Summer Intern Project

**Extracting Text From Images**

Using Azure Computer Vision

horizontal line

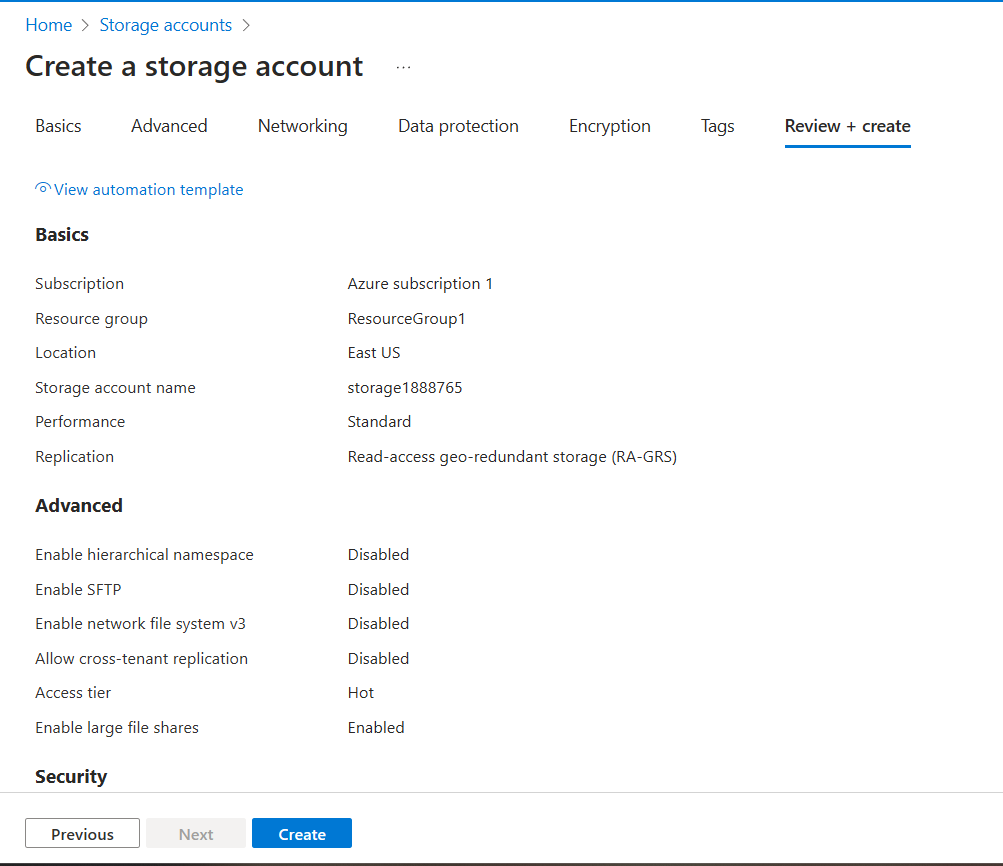
# Placeholder image

# Introduction

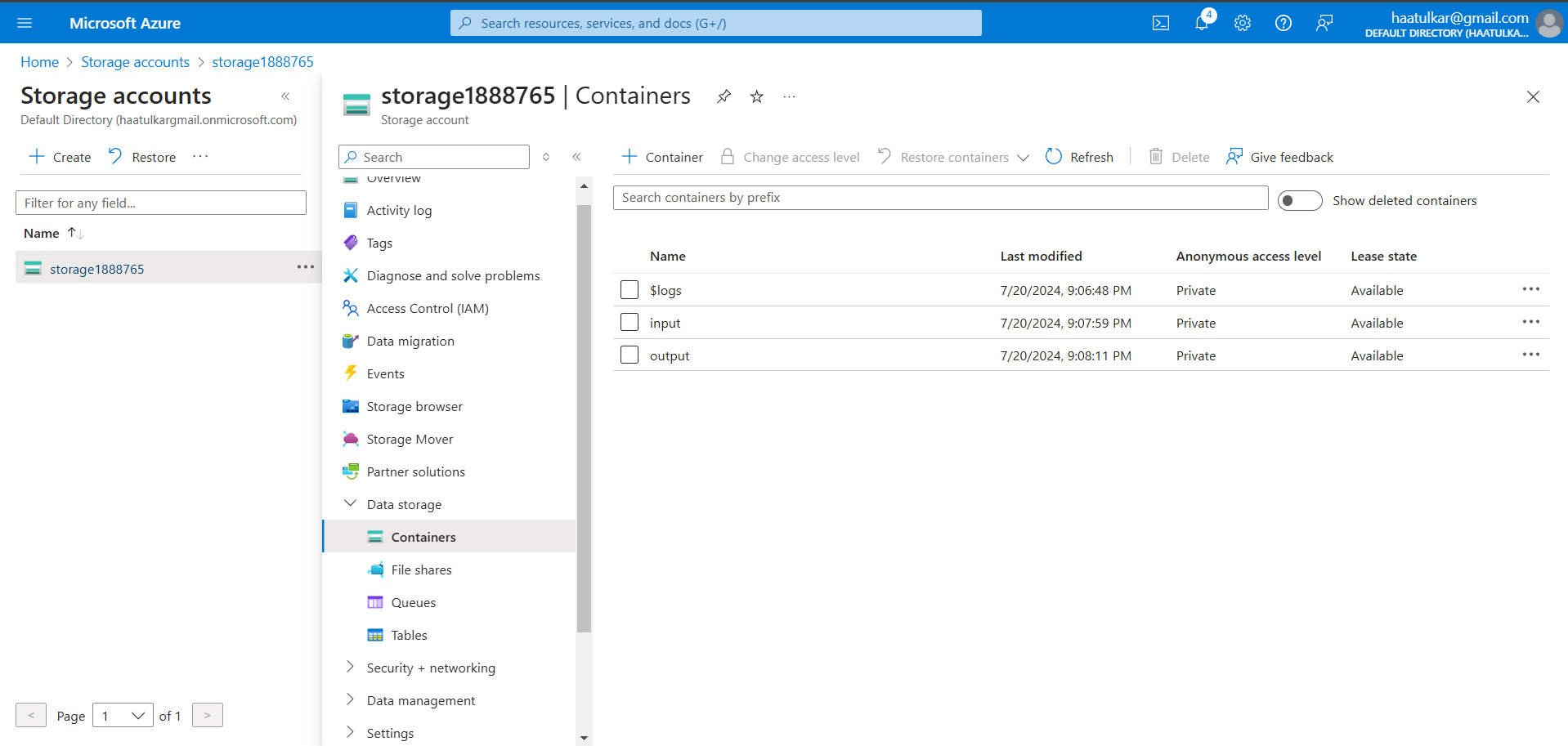
This project aims to Extract the handwritten or printed text from images in format that can be useful to get insights from or perform enhancements on top of it. Images should be fetched from the input Azure Blob container. Language to be used: Python. JSON Output file should be uploaded to blob in the output container. Application should be accessible only on a private network. Automate the task that whenever a new input image has been uploaded it automatically processes it and generates email alerts after the output file has been generated.

