CONTINO

Patterns for Succeeding with Azure in the Enterprise

Hibri Marzook • Principal Consultant

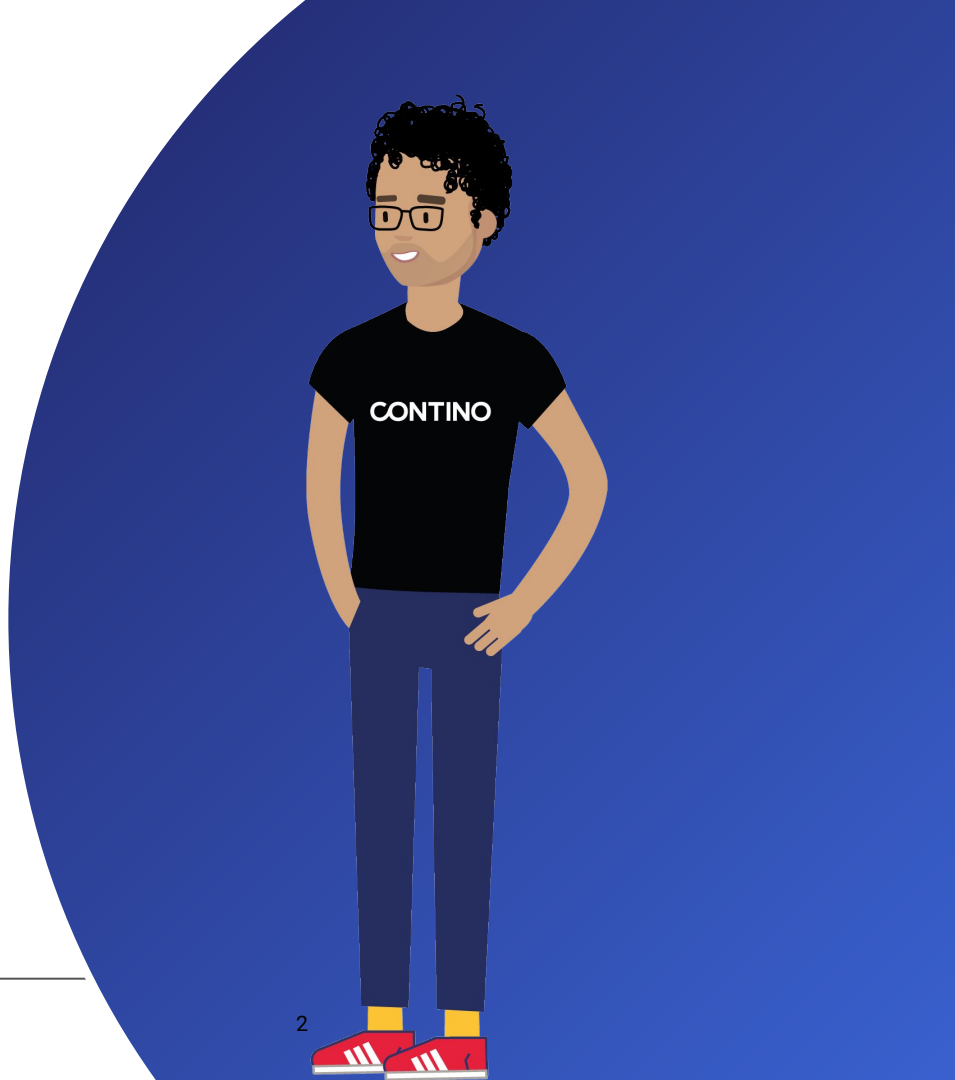
Hibri Marzook

Principal Consultant

Helps teams deliver **ideas and technology**.

Likes the challenge of using **Public Cloud** and **Continuous Delivery** to help teams deliver at a sustainable pace.

@hibri
www.hibri.net



Common Questions

1. How do we know what's going on our Azure estate?
2. Which team takes care of Azure?
3. How do we have control over Azure usage?
4. How do we make it easy for our engineers to use Azure?
5. We are in a regulated industry. Can we trust it?
6. How do I install Azure?

The 5 Essential Capabilities of Cloud Computing

“The highest performing teams were 24 times more likely than low performers to execute on all five capabilities of cloud computing”

Build the Walking Skeleton First

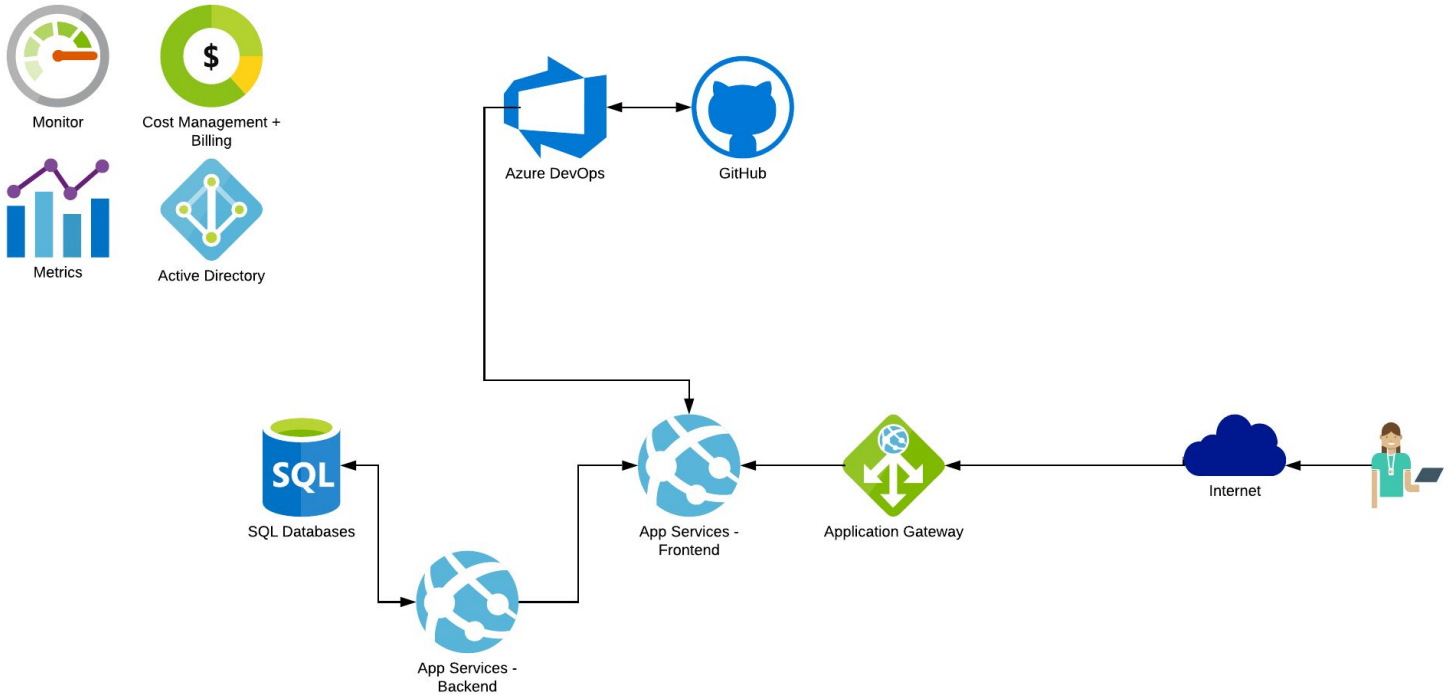
The Walking Skeleton

“A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together the main architectural components. The architecture and the functionality can then evolve in parallel”

