Cloud Computing

Caltech | Center for Technology & Management Education

Post Graduate Program in Cloud Computing

Powerd by simplilearn

Cloud Computing

Caltech | Center for Technology & Management Education

PG CC - Microsoft Azure Architect Design: AZ:304

Powerd by simplilearn

Design an Application Architecture

Learning Objectives

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

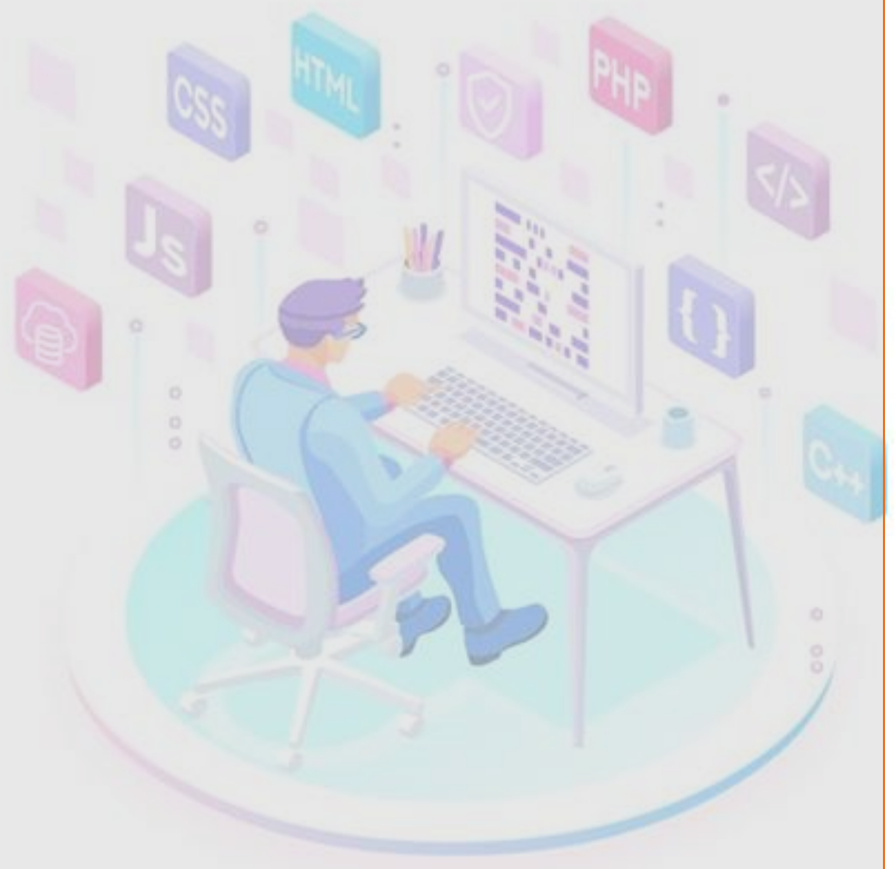
- Define the Azure key vaults and its benefits
- Explain the best practices of Azure key vaults
- Recommend an orchestration solution for deployment and maintenance of applications
- Recommend a solution for API integration

A Day in the Life of an Azure Architect

You are working as an Architect for an organization which is building a hospitality management application.

- The app would be containing 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

Patterns in event-based automation are as follows:

- Respond to events on resources
- Schedule tasks
- Process Azure alerts
- Orchestrate with external systems
- Expose as a web hook or API
- Create ChatOps interface

Architecture

The architecture consists of the following blocks:

Azure Functions:

- Provide event-driven and serverless compute capabilities in this architecture
- Perform automation tasks, when triggered by events or alerts.

Logic Apps:

- Used to perform simple tasks, easily implemented using the built-in connectors
- Tasks can range from email notifications to integrating with external management applications.

Architecture

The architecture consists of the following blocks:

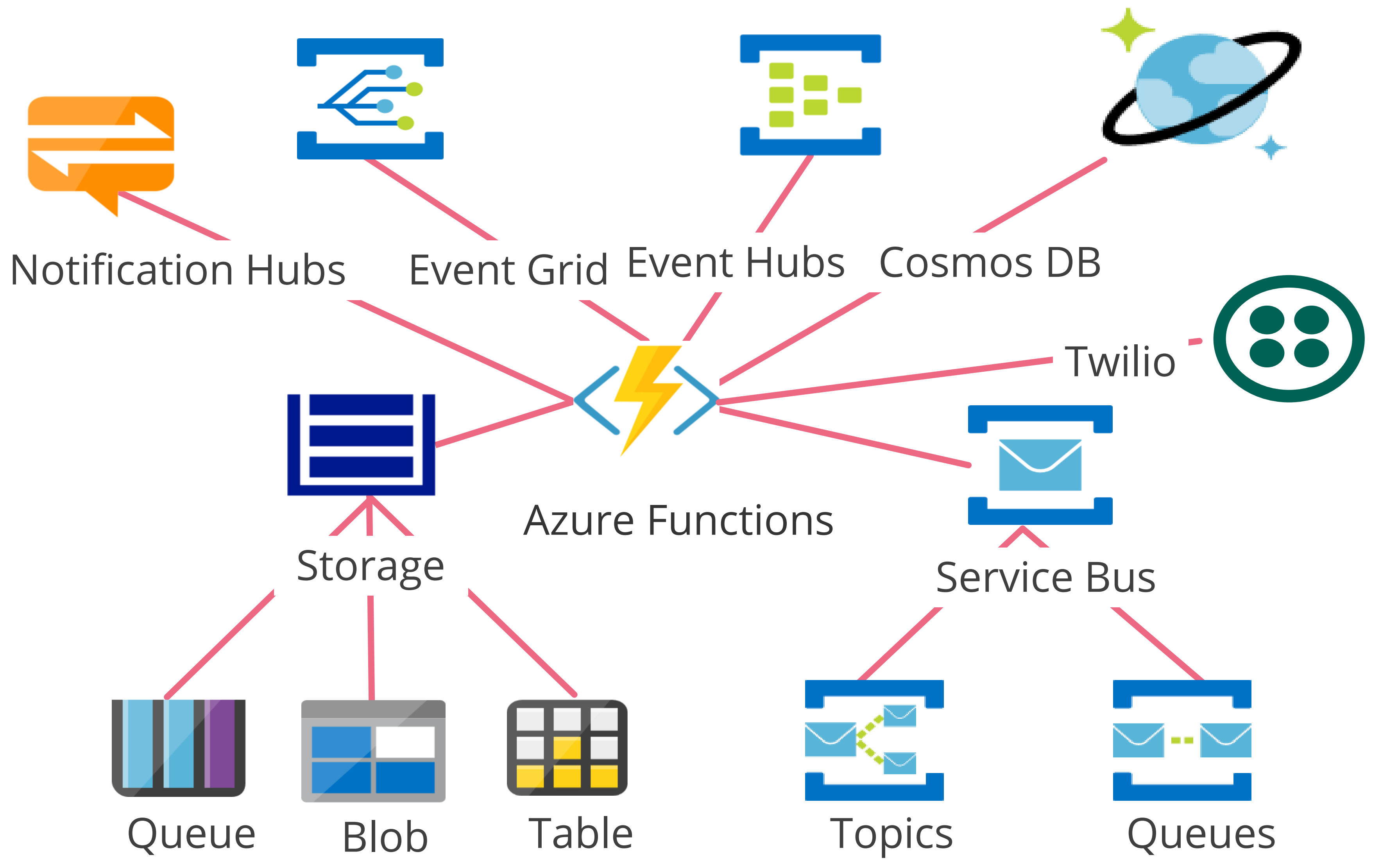
Event Grid: Built-in support for events from other Azure services, as well as custom events (also called custom topics).

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

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

Azure Functions

Azure Functions allow you to run small pieces of code (called **functions**) without worrying about application infrastructure.



