AZURE PROJECT

1. Deployed a full-stack Spotify clone using Node.js for the backend and integrated Spotify OAuth for secure user login and access token handling.

2. Hosted the backend on Azure App Services with GitHub Actions CI/CD pipeline for automated build and deployment from a private GitHub repository.

3. Used Azure Cosmos DB (MongoDB API) to store and retrieve song data dynamically through defined REST API endpoints.

4. Configured environment variables like Cosmos DB connection string, JWT secret, client origin, and port for secure and flexible deployments.

5. Built a scalable infrastructure using Azure Virtual Network (VNet), subnets, and peering across backend, frontend, and jump VNets.

6. Provisioned Azure Virtual Machines (VMs) and deployed a jump box for secure access, along with a VPN Gateway using root/client certificates.

7. Created and configured a Load Balancer and Virtual Machine Scale Set (VMSS) to distribute frontend traffic and ensure high availability.

8. Configured monitoring tools such as Log Stream and App Service Logs for real-time diagnostics and application health.

9. Tested API endpoints using Postman and ensured that endpoints like /api/test and /api/songs functioned as expected.

10. Achieved a working end-to-end cloud deployment with frontend and backend integration, demonstrating scalability, security, and CI/CD workflows.

SERVICES USED-

1. Resource Group
Used to group and manage all related Azure resources like VNets, VMs, Web Apps, and DBs under a single logical container for easy access and monitoring.

2. Virtual Network (VNet)
Three VNets (backend, frontend, and jump) were created to segment and isolate different tiers of the application and enable secure communication between services.

3. Virtual Machines (VMs)
Deployed in backend and jump VNets, the VMs provided isolated environments for running application processes and enabling remote administration.

4. VPN Gateway
Configured to securely connect the on-premise or jump VM to Azure VNets through point-to-site VPN, enabling encrypted private network access.

5. Jump VM / Jump VNet
Used as a secure entry point to access private VMs in backend VNet via peering, ensuring controlled access through Bastion or SSH.

6. VNet Peering
Configured to allow communication between frontend VNet and jump VNet, and between backend VNet and jump VNet, without routing traffic over the internet.

7. Web App (App Service)
Hosted the Node.js Spotify backend server, with seamless integration to GitHub for automated deployment and easy scaling options.

8. Deployment Center (GitHub Actions)
Enabled CI/CD by connecting the GitHub repository to Azure App Service, automating build and deployment on every push to the main branch.

9. Environment Variables
Configured in App Service to securely store secrets such as Cosmos DB connection strings, JWT secret, and client origin for runtime access.

10. Log Stream / Monitoring
Used to view real-time application logs and debug deployment or runtime errors in the Azure portal, ensuring observability and troubleshooting.

11. Azure Cosmos DB
Served as the backend database using MongoDB API, storing and retrieving Spotify song data via Mongoose ORM.

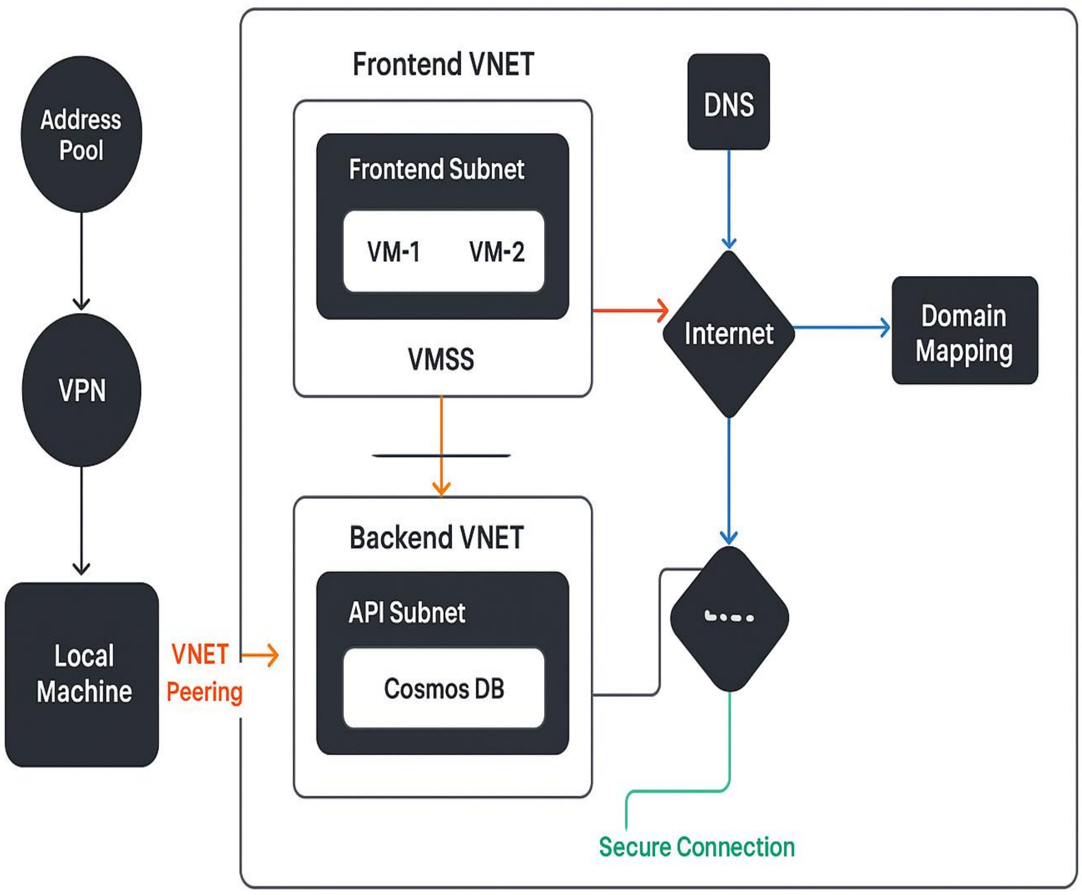
12. Load Balancer (LB)
A frontend Load Balancer (fend-lb) was created to distribute traffic across the VM Scale Set instances for high availability.

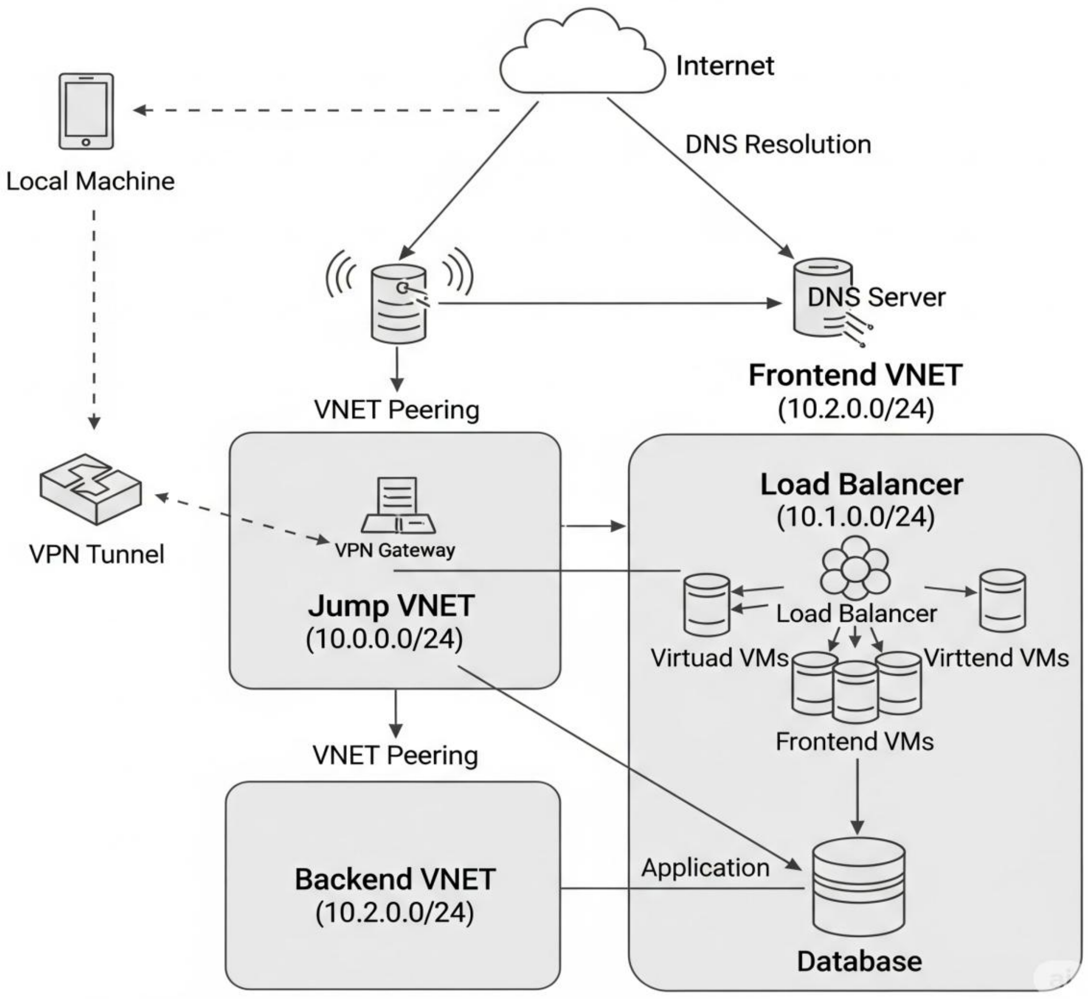
13. VM Scale Set (VMSS)
Deployed in the frontend VNet to create and manage multiple frontend VMs with auto-scaling and load balancing.

14. Azure Bastion
Provided secure RDP/SSH connectivity to VMs directly through the Azure portal without exposing public IPs, enhancing network security and simplifying remote access.