# Step 1: Set Up Azure Blob Storage

1. **Create a Storage Account**:
   * Navigate to the Azure portal.
   * Click on "Create a resource" and search for "Storage account".
   * Click on "Create" and fill in the necessary details (Subscription, Resource group, Storage account name, etc.).
   * Click "Review + create" and then "Create".

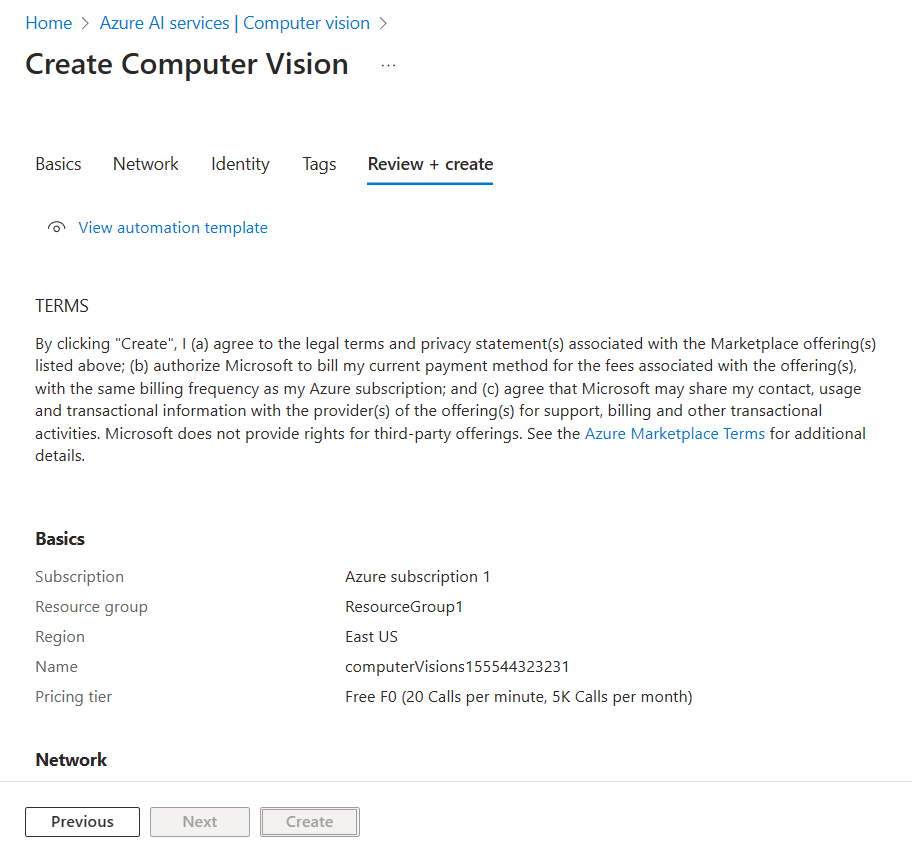


1. **Create Containers**:
   * Once the storage account is created, go to the storage account.
   * Under the "Data storage" section, click on "Containers".
   * Click on "+ Container" to create a new container.
   * Name one container input and another output.
   * Set the Public access level to "Private (no anonymous access)".
   * Click "Create".

