

## Experiment - 4

### 1. Create an Azure Storage Account

1. Open Azure Portal → Navigate to Storage Accounts → Click + Create.

2. Set the following configurations:

- Resource Group: secureCloud
- Storage Account Name: securecloudstorage2025
- Region: Central India
- Performance: Standard
- Replication: Locally Redundant Storage (LRS)

3. Click Review + Create → Click Create.

The screenshot shows the 'Create a storage account' wizard in the Azure portal. The 'Subscription' dropdown is set to 'Azure for Students'. The 'Resource group' dropdown is set to 'secureCloud'. Under 'Instance details', the 'Storage account name' is 'securecloudstorage', 'Region' is '(Asia Pacific) Central India', 'Primary service' is 'Azure Blob Storage or Azure Data Lake Storage Gen 2', 'Performance' is set to 'Standard' (selected), and 'Redundancy' is 'Locally-redundant storage (LRS)'. At the bottom, there are 'Previous' and 'Next' buttons, a 'Review + create' button, and a 'Give feedback' link.

### 2. Configure a Private Endpoint (VNet & Subnet)

1. Go to Virtual Networks → Click + Create.

2. Set the following configurations:

- Name: secureVNet
- Region: Central India
- Subnet Name: private-subnet

3. Click Create.

**Deployment**

**Overview**

**Your deployment is complete**

Deployment name : secure-vnet-1742786260164  
Subscription : Azure for Students  
Resource group : secureCloud  
Start time : [redacted]  
Correlation ID : e32e6eaf-5720-4798-b432-48bc1e61d0d9

**Deployment details**

Resource	Type
secure-vnet	Virtual network

**Next steps**

Give feedback  
[Tell us about your experience with deployment](#)

### 3. Deploy a Virtual Machine (VM) and Configure S

1. Go to Azure Portal → Click + Create a resource → Select Virtual Machine.

2. Configure:

- **Resource Group:** secureCloud
- **VM Name:** secureVM
- **Region:** Central India
- **Image:** Ubuntu 22.04
- **Authentication:** S Key

3. Click Review + Create → Click Create.

## Generate S Key and Connect

### 1. Open Powerell/CLI and run:

```
s-keygen -t rsa -b 4096 -f securecloud-key
```

```
C:\Users\KRISHNA>ssh-keygen -t rsa -b 4096 -f securecloud-key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in securecloud-key
Your public key has been saved in securecloud-key.pub
The key fingerprint is:
SHA256:uZfSoZXrroVFyhx5mF2YtBkbDXutLnpLChNkMMytaI krishna@LAPTOP-B1N030KM
The key's randomart image is:
+---[RSA 4096]---+
| ..*.. |
| XoO.. |
| B X . . |
| = O o . |
| o S * o |
| . o O * . |
| E * O o |
| = * . |
| .o=... |
+---[SHA256]---+
```

### 2. Copy the public key to Azure VM during setup.

### 3. Find Public IP of VM:

```
az vm list-ip-addresses --resource-group secureCloud --name secureVM --output table
```

### 4. Connect via S:

```
s -i C:\Users\KRISHNA\.s\id_rsa azureuser@<VM_Public_IP>
```

## Edit S Config for Passwordless Login

**1. Edit S config:**

```
sudo nano /etc/s/sd_config
```

**2. Save & restart S:**

```
sudo systemctl restart s
```

```
azureuser@secure-vm:~$ sudo nano /etc/ssh/sshd_config
azureuser@secure-vm:~$ sudo systemctl restart ssh
azureuser@secure-vm:~$ sudo ufw allow OpenSSH
sudo ufw enable
sudo ufw status
Rules updated
Rules updated (v6)
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
Status: active

To                         Action      From
--                         --          --
OpenSSH                      ALLOW       Anywhere
OpenSSH (v6)                  ALLOW       Anywhere (v6)
```

**4. Configure Network Security Group (NSG) and S Rule**

**1. Go to Azure Portal → Search Network Security Groups (NSG) → Click + Create.**

**2. Configure:**

- **Name:** secure-vm-nsg
- **Resource Group:** secureCloud
- **Region:** Central India