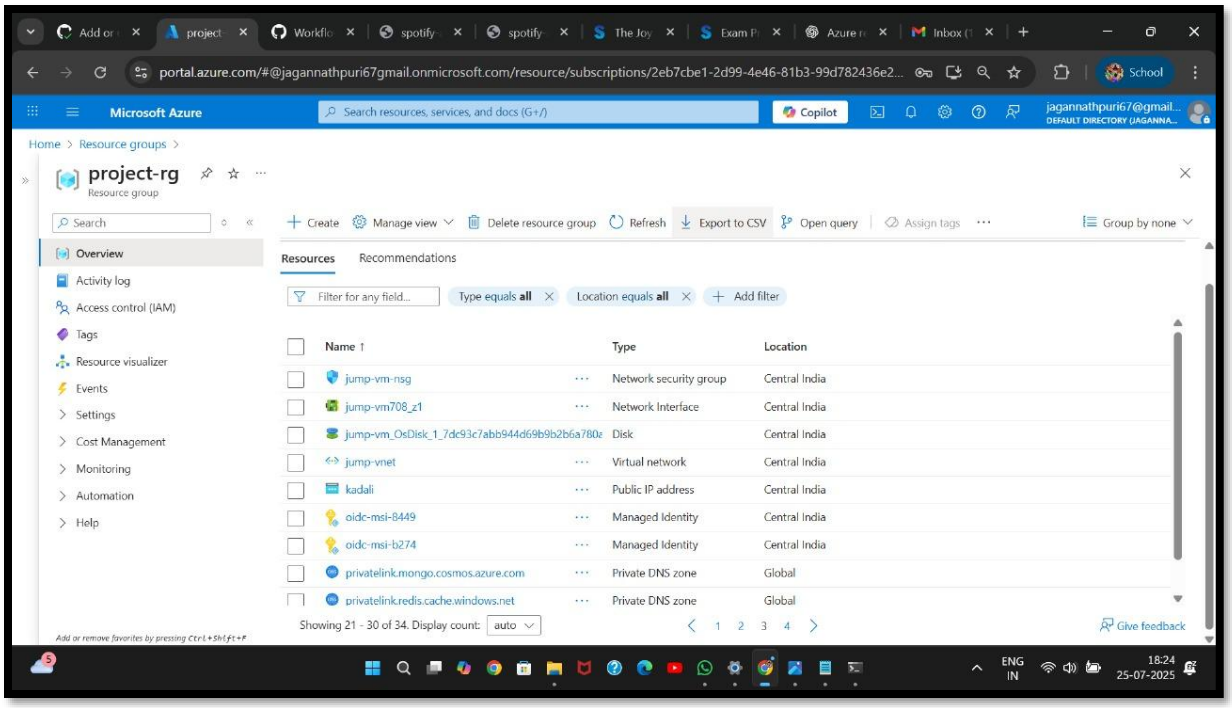
15. Azure Network Security Groups (NSG)
Configured to control inbound and outbound traffic to subnets and VMs, ensuring only required ports (like 22, 80, 443) were accessible and all other traffic blocked for security.

Architecture Diagram




Internet
Local Machine
DNS Resolution
DNS Server
Frontend VNET
(10.2.0.0/24)
VNET Peering
Load Balancer
(10.1.0.0/24)
VPN Tunnel
VPN Gateway
Jump VNET
(10.0.0.0/24)
Load Balancer
Virtuad VMs
Virttend VMs
Frontend VMs
VNET Peering
Backend VNET
(10.2.0.0/24)
Application
Database

Steps-



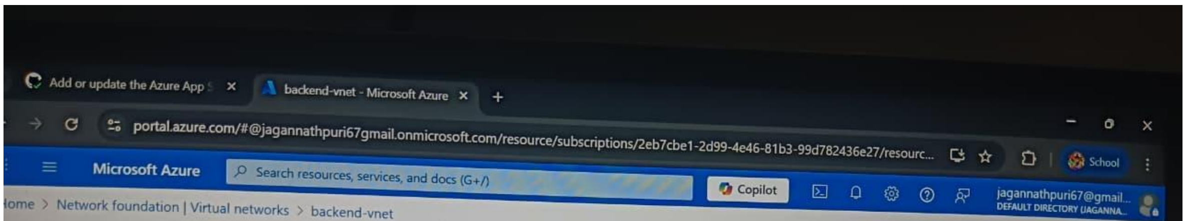
Home > Resource groups >

project-rg
Resource group

Name ↑	Type	Location
jump-vm-nsg	Network security group	Central India
jump-vm708_z1	Network Interface	Central India
jump-vm_OsDisk_1_7dc93c7abb944d69b9b2b6a780a	Disk	Central India
jump-vnet	Virtual network	Central India
kadali	Public IP address	Central India
oidc-msi-8449	Managed Identity	Central India
oidc-msi-b274	Managed Identity	Central India
privatelink.mongo.cosmos.azure.com	Private DNS zone	Global
privatelink.redis.cache.windows.net	Private DNS zone	Global

Showing 21 - 30 of 34. Display count: auto

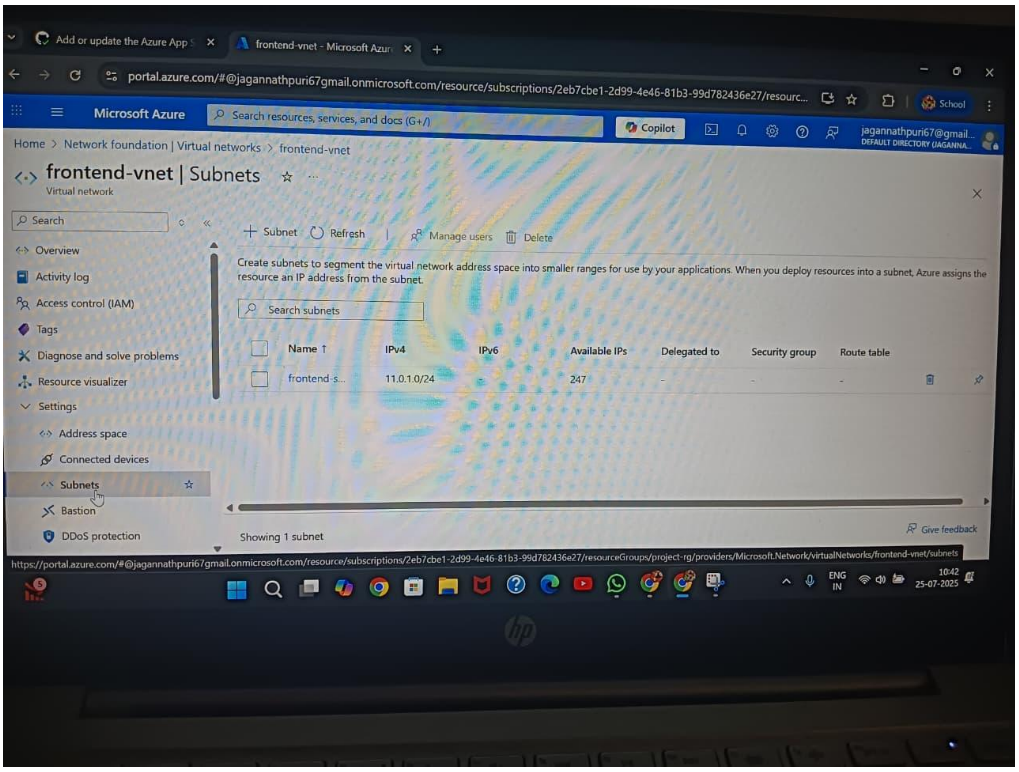
I created a Resource Group to logically group and manage all related Azure resources for my Spotify backend project. This grouping allowed me to deploy, monitor, and control resources like VMs, VNet, and App Services efficiently under one umbrella. It served as the foundational scope for my deployment.



portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourc...

Home > Network foundation | Virtual networks > backend-vnet

I created a Backend VNet to securely isolate backend services of my Spotify clone. Within this VNet, I configured two subnets — one for the API server and another for the database — ensuring controlled traffic flow and separation of concerns for enhanced security and scalability.


Home > Network foundation | Virtual networks > frontend-vnet

frontend-vnet | Subnets

Virtual network

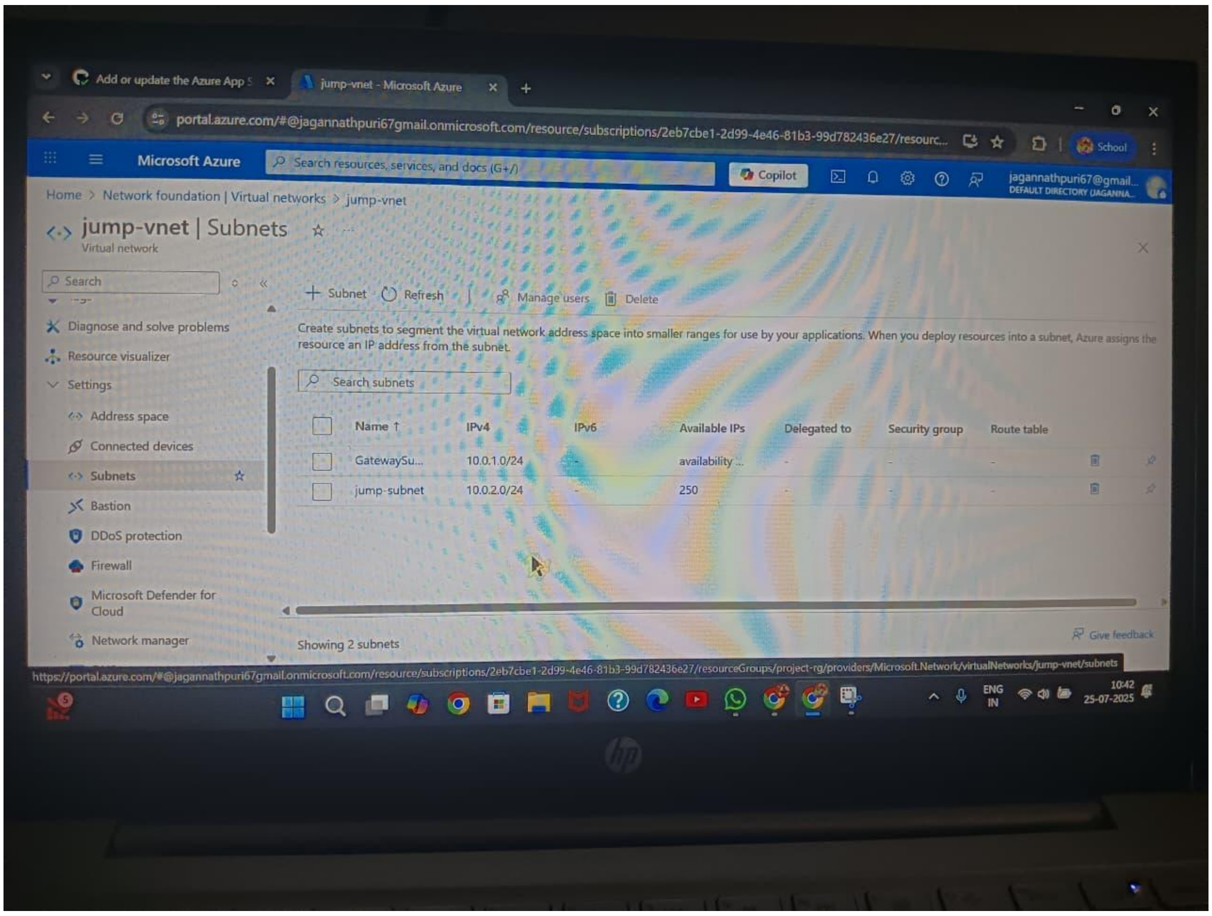
Create subnets to segment the virtual network address space into smaller ranges for use by your applications. When you deploy resources into a subnet, Azure assigns the resource an IP address from the subnet.

Name ↑	IPv4	IPv6	Available IPs	Delegated to	Security group	Route table
frontend-s...	11.0.1.0/24	-	247	-	-	-

Showing 1 subnet

https://portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourceGroups/project-rg/providers/Microsoft.Network/virtualNetworks/frontend-vnet/subnets

I created a Frontend VNet to host the user-facing components of the Spotify clone. This VNet is isolated from the backend and contains resources like the frontend VMSS and load balancer, ensuring a clean separation between UI and internal services.



portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourc...

