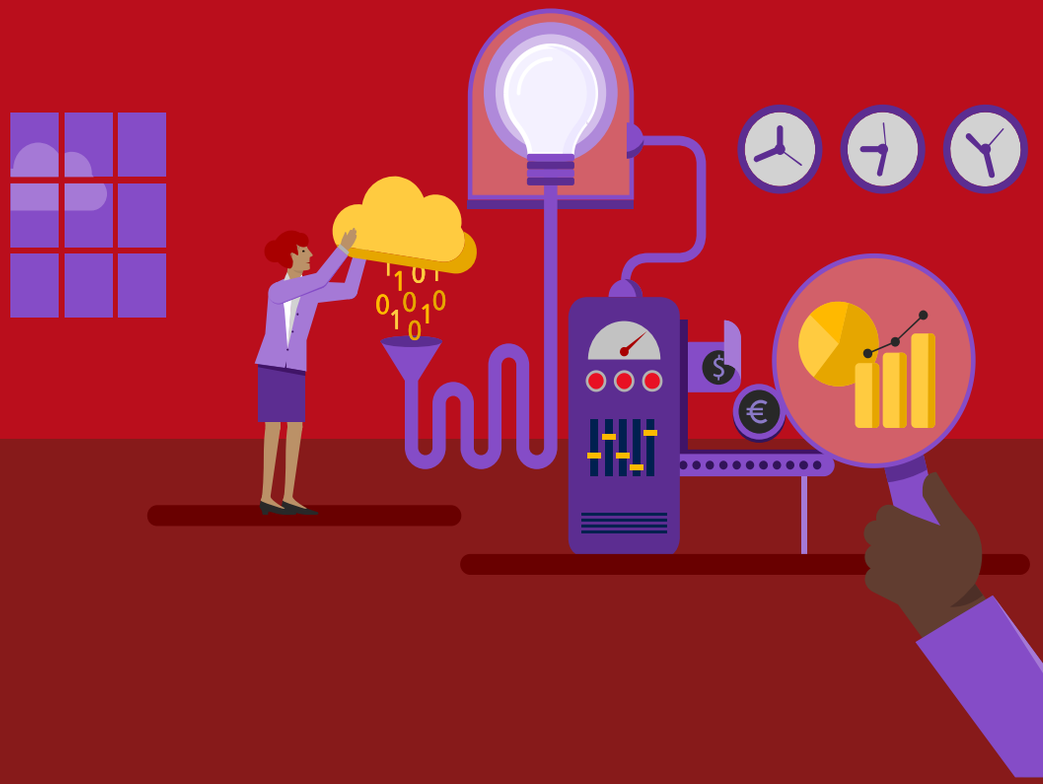
Microsoft Azure Workshop - Understanding cost modelling and sizing cloud solutions

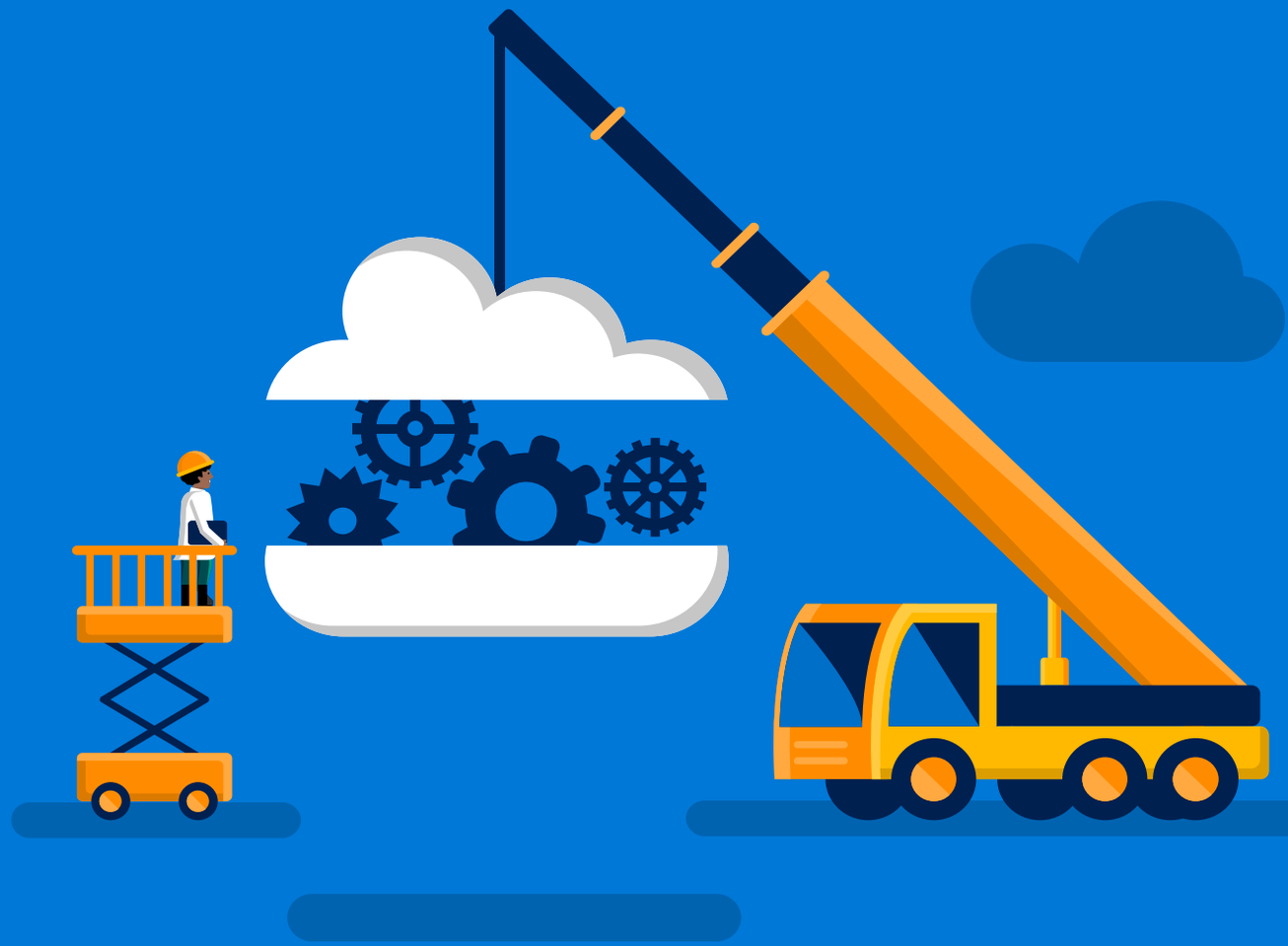
Chris Marchal
Technical Evangelist

David Gristwood
Technical Evangelist



Introduction and Housekeeping

Welcome to Microsoft UK, 2 Kingdom Street



- Fire Alarms
- Emergency Exits
- Toilets
- Lunch and Dietary Requirements

Today's Agenda

1. Introduction
2. What to do (Cloud Design Patterns and good practices) and what to avoid (Classic errors and mistakes)
    - With coffee break
3. Lunch
4. Discussion with the presenters

Microsoft

Today's Brief

How you can understand and control your Azure Cloud based services cost profile, whilst allowing your software and data to scale as and when you need it.

How do we predict workloads and frequency?

How do we allow for sudden demand without committing unnecessary costs?

Microsoft

The short answer….

How you can control your Azure Cloud based services cost profile, whilst allowing your software and data to scale as and when you need it. – **Cost Architecture**

How do we predict workloads and frequency? **Telemetry and Load Testing**

How do we allow for sudden demand without committing unnecessary costs? – **Queuing, Autoscaling and load balancing**

Microsoft

The Other Questions

How do ISV's build cloud solutions on the Azure platform for a predictable and commercially reasonable cost?

