Capstone Project – Integrating existing app with Azure services

Problem Statement:

Tailwind Traders is looking to integrate their E-commerce website QuickKart and database with Azure. The database, which follows strict schema, holds the product catalog, and all online orders. The website experiences slow response during peak hours, so, there is a need to implement auto scaling for the website. There is a need to automate the application development, when there are changes in the code base. Client is looking for CI/CD pipeline. Client is looking for microservices preferably PAAS . Clients are also looking for image storage and need to reduce cost if no one is accessing that image . Wherever possible reduce the cost and admin overhead. Also, the app needs to be monitored using proper Insights The solution also needs to be secure and store confidential data.

Resources or Technical stack:

- Frontend: Angular 13
- Backend: Dot Net Core 3.1
- Database: Azure SQL Database
- Web host platform: Azure Web App with scaling and deployment slots
- Microservices: Azure Function App
- Application security: Azure Key Vault
- CI/CD: GitHub/ Azure DevOps
- Monitoring solution: Azure Application Insights
- Cloud storage: Azure Blob storage

Task 1:

Installments to be done on local machine:

- Visual Studio Community edition 2022 with Azure SDK and Dot Net core 3.1
- Visual studio code
- Git
- Nodejs with Angular 13.3 installed
- Azure portal account

Testing the Frontend and Backend on local machine:

Uploading package source

- Open Visual studio and navigate to Tools -> Nuget Package manager -> Package manager setting -> package sources -> Add the package source
  Name : nuget.org

  Source: https://api.nuget.org/v3/index.json

  Select only this package. If you are having any other package sources, deselect it.

Importing the repos for Frontend and Backend:

- Clone the backend Project in your local system in Visual studio
- Backend Clone URL: https://akspks880187@dev.azure.com/akspks880187/QuickCart/_git/QuickCart
- Go to file in Visual Studio 2022 -> Open -> folder -> QuickKart(backend)
- Clone the frontend Project in your local system in Visual studio
- Frontend Clone URL: https://akspks880187@dev.azure.com/akspks880187/QuickCart/_git/Quick-Cart-FrontEnd
- Go to file in Visual Studio 2022 -> Open -> folder -> QuickKart(backend)

Create a storage account:

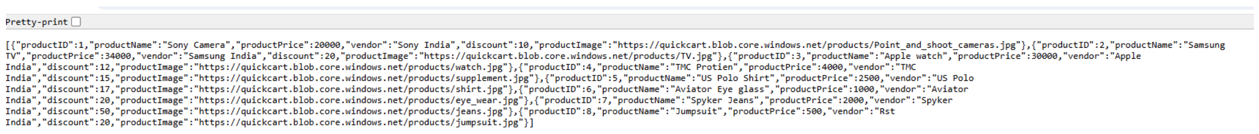
- Open Azure portal and create a storage account with a unique name.
- Then create a container and upload images in the container from the Frontend repo.
- Set the anonymous access to the container.

Editing the json file:

- Open the appsettings.json file from the backend repo in Visual studio and edit the file by adding the connection string provided.

Testing the backend in local machine:

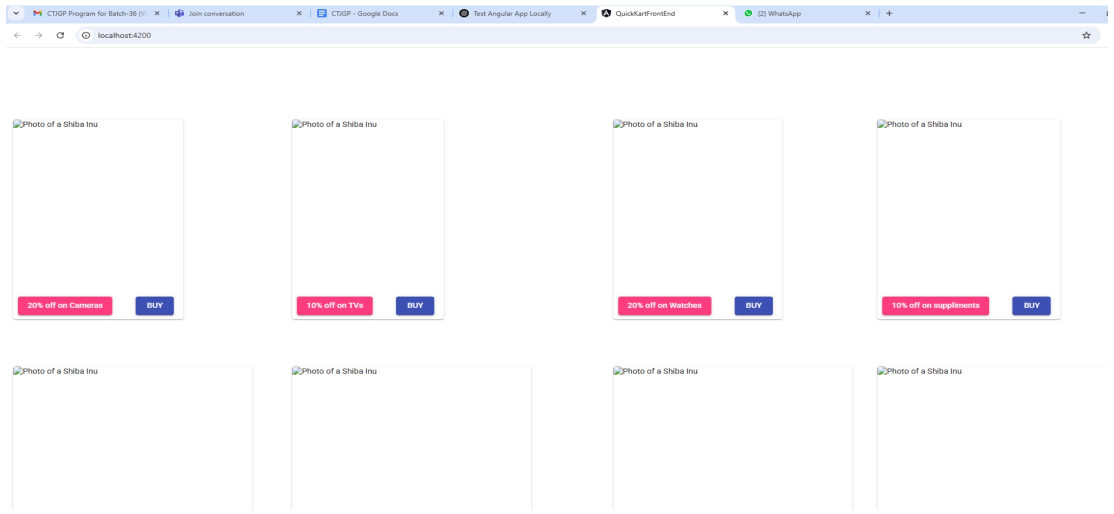
- Open command prompt and run dotnet run command from the path where the .csproj file exists.
- Check the backend by browsing to http://localhost:5001/api/home/getproducts//openbrowser


Pretty-print

```
[{"productID":1,"productName":"Sony Camera","productPrice":20000,"vendor":"Sony India","discount":10,"productImage":"https://quickcart.blob.core.windows.net/products/Point_and_shoot_cameras.jpg"},{"productID":2,"productName":"Samsung
TV","productPrice":34000,"vendor":"Samsung India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/TV.jpg"},{"productID":3,"productName":"Apple watch","productPrice":30000,"vendor":"Apple
India","discount":12,"productImage":"https://quickcart.blob.core.windows.net/products/watch.jpg"},{"productID":4,"productName":"TMC Protien","productPrice":4000,"vendor":"TMC
India","discount":15,"productImage":"https://quickcart.blob.core.windows.net/products/supplement.jpg"},{"productID":5,"productName":"US Polo Shirt","productPrice":2500,"vendor":"US Polo
India","discount":17,"productImage":"https://quickcart.blob.core.windows.net/products/shirt.jpg"},{"productID":6,"productName":"Aviator Eye glass","productPrice":1000,"vendor":"Aviator
India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/eye_wear.jpg"},{"productID":7,"productName":"Spyker Jeans","productPrice":2000,"vendor":"Spyker
India","discount":50,"productImage":"https://quickcart.blob.core.windows.net/products/jeans.jpg"},{"productID":8,"productName":"Jumpsuit","productPrice":500,"vendor":"Rst
India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/jumpsuit.jpg"}]
```

Testing the frontend in local machine:

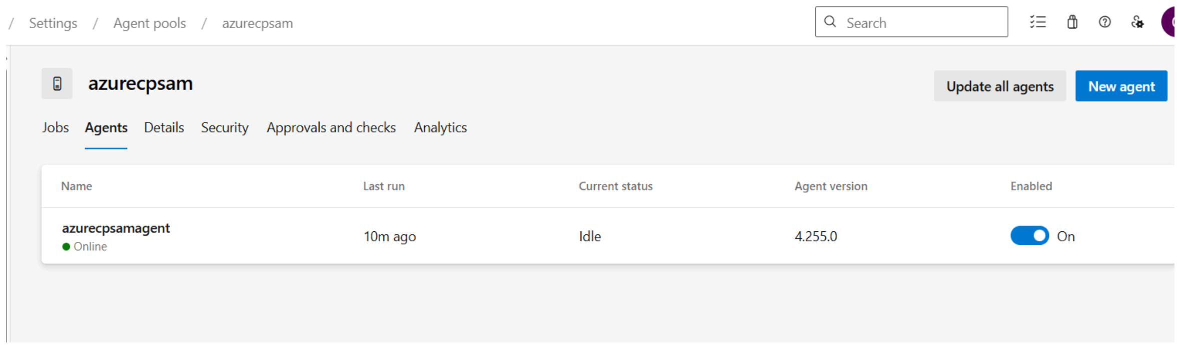
- Open command prompt where angular has already been installed.
- Run the command npm install
- Run the command ng serve
- Check the application by browsing to http://localhost:4200/



Task 2:

Create a new Agent Pool and Self-Hosted Agent in Azure Devops in VM:

To configure agent pools and agents follow the document url.
https://microsoftlearning.github.io/AZ400-DesigningandImplementingMicrosoftDevOpsSolutions/Instructions/Labs/AZ400_M02_L03_Configure_Agent_Pools_and_Understand_Pipeline_Styles.html



azurecpsam

Name	Last run	Current status	Agent version	Enabled
azurecpsamagent ● Online	10m ago	Idle	4.255.0	On

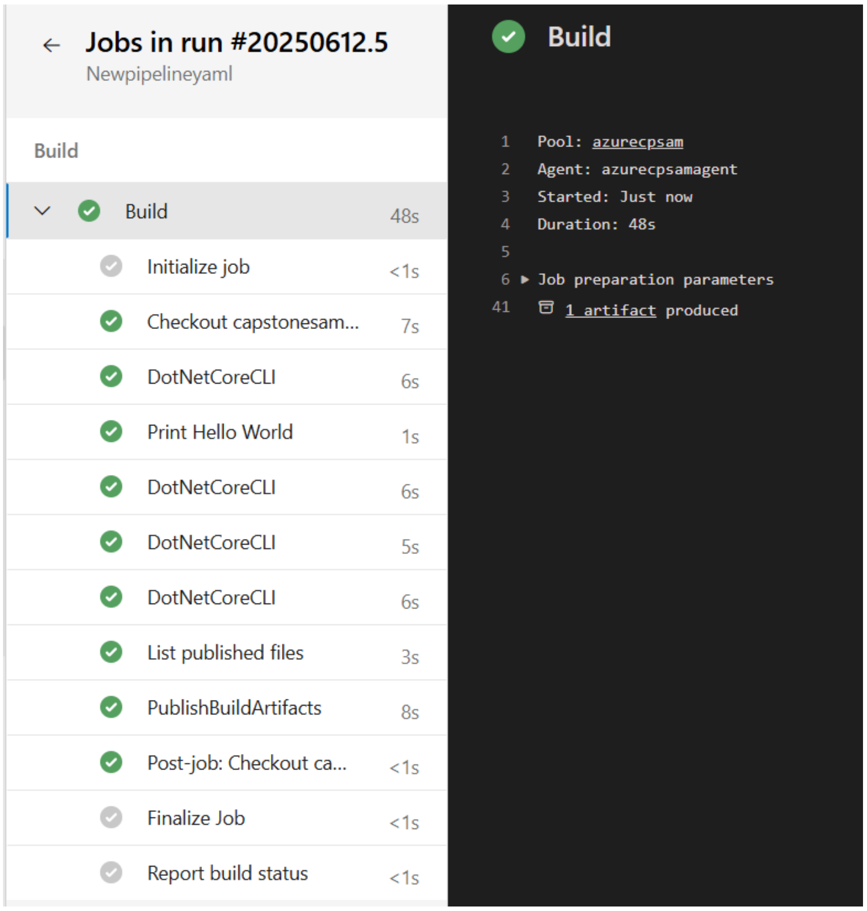
Once the agent has been configured and is online we can edit the pipeline by configuring it to use a self hosted agent for the deployments.

Implementing Continuous Integration for backend:

Open a starter pipeline and edit the code required:

```yaml
# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy
your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

trigger:

  branches:

    include:

    - master

variables:

- name: solution

  value: '**/*.sln'

- name: buildPlatform

  value: 'Any CPU'

- name: buildConfiguration

  value: 'Release'

stages:

- stage: Build

  jobs:

  - job: Build
```

```
pool:

  name: 'azurecpsam'

steps:

- task: NuGetToolInstaller@1

- task: NuGetCommand@2

  inputs:

    restoreSolution: '$(solution)'

- task: VSBuild@1

  inputs:

    solution: '$(solution)'

    msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package
/p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true
/p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp.z
ip" /p:DeployIisAppPath="Default Web Site"'

    platform: '$(buildPlatform)'

    configuration: '$(buildConfiguration)'

- task: PublishBuildArtifacts@1

  inputs:

    PathtoPublish: '$(Build.ArtifactStagingDirectory)'

    ArtifactName: 'drop'

    publishLocation: 'Container'
```


Jobs in run #20250612.5
Newpipelineyaml

Build

Build	48s
Initialize job	<1s
Checkout capstonesam...	7s
DotNetCoreCLI	6s
Print Hello World	1s
DotNetCoreCLI	6s
DotNetCoreCLI	5s
DotNetCoreCLI	6s
List published files	3s
PublishBuildArtifacts	8s
Post-job: Checkout ca...	<1s
Finalize Job	<1s
Report build status	<1s

Build

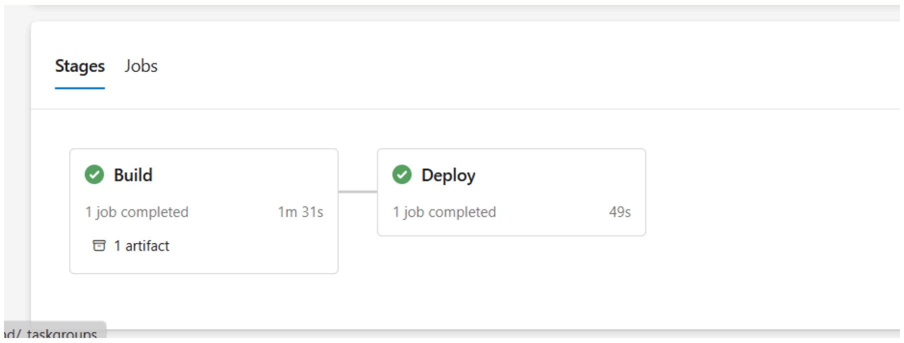
```
Pool: azurecpsam
Agent: azurecpsamagent
Started: Just now
Duration: 48s

▸ Job preparation parameters
1 artifact produced
```

Create a webapp:

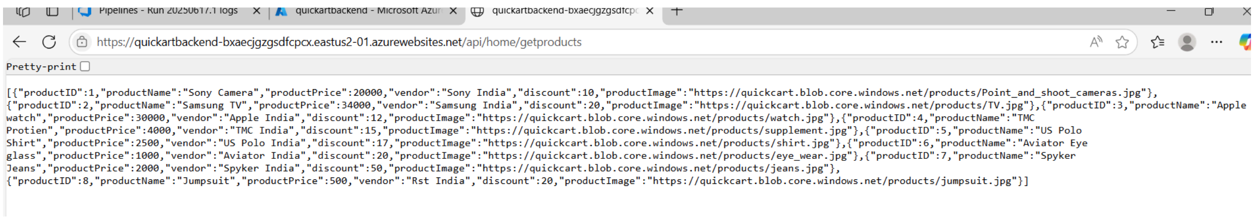
Now create a web app in azure portal to which the application can be deployed.
Now make changes to the existing code by adding the following lines. Also change the pool to microsoft hosted agent.

```
- stage: Deploy

  jobs:

  - job: Deploy

    pool:

      vmImage: 'windows-latest'

    steps:

    - task: DownloadBuildArtifacts@1
```

```yaml
        inputs:
          buildType: 'current'
          downloadType: 'single'
          artifactName: 'drop'
          downloadPath: '$(System.ArtifactsDirectory)'

      - task: AzureRmWebAppDeployment@5
        inputs:
          ConnectionType: 'AzureRM'
          azureSubscription: 'azuresubs2'
          appType: 'webApp'
          WebAppName: 'quickartbackend'
          packageForLinux: '$(System.ArtifactsDirectory)/drop/WebApp.zip'
```



https://quickartbackend-bxaecjgzgsdfcpcx.eastus2-01.azurewebsites.net/api/home/getproducts

