

DevOps Intern Assignment- ExactSpace

Contents:

1. Open-Source Project on Github with SQLite
2. Created an Azure Virtual Machine (VM) for Deployment
3. SSH Connection to the Azure Virtual Machine
4. Setting up Jenkins and NginX
5. Deployment Environment and Project Configuration
6. VM Monitoring Setup using Azure Monitor
7. Optimization Recommendations

1. Open-Source Project on Github with SQLite

The primary objective of this step was to identify a suitable open-source project on GitHub that uses SQLite as its database. The chosen application was a Notebook App constructed with Flask for the backend and HTML/CSS, JavaScript for the frontend, and SQLite for database.

The repository link of this project is:

<https://github.com/r-e-d-ant/Notebook-App>

2. Created an Azure Virtual Machine (VM) for Deployment

The Azure Virtual Machine was configured as follows:

Virtual Machine Details:

Name: project-vm01

Image: Ubuntu Server 22.04 LTS

Size: Standard B2s \$GiB memory

Authentication Details:

Type: SSH Public Key
Username: testuser
Key Pair: test_key
Network Configuration:
Ports Allowed:
HTTP (Port 80)
HTTPS (Port 443)
SSH (Port 22)

Clicked on Review + Create.

Finally, with the configurations validated, the VM instance project-vm01 was successfully created, and the test_key.pem file was downloaded.

The VM “project-vm01” public IP is **20.244.8.50**.

The screenshot displays the Azure portal interface for a virtual machine named 'project-vm01'. The left sidebar contains navigation options such as Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Connect, Networking, Settings, Disks, Configuration, Advisor recommendations, Properties, and Locks. The main content area is divided into several sections:

- Essentials:** Provides a quick overview of the VM's status and configuration. It shows the resource group as 'ExactSpace_Assignment', the status as 'Running', the location as 'Central India (Zone 1)', and the subscription as 'Azure for Students'. The subscription ID is '83b7fa4f-10b7-453b-b2c4-1e46127c6bd0', and the availability zone is '1'. The public IP address is '20.244.8.50'. The operating system is 'Linux (ubuntu 22.04)', the size is 'Standard B2s (2 vcpus, 4 GiB memory)', and the virtual network/subnet is 'project-vm01-vnet/default'. The DNS name is 'Not configured'.
- Properties:** A tab that provides detailed information about the VM. It lists the computer name as 'project-vm01', the operating system as 'Linux (ubuntu 22.04)', the image publisher as 'canonical', the image offer as '0001-com-ubuntu-server-jammy', the image plan as '22_04-lts-gen2', the VM generation as 'V2', the VM architecture as 'x64', the agent status as 'Ready', the agent version as '2.10.0.3', the host group as 'None', and the host as '-'. The public IP address is '20.244.8.50 (Network interface project-vm01183_x1)', the private IP address is '10.0.0.4', and the virtual network/subnet is 'project-vm01-vnet/default'.
- Networking:** A section that provides details about the network configuration, including the public IP address, private IP address, and virtual network/subnet.
- Size:** A section that provides details about the VM's size, including the size, vCPUs, and RAM.

Figure: project-vm01 details

3. SSH Connection to the Azure Virtual Machine

After setting up the Azure VM, the next step was to establish a secure connection to it. For this purpose, the SSH protocol was employed. Made sure that the key had the read permission.

```
$chmod 400 /home/ayush/Downloads/test_key.pem
```

Then,

```
$ssh -i /home/ayush/Downloads/test_key.pem testuser@20.244.8.50
```

This command opens a secure shell access to project-vm01 using the test_key.pem for authentication.

```
ayush@ayush:~$ chmod 400 /home/ayush/Downloads/test_key.pem
ayush@ayush:~$ ssh -i /home/ayush/Downloads/test_key.pem testuser@20.244.8.50
The authenticity of host '20.244.8.50 (20.244.8.50)' can't be established.
ED25519 key fingerprint is SHA256:yyyAgrwmVjk/r92Hm0bk1YobSs545xI8EFiksFYL2Io.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.244.8.50' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1014-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Oct  9 09:16:18 UTC 2023

System load:  0.2060546875      Processes:            118
Usage of /:   5.1% of 28.89GB   Users logged in:     0
Memory usage: 7%               IPv4 address for eth0: 10.0.0.4
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

4. Setting up Jenkins and NginX

4.1 Jenkins Installation and Configuration

- Updated system packages with
\$sudo apt update && sudo apt upgrade.
- Installed Java, a prerequisite for Jenkins, using
\$sudo apt install openjdk-11-jre
- Run these commands:

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

