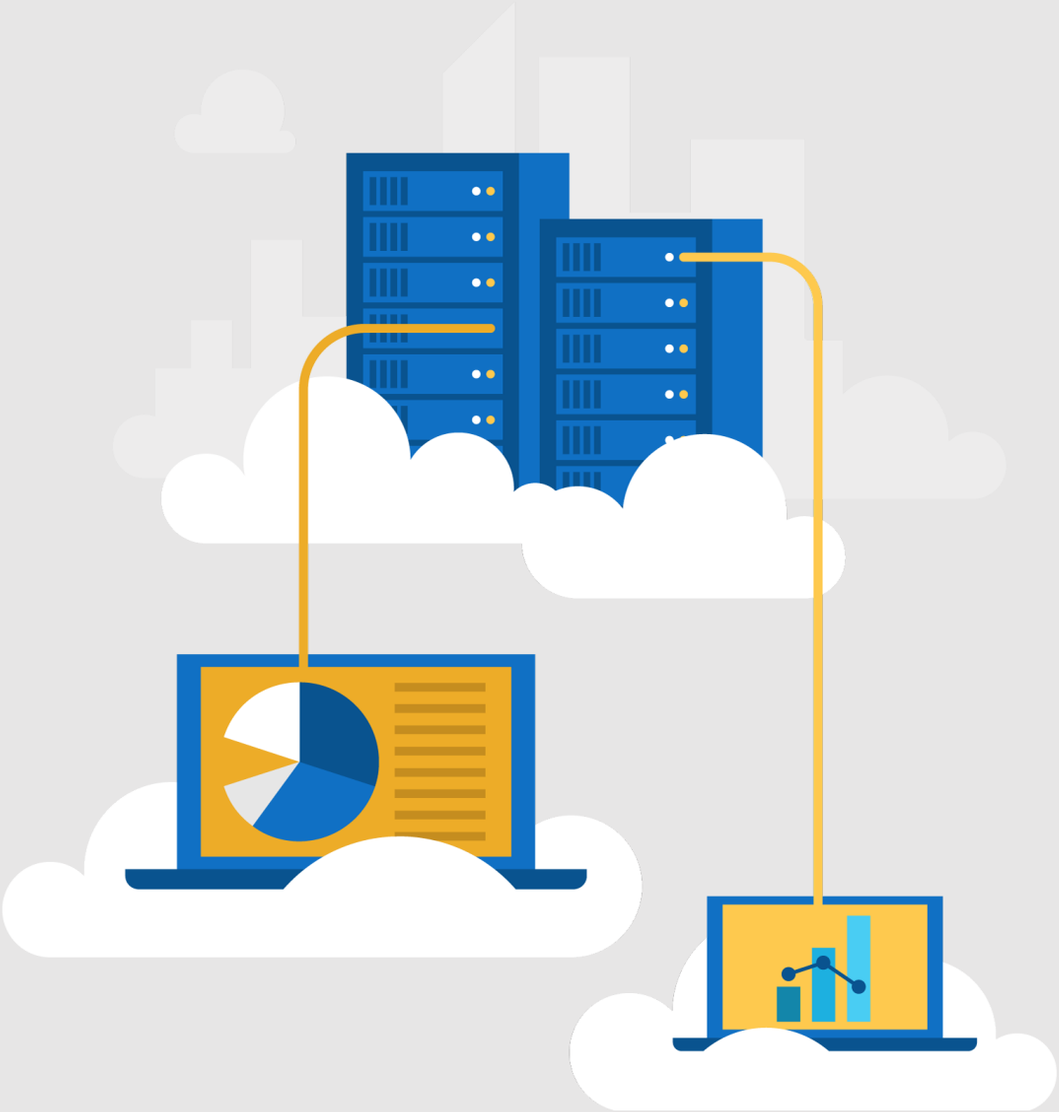
Microsoft

Azure Study Group

AZ-301 - Microsoft Azure Architect Design

Jeff Wagner
Partner Technology Strategist



Agenda

1

Agenda

2

Speaker Introduction

3

Feedback Loop

4

Objective Review

5

Open Mic

Series Agenda

1	Determine Workload Requirements (10-15%)
2	Design for Identity and Security (20-25%)
3	Design a Data Platform Solution (15-20%)
4	Design a Business Continuity Strategy (15-20%)
5	Design for Deployment, Migration, and Integration (10-15%)
6	Design an Infrastructure Strategy (15-20%)

https://aka.ms/azurecsg

Series Agenda

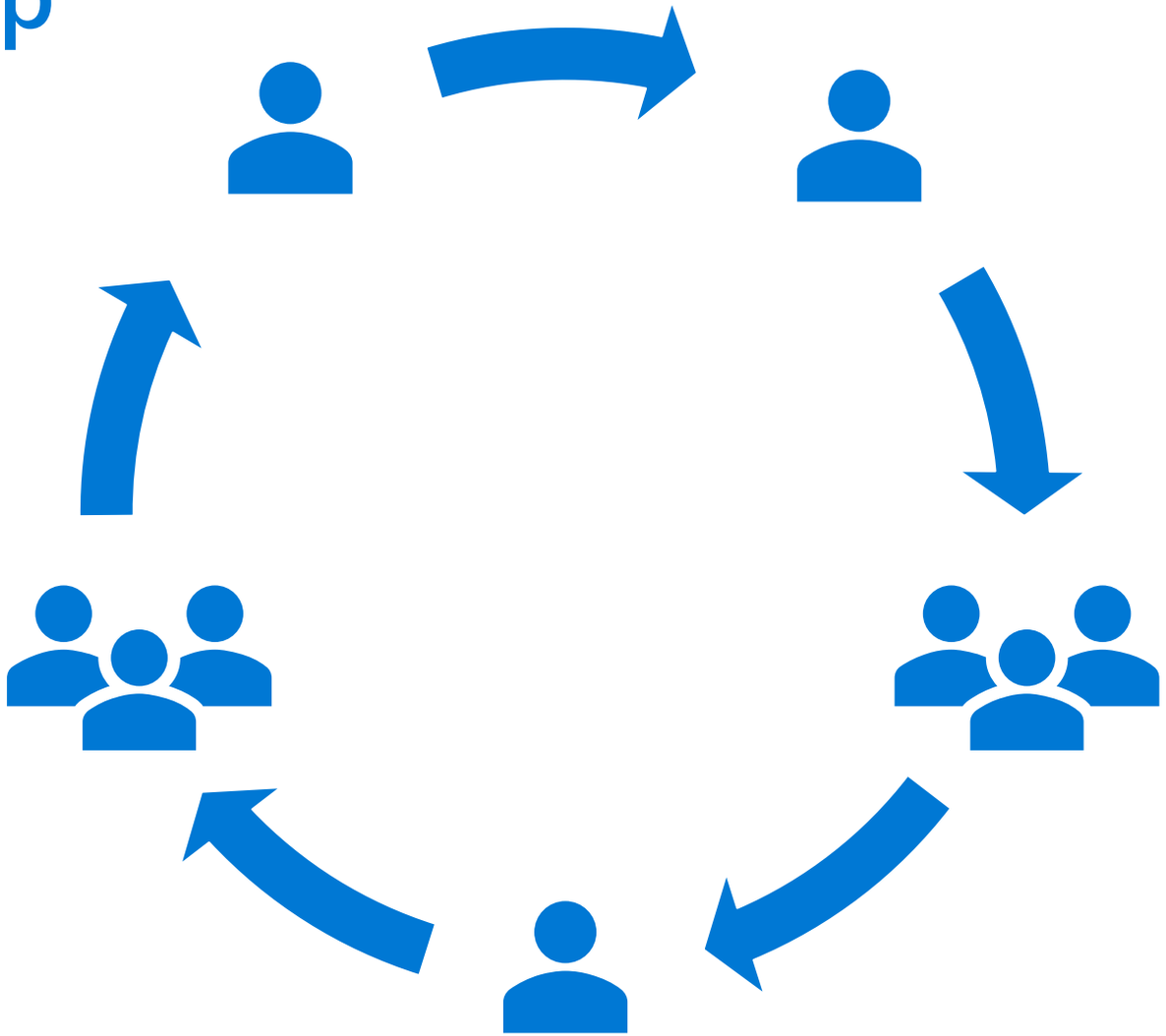
1	Determine Workload Requirements (10-15%)
2	Design for Identity and Security (20-25%)
3	Design a Data Platform Solution (15-20%)
4	Design a Business Continuity Strategy (15-20%)
5	Design for Deployment, Migration, and Integration (10-15%)
6	Design an Infrastructure Strategy (15-20%)

