Approach to Multi cloud
Myths, Reality, and way forward

Uli Homann
Corp. VP & Distinguished Architect
Cloud + AI engineering
Microsoft Corp.
Customer environments are evolving

100’s–1,000’s of apps
- VMs
- Containers
- Databases
- Serverless

Diverse infrastructure
- Datacenters
- Branch offices
- IoT devices
- Hosters
- OEM hardware
- Edge

Multi-cloud
- Microsoft Azure
- AWS
- Google Cloud
Motivation

Enterprise customer:
Risk management – cloud dependencies are technology + hosting dependencies and therefore more impacting than either one alone

Regulatory requirement – ECB for example requires banks to be able to move out of a specific environment within a reasonable amount of time

ISV’s: maximization of market impact
Definitions

Multi cloud or poly cloud: deploy solution/components across multiple clouds

Cloud agnostic: develop solutions that could potentially run on multiple cloud environments
Approaches to consider

- CLOUD-NATIVE
- CLOUD-CONSISTENT
- CLOUD-CONNECTED
Key questions

What is your approach to multi-cloud?

Are you looking at cloud agnostic asset development?

What is the operations model you are looking for?

Which abstractions are you comfortable with?

What kind of dependencies are ok with you?
Key themes

Cloud agnostic assets you care about are developer focused:

- Code
- Models
- Single process like CI/CD targeting multiple environments
- Key developer abstractions need to be multi-cloud (database, events, messages, etc.)

Pick a center of gravity for operations:

- Management
- Identity
- Security
- Logs
- Etc.
Three ways to manage dependencies

- OSS
- Technology is multi-cloud
- Consumer/Provider pattern/abstraction
Myths of Cloud agnostic applications
If I only use the most basic cloud services, my application is portable.
More portability

$$$$$

Standards

Less portability

$$$

$

Standards

How much are you spending “below the value line” in this scenario?

Image credit: Flickr, pshutterbug, CC 2.0
If it’s open source, I can port it wherever I need to
Along the spectrum

Innovation

Discoveries

Community

De facto standards

De jure standards

Standards

Photo Credit: © Jorge Royan / http://www.royan.com.ar / CC BY-SA 3.0
Cloud portability is at odds with Developer Choice

Data Management

- Document database
  - MongoDB
  - CouchDB

- Cache
  - GemFire
  - Redis

- SQL Database
  - SQL Server
  - PostgreSQL
Every Additional Developer Choice You Support Adds Proportionate Ops Overhead
API/Protocol

Engine
MongoDB

CosmosDB
How do I build applications in this world?
CNCF is the foundation

Containers
Kubernetes
Helm
SMI
Cloud Events

Cloud Events
Add -
A programming model
What is a programming model?

• The way developers write their application that interacts with other services and data stores.

• Increasingly polyglot, with microservice architecture
Dapr: Build apps using any language with any framework

Microservice application

Services written in:
- GO
- node
- python
- .NET Core
- Java
- Functions

HTTP/gRPC APIs
- Service-to-service invocation
- State management
- Publish and subscribe
- Resource bindings & triggers
- Actors
- Distributed tracing
- Extensible...

Any code or framework...

Any cloud or edge infrastructure
What about gaps such as flow architecture?
PaaS vNext approach

Azure PaaS vNext

API Management
- Web App
- Function App
- Logic App
- Container App
- Inference app

Integration
- Events
- Messaging

Azure Serverless
- Kubernetes Distro

Azure Resource Manager + Azure Arc

Hosting
- Azure
- On-premises
- IoT Edge
- Multi-cloud

App Developer
- VS Code
- GitHub
- CNAB

IT Pro
- Azure Policy
Arc-enabled developer services

- Primitives for specific use case
- Consistency across developer and operator lifecycle
- Integrates with common services

**Primitives**
- Azure
- Kubernetes

**Integration**
- dapr

**Govern**
- Policy

**Manage**
- ARM Clients (Portal, CLI)

**Deploy**
- GitHub Actions

**APIs**
- GitHub Accelerators

**Develop**
- Visual Studio / Code

**Monitor**
- Application Insights
- Log Analytics
How do I operate these clouds then?
End-to-end multi-cloud management (not exhaustive)

- ITSM
- Infrastructure as Code*
- Resource Management
- Extension to on-premises resources
- Monitoring
- Log Analytics
- SIEM
- Privacy & Identity

* consider using a formal source code management environment for your infrastructure as code repository
Some open community initiatives
Today: CSP’s all provide semantically equivalent elements, but elbow grease is required to get them into a common format suitable for downstream processing.

Common Information Model: We are asking CSP’s provide alerts using an open and common set of elements that will remove ambiguity and allow enterprise Cloud customers to process security alerts from all CSP’s using a common data pipeline.

Decorator: Alerts received through batch or stream processing will then be decorated or ‘enriched’ providing the the customer with the ability to provide additional context in order to speed decision making and automation across large data streams.

Decorated Alert Consumption: Decorated alerts are received and processed by the Cloud customer to answer questions and report on trends that are most important to them. Deep insights can be applied to identify anomalous activity, find indicators of compromise, or update risk scores across all Cloud provider workloads.
Delivering upon operations – current options:

- Build your own – service now, Splunk, data dog, etc.

<table>
<thead>
<tr>
<th>Vendor dependency:</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMWare Tanzu Mission control</td>
</tr>
<tr>
<td>IBM Satellite</td>
</tr>
<tr>
<td>Google Anthos</td>
</tr>
<tr>
<td>Azure Arc</td>
</tr>
</tbody>
</table>
Single control plane with **Azure Arc**

- **Azure Arc enabled infrastructure**
  - Connect and operate hybrid resources as native Azure resources

- **Azure Arc enabled services**
  - Deploy and run Azure services outside of Azure while still operating it from Azure

- Icon for **Multi-cloud**, **Datacenter**, and **Edge**
End-to-end management approach
Proven practices (few but important)

“Static” configuration such as VPC/VNET or similar structures should be cloud-native per environment using pre-designed IP per cloud environment

Daily operations tools need to be able to cover multi-cloud scenarios

# End-to-end multi-cloud management

<table>
<thead>
<tr>
<th>ITSM</th>
<th>ServiceNow or equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure as Code*</td>
<td>TerraForm or equivalent For DevOps scenarios consider using technologies such as GitHub Actions</td>
</tr>
<tr>
<td>Resource Management</td>
<td>Cloud-native resource management API, e.g. ARM for Azure</td>
</tr>
<tr>
<td>Extension to on-premises resources</td>
<td>Azure Arc or equivalent</td>
</tr>
</tbody>
</table>
| Monitoring     | De-Facto standards such as Prometheus
|                | Azure Monitor (x-platform | x-environment)                      |
|                | DataDog or equivalent                                                                   |
| Log Analytics  | Splunk or similar
|                | Azure LogAnalytics (x-platform | x-environment)                      |
| SIEM           | Qradar and equivalent (not built for cloud)
|                | Azure Sentinel (x-platform | x-environment)                      |
| Privacy & Identity | Standards (Oauth, etc.)
|                | Azure Active Directory, Okta, or equivalent                                              |

*consider using a formal source code management environment for your infrastructure as code repository
Things to worry about...

- Testing for compatibility across cloud providers
- Management
- Billing
- Costs
- Skills across cloud