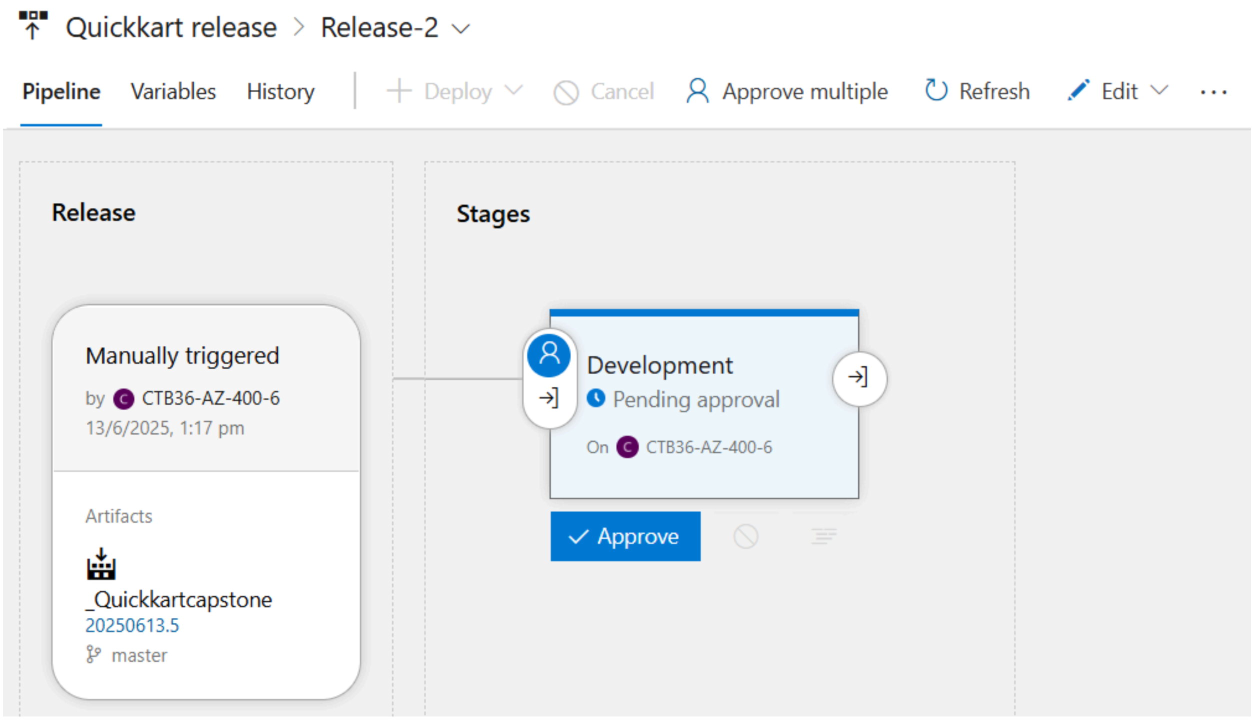

Pipelines - Run 20250617.1 logs
quickartbackend - Microsoft Azur
quickartbackend-bxaecjgzgsdfcpc
https://quickartbackend-bxaecjgzgsdfcpcx.eastus2-01.azurewebsites.net/api/home/getproducts
Pretty-print

```
[{"productID":1,"productName":"Sony Camera","productPrice":20000,"vendor":"Sony India","discount":10,"productImage":"https://quickcart.blob.core.windows.net/products/Point_and_shoot_cameras.jpg"},
{"productID":2,"productName":"Samsung TV","productPrice":34000,"vendor":"Samsung India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/TV.jpg"},{"productID":3,"productName":"Apple
watch","productPrice":30000,"vendor":"Apple India","discount":12,"productImage":"https://quickcart.blob.core.windows.net/products/watch.jpg"},{"productID":4,"productName":"TMC
Protien","productPrice":4000,"vendor":"TMC India","discount":15,"productImage":"https://quickcart.blob.core.windows.net/products/supplement.jpg"},{"productID":5,"productName":"US Polo
Shirt","productPrice":2500,"vendor":"US Polo India","discount":17,"productImage":"https://quickcart.blob.core.windows.net/products/shirt.jpg"},{"productID":6,"productName":"Aviator Eye
glass","productPrice":1000,"vendor":"Aviator India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/eye_wear.jpg"},{"productID":7,"productName":"Spyker
Jeans","productPrice":2000,"vendor":"Spyker India","discount":50,"productImage":"https://quickcart.blob.core.windows.net/products/jeans.jpg"},
{"productID":8,"productName":"Jumpsuit","productPrice":500,"vendor":"Rst India","discount":20,"productImage":"https://quickcart.blob.core.windows.net/products/jumpsuit.jpg"}]
```

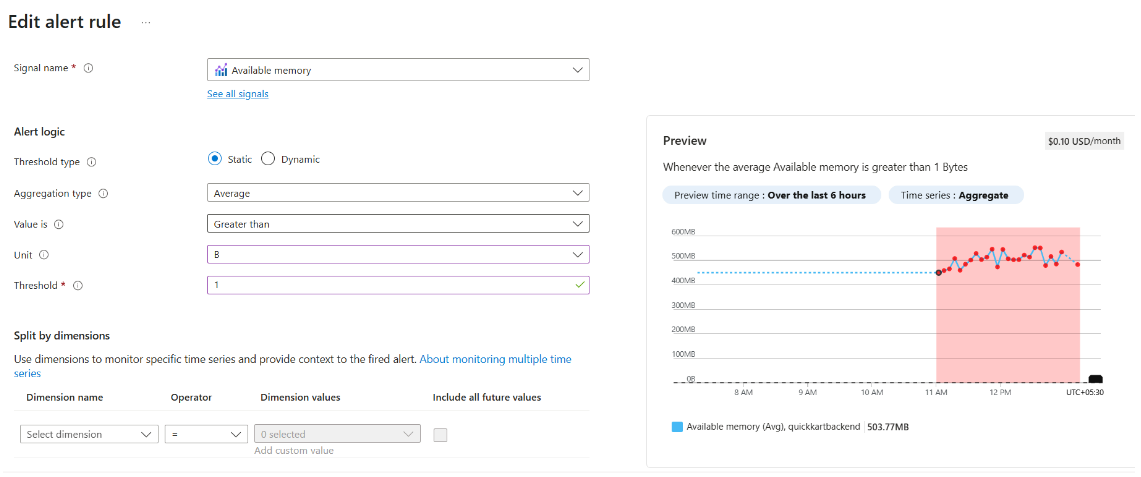
Task 3:

Configure a release pipeline by implementing pre deployment approvals and post deployment gates.

Configure pre deployment approvals by adding yourself as an approver.



Create an alert rule taking a metric and enable the alert.



Edit alert rule

Signal name *	Available memory
	See all signals

Alert logic

Threshold type	[✓ Static] Dynamic
Aggregation type	Average
Value is	Greater than
Unit	B
Threshold *	1

Split by dimensions

Use dimensions to monitor specific time series and provide context to the fired alert. About monitoring multiple time series

Dimension name	Operator	Dimension values	Include all future values
Select dimension	=	0 selected Add custom value	☐

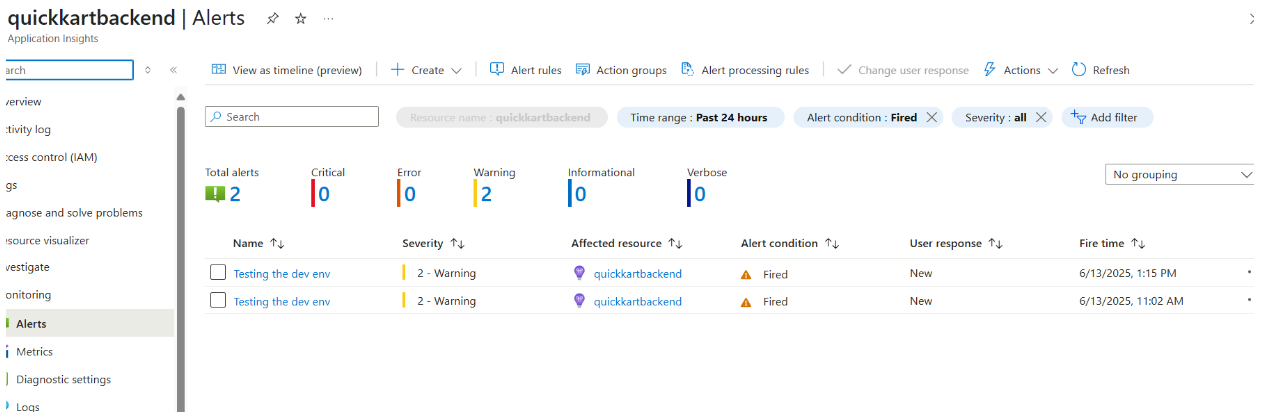
Preview — $0.10 USD/month

Whenever the average Available memory is greater than 1 Bytes

Preview time range : **Over the last 6 hours** Time series : **Aggregate**

🟦 Available memory (Avg), quickkartbackend | 503.77MB

Now generate active alerts.