https://aka.ms/azurecsg

Speaker Introduction - Jeff Wagner

- Partner Technology Strategist based in Atlanta
- 21+ years with Microsoft, more in the industry
- Constant learner - *Ancora Imparo*
- Working on the same certifications that you are

Feedback Loop



Objectives

Design Deployments

May include but not limited to: Design a compute, container, data platform, messaging solution, storage, and web app and service deployment strategy

Design Migrations

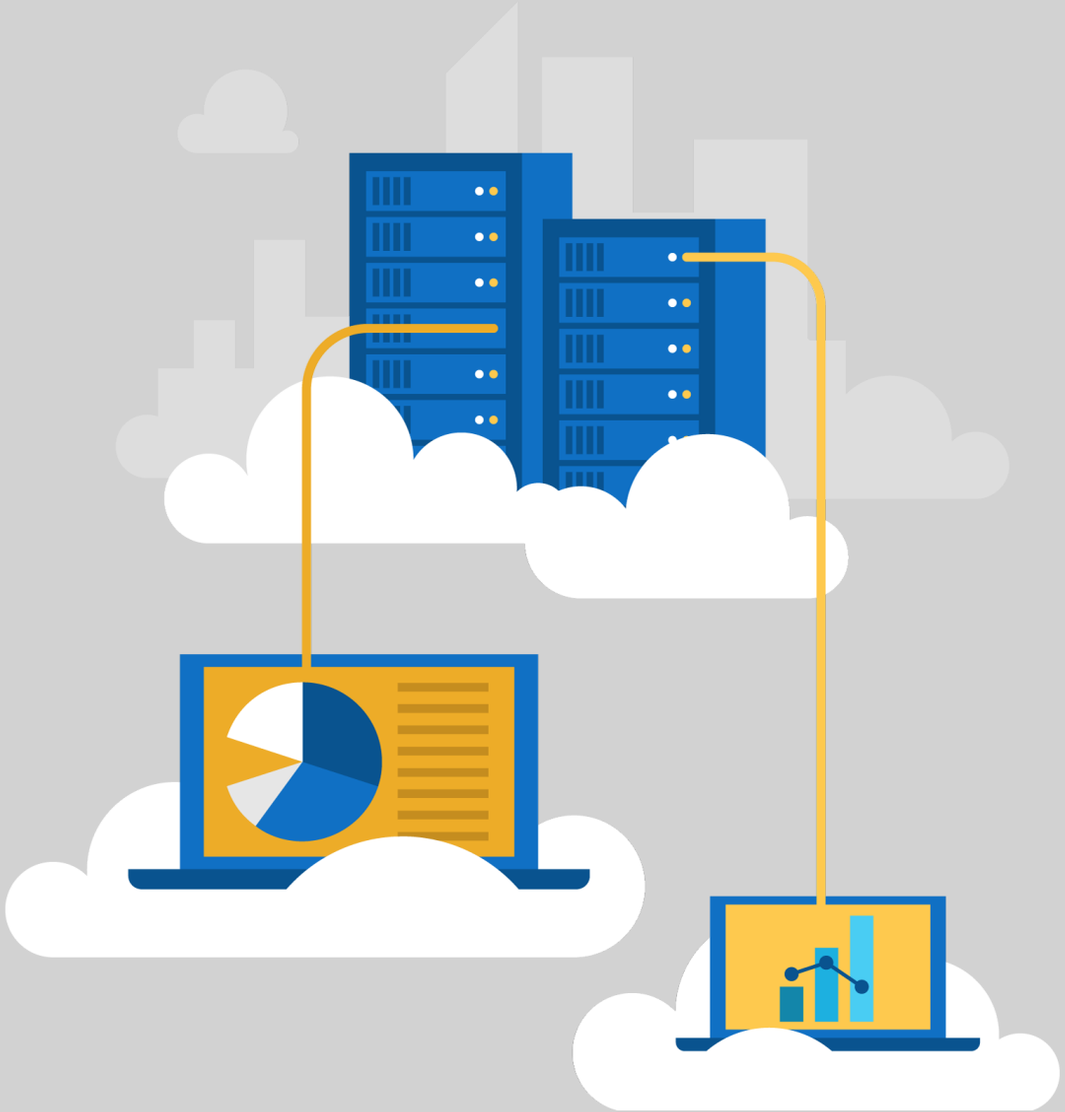
May include but not limited to: Recommend a migration strategy; design data import/export strategies during migration; determine the appropriate application migration, data transfer, and network connectivity method; determine migration scope, including redundant, related, trivial, and outdated data; determine application and data compatibility

Design an API Integration Strategy

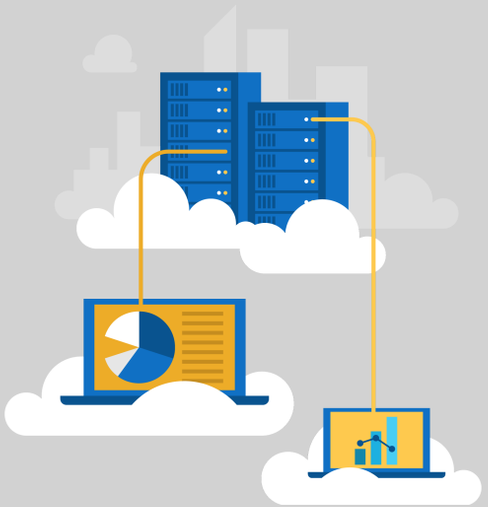
May include but not limited to: Design an API gateway strategy; determine policies for internal and external consumption of APIs; recommend a hosting structure for API management