Home > Network foundation | Virtual networks > jump-vnet

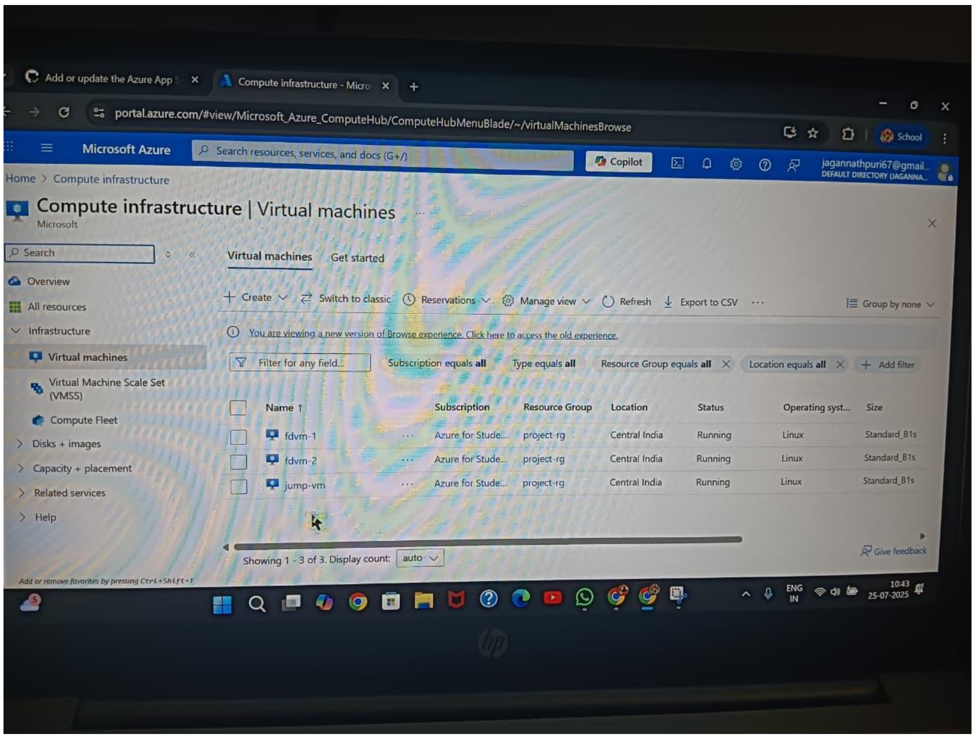
jump-vnet | Subnets

Virtual network

Create subnets to segment the virtual network address space into smaller ranges for use by your applications. When you deploy resources into a subnet, Azure assigns the resource an IP address from the subnet.

Name ↑	IPv4	IPv6	Available IPs	Delegated to	Security group	Route table
GatewaySu...	10.0.1.0/24	-	availability ...	-	-	-
jump-subnet	10.0.2.0/24	-	250	-	-	-

Showing 2 subnets



portal.azure.com/#view/Microsoft_Azure_ComputeHub/ComputeHubMenuBlade/~/virtualMachinesBrowse

Home > Compute infrastructure

Compute infrastructure | Virtual machines

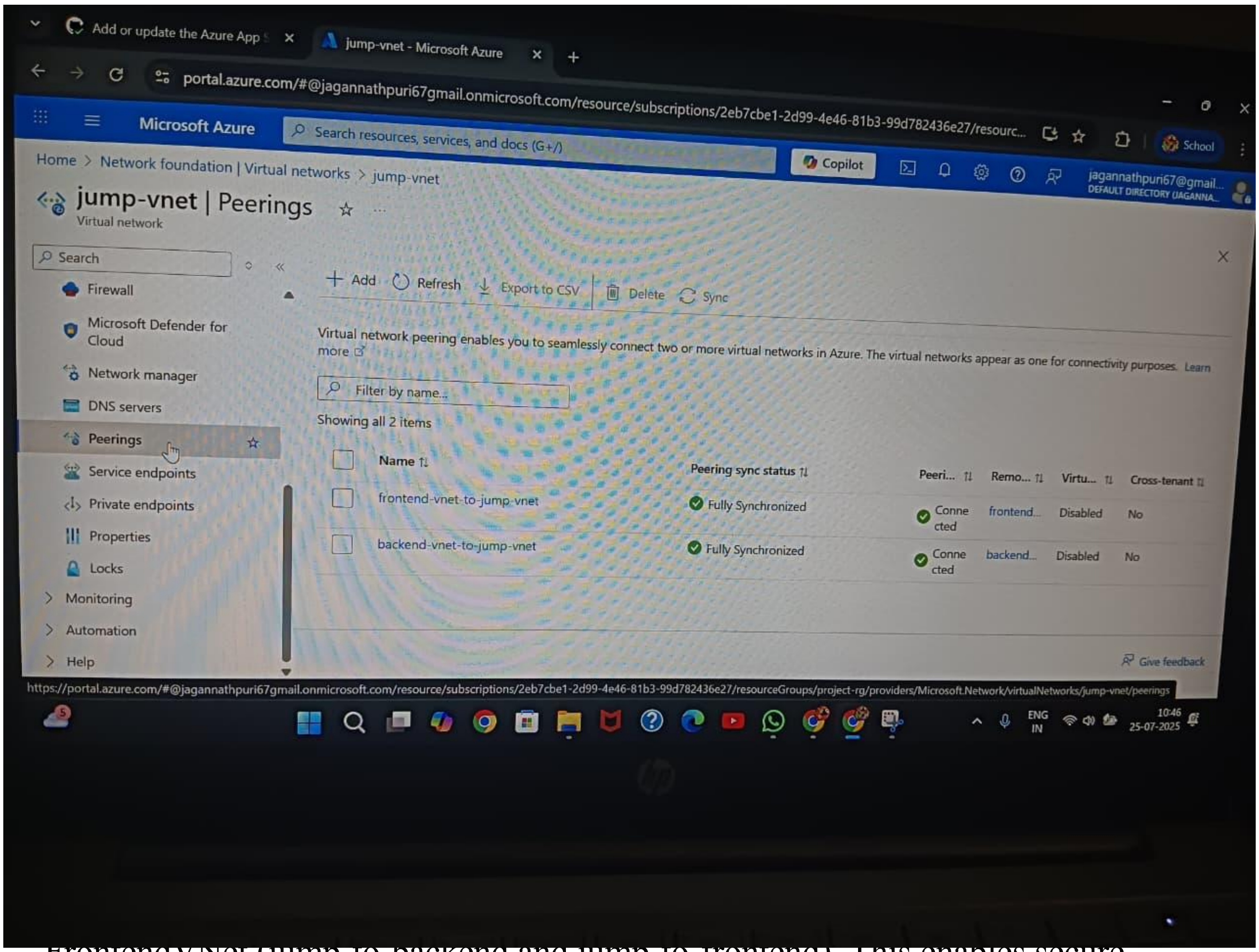
Microsoft

You are viewing a new version of Browse experience. Click here to access the old experience.

Name ↑	Subscription	Resource Group	Location	Status	Operating syst...	Size
fdvm-1	Azure for Stude...	project-rg	Central India	Running	Linux	Standard_B1s
fdvm-2	Azure for Stude...	project-rg	Central India	Running	Linux	Standard_B1s
jump-vm	Azure for Stude...	project-rg	Central India	Running	Linux	Standard_B1s

Showing 1 - 3 of 3. Display count: auto

I created three Virtual Machines: fdvm1 and fdvm2 under the frontend VNet for hosting frontend services, and jumpvm under the JumpNet VNet to securely access other VMs via SSH tunneling. These VMs aid in managing and testing deployments across networks.


Add or update the Azure App S × | jump-vnet - Microsoft Azure ×

portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourc...

Microsoft Azure | Search resources, services, and docs (G+/) | Copilot | jagannathpuri67@gmail... DEFAULT DIRECTORY (JAGANNA...

Home > Network foundation | Virtual networks > jump-vnet

jump-vnet | Peerings
Virtual network

Search
- Firewall
- Microsoft Defender for Cloud
- Network manager
- DNS servers
- Peerings
- Service endpoints
- Private endpoints
- Properties
- Locks
- Monitoring
- Automation
- Help

+ Add | Refresh | Export to CSV | Delete | Sync

Virtual network peering enables you to seamlessly connect two or more virtual networks in Azure. The virtual networks appear as one for connectivity purposes. Learn more

Filter by name...

Showing all 2 items

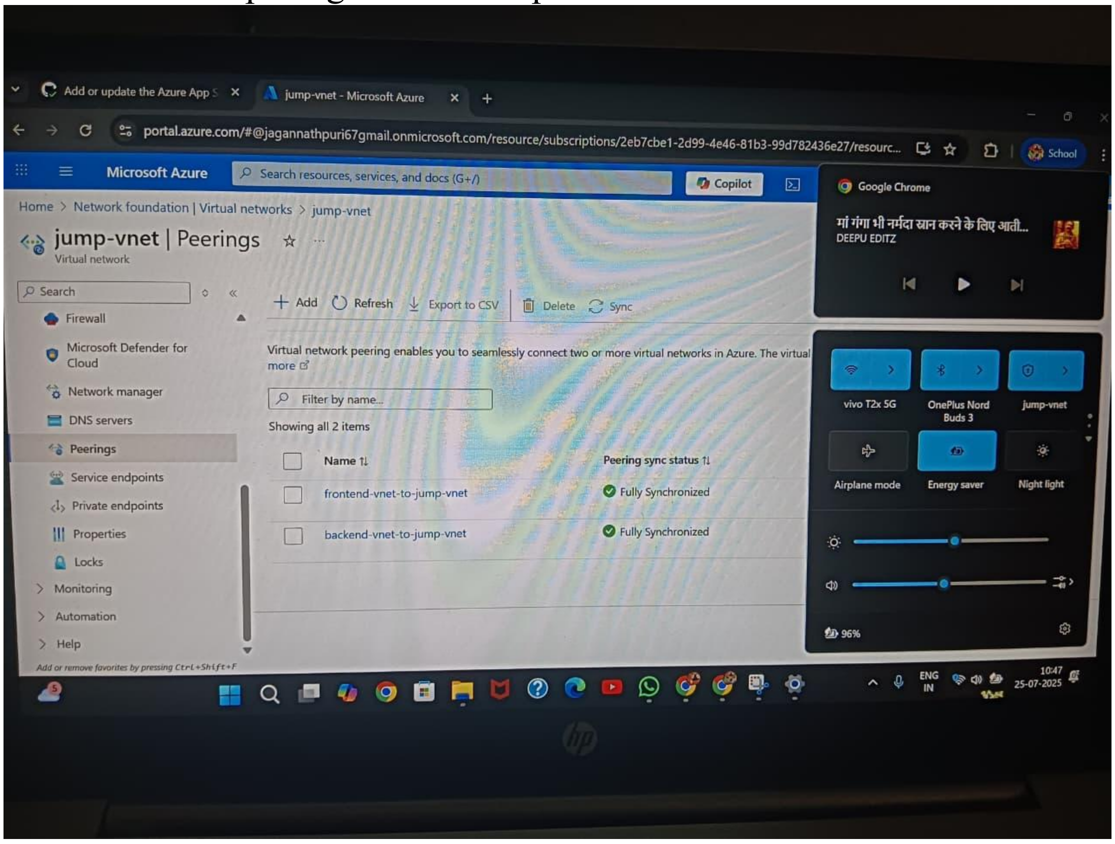
Name	Peering sync status	Peeri...	Remo...	Virtu...	Cross-tenant
frontend-vnet-to-jump-vnet	Fully Synchronized	Connected	frontend...	Disabled	No
backend-vnet-to-jump-vnet	Fully Synchronized	Connected	backend...	Disabled	No

Give feedback

https://portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourceGroups/project-rg/providers/Microsoft.Network/virtualNetworks/jump-vnet/peerings

ENG IN 10:46 25-07-2025

FrontendVNet (jump-to-backend and jump-to-frontend). This enables secure and seamless connectivity between the Jump VM and other network resources without exposing them to the public internet.