Features of Azure Functions

The key features of Azure functions are:

Support for a range of programming languages

C#, F#, Node.js, Python, PHP, batch, bash, or any executable

Support for custom dependencies

NuGet and NPM-based libraries

Integration with the most popular OAuth providers

Azure AD, Facebook, Google, Twitter, and Microsoft Account

Ease of code reuse

Functions can be reused by developers

Features of Azure Functions

The key features of Azure functions are:

Pay-per-use pricing model

Metered services

Integration with other apps

Azure services and SaaS apps

Flexible development

Directly from the Azure portal

Open-source

Available on GitHub

Azure Functions Use cases

The use cases for Azure Functions:

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

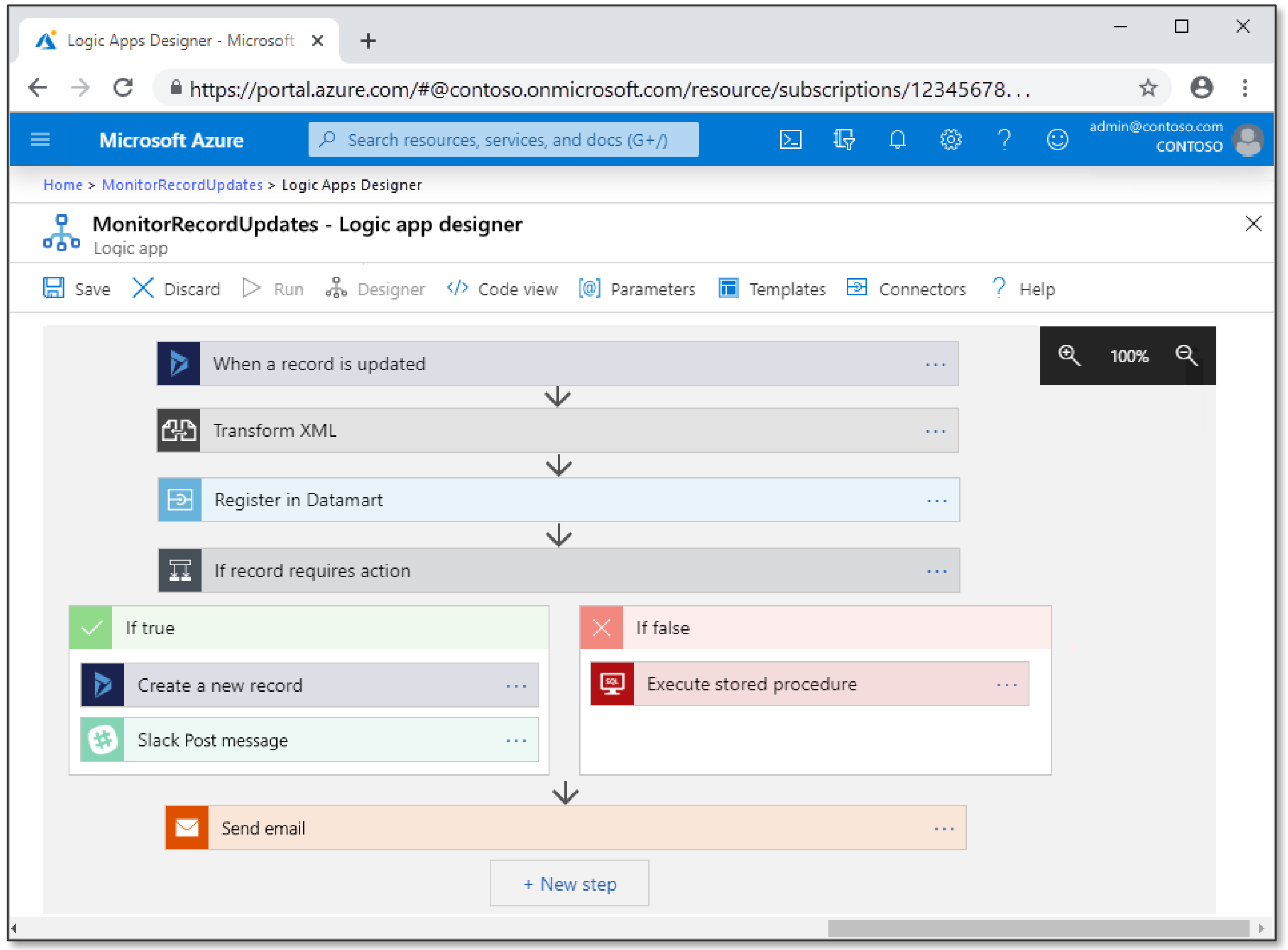
Respond to high volumes of Azure Event Hubs events and respond to Azure Service Bus queue and topic messages

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

A cloud service used to schedule, automate, and orchestrate tasks, business processes, and workflows

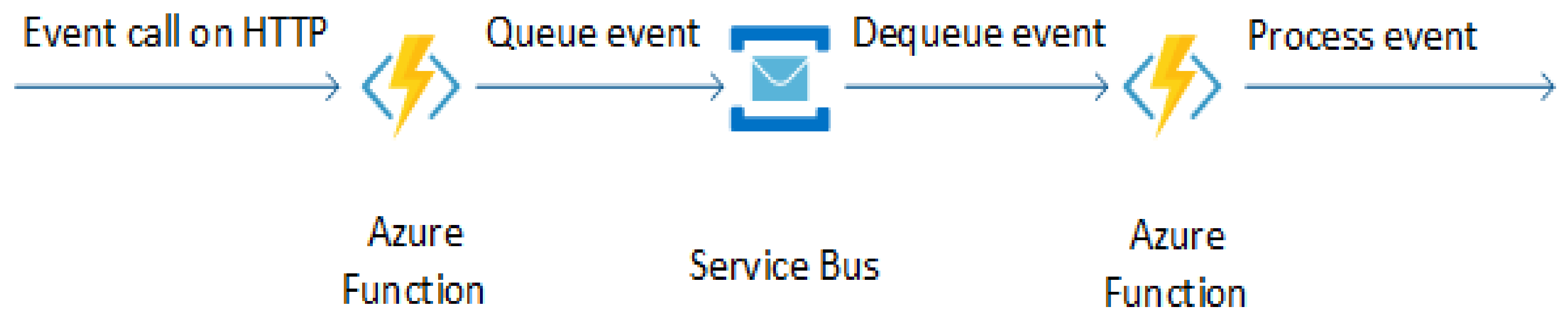

Logic Apps Designer - Microsoft
https://portal.azure.com/#@contoso.onmicrosoft.com/resource/subscriptions/12345678...
Microsoft Azure
Search resources, services, and docs (G+/)
admin@contoso.com
CONTOSO
Home > MonitorRecordUpdates > Logic Apps Designer
MonitorRecordUpdates - Logic app designer
Logic app
Save
Discard
Run
Designer
Code view
Parameters
Templates
Connectors
Help
100%
When a record is updated
Transform XML
Register in Datamart
If record requires action
If true
Create a new record
Slack Post message
If false
Execute stored procedure
Send email
+ New step

- It's used to integrate apps, data, systems, and services across enterprises or organizations.
- It simplifies how you design and build scalable solutions:
  - The workflow starts with a trigger, which fires when an event happens or new data meets specific criteria.
  - When trigger fires, the Logic Apps 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.



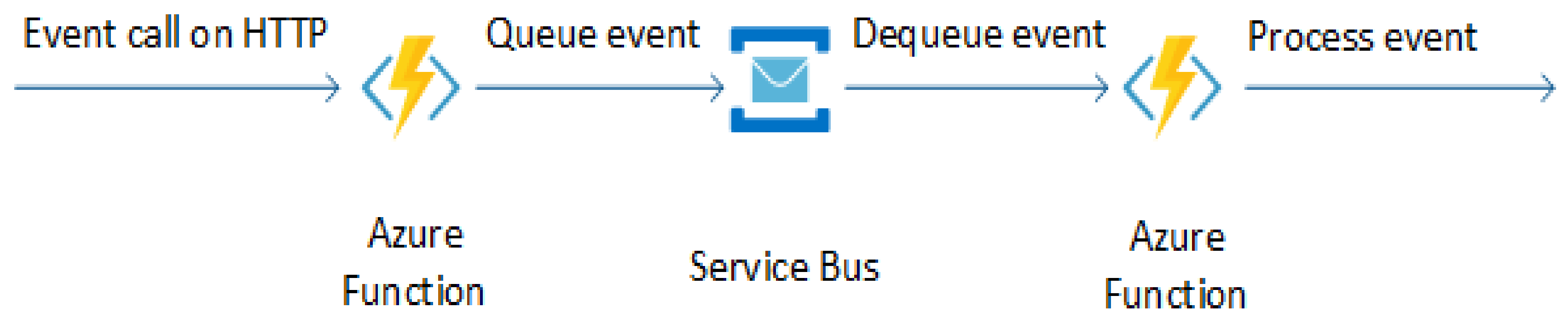
Resiliency

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, because 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.