Microsoft

Design Deployments



ARM Templates



Azure Resource Manager

- Objectives:
  - **Provide consistent deployment and management model in Azure and Azure Stack**
  - **Support grouping of resources**
  - **Facilitate Infrastructure as Code**
  - **Implement resource providers**
  - **Encourage DevOps practices**

Azure Resource Manager Objects

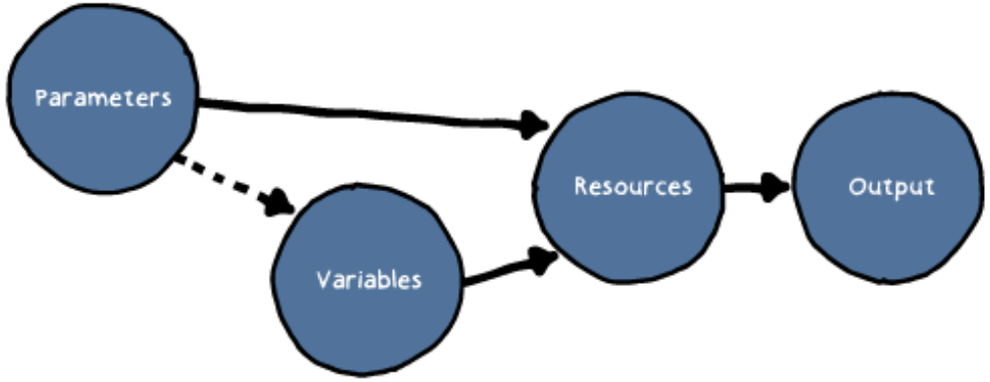
- An Azure Resource Manager resource:
  - **An manageable instance of an Azure service or component of an Azure service (e.g. Azure Web App, an App Service plan, or Azure SQL Database)**
- A resource group:
  - **Logical grouping of Azure Resource Manager resources**
- Resource group templates:
  - **JSON-based representation of Azure Resource Manager resources that can be deployed together into a resource group**

Interacting with Azure Resource Manager

- Azure PowerShell
- Azure CLI
- Client libraries
- Visual Studio and Visual Studio Code
- the Azure portal
- REST API

ARM Templates

- A JSON-based representation of Resource Manager resources:
  - **Allows for deployment via:**
    - **Visual Studio**
    - **Azure PowerShell**
    - **Azure CLI**
    - **the Azure portal**
  - **Facilitates idempotency**
  - **Simplifies orchestration**
  - **Allows to express resource dependencies**
  - **Promotes reuse through parameters and nesting**

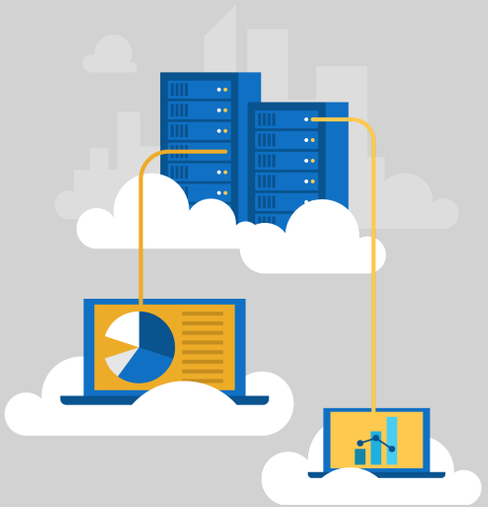


JSON

- JavaScript Object Notification (JSON):
  - **A lightweight format for transmitting data objects**
  - **Used in Azure Resource Manager templates**
- Azure Resource Manager template format:
  - **$schema (mandatory)**
  - **contentVersion (mandatory)**
  - **parameters (optional)**
  - **variables (optional)**
  - **functions (optional)**
  - **resources (mandatory)**
  - **outputs (optional)**

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "variables": {
  },
  "resources": [
  ],
  "outputs": {
  }
}
```

Role-Based Access Control (RBAC)



Role-Based Access Control

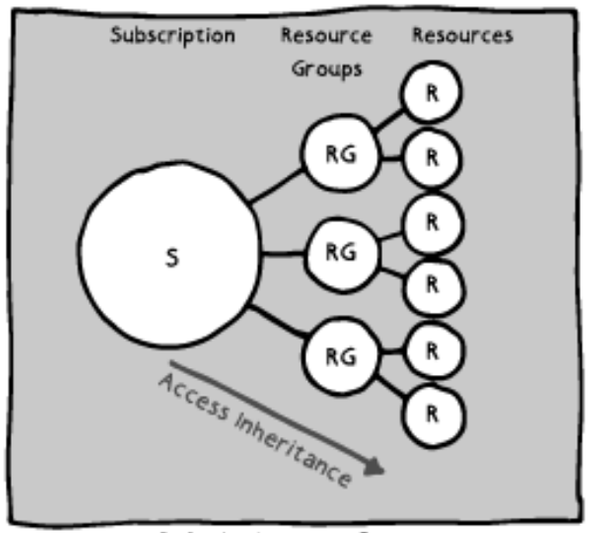
- An administration delegation mechanism that associates:
  - **A role**
  - **An Azure Active Directory identity**
  - **An Azure subscription, resource group, or resource**
- A role represents a set of permissions to carry out specific actions:
  - **Built-in, not resource specific (Owner, Contributor, Reader, User Access Administrator)**
  - **Built-in, resource specific (e.g. Virtual Machine Contributor)**
  - **Custom**
-

Role Assignment

- Role assignment associates a role with identities that exist in the Azure AD tenant associated with the Azure subscription:
  - **Users: include Microsoft and guest accounts that exist in the Azure AD tenant)**
  - **Service principals: represent Azure AD applications and Azure resources)**
  - **Groups: can contain users, service principals, and groups. Group-based assignments are recommended due to lower management overhead (in comparison with direct assignments to users and service principals).**

Resource Scope

- The role assignment can be scoped to:
  - **the subscription**
  - **a resource group**
  - **a resource**
- RBAC-based permissions propagate from a parent to children:
  - **It is possible to grant additional permissions by using child-level assignment**
  - **It is not possible to deny inherited permissions by using child-level assignments**



Role Assignment Scopes

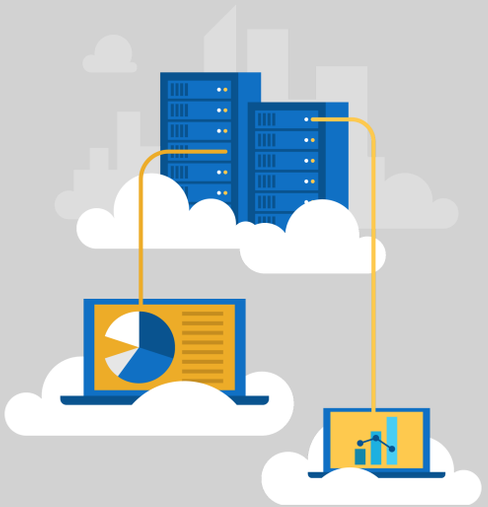
Custom Roles

- Complement built-in roles
- Can be provisioned by using:
  - **Azure PowerShell**
  - **Azure CLI**
  - **REST API**
- To create a custom role (by using Azure PowerShell):
  - **Create a JSON file containing role definition**
  - **Run New-AzureRmRoleDefinition with the InputFile parameter that references the role definition file**

