DevOps Intern Assignment- ExactSpace

Contents:

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.**

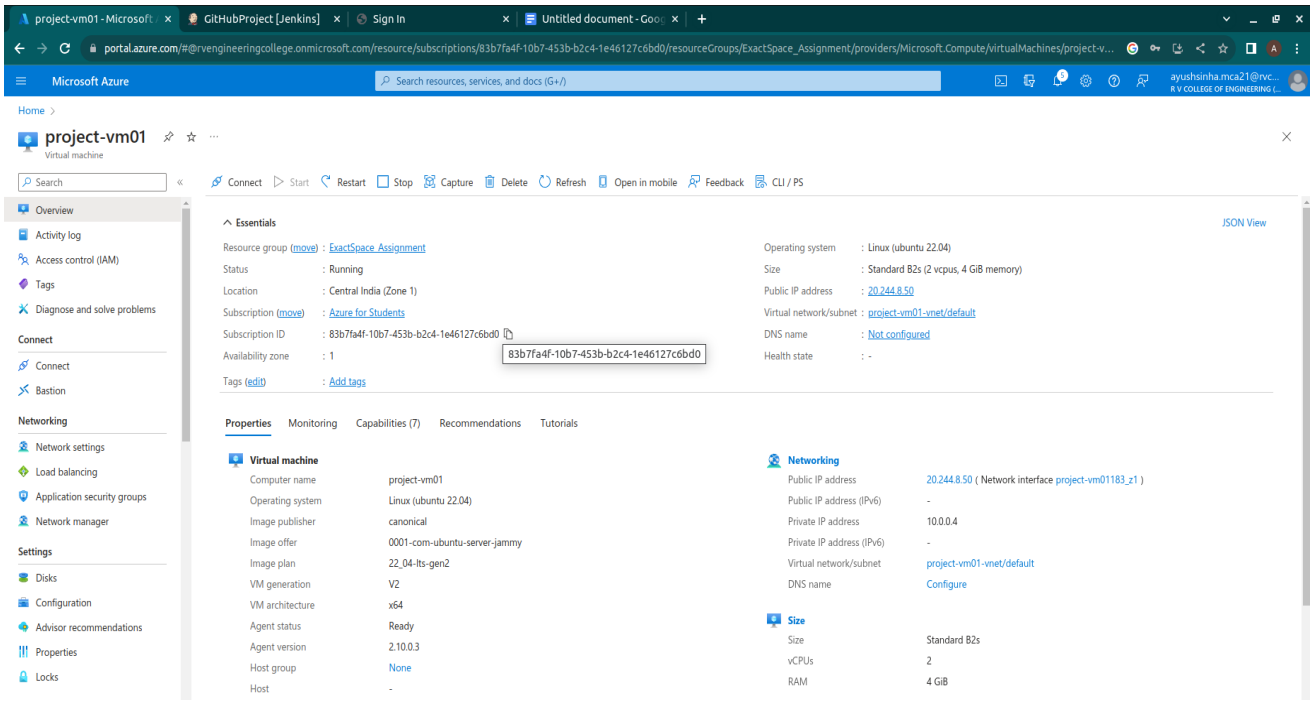


Figure: project-vm01 details

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.