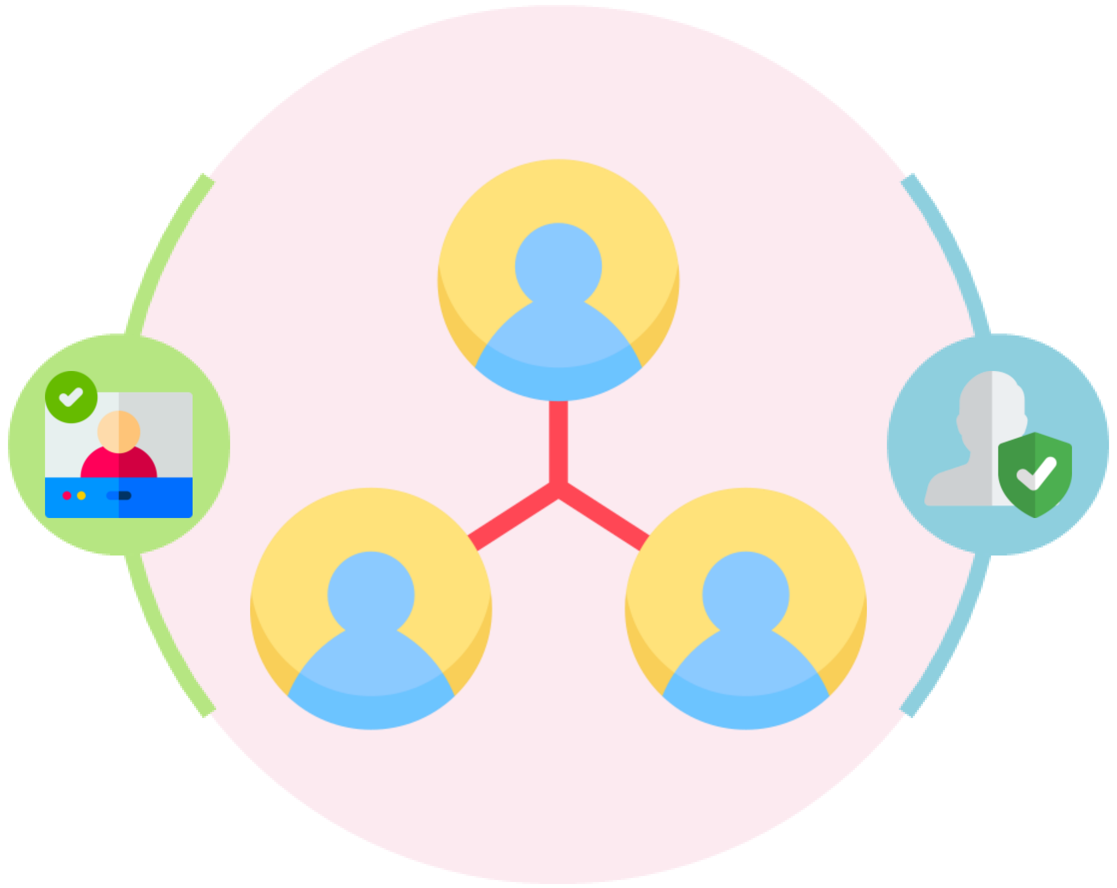
The code does not need to manage the authentication credentials because it is managed by Azure AD.

Security

Two types of managed identities:

System-assigned managed identities:

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



User-assigned managed identities:

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

Azure Kubernetes Service

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

Azure Kubernetes Service

Makes it easier to orchestrate large solutions using a variety of containers:

- Application containers
- Storage containers
- Middleware containers

Manages container-based applications:

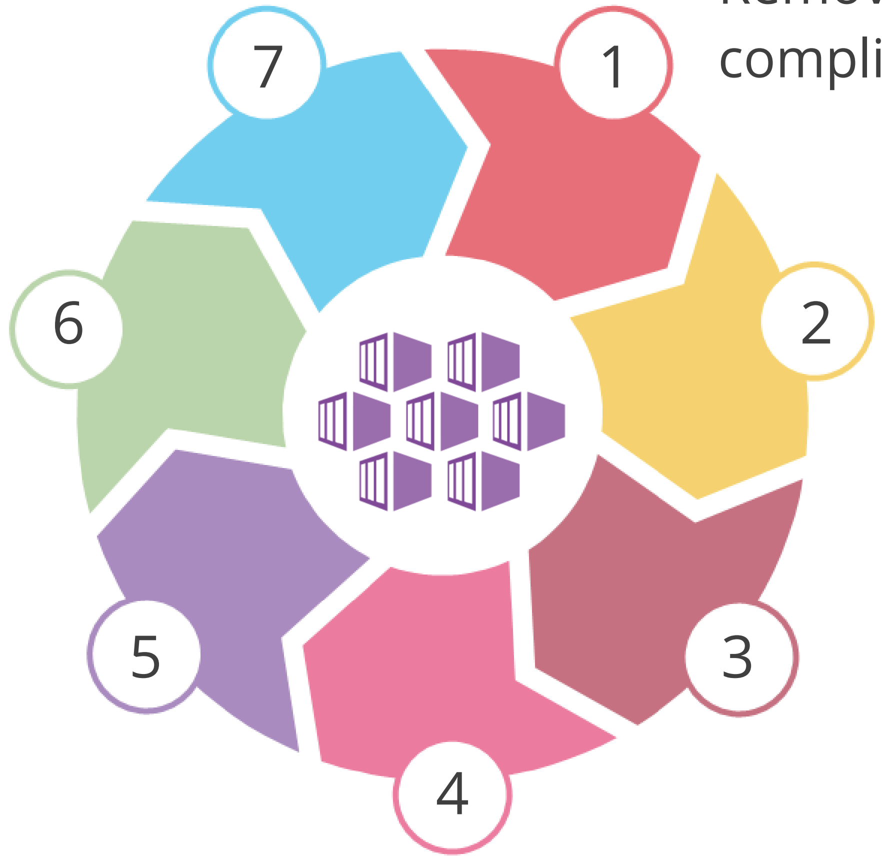
- Along with networking and storage requirements
- Focused on application workloads instead of infrastructure components

Applications are described in a declarative form:

- Use YAML files to describe the application
- Kubernetes handles management and deployment

Azure Kubernetes Service Features

Simple management of a cluster of VMs by using Kubernetes:



1. Removes infrastructure complication and planning
2. No cluster charges, just used resources
3. Supports in-place upgrade of clusters to the latest version of Kubernetes
4. Supports node autoscaling to meet resource utilization
5. Secure, reliable, highly scalable
6. Direct integration with Azure Container Registry for Docker images
7. Azure Virtual Network integration for isolated network traffic

Assisted Practice

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

Problem Statement:

You've been asked to provide your organization with an Azure application architecture solution that facilitates the deployment of a managed Kubernetes cluster on Azure by offloading the operational overhead to Azure as an Azure Architect.

Assisted Practice: Guidelines

Steps to create Kubernetes service are:

1. Login to your Azure portal
2. Create a resource
3. Enable Kubernetes Service
4. Configure the Basic page and Authentication Page
5. Deploy Kubernetes service

Unassisted Practice

Azure App Service Plan **Duration: 10 Min.**

Problem Statement:

Demonstrate a solution for an Azure application architecture that defines a set of computing resources for a web app to run on.

Unassisted Practice: Guidelines

Steps to configure Azure App Service plan are:

1. Login to your Azure portal
2. Click on Create a resource
3. Search for and select Web App
4. Provide the Instance Details before configuring the App Service plan
5. In the App Service Plan section, configure the plan

Unassisted Practice

Create an App Service for Container **Duration: 10 Min.**

Problem Statement:

As an Azure Architect, you've been asked to provide your company with an Azure application architecture solution that allows you to easily deploy and run containerized applications on both Windows and Linux.