```
{
  "Name": "New Role 1",
  "Id": null,
  "IsCustom": true,
  "Description": "Allows for read access to Azure storage and compute
resources",
  "Actions": [
    "Microsoft.Compute/*/read",
    "Microsoft.Storage/*/read",

  ],
  "NotActions": [
  ],
  "AssignableScopes": [
    "/subscriptions/c489345-9cd4-44c9-99a7-4gh6575315336g"
 ]
}
```

Resource Policies



Lesson Objectives

- Describe the purpose and basic components of Azure policies.
- Explain the differences between Azure policy and RBAC.
- List most common built-in policies.
- Describe the format of a policy definition.
- Describe the process of assigning policies.
- Provide an example of a policy that enforces a naming convention.

Azure Resource Policies

- Enhances governance and compliance by controlling settings of Azure resources, e.g.:
  - **Resource types**
  - **Location**
  - **Size**
  - **Naming convention**
- To implement policies, you need:
  - **A policy definition**
  - **A policy assignment**

Azure Policies vs RBAC

- RBAC:
  - **Grants Azure AD identities permissions to carry out actions on Azure resources**
  - **Can be scoped to management group, subscription, resource group, or resource**
  - **Default deny, explicit allow**
- Azure policies:
  - **Compares policy conditions to properties of a resource during or after its deployment**
  - **If the comparison is successful, automatically applies the 'effect' defined in the policy**
  - **Can be scoped to management group, subscription, resource group, or resource**
  - **Default allow, explicit deny**
- To implement policies, it is required to have two RBAC permissions:
  - **Microsoft.Authorization/policydefinitions/write permission to define a policy**
  - **Microsoft.Authorization/policyassignments/write permission to assign a policy**

Built-in Policies

- Azure offers a library of built-in policy definitions, including:
  - **Allowed locations**
  - **Allowed resource types**
  - **Allowed storage account SKUs**
  - **Allowed virtual machine SKUs**
  - **Apply tag and default value**
  - **Enforce tag and value**
  - **Not allowed resource types**
  - **Require SQL Server version 12.0**
  - **Require storage account encryption**

Policy Definitions

- A policy definition is a JSON-formatted file consisting of:
  - **mode**
  - **parameters**
  - **display name**
  - **description**
  - **policy rule**
  - **logical evaluation**
  - **effect:**
    - **Deny**
    - **Audit**
    - **Append**
    - **AuditIfNotExists**
    - **DeployIfNotExists**

policy not allowed.json