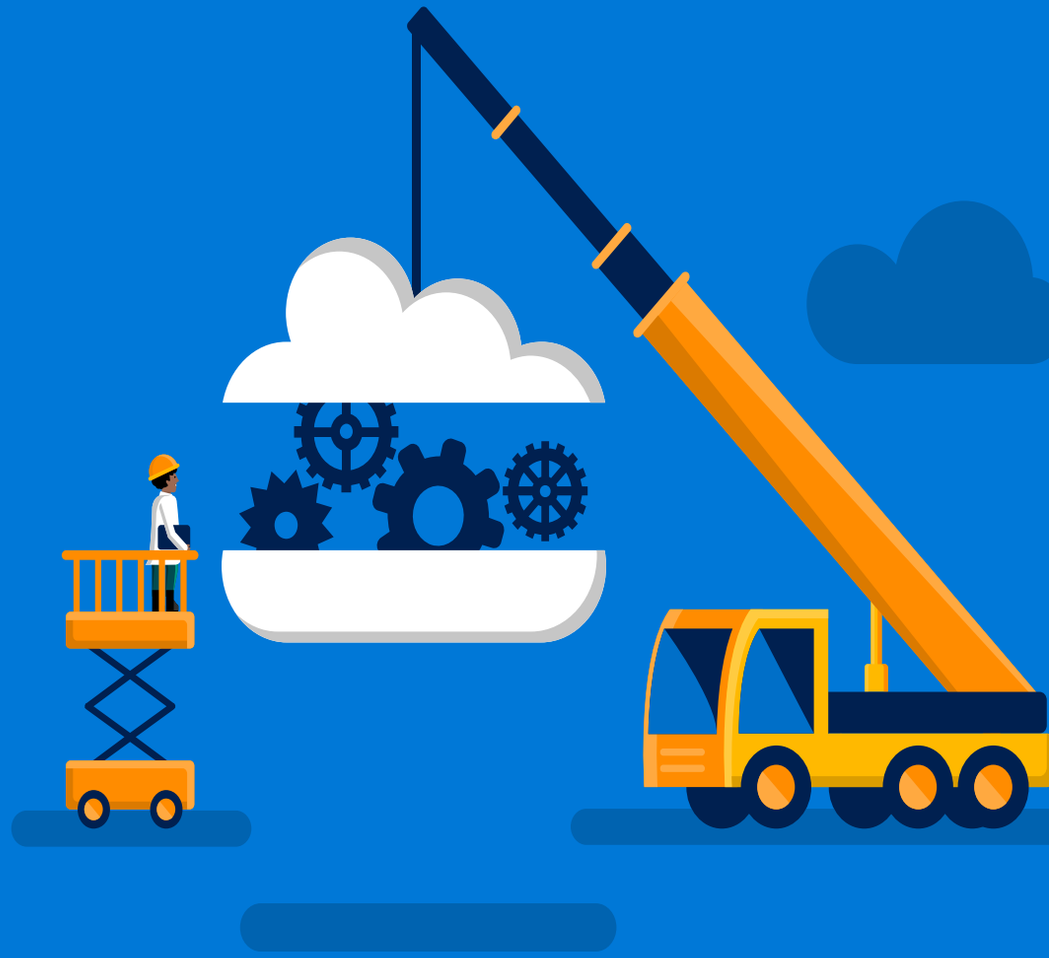
Or: How much will it cost ?
And: How much will we make (in margin)?
And: How much will we need to build / invest?

Microsoft

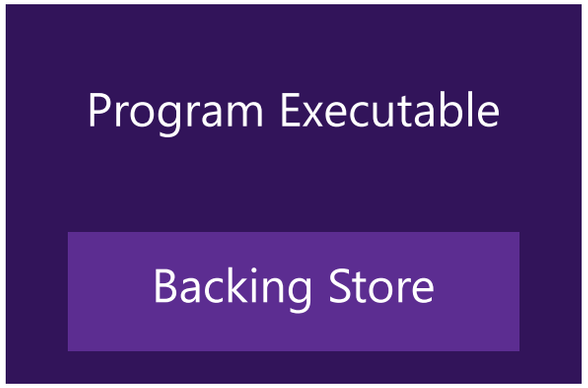
Architecture, Apps & Azure

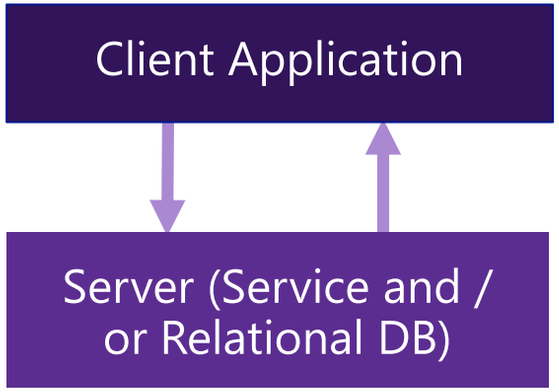
High level approach

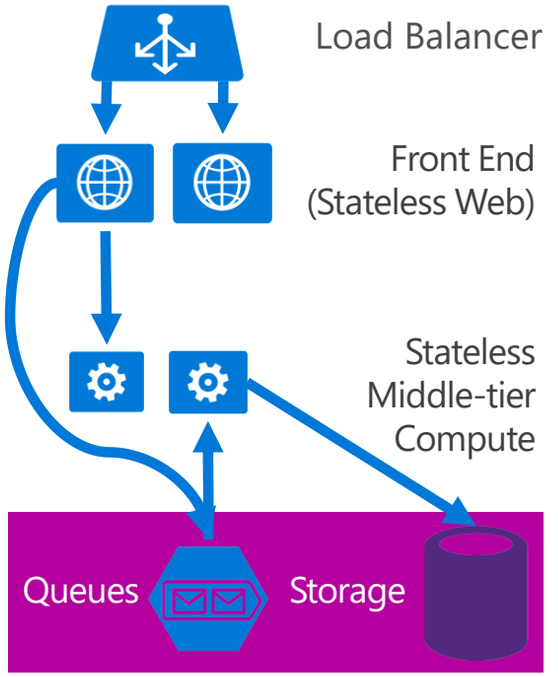
Chris Marchal



Software: Architectural Types







C-ISAM, Monolithic applications

1. 1960s to 1970s

RDBMS, Client-Server

2. 1980s to 1990s

XML, Web Services, n-tier apps, HA and Disaster Recovery

3. 1990s to 2000s

Microservices, Cloud Scale,

Distributed and Replicated State

4. 2000s to Now()

Software: Architectural Types

1. Monolithic “Big ball of mud”, no internal layering.
   Virtual Machines (IaaS)

2. Monolithically Compiled App, multi-layered design with class separation between layers.
   Virtual Machines (IaaS), App Service

3. N-tier designed app, with physical separation between tiers.
   Virtual Machines (IaaS), App Service, Service Fabric

4. Micro-services architecture.
   App Service, Service Fabric, Azure Functions

Tenancy Models

1. Single Tenant Deployment
   Manual Deployment and Build
2. Multiple Single Tenant Deployments (templated)
   ARM Templated deployment – repeated costs
3. Multi-Tenant App, with a data-tier tenant key used to isolate tenants in a database
   ARM Templated deployment – consolidated costs
4. Multi-Tenant App, with multiple data services / databases
   ARM Templated deployment – consolidated costs & data isolation
5. Multi-Tenant App, with multiple data services / databases, with clustering and balancing of services across scale units
   ARM Templated deployment / Service Fabric – flexibility and self-manage

Hardware: Public, Private, Hybrid

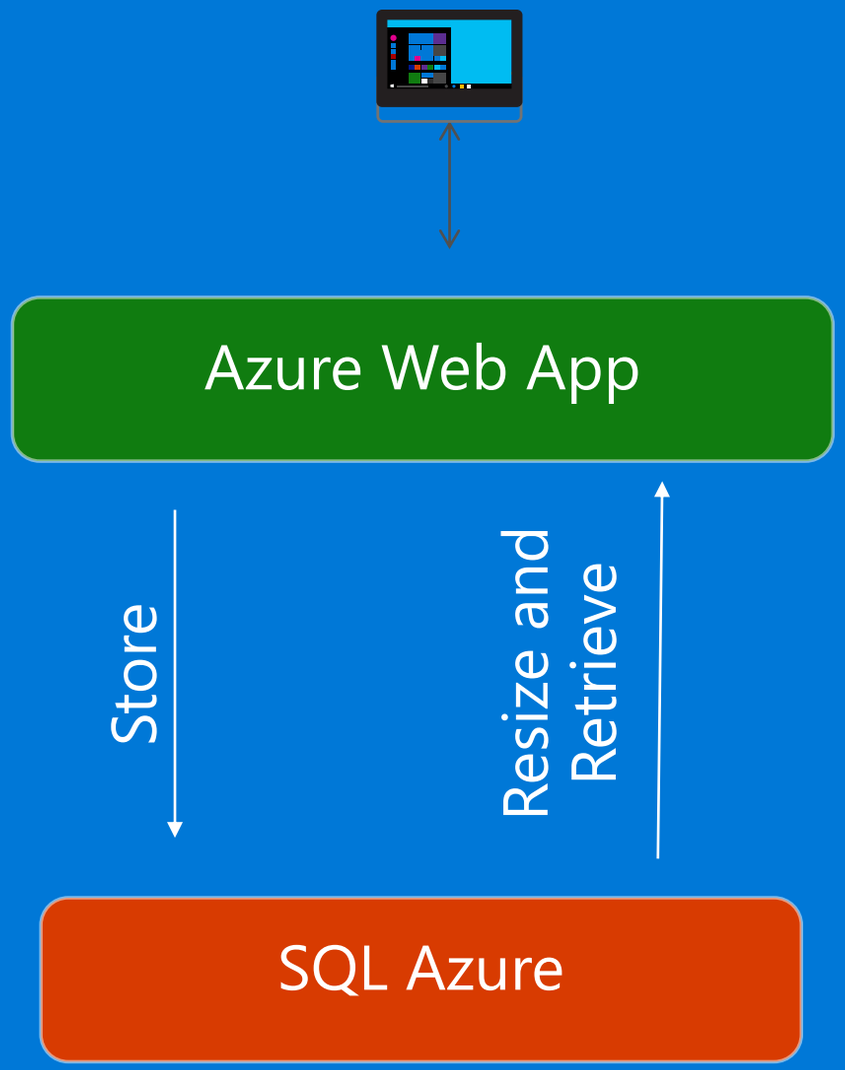
- Private cloud
  - Cloud operation efficiencies
- Public Cloud
  - We're talking about this today
- Hybrid
  - Think about the complexity in architecture
  - Part of migration or steady state?
  - Check the cost/benefit