Alistair Cockburn

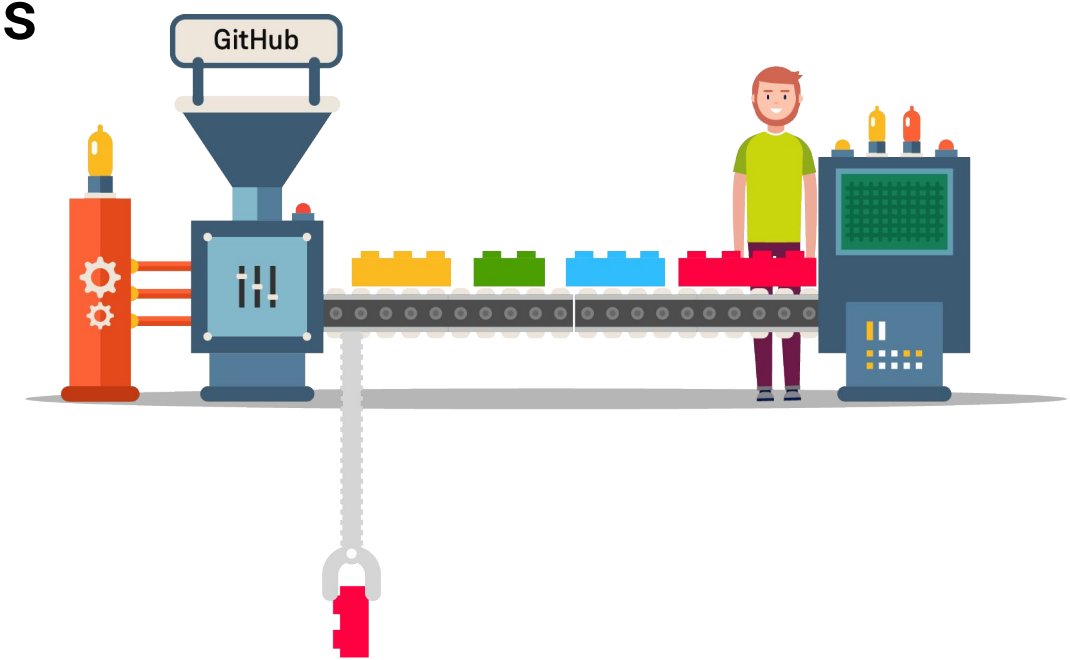
https://www.flickr.com/photos/liverpoolhls/

A Walking Skeleton in Azure



Deploy all the way to a Production environment

Prove that there is a path to your users



Use the Walking Skeleton to validate that you ***can*** use Azure in **your** organisation

Public Cloud is seen as a risk

- Deal with risk **incrementally** rather than in one go
- Flag up heavyweight design reviews and manual approvals **early**

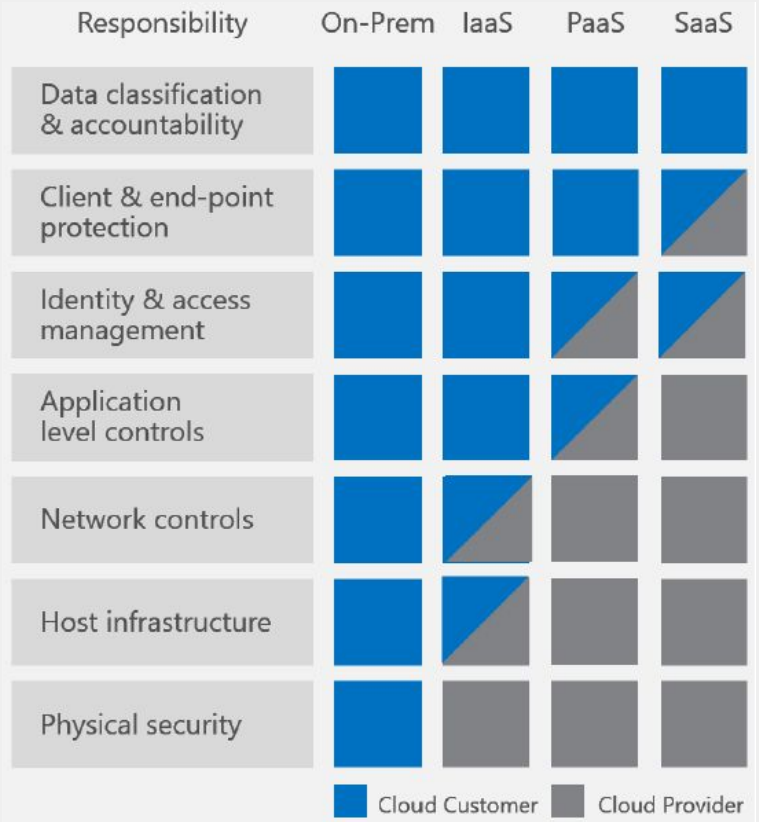
Adopt a PaaS first approach

Large Enterprises tend to **fear things they can't control** and **default to IaaS** in the cloud

Lift and Shift Is Not a Sustainable Strategy


SQL

Understand the **Azure Shared Responsibility** Model



https://blogs.msdn.microsoft.com/azuresecurity/2016/04/18/what-does-shared-responsibility-in-the-cloud-mean/

Adopt Evolutionary Architecture Practices

Evolutionary Architecture

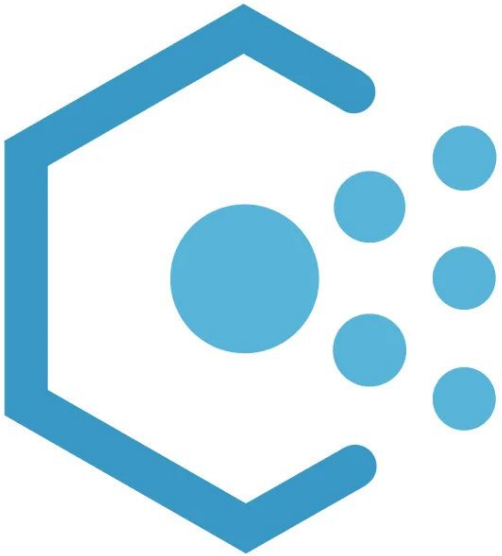
An evolutionary architecture supports ***guided***, **incremental** change across multiple dimensions

Evolutionary Architecture needs Fitness Functions

An architectural ***fitness function*** provides an objective integrity assessment of some architectural characteristic(s)

Ford N, Parsons R, and Kua P (2017) - Building Evolutionary Architectures, O'Reilly Press

