

Name : Hamza Jamshaid

Project title:

Cloud Monitoring and Log Analysis

Introduction:

The Internee.pk system is a web application used by interns and companies to manage internship processes. To ensure its performance, reliability, and security, we implemented **cloud monitoring and log analysis** using **Azure Monitor and Application Insights**. This allows real-time tracking of system metrics, error detection, and proactive alerting for system failures.

Objective:

- Monitor the Internee.pk backend for **performance issues** such as CPU usage, memory usage, and response time.
- Track **system reliability** by analyzing failed requests and downtime.
- Identify **errors and security anomalies** using log analysis.
- Configure **alerts** to notify the team of high CPU, failed requests, or system outages.

Project Structure:

internee-cloud-monitoring/

```
|── backend/  
|   └── server.js  
|── monitoring/  
|   ├── metrics.md  
|   ├── alerts.md  
|   └── kql-queries.md  
|── screenshots/  
└── README.md
```

[GITHUB LINK HERE..](#)

Project Procedure:

Below is a high-level breakdown of all procedures performed in the project. Each step mentions only actions and file names.

1. Create Backend first :

- Create folder and initialize it.
- Install express that helps in building process

```
hamza@hamza-HP-EliteBook-840-G4:~$ cd internee-monitoring-backend
hamza@hamza-HP-EliteBook-840-G4:~/interner-monitoring-backend$ npm init -y
Wrote to /home/hamza/interner-monitoring-backend/package.json:

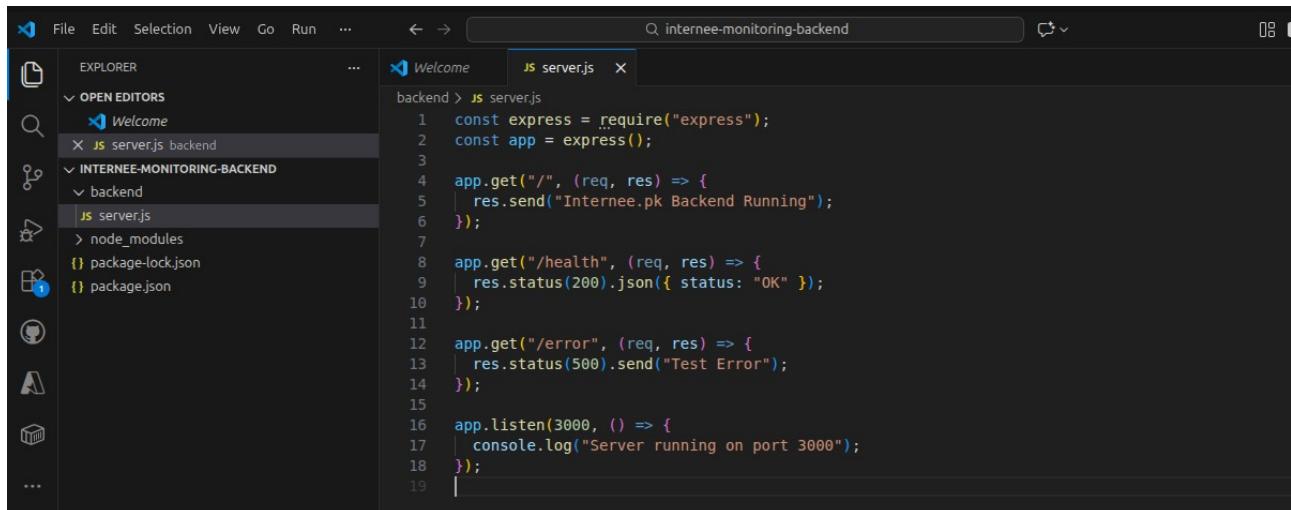
{
  "name": "interner-monitoring-backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

hamza@hamza-HP-EliteBook-840-G4:~/interner-monitoring-backend$ npm install express
added 65 packages, and audited 66 packages in 13s

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

found 0 vulnerabilities
hamza@hamza-HP-EliteBook-840-G4:~/interner-monitoring-backend$
```

Create Server.js:



The screenshot shows the Visual Studio Code interface. The left sidebar displays the file structure under 'OPEN EDITORS' and 'INTERNEE-MONITORING-BACKEND'. The 'server.js' file is open in the main editor area. The code is as follows:

```
backend > JS server.js
1  const express = require("express");
2  const app = express();
3
4  app.get("/", (req, res) => {
5    res.send("Internee.pk Backend Running");
6  );
7
8  app.get("/health", (req, res) => {
9    res.status(200).json({ status: "OK" });
10 );
11
12 app.get("/error", (req, res) => {
13   res.status(500).send("Test Error");
14 );
15
16 app.listen(3000, () => {
17   console.log("Server running on port 3000");
18 });
19
```

2. Deploy Backend to Azure App Service:

- Create App Services with Runtime “Node 22 LTS”.
- Region “Canada Central” and Name “Moniter”.

The screenshot shows the Microsoft Azure Overview page for a deployment named "Microsoft.Web-WebApp-Portal-78f94668-a040". The status bar indicates "Your deployment is complete". Deployment details show the deployment name, start time (1/13/2026, 12:34:04 AM), subscription (Azure subscription 1), correlation ID (deda4d04-c380-472a-b522-2fcf8d37...), and resource group (NetworkWatcherRG). The table lists five resources: Moniter/ftp, Moniter/scm, Moniter, and two instances of Moniter under Application Insights, all in OK status. A sidebar on the right provides links to Cost management, Microsoft Defender for Cloud, and Free Microsoft tutorials.

3. Connect GitHub:

- Attach GITHUB to link with Azure and App services.
- Attach related Repo.