Add or update the Azure App S × | jump-vnet - Microsoft Azure ×

portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/resourc...

Microsoft Azure | Search resources, services, and docs (G+/) | Copilot

Home > Network foundation | Virtual networks > jump-vnet

jump-vnet | Peerings
Virtual network

Search
- Firewall
- Microsoft Defender for Cloud
- Network manager
- DNS servers
- Peerings
- Service endpoints
- Private endpoints
- Properties
- Locks
- Monitoring
- Automation
- Help

+ Add | Refresh | Export to CSV | Delete | Sync

Virtual network peering enables you to seamlessly connect two or more virtual networks in Azure. The virtual more

Filter by name...

Showing all 2 items

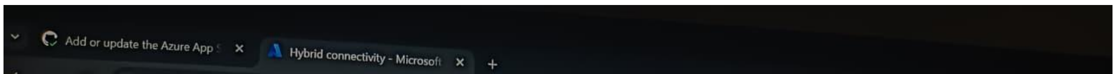
Name	Peering sync status
frontend-vnet-to-jump-vnet	Fully Synchronized
backend-vnet-to-jump-vnet	Fully Synchronized

Add or remove favorites by pressing Ctrl+Shift+F

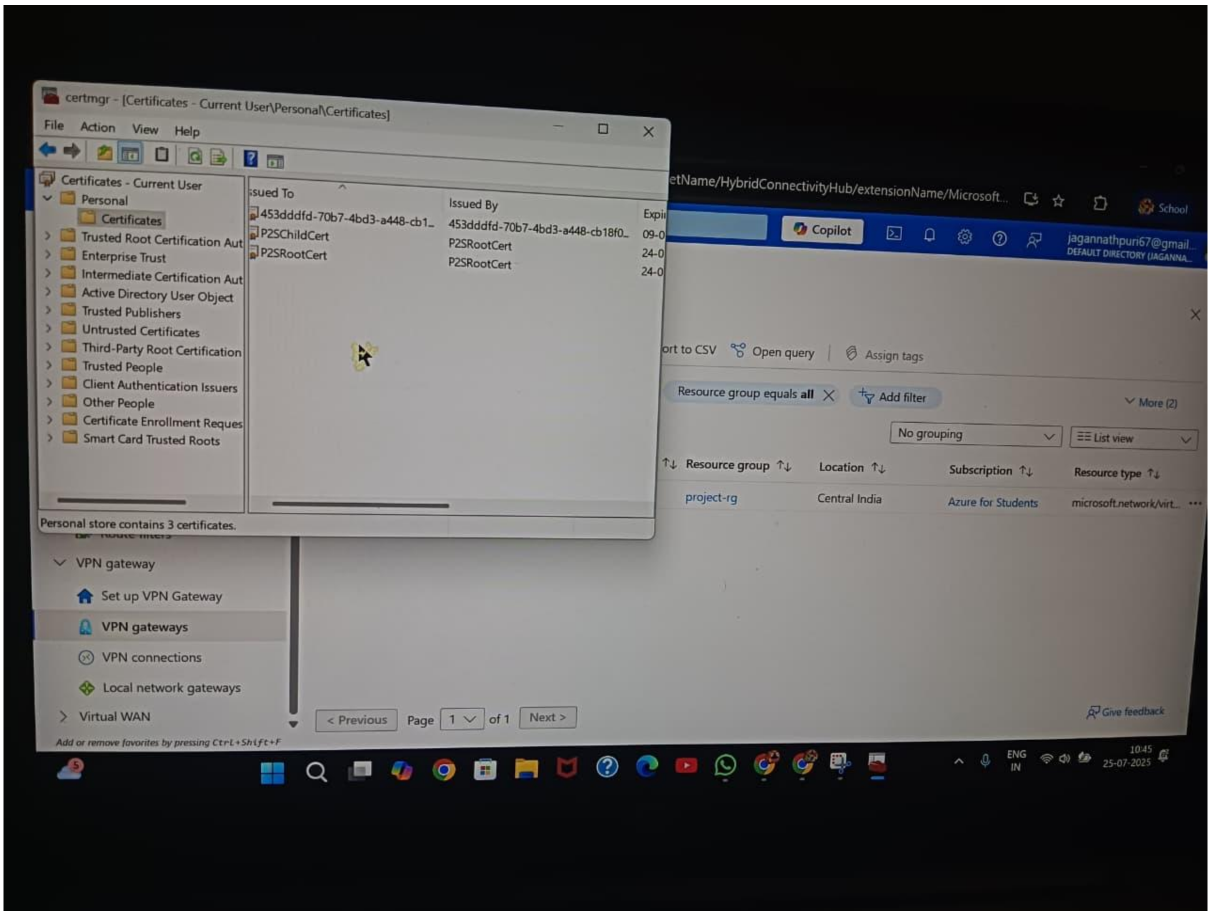
Google Chrome
माँ गंगा भी नर्मदा स्नान करने के लिए आती...
DEEPU EDITZ

vivo T2x 5G | OnePlus Nord Buds 3 | jump-vnet
Airplane mode | Energy saver | Night light
96%

ENG IN 10:47 25-07-2025


Add or update the Azure App S × | Hybrid connectivity - Microsoft ×

I created VPN Gateway (gate-2) in the Backend VNet to enable secure site-to-site or point-to-site communication over IPsec/IKE tunnels. This allows encrypted access to Azure resources from on-premises or other VNets.



certmgr - [Certificates - Current User\Personal\Certificates]

ssued To	Issued By	Expi
453dddfd-70b7-4bd3-a448-cb1...	453dddfd-70b7-4bd3-a448-cb18f0...	09-0
P2SChildCert	P2SRootCert	24-0
P2SRootCert	P2SRootCert	24-0

Personal store contains 3 certificates.

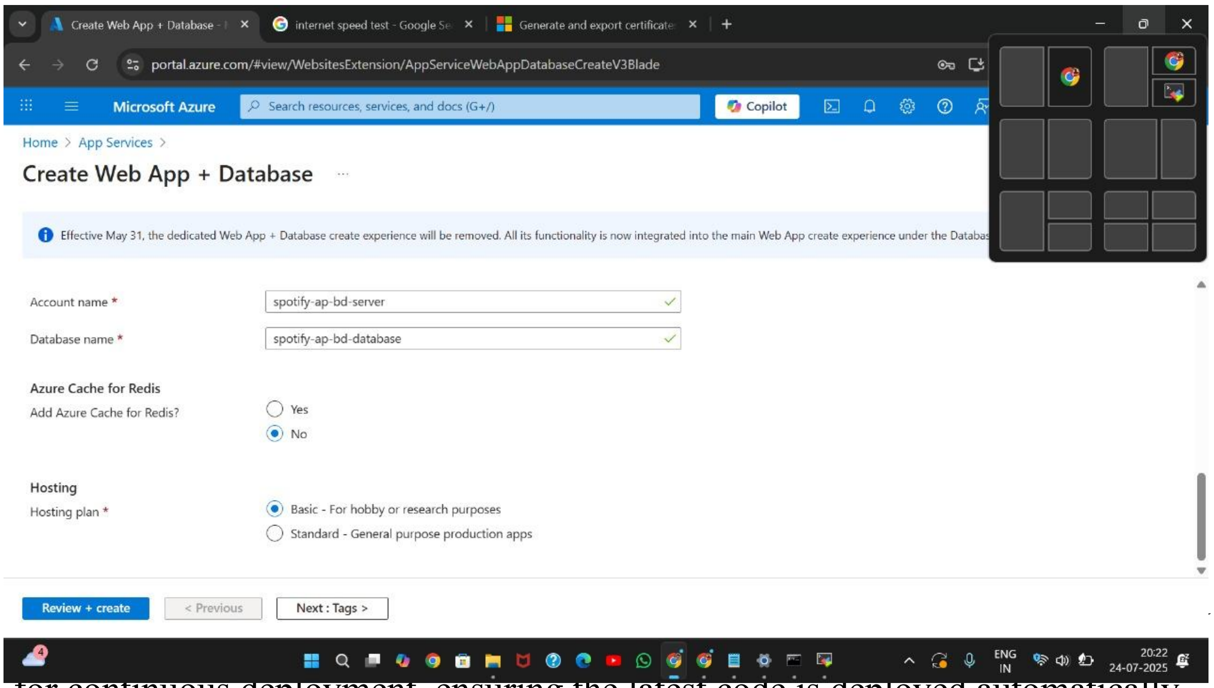
etName/HybridConnectivityHub/extensionName/Microsoft...

jagannathpuri67@gmail...
DEFAULT DIRECTORY (JAGANNA...

Resource group equals **all**

Resource group	Location	Subscription	Resource type
project-rg	Central India	Azure for Students	microsoft.network/virt...

I generated a Root certificate and derived a Client certificate from it to establish secure Point-to-Site (P2S) VPN connectivity. The Root cert was uploaded to Azure VPN Gateway, while the Client cert was installed on the local machine to authenticate VPN access.