Implementing Fitness Functions in Azure

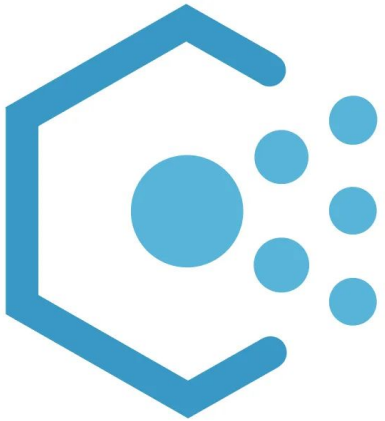


Azure Policy

Azure Monitor

Azure Policy

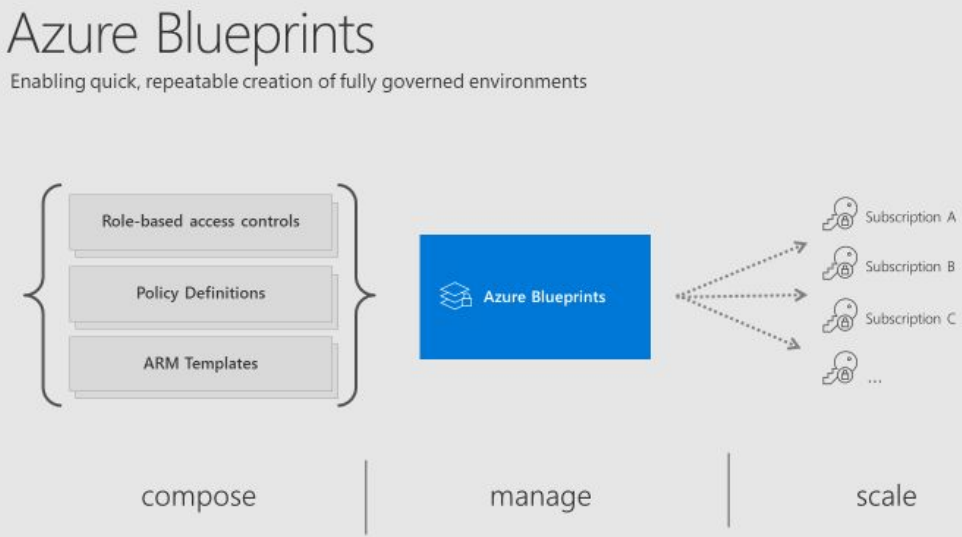
- Allows teams to use Azure resources as long as they don't break the policy
- Don't need to rely only on build time controls
- Allows iteration towards a compliant solution, without intervention from a central authority



Azure Monitor

- Use Log Analytics to query across all resources
- Identify services that are non-performant
- Query resource attributes to find bad patterns

Package it all with Blueprints

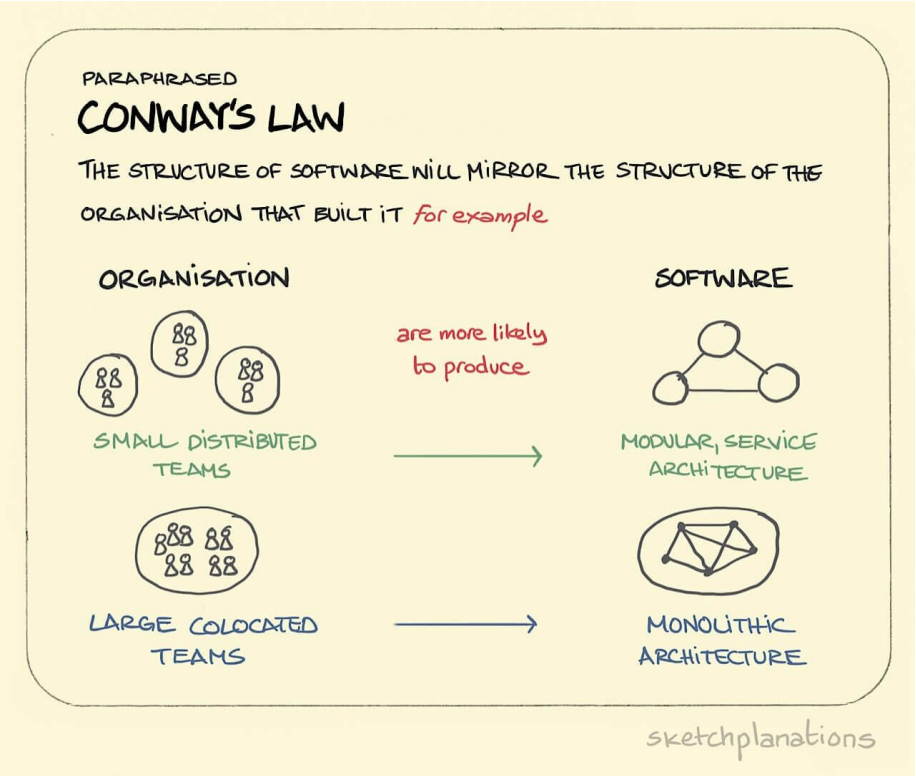


https://cloudblogs.microsoft.com/industry-blog/en-gb/technetuk/2019/04/12/what-are-azure-blueprints/

Be Aware of Conway's Law

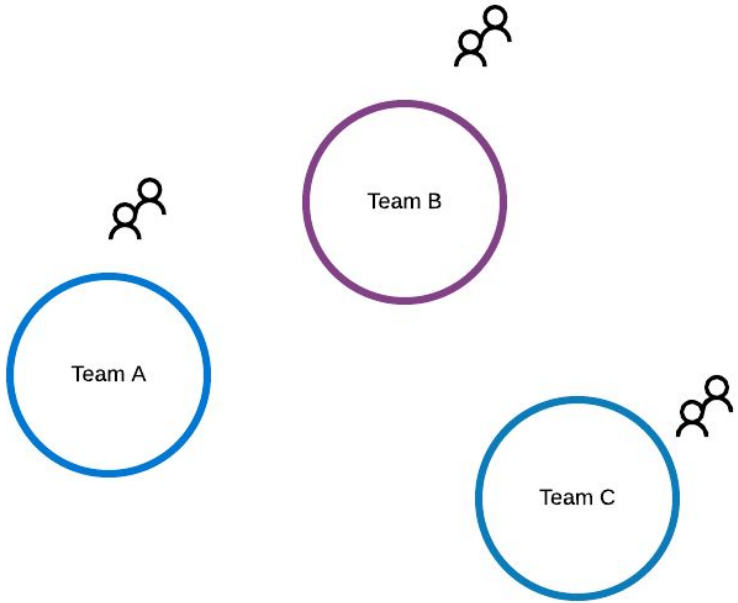
“Organisations which design systems are constrained to produce designs which are copies of the communication structures of these organisations”