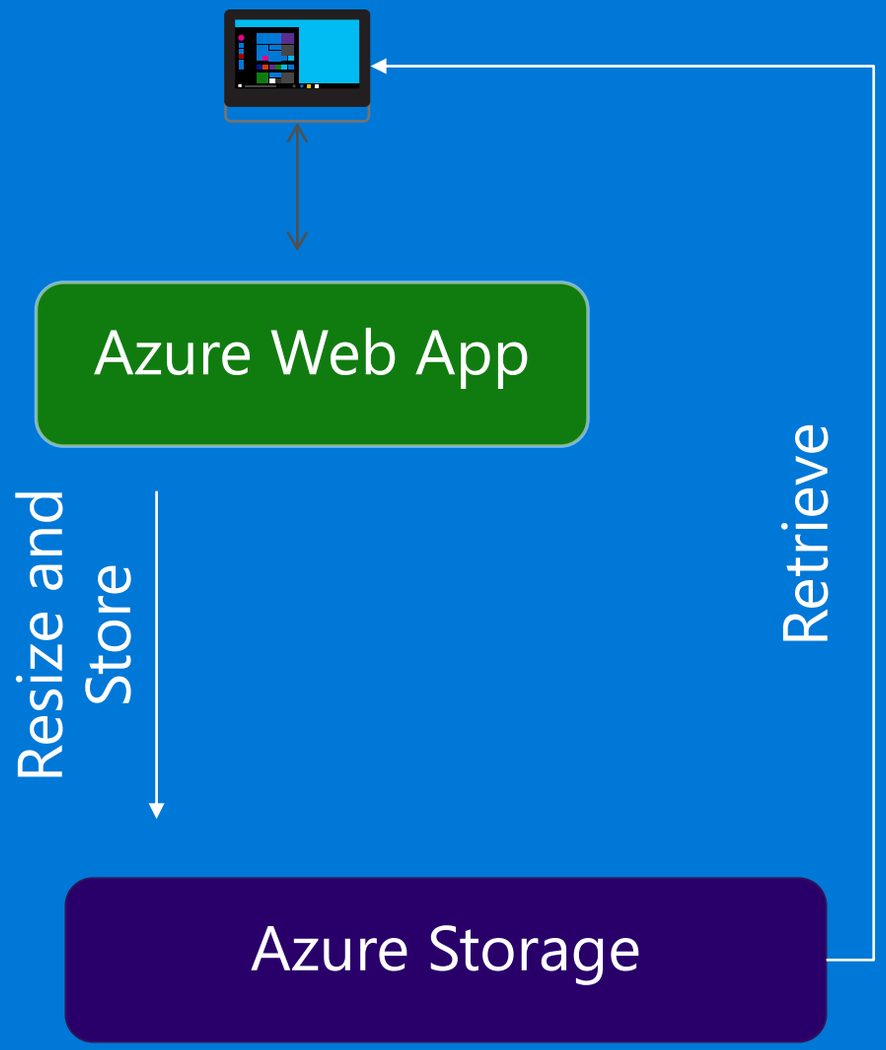
Demo: Cost and Architecture

TWO SITES – TWO ARCHITECTURES – TWO PRICES

Microsoft

Technical Design



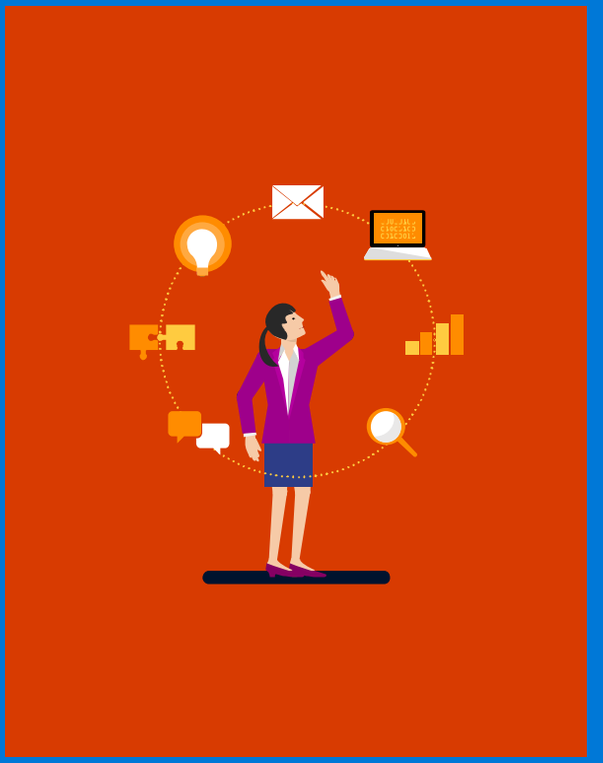


Cost model

- The explicit cost of services can be understood
  - Azure pricing calculator and public pricing

- Relating that to your architecture
  1. What services do you want to consume?
  2. How much of that service do you need?
  3. How costly are the activities your users require?
  4. How will your application scale with growth?
  5. What is the predicted growth in your user base?

- ROM costs without measurement, Indicative costs with measurement, Accurate costs through production monitoring



Development & Operational Process

How do you create software and are you agile enough to benefit?

There are costs to retarget you app dev capability

Do your operational procedures need updating for the cloud

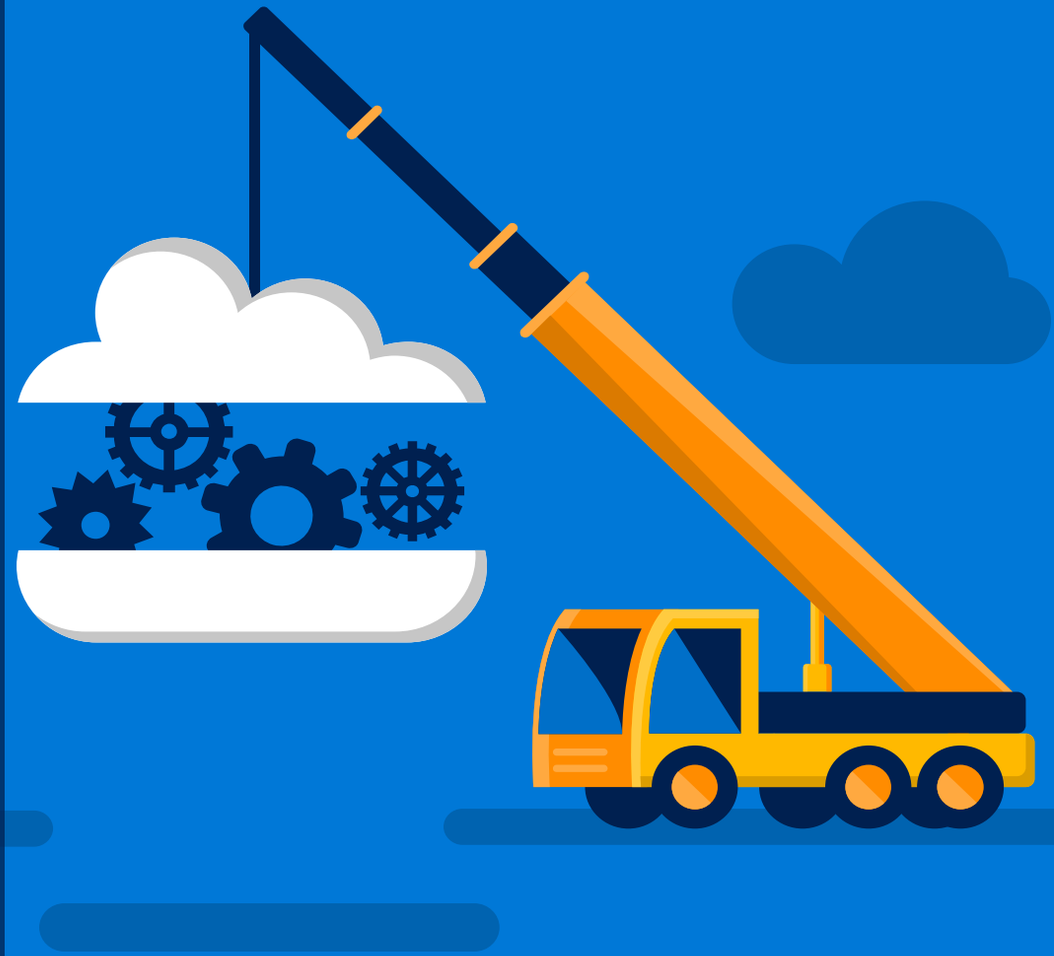
Your release and BAU processes will need updating

Do you have a DevOps cycle that includes cost and performance measurement and feedback?