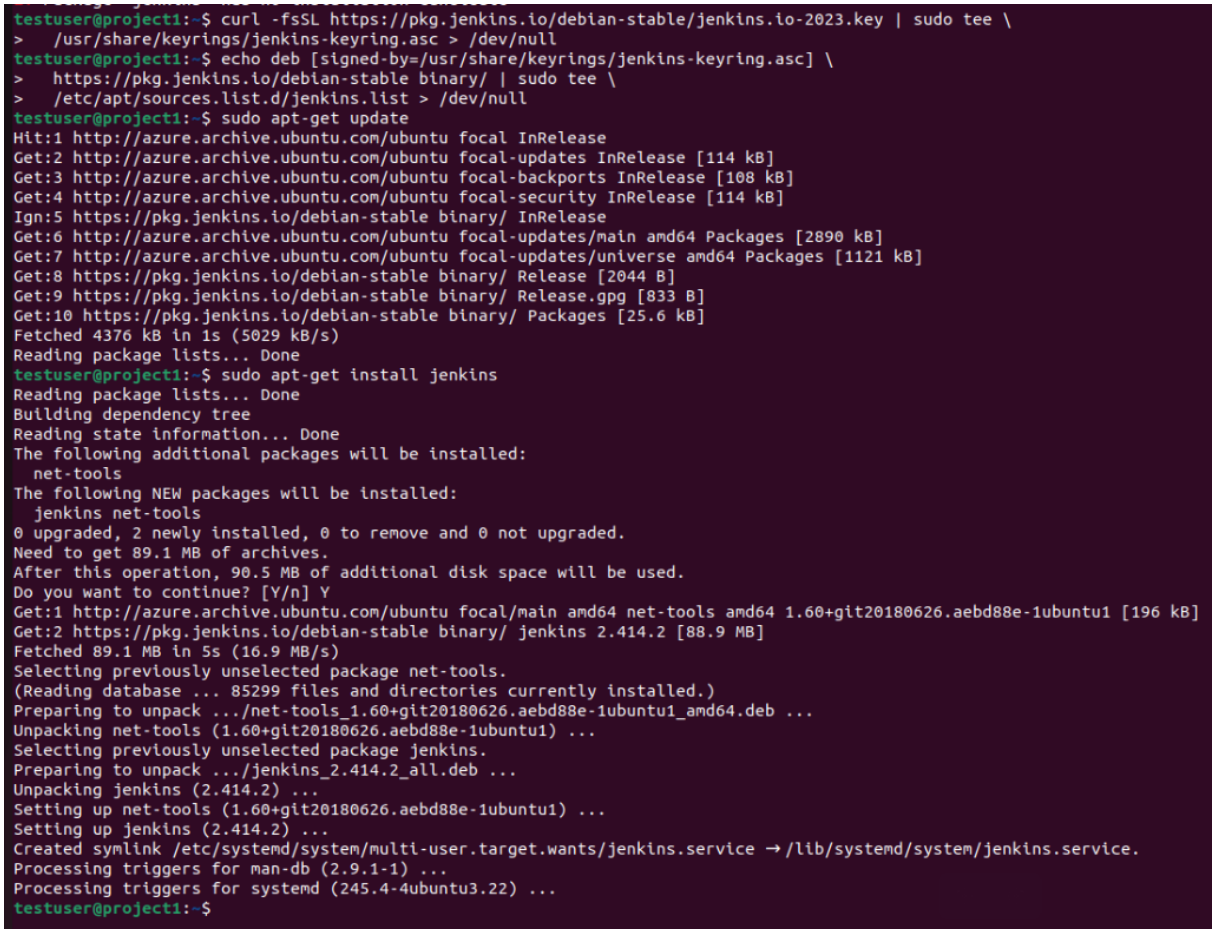
```
ayush@ayush:~$ chmod 400 /home/ayush/Downloads/test_key.pem
ayush@ayush:~$ ssh -i /home/ayush/Downloads/test_key.pem testuser@20.244.8.50
The authenticity of host '20.244.8.50 (20.244.8.50)' can't be established.
ED25519 key fingerprint is SHA256:yyyAgrwmVjk/r92Hm0bk1YobSsS45xI8EFiksfYL2Io.
This 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

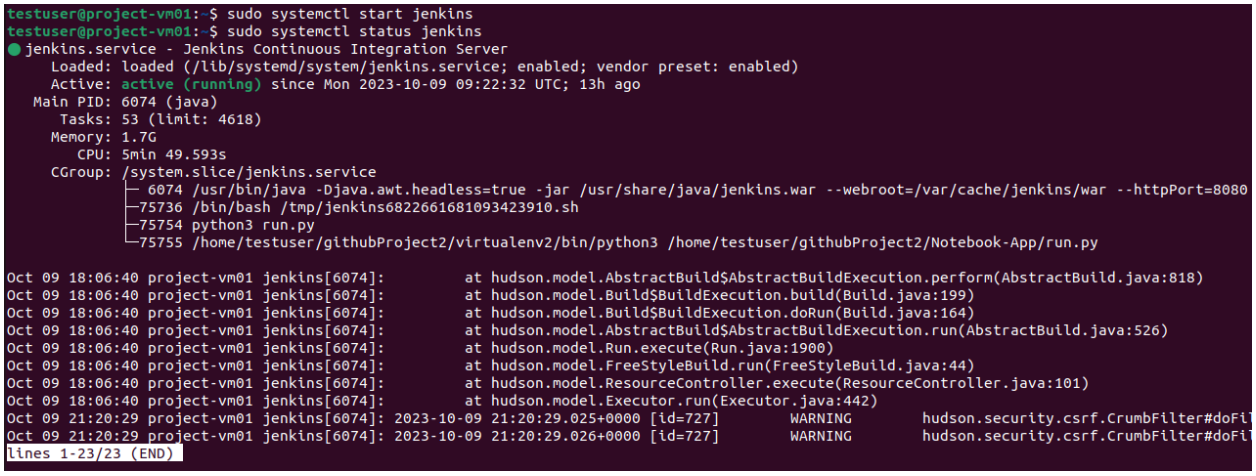
- Updated system packages with
  $sudo apt update && sudo apt upgrade.
