Welcome to Country

We wish to acknowledge the traditional custodians of the land we are meeting on, the Whadjuk (Perth region) people. We respect their continuing culture and the contribution they make to the life of this city and this region.

This meeting is being held on the traditional lands of the Noongar people. We acknowledge that this meeting is being held on Aboriginal land and recognise the strength, resilience and capacity of Noongar people in this land.

South West Aboriginal Land & Sea Council

Microsoft

Upgrade my PaaS

An Azure Story

17 August 2017 | Derek Bingham

Tweet: @deekob

Agenda

01 The What and Why of PaaS

02 Azure Compute PaaS Options

03 Current flavours of the month

04 Case Study – Upgrade my PaaS

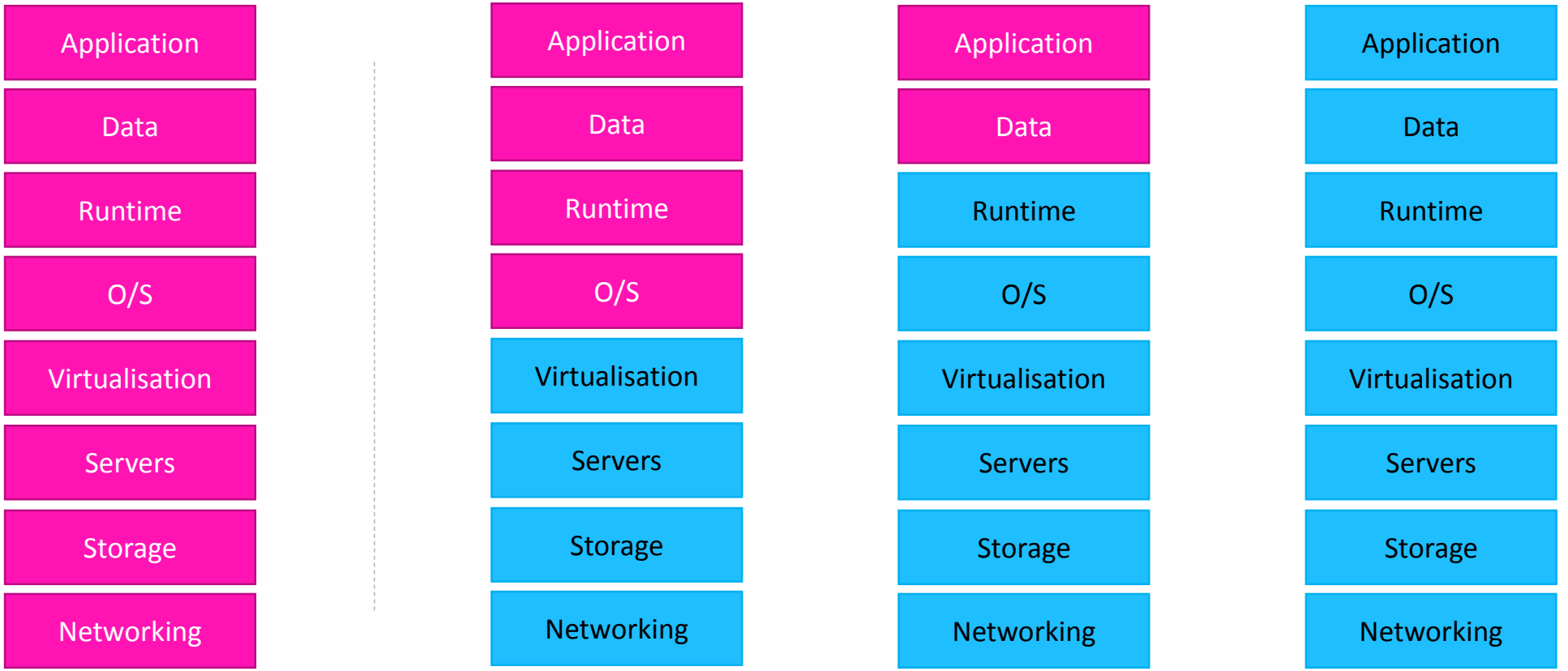
05 Steps to solve

06 Tips and Tricks

07 Questions

08 #beertime

What is PaaS?



What is PaaS?