Unassisted Practice: Guidelines

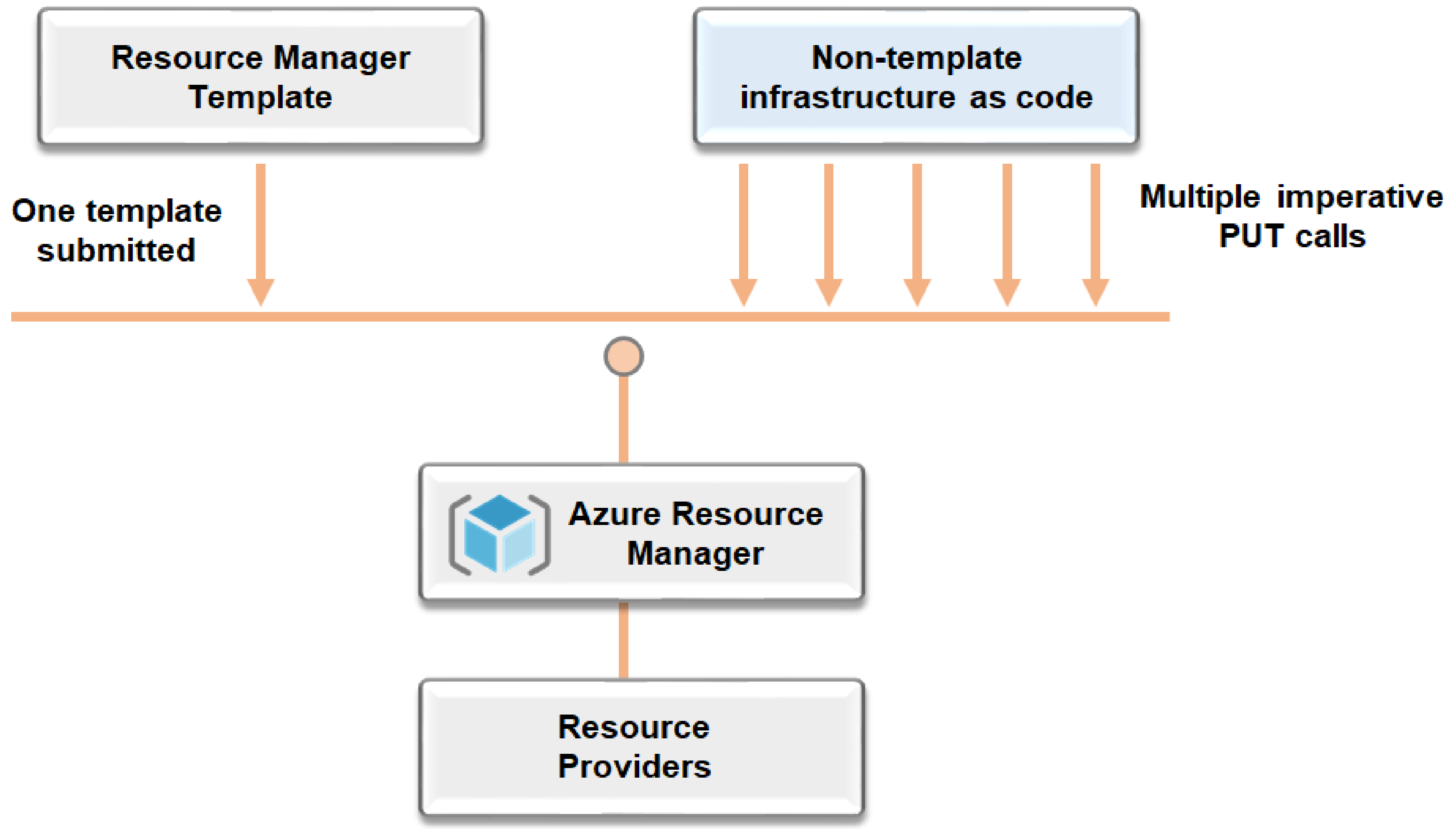
Steps to create an App Service for Container are:

1. Go to the Azure portal
2. Click on create a resource
3. Search for and select Web App
4. Create a web app by providing required details

Recommend an Orchestration Solution for Deployment and Maintenance of Applications

Azure Resource Manager Templates

Azure Resource Manager templates are used to implement infrastructure as code for Azure solutions.

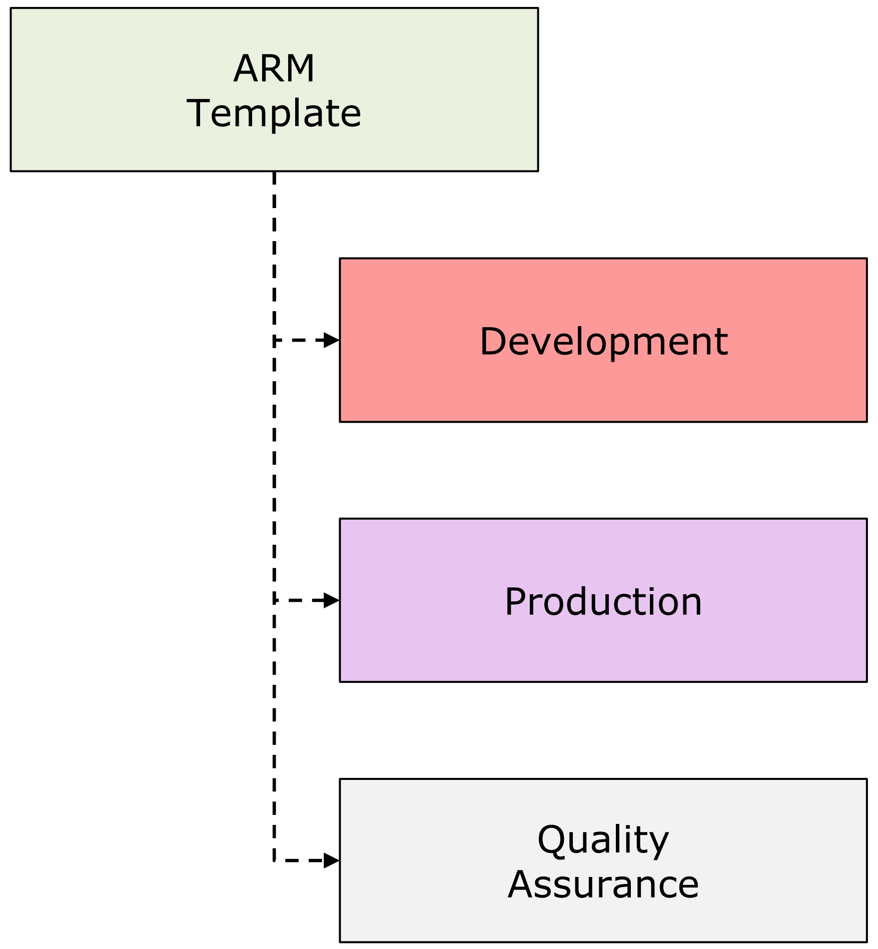

Resource Manager Template
Non-template infrastructure as code
One template submitted
Multiple imperative PUT calls
Azure Resource Manager
Resource Providers

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

These are the template advantages:

- Improves consistency
- Express complex deployments
- Reduce manual tasks
- Reduce error-prone tasks
- Promotes reuse
- Modular and can be linked
- Simplifies orchestration
- Express requirements through code

Template Design

Depending on how they wish to manage the solution, users can create templates and resource groups.