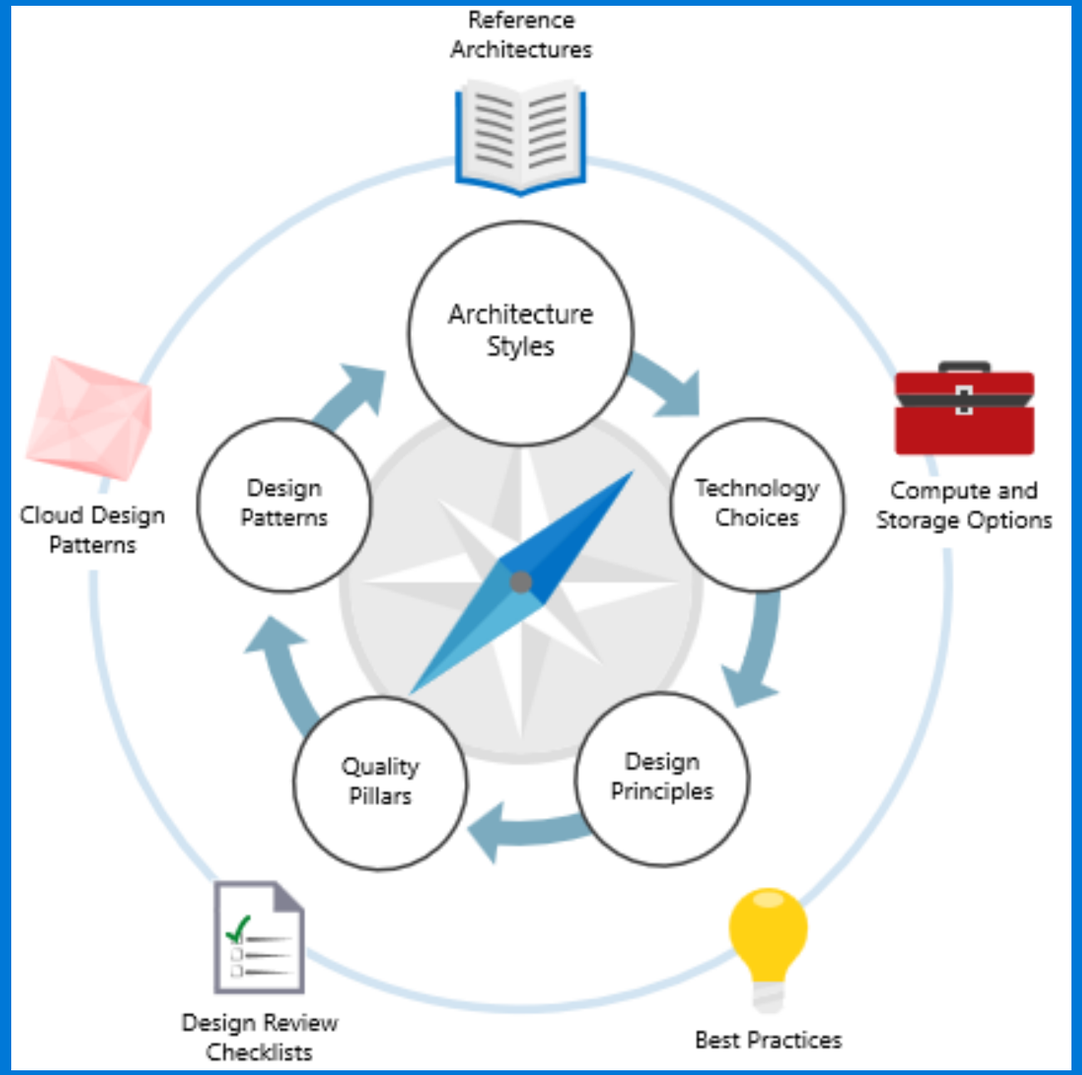
This is how you optimise for cost

Patterns and practices

What types of things can we do to balance cost and performance?