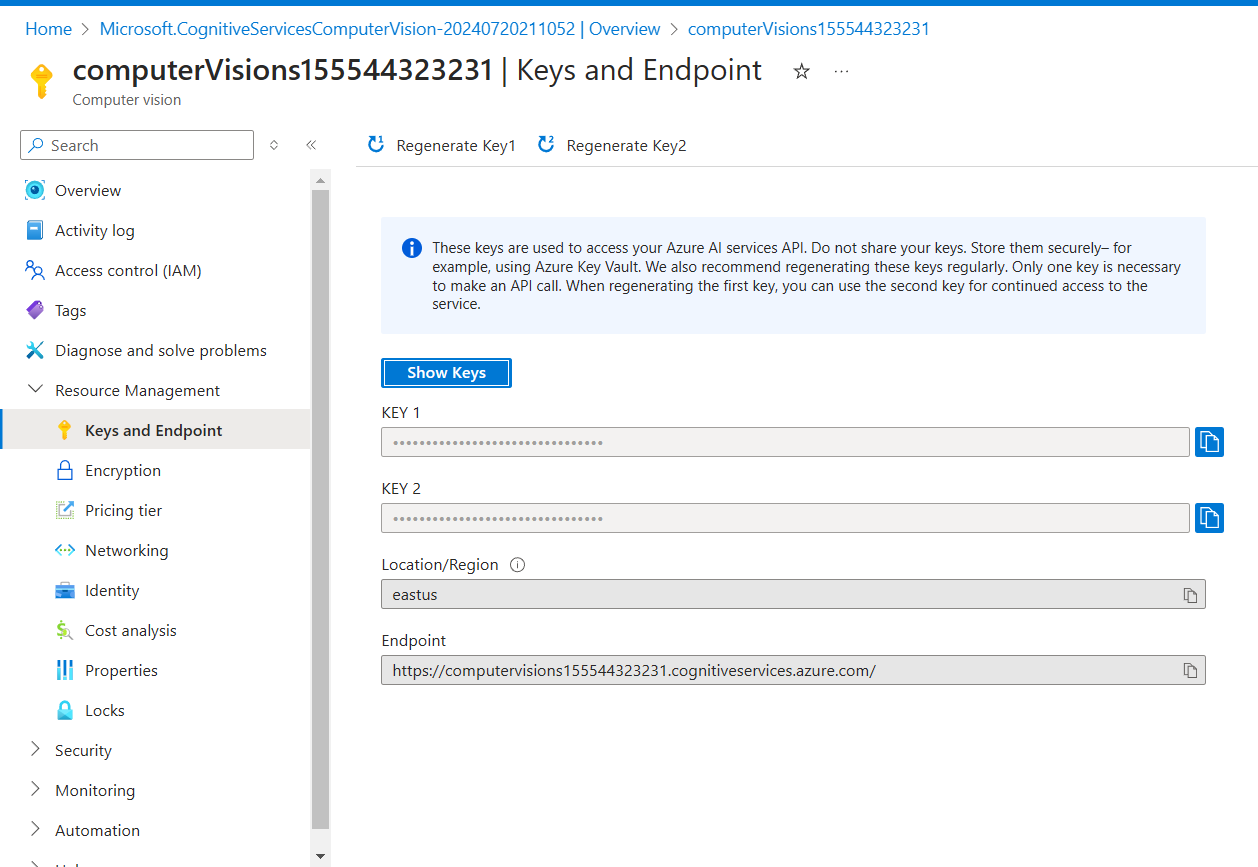


# Step 2: Set Up Azure Cognitive Services (Computer Vision)

1. **Create a Computer Vision Resource**:
   * Navigate to the Azure portal.
   * Click on "Create a resource" and search for "Computer Vision".
   * Click on "Create" and fill in the necessary details (Subscription, Resource group, Name, Region, Pricing tier, etc.).
   * Click "Review + create" and then "Create".