Home > App Services >

Create Web App + Database

Effective May 31, the dedicated Web App + Database create experience will be removed. All its functionality is now integrated into the main Web App create experience under the Databas

Account name *	spotify-ap-bd-server
Database name *	spotify-ap-bd-database

Azure Cache for Redis

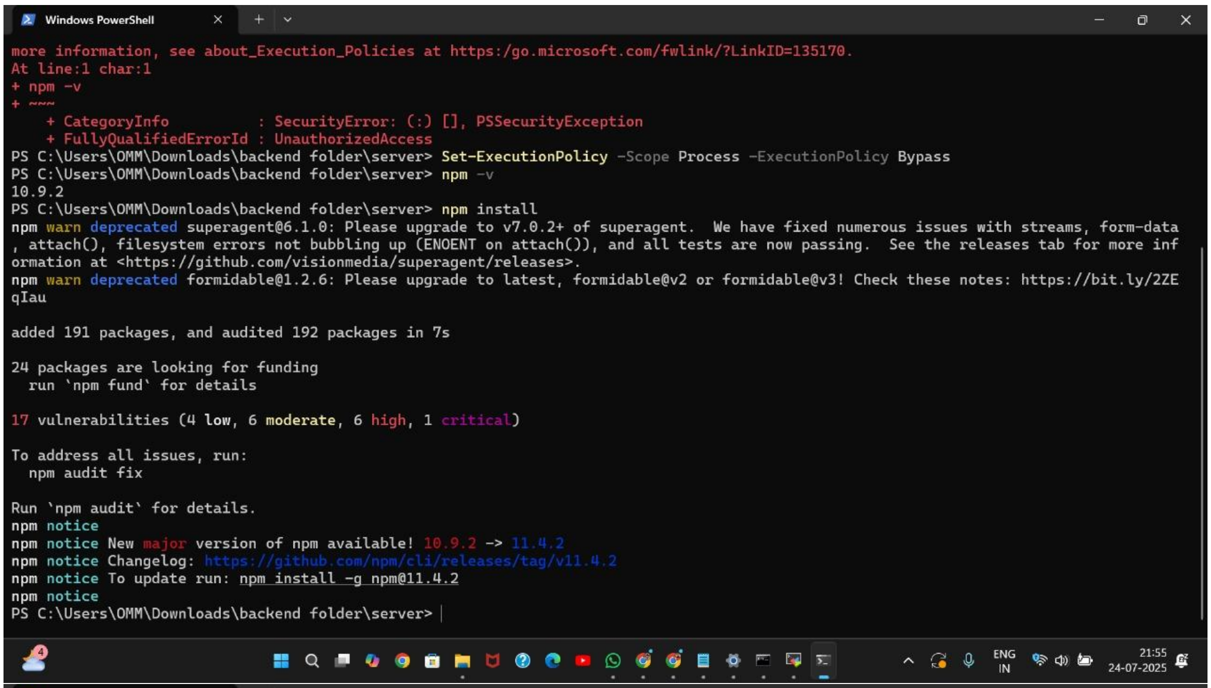
Add Azure Cache for Redis? Yes [✓ No]

Hosting

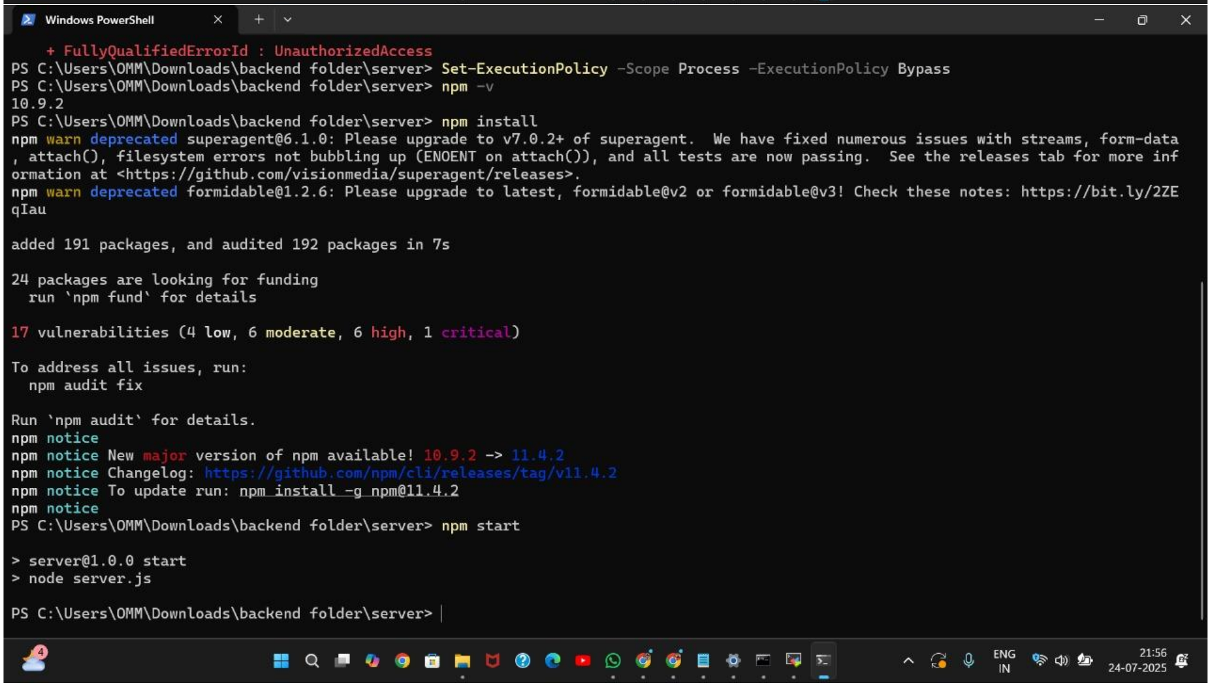
Hosting plan * [✓ Basic - For hobby or research purposes] Standard - General purpose production apps

Create Web App + Database
internet speed test - Google Se
Generate and export certificate
portal.azure.com/#view/WebsitesExtension/AppServiceWebAppDatabaseCreateV3Blade
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Review + create
< Previous
Next : Tags >
ENG IN
20:22
24-07-2025

for continuous deployment, ensuring the latest code is deployed automatically to the web app.

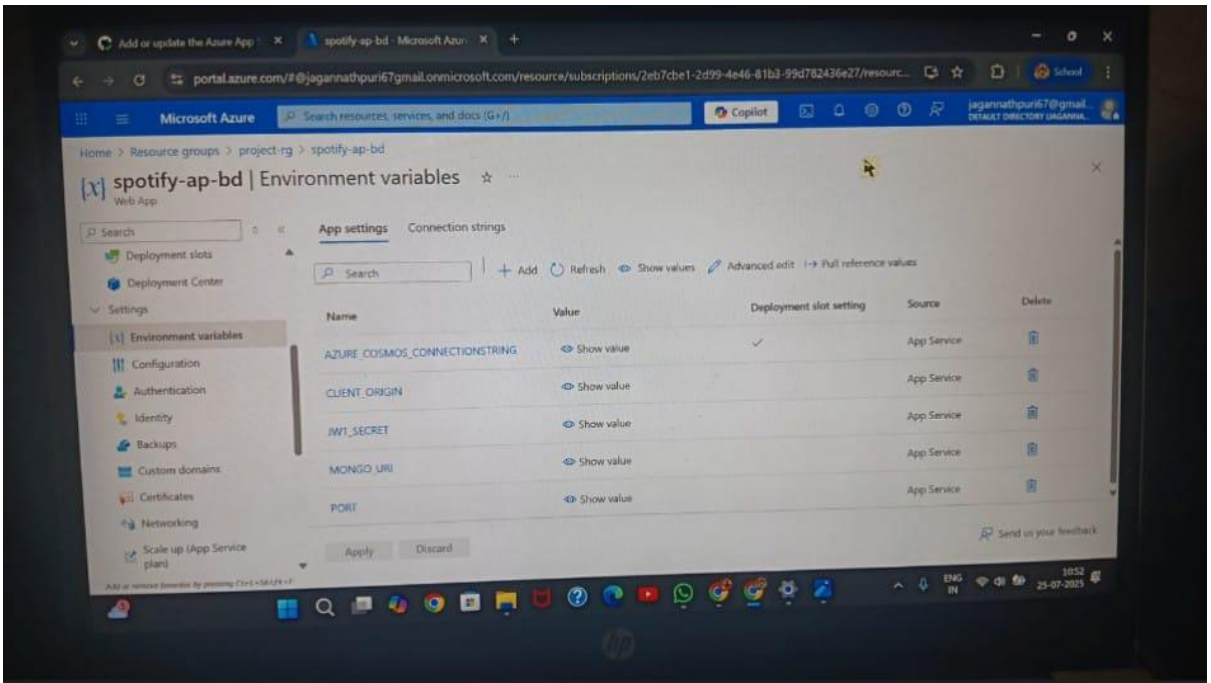


```
more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ npm -v
+ ~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\OMM\Downloads\backend folder\server> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
PS C:\Users\OMM\Downloads\backend folder\server> npm -v
10.9.2
PS C:\Users\OMM\Downloads\backend folder\server> npm install
npm warn deprecated superagent@6.1.0: Please upgrade to v7.0.2+ of superagent.  We have fixed numerous issues with streams, form-data
, attach(), filesystem errors not bubbling up (ENOENT on attach()), and all tests are now passing.  See the releases tab for more inf
ormation at <https://github.com/visionmedia/superagent/releases>.
npm warn deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZE
qIau

added 191 packages, and audited 192 packages in 7s

24 packages are looking for funding
  run `npm fund` for details

17 vulnerabilities (4 low, 6 moderate, 6 high, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.4.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.2
npm notice To update run: npm install -g npm@11.4.2
npm notice
PS C:\Users\OMM\Downloads\backend folder\server>
```



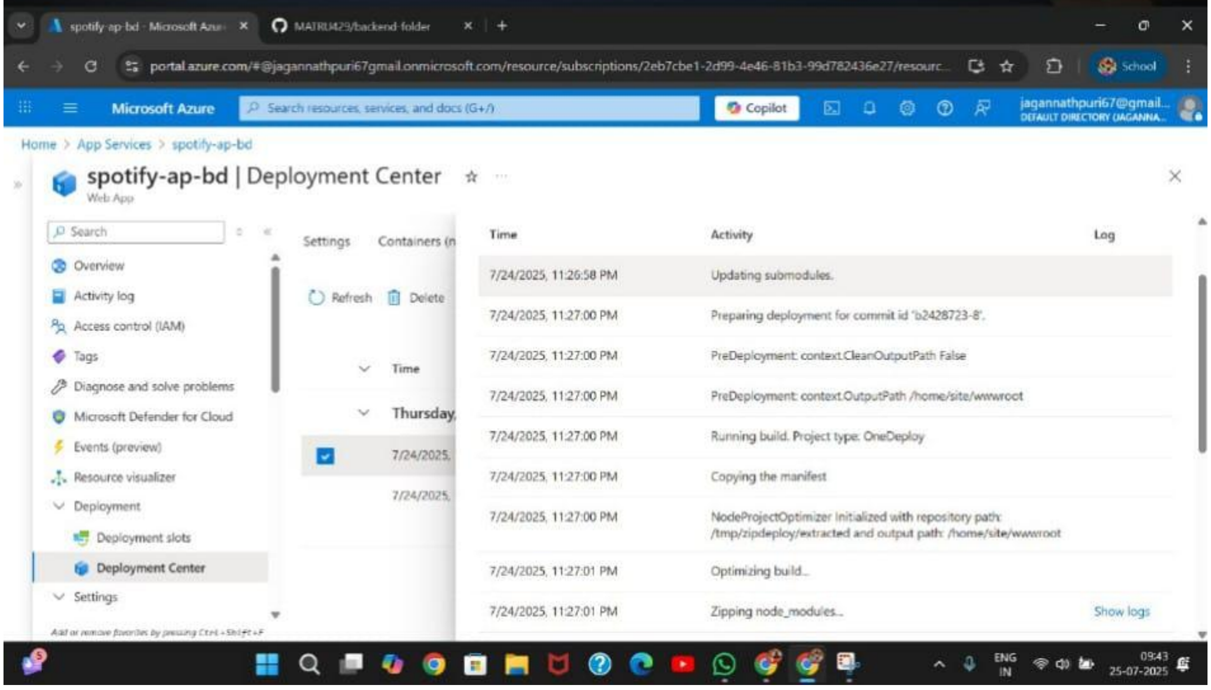
```
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\OMM\Downloads\backend folder\server> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
PS C:\Users\OMM\Downloads\backend folder\server> npm -v
10.9.2
PS C:\Users\OMM\Downloads\backend folder\server> npm install
npm warn deprecated superagent@6.1.0: Please upgrade to v7.0.2+ of superagent.  We have fixed numerous issues with streams, form-data
, attach(), filesystem errors not bubbling up (ENOENT on attach()), and all tests are now passing.  See the releases tab for more inf
ormation at <https://github.com/visionmedia/superagent/releases>.
npm warn deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZE
qIau

added 191 packages, and audited 192 packages in 7s

24 packages are looking for funding
  run `npm fund` for details

17 vulnerabilities (4 low, 6 moderate, 6 high, 1 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.4.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.2
npm notice To update run: npm install -g npm@11.4.2
npm notice
PS C:\Users\OMM\Downloads\backend folder\server> npm start

> server@1.0.0 start
> node server.js

PS C:\Users\OMM\Downloads\backend folder\server>
```