Patterns and Architectures

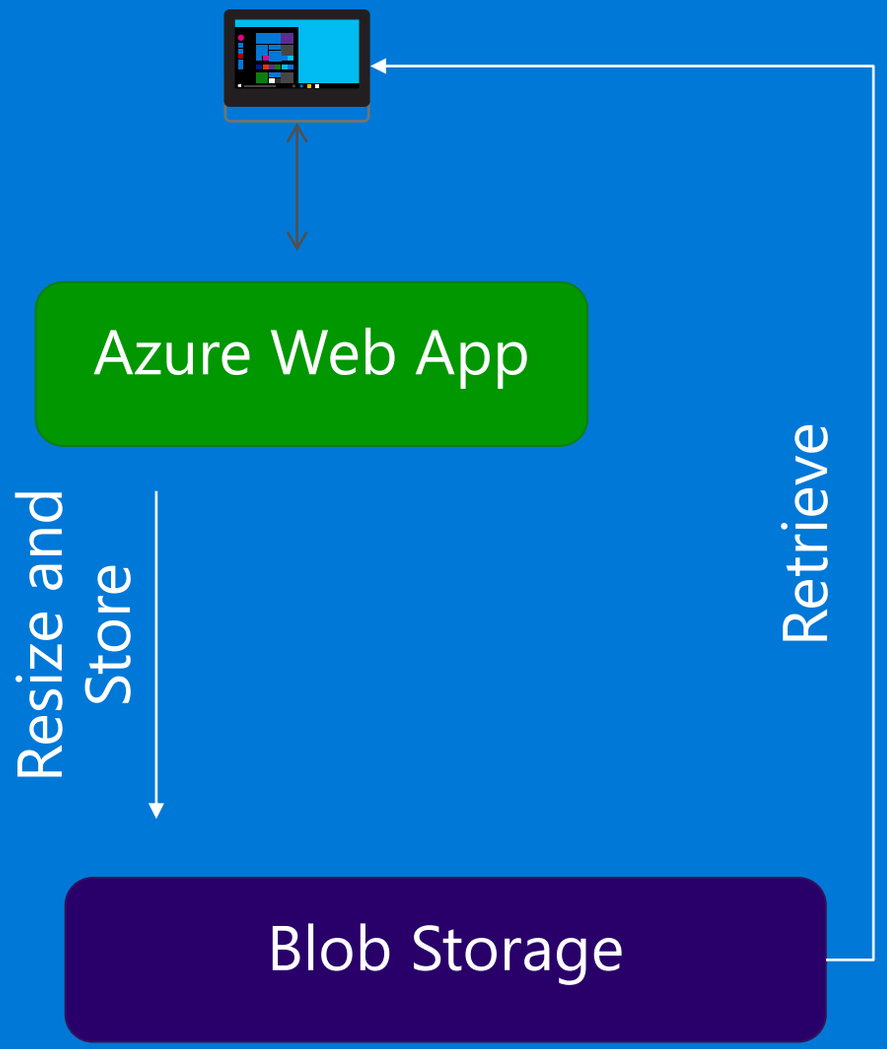


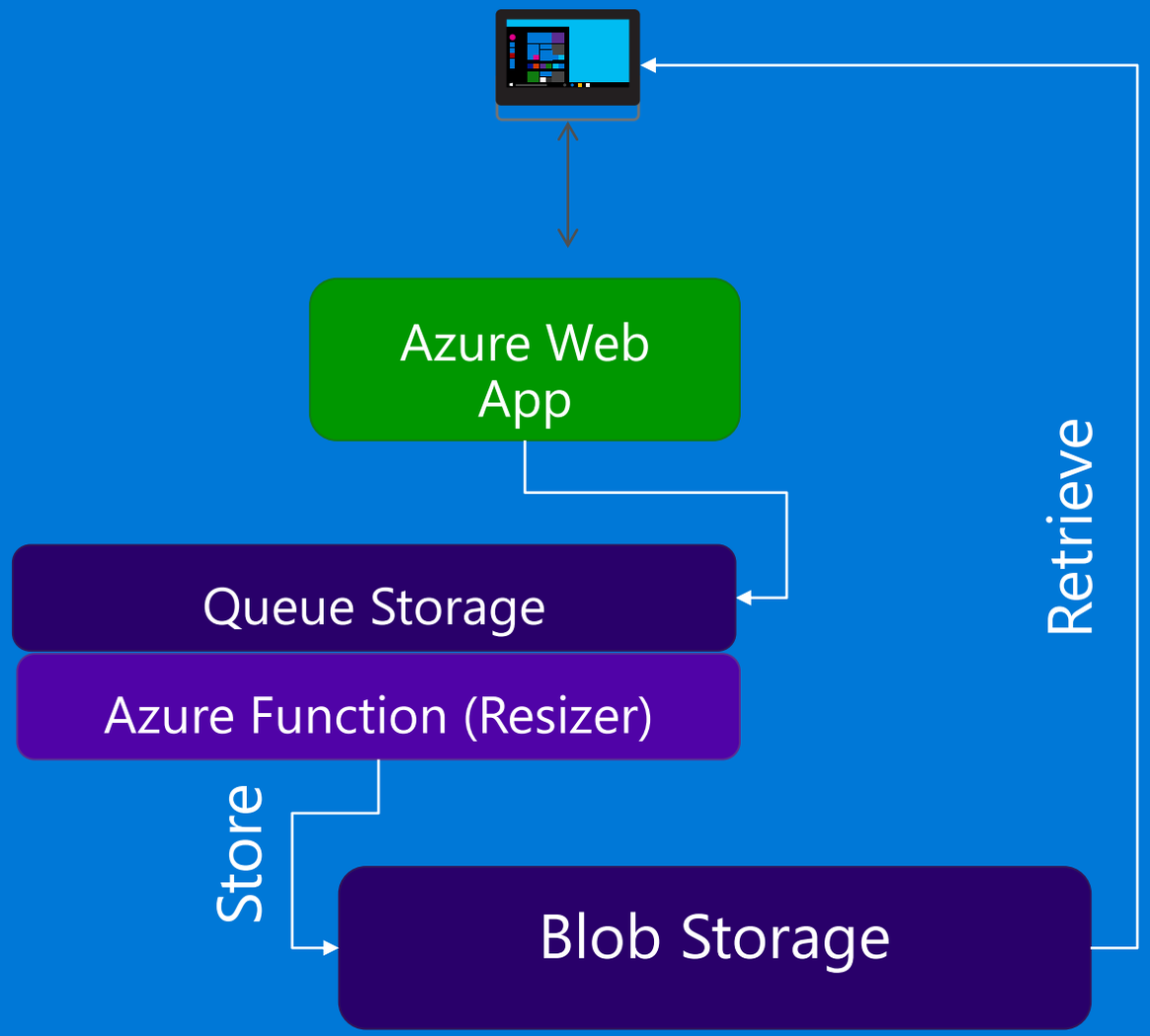
Azure Architecture Centre

https://docs.microsoft.com/en-us/azure/architecture/

- Azure App Architecture Guides
  What should I consider?
- Reference Architectures
  What have others done?
- Cloud Design Patterns
  Repeatable knowledge
- CAT Groups
  Information from the field

Technical Design vNext





Flexible architectures

Can your application **scale**?

Is that scalability **out** or **up**?

Do you know **where** to scale the application?

Is your application **decoupled**?

Can you take advantage of alternate cloud **services?**

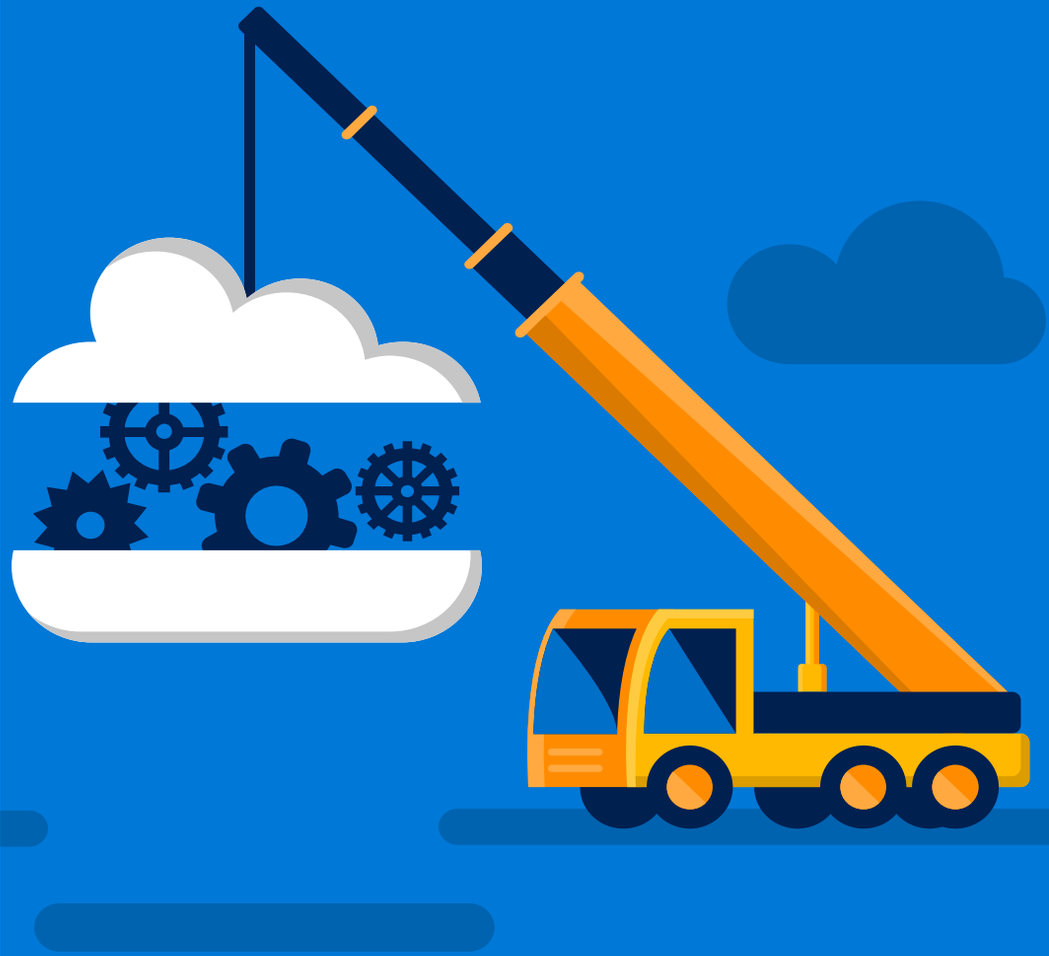
What services are **available** to me and how are people using them?

What is the return on my investment in a **change** of architecture and scale?

