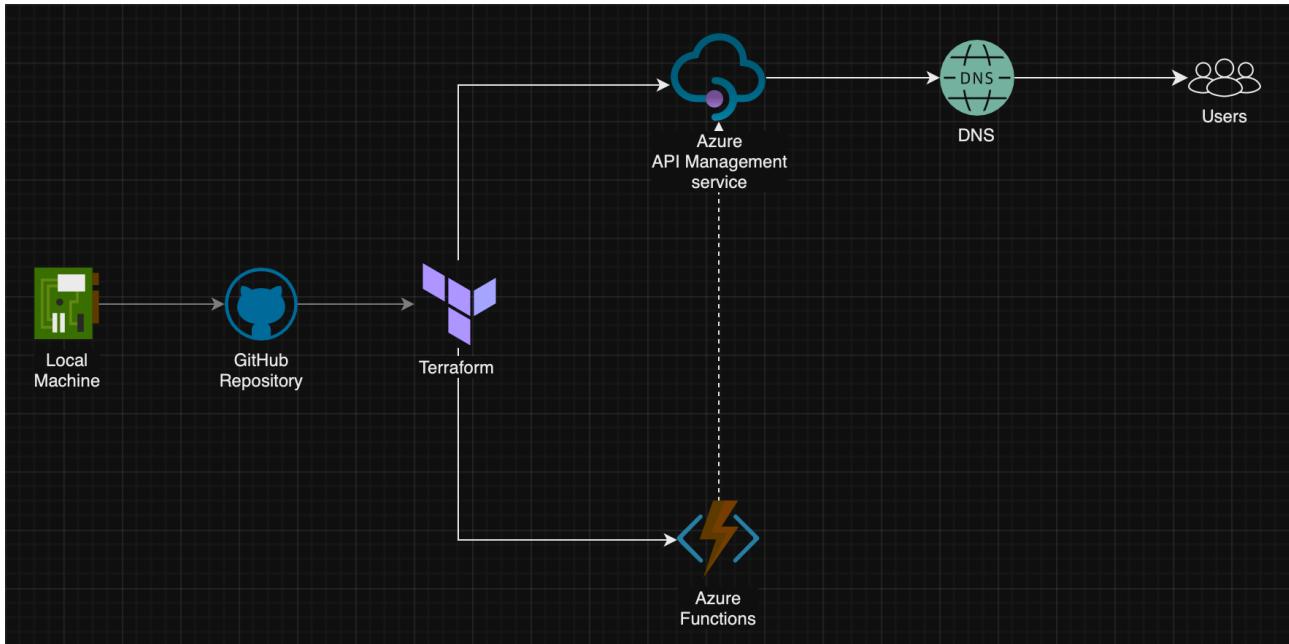


# Assignment result

## 1. A detailed design document outlining the solution architecture.

Architecture:



## 2. A Terraform configuration file that automates the deployment of the APIM and Function App.

Link: [api-management-assignment](#)

Prerequisites (Created Storage Account for Terraform backend configuration):

Screenshot of the Microsoft Azure portal showing the creation of a new storage account. The account is named 'tfstatesurgetg' and is located in the 'tfstaterg' resource group, which is part of the '(New) tfstaterg' project. The storage account is being created in the '(Asia Pacific) Central India' region. The 'Locally-redundant storage (LRS)' redundancy option is selected. The 'Standard' performance tier is chosen. The 'Review + create' button is visible at the bottom.

## Prerequisites (Created Container for Terraform backend configuration):

The screenshot shows the Microsoft Azure Storage Explorer interface. On the left, there's a sidebar with various storage-related options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Storage Mover, Partner solutions, and Data storage. Under Data storage, the 'Containers' option is selected. In the main pane, there's a table showing existing containers: 'Slogs' (Last modified: 31/12/2024, 12:20:38 AM, Private). To the right, a 'New container' dialog is open with the name 'tfstatesurgetr' entered. The 'Anonymous access level' dropdown is set to 'Private (no anonymous access)'. A note at the bottom says 'The access level is set to private because anonymous access is disabled on this storage account.' At the bottom right of the dialog are 'Create' and 'Give feedback' buttons.