- Installed Java, a prerequisite for Jenkins, using
  $sudo apt install openjdk-11-jre
- Run these commands:

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



- starting and checking status of Jenkins:

  $sudo systemctl start jenkins
  $sudo systemctl status jenkins



- 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.

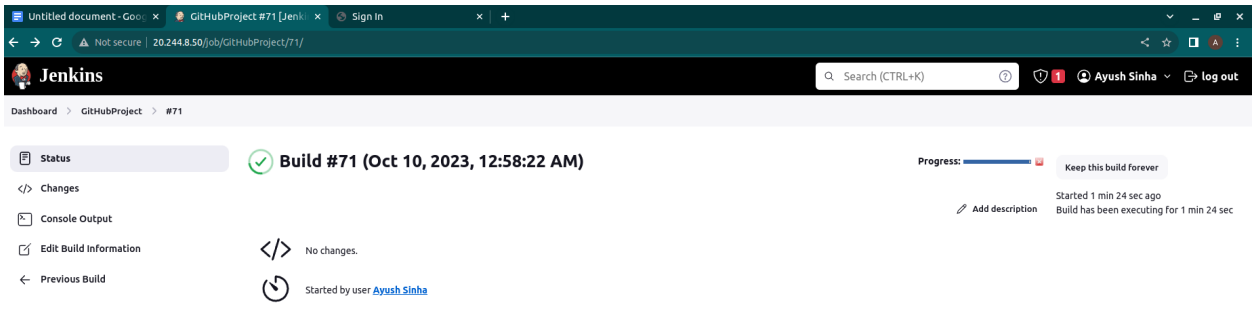


Figure: deployment has been triggered

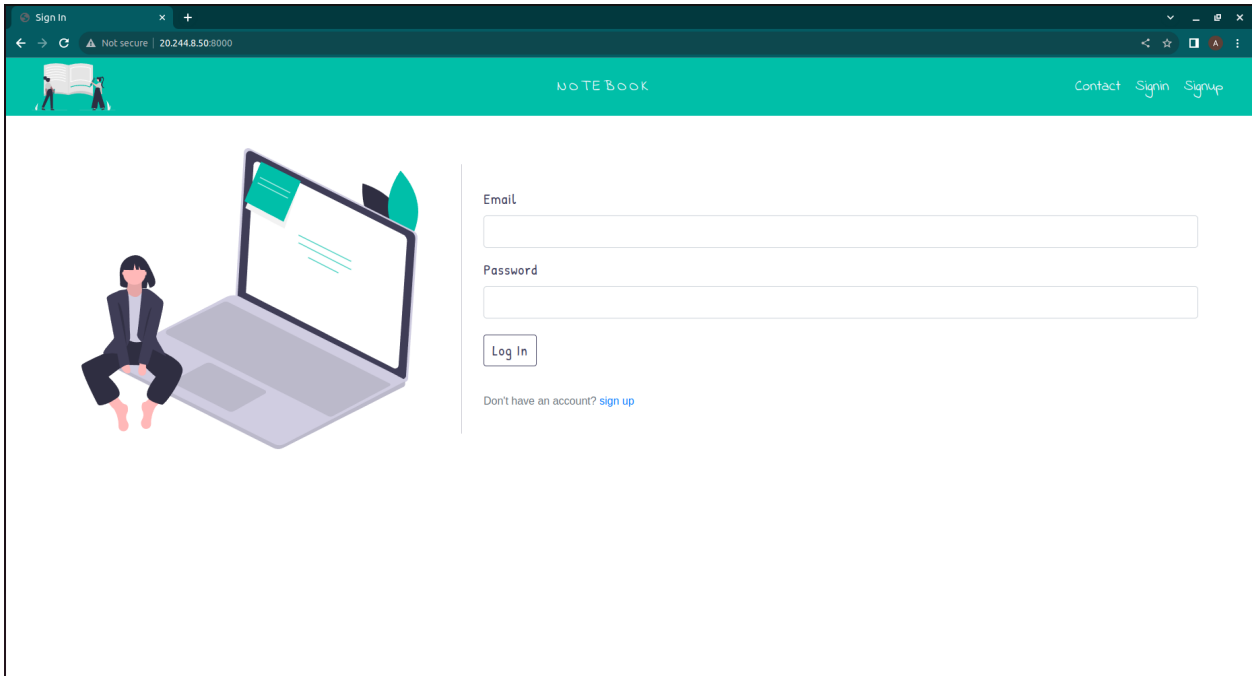


Figure: Live Project running on **20.244.8.50:8000**

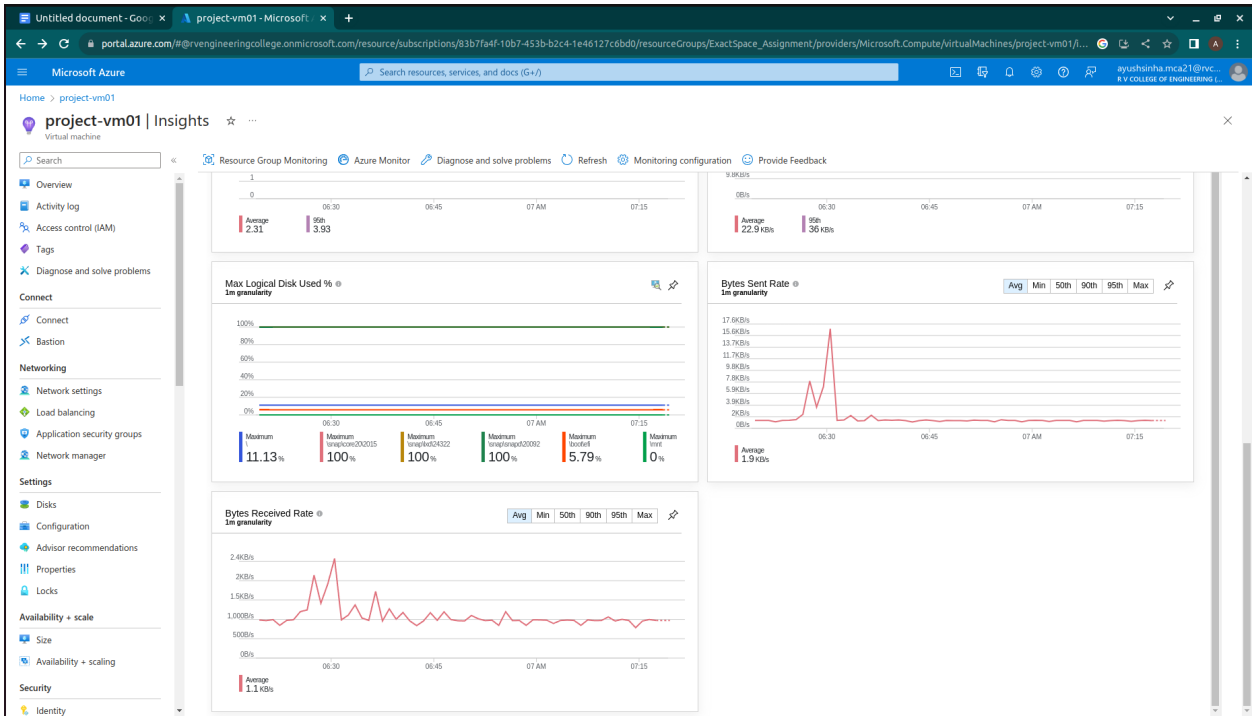
6. VM Monitoring Setup using Azure Monitor

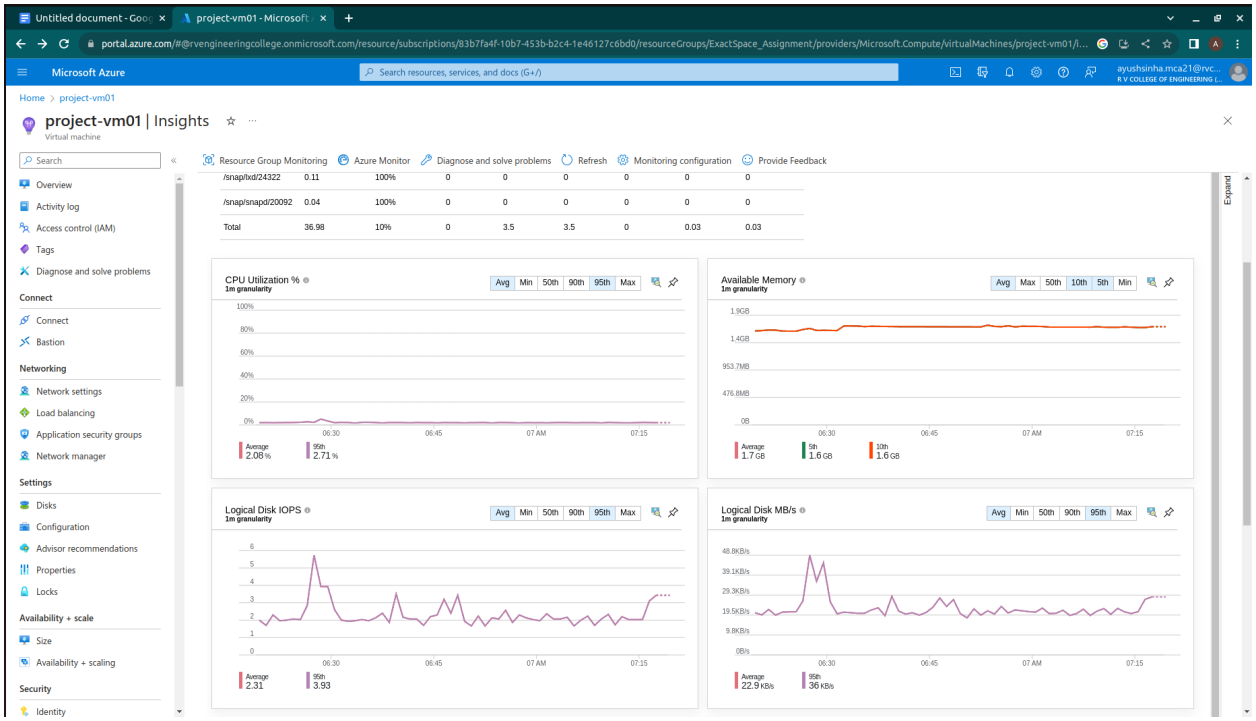
