Caltech | Center for Technology & Management Education

Designing Infrastructure Solutions on Azure

Powerd by simplilearn

Design an Application Architecture

Learning Objectives

By the end of this lesson, you will be able to:

- Define Azure functions
- List the features of Azure Kubernetes Service
- Recommend an orchestration solution for deployment and maintenance of applications
- Recommend a solution for API integration

A Day in the Life of an Azure Architect

You are working as an architect for an organization that is building a hospitality management application.

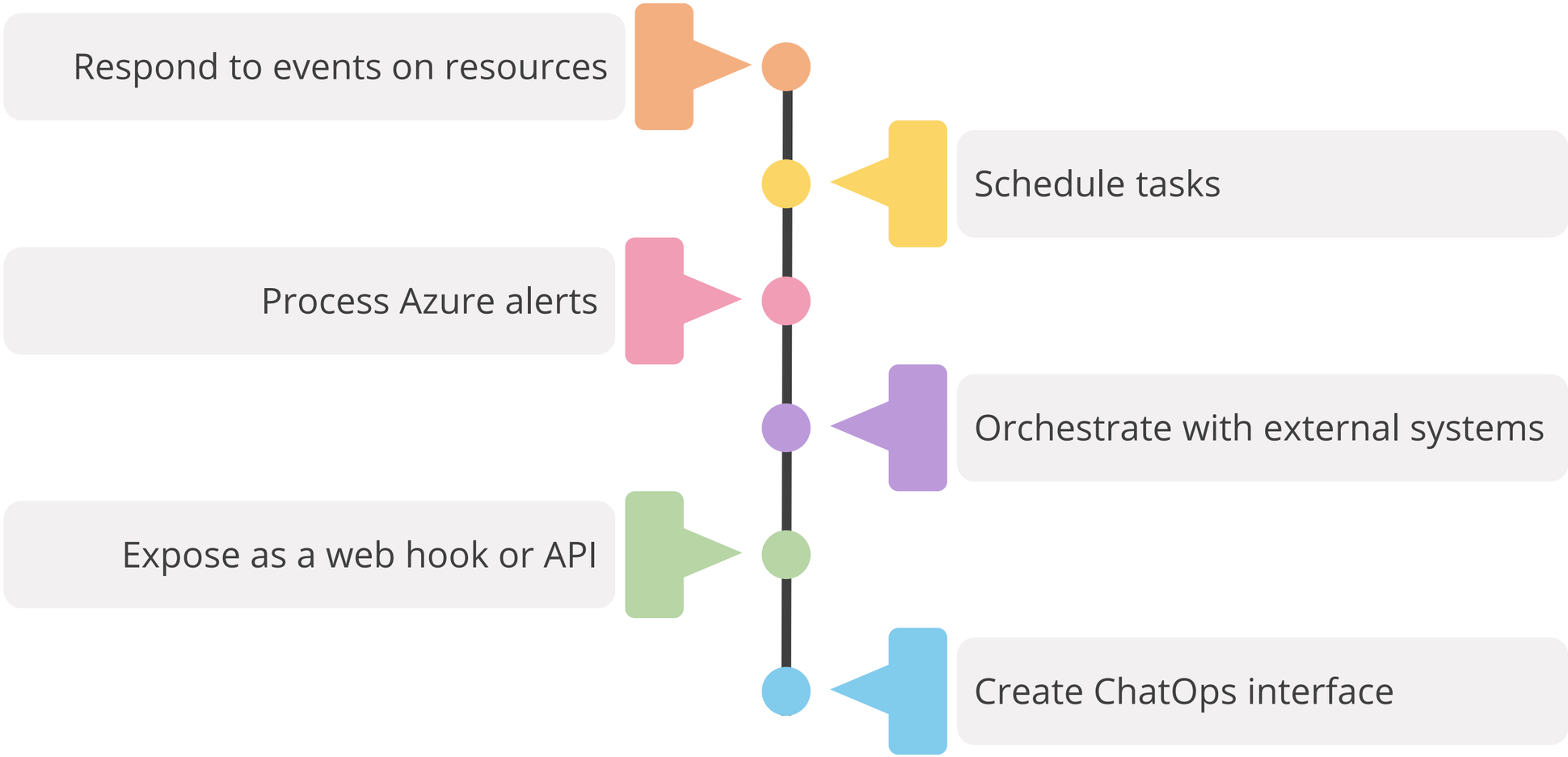
- The app would contain microservices hosted in Azure. You need to decide on the right choice of service for this application.
- You need to also suggest a solution that will allow you to easily deploy and run containerized applications on both Windows and Linux.

To achieve all of the above, along with some additional features, we would be learning a few concepts in this lesson that will help you find a solution for the above scenario.

Recommend a Microservices Architecture

Patterns in Event-Based Automation

Event-based automation scenarios work best with Azure Functions. They follow these common patterns:

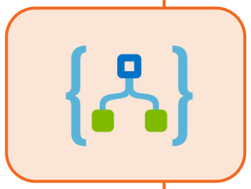


Architecture

The architecture consists of the following blocks:

Azure Functions:

- Provides event-driven and serverless compute capabilities in the architecture
- Performs automation tasks when triggered by events or alerts

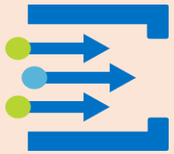


Logic Apps:

- Performs simple tasks that are easily implemented using the built-in connectors
- Allows users to perform tasks that range from email notifications to integration with external management applications

Architecture

The architecture consists of the following blocks:



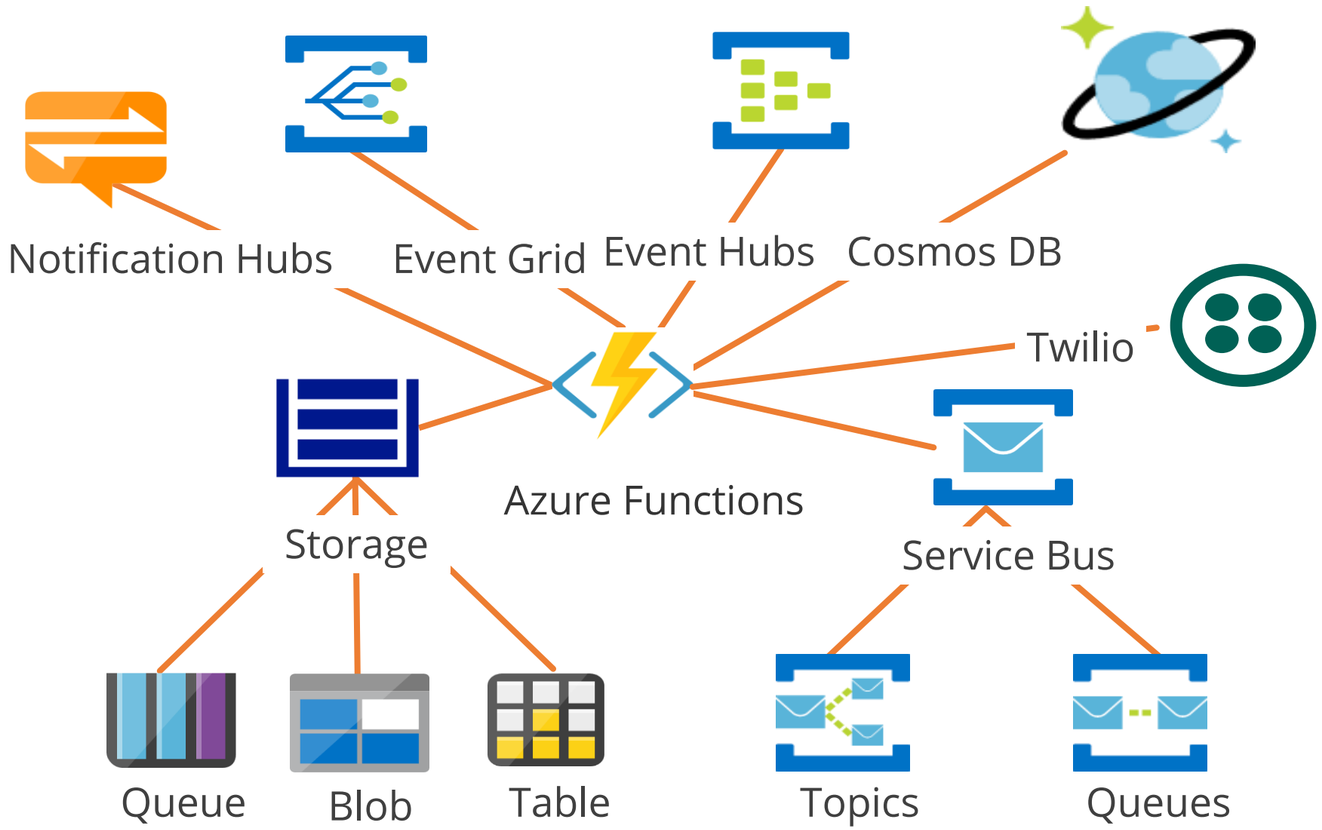
Event Grid: Built-in support for events from other Azure services, as well as custom events (also called custom topics).