Melvin Conway

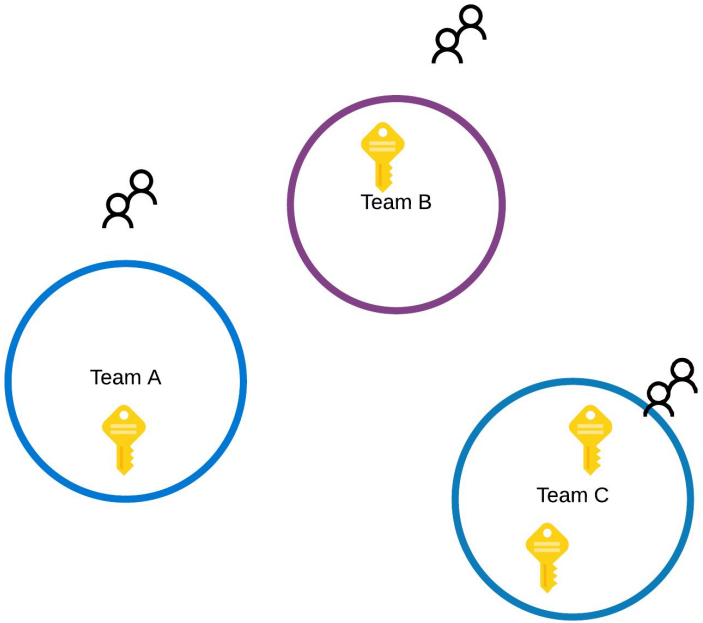


Conway’s Law and Team Autonomy

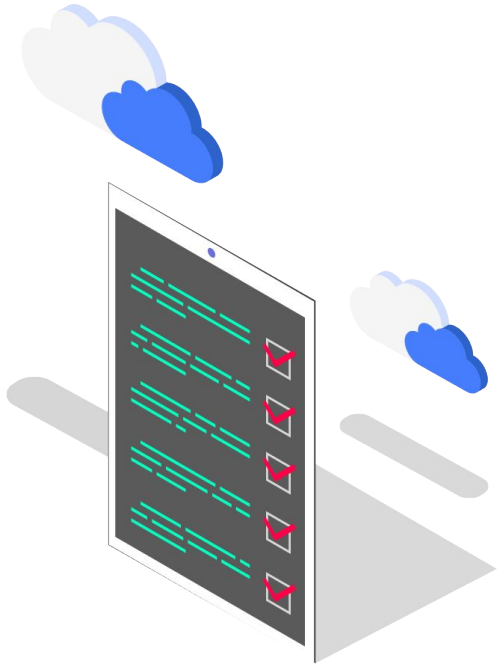
- Self organising teams focussing on customer problems, aligned with the company goals and vision
- Enable teams to have autonomy and responsibility
- Help teams reduce their blast radius
- Don’t copy the org chart to Azure



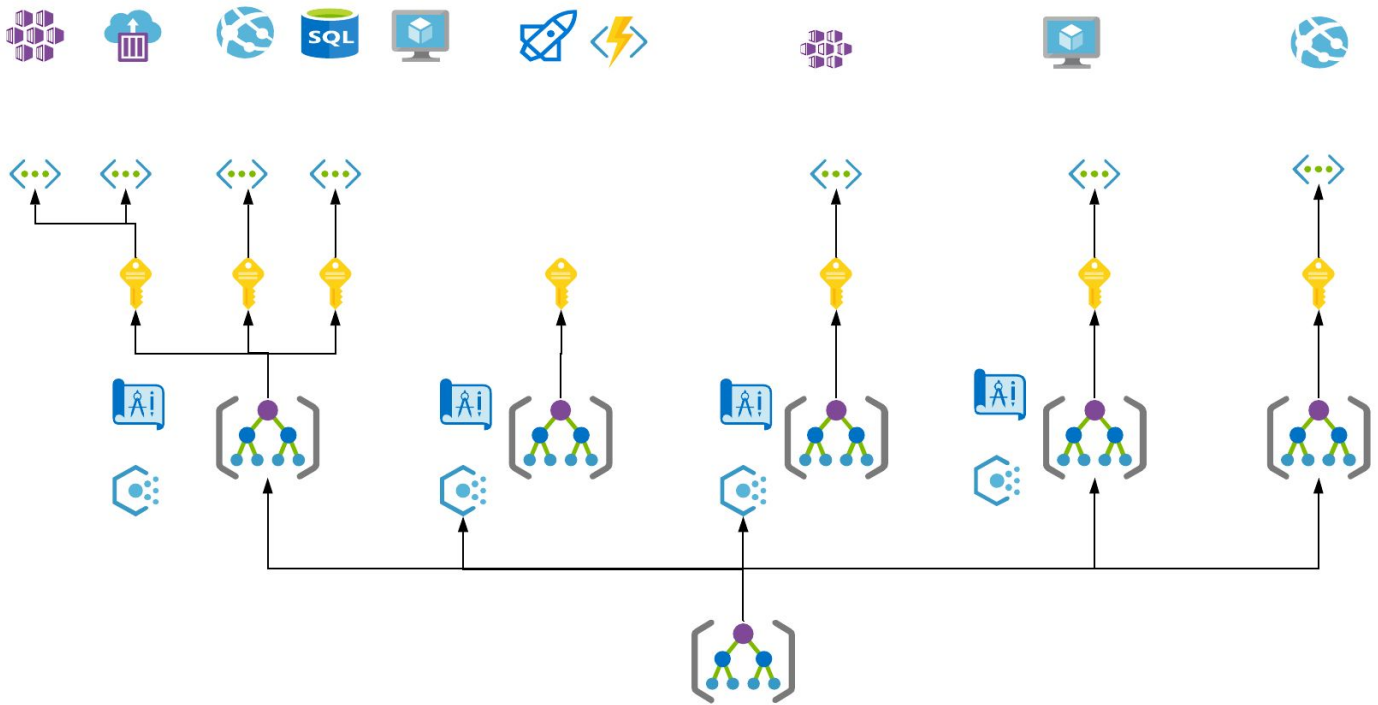
Subscriptions should enable team autonomy and reduce blast radius



Apply governance at the Management Group level

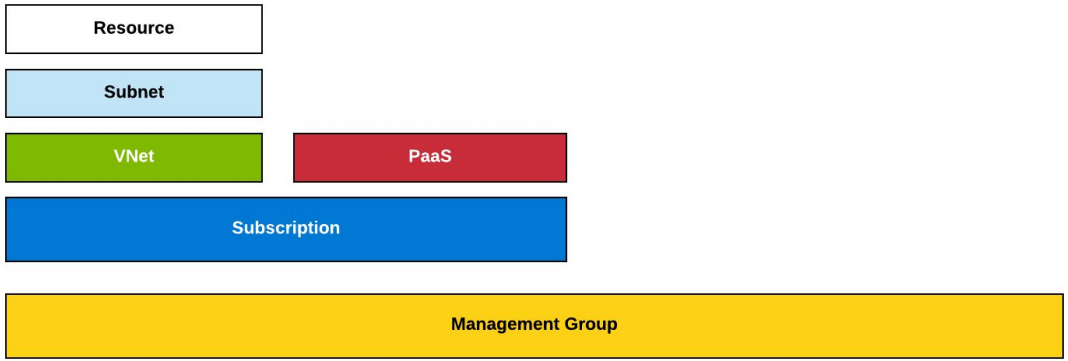


Put Subscriptions into Management Groups



Architect to Enable Team Autonomy

- Architectural layers should allow teams who own each layer to iterate independently
- Layers at the bottom provide a service to layers above



Build in Trust

How to Build in Trust?