Amazon Lightsail

Azure VM Extensions

Azure VM Images

Azure Cloud Services

AWS Elastic Beanstalk

Azure Service Fabric

Azure Container Service

EC2 Container Service

Service Fabric on VMs

Azure Logic Apps

Azure Batch

Azure Functions

AWS Lambdas

AWS Batch

Amazon EC2

Azure VMs

Azure App Service

Azure VM Scale Sets

AWS Auto Scaling

Azure App Service Environments

Azure Container Instances

IaaS
(Control / flexibility)

PaaS as a continuum

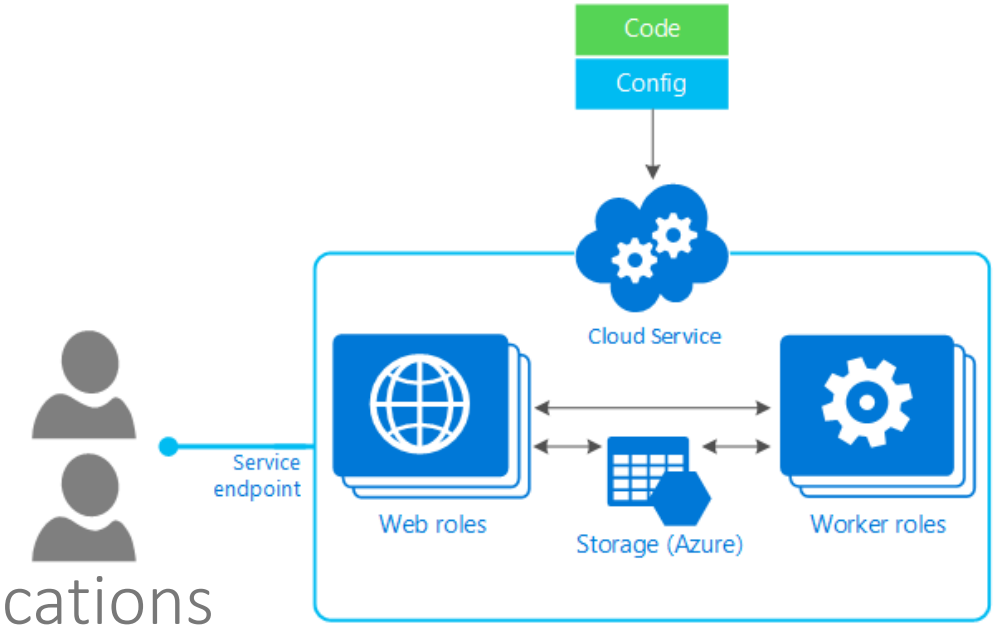
Showing AWS/Azure compute

PaaS
(Simplicity / ↓ maintenance / ↓ h/w access)

Part 1
Azure Compute PaaS

Cloud Services – PaaSv1

- Thin abstraction over Underlying VM
  - Can RDP / SSH
  - Can install 3rd party binaries on VM
- Applications are split into Roles
  - Web Role – Website running on IIS
  - Worker Role – Background standalone applications
- Scales up and out
- Supports HA and Geo Redundancy
- But any new service deployment can take 10-15 mins to start
- Each new service needs a new cloud service – needs a new VM - $$



App Services – PaaS V2

- Web Apps
  - Web Sites , Web Jobs
- Mobile Apps
  - The ‘backend’ for your Mobile apps – data , content, security..
- Logic Apps
  - Workflow, integrations
- Api Apps
  - Exposing your API using REST endpoints

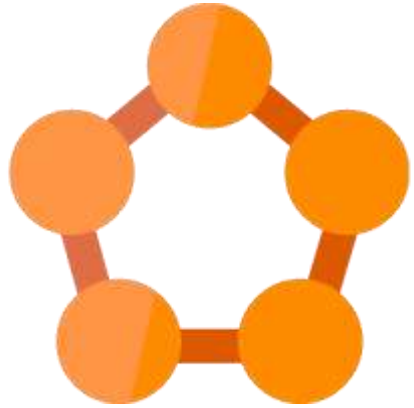
Web Apps

- Runs under IIS
- Uses Shared resources – App Service Plan
- Can scale automatically up and out
- You can hook into GitHub and other source control tools to set up CD
- You don't get access to the underlying OS instance so **there is less flexibility**
- The first PaaS you'll build in Azure