Azure Monitor: Alerts can monitor critical conditions and take corrective action using Azure Monitor action groups.

Automation Action: Represents other services that a function can interact with, so that it can provide the automation functionality

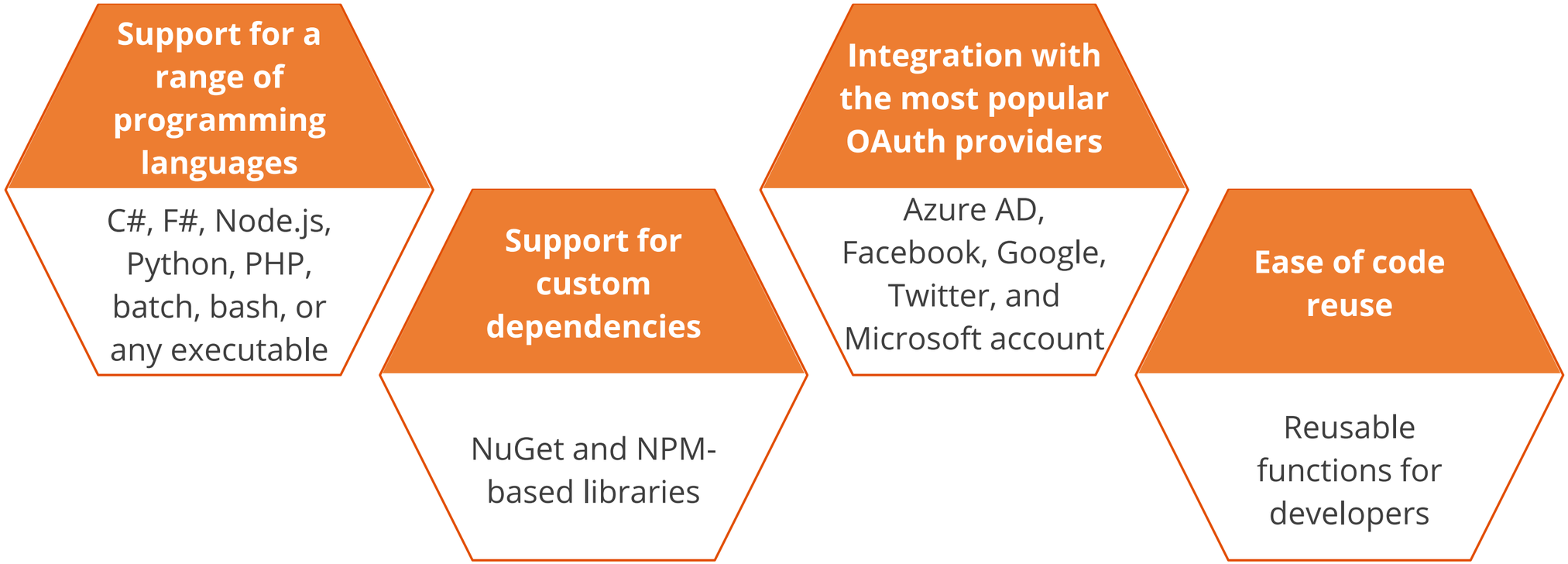
Azure Functions

Azure functions allow users to run small pieces of code (functions) without worrying about the application infrastructure.



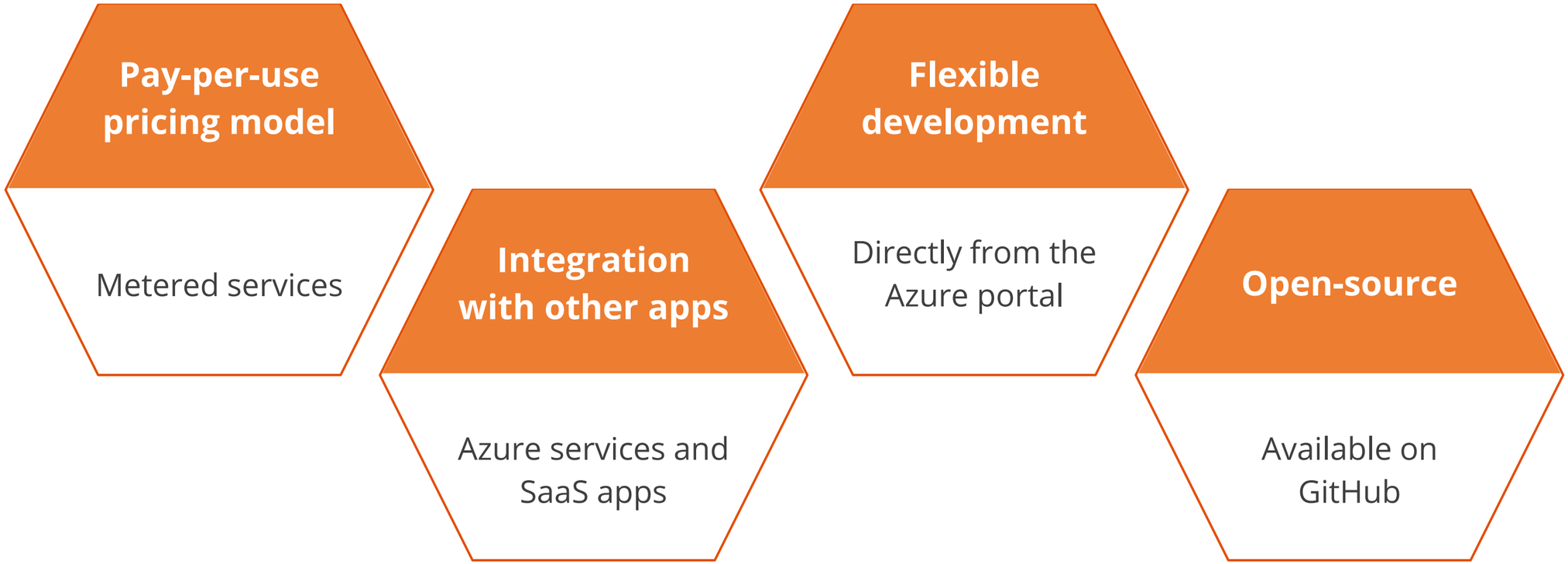
Features of Azure Functions

The key features of Azure functions are:



Features of Azure Functions

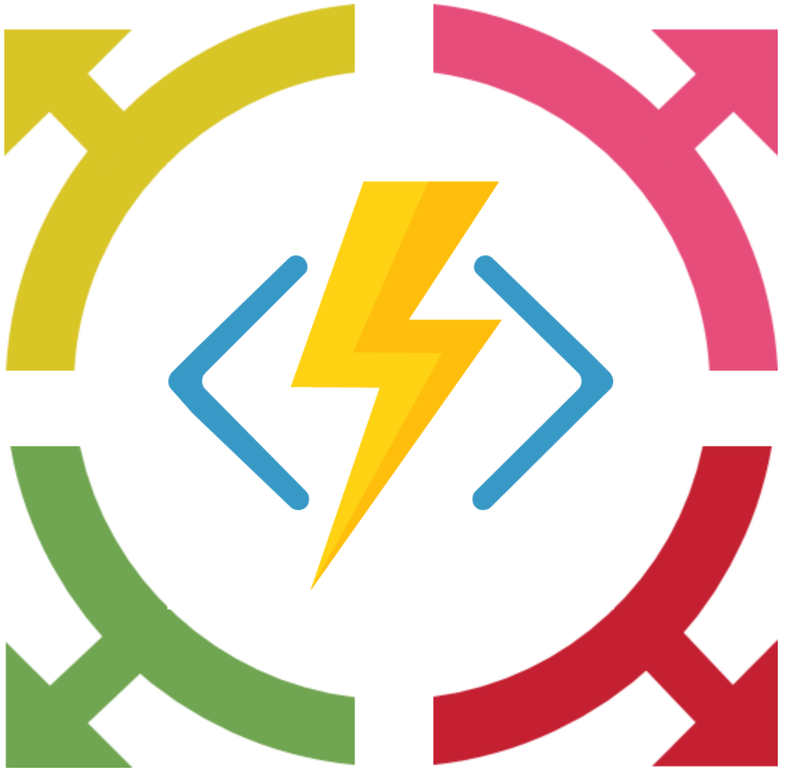
The key features of Azure functions are:



Azure Functions Use cases

The use cases for Azure Functions are as follows:

Run code based on HTTP requests and schedule code to run at predefined times



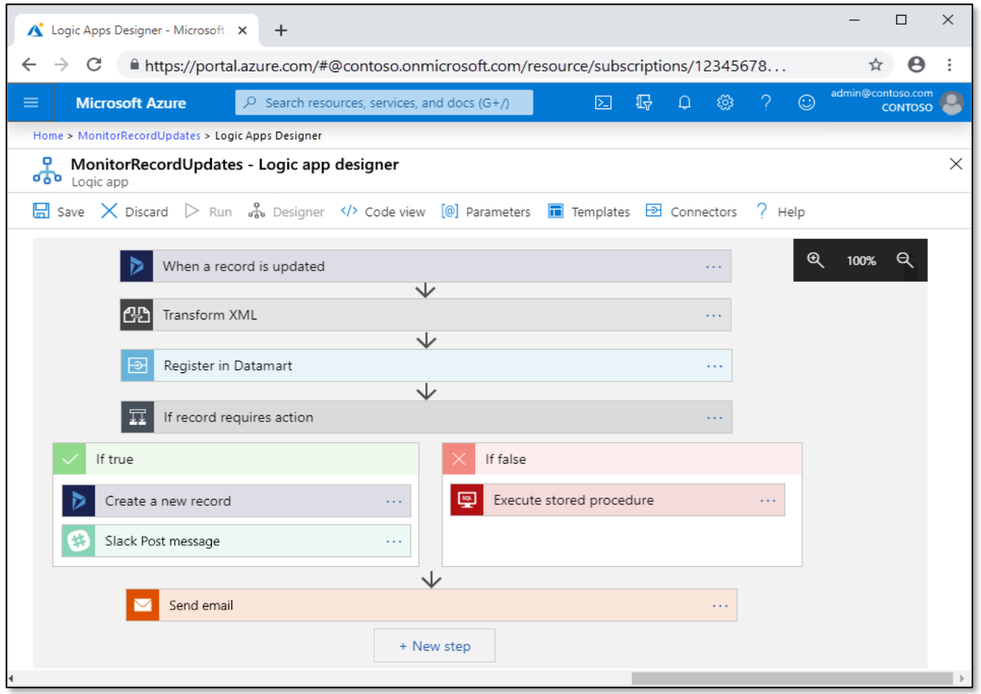
Respond to high volumes of Azure Event Hubs events and respond to Azure Service Bus queue and topic messages

Respond to Azure Event Grid events by using subscriptions and filters

Process new and modified Azure Cosmos DB documents, Azure Storage Blobs, and Azure Queue storage messages

Azure Logic Apps

Logic App is a cloud service used to schedule, automate, and orchestrate tasks, business processes, and workflows. It is useful for integrating apps, data, systems, and services across organizations.



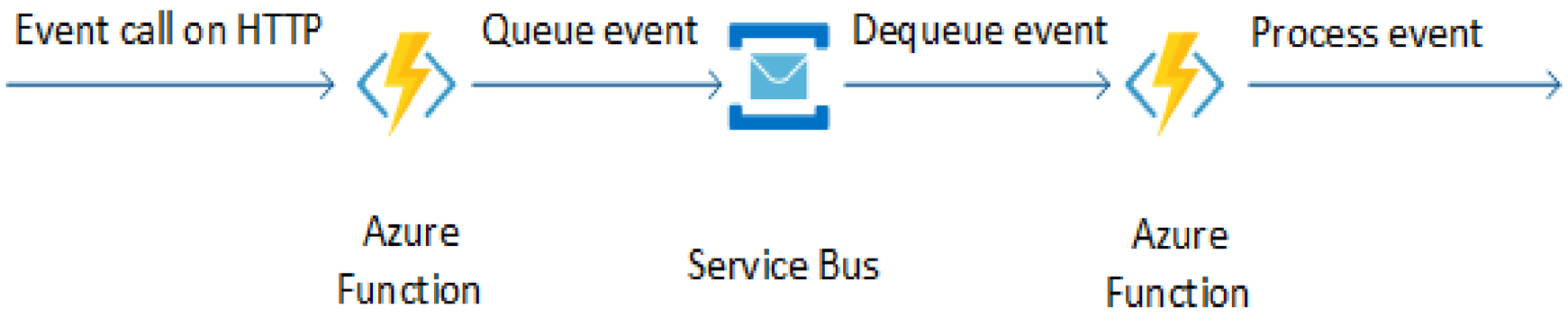
Logic App simplifies how users design and build scalable solutions:

- The workflow starts with a trigger that fires when an event happens or when new data meets specific criteria.
- When trigger fires, the Logic App's engine creates a logic app instance that runs the actions in the workflow.

Resiliency

Azure Functions

To avoid HTTP timeouts for a longer automation task, queue the event in a service bus and handle the actual automation in another function.



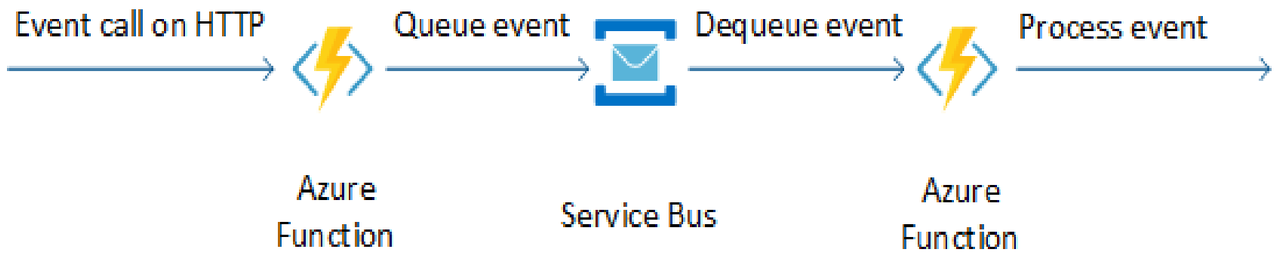
Resiliency

Event Grid

Check if there is a high volume of generated events and enough to clog the grid. The cost center workflow does not implement additional checks for this, as it only watches for resource creation events in a resource group.

Azure Monitor

If many alerts are generated and the automation updates Azure resources, throttling limits of the Azure Resource Manager might be reached. Avoid this situation by limiting the frequency of alerts getting generated by the Azure Monitor.



Security

Control access to the function

Restrict access to an HTTP-triggered function by setting the authorization level. With anonymous authentication, the function is accessible with a URL:

http://<APP_NAME>.azurewebsites.net/api/<FUNCTION_NAME>

Control function access

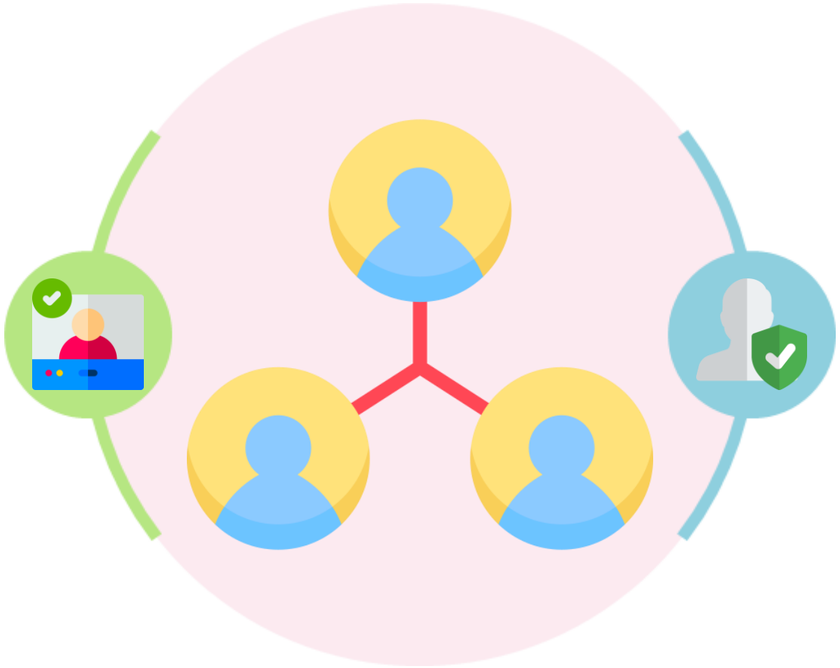
Managed identities for Azure resources simplifies how a function authenticates and accesses other Azure resources and services. The code does not need to manage the authentication credentials because it is managed by Azure AD.

Security

Two types of managed identities are as follows:

System-assigned managed identities:

These are created as part of the Azure resource and cannot be shared among multiple resources. They get deleted when the resource is deleted.