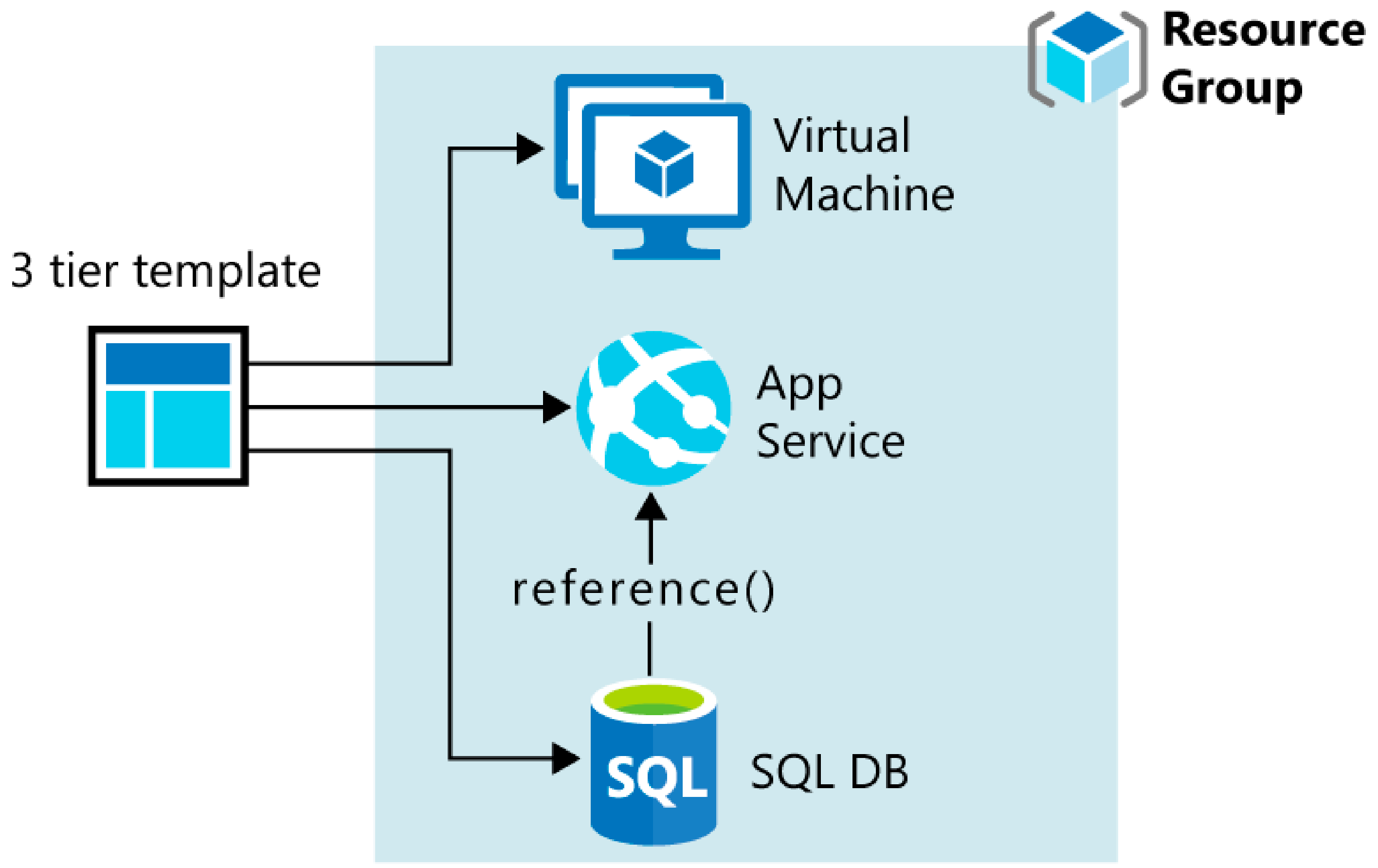


Image source: https://docs.microsoft.com/en-in/

Template Design

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

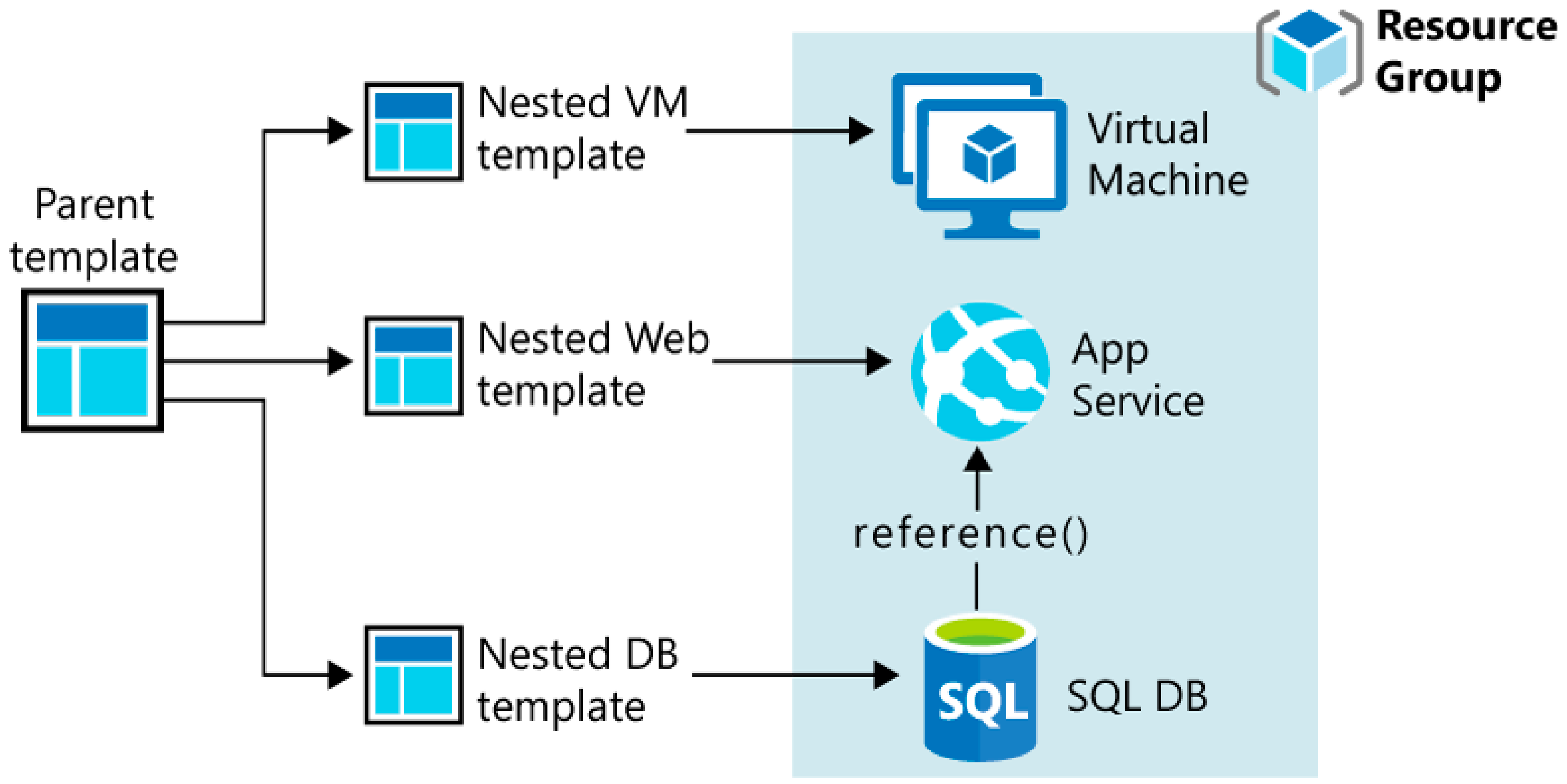


Image source: https://docs.microsoft.com/en-in/

Template Design

If the user tiers have separate lifecycles, the user can deploy his three tiers to separate resources.