Azure Web Jobs

- For running background tasks
  - OnDemand – Queues, Blobs, storage…
  - Scheduled
  - Continuous
- Runs in same context as WebApps
  - Watch out
- Scale with the containing instance

Service Fabric



- Azure ‘Microservices Platform’ ™
  - Distributed services platform
- The Service Fabric Cluster…
  - Runs on linux / windows / onPrem / Azure / AWS
  - Runs 30% of ALL Azure compute
- Supports deployment of existing executables / .Net Core binaries
- Run Docker inside cluster
- Overkill for a simple website and a lot to get your head around..
  - Both program and deployment model
- But **one to watch** !!
  - especially with container integration

Azure Functions / FaaS / Serverless (WebJobs 2.0)

- Develop Functions in C#, Node.js, F#, Python, PHP … more
- Schedule event-driven tasks across services
- Expose Functions as HTTP API endpoints
- Scale based on demand
- Just like WebJobs….

So Why Functions?

- Run in consumption plan
  - Timeout after 5 mins – short running process
  - Pay for what you use(memory and cycles) – no heavy lifting
- Run in AppService Plan
  - A WebJob as a method
- Flexible – simple deploy from Git
- ~~Still waiting on VS integration~~ – Released Yesterday
  - But can use VSCode – so its ok..

Let’s have a look…

Azure Batch

- Job Scheduling as a Service
- Large Scale Compute !!
  - Monte Carlo Estimations
- Control VM Pool size (100 – 1000)s
- Plumbing all done for you

Container Services

- Controlling Containers across VMs
- Scaling inside VM's
- Runs on Scale Sets
- Uses Kubernetes and Docker for orchestration

Container Instance (Preview)

- Containers as a Service (CaaS)
- Single Container deployed billed by second (like Functions)
- No infrastructure to provision
- No cluster orchestration
- Currently Linux only – Windows coming
- Can connect to Kubernetes for Orchestration
  - VM based containers for predicable work loads
  - On demand Container Instance for fast bursts and scaling
  - #mikedrop

Containers, Containers, Confusion….

IF YOU'RE LOOKING FOR THIS…	USE THIS
Scale and orchestrate containers using Kubernetes, DC/OS or Docker Swarm	Container Service
Easily run containers on Azure with a single command	Container Instances
Store and manage container images across all types of Azure deployments	Container Registry
Develop microservices and orchestrate containers on Windows or Linux	Service Fabric
Deploy web applications on Linux using containers	App Service
Run repetitive compute jobs using containers	Batch

Take away: PaaS options are many and varied – choose **wisely**.

readify

Part 2
An Azure Story

CASE STUDY

Cloud Services Migration

Brief:

Op-Ex costs are blowing out on

Azure resources. Can they be consolidated? and if so how?

Problem Statement:

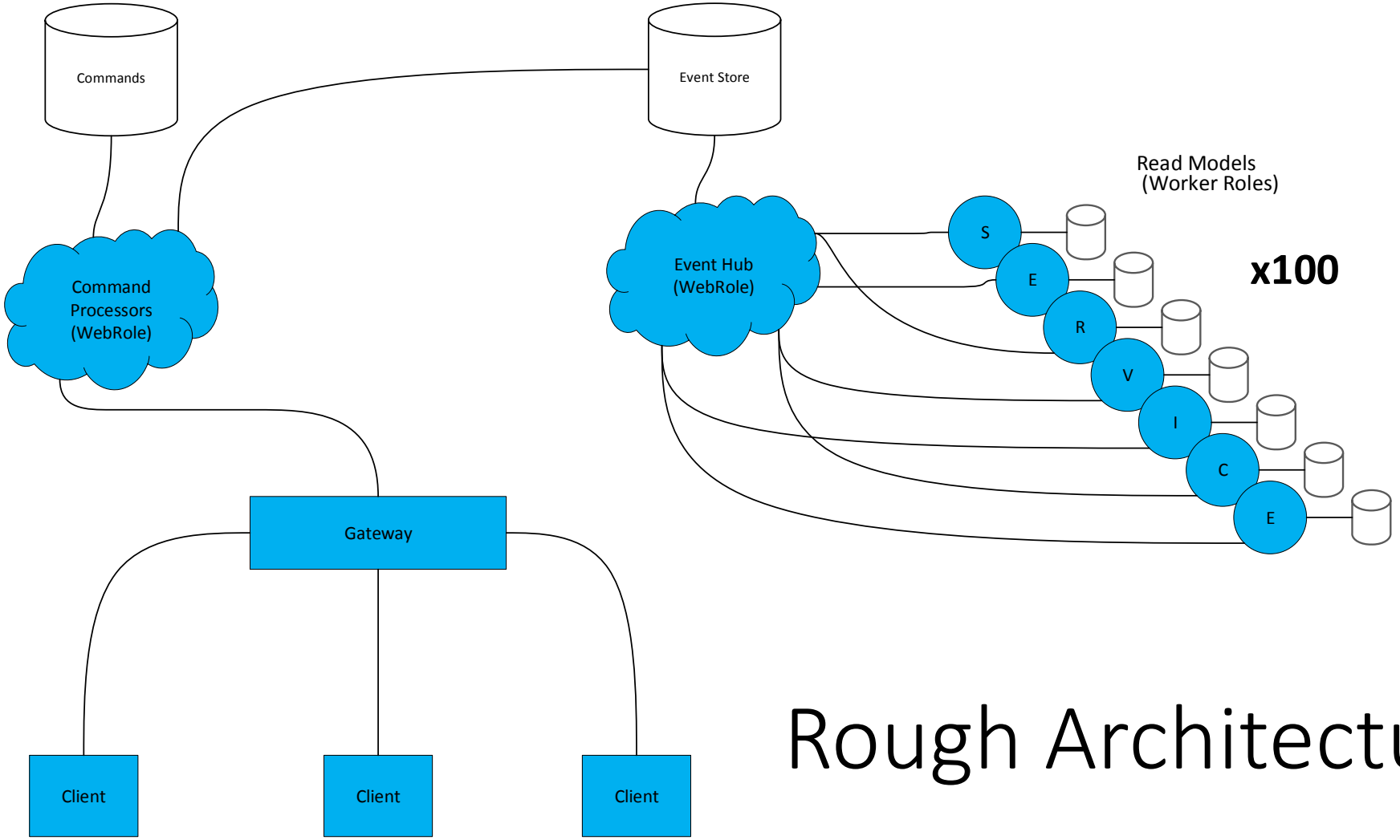
Running system in Azure, that has been deployed and running well for 2+ years.

Due to organic growth in application services, current PaaS platform isn't cost effective.

Find a better more cost efficient PaaS platform and migrate to it…..

