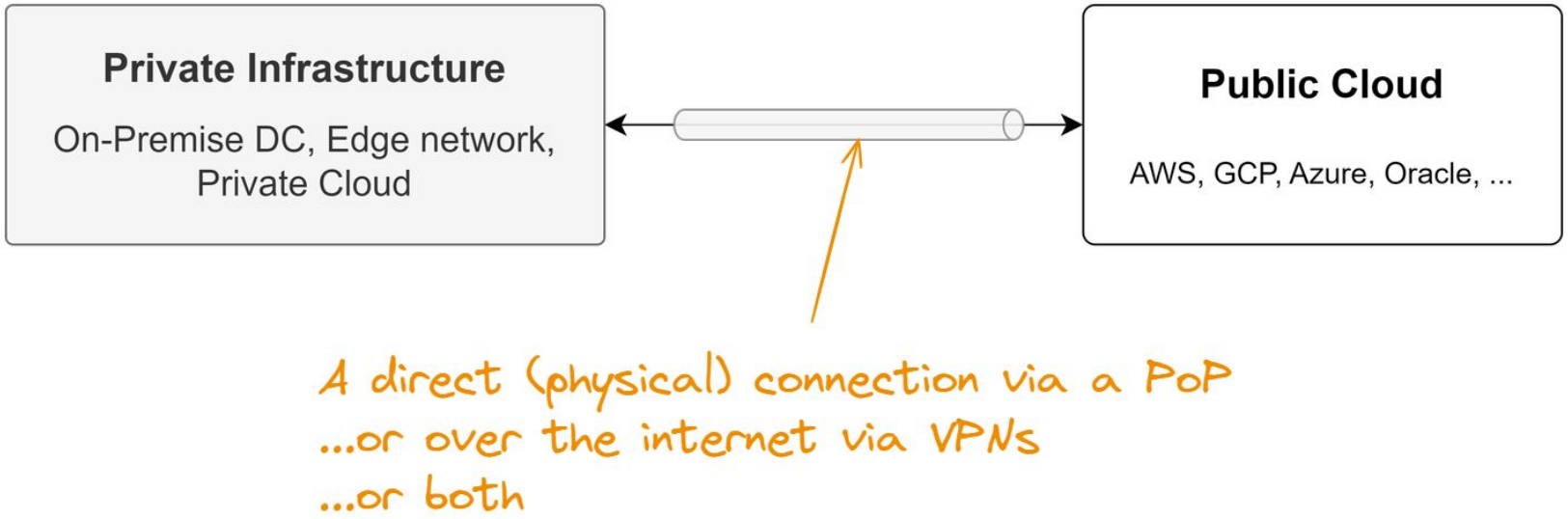
There are two kinds of people:
those who think Hybrid Clouds are a fantastic idea, and those who have had to run them in production.