**3. Click Create.**

**Create an NSG Rule to Allow S**

**Run in Powerell/CLI:**

```
az network nsg rule create --resource-group secureCloud --nsg-name secure-vm-nsg --
name S-Rule --priority 1000 --access Allow --direction Inbound --protocol Tcp --source-
address-prefixes <YOUR_PUBLIC_IP> --source-port-ranges "*" --destination-address-
prefixes "*" --destination-port-ranges 22
```

```
PS C:\Users\KRISHNA> az network nsg rule create --resource-group secureCloud --nsg-name secure-vmNSG --name SSH-Rule --priority 1100 --access Allow --direction Inbound --protocol Tcp --source-address-prefixes 203.197.213.4 --source-port-ranges "*" --destination-address-prefixes "*" --destination-port-ranges 22
{
  "access": "Allow",
  "destinationAddressPrefix": "*",
  "destinationAddressPrefixes": [],
  "destinationPortRange": "22",
  "destinationPortRanges": [],
  "direction": "Inbound",
  "etag": "W/\"153580c1-2a41-4072-81c4-d9f7ddfd1b41\",
  "id": "/subscriptions/3b5ad032-42d7-4c96-9ca9-ddc07d08b8c5/resourceGroups/secureCloud/providers/Microsoft.Network/networkSecurityGroups/secure-vmNSG/securityRules/SSH-Rule",
  "name": "SSH-Rule",
  "priority": 1100,
  "protocol": "Tcp",
  "provisioningState": "Succeeded",
  "resourceGroup": "secureCloud",
  "sourceAddressPrefix": "203.197.213.4",
  "sourceAddressPrefixes": [],
  "sourcePortRange": "*",
  "sourcePortRanges": [],
  "type": "Microsoft.Network/networkSecurityGroups/securityRules"
}
```

## 5. Configure Encryption for Cloud Storage

1. Go to Storage Account → Click Encryption.
2. Select Customer Managed Keys (CMK).
3. Create a Key Vault:

`az keyvault create --name secureCloudVault --resource-group secureCloud --location centralindia`

4. Generate a Key:

`az keyvault key create --vault-name secureCloudVault --name secureStorageKey --protection software`

5. Attach Key to Storage Account under Encryption Settings.

## 6. Implement Role-Based Access Control (RBAC)

Assign "Reader" Role to a User

1. Run:

`az role assignment create --assignee <YOUR_OBJECT_ID> --role "Reader" --scope "/subscriptions/3b5ad032-42d7-4c96-9ca9-ddc07d08b8c5/resourceGroups/secureCloud/providers/Microsoft.Storage/storageAccounts/securecloudstorage2025"`

## 7. Create a Test User for Intrusion Detection

### 1. Create a test user:

```
az ad user create --display-name "TestUser" --user-principal-name
testuser@kpolaswargmail.onmicrosoft.com --password "Test@12345"
```

```
PS C:\Users\KRISHNA> az ad user create --display-name "TestUser" --user-principal-name testuser@kpolaswargmail.onmicrosoft.com --password "Test@12345"
{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
    "businessPhones": [],
    "displayName": "TestUser",
    "givenName": null,
    "id": "d772e77a-8294-477a-a5ea-c4baa5e6ae9b",
    "jobTitle": null,
    "mail": null,
    "mobilePhone": null,
    "officeLocation": null,
    "preferredLanguage": null,
    "surname": null,
    "userPrincipalName": "testuser@kpolaswargmail.onmicrosoft.com"
}
```

### 2. Assign Reader role:

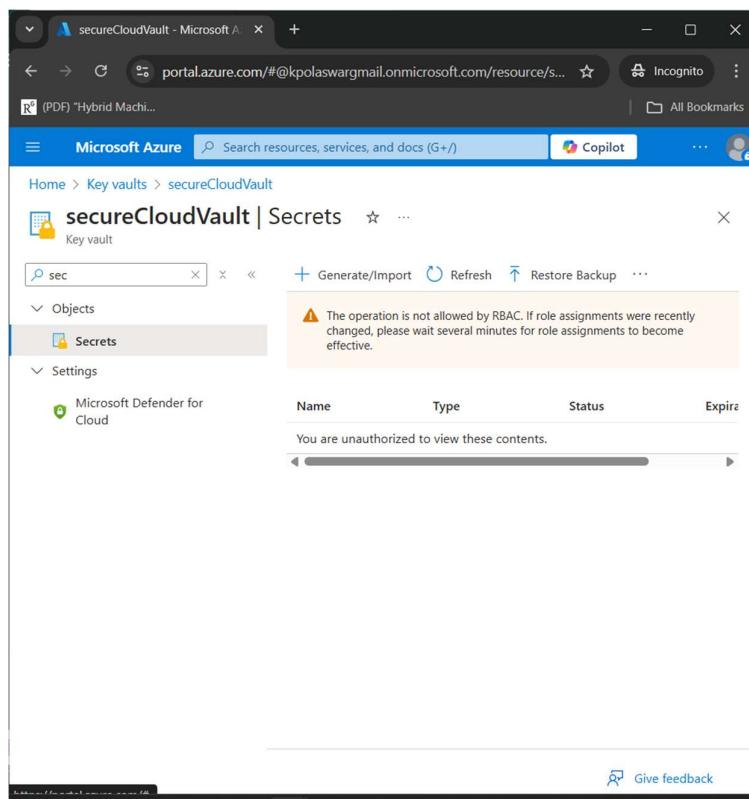
```
az role assignment create --assignee testuser@kpolaswargmail.onmicrosoft.com --role
"Reader" --scope "/subscriptions/3b5ad032-42d7-4c96-9ca9-ddc07d08b8c5"
```