User-assigned managed identities:

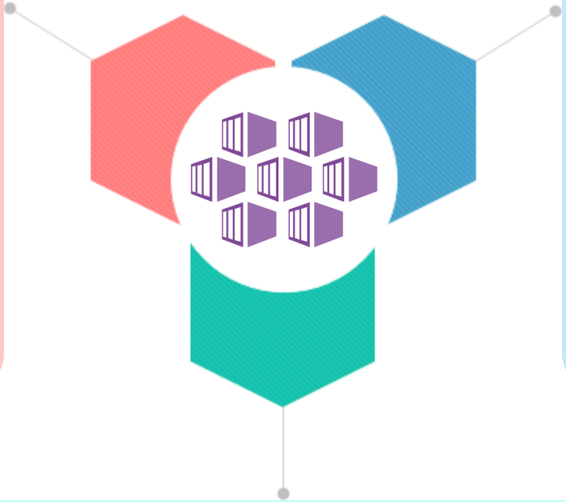
These are created as stand-alone Azure resources. They can be shared across multiple resources and need to be explicitly deleted.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) manages the user's hosted Kubernetes environment and makes it simple to deploy and manage containerized applications in Azure.

Azure Kubernetes Service

Allows users to easily orchestrate large solutions using a variety of containers such as application containers, storage containers, and middleware containers



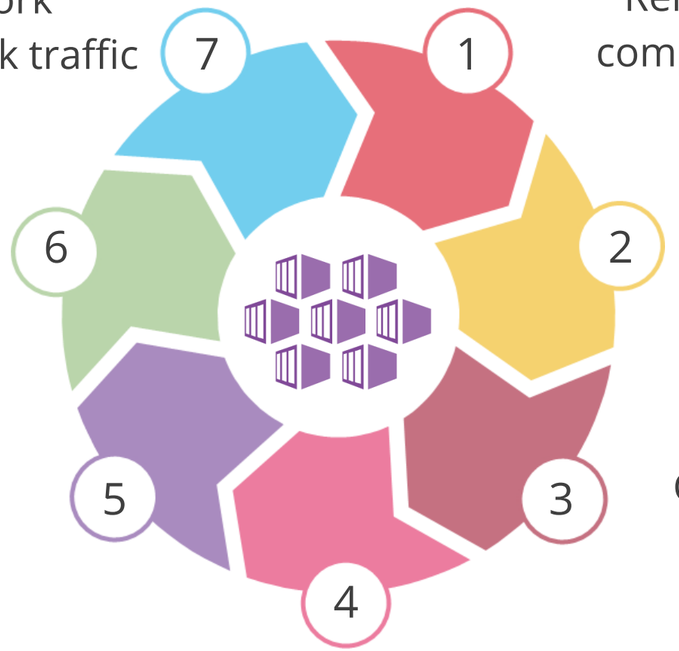
- Manages container-based applications along with networking and storage requirements
- Focuses on application workloads instead of infrastructure components

- Describes applications in a declarative form using YAML files
- Handles management and deployment

Azure Kubernetes Service Features

Simple management of a cluster of VMs using Kubernetes:

1. Removes infrastructure complication and planning
2. Supports in-place upgrade of clusters to the latest version of Kubernetes
3. Charges users based on the resource consumption
4. Provides secure, reliable, and highly scalable environment
5. Supports node autoscaling to meet resource utilization
6. Supports direct integration with Azure Container Registry for Docker images
7. Allows Azure Virtual Network integration for isolated network traffic



Assisted Practice

Azure Kubernetes Service **Duration: 10 Min.**

Problem Statement:

Contoso has several multitier applications that are not suitable to run by using Azure Container instances. To determine whether they can be run as containerized workloads, you want to evaluate using Kubernetes as the container orchestrator. To further minimize management overhead, you want to test Azure Kubernetes Service, including its simplified deployment experience and scaling capabilities.

Assisted Practice: Guidelines

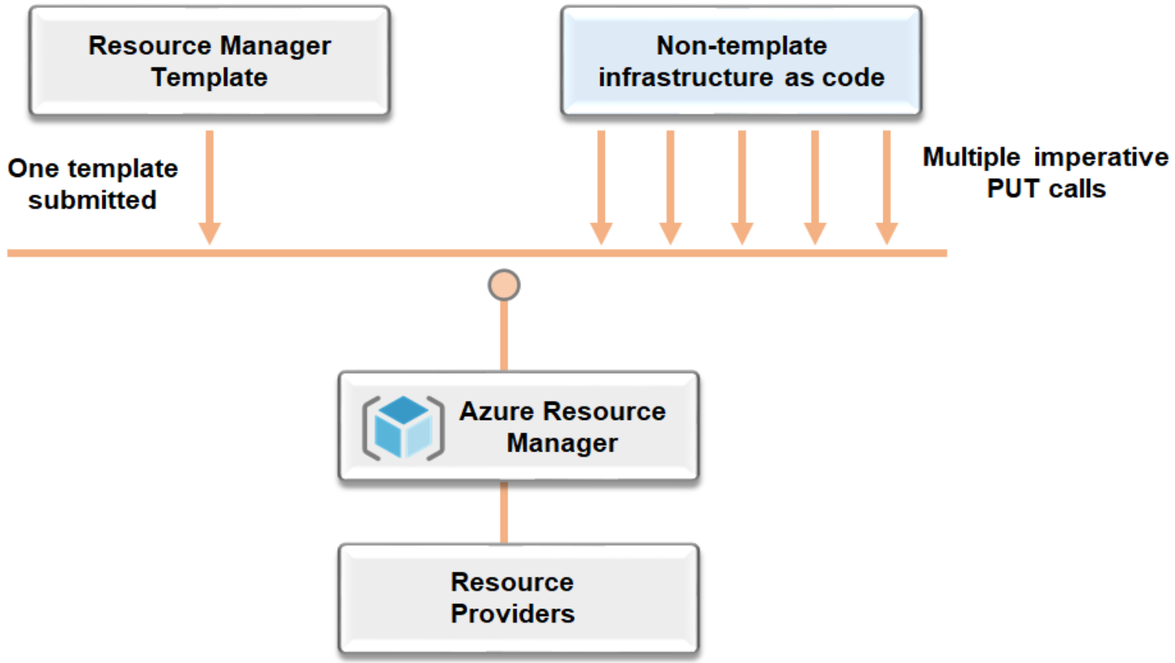
Steps to create Kubernetes service are:

1. Registering the Microsoft Kubernetes resource providers
2. Deploying an Azure Kubernetes Service cluster
3. Deploying pods into the Azure Kubernetes Service cluster
4. Scaling containerized workloads in the Azure Kubernetes Service cluster

Recommend an Orchestration Solution for Deployment and Maintenance of Applications

Azure Resource Manager Templates

Azure Resource Manager templates are useful for implementing infrastructure as code for Azure solutions.

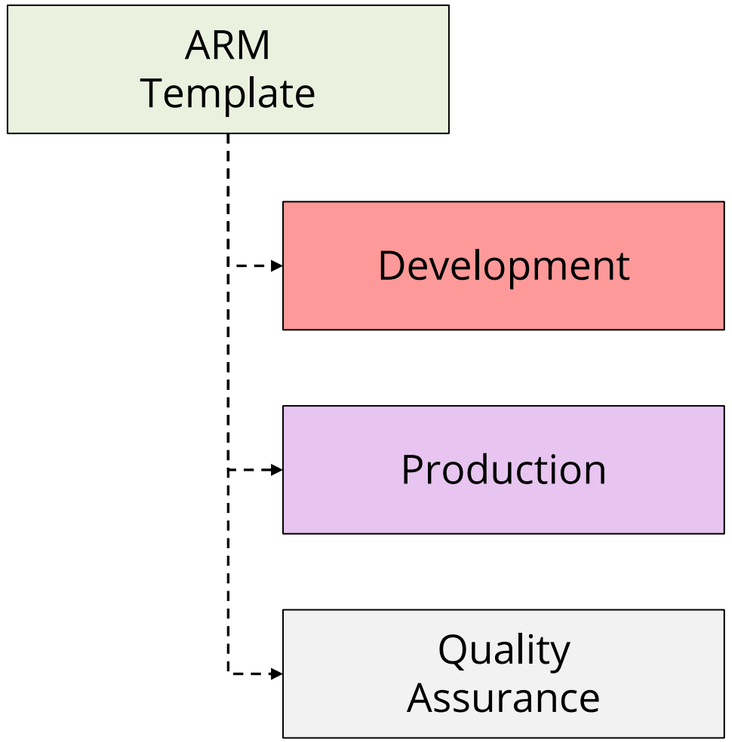


It is a JavaScript Object Notation (JSON) file that defines all the resource manager resources in a deployment.