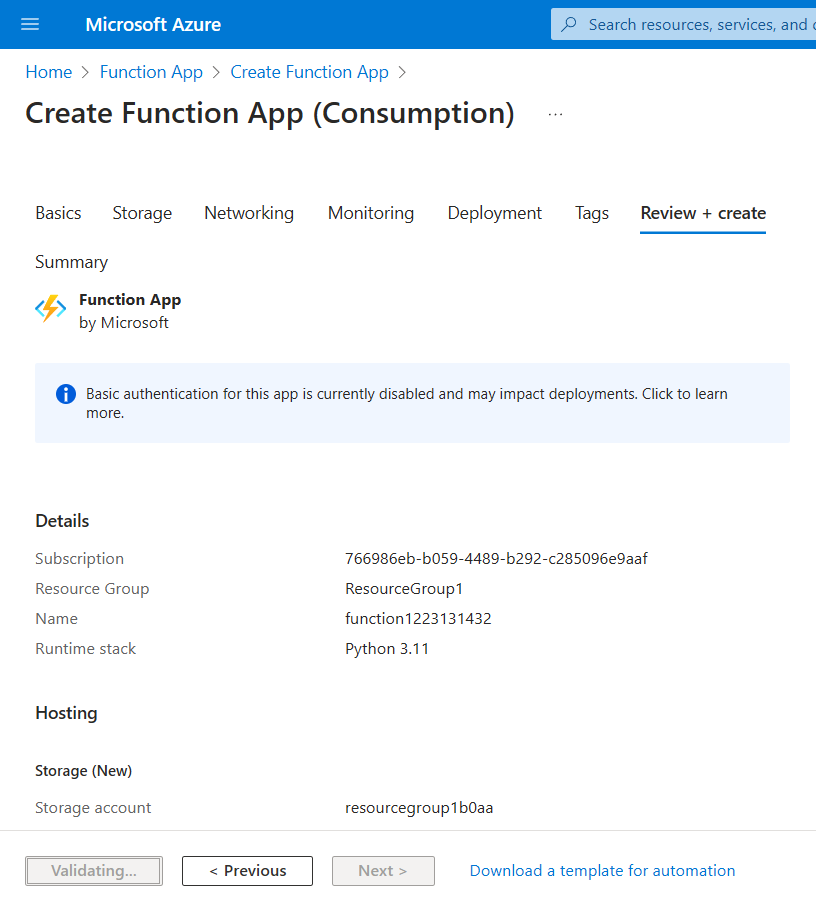


1. **Get Endpoint and Key**:
   * Once the resource is created, go to the resource.
   * Under the "Resource Management" section, click on "Keys and Endpoint".
   * Copy the Endpoint URL and one of the keys. You will need these for your Azure Function.