```
1 {
2   "if": {
3     "field": "type",
4     "in": "[parameters('listOfResourceTypesNotAllowed')]"
5   },
6   "then": {
7     "effect": "Deny"
8   }
9 }
```

Policy Assignments

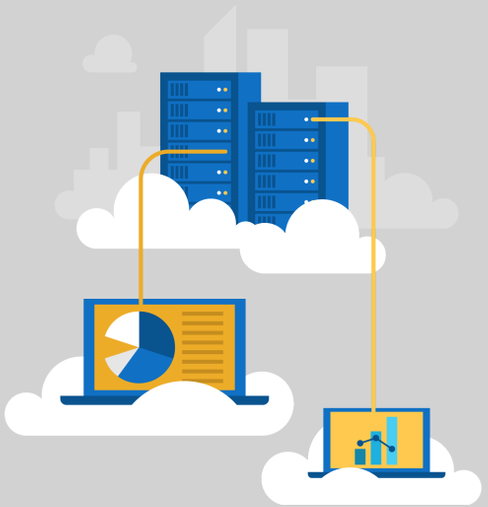
- To take effect, a policy definition must be assigned.
- An assignment consists of:
  - **A policy definition**
  - **Values of parameters of the policy definition**
  - **A scope (a management group, subscription, resource group, or resource)**
  - **Zero or more exclusions (by default, policy assignments at a higher level are inherited by child all resources)**
- You can combine multiple policy definitions into policy initiatives:
  - **Policy initiatives simplify policy management and maintenance**
  - **Assigning policy initiatives follow the same rules as policy assignments**

Policies for Naming Conventions

- A policy definition to enforce naming conventions:
  - **Relies on pattern matching and wildcards**
  - **Causes deployment failure of non-compliant resources**
  - **Reports presence of existing non-compliant resources**

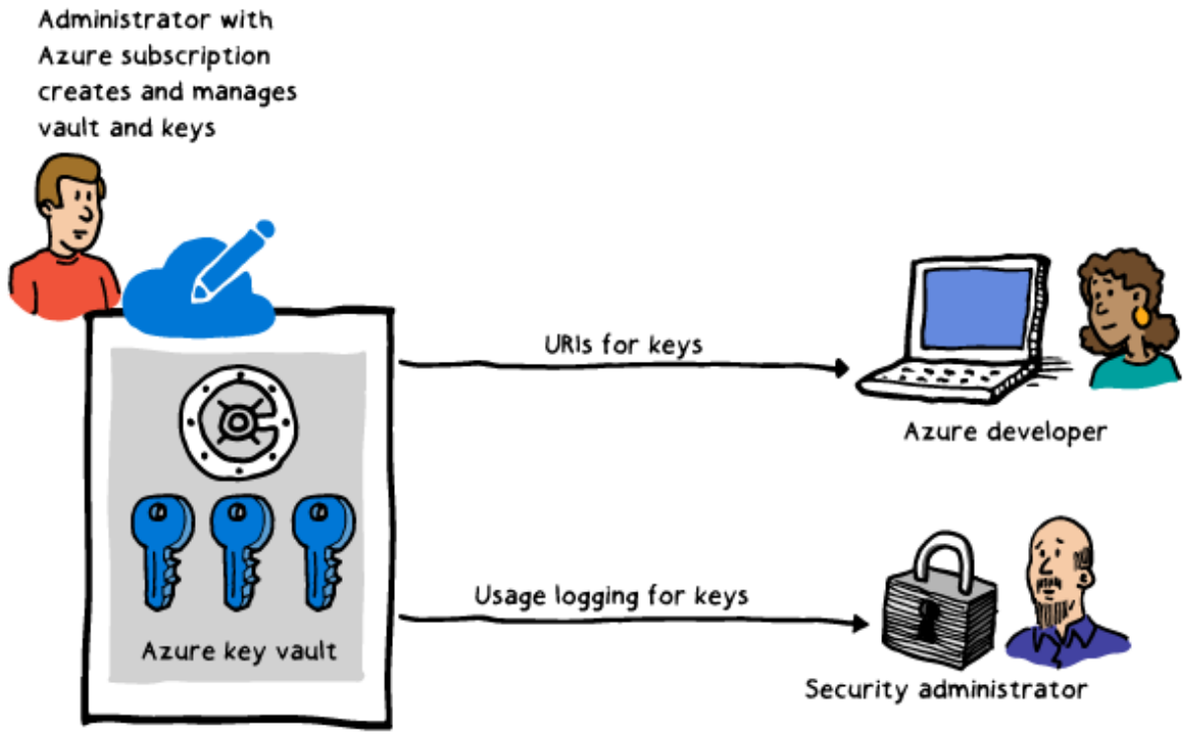
```
{
  "if": {
    "not": {
      "field": "name",
      "match": "contoso??????"
    }
  },
  "then": {
    "effect": "deny"
  }
}
```

Security



Azure Key Vault

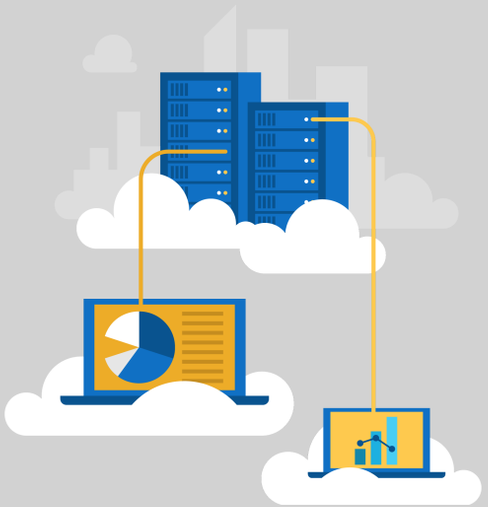
- An Azure service that uses an HSM to store:
  - **Cryptographic keys**
  - **Certificates**
  - **Secrets**
- Supports role-based separation:
  - **Azure Administrators:**
    - **Manage Key Vault secrets, keys, and certificates**
  - **Developers:**
    - **Reference secrets, keys, and certificates**
  - **Security administrators:**
    - **Monitor usage logs**
    -



Azure Key Vault in ARM Templates

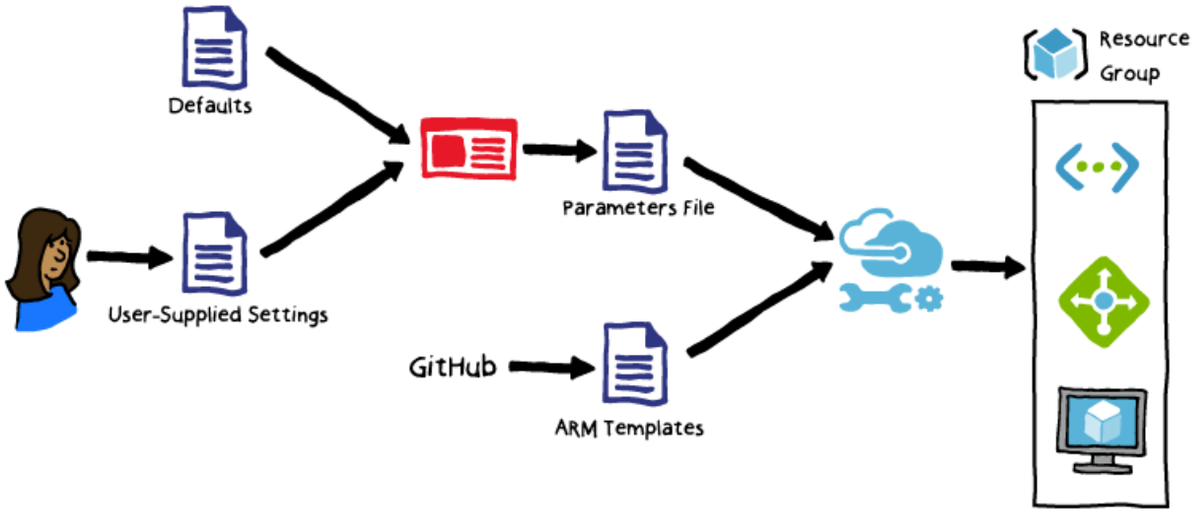
- Protects secrets used during deployment:
  - **Secrets are retrieved automatically during deployment**
  - **An Azure AD identity used to perform deployment does not need permission to access to the secrets directly**
- To implement:
  - **Set the Key Vault's enabledForTemplateDeployment advanced policy to True**
  - **Use RBAC to grant the Azure AD identity used to perform deployment the Microsoft.KeyVault/vaults/deploy/action permission to the Key Vault**
  - **Reference the Key Vault and the secrets in the ARM template**

Building Blocks



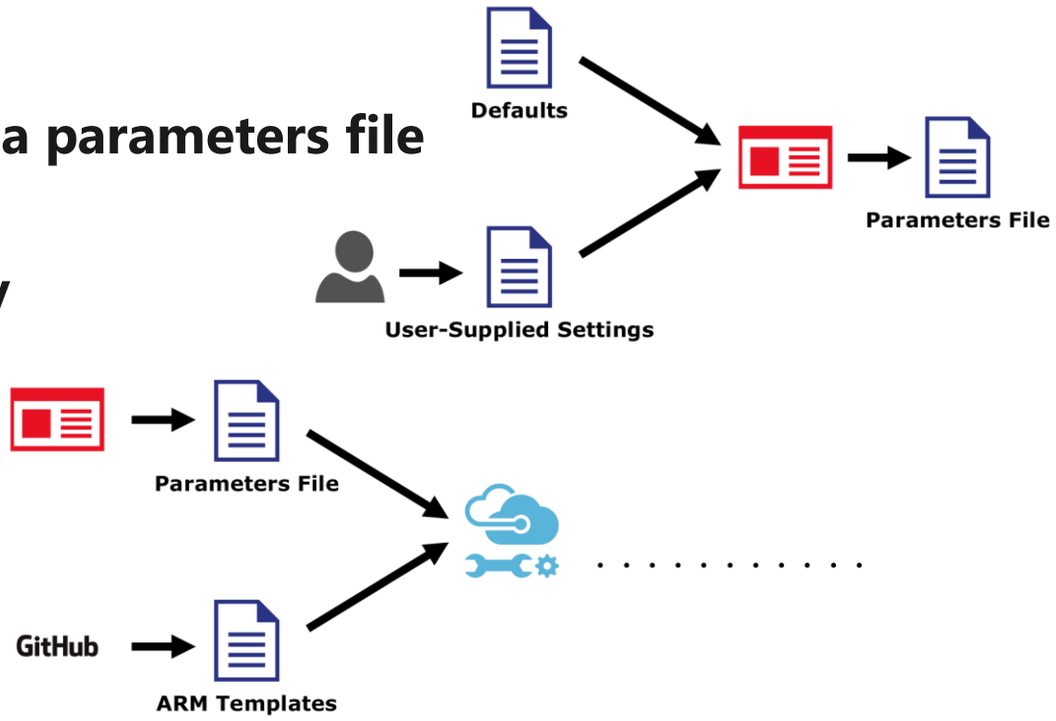
Azure Building Blocks

- GitHub-based tool and repository for ARM template authoring and deployment:
  - **Minimizes effort required to author ARM deployments**
  - **Implements Microsoft's Patterns & Practices**
  - **Uses a dedicated command-line tool (azbb)**
  - **Relies on Building Blocks JSON schema**