quickkartbackend | Alerts

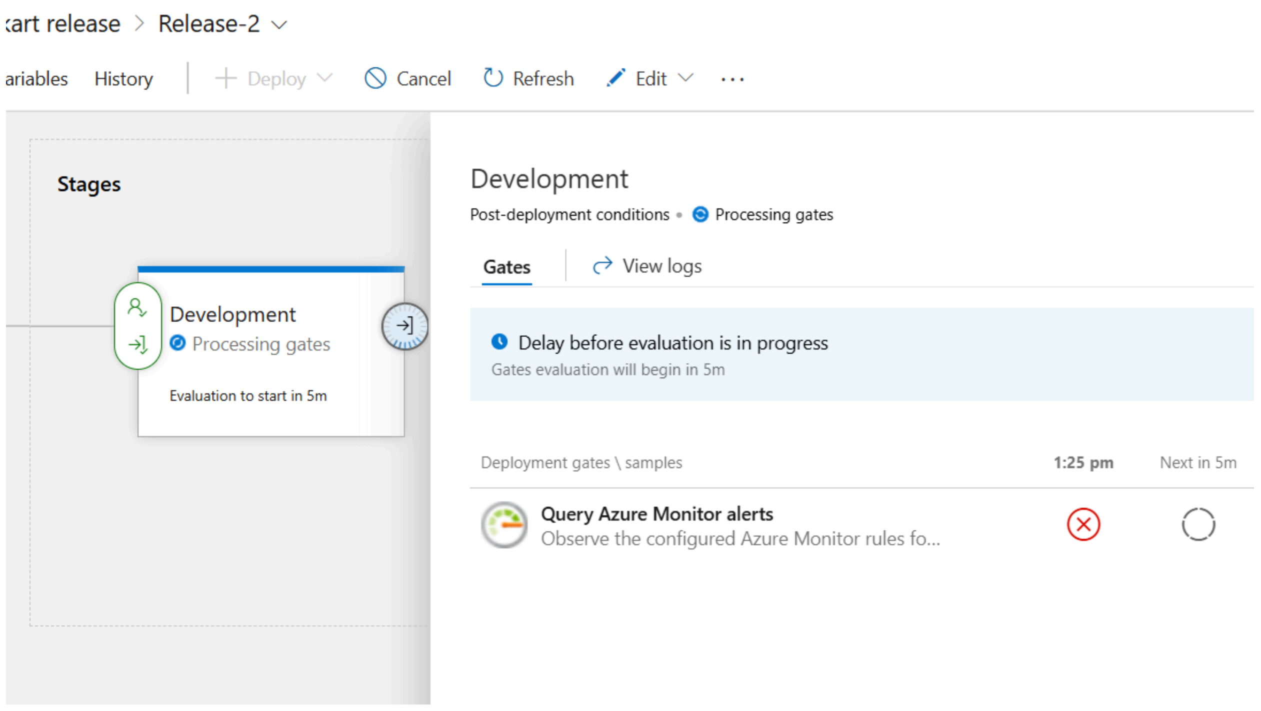
Application Insights

Name	Severity	Affected resource	Alert condition	User response	Fire time
Testing the dev env	2 - Warning	quickkartbackend	Fired	New	6/13/2025, 1:15 PM
Testing the dev env	2 - Warning	quickkartbackend	Fired	New	6/13/2025, 11:02 AM

Total alerts 2, Critical 0, Error 0, Warning 2, Informational 0, Verbose 0

Time range : Past 24 hours · Alert condition : Fired · Severity : all

Check the gates fail as there is an active alert.



Release-2

Development

Post-deployment conditions • Processing gates

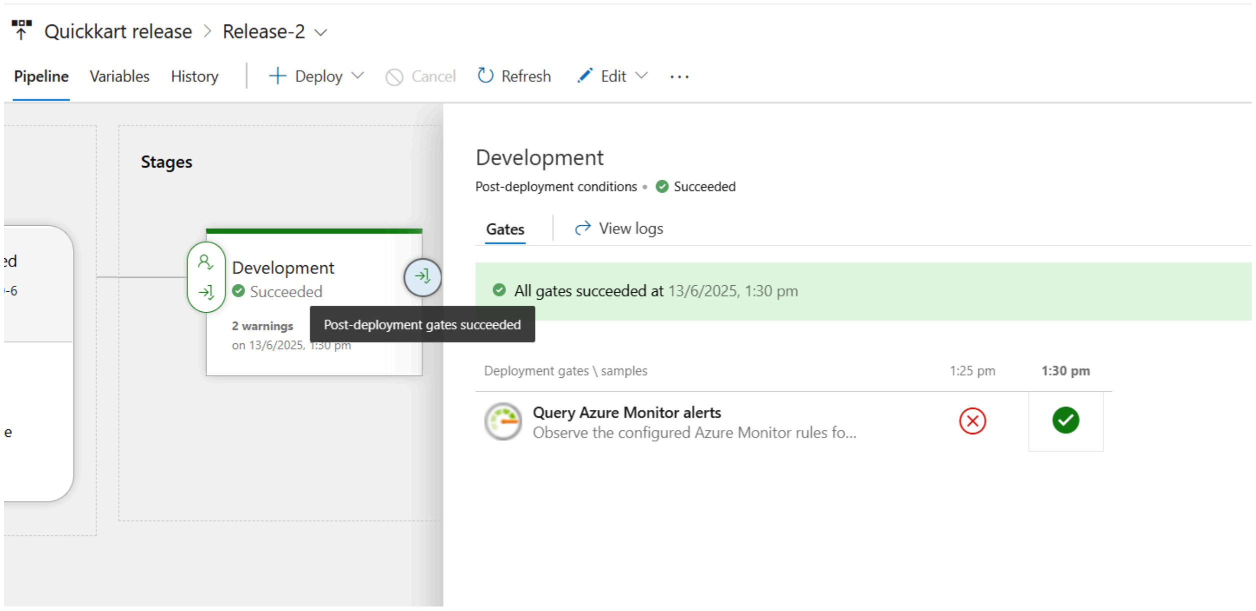
Delay before evaluation is in progress

Gates evaluation will begin in 5m

Evaluation to start in 5m

Deployment gates \ samples	1:25 pm	Next in 5m
Query Azure Monitor alerts Observe the configured Azure Monitor rules fo...	✗	○

Now, change the user response of the alert to closed and check that the gates pass this time.