Azure Monitor was employed to get full-stack monitoring and advanced analytics for the VM.

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.


Microsoft Azure
Home > project-vm01
project-vm01 | Insights
Virtual machine
Resource Group Monitoring
Azure Monitor
Diagnose and solve problems
Refresh
Monitoring configuration
Provide Feedback
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Connect
Connect
Bastion
Networking
Network settings
Load balancing
Application security groups
Network manager
Settings
Disks
Configuration
Advisor recommendations
Properties
Locks
Availability + scale
Size
Availability + scaling
Security
Identity
Max Logical Disk Used %
1m granularity
Bytes Sent Rate
1m granularity
Avg Min 50th 90th 95th Max
Bytes Received Rate
1m granularity
Avg Min 50th 90th 95th Max


Microsoft Azure
Home > project-vm01
project-vm01 | Insights
Virtual machine
Resource Group Monitoring
Azure Monitor
Diagnose and solve problems
Refresh
Monitoring configuration
Provide Feedback
/snap/lxd/24322 0.11 100% 0 0 0 0 0 0
/snap/snapd/20092 0.04 100% 0 0 0 0 0 0
Total 36.98 10% 0 3.5 3.5 0 0.03 0.03
Expand
CPU Utilization %
1m granularity
Avg Min 50th 90th 95th Max
Available Memory
1m granularity
Avg Max 50th 10th 5th Min
Logical Disk IOPS
1m granularity
Avg Min 50th 90th 95th Max
Logical Disk MB/s
1m granularity
Avg Min 50th 90th 95th Max

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.