  - **Supports user-supplied setting files**
  - **Available for provisioning:**
    - **Virtual Networks**
    - **Virtual Machines**
    - **Load Balancers**
    - **Route Tables**
    - **Network Security Groups**
    - **Virtual Network Gateways**
    - **Virtual Network Connection**



Deploying Resources Using Building Blocks

- Install Azure CLI and Azure Building Blocks (including azbb)
- Create a settings file containing custom parameters
- Run azbb command line tool, which:
  - **Merges defaults with user-supplied settings into a parameters file**
  - **Retrieves ARM templates from GitHub repository**
  - **Initiates deployment using the templates and the parameters file**



Building Block Resource

- Each building block is represented as a JSON object of a specific type:
- E.g. a VirtualNetwork Building Block type has the following properties:
  - **name: the resource identifier**
  - **addressPrefixes: an array containing one or more IP address spaces in CIDR notation**
  - **subnets: named IP address ranges within the IP address space of the virtual network**

Settings File

- A sample VirtualNetwork building block settings file

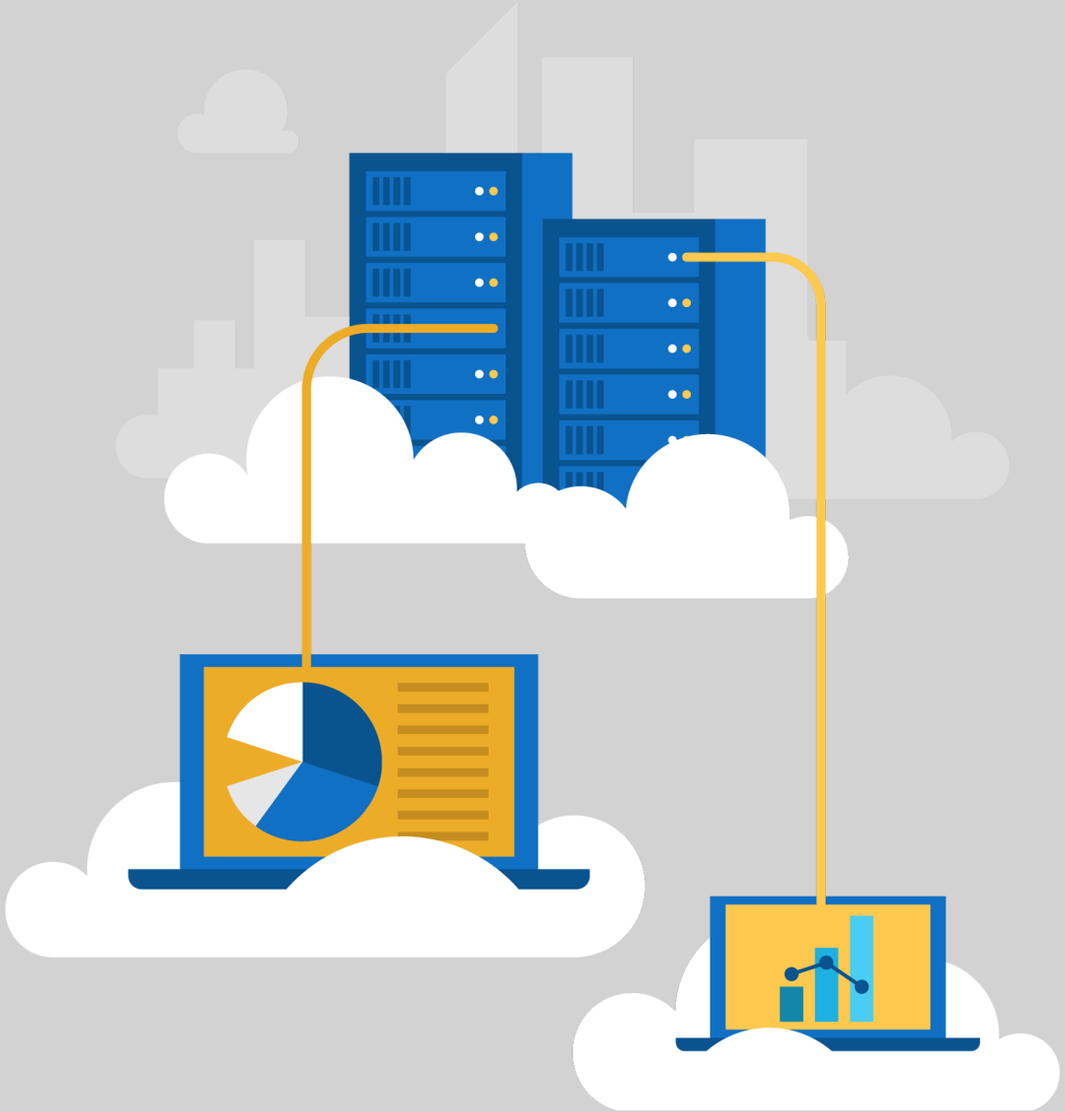
```
{
  "$schema": "https://raw.githubusercontent.com/mspnp/template-building-
blocks/master/schemas/buildingBlocks.json",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "buildingBlocks": {
      "value": [
        {
          "type": "VirtualNetwork",
          "settings": [
            {
              "name": "msft-hub-vnet",
              "addressPrefixes": [
                "10.0.0.0/16"
              ],
              "subnets": [
                {
                  "name": "firewall",
                  "addressPrefix": "10.0.1.0/24"
                }
              ]
            }
          ]
        }
      ]
    }
  }
}
```

Deploying a Settings File

- Authenticate to your Azure AD tenant:
  - **az login**
- Identify subscription ID of your Azure subscription:
  - **az account list**
- Decide whether to deploy into a new or existing resource group
- Choose an Azure region where you want to perform deployment
- Create a deployment:
  - **azbb -g <new or existing resource group> -s <subscription ID> -l <region> -p <path to your settings file> --deploy**

Microsoft

Design Migrations

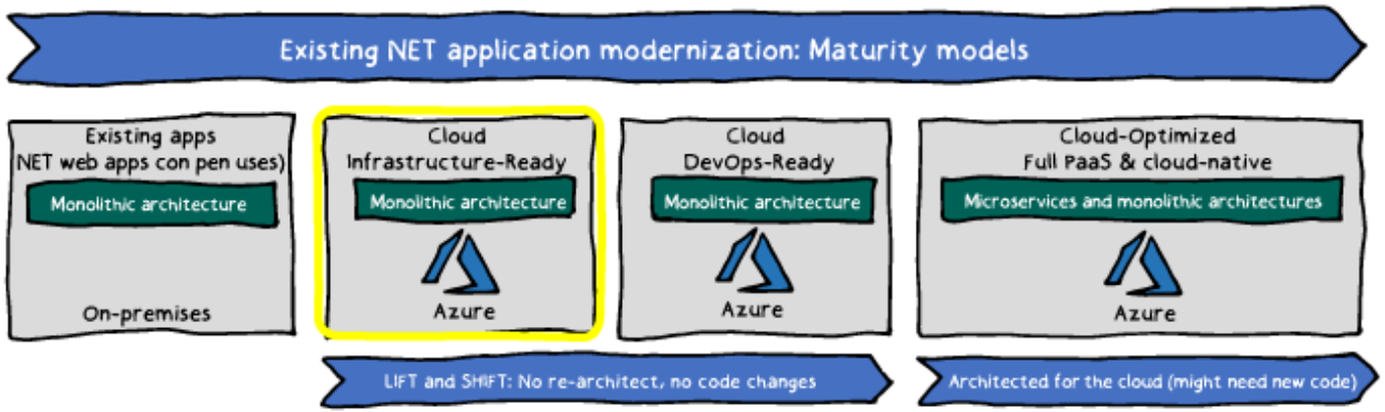


Cloud migration & modernization strategies

	Rehost	Refactor	Rearchitect	Rebuild
Description	**Redeploy as-is to cloud**	**Minimally alter to take better advantage of cloud**	**Materially alter/decompose application to services**	**New code written with cloud native approach**