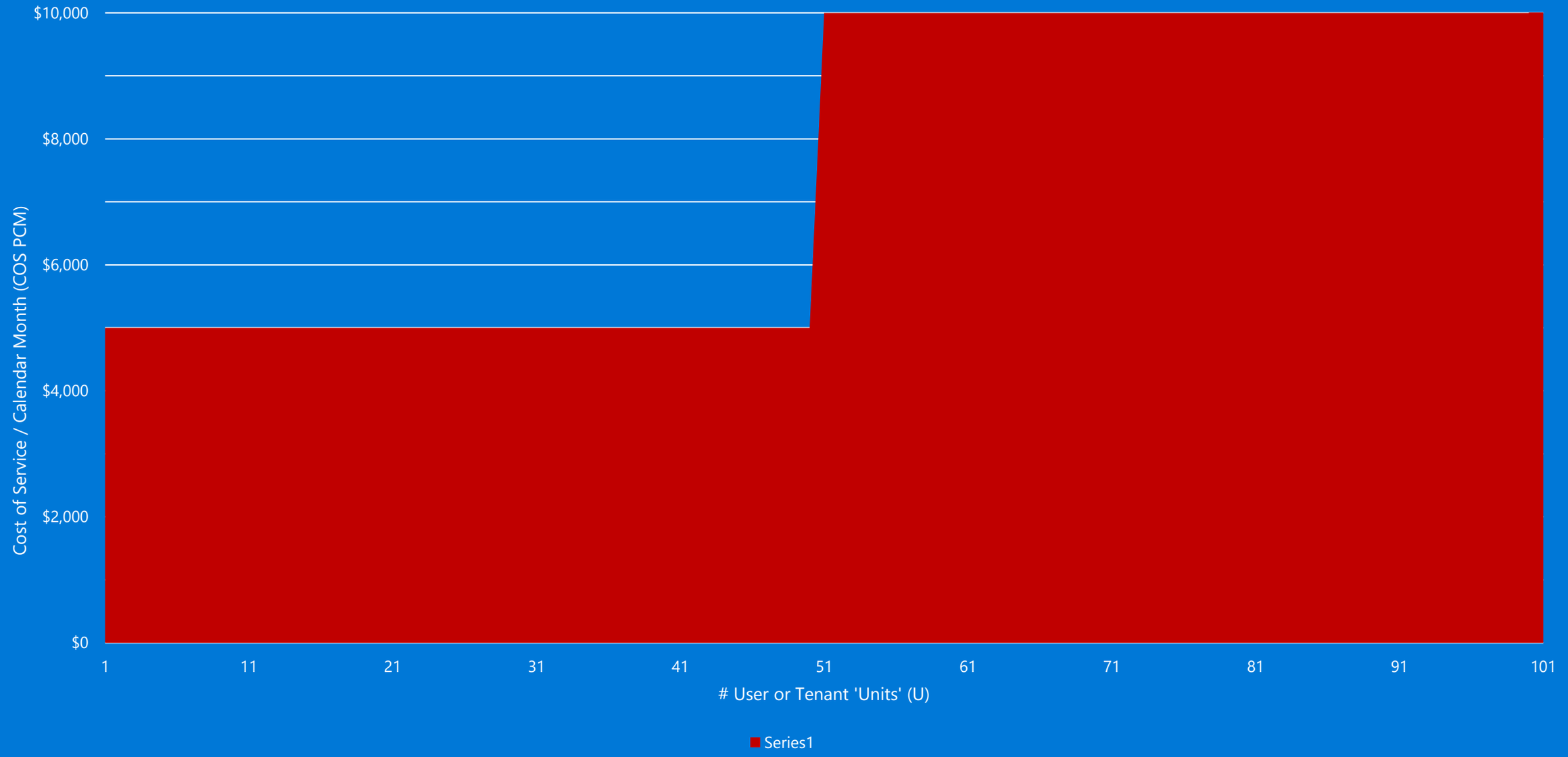
Scale Units and Profiling

What is the smallest unit of scale I can deploy ?

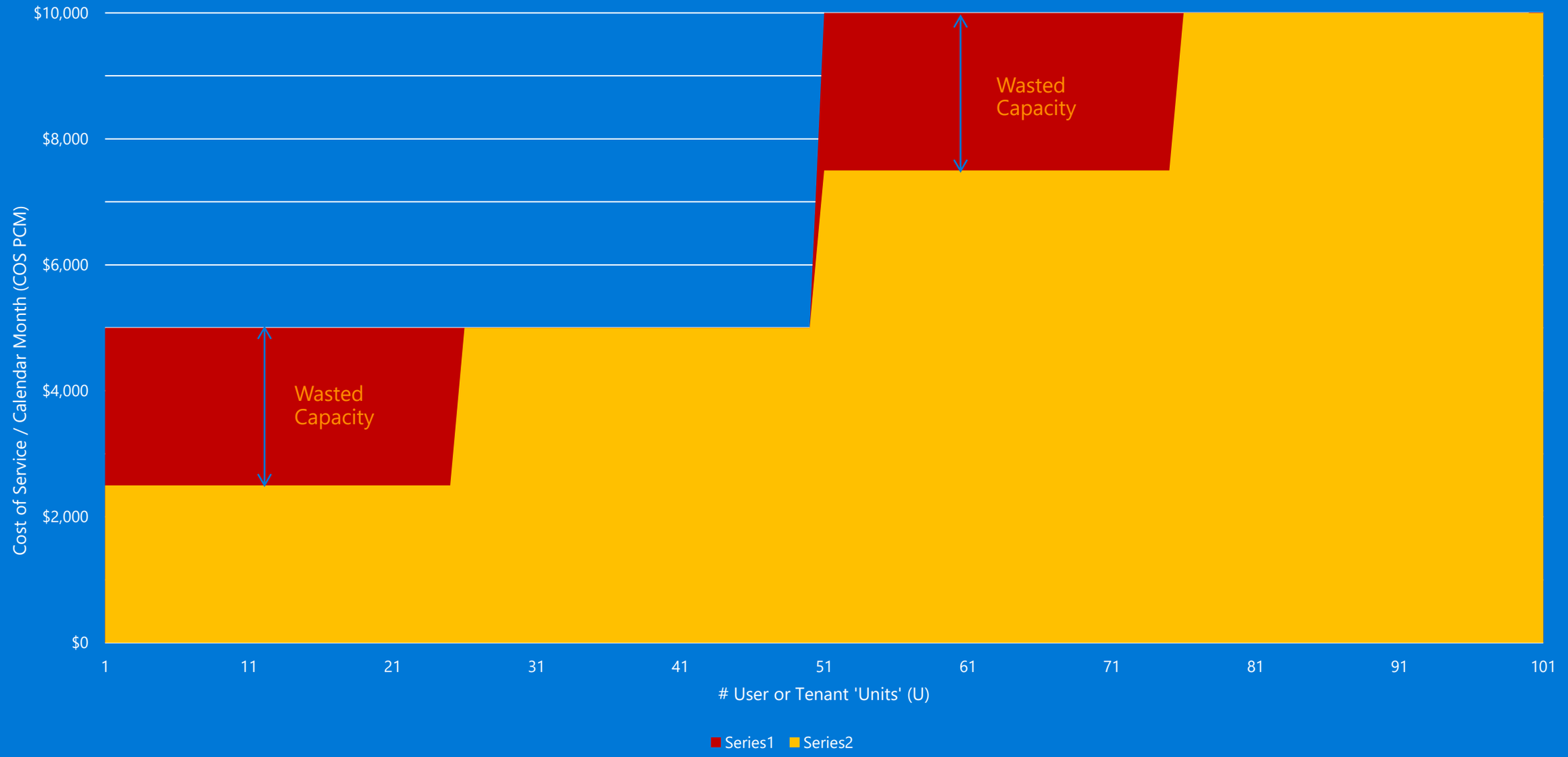
Chris Marchal



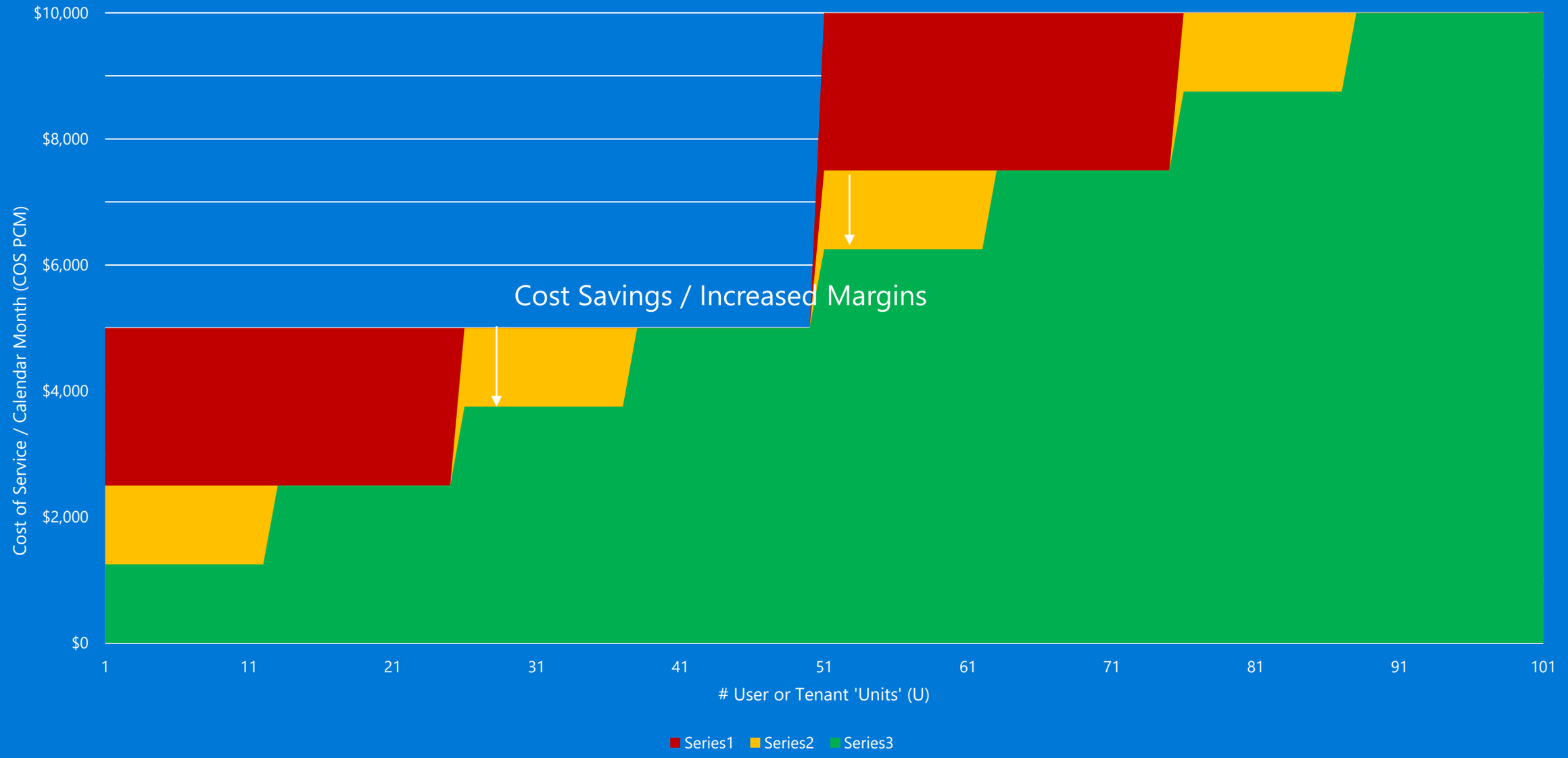
Classic On Premise Workload – over provisioned