I ran the command `npm install` in the project directory to install all the required dependencies listed in the `package.json` file. This sets up the Node.js environment with the necessary libraries needed to run the Spotify backend server.


spotify-ap-bd | Environment variables

App settings | Connection strings

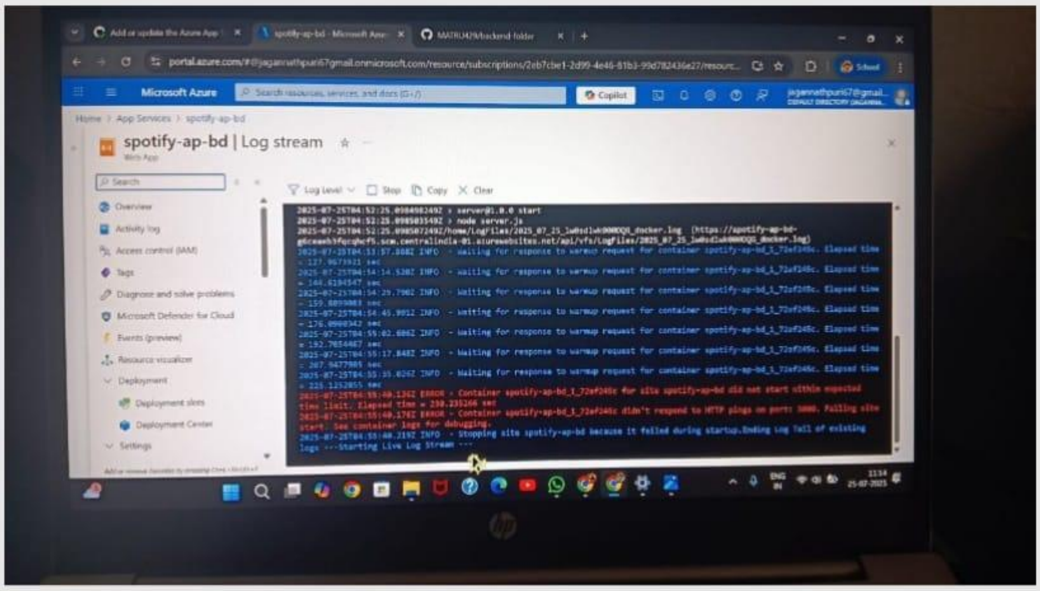
Name	Value	Deployment slot setting	Source	Delete
AZURE_COSMOS_CONNECTIONSTRING	Show value	✓	App Service	🗑
CLIENT_ORIGIN	Show value		App Service	🗑
JWT_SECRET	Show value		App Service	🗑
MONGO_URI	Show value		App Service	🗑
PORT	Show value		App Service	🗑


spotify-ap-bd | Deployment Center

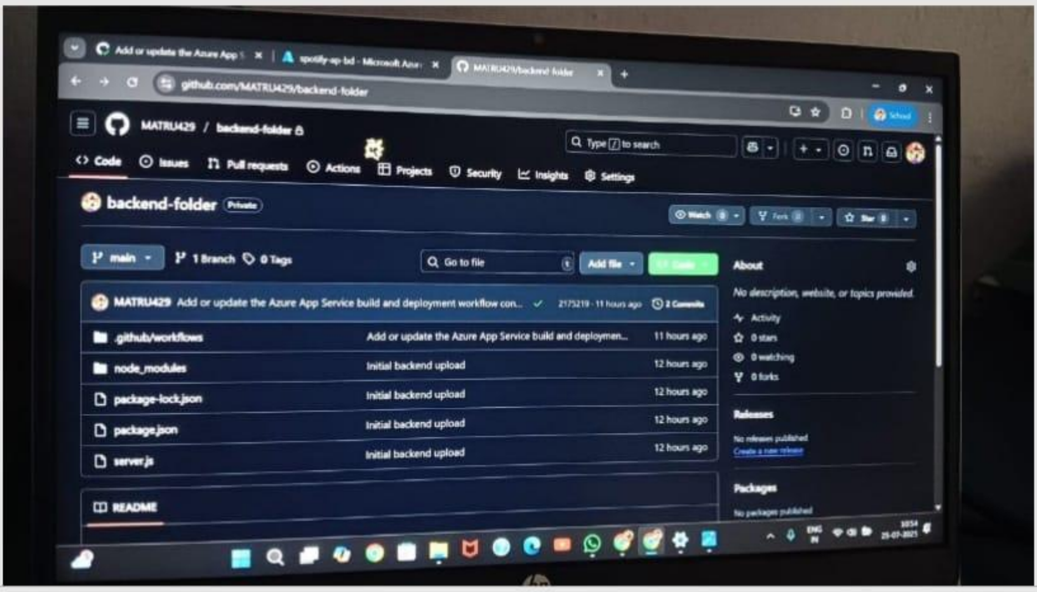
Time	Activity	Log
7/24/2025, 11:26:58 PM	Updating submodules.	
7/24/2025, 11:27:00 PM	Preparing deployment for commit id 'b2428723-8'.	
7/24/2025, 11:27:00 PM	PreDeployment: context.CleanOutputPath False	
7/24/2025, 11:27:00 PM	PreDeployment: context.OutputPath /home/site/wwwroot	
7/24/2025, 11:27:00 PM	Running build. Project type: OneDeploy	
7/24/2025, 11:27:00 PM	Copying the manifest	
7/24/2025, 11:27:00 PM	NodeProjectOptimizer Initialized with repository path: /tmp/zipdeploy/extracted and output path: /home/site/wwwroot	
7/24/2025, 11:27:01 PM	Optimizing build...	
7/24/2025, 11:27:01 PM	Zipping node_modules...	Show logs

I configured environment variables such as AZURE_COSMOS_CONNSTRING, CLIENT_ORIGIN, JWT_SECRET, MONGO_URI, and PORT in the Azure Web App to securely pass sensitive values to the backend server without hardcoding them in the codebase.

Then, I set up the Deployment Center in Azure App Services by connecting my GitHub repository, enabling automated deployment via GitHub Actions whenever I push changes to the main branch.



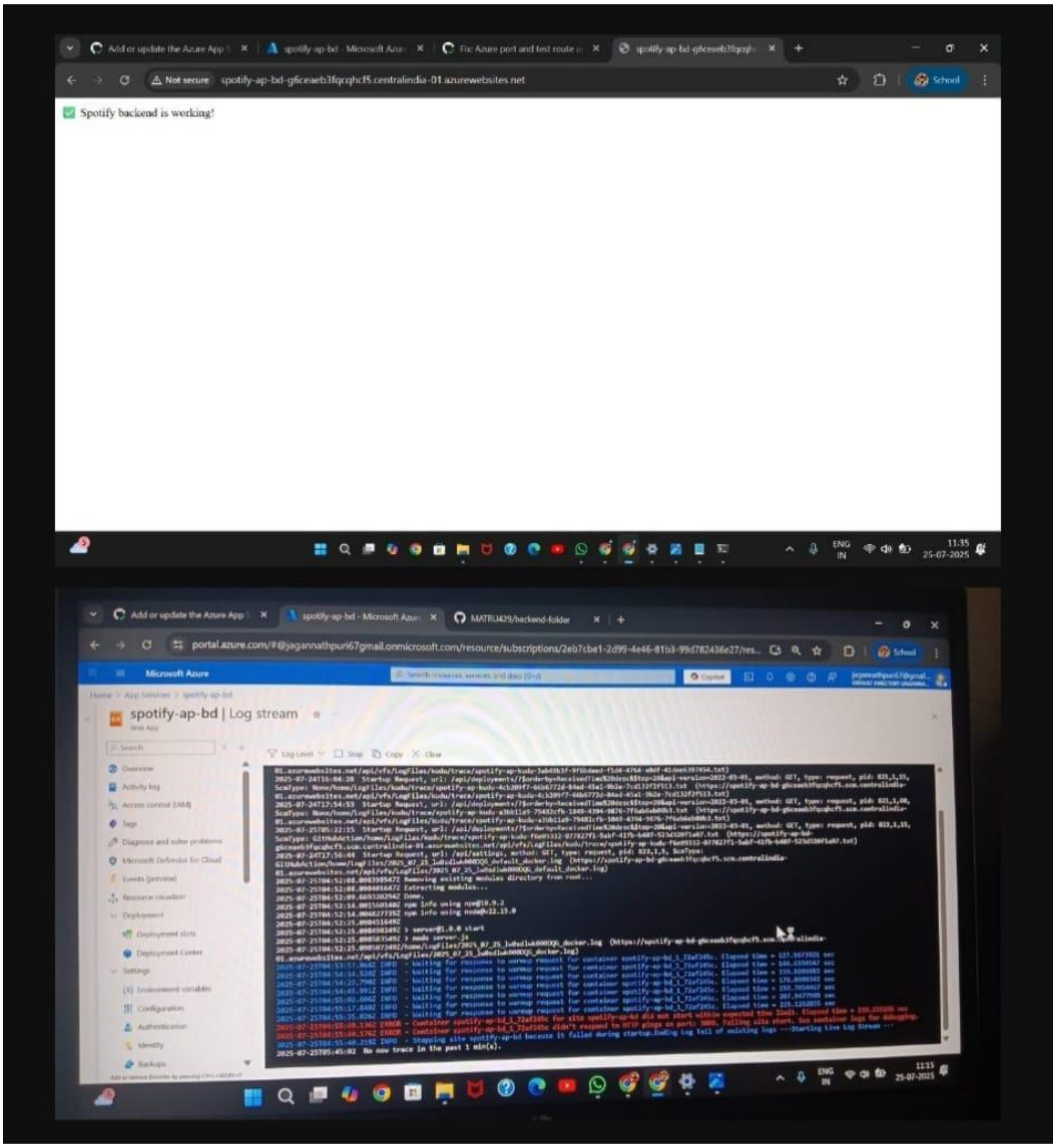
```
2025-07-25T04:52:25.0984982492 > server@1.0.0 start
2025-07-25T04:52:25.0985035492 > node server.js
2025-07-25T04:52:25.0985072492/home/LogFiles/2025_07_25_lw0sdlwk000QQ_docker.log  (https://spotify-ap-bd-
g6cewehbfqcghcf5.scm.centralindia-01.azurewebsites.net/api/vfs/LogFiles/2025_07_25_lw0sdlwk000QQ_docker.log)
2025-07-25T04:53:57.888Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 127.9673921 sec
2025-07-25T04:54:14.528Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 144.6194547 sec
2025-07-25T04:54:29.798Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 159.8899083 sec
2025-07-25T04:54:45.991Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 176.0900342 sec
2025-07-25T04:55:02.686Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 192.7854467 sec
2025-07-25T04:55:17.848Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 207.9477985 sec
2025-07-25T04:55:35.026Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72ef245c. Elapsed time
= 225.1252855 sec
2025-07-25T04:55:40.136Z ERROR - Container spotify-ap-bd_1_72ef245c for site spotify-ap-bd did not start within expected
time limit. Elapsed time = 230.235266 sec
2025-07-25T04:55:40.176Z ERROR - Container spotify-ap-bd_1_72ef245c didn't respond to HTTP pings on port: 3000, failing site
start. See container logs for debugging.
2025-07-25T04:55:40.219Z INFO  - Stopping site spotify-ap-bd because it failed during startup.Ending Log Tail of existing
logs ---Starting Live Log Stream ---
```