Drivers	• Reduce Capex • Free up datacenter space • Quick cloud ROI	• Faster, shorter, updates • Code portability • Greater cloud efficiency (resources, speed, cost)	• App scale and agility • Easier adoption of new cloud capabilities • Mix technology stacks	• Accelerate innovation • Build apps faster • Reduce operational cost
Technologies	**IaaS**	**Containers** **PaaS**	**PaaS** **Serverless** **Microservices**	

On-Premises Lift and Shift

- The most common and simplest migration approach:
  - **Delivers the benefits of the pay-as-you-go pricing model**
  - **Eliminates the need to re-architect or re-write workloads**
- Azure Migrate:
  - **A discovery and assessment tool from Microsoft for lift and shift migrations**
  - **Identifies VM sizing requirements**
  - **Accounts for VM dependencies**
  - **Assesses migration suitability**
  - **Provide cost estimates**
  - **Currently VMware only**
  - **Soon to include Hyper-V**

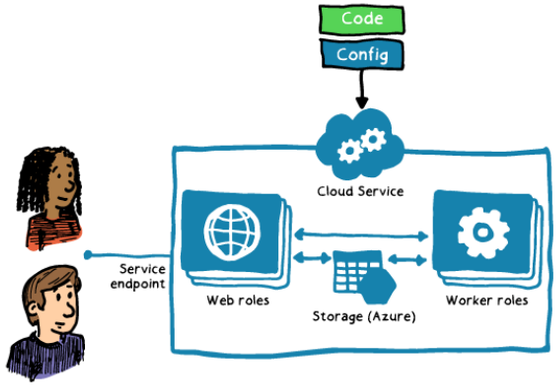


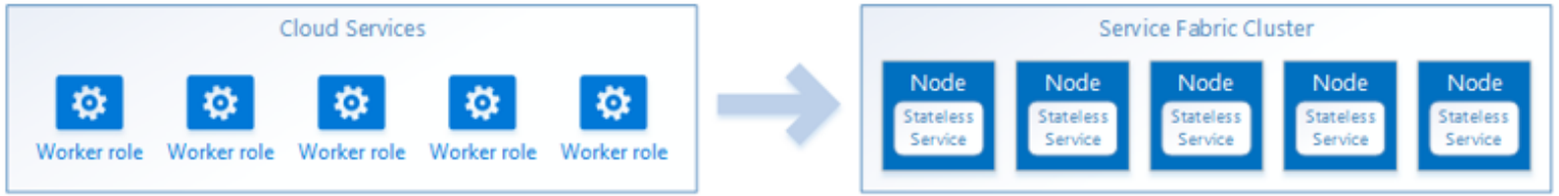
Migration from Classic IaaS

- Majority of classic resources can be migrated to the Azure Resource Manager deployment model, including:
  - **Virtual Machines**
  - **Availability Sets**
  - **Cloud Services**
  - **Storage Accounts**
  - **Virtual Networks**
  - **VPN Gateways**
  - **Express Route gateways**
  - **Network Security Groups**
  - **Route Tables**
  - **Reserved IPs**

Migration from Cloud Services

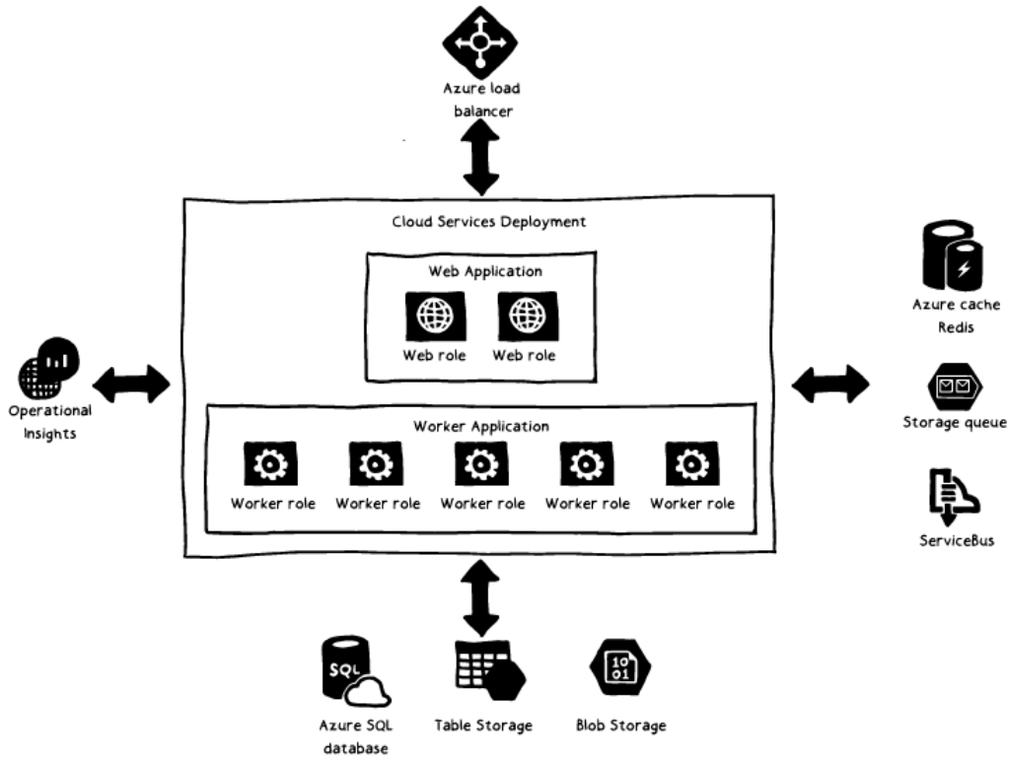
- Azure Cloud Service does not support direct migration path:
  - **A classic resource consisting typically of one or more web roles and worker roles**
  - **Implemented in code by using deployment packages**
- The recommended migration path:
  - **Use a Service Fabric cluster to provide compute**
  - **Convert roles into microservices**





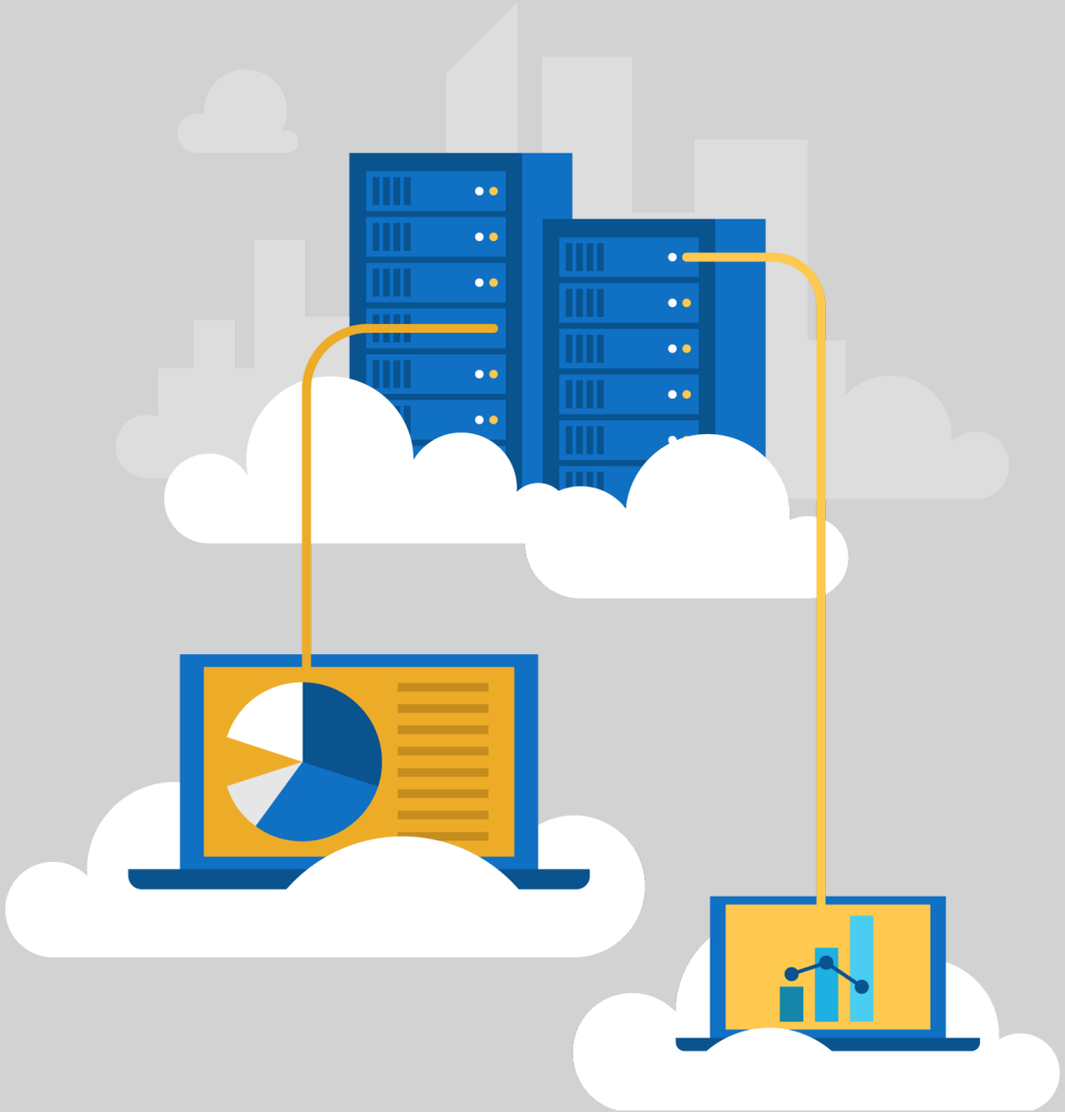
Deployment package

- Migrating a cloud service to Service Fabric:
  - **Hides the compute layer from the application hosted by the cloud service**
  - **Allows multiple applications to be deployed to the same cluster**
  - **Alters communication methods of components running on the web and worker roles**
- Migration methods:
  - **Quick and easy (limited benefits):**
    - **Convert web and worker roles to stateless services**
    - **Preserve existing application architecture**
    - **Retain external dependencies**
  - **Modernization (full benefits):**
    - **Refactor web and worker roles into microservices**
    - **Use a combination of stateful and stateless services**
    - **Minimize or eliminate external dependencies**

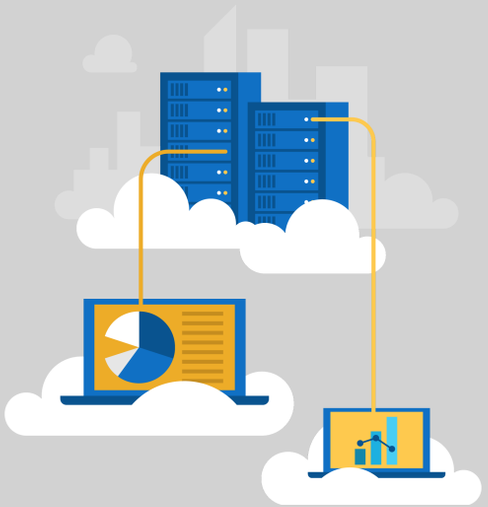