– Me, *circa today*

Recap

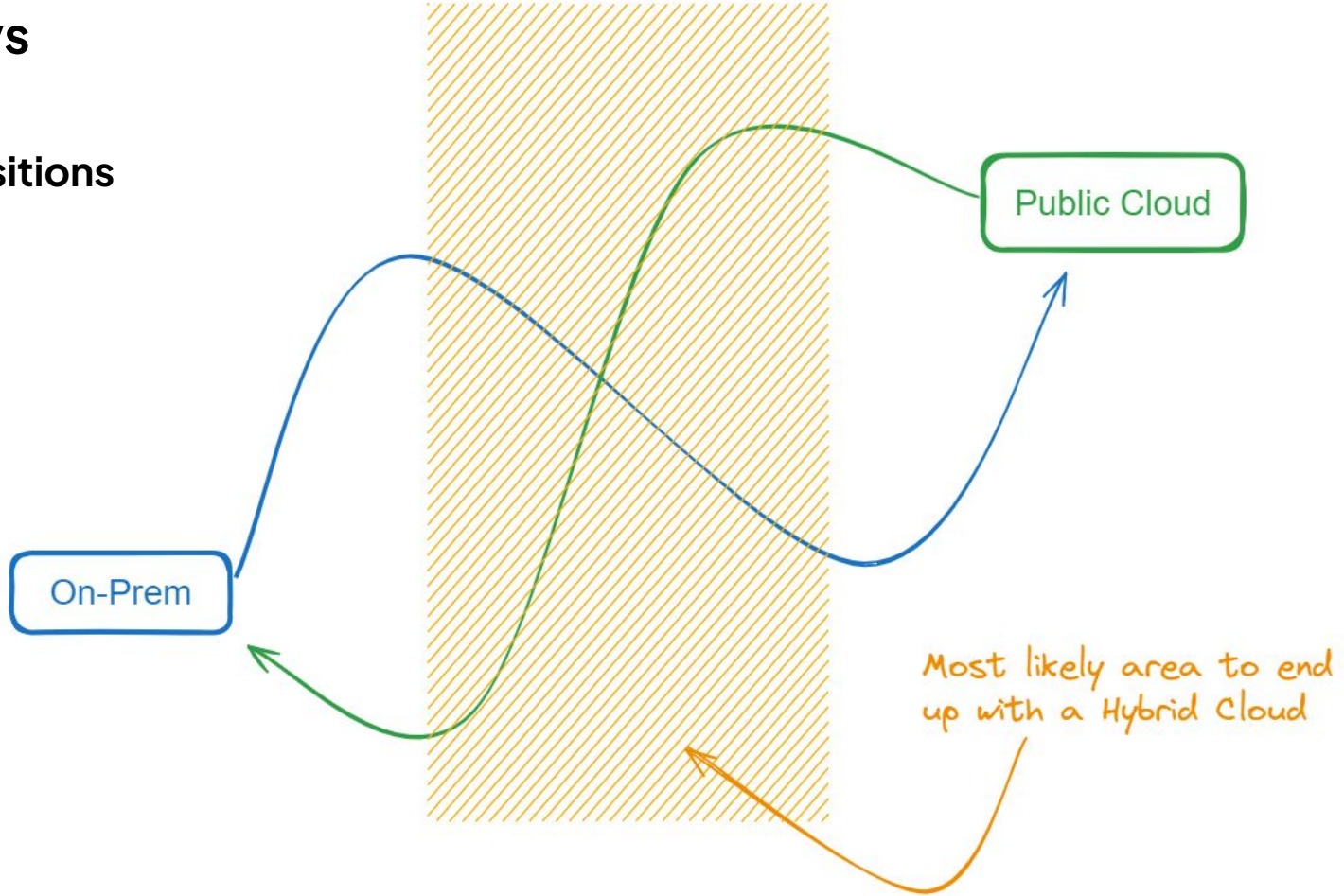
Standard issue definition:

Private infrastructure ↔ Public Cloud



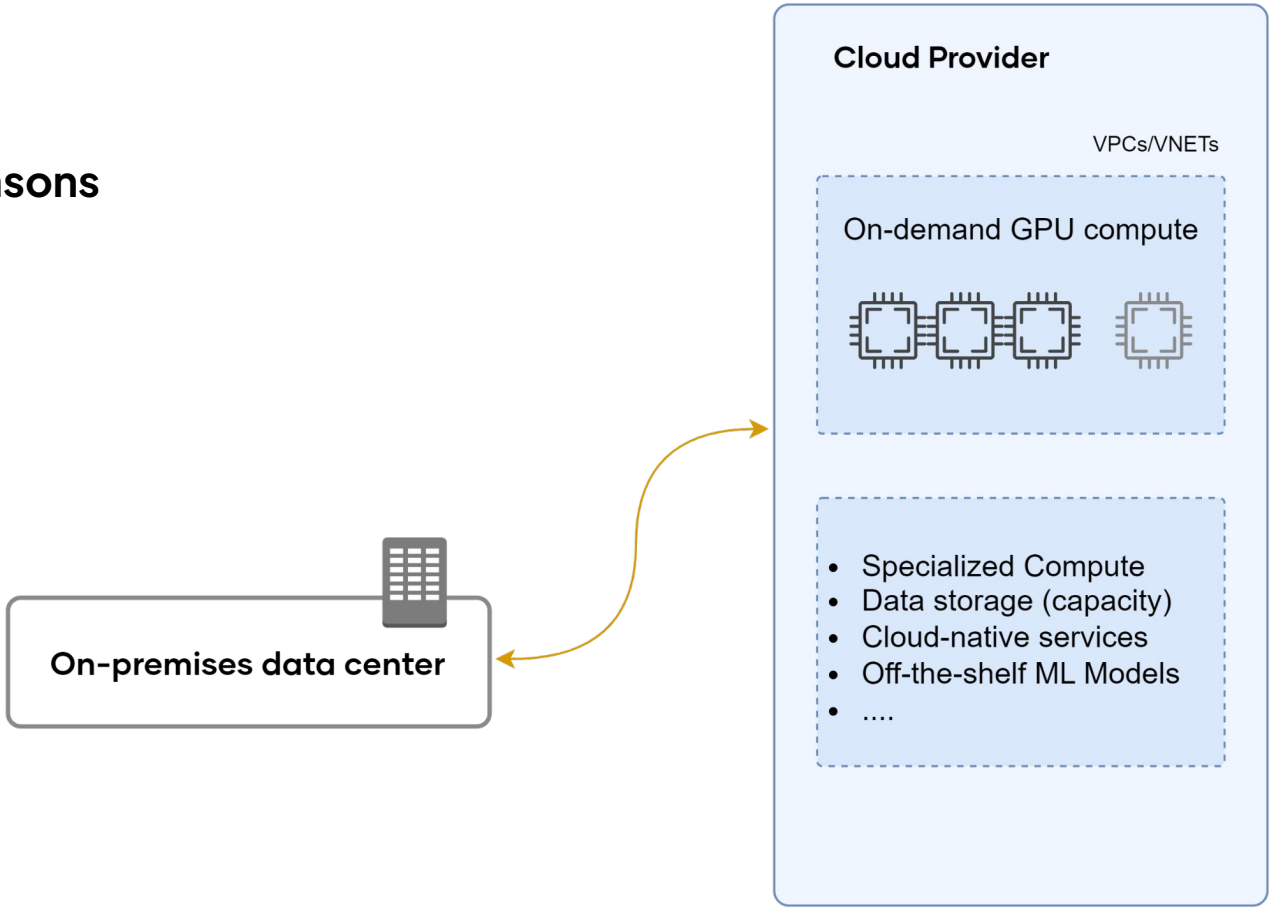
The Whys

1. **Transitions**



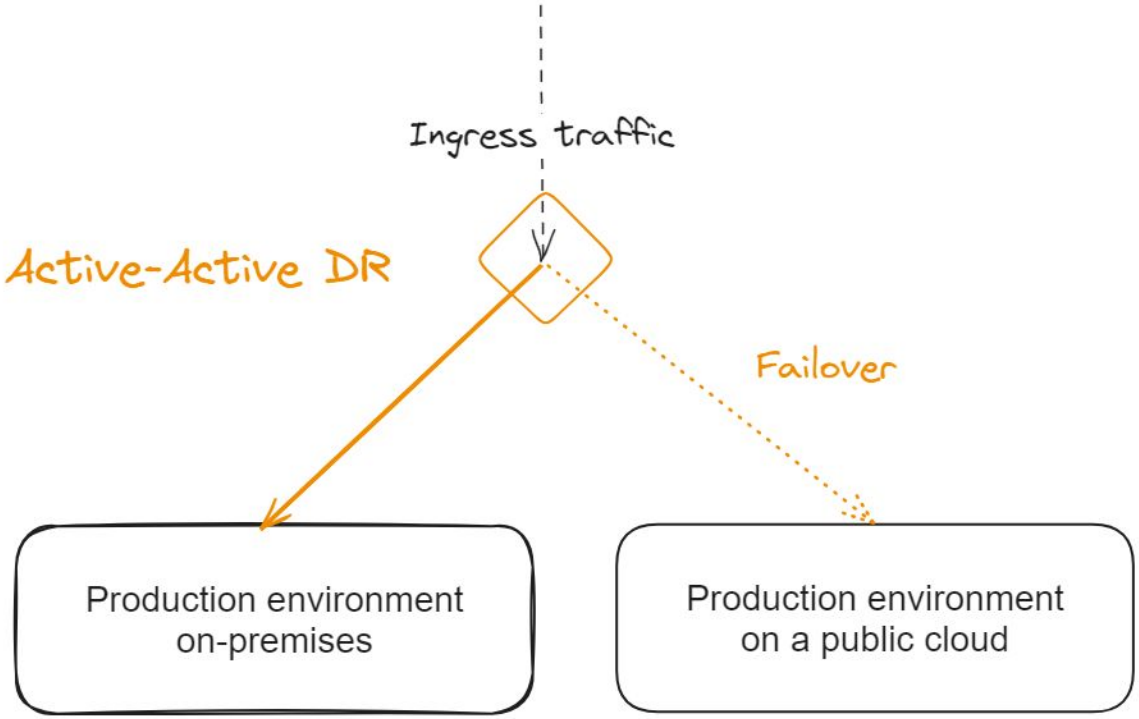
The Whys

2. Functional reasons



The Whys

3. Business continuity



Failure Points

1. Design rot
2. Chasing *(the illusion of)* portability
3. Developer ~~experience~~ adversity
    a. Tooling chaos