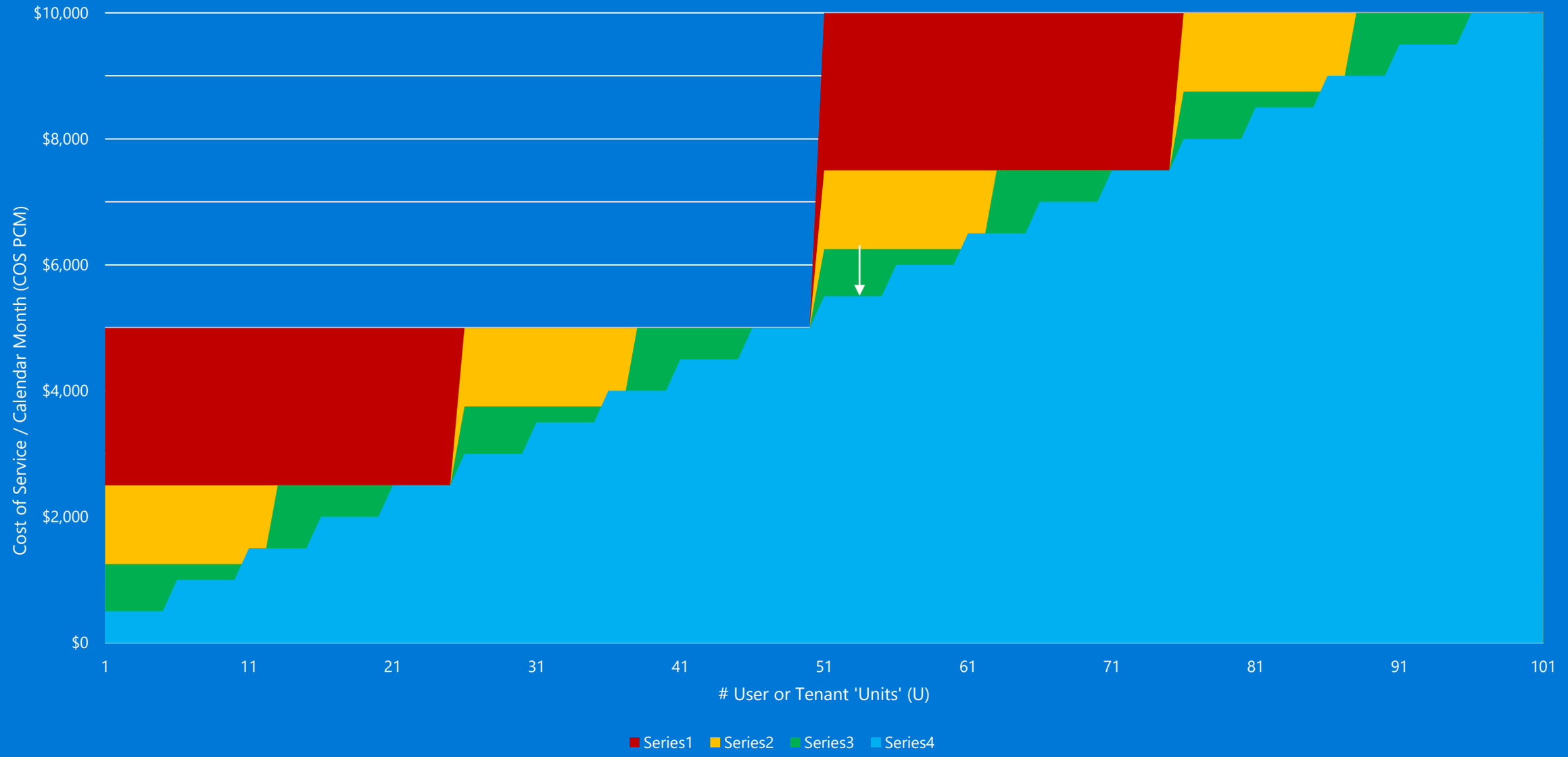
IaaS - Right-sizing Infrastructure Capacity



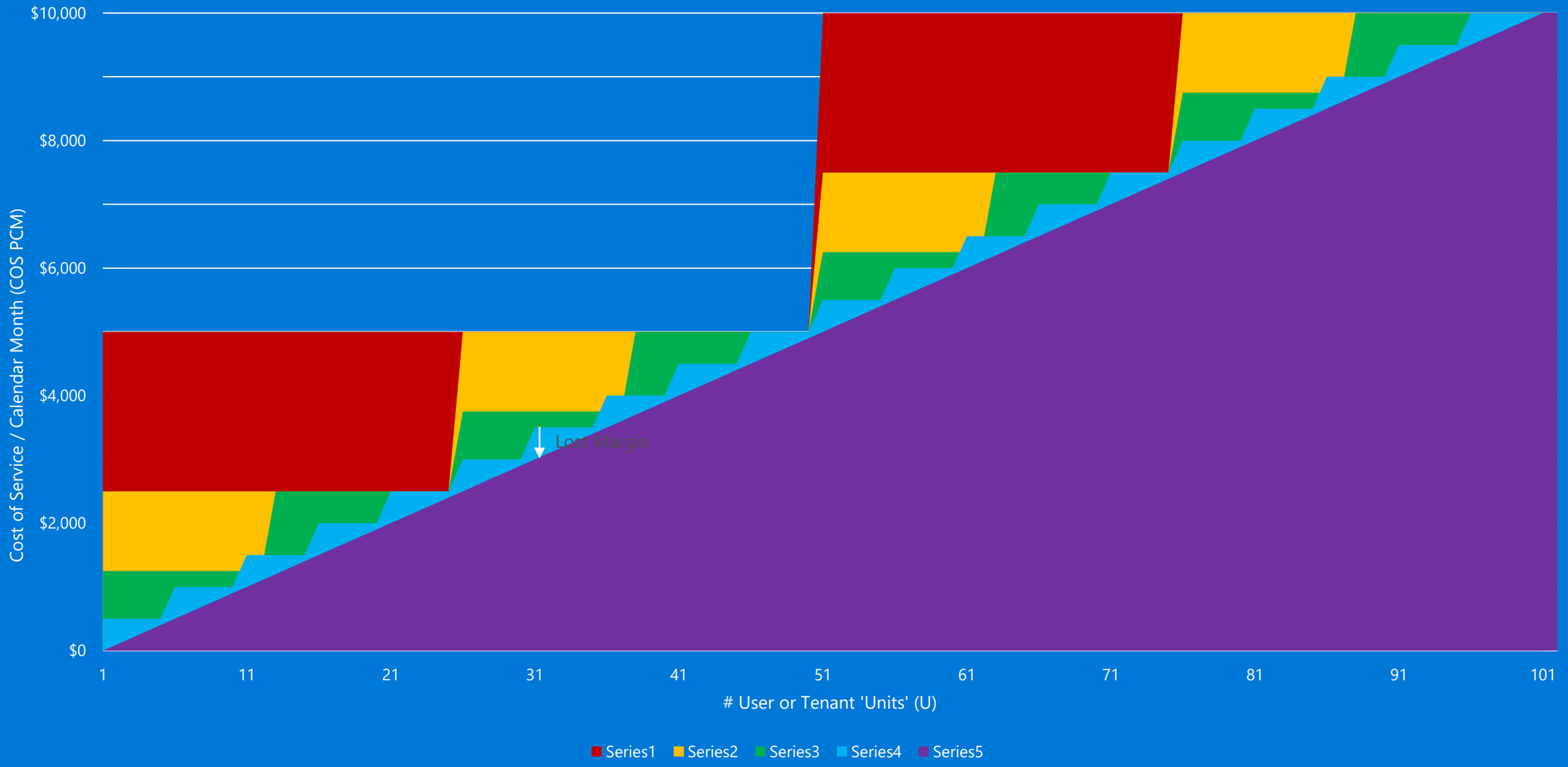
Shrinking the Stamp Size – Azure IaaS with VM Scale Sets