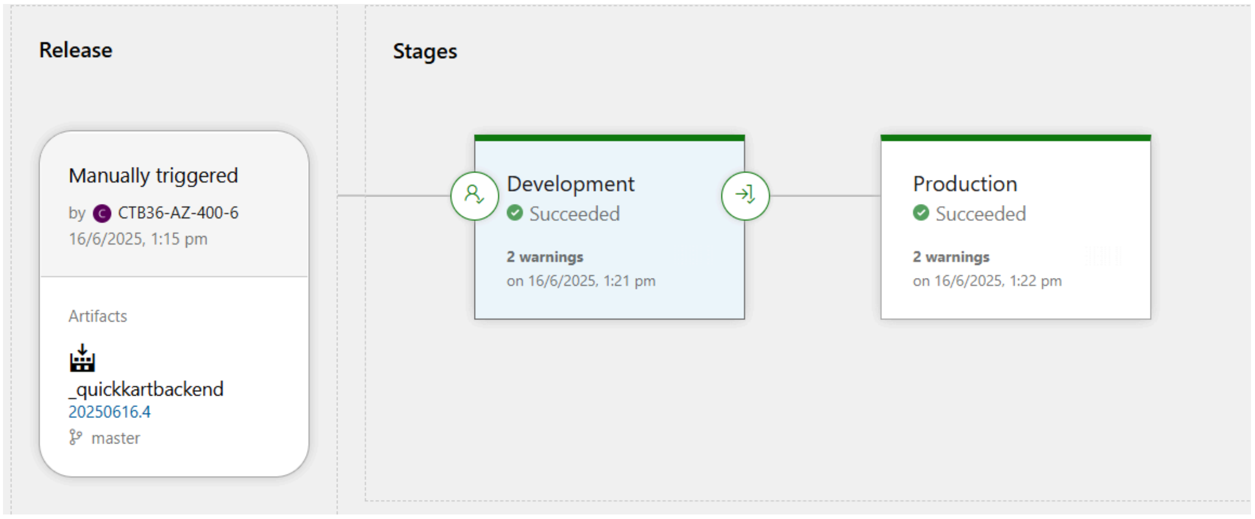
Development
Succeeded
2 warnings
on 13/6/2025, 1:30 pm
Post-deployment gates succeeded

Development

Post-deployment conditions • Succeeded

All gates succeeded at 13/6/2025, 1:30 pm

Deployment gates \ samples	1:25 pm	1:30 pm
Query Azure Monitor alerts Observe the configured Azure Monitor rules fo...	✗	✓



Manually triggered
by CTB36-AZ-400-6
16/6/2025, 1:15 pm

Artifacts
_quickkartbackend
20250616.4
master

Development
Succeeded
2 warnings
on 16/6/2025, 1:21 pm

Production
Succeeded
2 warnings
on 16/6/2025, 1:22 pm

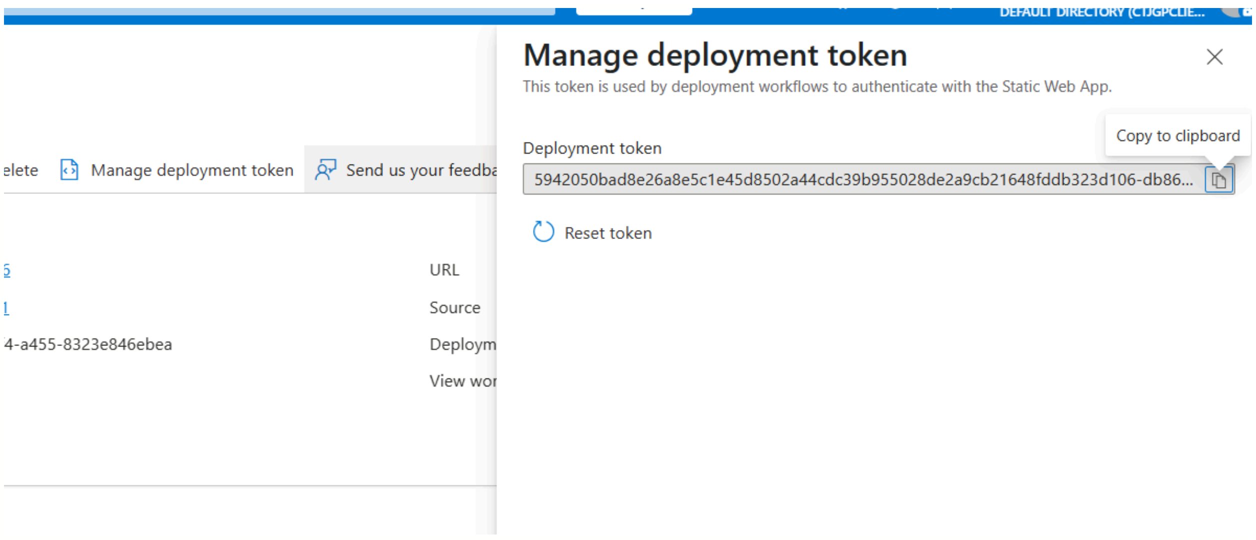
Task 4:

Create a static web app to deploy frontend application:

- Import the frontend repo to the existing project.
- Create a static web app in the Azure Portal.
- Create a new pipeline to deploy the frontend application to the static web app.

```
name: Azure Static Web Apps CI/CD

pr:
  branches:
    include:
      - master
trigger:
  branches:
    include:
      - master

jobs:
- job: build_and_deploy_job
  displayName: Build and Deploy Job
  condition: or(eq(variables['Build.Reason'],
'Manual'),or(eq(variables['Build.Reason'],
'PullRequest'),eq(variables['Build.Reason'], 'IndividualCI')))
  pool:
    vmImage: ubuntu-latest
  variables:
  - group:
Azure-Static-Web-Apps-witty-water-04b62000f-variable-group
  steps:
  - checkout: self
    submodules: true
  - task: AzureStaticWebApp@0
    inputs:
      azure_static_web_apps_api_token:
$(AZURE_STATIC_WEB_APPS_API_TOKEN_WITTY_WATER_04B62000F)
###### Repository/Build Configurations - These values can be
configured to match your app requirements. ######
# For more information regarding Static Web App workflow
configurations, please visit: https://aka.ms/swaworkflowconfig
      app_location: "/" # App source code path
      api_location: "" # Api source code path - optional
      output_location: "dist/quick-kart-front-end" # Built app
content directory - optional
###### End of Repository/Build Configurations ######
```

Make sure to add the correct output_location in the pipeline code. Also add the static web app api token which refers to the web app created in the azure portal.



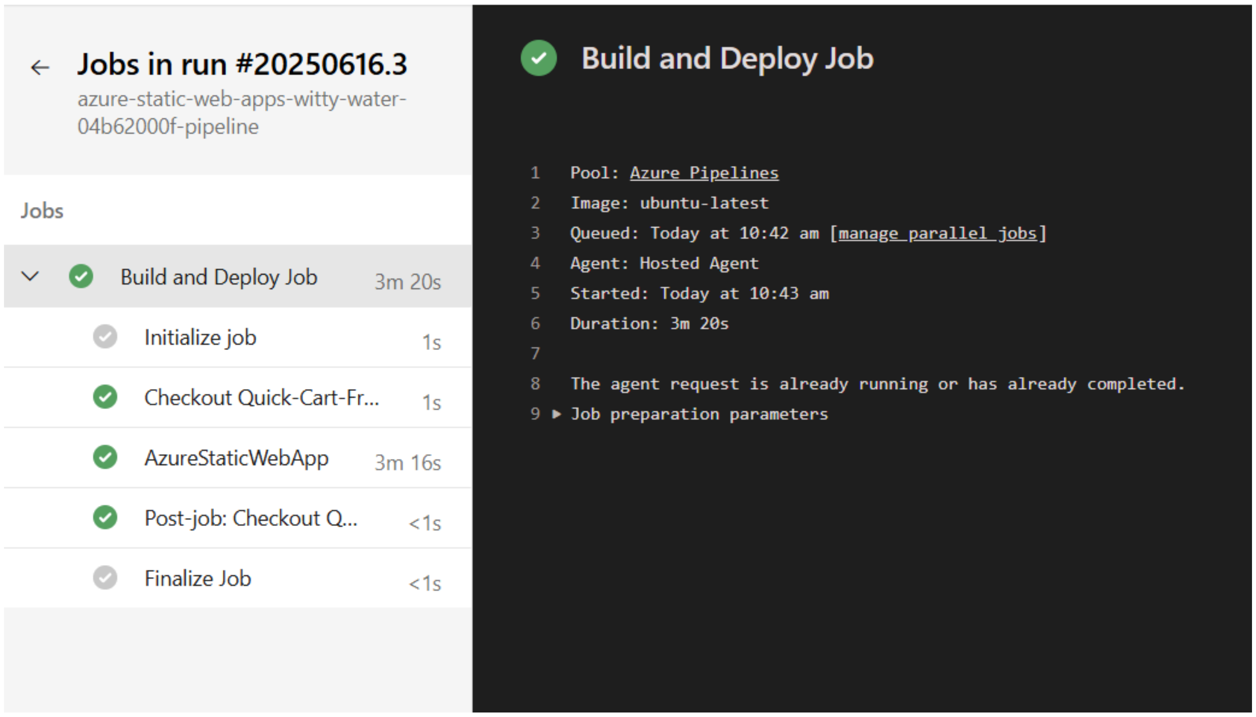
Manage deployment token

This token is used by deployment workflows to authenticate with the Static Web App.