    b. Platform overhead

Design Rot

- “No one knows what that does”
- “No one knows why it works”
- “No one knows where that (network) goes”



“Where should this new thing go?”

“Not sure… chuck it wherever”.

“Portability” might only be skin-deep

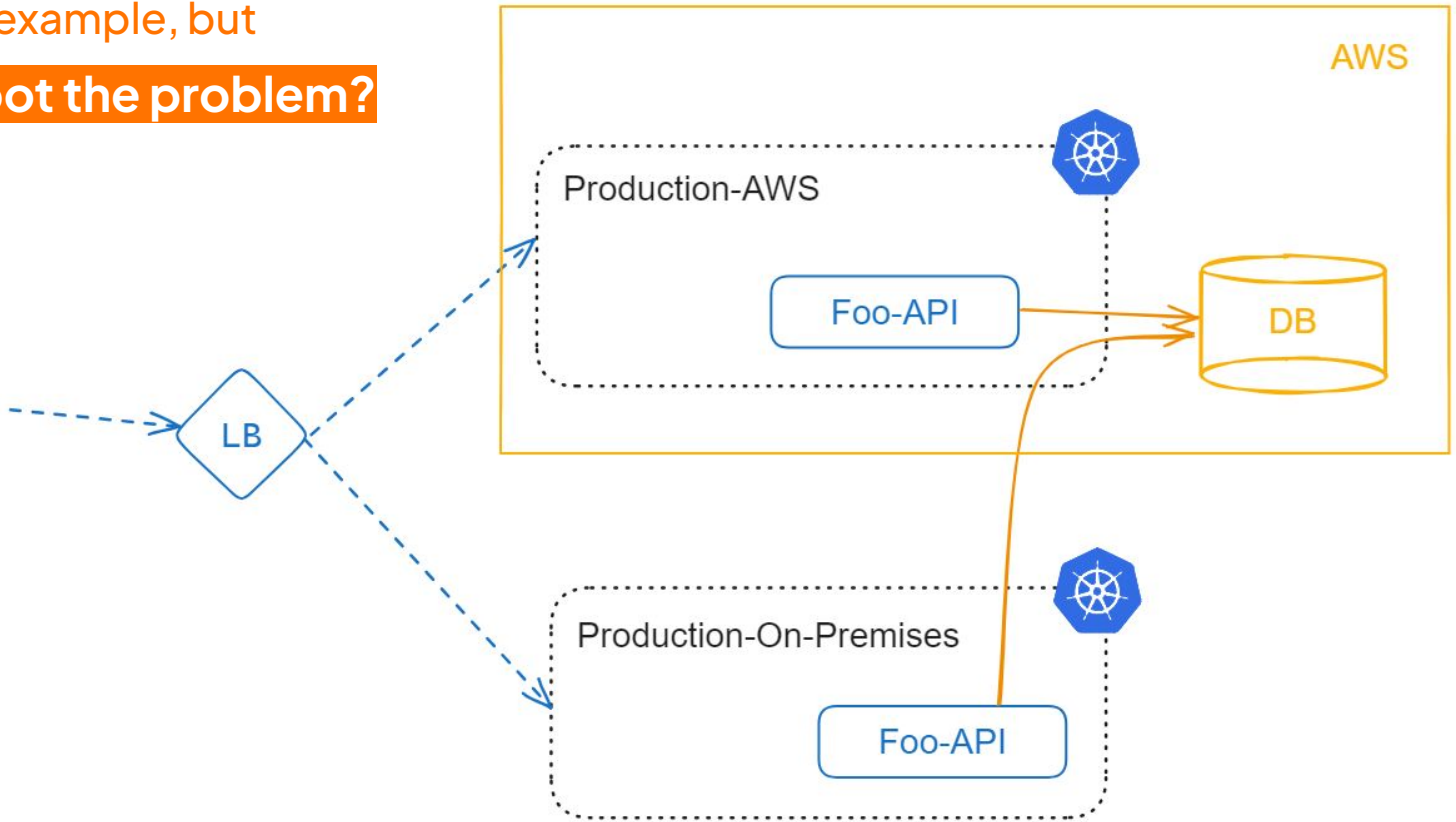
- It’s (relatively) **easy** to move containerized applications around.

