**Challenge 1**  
  
Creating a 3-tier environment in Azure involves setting up infrastructure for a web application that consists of three tiers: a presentation tier, an application tier, and a data tier. Below, I'll guide you through creating these resources using Azure services.

1. **Presentation Tier (Web Tier):** This tier handles user interface and web client interaction.
   * **Resource**: Azure App Service
     1. Go to the Azure Portal.
     2. Click on "Create a resource" > "Web" > "Web App."
     3. Follow the wizard to configure your web app. You'll need to specify the runtime stack, OS, and other settings.
2. **Application Tier (Logic Tier):** This tier contains the application logic and services.
   * **Resource**: Azure Kubernetes Service (AKS)
     1. Go to the Azure Portal.
     2. Click on "Create a resource" > "Containers" > "Kubernetes Service."
     3. Follow the wizard to configure your AKS cluster. You can deploy containerized applications here.
3. **Data Tier (Database Tier):** This tier manages data storage and database services.
   * **Resource**: Azure SQL Database
     1. Go to the Azure Portal.
     2. Click on "Create a resource" > "Databases" > "SQL Database."
     3. Follow the wizard to configure your SQL database. You can choose the performance tier, security settings, and connection details.
4. **Connect the Tiers:**
   * To create a functional 3-tier environment, you need to ensure that the tiers can communicate effectively.
     1. **Application Tier to Data Tier:** Your application tier (AKS) should have the connection strings and configurations to connect to the database tier (Azure SQL Database). You may need to set up the necessary firewall rules and credentials to secure the connection.
     2. **Presentation Tier to Application Tier:** The web tier (Azure App Service) can communicate with the application tier (AKS) to access the application's logic and services. You can use the AKS cluster's public IP or domain for this purpose. Alternatively, you can set up an Azure Application Gateway for more advanced routing and load balancing.

**Challenge-2**

PowerShell to retrieve Azure instance metadata and convert it into a JSON format:

|  |
| --- |
| # Send a GET request to the Azure Instance Metadata service$metadata = Invoke-RestMethod -Uri "http://169.254.169.254/metadata/instance?api-version=2020-09-01" -Headers @{"Metadata"="true"}  # Convert the metadata to JSON  $jsonMetadata = $metadata | ConvertTo-Json  # Print the JSON formatted metadata  $jsonMetadata |

**Challenge 3**

Java script

|  |
| --- |
| function getValueByNestedKey(object, key) {  const keys = key.split('/');  let value = object;  for (const k of keys) {  if (value && typeof value === 'object' && k in value) {  value = value[k];  } else {  return undefined; // Key not found  }  }  return value;  } |
| // Example Usage:  const object1 = {"a":{"b":{"c":"d"}};  const key1 = "a/b/c";  const object2 = {"x":{"y":{"z":"a"}};  const key2 = "x/y/z";  const value1 = getValueByNestedKey(object1, key1); // Returns "d"  const value2 = getValueByNestedKey(object2, key2); // Returns "a" | | |

This **getValueByNestedKey** function takes the object and key as input and iterates through the nested structure using the keys in the path, returning the final value if found or **undefined** if the key path is not valid.