- Demonstrate security best practices early to win trust
- Use RBAC
- Access with least privilege
- Don’t use God accounts to make things just work

Use Azure AD Effectively

- Use service principals and managed identities from day one
- Add users to Groups
- Assign Azure AD Groups to roles
- Avoid God accounts for automation

Show that what you are building can be trusted

Build Together

Empathise with product teams and users

Build with a product that has **real users**, and has **real business value**

Continuously validate the platform

Is it **fit for purpose**?

Make the Dev Experience Awesome !

Who Are Your Developers?

- Devs who build on the platform
- Devs who build the platform

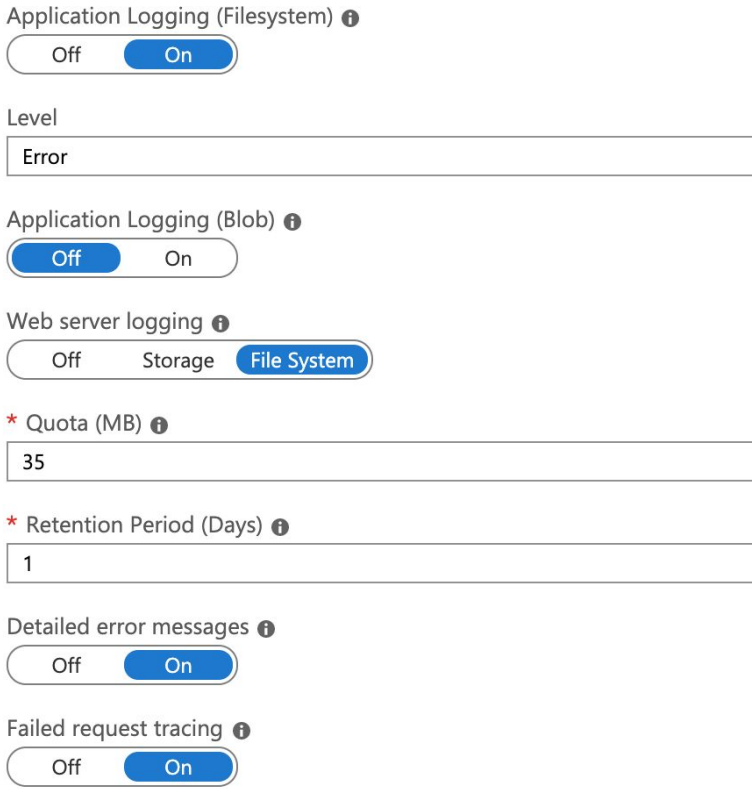
Developers ***will*** find a workaround to help them deliver

Make it easier for devs to do the right thing, safely

Give Developers Visibility

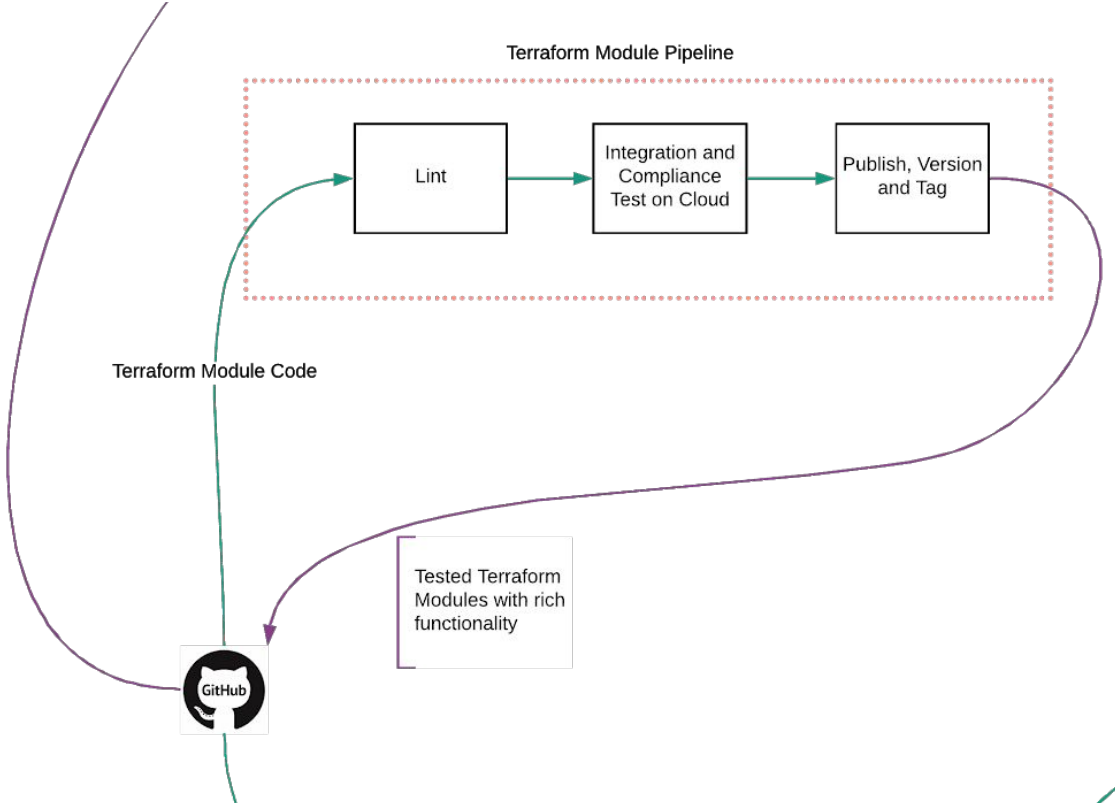
- Don’t lock developers out of the Azure portal
- Provide sandboxes for devs to play around with
- Read only access to production environments
- Access to metrics and alerts
- Allow teams to make their own dashboards

- Turn on diagnostics early
- Show devs the debugging options in Azure, if not you'll have to resolve every issue