MATRU429 Add or update the Azure App Service build and deployment workflow con... ✓ 2175219 · 11 hours ago 2 Commits

.github/workflows	Add or update the Azure App Service build and deploymen...	11 hours ago
node_modules	Initial backend upload	12 hours ago
package-lock.json	Initial backend upload	12 hours ago
package.json	Initial backend upload	12 hours ago
server.js	Initial backend upload	12 hours ago

I used Azure App Service's Log Stream feature to monitor real-time logs from the deployed backend application. This helped me track server startup, debug application errors, and verify successful connection to services like Cosmos DB directly from the Azure portal.



spotify-ap-bd-g6ceaeb3fqcqhcf5.centralindia-01.azurewebsites.net

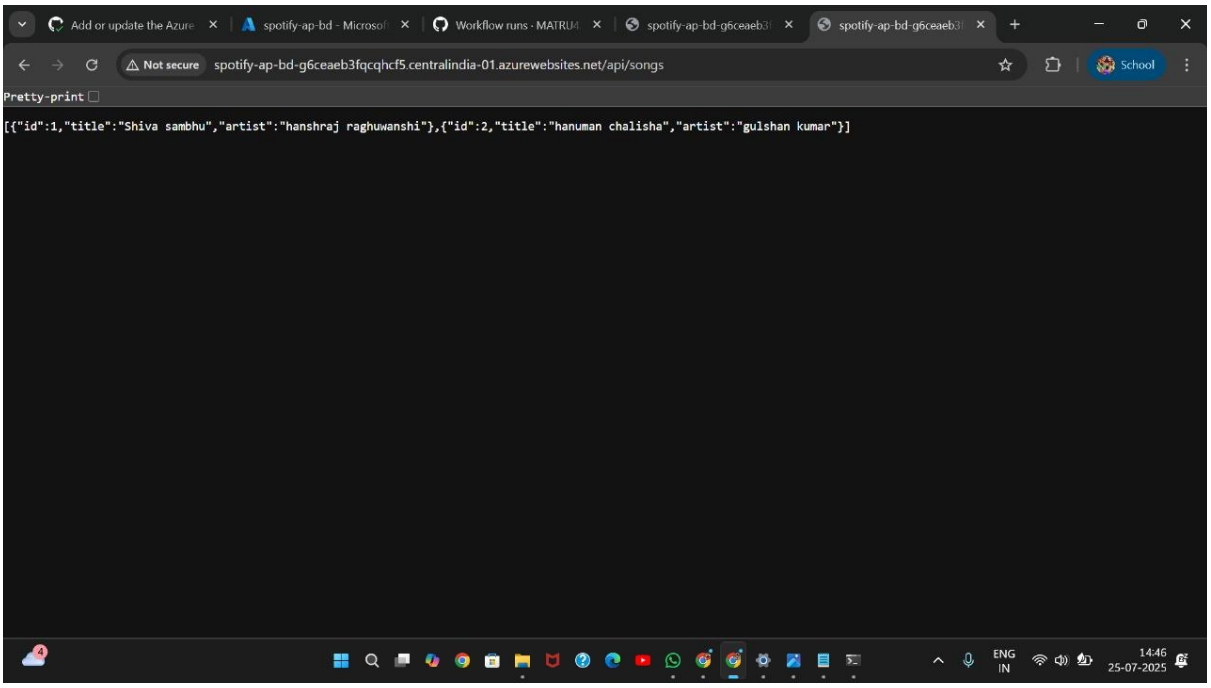
✅ Spotify backend is working!

portal.azure.com/#@jagannathpuri67gmail.onmicrosoft.com/resource/subscriptions/2eb7cbe1-2d99-4e46-81b3-99d782436e27/res...

spotify-ap-bd | Log stream

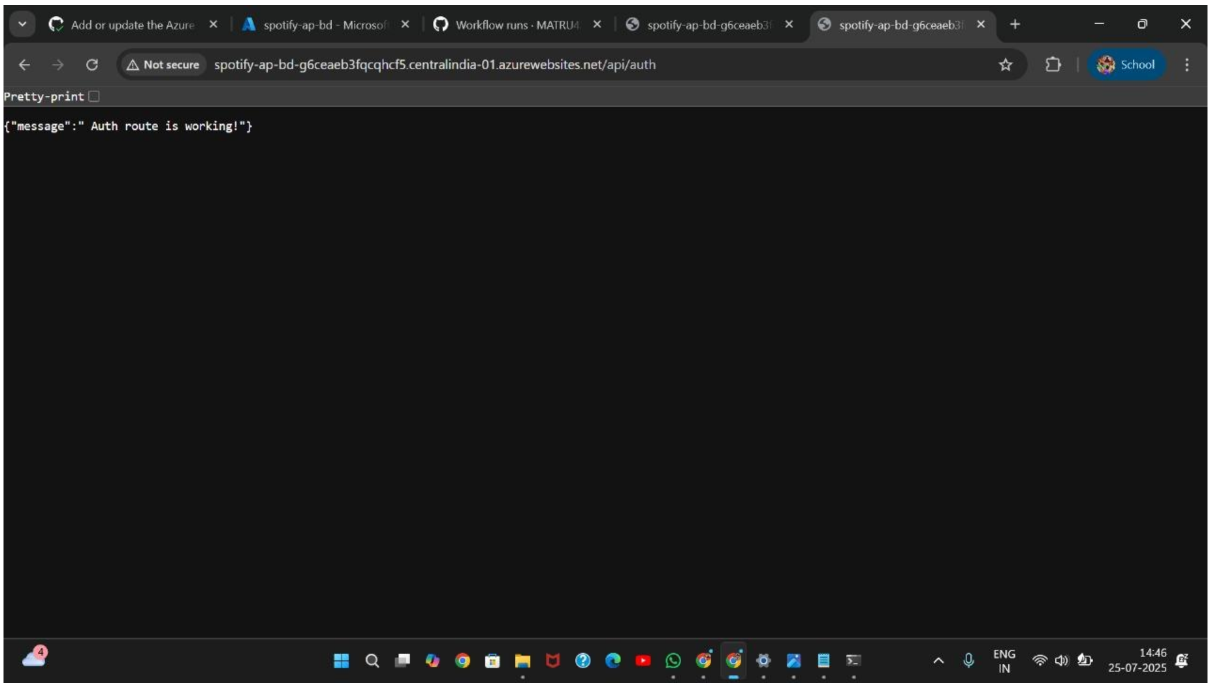
```
01.azurewebsites.net/api/vfs/LogFiles/kudu/trace/spotify-ap-kudu-3ab49b3f-9f5bdaed-f1d4-4764-a8df-41dee5397454.txt)
2025-07-24T16:04:28  Startup Request, url: /api/deployments/?$orderby=ReceivedTime%20desc&$top=20&api-version=2022-03-01, method: GET, type: request, pid: 821,1,15,
ScmType: None/home/LogFiles/kudu/trace/spotify-ap-kudu-4cb209f7-66b6772d-84ed-45a1-9b2a-7cd132f2f513.txt  (https://spotify-ap-bd-g6ceaeb3fqcqhcf5.scm.centralindia-
01.azurewebsites.net/api/vfs/LogFiles/kudu/trace/spotify-ap-kudu-4cb209f7-66b6772d-84ed-45a1-9b2a-7cd132f2f513.txt)
2025-07-24T17:54:53  Startup Request, url: /api/deployments/?$orderby=ReceivedTime%20desc&$top=20&api-version=2022-03-01, method: GET, type: request, pid: 821,1,08,
ScmType: None/home/LogFiles/kudu/trace/spotify-ap-kudu-a3bb11a9-79482cfb-1049-4394-9876-7f6ab6eb00b3.txt  (https://spotify-ap-bd-g6ceaeb3fqcqhcf5.scm.centralindia-
01.azurewebsites.net/api/vfs/LogFiles/kudu/trace/spotify-ap-kudu-a3bb11a9-79482cfb-1049-4394-9876-7f6ab6eb00b3.txt)
2025-07-25T05:22:15  Startup Request, url: /api/deployments/?$orderby=ReceivedTime%20desc&$top=20&api-version=2022-03-01, method: GET, type: request, pid: 821,1,15,
ScmType: GitHubAction/home/LogFiles/kudu/trace/spotify-ap-kudu-f6e09332-077827f1-5abf-41fb-b407-523d320f1a07.txt  (https://spotify-ap-bd-
g6ceaeb3fqcqhcf5.scm.centralindia-01.azurewebsites.net/api/vfs/LogFiles/kudu/trace/spotify-ap-kudu-f6e09332-077827f1-5abf-41fb-b407-523d320f1a07.txt)
2025-07-24T17:56:44  Startup Request, url: /api/settings, method: GET, type: request, pid: 821,1,5, ScmType:
GitHubAction/home/LogFiles/2025_07_25_lw0sdlwk000QQS_default_docker.log  (https://spotify-ap-bd-g6ceaeb3fqcqhcf5.scm.centralindia-
01.azurewebsites.net/api/vfs/LogFiles/2025_07_25_lw0sdlwk000QQS_default_docker.log)
2025-07-25T04:52:08.0083905477Z Removing existing modules directory from root...
2025-07-25T04:52:08.0984016477Z Extracting modules...
2025-07-25T04:52:09.6693202942Z Done.
2025-07-25T04:52:14.0015601482Z npm info using npm@10.9.2
2025-07-25T04:52:14.0048277735Z npm info using node@v22.15.0
2025-07-25T04:52:25.0984516492Z
2025-07-25T04:52:25.0984502492Z > server@1.0.0 start
2025-07-25T04:52:25.0985035492Z > node server.js
2025-07-25T04:52:25.0985072492/home/LogFiles/2025_07_25_lw0sdlwk000QQS_docker.log  (https://spotify-ap-bd-g6ceaeb3fqcqhcf5.scm.centralindia-
01.azurewebsites.net/api/vfs/LogFiles/2025_07_25_lw0sdlwk000QQS_docker.log)
2025-07-25T04:53:57.066Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 127.5673921 sec
2025-07-25T04:54:14.528Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 144.6350547 sec
2025-07-25T04:54:29.798Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 159.8899083 sec
2025-07-25T04:54:45.991Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 176.0980342 sec
2025-07-25T04:55:02.606Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 192.7050467 sec
2025-07-25T04:55:17.840Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 207.9477605 sec
2025-07-25T04:55:35.026Z INFO  - Waiting for response to warmup request for container spotify-ap-bd_1_72af245c. Elapsed time = 225.1212855 sec
2025-07-25T04:55:40.136Z ERROR - Container spotify-ap-bd_1_72af245c for site spotify-ap-bd did not start within expected time limit. Elapsed time = [illegible] sec
2025-07-25T04:55:40.176Z ERROR - Container spotify-ap-bd_1_72af245c didn't respond to HTTP pings on port: 5000, failing site start. See container logs for debugging.
2025-07-25T04:55:40.219Z INFO  - Stopping site spotify-ap-bd because it failed during startup.Ending Log Tail of existing logs ---Starting Live Log Stream ---
2025-07-25T05:45:02  No new trace in the past 1 min(s).
```