---

## 9. Login as TestUser and Attempt Unauthorized Actions

### (A) Try Accessing Key Vault

Run:



## (B) Try Modifying Storage Account Settings

## X Expected Error: Authorization Failed

## (C) Try Assigning a Role

## X Expected Error: Permission Denied

**10 Monitor Logs & Security Alerts**

1. Go to Azure Portal → Search "Azure Monitor".
  2. Click Logs → Select Activity Logs.
  3. Verify Unauthorized Access Attempts by TestUser.
- 

 **Final Step: Shut Down VM to Save Credits**

```
az vm deallocate --resource-group secureCloud --name secureVM
```

---

## Experiment-03

**Deploy a multi-tier web application with a secure architecture on a cloud platform like AWS or Azure. Configure firewalls, access controls, and encryption to protect the application and its data.**

### Step 1: Create a Resource Group

A resource group is required to manage all Azure resources.

Command:

```
az group create --name SecureWebAppRG --location centralindia
```

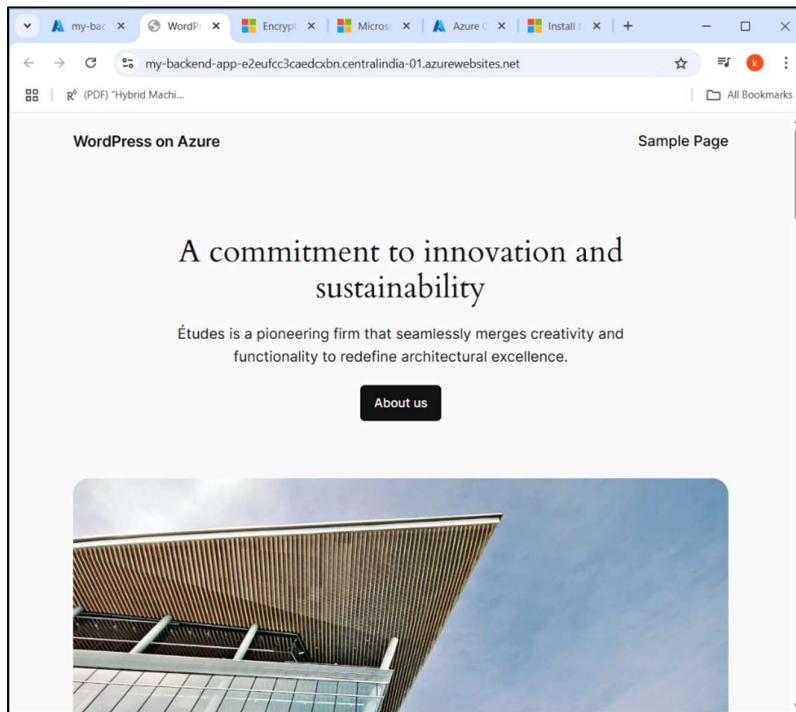
Name	Type	Location	Actions
SecureAppPlan	App Service plan	East US	...
SecureBackendApp	App Service	East US	...
SecureFrontendApp	App Service	East US	...
securemysqlserver	Azure Database for MySQL flexible server	Central India	...

### Step 2: Deploy WordPress Template (Optional for CMS-Based App)

If using a WordPress-based frontend, deploy the WordPress template.

Command:

```
az webapp create --resource-group SecureWebAppRG --name SecureFrontendApp --plan SecureAppPlan --runtime "PHP:8.0"
```



### Step 3: Create an App Service Plan

This step sets up a scalable hosting environment for the backend.