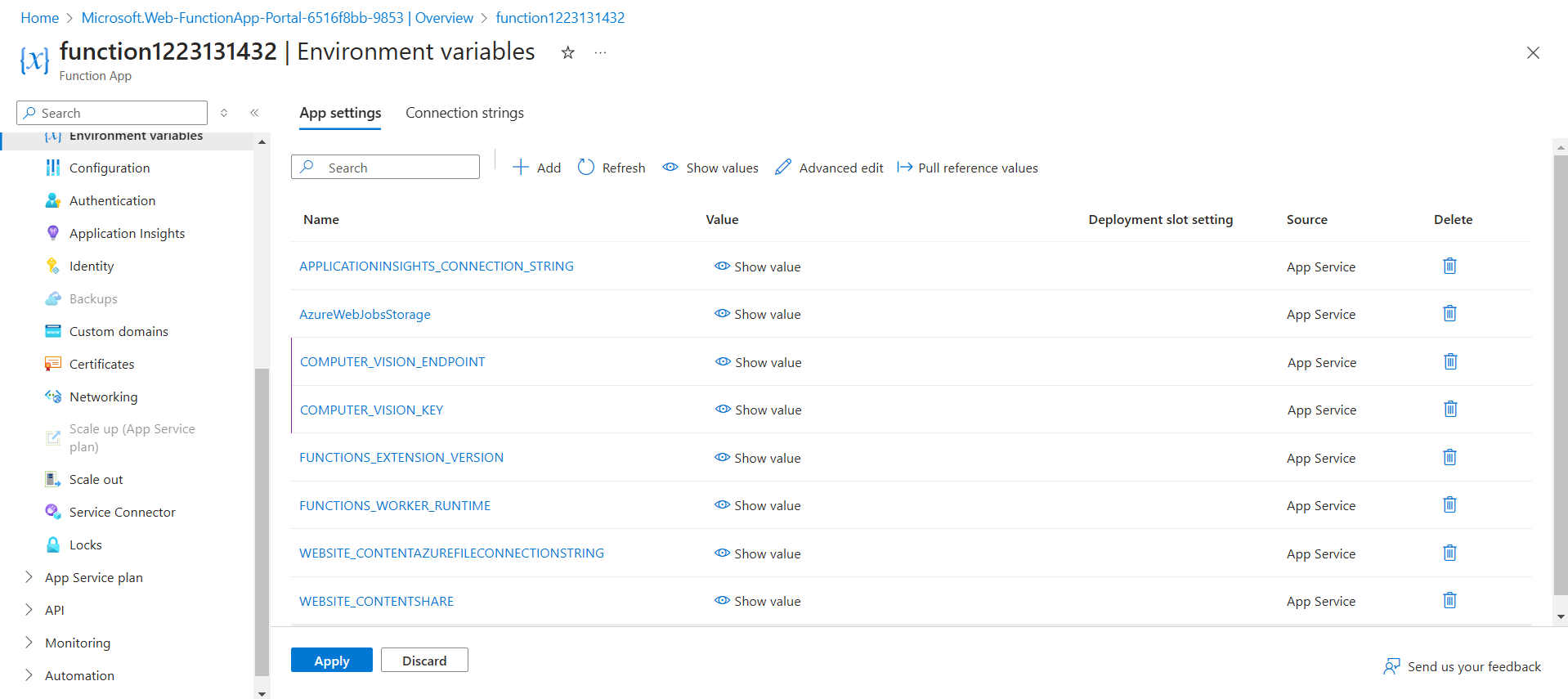


# Step 3: Set Up Azure Functions

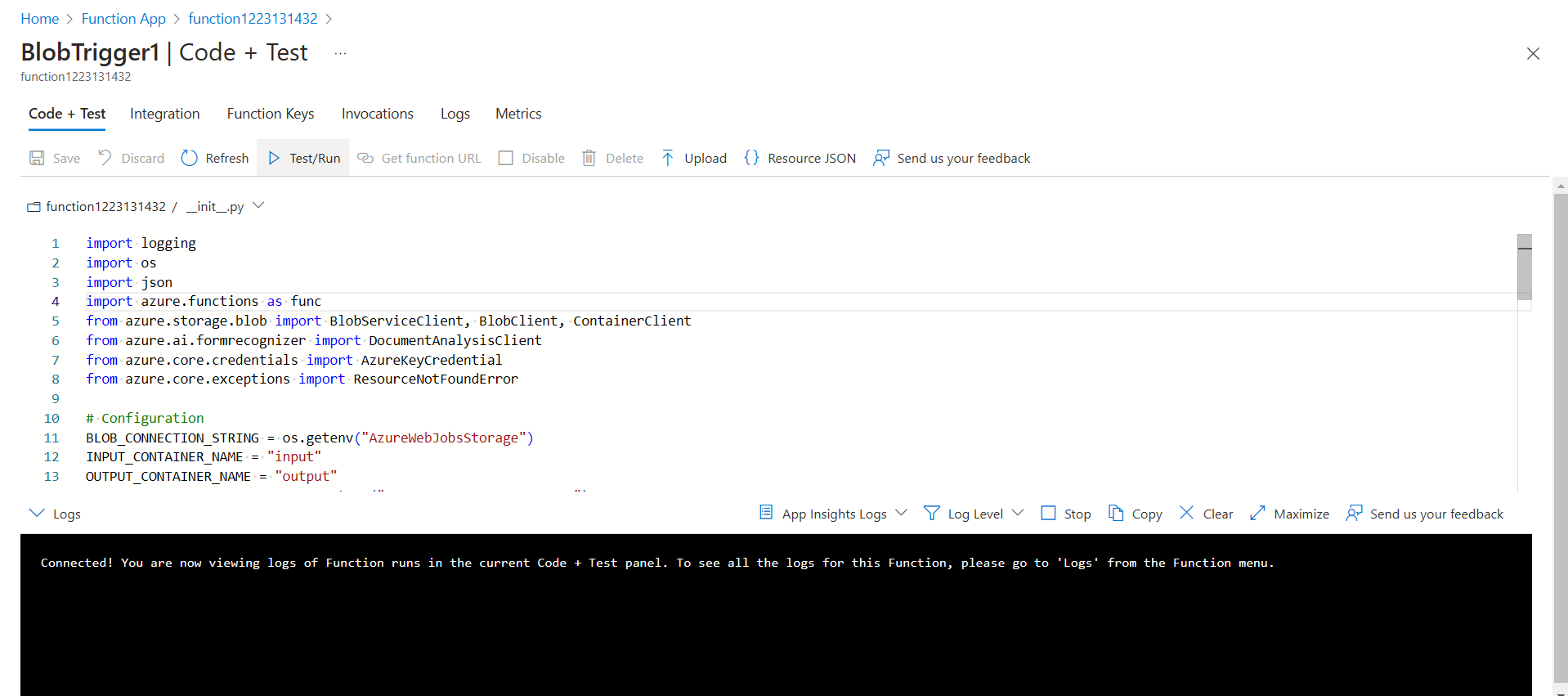
1. **Create a Function App**:
   * Navigate to the Azure portal.
   * Click on "Create a resource" and search for "Function App".
   * Click on "Create" and fill in the necessary details (Subscription, Resource group, Function App name, Publish as Code, Runtime stack as Python, Region, etc.).
   * Click "Review + create" and then "Create".