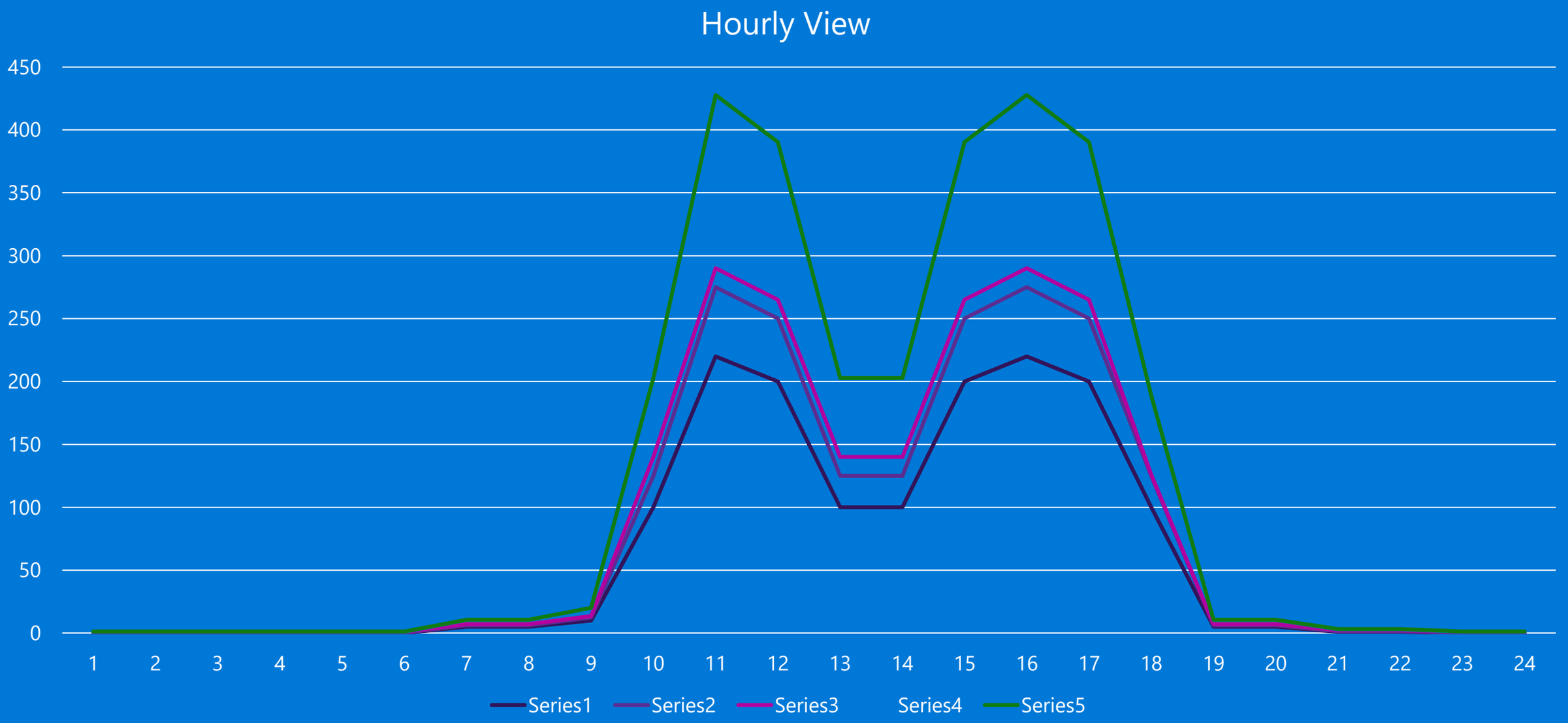
App Service (PaaS), Faster Auto Scaling



Azure Functions – Charged by Consumption (Serverless)



But we know that service consumption isn't really linear



VM Scale Sets, Azure App Service, Azure Functions and decreasing scaling latency

- Seriously consider the differences in terms of speed of deployment and scaling in your architecture between 'Classic' IaaS, Auto-Scaling IaaS (VM Scale Sets) and Azure App Service.

- The faster you can spin up a new instance to handle demand, *the smaller your minimum size of scale unit can be*.

- The design of your service is key here, the better it queues and copes with spikes in demand, *the smaller your scale unit can be*.

- If you don't want to care about scale units and just want commodity compute, look to Azure Functions to host your services.

Autoscaling – Stop wasting resources !

- Azure Virtual Machines

  No automatic scaling is provided out of the box, you can roll your own to spin pre-provisioned VMs up and down using Azure Automation and the ARM APIs if you need to.

- Virtual Machine Scale Sets

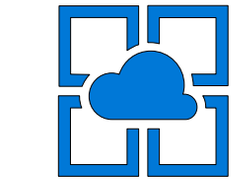
  Autoscaling rules are set to build / boot up new VMs and auto-add them to the load balancer in response to demand and trigger levels.

- Azure App Service + Webjobs

  A pre-built, pre-booted machine is pulled in from a pool of available machines and we simply spin up a w3wp.exe process running your app against a shared disk location. (As such autoscaling is almost instantaneous with app service).

- Azure Functions

  As per app service above, but your function app can be set to use a 'consumption plan' where your app is provisioned on demand to a large farm of standby servers and you are billed only for what you consume in terms of compute and memory usage.



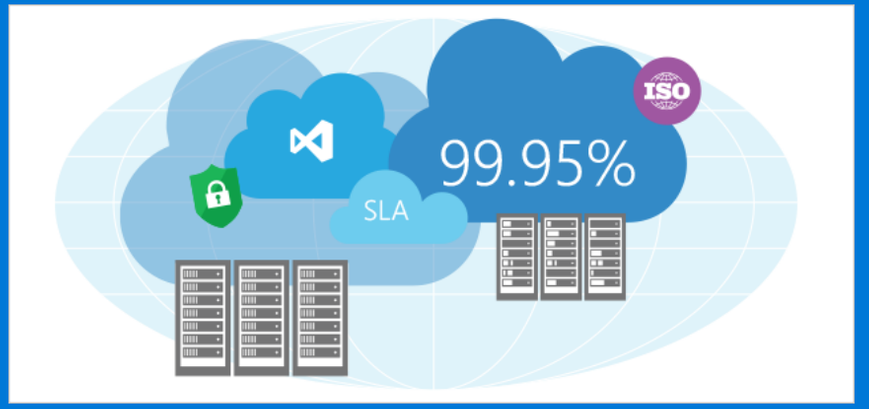
Demo: Azure App Service Scaling

Demonstrate how simple it is to configure autoscaling on Azure App Service.

Microsoft

Do: Load test your application before deployment

- At the absolute least, ensure that some kind of automated load testing is performed to detect a noticeable change in performance of specific, critical endpoints.
- Test the application at significant load to ensure that spikes in user activity and load will not take down the application at burst times.
- Test your application automatically **before** deployment using CI / CD Practices, the earlier you find problems, the cheaper they are to fix.
- Knowing your baseline operational cost of a service on a fixed hardware stamp makes it simpler to predict how to cost and upscale the service.



Demo: Cloud Load Test

Let's baseline our application's initial performance so we know where we are starting from, let's use the VSTS Cloud Load Test Service to do this.

Demo: Application Insights

Getting telemetry about how your application is used and performs.

Microsoft

High level take aways aka “what not to do”

Chris Marchal

What to avoid in Cost Management

LACK OF COST OWNERSHIP AND REPORTING

Many Customers do not have someone in place to monitor cost on Azure.

This should be done at a level appropriate for the organization, e.g. project or team level.

OVER-PROVISIONING OF RESOURCES CAUSING WASTAGE

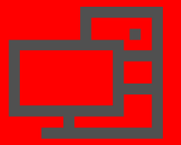
Some customers over-provisioned resources, even though such sizing was not required.

Customers should perform load testing, and be aware of the necessary limits of their service.

PAYMENTS MADE BY CORPORATE CREDIT CARD

Some customers are managing their Azure payments via credit card.

The customer could be missing out on a number of discounts and enhanced support offerings via their Enterprise Agreement.



DEV/TEST ENVIRONMENTS NOT DE-PROVISIONED

A number of customers have not de-provisioned their dev/test when not required.

To reduce cost, customers should consider scaling down / shutting off their dev/test environments when not in use.

What to avoid in Efficiency

Over complex solution design

In a number of scenarios, customers have over-engineered their solution, to solve the problem at hand. This has typically introduced unnecessary cost and risk/complexity within the solution.

Take an external view when working with the customer, and challenge their approach. Could they be using an alternate approach, which would be simpler and reduce their end-cost?

Lack of clarity on non-functional requirements

Some solutions have been over-engineered for the level of SLA that they require, or they have an SLA which is impossible with their solution architecture. This can result in either excessive risk, or cost.

Customers should ensure that they are fully aware of their requirements in advance of design. Strongly challenge them on this point – How can they build a solution meeting availability , scalability and cost requirements, if they are undetermined?

Links and Downloads

This session - Presentation Repository

https://github.com/dxuk/AzureWorkshops

Evaluation Forms

We hope you have enjoyed the morning, we have one ask of you.

Please out your evaluation forms to give us feedback on what we did right and how we can improve the sessions next time.

Your feedback is critical to making more of these sessions a success !

Lunch

Group Discussion Track after lunch !

Chalk ‘n’ Talk

Interactive Discussion