Deployment token

5942050bad8e26a8e5c1e45d8502a44cdc39b955028de2a9cb21648fddb323d106-db86...

Run the pipeline and check the build and deploy job to get succeeded.



Jobs in run #20250616.3

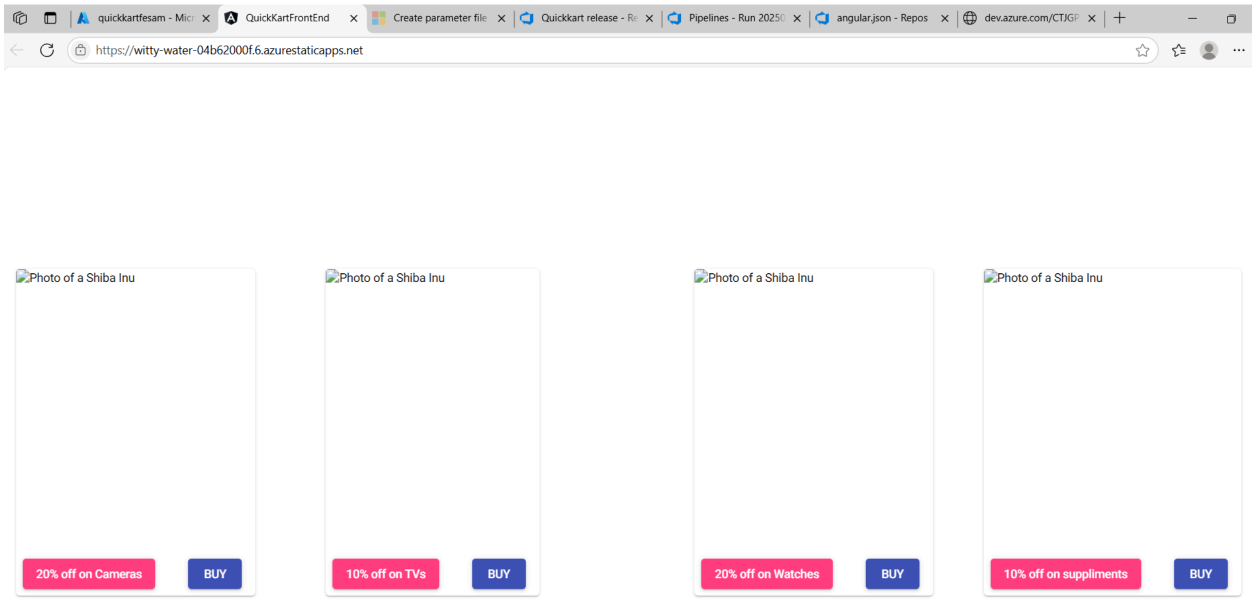
azure-static-web-apps-witty-water-04b62000f-pipeline

Jobs	
Build and Deploy Job	3m 20s
Initialize job	1s
Checkout Quick-Cart-Fr...	1s
AzureStaticWebApp	3m 16s
Post-job: Checkout Q...	<1s
Finalize Job	<1s

Build and Deploy Job

```
Pool: Azure Pipelines
Image: ubuntu-latest
Queued: Today at 10:42 am [manage parallel jobs]
Agent: Hosted Agent
Started: Today at 10:43 am
Duration: 3m 20s

The agent request is already running or has already completed.
▸ Job preparation parameters
```

Open the url of the static web app and check if the application is up and running.



Implement Login functionality using Azure Function:

- Clone the artifact which has the login code into visual studio.
  https://akspks880187@dev.azure.com/akspks880187/QuickCart/_git/LoginService.git
- Create a local.settings.json file in the login service folder.
  ```
  {
    "IsEncrypted": false,
    "Values": {
      "AzureWebJobsStorage": "UseDevelopmentStorage=true",
      "FUNCTIONS_WORKER_RUNTIME": "dotnet"
    }
  }
  ```
- Create an Azure key vault and add the connection string as a secret in the key vault. Fill in details in the Basic Tab. In Access Configurations --> Select Vault Access Policy --> Select Your Account --> Create Keyvault account.
- Add secrets in Keyvault. --> Add Name as DBConnectionString --> Add URL as vaule --> Create
- Now to integrate this with Azure pipeline, go to Azure DevOps --> Azure Pipeline --> Library
  Create a New Variable Group
  Give name --> Enable "Link Secrets from Azure KeyVault
  Authenticate the subscription and keyvault here
  If not connecting --> Go to Keyvault --> Go to Access Policy --> Select all the permissions

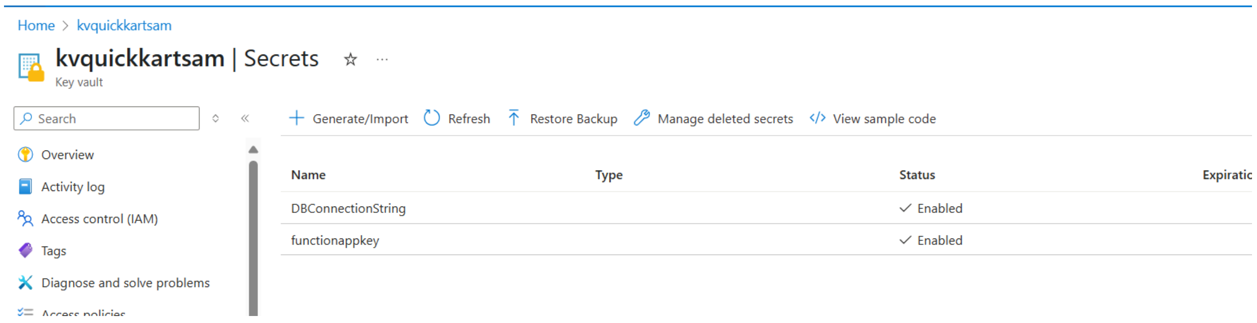
Once a variable group is created, reference it in the release pipeline.

- Edit release pipeline --> Go to variables --> Go to Variable Groups --> Click on "Link Variable Group" --> Select Scope --> Link
- Goto Task in the same release pipeline --> Add "Azure App Service Settings" --> This task should be the first task in that stage.
- Fill in the necessary details --> Scroll down --> In connection string section --> Add keyvault details

```
[
  {
      "name": "kvquickkartsam",
      "value": "$(DBConnectionString)"
  }
]
```

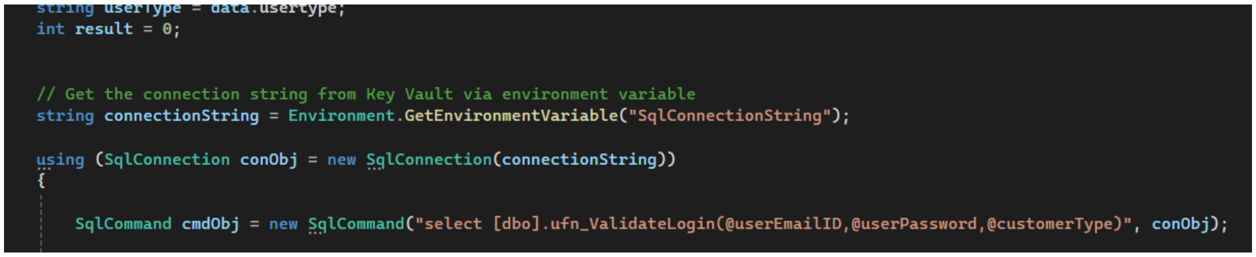
- Keyvault name is Variable Group name. Value to be referenced with $ symbol. --> Save and run

```
[

  {

      "name": "kvquickkartsam",

      "value": "$(DBConnectionString)"



  }

]
```