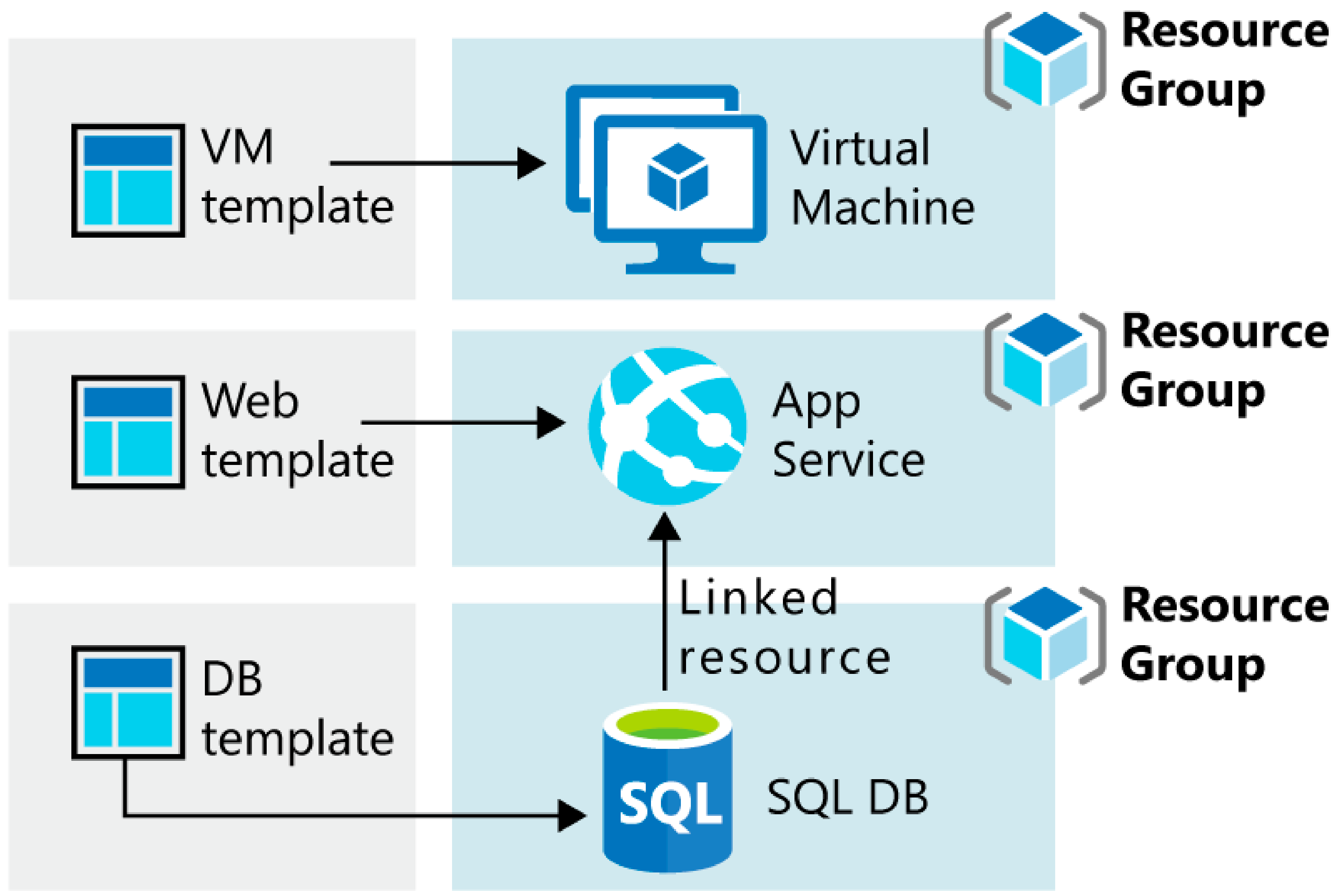


Image source: https://docs.microsoft.com/en-in/

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 an 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

The templates can be deployed by these options:

Using Azure Portal

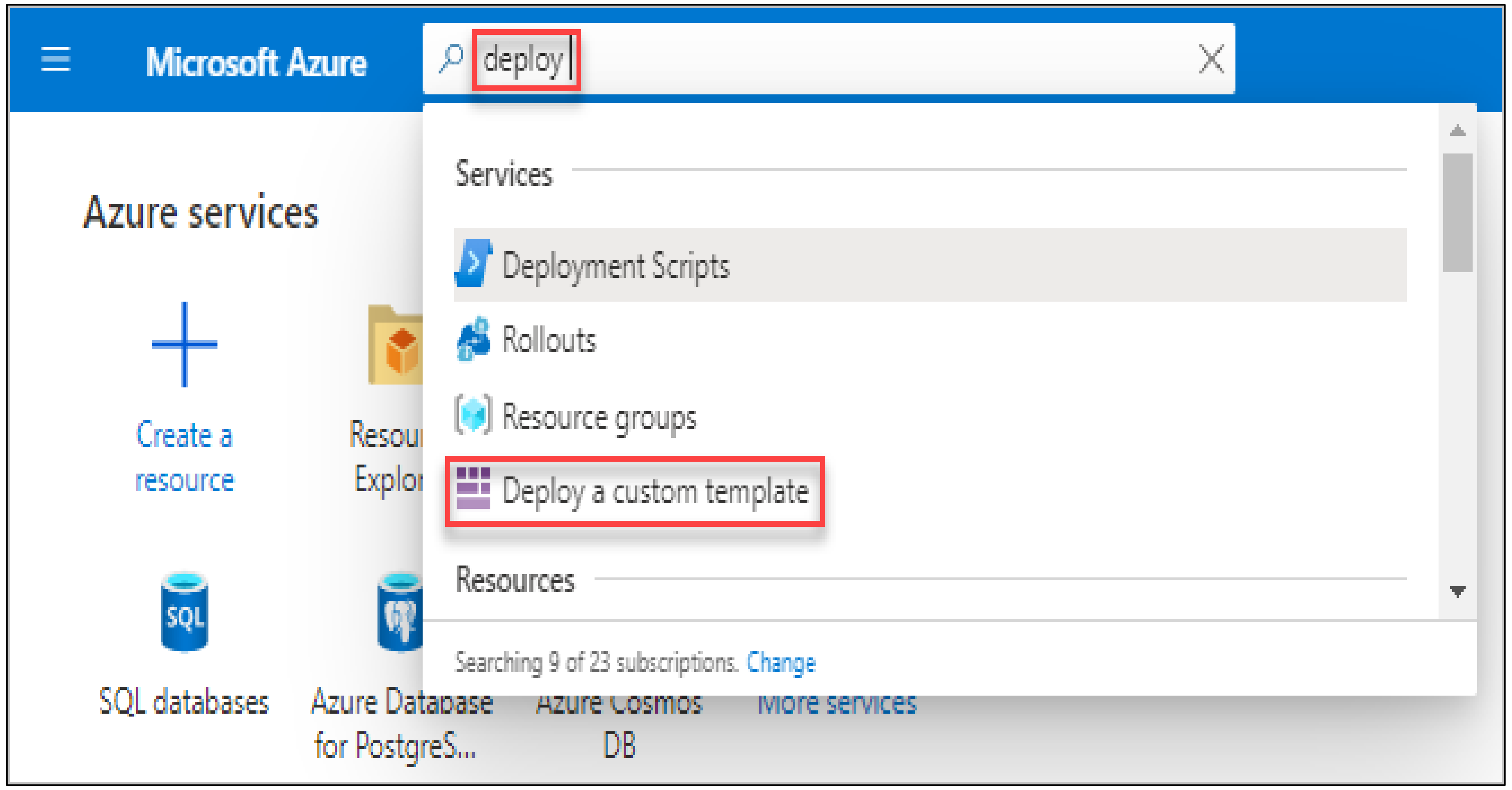
Using PowerShell

Using CLI

Creating ARM template using VS code

Template Deployment Options

The user can deploy the custom template from the Azure portal.