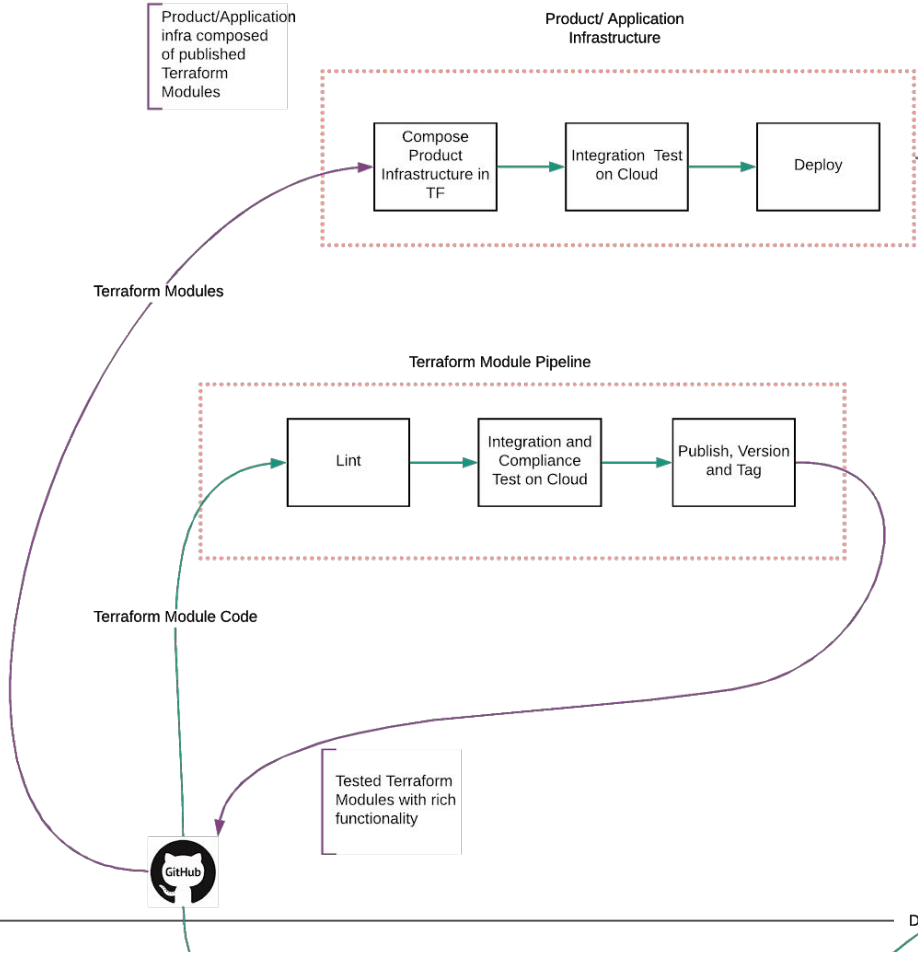


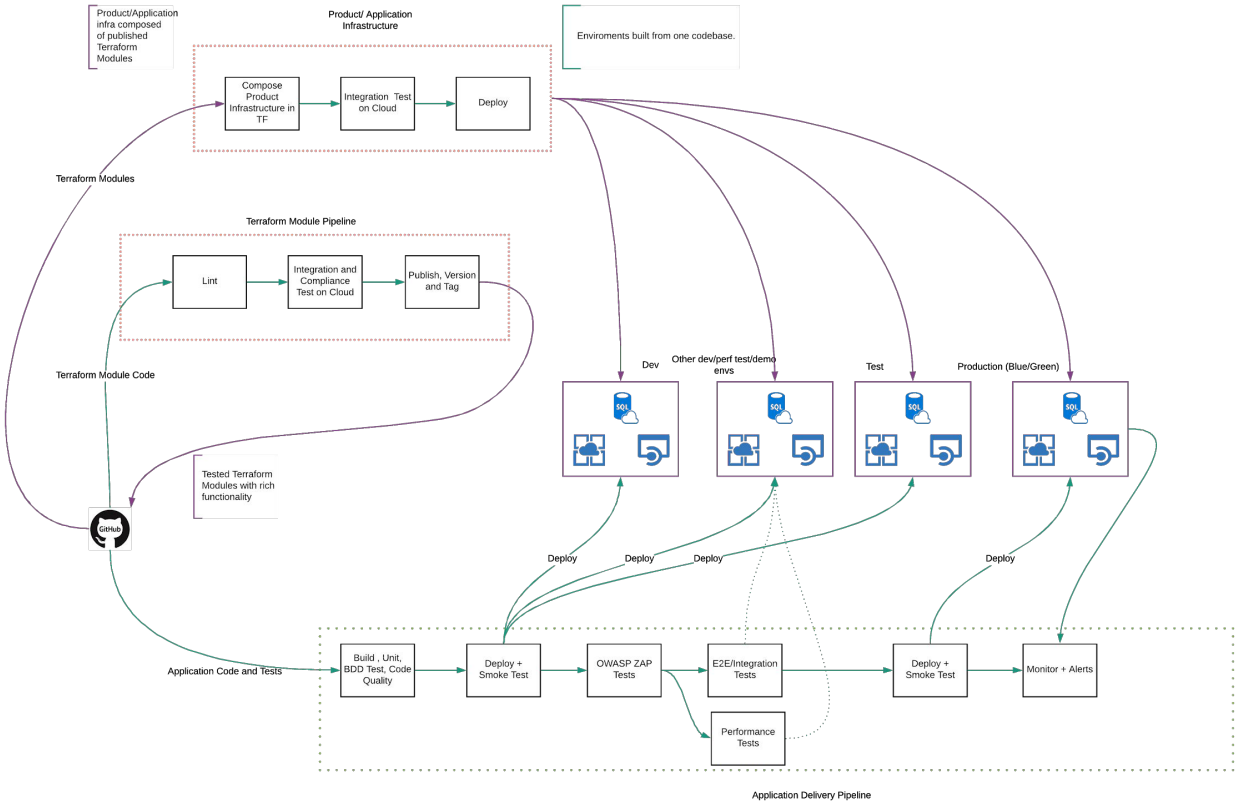
Continuous Delivery From Day One

Use CI for Terraform modules



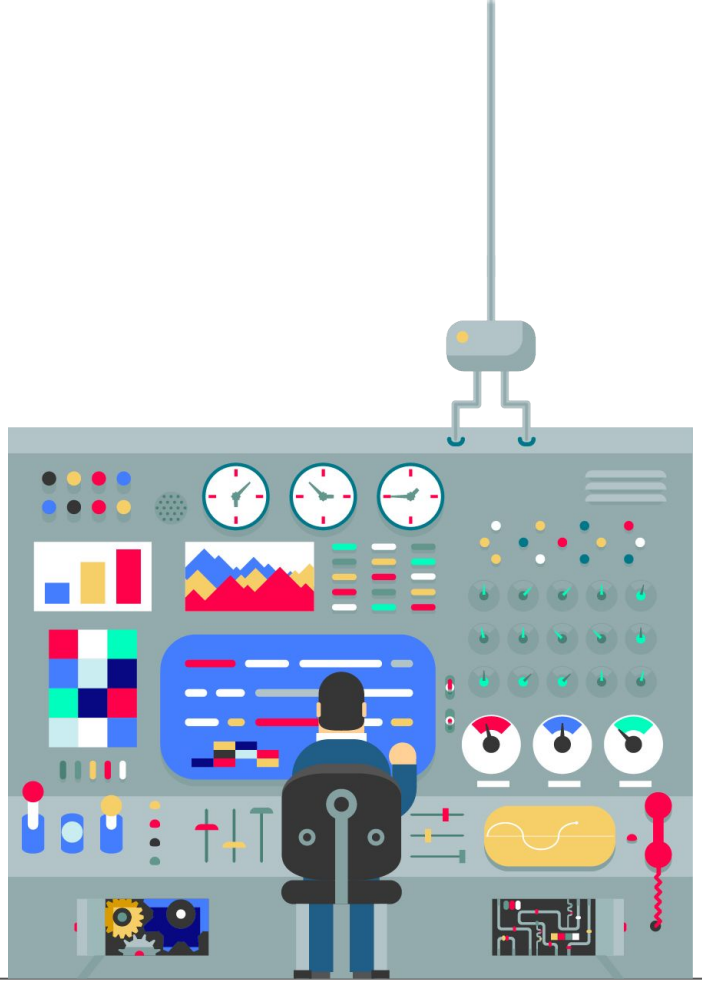
Apply CI to Terraform projects, which consume modules