```
testuser@project1:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
> /usr/share/keyrings/jenkins-keyring.asc > /dev/null
testuser@project1:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
> https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
> /etc/apt/sources.list.d/jenkins.list > /dev/null
testuser@project1:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2890 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1121 kB]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:9 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:10 https://pkg.jenkins.io/debian-stable binary/ Packages [25.6 kB]
Fetched 4376 kB in 1s (5029 kB/s)
Reading package lists... Done
testuser@project1:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 89.1 MB of archives.
After this operation, 90.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 net-tools amd64 1.60+git20180626.aebd88e-1ubuntu1 [196 kB]
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.414.2 [88.9 MB]
Fetched 89.1 MB in 5s (16.9 MB/s)
Selecting previously unselected package net-tools.
(Reading database ... 85299 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.414.2_all.deb ...
Unpacking jenkins (2.414.2) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up jenkins (2.414.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...
testuser@project1:~$
```

- starting and checking status of Jenkins:

`$sudo systemctl start jenkins`

`$sudo systemctl status jenkins`

```
testuser@project-vm01:~$ sudo systemctl start jenkins
testuser@project-vm01:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-10-09 09:22:32 UTC; 13h ago
     Main PID: 6074 (java)
       Tasks: 53 (limit: 4618)
      Memory: 1.7G
         CPU: 5min 49.593s
    CGroup: /system.slice/jenkins.service
            └─ 6074 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
               75736 /bin/bash /tmp/jenkins6822661681093423910.sh
               75754 python3 run.py
               75755 /home/testuser/githubProject2/virtualenv2/bin/python3 /home/testuser/githubProject2/Notebook-App/run.py

Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.Build$BuildExecution.build(Build.java:199)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.Build$BuildExecution.doRun(Build.java:164)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.AbstractBuild$AbstractBuildExecution.run(AbstractBuild.java:526)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.Run.execute(Run.java:1900)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.ResourceController.execute(ResourceController.java:101)
Oct 09 18:06:40 project-vm01 jenkins[6074]: at hudson.model.Executor.run(Executor.java:442)
Oct 09 21:20:29 project-vm01 jenkins[6074]: 2023-10-09 21:20:29.025+0000 [id=727] WARNING hudson.security.csrf.CrumbFilter#doFilter
Oct 09 21:20:29 project-vm01 jenkins[6074]: 2023-10-09 21:20:29.026+0000 [id=727] WARNING hudson.security.csrf.CrumbFilter#doFilter
lines 1-23/23 (END)
```

- Created a port rule in the Azure VM, having port 8080 with name AllowJenkins8080.
- Accessed Jenkins through a web browser using the 20.244.8.50:8080.
- Followed on-screen instructions to complete the Jenkins setup, which included unlocking Jenkins by giving administrative password from `/var/lib/jenkins/secrets/initialadminpassword`, then signing in to the Jenkins

4.2 NginX Installation and Configuration

- Installed NGINX using
`$sudo apt install nginx.`
- Configuration as a Reverse Proxy for Jenkins:

Edited the default NGINX configuration (`/etc/nginx/sites-available/default`) to act as a reverse proxy, so that Jenkins isn't directly exposed on port 8080. The configuration included:

```
server {
    listen 80;
    server_name 20.244.8.50;
    location / {
        proxy_pass http://localhost:8080;
```

```

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

```

- Restarted NGINX to apply the changes using

```
$sudo systemctl restart nginx.
```

5. Deployment Environment and Project Configuration:

Project Directory Creation:

- First created a port rule in the Azure VM, having port 8000.
- Created a directory: \$sudo mkdir /home/testuser/githubProject2
- Navigated to the directory: \$cd /home/testuser/githubProject2
- Initialized a Python virtual environment: \$sudo python3 -m venv virtualenv2
- Assigned directory ownership to 'jenkins' user: \$sudo chown -R jenkins /home/testuser/githubProject2

Jenkins Configuration:

- Accessed Jenkins dashboard and initiated a "Freestyle Project" titled GithubProject.
- Within the build steps, added 'execute shell', then Incorporated the following commands in the Jenkins shell execution:

```
#!/bin/bash
```

```
# Exit the script as soon as a command fails
```

```
set -ex
```

```
export BUILD_ID=dontKillMe
```

```
# Navigate to the project directory
```

```
cd /home/testuser/githubProject2
```

```
if [ ! -d "Notebook-App" ]; then
```

```
    git clone https://github.com/r-e-d-ant/Notebook-App.git
```

fi

```
cd Notebook-App
```

```
# Pull the latest code from the repository  
git pull origin main
```

```
# Activate the virtual environment  
source /home/testuser/githubProject2/virtualenv2/bin/activate
```

```
# Install required modules  
pip3 install -r requirements.txt
```

```
# Kill the previous Gunicorn process if it exists and start the new one  
pkill -f 'run' || true
```

```
#run the program  
python3 run.py
```

Then Clicked "Save" and "Build Now" to trigger the deployment.

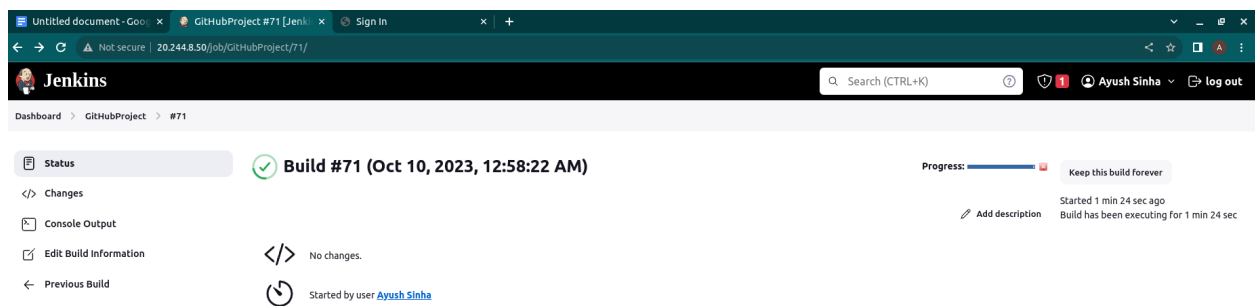


Figure: deployment has been triggered

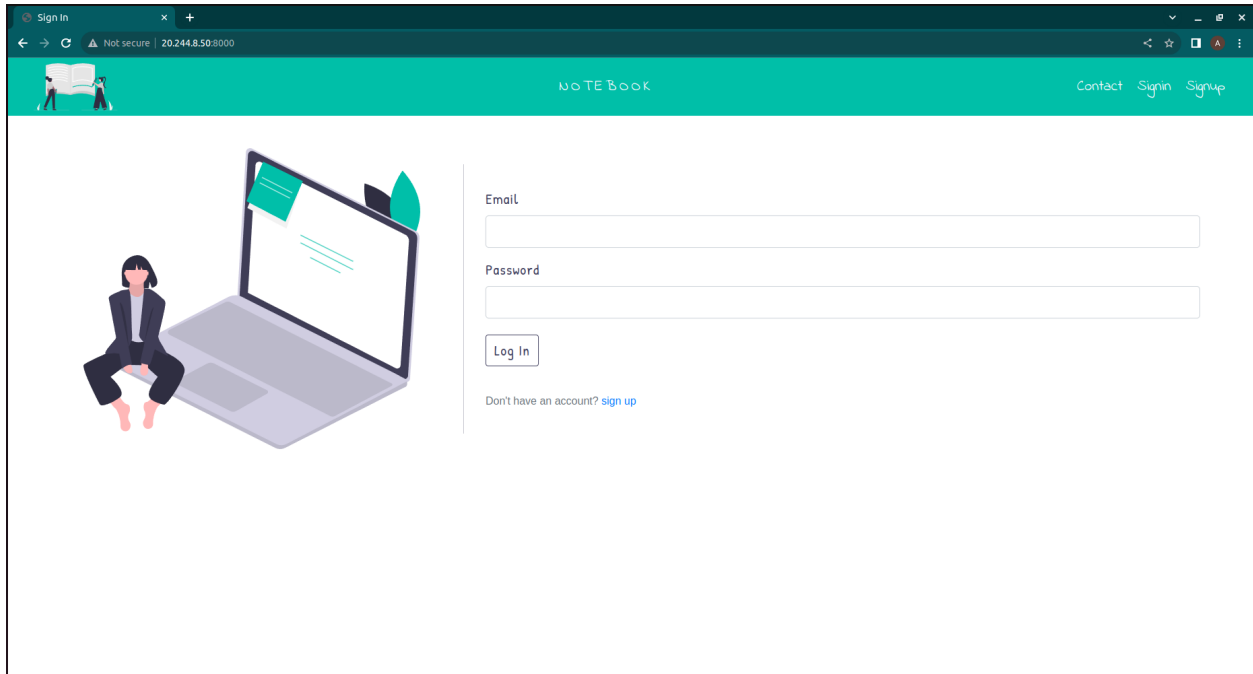


Figure: Live Project running on **20.244.8.50:8000**

6. VM Monitoring Setup using Azure Monitor

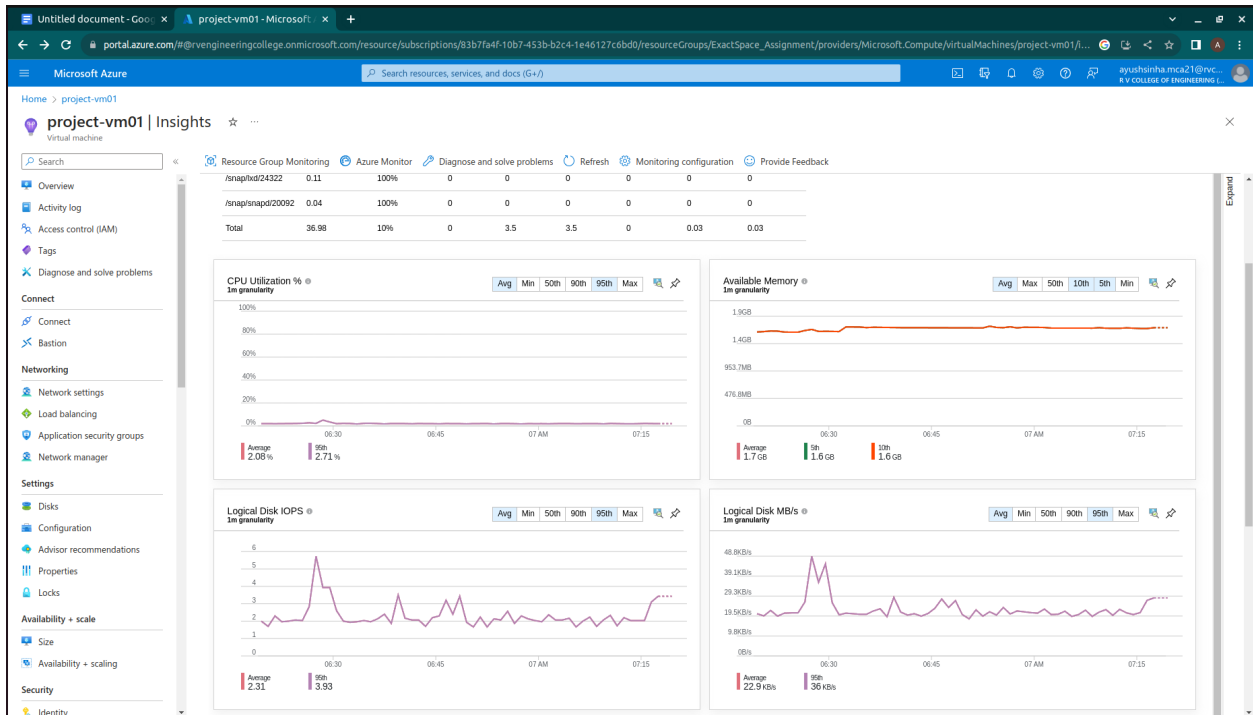
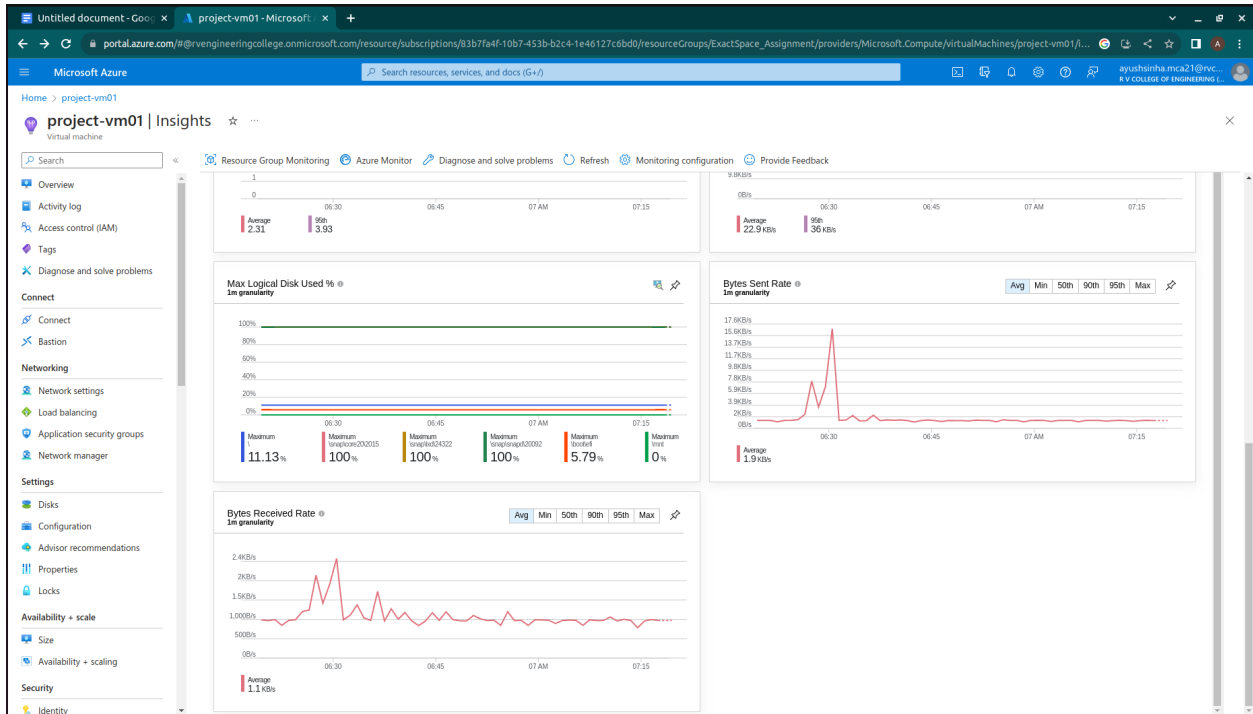
Azure Monitor was employed to get full-stack monitoring and advanced analytics for the VM.

Navigated to the “project-vm01” VM, and enabled the monitoring by creating a Azure Monitor resource, after enabling the insights within the Monitoring section

Once the Azure Monitor resource was deployed and associated with the VM, monitoring started.

The following metrics were actively monitored:

CPU Utilization, Available Memory, Logical Disk MB/s, Max Logical Disk Used%, Bytes Sent Rate, Bytes Received Rate.



7. Optimization Recommendations

- **Database Optimization:** Consider transitioning from SQLite to a robust system like PostgreSQL or Azure SQL for enhanced scalability. Also SQLite supports only one writer at a time. If you have a high write load, this can be a bottleneck.
- **Automated Deployment with Jenkins:** Configure Jenkins to periodically check the GitHub repository. If updates are detected, initiate an automatic pull and deployment process, ensuring the application always runs the latest version.
- **Implement Autoscaling:** Depending on the load, the infrastructure could automatically scale up or down. This not only optimizes costs but also ensures consistent performance.
- **Alerting with Azure Monitor:** With Azure Monitor already in place, set up targeted alerts. Receive real-time notifications for any anomalies or thresholds breached.