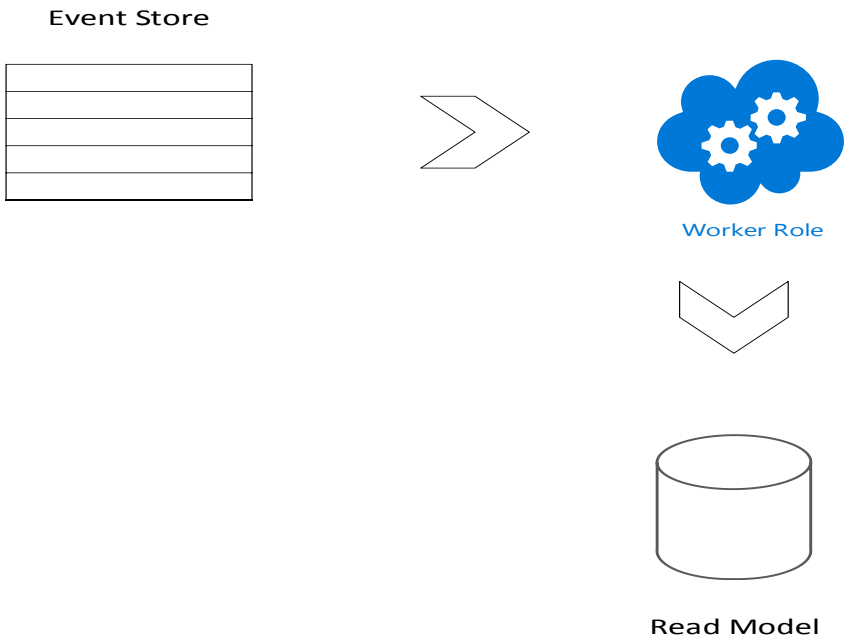
Duration:
Four Months

readify



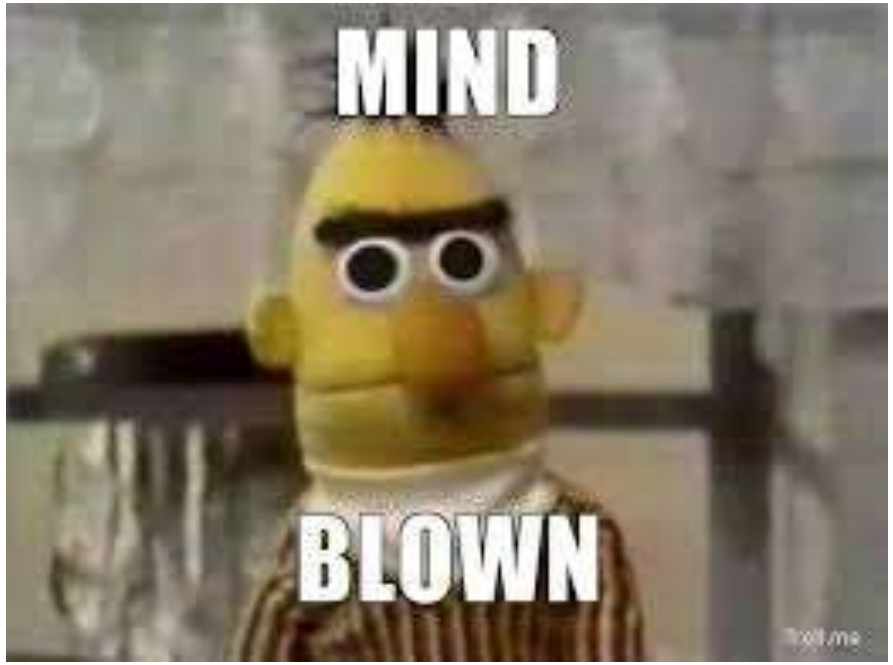
Rough Architecture....

The Humble Read Model Populator



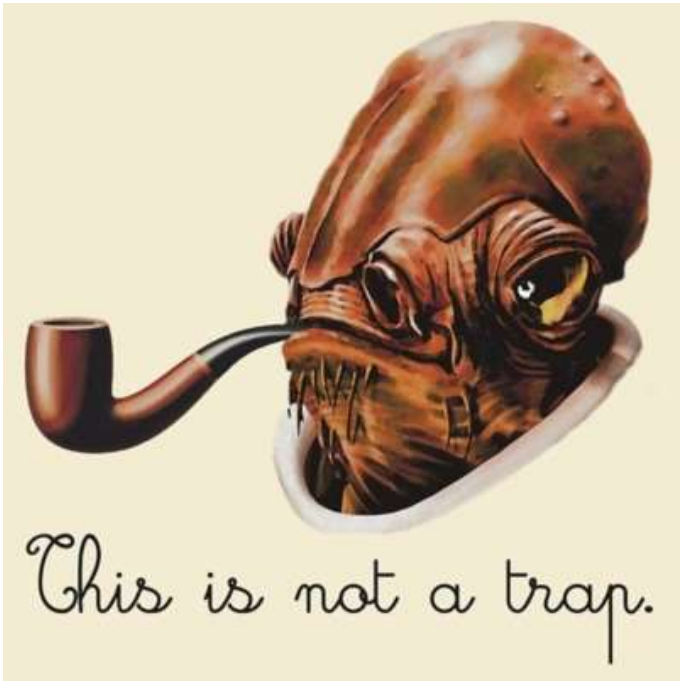
Some Numbers

- Number of Could Services – **123**
- Number of C# projects – **729**
- Number of Team City Agents – **21**
- Number of .Net Solutions – **130**
- Time to deploy 1 Service – **15** mins



The Challenge

- Choose a new PaaS platform
- Migrate 130 + solutions
- Support intra day deployments
  - At least beat 15 mins outage
- Maintain support to existing ALM
- Leverage existing Team City Agents
- Support scaling at Micro and Macro levels
- In under 4 months



Where to start?