- Applications front things that are **not-so-easy** to move around.
  - Or moving them around could be **costly**.
  - *Eg.* Databases, Queues, …so on and so forth.

- Replicated architectures… on hybrid clouds… for high availability and/or portability – *Is your stack really portable?*

A simplified example, but

can you spot the problem?



Developer Adversity



Fancy CI/CD for the cloud, SSH and cp (copy) to that on-prem box.

- Observability? *What observability?*
  - Guesswork
  - Frustration

- Processes and tools designed to fight you; not support you.

- Too many tools. Too little time to get actual work done.

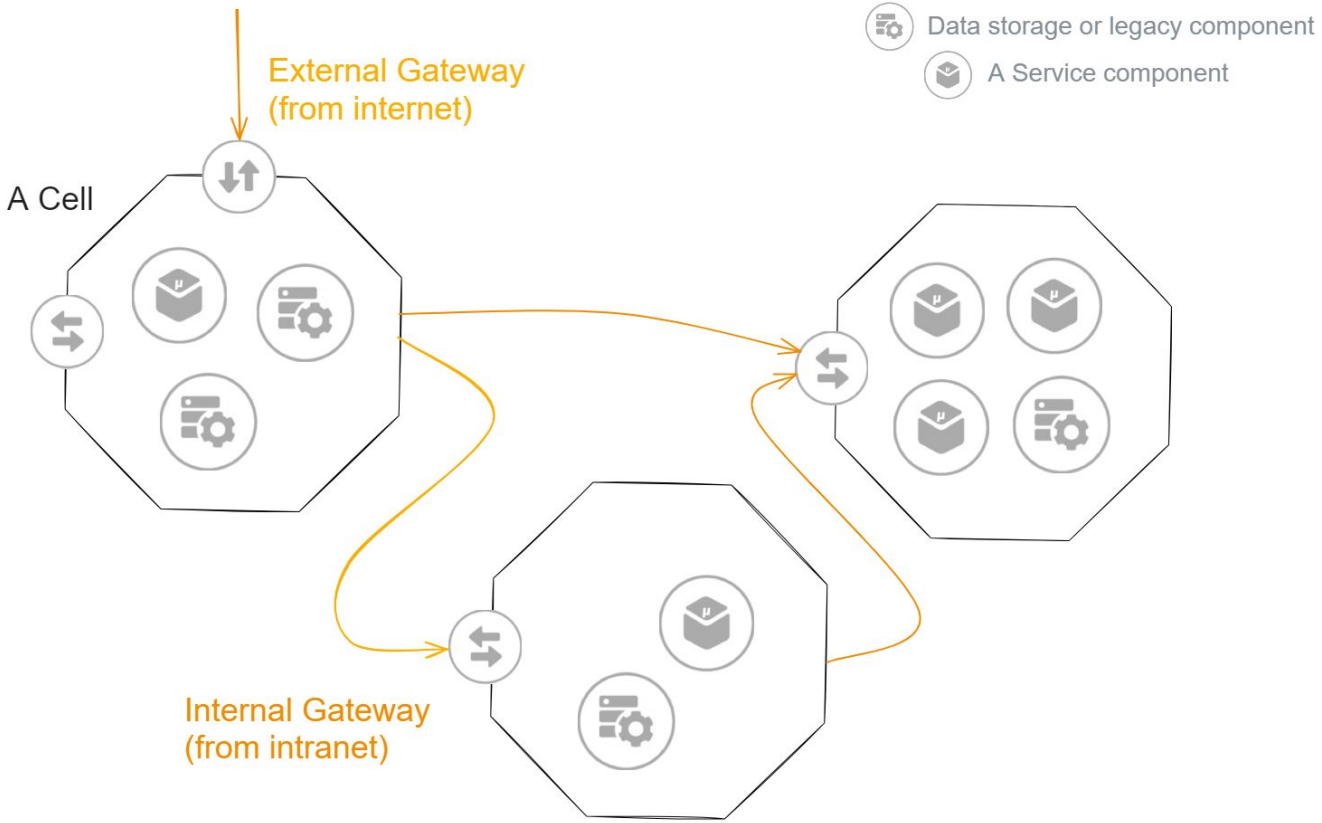
Getting it right

Avoiding design rot.

- Design to optimize the benefits of a Hybrid Cloud.
  - You can also design to make your life miserable.

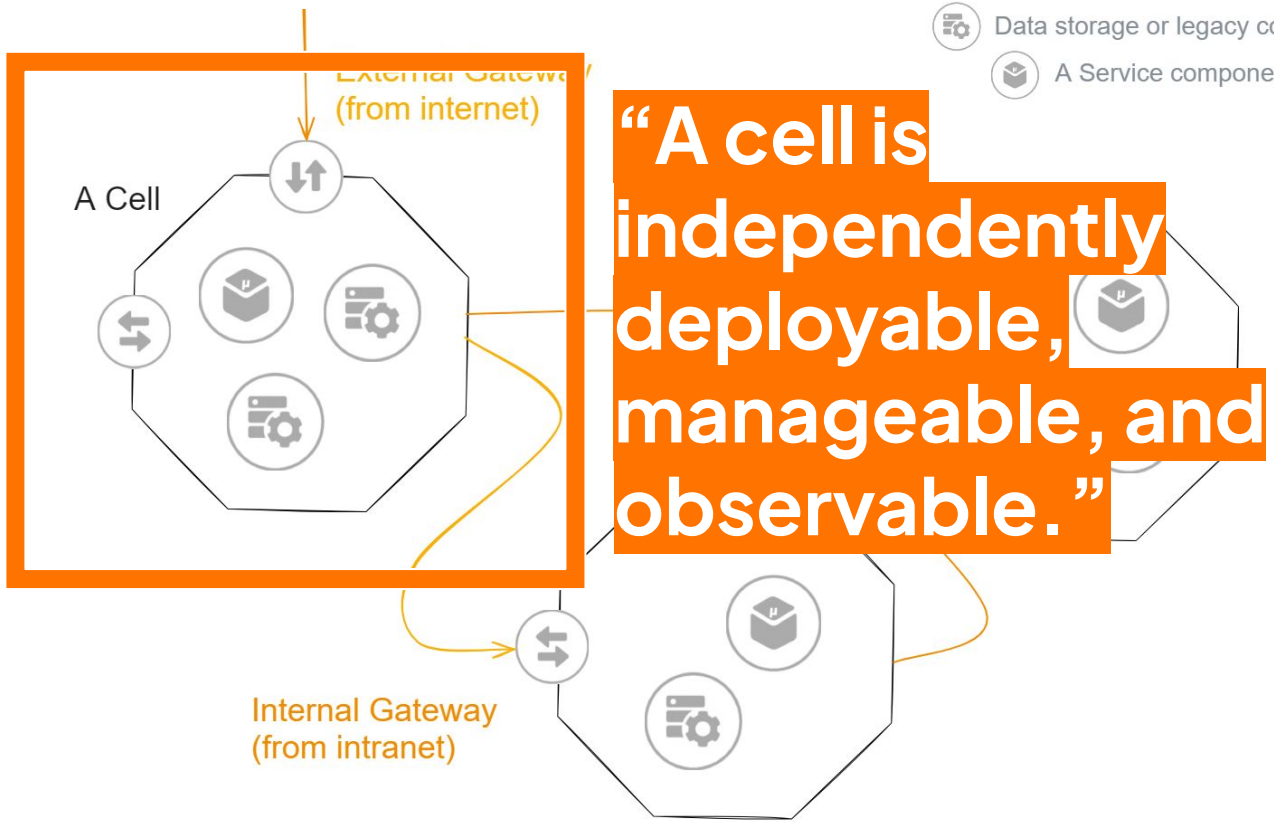
- Prefer domain-driven partitioning over replication *(if possible)*
- ***Cell-based architecture** gives you a scalable framework to build, reason with and visualize your enterprise architecture.*
  - A framework to model your enterprise architecture into self-contained implementation units.
  - Controlled ingress and egress points:
    - Observability
    - Access controls
    - …towards zero-trust