The screenshot shows the Azure App Services Deployment Center for the "Moniter" web app. The left sidebar lists "App Services" and "Moniter". The main pane displays the "Deployment Center" settings, showing GitHub integration with organization "Hamza-jamshaid" and repository "internee-monitoring-backend" on the "main" branch. Other tabs include "Settings", "Containers (new)", "Logs", and "FTPS Credentials". A message at the top indicates a new version of the experience is available.

4. Enable Azure Monitor:

- Turn on Application insight
- Create new log Analytics Workspace

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', 'Search resources, services, and docs (G+)', 'Copilot', and user information 'hamzajamshaid339@gmail.com... DEFAULT DIRECTORY (HAMZAJA...)'. The main page title is 'Moniter | Application Insights' under 'Web App'. On the left, a sidebar lists 'Activity log', 'App Service plan', 'Monitoring' (which includes 'Logs' and 'Application Insights'), and 'Alerts'. The 'Application Insights' section is currently selected. A central form is being used to create a new resource, with fields for 'New resource name' (set to 'Monitor202601121948'), 'Location' (set to 'Canada Central'), and 'Log Analytics Workspace' (set to 'DefaultWorkspace-6e6ebf69-df15-4067-b3c7-44d9db22699e-CCAN...'). Below these fields, there's a note about selecting an existing resource and a dropdown for 'Azure subscription 1'. A table below shows 'Top 5 relevant resources - Relevance is determined by resource group, location, or in alphabetical order'. At the bottom of the form is an 'Apply' button. To the right, a 'Notifications' sidebar displays several recent events: 'Apply Changes' (Changes are applied, a few seconds ago), 'Save code settings' (Successfully setup GitHub Action build and deployment pipeline, 8 minutes ago), 'Deployment succeeded' (Deployment 'Microsoft.Web-WebApp-Portal-78f94668-a040' to resource group 'NetworkWatcherRG' was successful, 16 minutes ago), and '\$168.14 credit remaining' (Subscription 'Azure subscription 1' has a remaining credit of \$168.14). There are also 'Go to resource' and 'Go to resource group' buttons.

5: Monitor Metrics (CPU, Response Time, Errors):

Add metrics:

- CPU Percentage
- Requests
- Average Response Time
- Failed Requests

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', 'Search resources, services, and docs (G+)', 'Copilot', and user information 'hamzajamshaid339@gmail.com... DEFAULT DIRECTORY (HAMZAJA...)'. The main page title is 'App Services' under 'Moniter'. On the left, a sidebar lists 'Create', 'Development Tools', 'API', 'Monitoring' (which includes 'Metrics', 'Logs', 'Health check', 'Application Insights', 'Diagnostic settings', 'App Service logs', 'Automation', and 'Support + troubleshooting'), and 'Name ↑'. The 'Metrics' section is currently selected. A central chart displays three metrics over time: 'Average Response Time (deprecated) (Avg), Moniter | 10.02sec' (blue line), 'CPU Time (Sum), Moniter | 15.49sec' (pink line), and 'Requests (Sum), Moniter | 11' (green line). The chart shows a sharp peak in CPU time around 1:11 PM on January 12, 2024. The x-axis shows the date from '1/12 1:11 PM' to '1/13 1:11 PM' and the time from '6 PM' to '6 AM'. The y-axis shows time intervals from '1.00sec' to '15.00sec'. At the bottom of the chart area, there are buttons for 'Give Feedback' and 'Add or remove favorites by pressing Ctrl+Shift+F'.

6. Create Alerts (System Failure):

Alert 1: High CPU

- Create Alert Rule
- Metric: CPU Percentage
- Condition: > 80% for 5 minutes
- Action: Email

The screenshot shows the Microsoft Azure Monitor interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', 'Search resources, services, and docs (G+)', 'Copilot', and user information 'hamzajamshaid339@gmail.com... DEFAULT DIRECTORY (HAMZAJA...)'. The main title is 'Monitor | Alerts' for a 'Web App'. On the left, there's a sidebar with 'Tags', 'Diagnose and solve problems', 'Microsoft Defender for Cloud', 'Events (preview)', 'Log stream', 'Resource visualizer', 'Deployment' (with 'Deployment slots' and 'Deployment Center'), 'Settings', 'Performance', and 'App Service plan'. Below the sidebar, a message says 'Showing 1 - 1 of 1. Display count: auto'. The main area displays alert statistics: 'Total alerts' (1), 'Critical' (0), 'Error' (0), 'Warning' (0), 'Informational' (0), and 'Verbose' (0). It also includes columns for 'Name', 'Severity', 'Affected resource', 'Alert condition', and 'User response'. A large exclamation mark icon is at the bottom right.

Alert 2: Errors

- Metric: Failed Requests
- Threshold: > 2 in 5 minutes
- Action: Email

This screenshot is similar to the previous one but includes a success message in a toast notification: 'Alert rule created' with a green checkmark, followed by the text 'Alert rule Alert2 successfully created. It might take a few minutes for changes to be shown.' The interface and alert statistics are identical to the first screenshot.

Results:

- Successfully deployed **Node.js backend** on **Azure App Service**.
- Enabled **Azure Monitor and Application Insights**, which tracked:
 - CPU and memory usage
 - Request count and response times
 - Failed requests and 500 errors
- Configured **alerts** for high CPU and failed requests.
- Log queries provided insights into performance bottlenecks and error patterns.
- Generated a professional **GitHub repository** with metrics, alerts, and screenshots for documentation.

Conclusion:

The project demonstrates the importance of **cloud-based monitoring** for web applications. By implementing Azure Monitor and log analysis, the Internee.pk system became more **reliable and observable**, enabling proactive detection of issues. This approach ensures **better performance, faster troubleshooting**, and prepares the system for **scalable production deployment**.