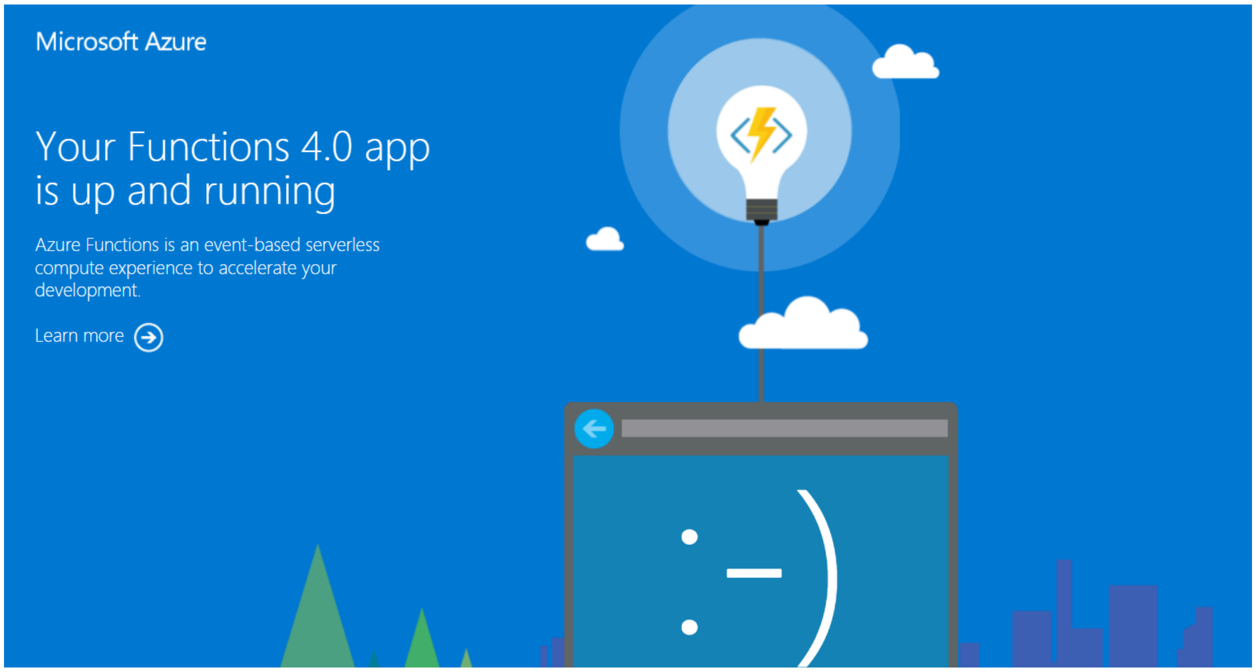
Name	Type	Status	Expiratic
DBConnectionString		✓ Enabled	
functionappkey		✓ Enabled	

- Change the Loginfunction.cs file to refer to the variable groups declared in the library to refer to the keyvault where the db connection string is stored.



```
string userType = data.usertype;
int result = 0;


// Get the connection string from Key Vault via environment variable
string connectionString = Environment.GetEnvironmentVariable("SqlConnectionString");

using (SqlConnection conObj = new SqlConnection(connectionString))
{

    SqlCommand cmdObj = new SqlCommand("select [dbo].ufn_ValidateLogin(@userEmailID,@userPassword,@customerType)", conObj);
```

- Once the changes are made we can build the code by pressing F5.
- Once the code is built we can then publish it to the Azure portal by selecting publish project and connect to Azure portal and create a new function app by filling required fields.
- Once the function app is created we can check if it is up and running by clicking its url.



Microsoft Azure

Your Functions 4.0 app is up and running

Azure Functions is an event-based serverless compute experience to accelerate your development.

Learn more

- Make the required change in the home-page.service.ts file from C:\Users\Sameera\source\repos\Quick-Cart(FrontEnd)\src\app\home\HomePage-Services in visual studio.

```
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, Observable, throwError } from 'rxjs';
import { IProduct } from '../Home-Interfaces/IProduct';
import { environment } from 'src/environments/environment';


@Injectable({
  providedIn: 'root'
})
```

```
export class HomePageService {




  products: IProduct[]=[];
  constructor(private http: HttpClient)
  {

  }
//
  //Getting the Products from backend API
  getProducts():Observable<IProduct[]>{
    let tempVar =
this.http.get<IProduct[]>('https://qucktest-test-ceaca0akbjajeaen.centralindia-01.azurewebsites.ne
t/api/home/getproducts')
    console.log(tempVar)
    return tempVar
  }


MakePayment(CardNumber1:string,cvv1:string,ex:string,pid:number,cost:number):Observable<
boolean>{

    var pay:Payment
    pay={cardNumber:CardNumber1,CVV:cvv1,Expiry:ex,ProdCost:cost,ProdID:pid}
    console.log(pay)

    let tempVar = this.http.post<boolean>('http://localhost:7181/api/PaymentFunction',pay)
    return tempVar
  }

  PostNewSubscriber(emailID:string):Observable<boolean>{

    console.log(emailID);

    const url =
`${environment.subscribeFunctionBaseUrl}/SubscribeFunction?code=${environment.subscribeF
unctionKey}&emailID=${emailID}`;
```

```
    let tempVar = this.http.get<boolean>(url);


    console.log(tempVar);
    return tempVar;
  }


  ValidateUser(userEmailID:string, userPassword:string, type:string):Observable<number>
  {
    var user:User
    user = { emailID: userEmailID, password: userPassword, usertype: type };
    const url =
`${environment.functionAppBaseUrl}/LoginFunction?code=${environment.functionKey}`;
    console.log(user)


    return this.http.post<number>(url, user);

  }

  public uploadImage(image: File): Observable<Response>{
    const formData = new FormData();

    formData.append('image', image);
    console.log(formData)
    let
result=this.http.post<Response>('https://localhost:5001/api/admin/upload',formData).pipe(catchE
rror(this.errorHandler))
    console.log(result)
    return result
  }

  errorHandler(error: HttpErrorResponse) {
    console.log(error);
    return throwError(error.message|| "server error")
  }
}

export class User{
  emailID:string='';
  password:string='';
  usertype:string='';


}

export class Payment{

  cardNumber:string='';
  CVV:string='';
  Expiry:string='';
  ProdCost:number=0;
  ProdID:number=0;

}
```

- Make the necessary changes in environment.ts and environment.prod.ts files from C:\Users\Sameera\source\repos\Quick-Cart(FrontEnd)\src\environments folder.

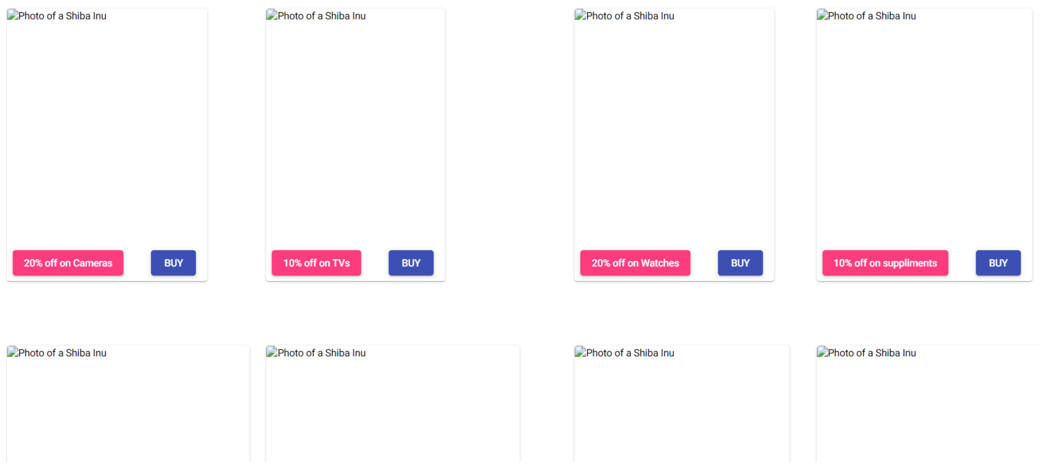
environment.ts

```
export const environment = {
  production: false,
  functionAppBaseUrl: 'https://funappsam.azurewebsites.net/api',
  functionKey: 'X1zM66M9kCPyqdUSwrYlF4w5MGysJJ_TPZwKCfziHlKIAzFu2RZRaA==',
  subscribeFunctionBaseUrl: 'https://quickcart-microservice.azurewebsites.net/api',
  subscribeFunctionKey:
'pIOIb80woJnaC8N77yQl1nSLxlDAvSa5mw9rli414zaoAzFuF3cBhA=='
};
```