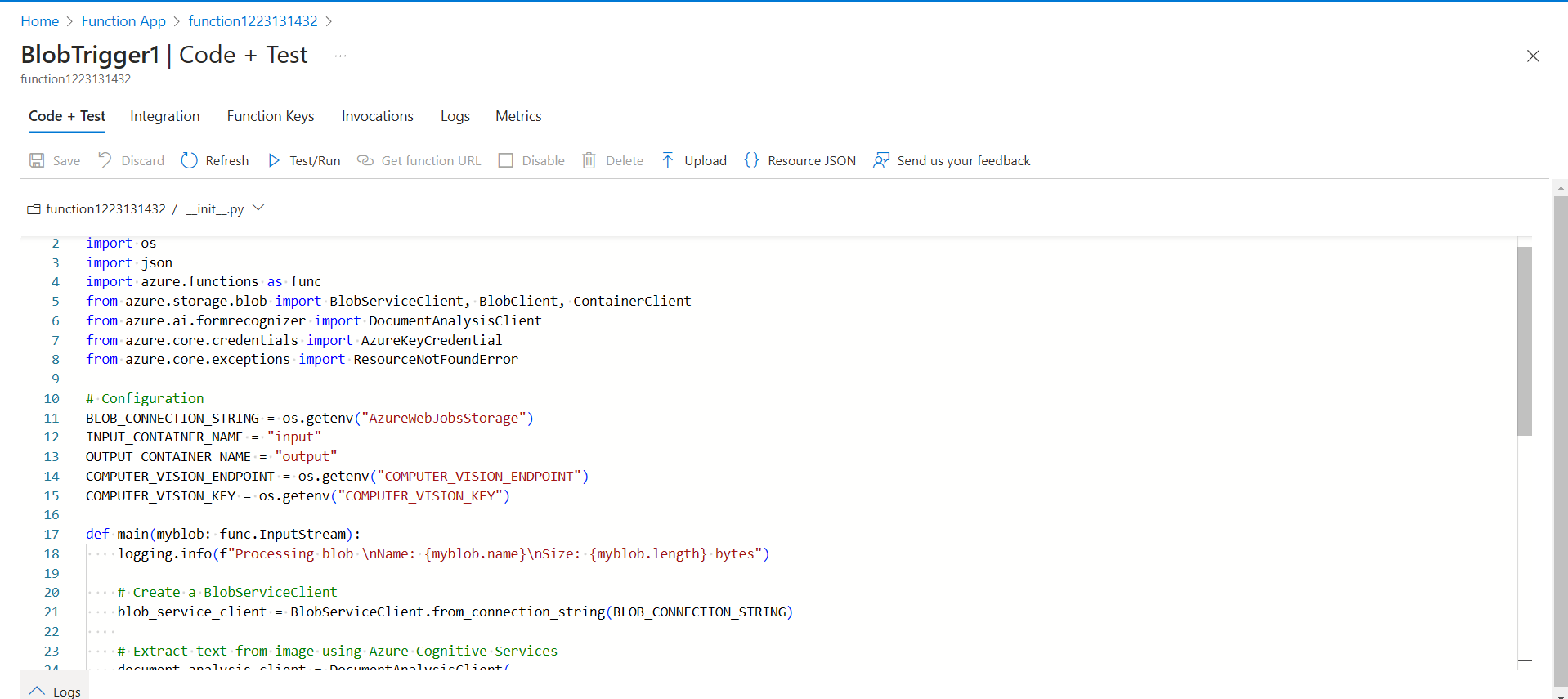
1. **Configure Function App**:
   * Once the Function App is created, go to the Function App.
   * Click on "Configuration" under the "Settings" section.
   * Add the following application settings:
     + AzureWebJobsStorage: Connection string for your Azure Storage account.
     + COMPUTER\_VISION\_ENDPOINT: Endpoint URL for your Computer Vision resource.
     + COMPUTER\_VISION\_KEY: Key for your Computer Vision resource.



1. **Create a New Function**:
   * In the Function App, click on "Functions" and then "+ Add".
   * Select "Azure Blob Storage trigger" and provide the necessary details (Function name, Path to the blob container as input/{name}, Storage account connection as the one you created).
   * Click "Create".