- Respect your patterns
  - https://docs.microsoft.com/en-us/azure/architecture/patterns/compute-resource-consolidation
  - Compute partitioning

- So what can I partition ?
  - WebRole
  - WorkerRole

- What PaaS platforms would support this?
  - Service Fabric – preferred MSFT option
  - WebApps/WebJobs

Find THE Platform ?

- Spike two platforms that could be considered ‘best fit’
  - Application Services / Web Apps
  - Service Fabric
  - Time constraints ruled out spiking more
  - Containers and Functions would have also been reasonable options
- Build an Worker Role replacement in each platform
  - Biggest risk as there are > 100
  - Validate it against criteria
  - Good vertical slice

Results

Spike Criteria	Web App	Service Fabric
Perfect Architectural replacements	No	Yes
Can control singleton service instances	Yes	Yes
Easy to Scale	Yes	Yes
Almost instant deployments	Yes	Yes
Team City deployments	Yes	Yes
Minimal code change required to each solution	Yes	No
Well documented and understood set of tooling	Yes	No
Existing build and deployment pipeline can be leveraged, so deployments to new environments can be 'feature' toggled	Yes	Yes

Pick Me!

Proposed Solution

- Worker Roles -> Web Jobs
  - Can we find a cross cutting technique ?
  - Limit code churn
  - Reduce downtime
  - Deploy into a compute pool
- Web Roles -> Web Apps
  - Apply same cross cutting technique
  - Standardise the entire migration approach

Solution Risks / Benefits

- Risks
  - Technical cross cutting may not be possible
  - Will need to change 100 + solutions
  - Extensive testing required
  - Will take a looooooong time….

- Benefits
  - No / Limited actual code change
  - Changes limited to build and deployment pipeline
  - Limited regression testing
  - Can Stand up services side by side / new and old
  - Maybe actually deliver on time

Solution – Part 1 - Use the folder structure

> Web App Directory structure is static

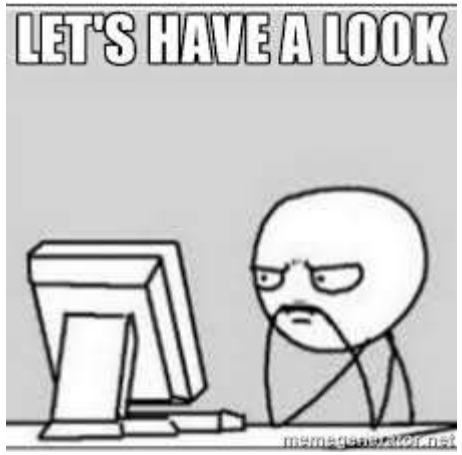
site\wwwroot\App_Data\

> Web Jobs live in here too

site\wwwroot\App_Data\jobs\continous\{job name}

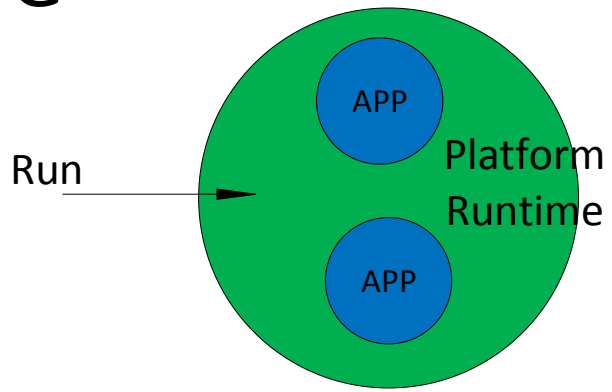
site\wwwroot\App_Data\jobs\scheduled\{job name}