Cell-based architecture



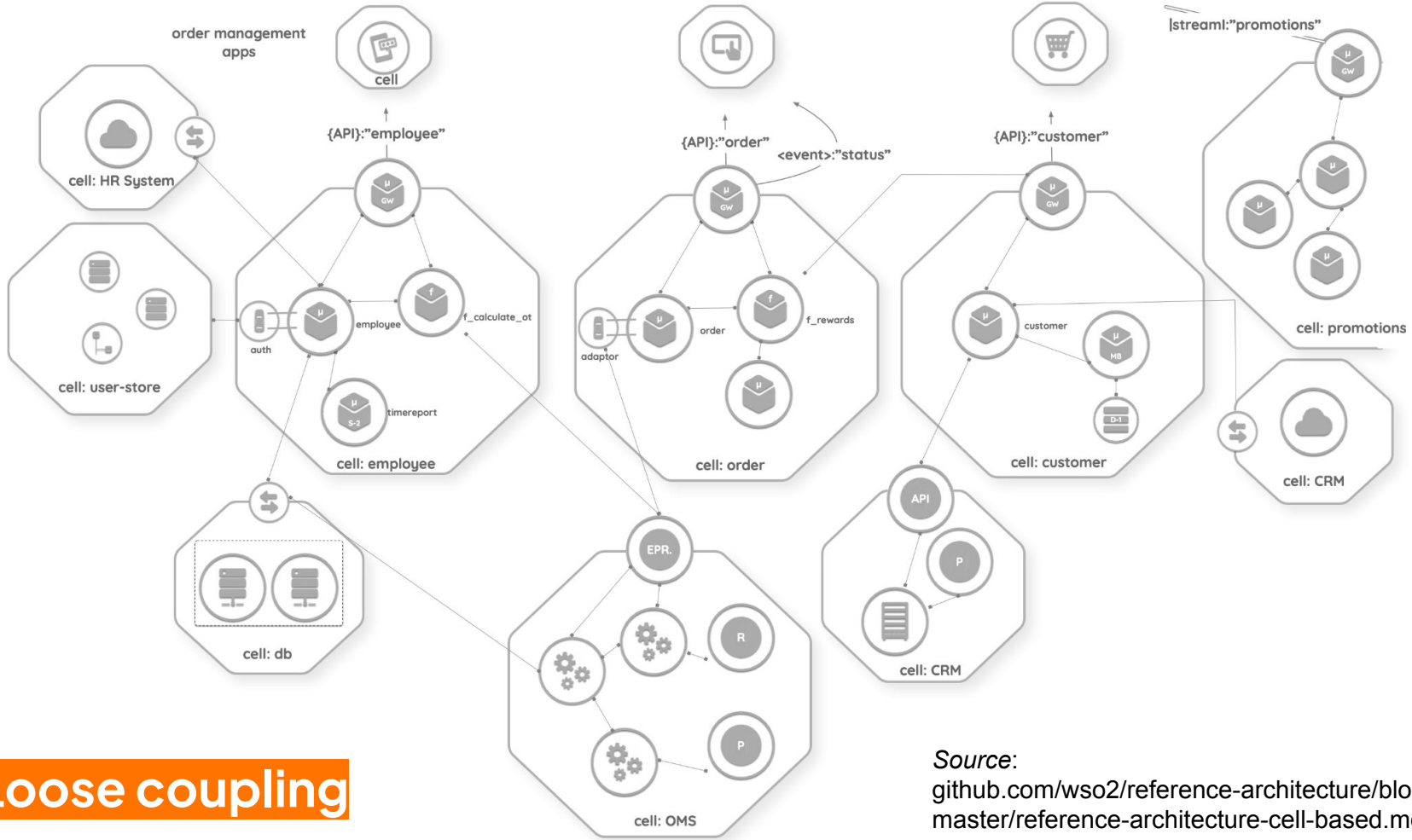
https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

Cell-based architecture



“A cell is independently deployable, manageable, and observable.”

https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md



Loose coupling

Source:
github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

Portable units of deployment

- You now have self-contained, portable* units of business/functional capability – loose coupling.