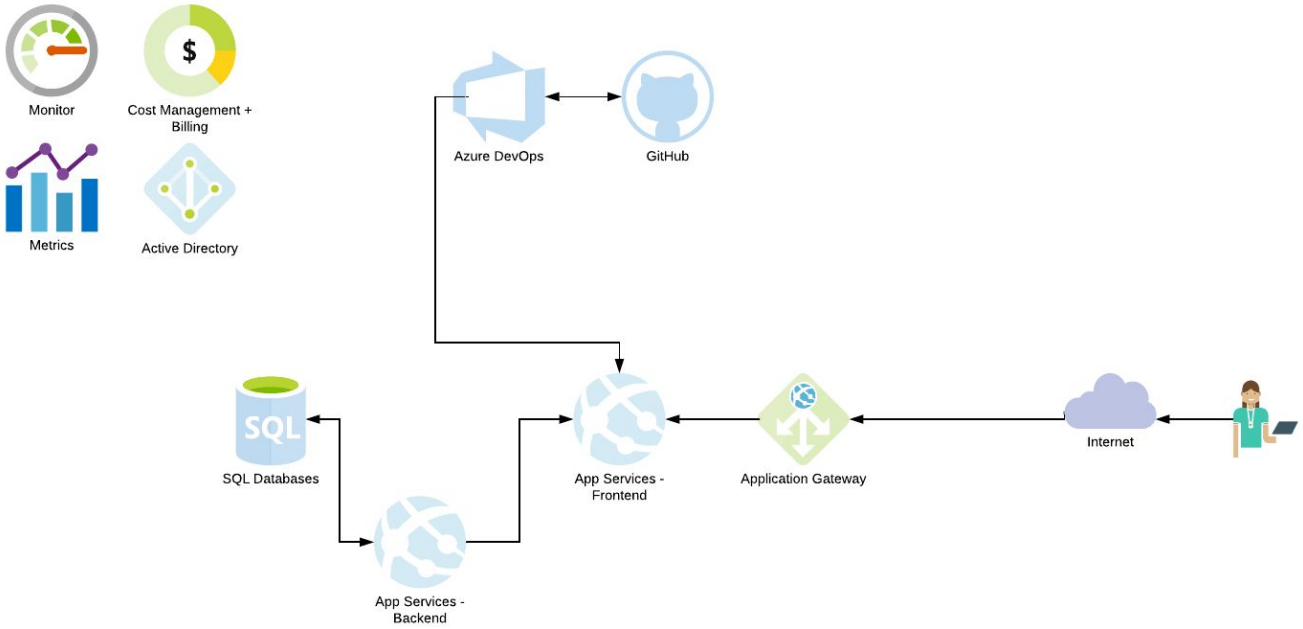

Product/Application infra composed of published Terraform Modules
Product/ Application Infrastructure
Enviroments built from one codebase.
Compose Product Infrastructure in TF
Integration Test on Cloud
Deploy
Terraform Modules
Terraform Module Pipeline
Lint
Integration and Compliance Test on Cloud
Publish, Version and Tag
Terraform Module Code
Tested Terraform Modules with rich functionality
GitHub
Dev
Other dev/perf test/demo envs
Test
Production (Blue/Green)
SQL
Deploy
Deploy
Deploy
Deploy
Application Code and Tests
Build , Unit, BDD Test, Code Quality
Deploy + Smoke Test
OWASP ZAP Tests
E2E/Integration Tests
Performance Tests
Deploy + Smoke Test
Monitor + Alerts
Application Delivery Pipeline

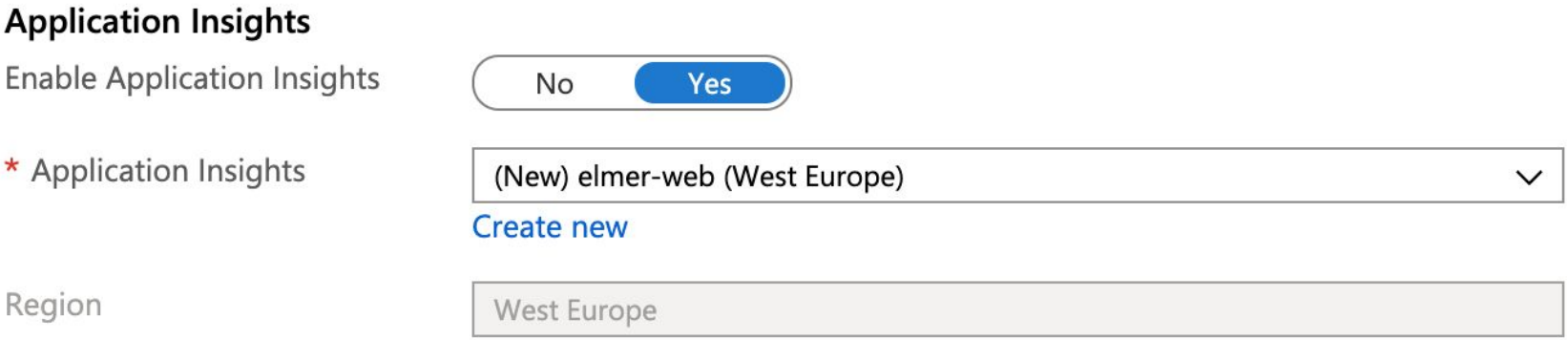
Build in Operability

Think about how your system will behave in production from day one



Add alerts and metrics to the walking skeleton




Application Insights
Enable Application Insights
No
Yes
* Application Insights
(New) elmer-web (West Europe)
Create new
Region
West Europe

Basics Scale Authentication Networking Monitoring Tags Review + create

With Azure Kubernetes Service, you will get CPU and memory usage metrics for each node. In addition, you can enable container monitoring capabilities and get insights into the performance and health of your entire Kubernetes cluster. You will be billed based on the amount of data ingested and your data retention settings.

Learn more about container performance and health monitoring
Learn more about pricing

Azure Monitor

Enable container monitoring: No | Yes

Log Analytics workspace: (new) DefaultWorkspace-63f8f81b-499e-4f02-ac12-d98658ab079c-WEU

Create new

Practice blameless post-mortems even when the platform is in its infancy

Help teams build, run and own their product, from the start

Build an Internal Azure Community of Practice

Public Cloud is a scary world for many people

How to Build an Azure Community of Practice?