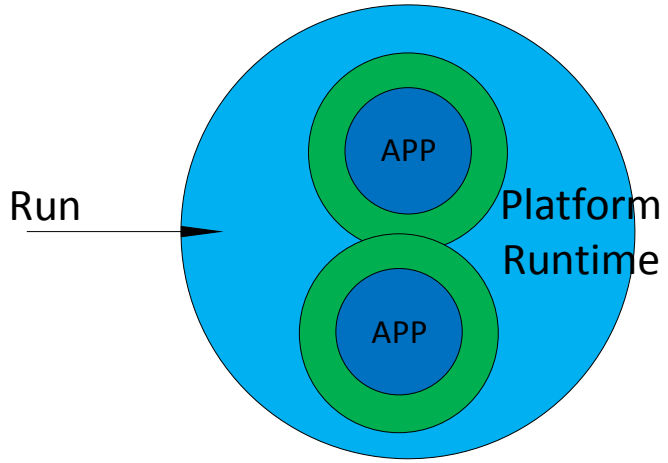
> Any files below job framework will try and run
  > .cmd,.bat,.exe,.ps1,.sh,.php,.py,.js,.jar



Solution – Part 2 - Fool the runtime

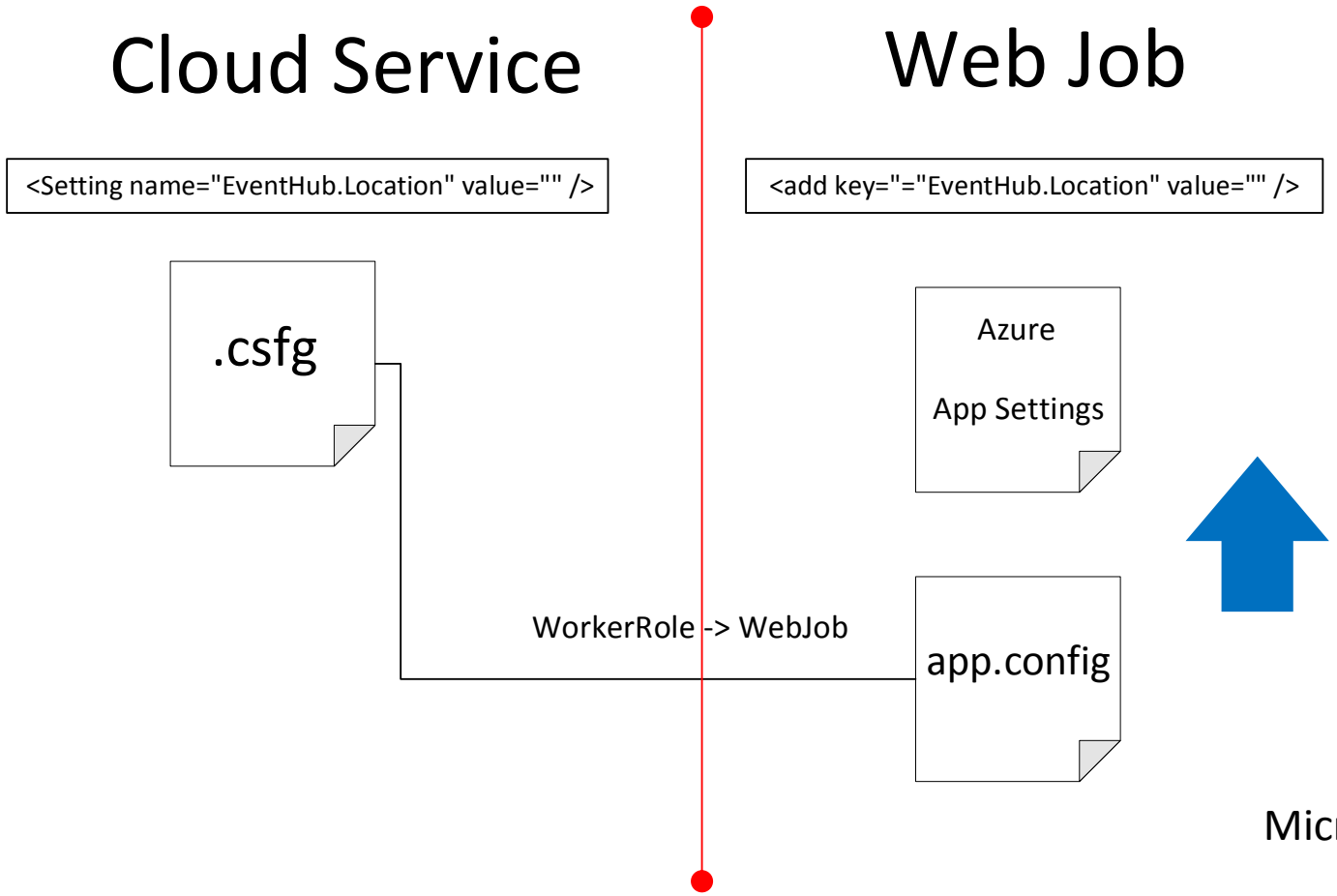
- Cloud Services Worker Roles Entry Points
  - OnStart()
  - Run()
- Fool an application that its running on Cloud Services?
- Onion Console was born..
  - Simple console app
  - Runs from the application binary directory
  - Finds the Cloud Server entry points
  - Calls them