## Terraform provider and backend configuration with terraform init:

```

apiManagementAssignment % terraform init
Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v4.14.0...
- Installed hashicorp/azurerm v4.14.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

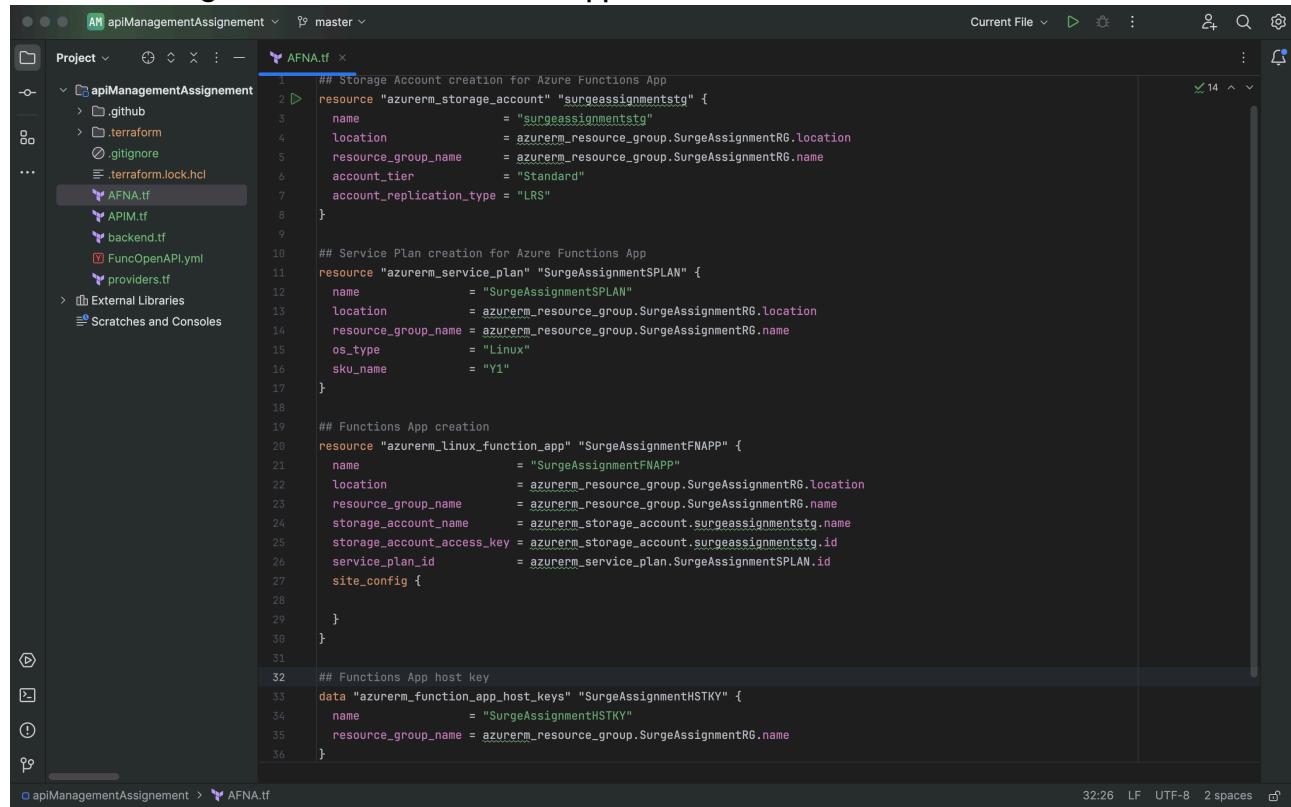
```

## Terraform state file created in Azure storage account container:

The screenshot shows the Microsoft Azure Storage Explorer interface. On the left, there's a sidebar with Overview, Diagnose and solve problems, Access Control (IAM), and Settings. In the main pane, there's a table showing blobs in the 'tfstatesurgetr' container. There is one blob named 'terraform.tfstate' with the following details:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
terraform.tfstate	31/12/2024, 2:20:12 ...	Hot (Inferred)		Block blob	12.04 KiB	Leased