1. Show what's been built, **not diagrams**. It's easier to address concerns with a working system
2. Weekly/fortnightly technology and architectural sessions

Don’t Forget Internal Evangelism

1. Do regular demos, show and tells to other teams/departments
2. Do take time to document
3. Do Azure Certs together

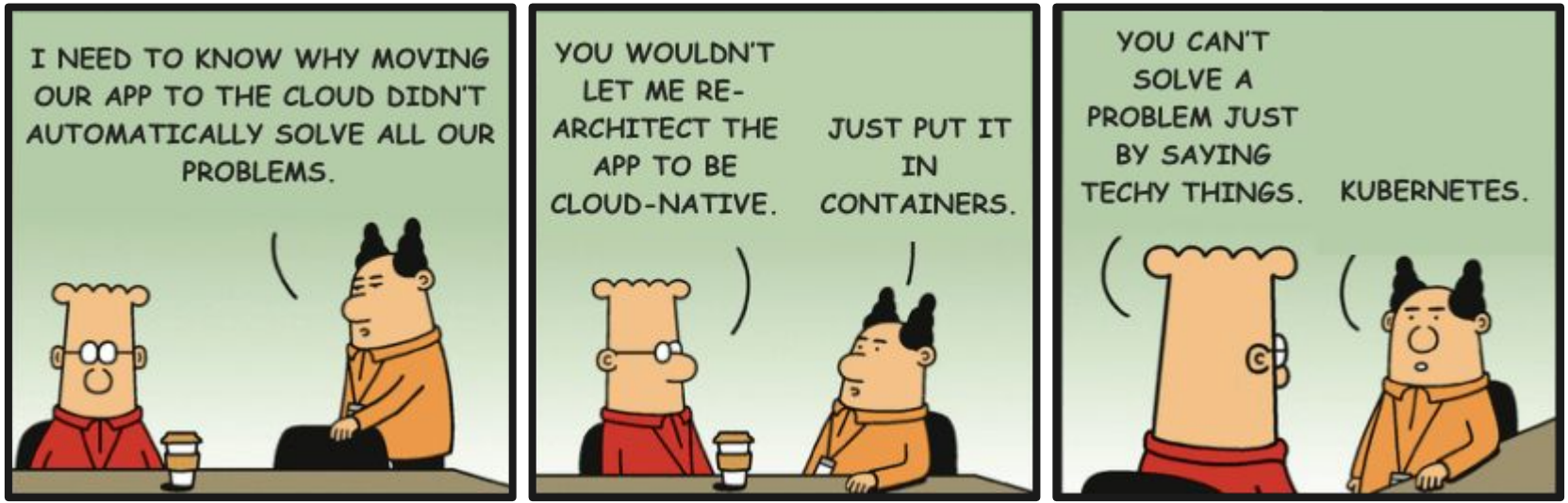
Educate your organisation along with Public Cloud adoption

An Azure Community of Practice will sustain adoption, rather than a technology only focussed initiative

It’s Never Been Easy

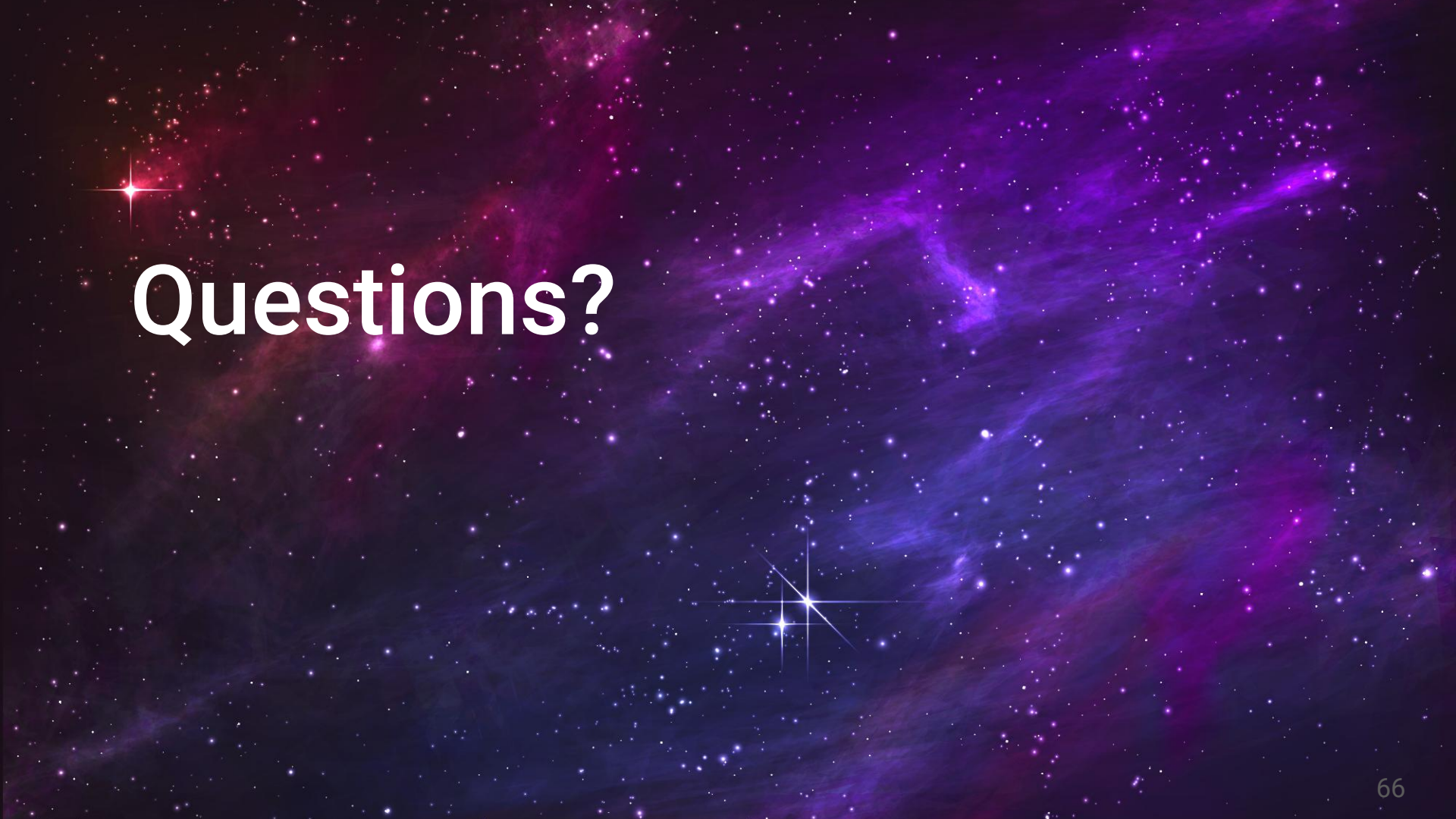
You'll have to slay many dragons

Sustainable Azure adoption is important


I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.

The 10 Patterns

1. Build the Walking Skeleton first
2. Adopt a PaaS first approach
3. Adopt Evolutionary Architecture practices
4. Be aware of Conway's Law
5. Build in trust
6. Build together
7. Make the developer experience awesome
8. Adopt CI/CD
9. Build in Operability
10. Build an internal Azure community of practice


Questions?

CONTINO

Thank You

London
london@contino.io

New York
newyork@contino.io

Melbourne
melbourne@contino.io

Sydney
sydney@contino.io

Atlanta
atlanta@contino.io

www **contino.io** **continohq** in **contino**