Source: docs.microsoft.com

Template Advantages

Resource Manager templates will make the deployments quicker and more repeatable.



Template Advantages

The advantages of the template are as follows:

- Improves consistency
- Promotes reusability
- Supports complex deployments
- Allows users to modularize templates
- Reduces manual tasks
- Simplifies orchestration
- Reduces error-prone tasks
- Allows users to express requirements through code

Template Design

Users can create templates and resource groups depending on how they wish to manage the solution.

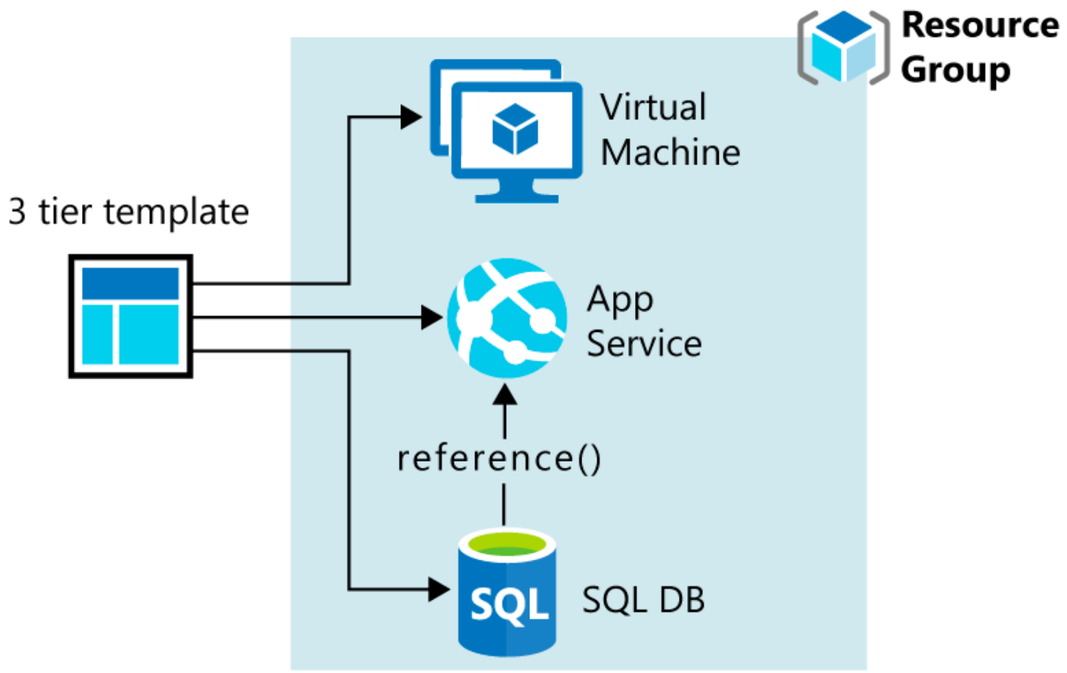


Image source: https://docs.microsoft.com/en-in/

Template Design

Users can divide their deployment requirements into a set of targeted, purpose-specific templates.

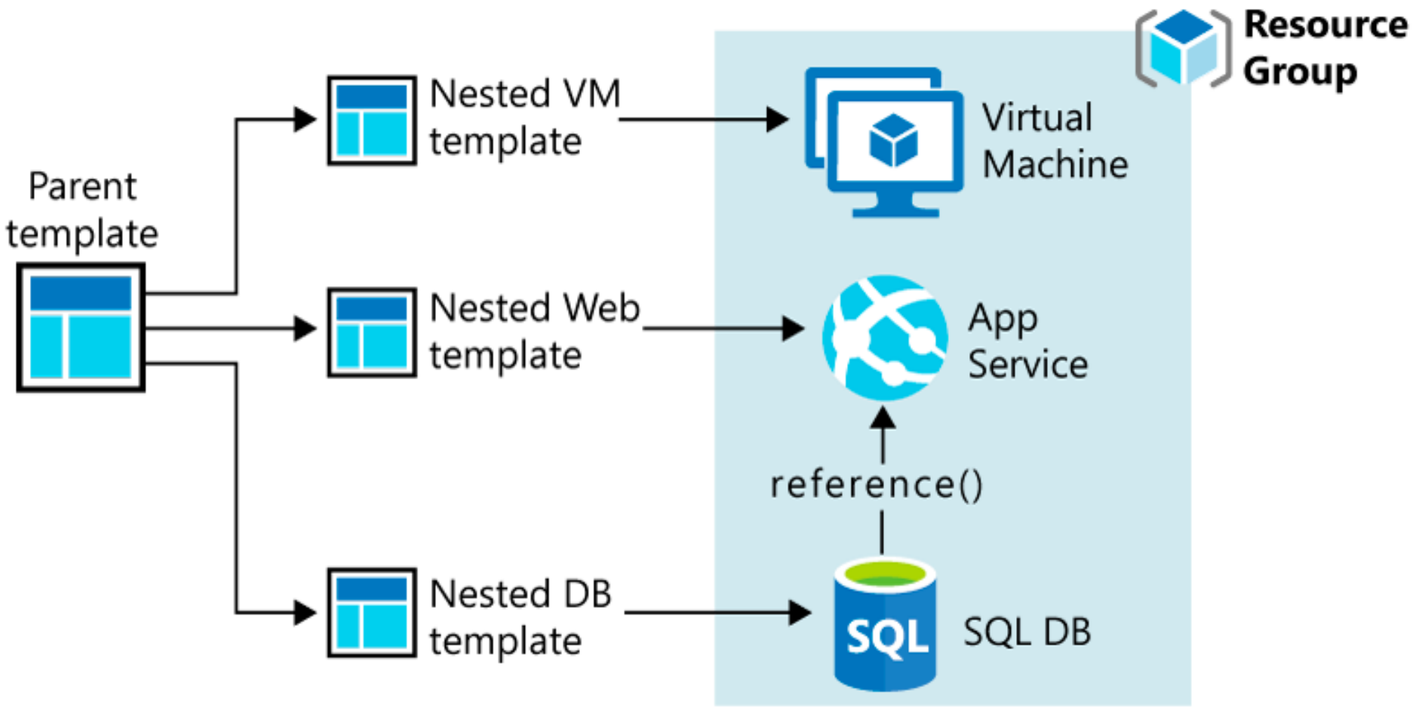


Image source: https://docs.microsoft.com/en-in/

Template Design

Users can deploy three tiers to separate resources if their tiers have separate life cycles.