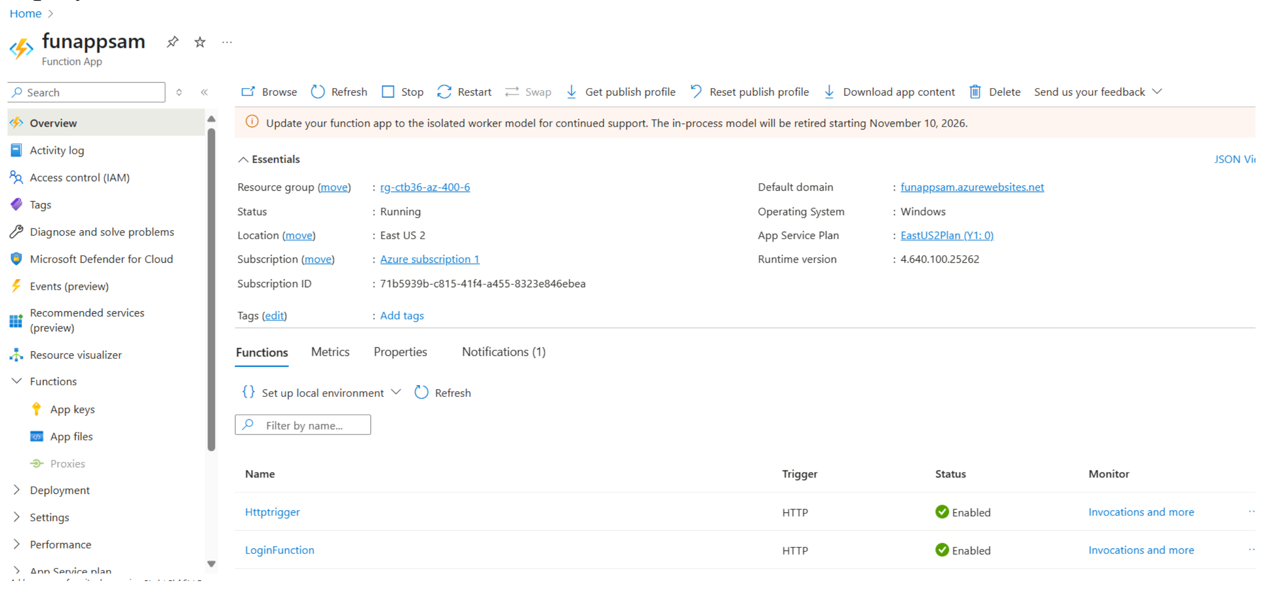
environment.prod.ts

```
export const environment = {
  production: true,
  functionAppBaseUrl: 'https://funappsam.azurewebsites.net/api',
  functionKey: 'X1zM66M9kCPyqdUSwrYlF4w5MGysJJ_TPZwKCfziHlKIAzFu2RZRaA==',
  subscribeFunctionBaseUrl: 'https://quickcart-microservice.azurewebsites.net/api',
  subscribeFunctionKey:
'pIOIb80woJnaC8N77yQl1nSLxlDAvSa5mw9rli414zaoAzFuF3cBhA=='
};
```

- The files are changed such that the [home-page.service.ts](#) file refers to the environment files for the function key instead of hardcoding the value in the code itself which is publicly accessible.
- Once the changes are made and saved we can test it in the local machine by once again building the application using ng build and testing it on the local host.
- We can then re deploy the application by running the pipeline for frontend and check the changes made.



- Download and install Azure Functions Runtime https://learn.microsoft.com/en-us/azure/azure-functions/functions-run-local?tabs=windows%2Cisolated-process%2Cnode-v4%2Cpython-v2%2Chttp-trigger%2Ccontainer-apps&pivots=programming-language-csharp
- Click on Publish
- Go to Azure Portal with Function App. Scroll down and see LoginFunction getting deployed



Home >

funappsam
Function App

Search | Browse | Refresh | Stop | Restart | Swap | Get publish profile | Reset publish profile | Download app content | Delete | Send us your feedback

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Events (preview)
Recommended services (preview)
Resource visualizer
Functions
App keys
App files
Proxies
Deployment
Settings
Performance
App Service plan

Update your function app to the isolated worker model for continued support. The in-process model will be retired starting November 10, 2026.

Essentials | JSON Vi

Resource group (move)	: rg-ctb36-az-400-6	Default domain	: funappsam.azurewebsites.net
Status	: Running	Operating System	: Windows
Location (move)	: East US 2	App Service Plan	: EastUS2Plan (Y1: 0)
Subscription (move)	: Azure subscription 1	Runtime version	: 4.640.100.25262
Subscription ID	: 71b5939b-c815-41f4-a455-8323e846ebea		
Tags (edit)	: Add tags		

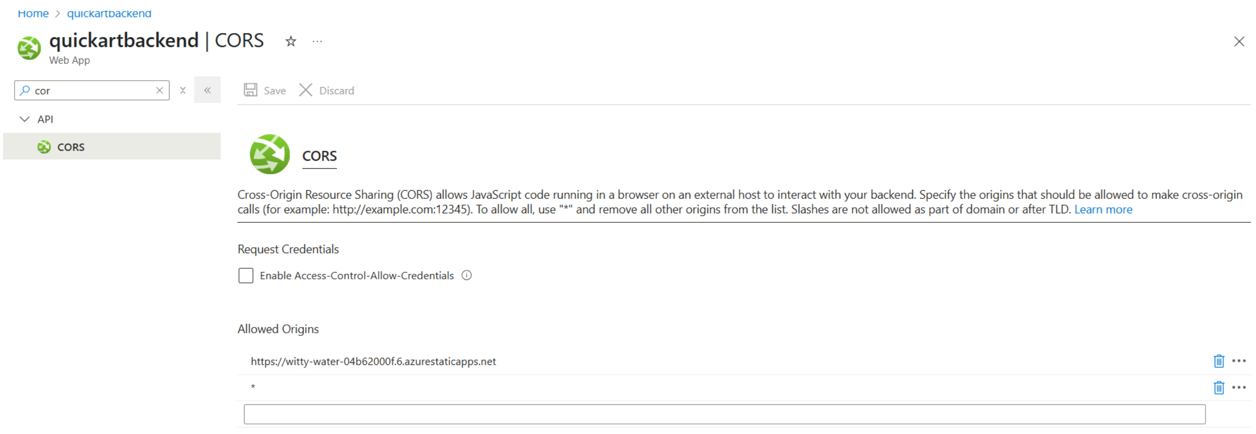
Functions | Metrics | Properties | Notifications (1)

Set up local environment | Refresh

Filter by name...

Name	Trigger	Status	Monitor
Httptrigger	HTTP	✓ Enabled	Invocations and more
LoginFunction	HTTP	✓ Enabled	Invocations and more

- Enable CORS policy for Azure Functions
  - Copy static web app URL.
  - Go inside the LoginFunction App
  - Search for CORS
  - Add your static web app link
  - Add * in next text box
  - Save



CORS

Cross-Origin Resource Sharing (CORS) allows JavaScript code running in a browser on an external host to interact with your backend. Specify the origins that should be allowed to make cross-origin calls (for example: http://example.com:12345). To allow all, use "*" and remove all other origins from the list. Slashes are not allowed as part of domain or after TLD. Learn more

Request Credentials

☐ Enable Access-Control-Allow-Credentials

Allowed Origins

https://witty-water-04b62000f.6.azurestaticapps.net
*

- It will take up to 10 min to reflect this functionality.
- See to it that the pipeline is running. Once the pipeline finishes, then login.
- You should be able to login now.