- *Portability can still be restricted by technology choices.
  Eg. Being backed by a proprietary database.

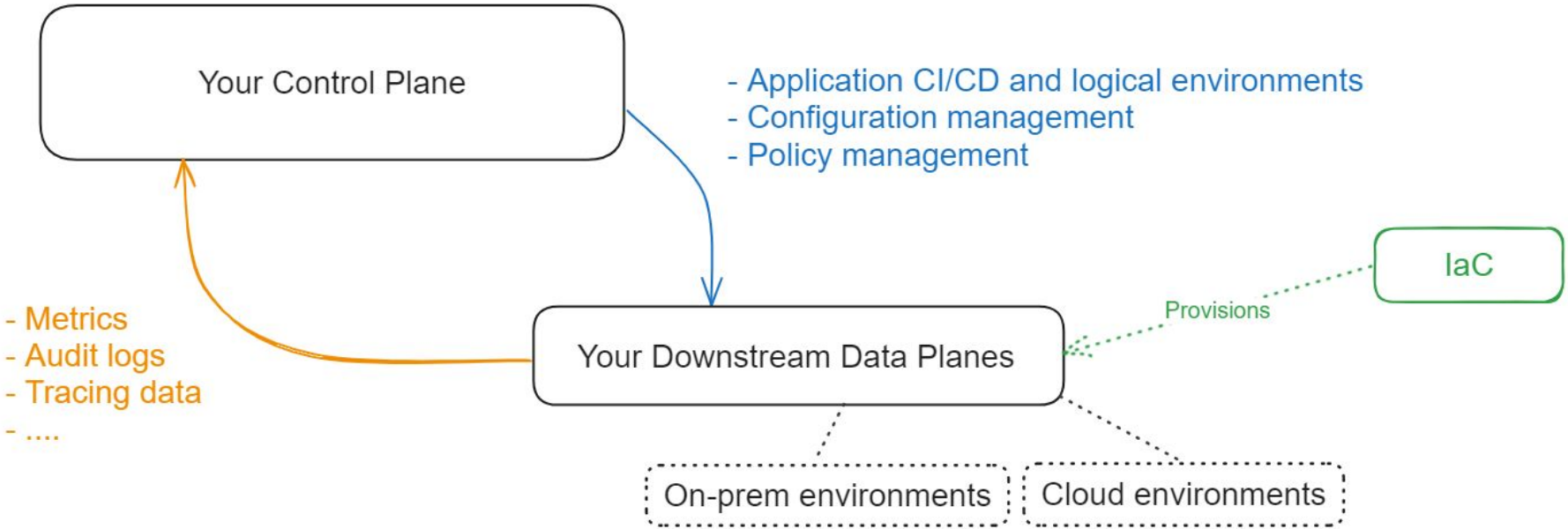
- ***A great read: Google–"Don't get locked up into avoiding lock-in"***
  - *Reference: https://martinfowler.com/articles/oss-lockin.html*

A centralized control plane

- $$\text{Effective governance} = \frac{f(\text{Automation}, \text{Processes})}{\text{Number of control interfaces}}$$

Disclaimer: This is definitely not "scientific" ;)

- Centralize control - distribute capability.
- Be (reasonably) opinionated.
  - Finding the golden path can take a long time.
  - What you think is right today can be wrong tomorrow.
  - This will be a continuous, ever-evolving process.


Your Control Plane
- Application CI/CD and logical environments
- Configuration management
- Policy management
IaC
Provisions
- Metrics
- Audit logs
- Tracing data
- ....
Your Downstream Data Planes
On-prem environments
Cloud environments

Build to enable your developers.

North star: *Become Platformless!*

...How do you do that?

Thank You!