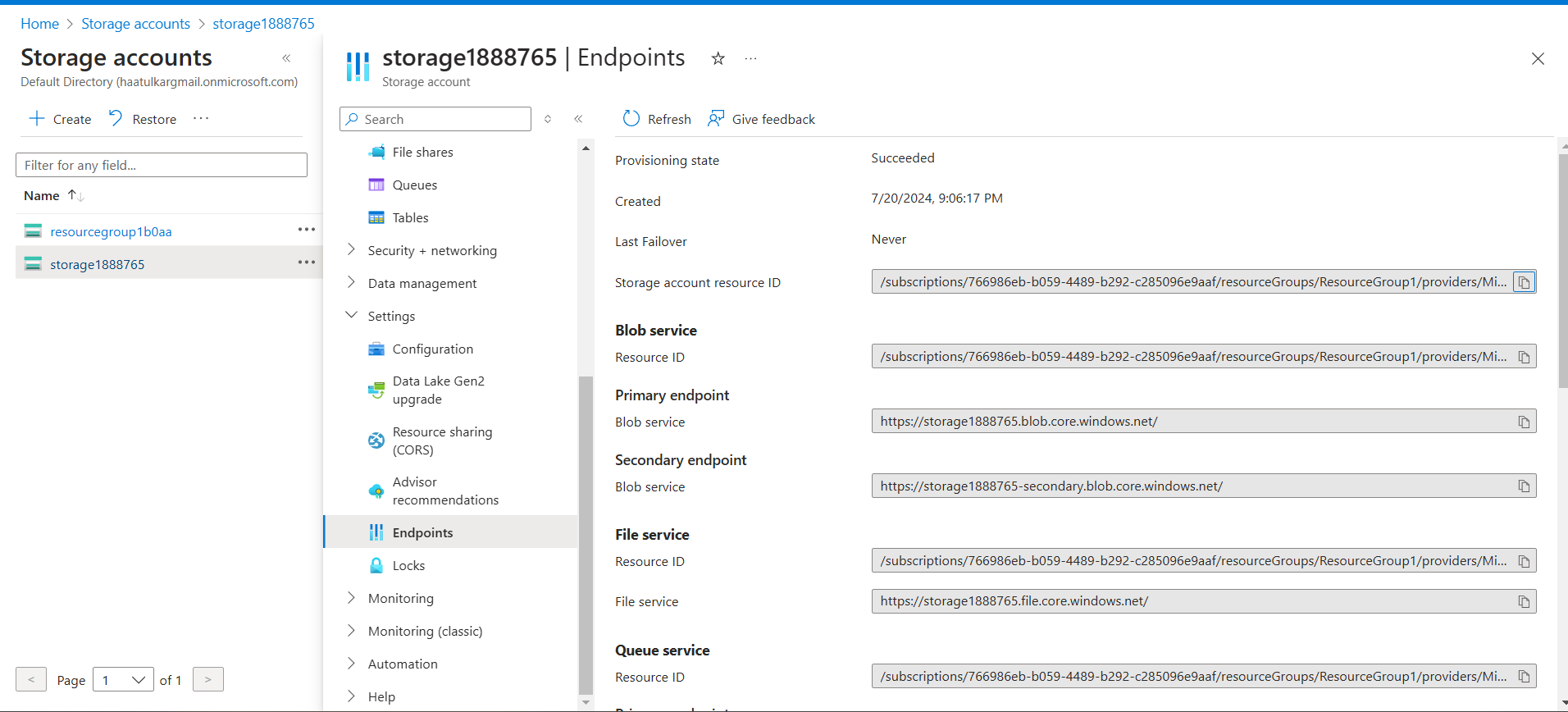


1. **Upload the Function Code**:
   * Use the Azure portal or an integrated development environment (IDE) to upload the function code (\_\_init\_\_.py) .
   * Ensure that your function code uses the appropriate libraries and imports.



# Step 4: Configure Networking

1. **Set Up VNet Integration**:
   * Navigate to the Function App in the Azure portal.
   * Under the "Settings" section, click on "Networking".
   * Click on "VNet Integration" and add the VNet you want to integrate with your Function App.
2. **Set Up Private Endpoints**:
   * Navigate to the storage account.
   * Under the "Settings" section, click on "Private endpoint connections".
   * Click on "+ Private endpoint".
   * Follow the wizard to create a private endpoint and link it to the VNet and subnet you configured for the Function App.



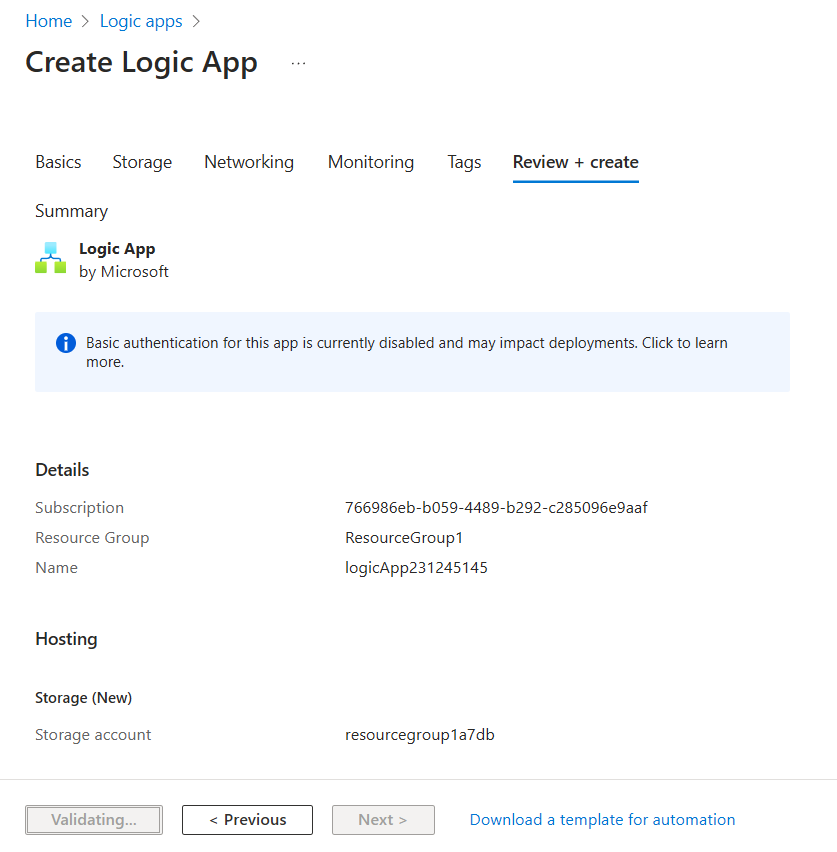
1. **Restrict Public Network Access**:
   * For both the Function App and the storage account, ensure that network access is restricted to the private network.
   * For the storage account:
     + Under "Networking", select "Selected networks".
     + Add your VNet to the allowed networks.
   * For the Function App:
     + Ensure that the VNet integration is properly configured and public access is restricted.