Template Deployment Options

The user 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

The user 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 lifecycle of a runbook job for different types of runbooks:

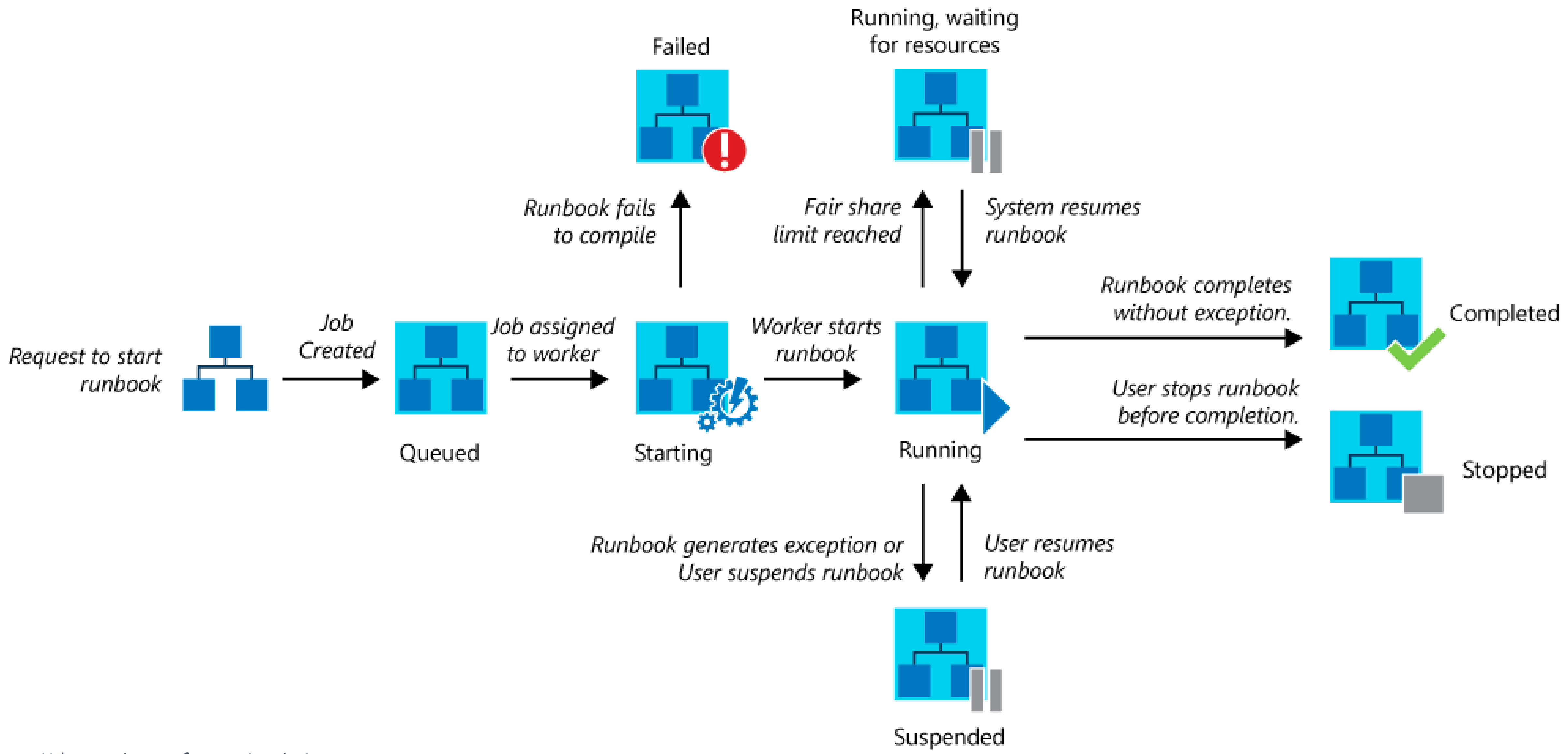


Image source: https://docs.microsoft.com/en-in/

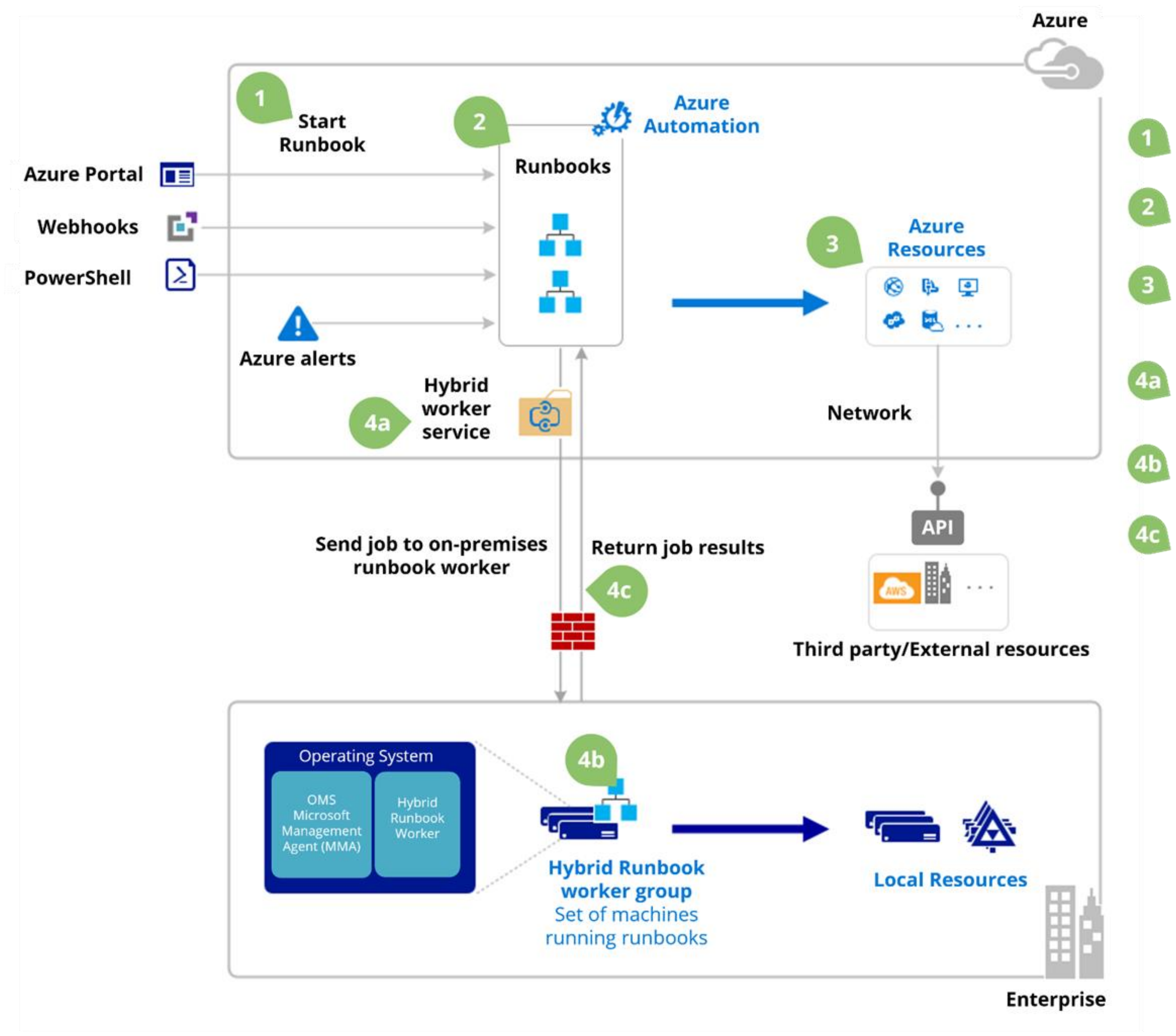
Runbooks in Azure Automation

Azure automation runbooks can run in:

1. An Azure sandbox
2. A hybrid runbook worker

Start a Runbook in Azure Automation

The following diagram shows the life cycle of a runbook:



1. An actor starts a runbook
2. Azure automation notes that a runbook should be started
3. Cloud resources – Runbook acts on local Azure resources or other external resources reachable via the network
4a. On-premises – Hybrid runbook group sends the runbook to an on-premises machine to run
4b. Runbook acts on its local networked resources
4c. Job results are returned