Microsoft

Design an API Integration Strategy



Infrastructure-Backed Platform-as-a-Service (PaaS)



Infrastructure-Backed PaaS

- Intended for highly scalable, isolated workloads:
  - **App Service Environments: hosting web apps, API apps, and mobile apps**
  - **Azure Service Fabric: hosting microservices and containers**
  - **Azure Container Service: hosting and orchestrating containerized workloads**

App Service Environments

- An Azure App Service offering that provides highest degree of control:
  - **Deployments into a virtual network**
  - **Support for Network Security Groups**
  - **Ability to control outbound requests (for PCI compliance)**
  - **Increased scalability (beyond limits of a regular App Service plan)**
- Available in two versions:
  - **ASE v1 – supports both Classic and Azure Resource Manager deployment models**
  - **ASE v2 – supports Azure Resource Manager deployment model only, but offers a number of benefits over ASE v1:**
    - **Enhanced automation**
    - **Increased scalability**

Azure Service Fabric

- A distributed system solution for deployment and management of containers and microservices:
  - **Comprises of a cluster of VMs that host containers with microservices.**
  - **Can host stateless and stateful microservices (individual software components that handle specific function within an application, such as queuing, caching, or operating as a customer's shopping cart).**
  - **Handles provisioning, monitoring, and management of microservices.**
  - **Powers many existing Microsoft cloud services (including Azure SQL Database, Azure Cosmos DB, Cortana, Power BI, Microsoft Intune, Azure Event Hubs, Azure IoT Hub, and Dynamics 365).**
  - **Integrates with DevOps tools, such as Visual Studio Team Services, Jenkins, and Octopus Deploy.**

Azure Container Service

- Implements managed cluster of VMs running containerized workloads:
  - **Offers the choice of an open-source orchestrator:**
    - **Docker Swarm**
    - **Mesosphere DC/OS**
    - **Kubernetes**
  - **Supports Docker for container portability**
  - **Scales to 10,000s of containers**
- Azure Kubernetes Service (AKS):
  - **The latest orchestrator offering**
  - **A fully managed Kubernetes cluster**
    - **Azure handles management of cluster infrastructure**
    - **Kubernetes handles management of containerized applications**
  - **Variety of management features (including automatic binpacking, self-healing, horizontal scaling, service discovery, load balancing, automated rollouts and rollbacks)**

Questions?

Microsoft

Homework Assignment

https://aka.ms/az301asg

Microsoft Azure

Open Mic