## Terraform configuration for Azure Functions App:



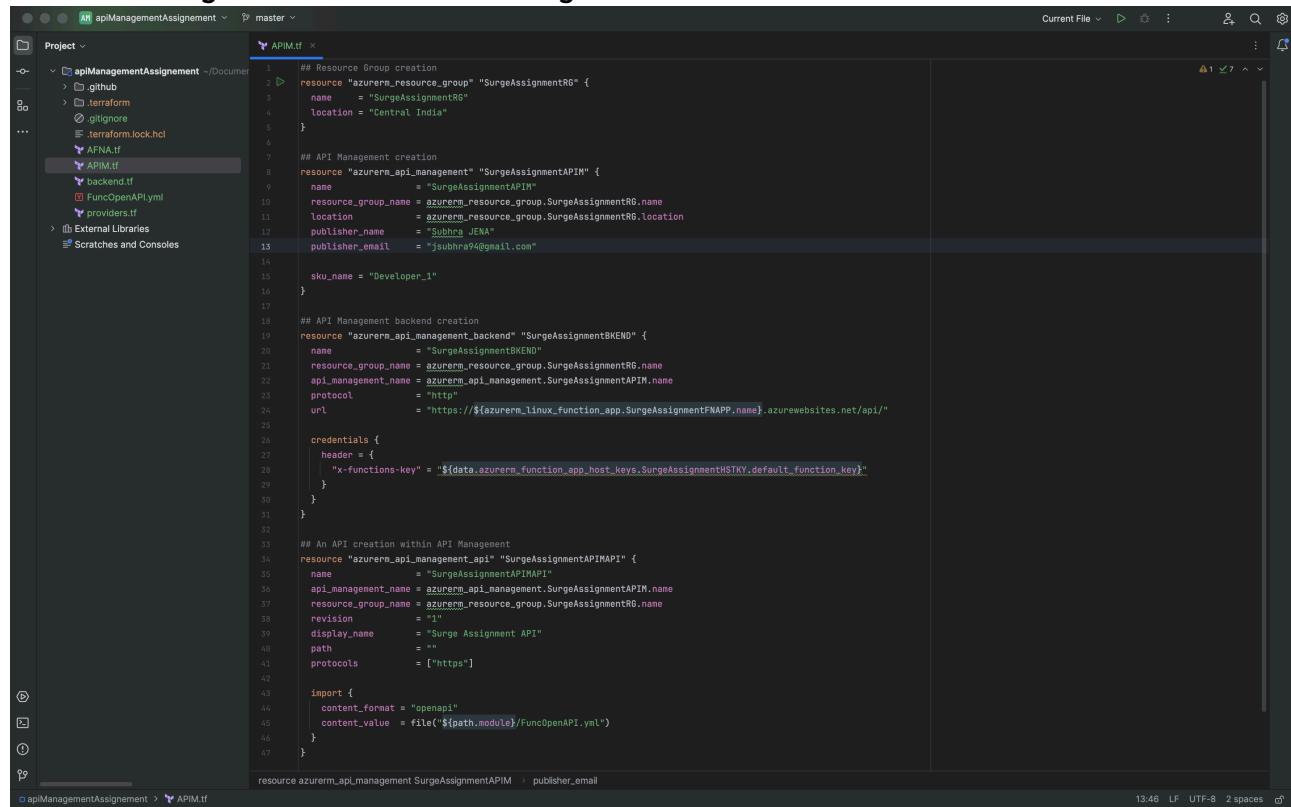
```
## Storage Account creation for Azure Functions App
resource "azurerm_storage_account" "surgeassignmentstg" {
    name                = "surgeassignmentstg"
    location            = azurerm_resource_group.SurgeAssignmentRG.location
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    account_tier        = "Standard"
    account_replication_type = "LRS"
}

## Service Plan creation for Azure Functions App
resource "azurerm_service_plan" "SurgeAssignmentSPLAN" {
    name                = "SurgeAssignmentSPLAN"
    location            = azurerm_resource_group.SurgeAssignmentRG.location
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    os_type             = "Linux"
    sku_name            = "Y1"
}

## Functions App creation
resource "azurerm_linux_function_app" "SurgeAssignmentFNAPP" {
    name                = "SurgeAssignmentFNAPP"
    location            = azurerm_resource_group.SurgeAssignmentRG.location
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    storage_account_name = azurerm_storage_account.surgeassignmentstg.name
    storage_account_access_key = azurerm_storage_account.surgeassignmentstg.id
    service_plan_id     = azurerm_service_plan.SurgeAssignmentSPLAN.id
    site_config {
    }
}

## Functions App host key
data "azurerm_function_app_host_keys" "SurgeAssignmentHSTKY" {
    name                = "SurgeAssignmentHSTKY"
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
}
```