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

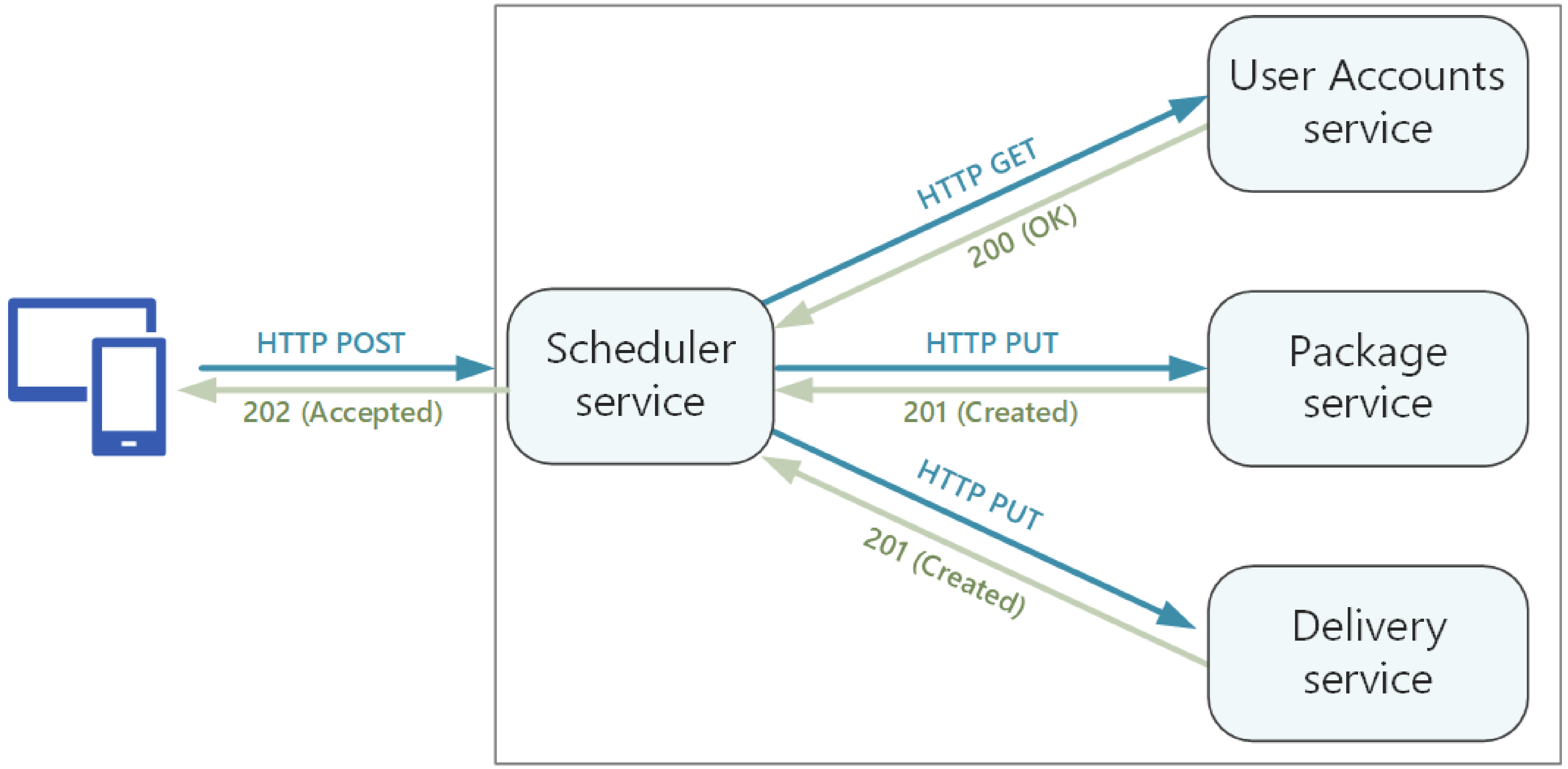
Steps to create an ARM template are:

1. Login 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.

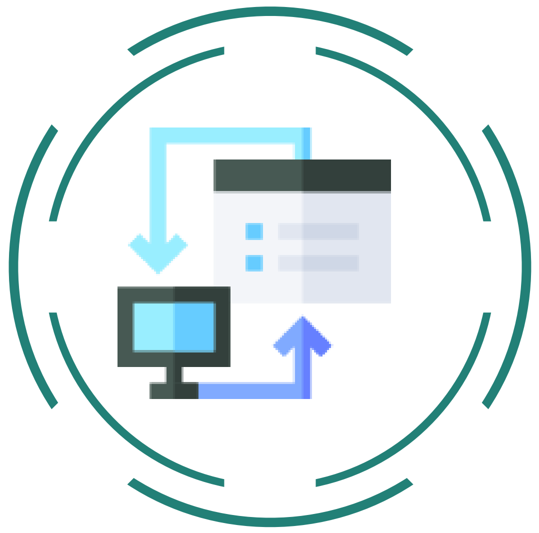


All APIs must have well-defined semantics and versioning schemes so that updates don't break other services.

APIs Types

There are two types of APIs:

Public APIs



Backend APIs

APIs Considerations

REST

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

RPC

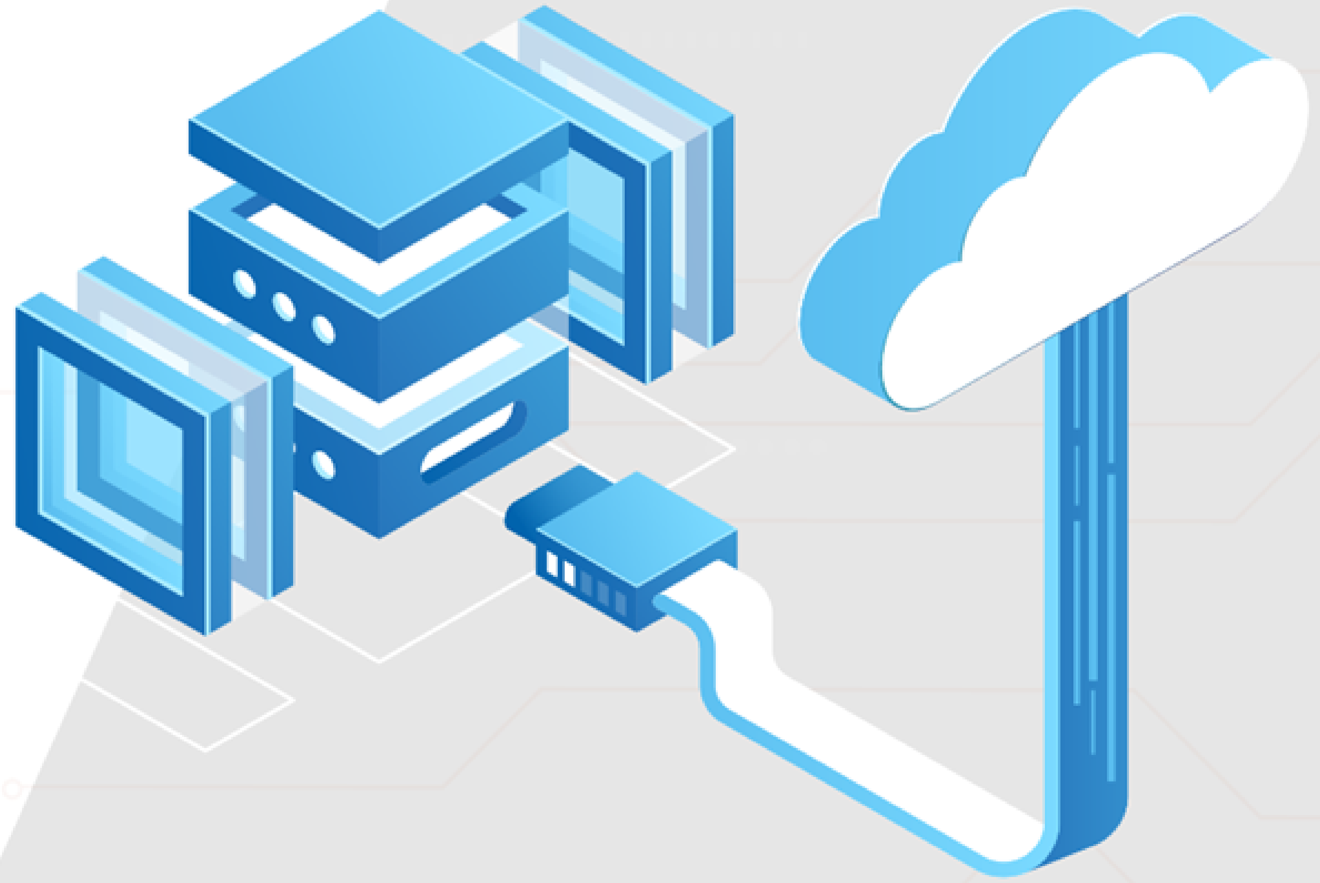
Based on operations or commands. RPC interfaces can affect the design with overly chatty APIs

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 your 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.



Thank you