After deploying the backend to Azure via GitHub Actions, I verified the deployment by visiting the Azure-hosted endpoint (e.g., https://.azurewebsites.net/api/test). The message “Spotify backend is working” confirmed that the backend server was successfully deployed and running on Azure.


spotify-ap-bd-g6ceaeb3fqcqhcf5.centralindia-01.azurewebsites.net/api/songs
Pretty-print

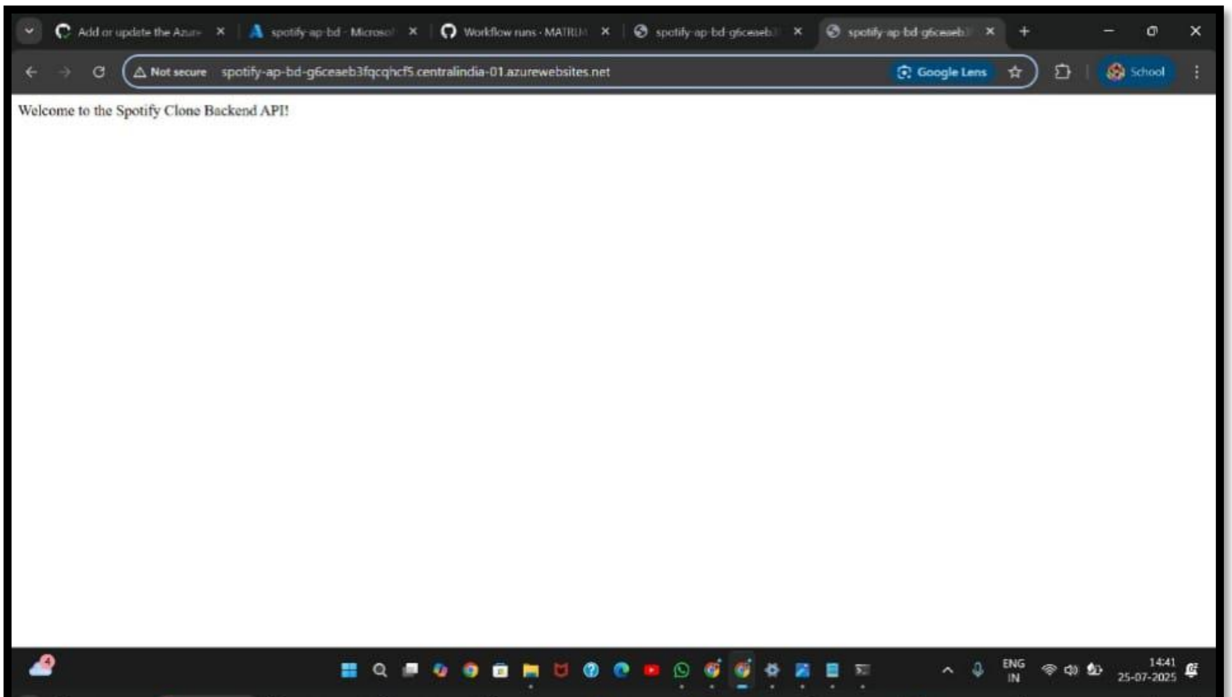
```
[{"id":1,"title":"Shiva sambhu","artist":"hanshraj raghuwanshi"},{"id":2,"title":"hanuman chalisha","artist":"gulshan kumar"}]
```

After deploying the backend, I successfully added a song to the database using a POST request to /api/songs. I provided song details like title and artist, and verified the entry through GET requests or Cosmos DB, confirming that data was being stored and retrieved correctly.

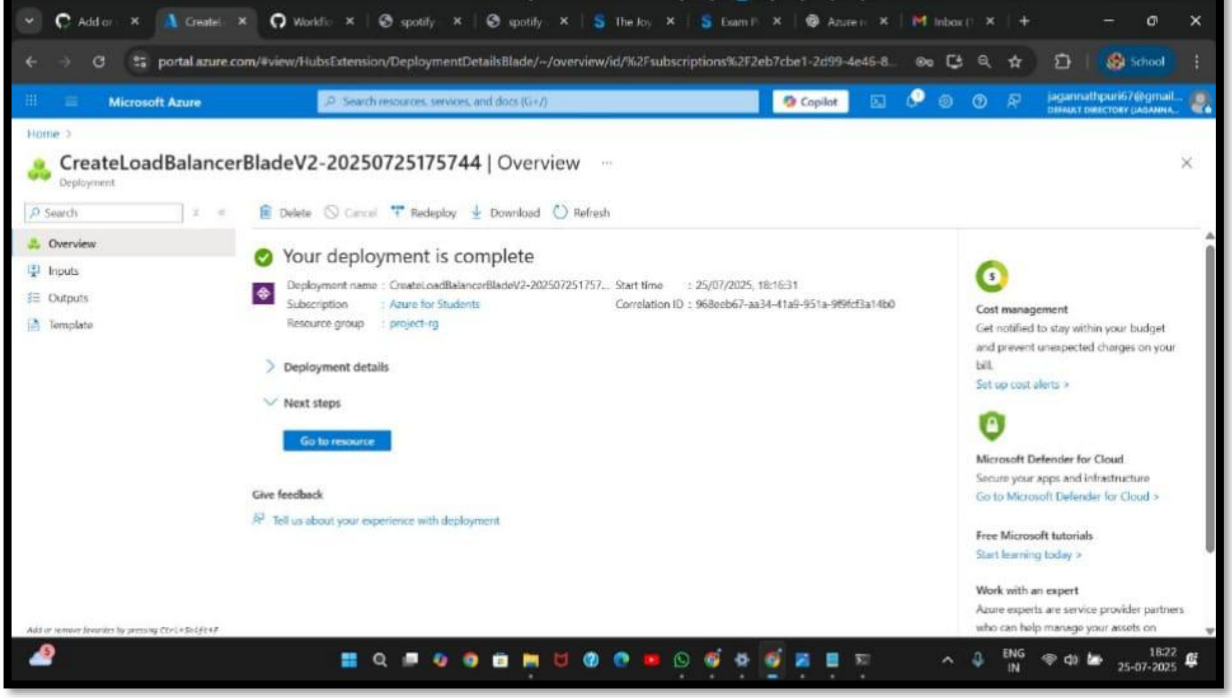

spotify-ap-bd-g6ceaeb3fqcqhcf5.centralindia-01.azurewebsites.net/api/auth
Pretty-print

```
{"message":" Auth route is working!"}
```

The /api/auth route is successfully responding, confirming that the authentication endpoint is functional. This indicates proper server-side route setup and integration of any required middleware or token handling.


spotify-ap-bd-g6ceaeb3fqcqhcf5.centralindia-01.azurewebsites.net

Welcome to the Spotify Clone Backend API!


portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2F2eb7cbe1-2d99-4e46-8...

CreateLoadBalancerBladeV2-20250725175744 | Overview

Deployment

Your deployment is complete

Deployment name : CreateLoadBalancerBladeV2-202507251757...
Subscription : Azure for Students
Resource group : project-rg
Start time : 25/07/2025, 18:16:31
Correlation ID : 968eeb67-aa34-41a9-951a-9f9fcf3a14b0

The Spotify app backend is now successfully deployed and operational on Azure. All key routes like /api/test, /api/songs, and /api/auth are functioning correctly, confirming that the backend server, database connection, and deployment pipeline are properly configured.

After successfully deploying the Spotify backend and verifying its functionality, I proceeded to enhance the scalability and availability of the frontend by creating a Load Balancer (frontend-lb). This load balancer distributes incoming user traffic across multiple instances efficiently. Following this, I configured a Virtual Machine Scale Set (front-vmss) to automatically manage and scale multiple frontend VMs behind the load balancer, ensuring the application can handle increased load and maintain high performance.

Conclusion-

- Successfully designed and deployed a cloud-based Spotify backend application on Microsoft Azure.

- Created and configured three VNets (frontend, backend, and jumpnet) to isolate application tiers securely.

- Provisioned multiple subnets for APIs, databases, and management operations within respective VNets.

- Established VM instances (fdvm1, fdvm2, and jumpvm) to host and manage workloads and enable secure access.

- Implemented secure VNet peering and VPN gateway with root and client certificates for encrypted connectivity.

- Deployed a Node.js backend integrated with Azure Cosmos DB using Mongoose, hosted via Azure Web App.

- Environment variables like Mongo URI, JWT secrets, and client origins were configured securely in Azure settings.

- GitHub Actions was used to automate CI/CD workflows for backend deployment from the linked repository.

- Verified functionality using Log Stream and Postman, confirming routes like /api/test, /api/auth, and /api/songs worked correctly.

- Added load balancer and created VMSS (Virtual Machine Scale Set) for front-end scalability and high availability.