# Step 5: Automate and Notify

To automate the task of sending email alerts after the output file has been generated, we can use Azure Logic Apps. Here's how to set it up:

### Create an Azure Logic App

1. **Create a Logic App**:
   * Navigate to the Azure portal.
   * Click on "Create a resource" and search for "Logic App".
   * Click on "Create" and fill in the necessary details (Subscription, Resource group, Logic App name, Region, etc.).
   * Click "Review + create" and then "Create".



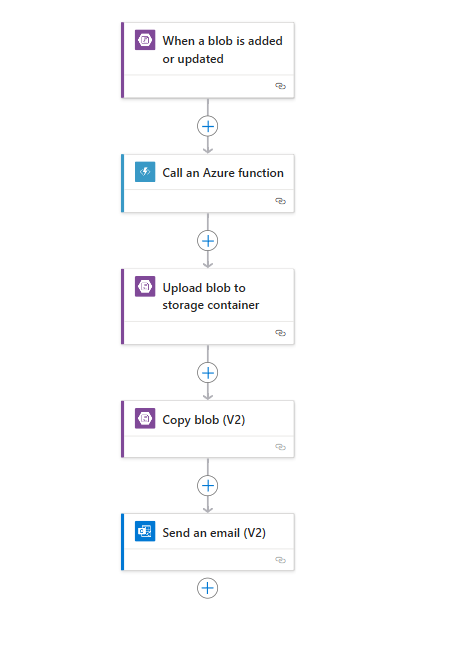
1. **Set Up the Trigger**:
   * Once the Logic App is created, go to the Logic App.
   * In the Logic App Designer, choose the "When a blob is added or modified (properties only)" trigger from the Azure Blob Storage connector.
   * Provide the connection details for your storage account (you might need to create a connection).
   * Set the Container name to output.
2. **Add an Action to Get Blob Content**:
   * Click on "New step" and search for "Azure Blob Storage".
   * Select "Get blob content" and provide the necessary details:
     + Container name: output
     + Blob name: Use the dynamic content selector to pick the "Path" property from the trigger.
3. **Add an Action to Send an Email**:
   * Click on "New step" and search for "Send an email".
   * You can use Office 365 Outlook, SMTP, or any other email service connector available.
   * Sign in to your email service account and provide the necessary permissions.
   * Configure the email action:
     + To: Your email address or the recipients you want to notify.
     + Subject: Set a suitable subject, such as "New JSON Output File Generated".
     + Body: Include relevant details. You can use dynamic content to include the blob name, URL, or other properties.

A new JSON output file has been generated and uploaded to the output container.

Blob Name: @{triggerOutputs()?['body']['Name']}

Blob URL: https://<your\_storage\_account>.blob.core.windows.net/output/@{triggerOutputs()?['body']['Name']}

1. **Save and Test the Logic App**:
   * Click on "Save" to save the Logic App.
   * Upload a new image to the input container and wait for the process to complete.
   * Check your email to confirm that the notification has been sent.



### Deployment and Testing

1. **Deploy the Azure Function and Logic App**:
   * Ensure that your Azure Function is correctly deployed and the code is functioning as expected.
   * Test the Function by uploading an image to the input container and verifying that the JSON output file is generated in the output container.
2. **Verify Network Configuration**:
   * Make sure that your Function App and storage account are correctly integrated with your VNet and that public access is restricted.
3. **Test End-to-End Workflow**:
   * Upload an image to the input container.
   * Monitor the Azure Function execution in the Azure portal.
   * Check the output container for the generated JSON file.
   * Verify that the Logic App triggers and sends the email notification.

# Conclusion

This project outlines a comprehensive solution to automate the process of extracting text from images, generating JSON output, and sending email notifications, leveraging Azure services. By setting up Azure Blob Storage, Azure Cognitive Services, and Azure Functions, we established a robust pipeline that processes images uploaded to an input container. The integration with Azure Logic Apps ensures that stakeholders are promptly notified whenever a new JSON output file is generated.