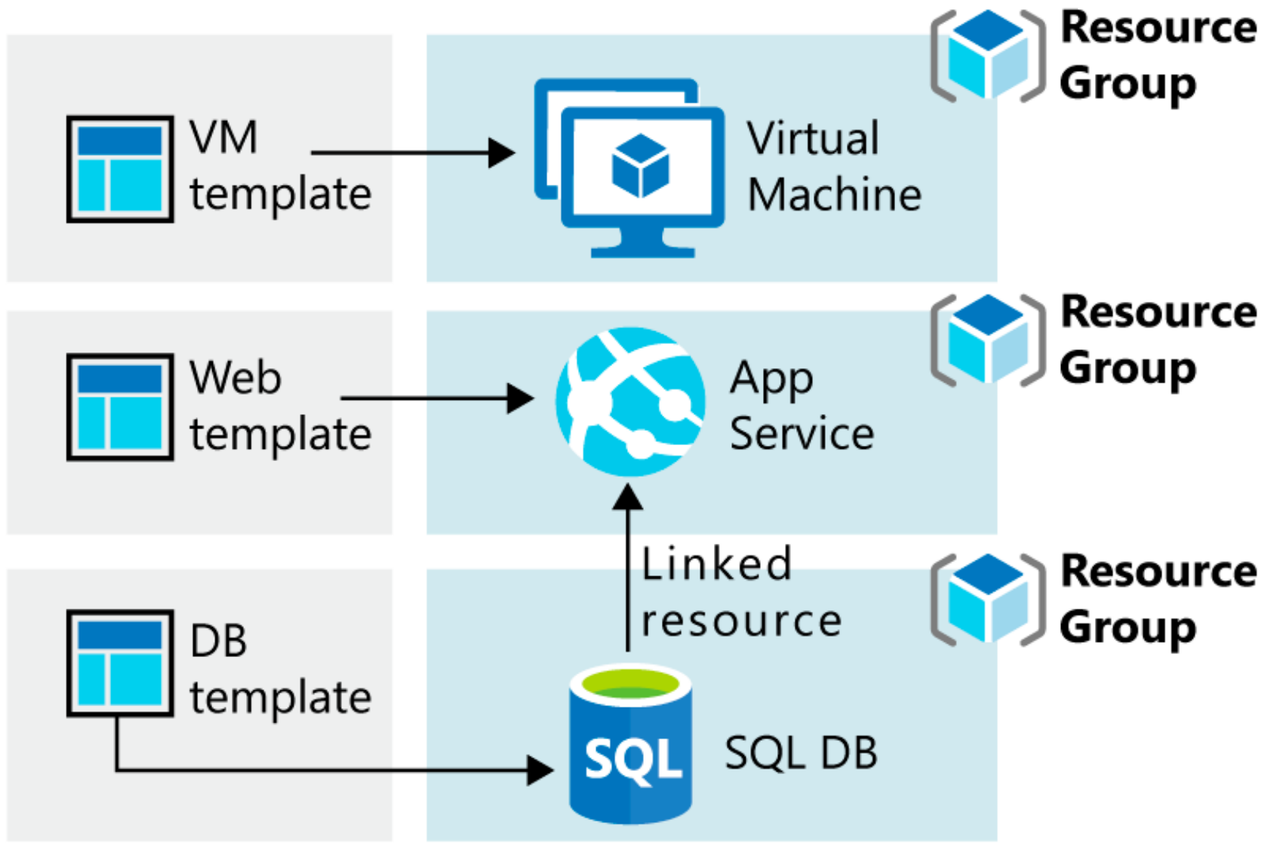


Image source: https://docs.microsoft.com/en-in/

Template Schema

Template Schema defines all the resource manager resources in a deployment.

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

Template Schema

These are the features of template schema:

- It is written in JSON.
- It is a collection of key-value pairs.
- Each key is a string.
- Each value can be a string, number, Boolean expression, list of values, or object.

Resource Manager Template Sections

Element name	Required	Description
$Schema	Yes	Location of the JSON schema file that describes the version of the template language
ContentVersion	Yes	Version of the template (such as 1.0.0.0). The users can provide any value for this element. Use this value to document significant changes in the template. When deploying resources using the template, this value can be used to make sure that the right template is being used
Parameters	No	Values that are provided when deployment is executed to customize resource deployment
Variables	No	Values that are used as JSON fragments in the template to simplify template language expressions
Functions	No	User-defined functions that are available within the template
Resources	Yes	Resource types that are deployed or updated in a resource group
Outputs	No	Values that are returned after deployment

Template Deployment Options

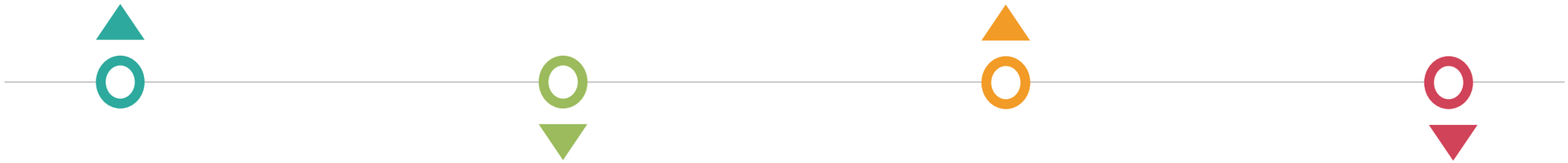
Users can deploy the templates by:

Using Azure Portal

Using PowerShell

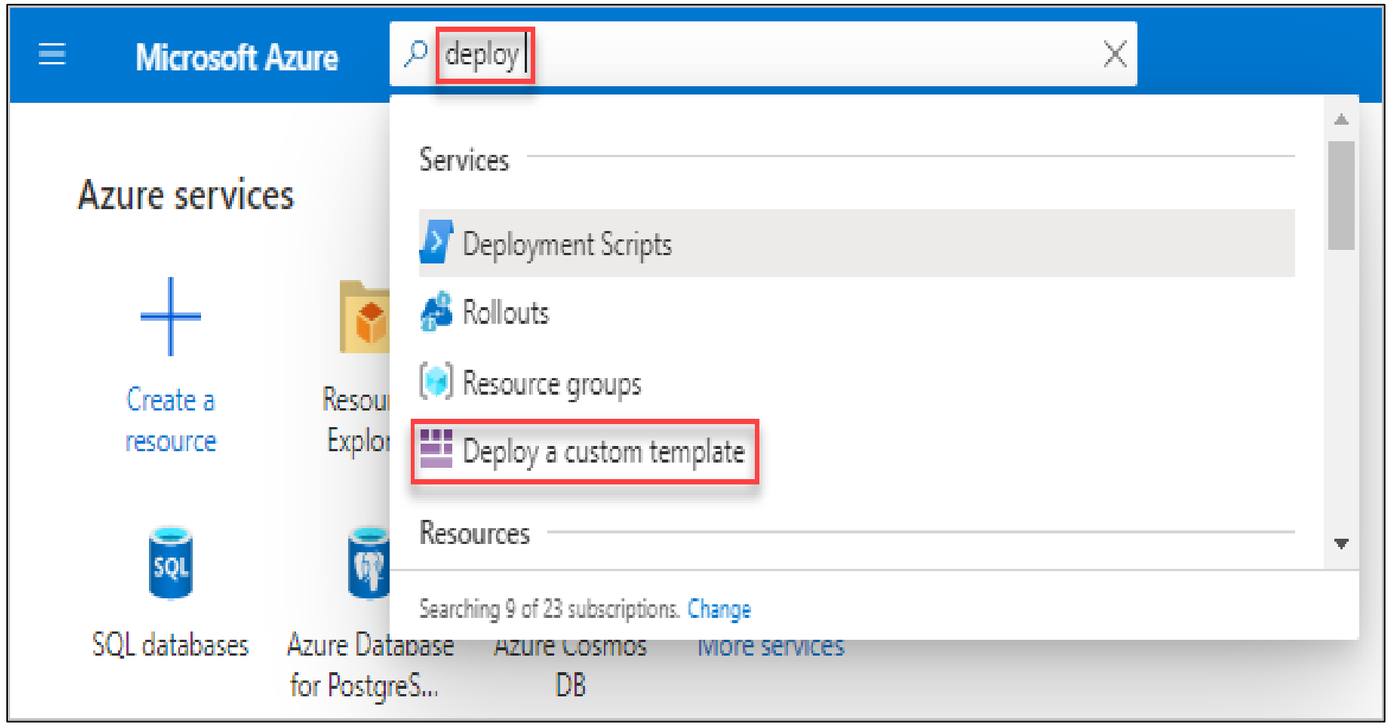
Using CLI

Creating ARM template using VS code



Template Deployment Options

Users can deploy the custom template from the Azure portal.



Template Deployment Options

Users can also deploy the template using PowerShell.

Example:

```
$templateFile = "{provide-the-path-to-the-template-file}"
New-Az Resource Group Deployment `
  -Name blank template `
  -ResourceGroupName my Resource Group `
  -Template File $template File
```

Deploy an ARM Template Using CLI

Users can deploy the ARM template using the CLI.

Example:

```
"resources": [
   {
     "type": "Microsoft.Storage/storageAccounts",
     "apiVersion": "2019-04-01",
     "name": "[variables('storageAccountName')]",
     "location": "[parameters('location')]",
     "sku": {
       "name": "[parameters('storage SKU')]"
     },
     "kind": "Storage V2",
     "properties": {
       "supports Https Traffic Only": true
     }
   },
```

Runbooks in Azure Automation

The following figure illustrates the life cycle of a runbook job for different types of runbooks:

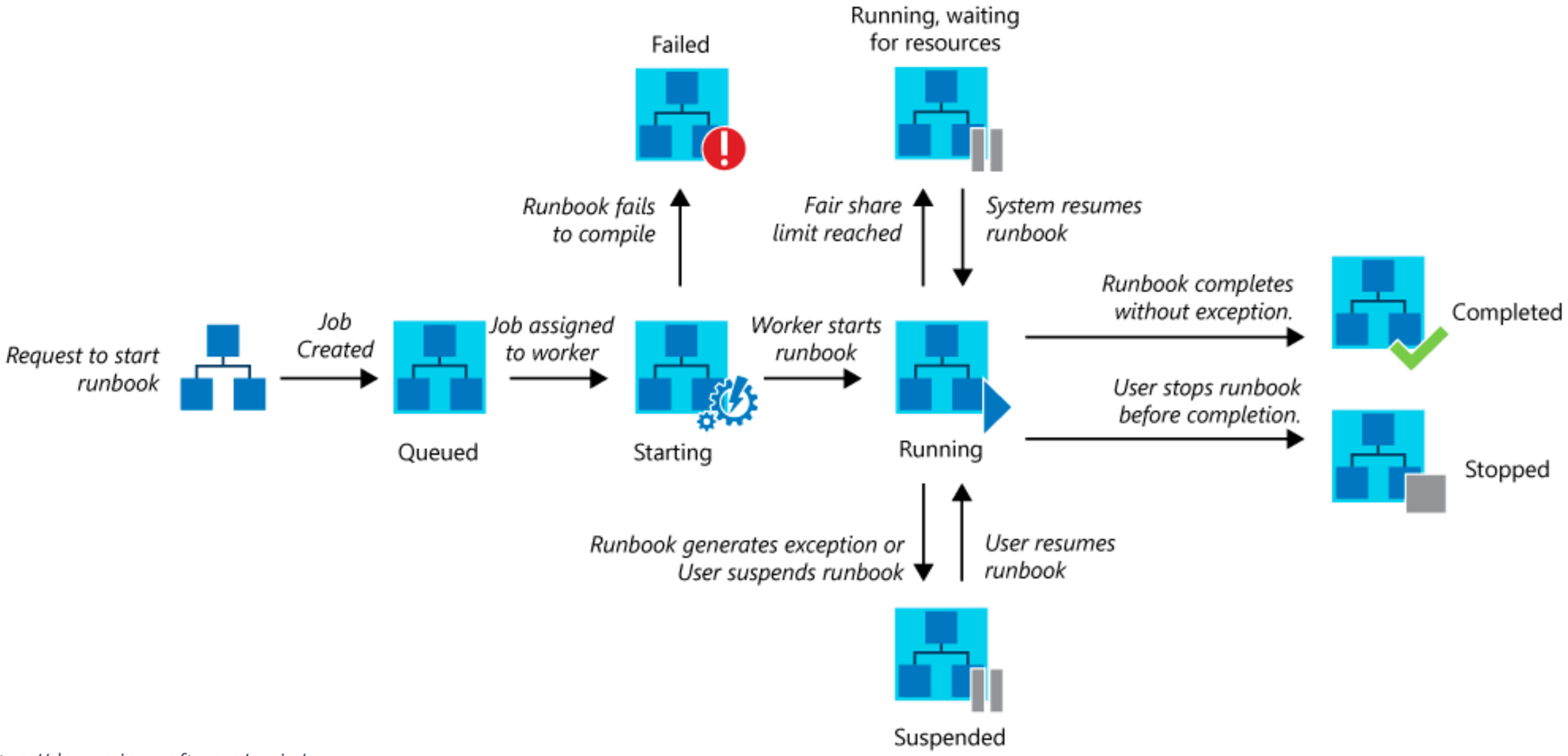
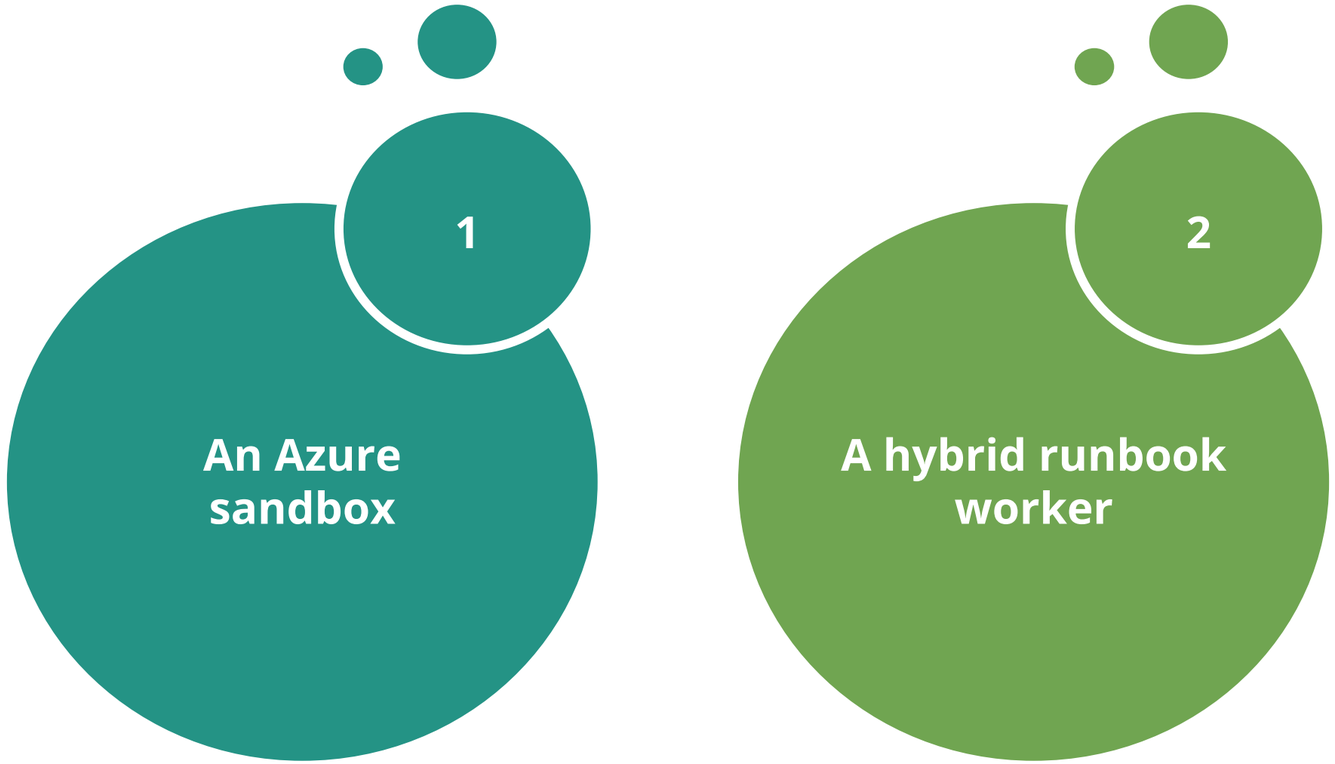