https://github.com/deekob/OnionConsole

Solution - Part 3 - Configuration migration



> Transform CloudSevice config
> Add to app.config of onion
> User config inheritance
> Apply for all role types

Microsoft.WindowsAzure.ConfigurationManager

Solution – Part 4 - App Service Plans

- Organise your applications/services into ‘Compute Pools’
  - Long running continuous Jobs
  - On Demand Jobs
  - Using ARM template to provision
- Create a ‘Headless’ WebApp for your Web Job pool
  - Removes risk of Web Request flooding your Job Pool
- Consider Use ‘Per App’ scaling
  - Defaults to scale of AppService Plan
  - But can over ride at the App Level
- Watch out on Swap Slots
  - They consume resources from same App Service Plan

Solution – Part 5 - Msdeploy

- Can add/update webjobs
  - Extended to do the same for webapps
- -skipDirective – used to ignore certain files
  - Typically web extension directories
  - But also job.settings
- Flexibility on how to authenticate
  - Publish setting
  - Credentials
- Could call from cake or powershell or commandline
- Tip - Look in package/obj folder for some guidance

How did it go

Team City Agent time per service	33% saving – 90 secs to 60secs
Deploy down Time	15 mins to < 1 min
Web Endpoint downtime	15 min to 1 min (can be 0 with slots)
Overall Cost Savings	Non-Prd 50% - Prd TBD
Compute Cost Savings	Non-Prd 75% - Prd TBD
Intra Day Deployments	☑

Let them eat Cake

- Cake is C# make – Common build/deploy platform
- Runs on Win / Linux and OS X
- Its CI system agnostic – ideal if you plan to migrate…
- Has basic OOB functions
- Can be extended by Open Source Addins – MsDeploy, CakeAzure….
- Or C# std libraries
- Great for tying things together

Cake Disclaimer…..

- The Addins can be immature
  - Plenty of Bugs
  - Low on functionality
- Don’t use Relative Paths – can pass in as args but transform to absolute inside script
- There is a compile time penalty each invocation
- Calling powershell scripts from cake is fraught with danger – return codes, errors …. Beware
- However using cake instead on PShell brought deployment times down considerably

Tip Time - Webjobs

- Make them start up faster
  - "WEBJOBS_RESTART_TIME": "5", -default is 30 secs
  - Use webjob.settings for granular control or over ride in appsettings
- Deploy them as singletons
  - Change settings.job -> { "is_singleton": true }
- Stop and start them individually using API
  - scm.azurewebsites.net/api/continuouswebjobs/{jobname}/stop
  - Great when controlling deployments
- If running on multi instance App Service Plans
  - They fall over
  - They don’t all pin to one instance – round robin deploy behaviour

Tip Time – WebApps

- Always use ARM (Yes its obvious but….)
  - DR provisioning is so easy
  - No longer afraid to tear it up and start again
  - dependsOn – be careful ordering these
- Use awverify in your DNS records to test new azurewebsites
- Trick to adding non SSL Certs
  - Add the Cert then remove the SSL Binding
- The file system is shared between instances
  - Avoid using it
  - If you MUST then prevent file deadlocks by using uniquenames…

More Time for Tips??

- What deploy technique?
  - REST, Pshell, Azure CLI, ARM
  - Know resources.azure.com
  - Invoke-AzureRmResourceAction
    - Powershell wrapper to REST API
- Organise Resources in groups
  - Cost consolidation, Convenience
- Any Others ?

Take away: PaaS **can** be **flexible**. But better to build/design the system to be platform agnostic.

Questions ?

Thank you | readify.net