## Terraform configuration for Azure API Management Service:



```
## Resource Group creation
resource "azurerm_resource_group" "SurgeAssignmentRG" {
    name                = "SurgeAssignmentRG"
    location            = "Central India"
}

## API Management creation
resource "azurerm_api_management" "SurgeAssignmentAPIM" {
    name                = "SurgeAssignmentAPIM"
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    location            = azurerm_resource_group.SurgeAssignmentRG.location
    publisher_name      = "Subhra JENA"
    publisher_email     = "jsuhbra9@gmail.com"
    sku_name            = "Developer_1"
}

## API Management backend creation
resource "azurerm_api_management_backend" "SurgeAssignmentBKEND" {
    name                = "SurgeAssignmentBKEND"
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    api_management_name = azurerm_api_management.SurgeAssignmentAPIM.name
    protocol            = "https"
    url                = "https://${azurerm_linux_function_app.SurgeAssignmentFNAPP.name}.azurewebsites.net/api/"

    credentials {
        header = {
            "x-functions-key" = "${data.azurerm_function_app_host_keys.SurgeAssignmentHSTKY.default_function_key}"
        }
    }
}

## An API creation within API Management
resource "azurerm_api_management_api" "SurgeAssignmentAPIMAPI" {
    name                = "SurgeAssignmentAPIMAPI"
    api_management_name = azurerm_api_management.SurgeAssignmentAPIM.name
    resource_group_name = azurerm_resource_group.SurgeAssignmentRG.name
    revision           = "1"
    display_name       = "Surge Assignment API"
    path               = ""
    protocols          = ["https"]

    import {
        content_format = "openapi"
        content_value  = file("${path.module}/FuncOpenAPI.yaml")
    }
}

resource azurerm_api_management SurgeAssignmentAPIM > publisher_email
```

## Terraform plan logs:

```
subhrapakashjena@Mac apiManagementAssignment % terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
<= read (data resources)

Terraform will perform the following actions:

# data.azure_re_function_app_host_keys.SurgeAssignmentHSTKY will be read during apply
# (depends on a resource or a module with changes pending)
<= data "azuren_re_function_app_host_keys" "SurgeAssignmentHSTKY" {
  - blobs_extension_key      = (sensitive value)

  + secondary_file_microsoft_host = (known after apply)
  + secondary_location          = (known after apply)
  + secondary_queue_endpoint    = (known after apply)
  + secondary_queue_host        = (known after apply)
  + secondary_queue_microsoft_endpoint = (known after apply)
  + secondary_queue_microsoft_host = (known after apply)
  + secondary_table_endpoint    = (known after apply)
  + secondary_table_host        = (known after apply)
  + secondary_table_microsoft_endpoint = (known after apply)
  + secondary_table_microsoft_host = (known after apply)
  + secondary_web_endpoint     = (known after apply)
  + secondary_web_host          = (known after apply)
  + secondary_web_internet_endpoint = (known after apply)
  + secondary_web_microsoft_endpoint = (known after apply)
  + secondary_web_microsoft_host = (known after apply)
  + sftp_enabled                = false
  + shared_access_key_enabled   = true
  + table_encryption_key_type   = "Service"

  - blob_properties (known after apply)

  - network_rules (known after apply)

  - queue_properties (known after apply)

  - routing (known after apply)

  - share_properties (known after apply)

  - static_website (known after apply)
}

Plan: 7 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

## Terraform configuration for Azure API Management Service:

```
subhrapakashjena@Mac apiManagementAssignment % terraform validate
Success! The configuration is valid.

subhrapakashjena@Mac apiManagementAssignment % terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- create
<= read (data resources)

Terraform will perform the following actions:

# data.azure_re_function_app_host_keys.SurgeAssignmentHSTKY will be read during apply
# (depends on a resource or a module with changes pending)
<= data "azuren_re_function_app_host_keys" "SurgeAssignmentHSTKY" {
  - blobs_extension_key      = (sensitive value)
  - default_function_key     = (sensitive value)
  - durabletaskextension_key = (sensitive value)
  - event_grid_extension_config_key = (sensitive value)
  - event_grid_extension_key   = (sensitive value)
  - id                       = (known after apply)
  - name                     = "SurgeAssignmentHSTKY"
  - primary_key               = (sensitive value)
  - resource_group_name       = "SurgeAssignmentRG"
  - signalr_extension_key    = (sensitive value)
  - webpubsub_extension_key  = (sensitive value)
}

# azure_re_apl_management.SurgeAssignmentAPIM will be created
resource "azuren_re_apl_management" "SurgeAssignmentAPIM" {
  client_certificate_enabled = false
  developer_portal_url       = (known after apply)
  gateway_disabled            = false
  gateway_regionial_url      = (known after apply)
  gateway_url                 = (known after apply)
  id                          = (known after apply)
  location                    = "centralindia"
  management_api_url          = (known after apply)
  name                        = "SurgeAssignmentAPIM"
}

queue_properties (known after apply)

routing (known after apply)

share_properties (known after apply)

static_website (known after apply)

Plan: 7 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

azuren_re_resource_group.SurgeAssignmentRG: Creating...
azuren_re_resource_group.SurgeAssignmentRG: Still creating... [10s elapsed]
```

## Azure API Management Service:

The screenshot shows the Azure API Management Service overview for the resource group "SurgeAssignmentAPIM".

**Essentials:**

- Resource group: SurgeAssignmentAPIM
- Status: Online
- Location: Central India
- Subscription: Pay-As-You-Go
- Subscription ID: 4e199f71-01b7-42b1-84e4-d1cd6dc1a7b5
- Tags: Tags (edit) → Add tags
- Developer portal URL: <https://surgeassignmentapim.developer.azure-api.net>
- Gateway URL: <https://surgeassignmentapim.azure-api.net>
- Tier: Developer (No SLA)
- Virtual IP (VIP) addresses: public: 40.81.251.81
- Platform version: stv2.1

**Properties:**

- Pricing tier: Developer
- SLA: No
- Scale: 1 unit
- External cache: None
- Virtual network: None
- Custom domains:
  - Gateway URL: https://surgeassignmentapim.azure-api.net
  - Management URL: https://surgeassignmentapim.management.azure-api.net

**Delegation:** Delegation → Disabled

**Identities:** Identities → Username and password

**APIs:**

- Total APIs: 1
- Current APIs: 1
- Revisions: 0

**Subscriptions:** Subscriptions → 3

**Users:** Users → 1

**JSON View**

## Azure Functions App:

The screenshot shows the Azure Functions App overview for the function app "surge-assignment-azure-functions-app-1735596872585".

**Essentials:**

- Resource group: java-functions-group
- Status: Running
- Location: West US
- Subscription: Pay-As-You-Go
- Subscription ID: 4e199f71-01b7-42b1-84e4-d1cd6dc1a7b5
- Tags: Tags (edit) → Add tags
- Default domain: surge-assignment-azure-functions-app-1735596872585.azure...
- Operating System: Windows
- App Service Plan: java-functions-app-service-plan (Y1-0)
- Runtime version: 4.1036.3.23284

**Functions:**

Name	Trigger	Status	Monitor
HttpTriggerJava	HTTP	Enabled	Invocations and more

**Recommended services (preview):**

- Functions
- Deployment
- Settings
- Performance
- App Service plan
- Development Tools
- API
- Monitoring
- Automation
- Support + troubleshooting

### 3. An APIM configuration file that defines the API policies and security settings.

#### API inbound and outbound policies:

The screenshot shows the Microsoft Azure API Management interface for the 'SurgeAssignmentAPIM' service. On the left, the navigation menu is expanded to show the 'APIs' section. In the main content area, the 'Design' tab is selected for the 'surge-assignment-azure...' API. The 'Operations' tab is active under the 'Definitions' section. A code editor displays the APIM configuration XML for the 'HttpTriggerJava' operation, specifically the 'policies' section:

```
<policies>
<inbound>
  <set-backend-service id="apim-generated-policy" backend-id="surge-assignment-azure-fun">
    <base />
  </inbound>
  <backend>
    <base />
  </backend>
<outbound>
  <base />
</outbound>
<on-error>
  <base />
</on-error>
</policies>
```

Below the code editor are 'Save' and 'Discard' buttons, and links to 'Reset to default' and 'Calculate effective policy'.

#### API enabled with OAuth2 authorisation to make it secure:

The screenshot shows the Microsoft Azure API Management interface for the 'SurgeAssignmentAPIM' service. The left navigation menu is expanded to show the 'APIs' section. In the main content area, the 'Settings' tab is selected for the 'surge-assignment-azure...' API. The 'Subscription' and 'Security' sections are visible. Under 'Subscription', the 'Subscription required' checkbox is checked, and the 'Header name' is set to 'Ocp-Apim-Subscription-Key'. Under 'Security', the 'User authorization' dropdown is set to 'OAuth 2.0', and the note 'No OAuth 2.0 servers were configured.' is displayed. At the bottom are 'Save' and 'Discard' buttons.

## 4. A Function App configuration file that defines the API handlers and security settings.

### Azure Functions App (function.json):

The screenshot shows the Azure Functions App interface. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the URL is 'Home > surge-assignment-azure-functions-app-1735596872585 > HttpTriggerJava | Code + Test'. The main area has tabs for 'Code + Test', 'Integration', 'Function Keys', 'Invocations', 'Logs', and 'Metrics'. Under 'Code + Test', there are buttons for 'Save', 'Discard', 'Refresh', 'Test/Run', 'Get function URL', 'Disable', 'Delete', 'Upload', 'Resource JSON', and a copy icon. A note says 'Your app is currently in read only mode because you are running from a package file. To make any changes update the content in your zip file and WEBSITE\_RUN\_FROM\_ZIP environment variable.' Below this is the 'function.json' code:

```
1  {
2    "scriptFile": "../surge-assignment-azure-functions-app-1.0.0-SNAPSHOT.jar",
3    "entryPoint": "org.example.functions.HttpTriggerJava.run",
4    "bindings": [ {
5      "type": "httpTrigger",
6      "direction": "in",
7      "name": "req",
8      "methods": [ "GET", "POST" ],
9      "authLevel": "FUNCTION"
10     }, {
11       "type": "http",
12       "direction": "out",
13       "name": "$return"
14     } ]
15   }
```

Below the code editor is a 'Logs' section with a 'Logs' button and an 'App Insights Logs' dropdown. It displays a message: 'Connected! You are now viewing logs of Function runs in the current Code + Test panel. To see all the logs for this function, go to the Function App Overview page.' There's also a 'Log' button and a 'Close' button.

### Accessing the function app:

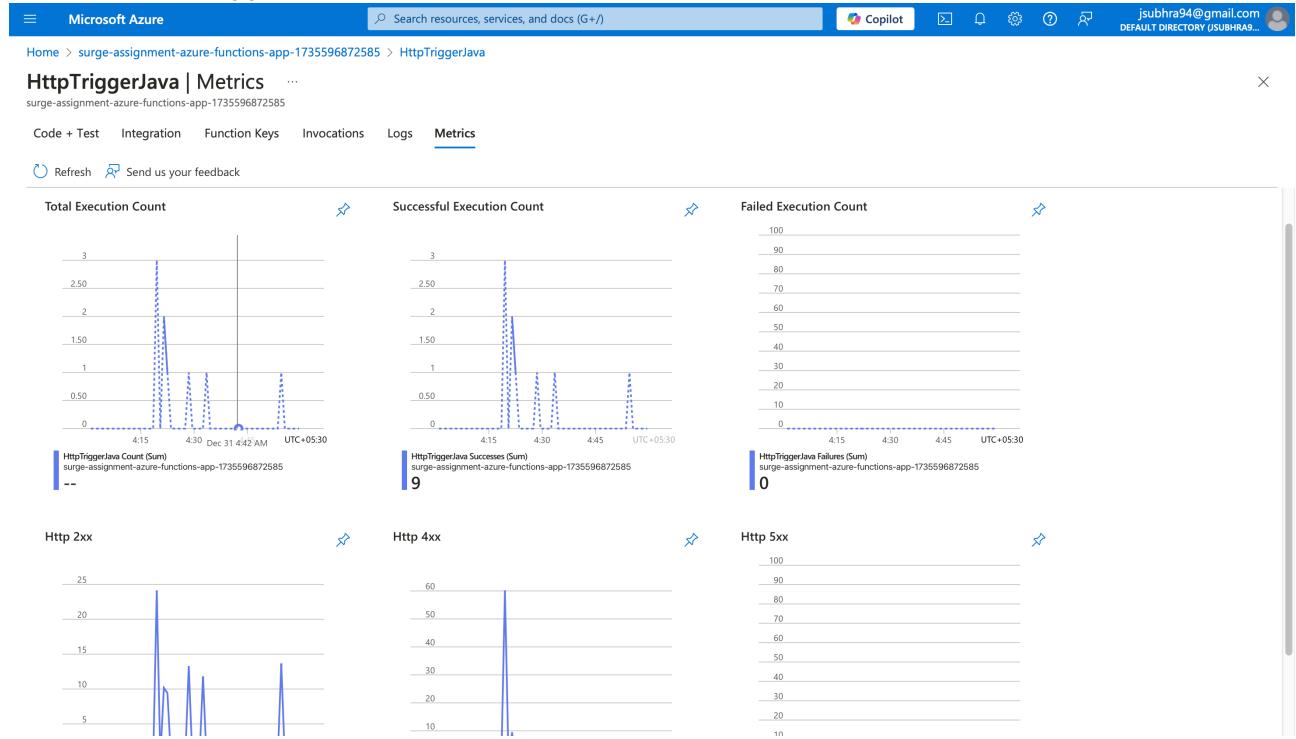
The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and a list of requests. The main area shows a 'GET https://surgeapi.duckdns.org' request. The 'Headers' tab is selected, showing a table with one row: 'Key' and 'Value'. The 'Body' tab shows the response: 'Hello, Subhra'. Other tabs include 'Cookies', 'Headers (6)', 'Tests', and 'Settings'. At the bottom, there's a status bar with 'Status: 200 OK Time: 1290 ms Size: 371 B Save Response' and a 'Console' button.

## Azure Functions app logs:

The screenshot shows the Azure Functions app logs for the 'HttpTriggerJava' function. The logs are displayed in a table with columns for timestamp, log level, and message. The messages show the function being executed via APIs, sending an invocation, posting to a worker, and finally executing successfully.

Timestamp	Log Level	Message
2024-12-30T23:24:08Z	[Information]	Executing 'Functions.HttpTriggerJava' (Reason='This function was programmatically called via the host APIs.', Id=6e2a41f8-69f7-47eb-be4e-a78361b58b8c)
2024-12-30T23:24:08Z	[Verbose]	Sending invocation id: 6e2a41f8-69f7-47eb-be4e-a78361b58b8c
2024-12-30T23:24:08Z	[Verbose]	Posting invocation id:6e2a41f8-69f7-47eb-be4e-a78361b58b8c on workerId:b7b52a5c-4af2-4980-acff-c2cf7f31130f
2024-12-30T23:24:08Z	[Information]	Java HTTP trigger processed a request.
2024-12-30T23:24:08Z	[Information]	Function "HttpTriggerJava" (Id: 6e2a41f8-69f7-47eb-be4e-a78361b58b8c) invoked by Java Worker
2024-12-30T23:24:08Z	[Information]	Executed 'Functions.HttpTriggerJava' (Succeeded, Id=6e2a41f8-69f7-47eb-be4e-a78361b58b8c, Duration=8ms)

## Azure Functions app metrics:



## 5. A report that summarises the solution and its benefits.

- Simplified API lifecycle management.
- Reduced operational overhead through serverless architecture.
- Enhanced security and compliance with advanced policies.
- Automated, consistent infrastructure deployment.
- Insightful analytics to improve performance and user experience.