Image source: https://docs.microsoft.com/en-in/

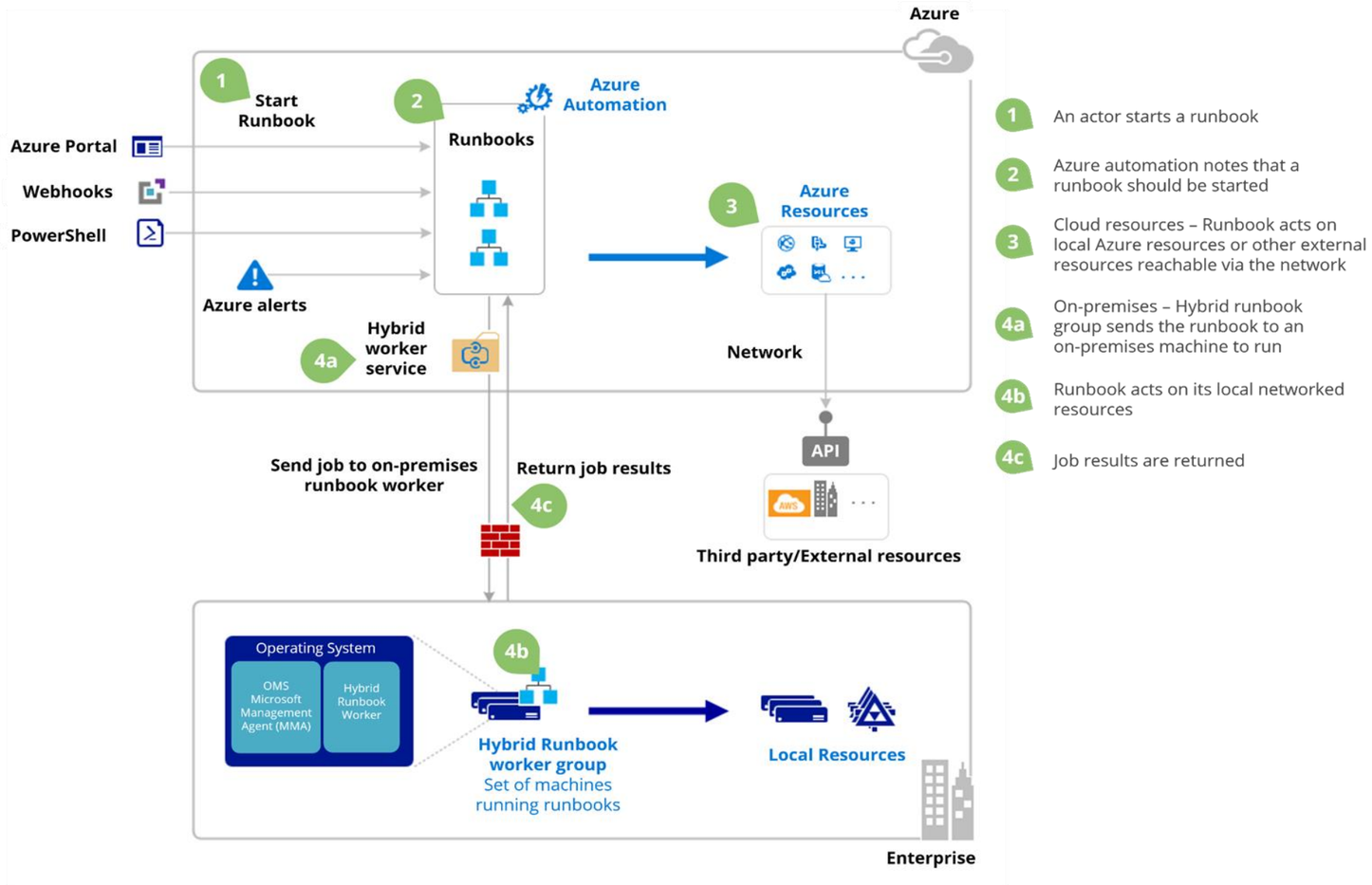
Runbooks in Azure Automation

Azure Automation runbooks can run in:



Start a Runbook in Azure Automation

The following diagram shows the life cycle of a runbook:



Start a Runbook in Azure Automation

A runbook can be started using:

1. Azure portal
2. PowerShell

```
Start-AzAutomationRunbook
     -AutomationAccountName "MyAutomationAccount" `
     -Name "Test-Runbook" `
     -ResourceGroupName "ResourceGroup01"
```

Assisted Practice

ARM Template

Duration: 10 Min.

Problem Statement:

You've been requested as an Azure Architect to offer a solution for automating Application Architecture. Create an ARM template for the requested resource.

Assisted Practice: Guidelines

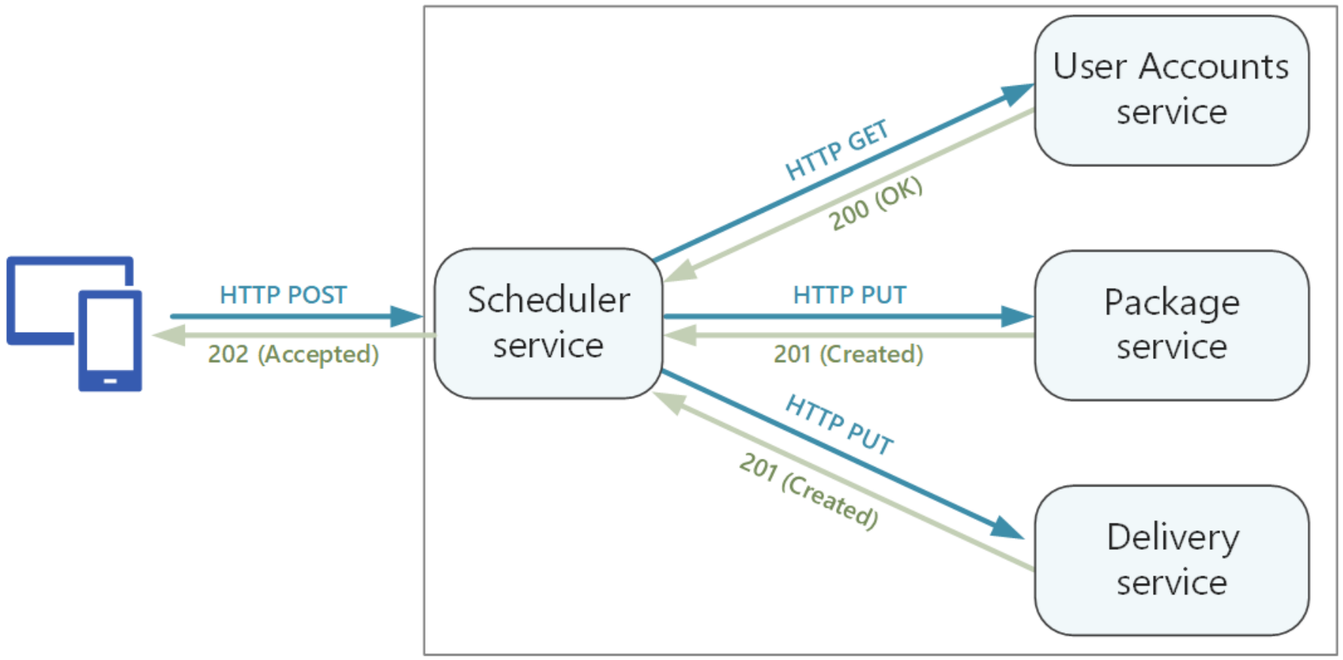
Steps to create an ARM template are:

1. Log in to your Azure portal
2. Create a resource
3. Download the ARM Template

Recommend a Solution for API Integration

Designing APIs for Microservices

All data exchange between services happens either through messages or API calls.



All APIs must have well-defined semantics and versioning schemes so that updates do not break other services.

APIs Types

There are two types of APIs:

Public APIs

Backend APIs

APIs Considerations

REST

- It defines a uniform interface based on HTTP verbs.
- It includes well-defined semantics for idempotency, side effects, and response codes.

RPC

Based on operations or commands, RPC interfaces can affect the design with overly chatty APIs.

Baseline Recommendation

Choose REST over HTTP unless the user needs the performance benefits of a binary protocol. REST over HTTP requires no special libraries.

Key Takeaways

- Azure Functions allows the user to run small pieces of code without worrying about application infrastructure.
- Azure Kubernetes Service manages the user's hosted Kubernetes environment and makes it simple to deploy.
- Azure Automation runbooks can run in an Azure sandbox and a hybrid runbook worker.
- REST defines a uniform interface based on HTTP verbs. It includes well-defined semantics for idempotency, side effects, and response codes.

Designing Solution for App Architecture

Duration: 25 min.

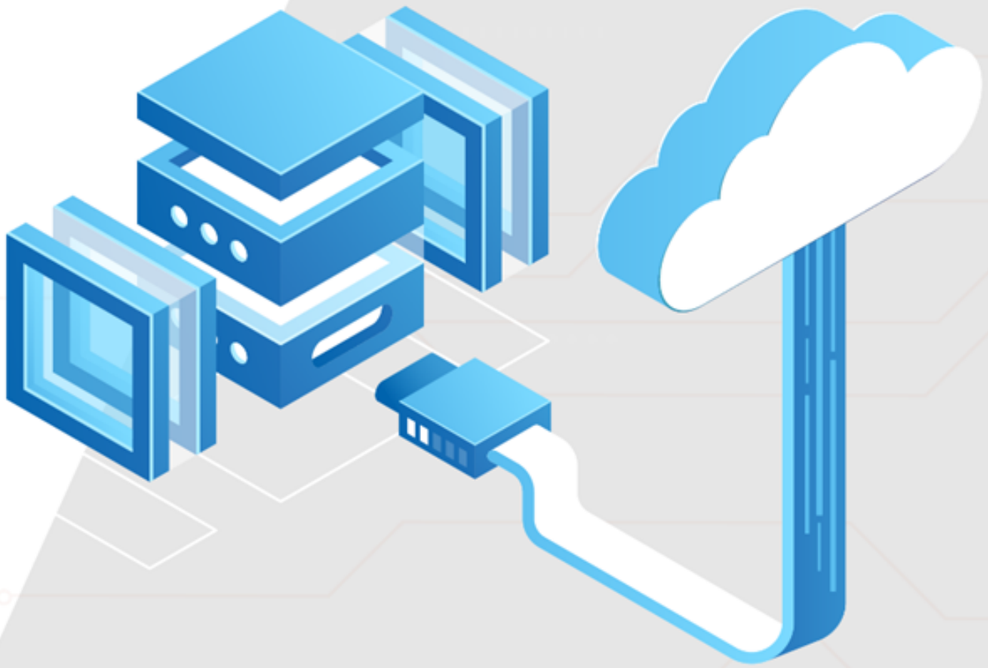
Project Agenda: To design the Azure Kubernetes Service Cluster

Description: You are working as an architect for an organization that is developing a hospitality management application. The application will contain microservices hosted in Azure. You need to decide on the right choice of service for this application. You also need to provide developers with the ability to proactively identify and fix performance issues. They should also be able to simulate user connections to the room booking service from the internet.

Perform the following:

1. Log in to the Azure Portal
2. Create a resource
3. Configure basic details and authentication methods
4. Deploy Kubernetes Service Cluster

LESSON-END PROJECT



Thank you