Command:

```
az appservice plan create --name SecureAppPlan --resource-group SecureWebAppRG --sku F1 --is-linux
```

```
C:\Users\KRISHNA>az appservice plan create --name SecureAppPlan --resource-group SecureWebAppRG --sku F1
{
  "elasticScaleEnabled": false,
  "extendedLocation": null,
  "freeOfferExpirationTime": null,
  "geoRegion": "East US",
  "hostingEnvironmentProfile": null,
  "hyperV": false,
  "id": "/subscriptions/3b5ad032-42d7-4c96-9ca9-ddc07d08b8c5/resourceGroups/SecureWebAppRG/providers/Microsoft.Web/serverFarms/SecureAppPlan",
  "isSpot": false,
  "isXenon": false,
  "kind": "app",
  "kubeEnvironmentProfile": null,
  "location": "eastus",
  "maximumElasticWorkerCount": 1,
  "maximumNumberOfWorkers": 0,
  "name": "SecureAppPlan",
  "numberOfSites": 0,
  "numberOfWorkers": 0,
  "perSiteScaling": false,
  "provisioningState": "Succeeded",
  "reserved": false,
  "resourceGroup": "SecureWebAppRG",
  "sku": {
    "capabilities": null,
    "capacity": 0,
    "family": "F",
    "locations": null,
    "name": "F1",
    "size": "F1",
    "skuCapacity": null
  }
}
```

### Step 4: Deploy Backend (Node.js App)

Deploy the backend API that interacts with the MySQL database.

1. Create Backend Web App:

```
az webapp create --name SecureBackendApp --resource-group SecureWebAppRG --plan SecureAppPlan --runtime "NODE:18-Its"
```

## 2. Configure Environment Variables:

```
az webapp config appsettings set --resource-group SecureWebAppRG --name SecureBackendApp --settings DB_HOST="securemysqlserver.mysql.database.azure.com" DB_USER="azureuser" DB_PASSWORD="MySecureP@ssw0rd" DB_NAME="flexibleserverdb"
```

The screenshot shows the Microsoft Azure portal interface for a 'my-backend-app' web application. The left sidebar has a tree view with 'Overview' selected. The main content area shows the following details:

- Resource group:** MultiTierApp
- Status:** Unknown
- Location:** Central India
- Subscription:** Azure for Students
- Tags:** AppProfile : Wordpress, WordPressDeploymentID : my-backend-app-afc637e588
- Properties:** Name: my-backend-app, Publishing model: Code, Runtime Stack: Node - 18-Its
- Deployment Center:** Deployment logs: Last deployment was successful on Sunday, March 23, 09:50:43 PM.
- Domains:** Default domain: my-backend-app-
- Application Insights:** Application insights status: Not configured.

## Step 5: Deploy and Configure MySQL Database

### 1. Create MySQL Flexible Server:

```
az mysql flexible-server create --resource-group SecureWebAppRG --name SecureMySQLServer --admin-user azureuser --admin-password MySecureP@ssw0rd --sku-name Standard_B1ms --location centralindia
```

### 2. Allow Public IP Access:

```
az mysql flexible-server firewall-rule create --resource-group SecureWebAppRG --server-name SecureMySQLServer --name AllowAllIPs --start-ip-address 0.0.0.0 --end-ip-address 255.255.255.255
```

The screenshot shows the Azure Database for MySQL - Single instance named 'mybackenda-afc637e588-wpdbserver'. The instance is in the 'Ready' status with a 'Burstable\_B1ms\_1 vCores, 2 GiB RAM, 32 storage, 640 IOFS' configuration. It was created on 2025-03-23 at 14:37:39 UTC.

## Step 6: Deploy Frontend (Custom HTML/CSS/JS)

### 1. Create the Frontend Web App:

```
az webapp create --resource-group SecureWebAppRG --name SecureFrontendApp --plan SecureAppPlan --runtime "PHP:8.0"
```

### 2. Deploy Static Frontend Files:

Ensure your frontend code (HTML, CSS, JS) is in a zip file named frontend.zip.

```
az webapp deploy --resource-group SecureWebAppRG --name SecureFrontendApp --src-path frontend.zip --type static --target-path /
```

Frontend Deployment Successful with backend integrated:

The screenshot shows a browser window displaying the deployed frontend application. The page contains a form with a text input field containing 'Krishna' and a 'Submit' button.

**Our Multi-Tier Web App has been deployed successfully.**

### 1. Check Backend API Status:

```
curl https://securebackendapp.azurewebsites.net/
```

### 2. Check Frontend Status:

```
Visit https://securefrontendapp.azurewebsites.net/
```