{mthree}

STRESS TESTING AND MONITORING WITH DEVOPS TOOLS THROUGH AI INTEGRATION

{mthree}

MEET OUR TEAM

Shreya Barnwal

Prachi Balla

Lalam Divya Sri

Sireesha Allampati

Hafeeza Bhanu

UNDER THE GUIDANCE OF :

Mr. Rajan Chettri

{mthree}

INDEX

{mthree}

INTRODUCTION

- Organizations face challenges in ensuring system reliability and performance under high loads.
- Current methods of stress testing and monitoring often lack integration, scalability, and intelligent feedback mechanisms.
- This project aims to address these gaps by creating an automated and comprehensive stress testing and monitoring solution that utilizes virtual machines, Prometheus, Grafana, and Python-based stress tests. The solution will incorporate CI/CD pipelines, configuration management with Ansible, containerization with Docker, deployment using Kubernetes, and advanced log analysis with AI-driven tools, coupled with real-time notifications for proactive incident management.

CONFIGURATION MANAGEMENT WITH ANSIBLE

1. **Ansible playbooks:**
   - Playbooks automate the installation of required software, ensuring each VM has the correct tools and configurations.
2. **To run the Ansible playbook:**
   - ansible-playbook -i myhosts myplaybook.yml

Monitoring_Server

Prometheus
Alert Manager
Grafana

Stress_Testing_Node

Node Exporter
MySQL Exporter
Jenkins

Database_Server

MySQL Sever

{mthree}

PYTHON SCRIPT OVERVIEW

- The Python script provides a menu-driven interface for conducting various stress tests on system resources, including memory, disk, CPU, network, and MySQL.

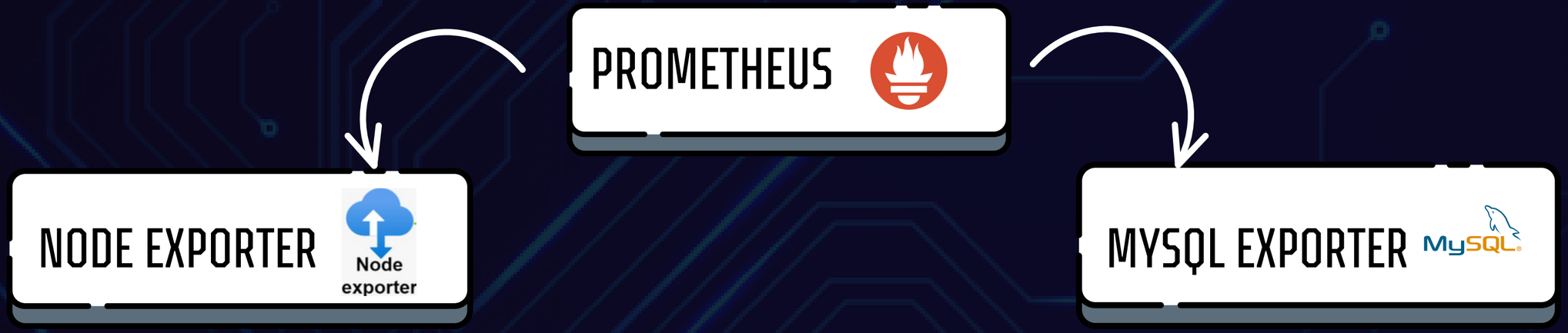
- The script uses stress-ng to conduct stress tests on CPU, memory, network, and disk resources. This allows for targeted testing of system performance under simulated heavy load conditions.

1. Memory
2. Disk
3. Network
4. CPU
5. MYSQL
6. Exit

{mthree}

LOGGING AND MONITORING THE METRICS

We configured Prometheus to scrape metrics from the Node Exporter and MySQLD Exporter, enabling comprehensive monitoring. The Grafana dashboards visualize these metrics in real-time, providing insights into system performance and health.

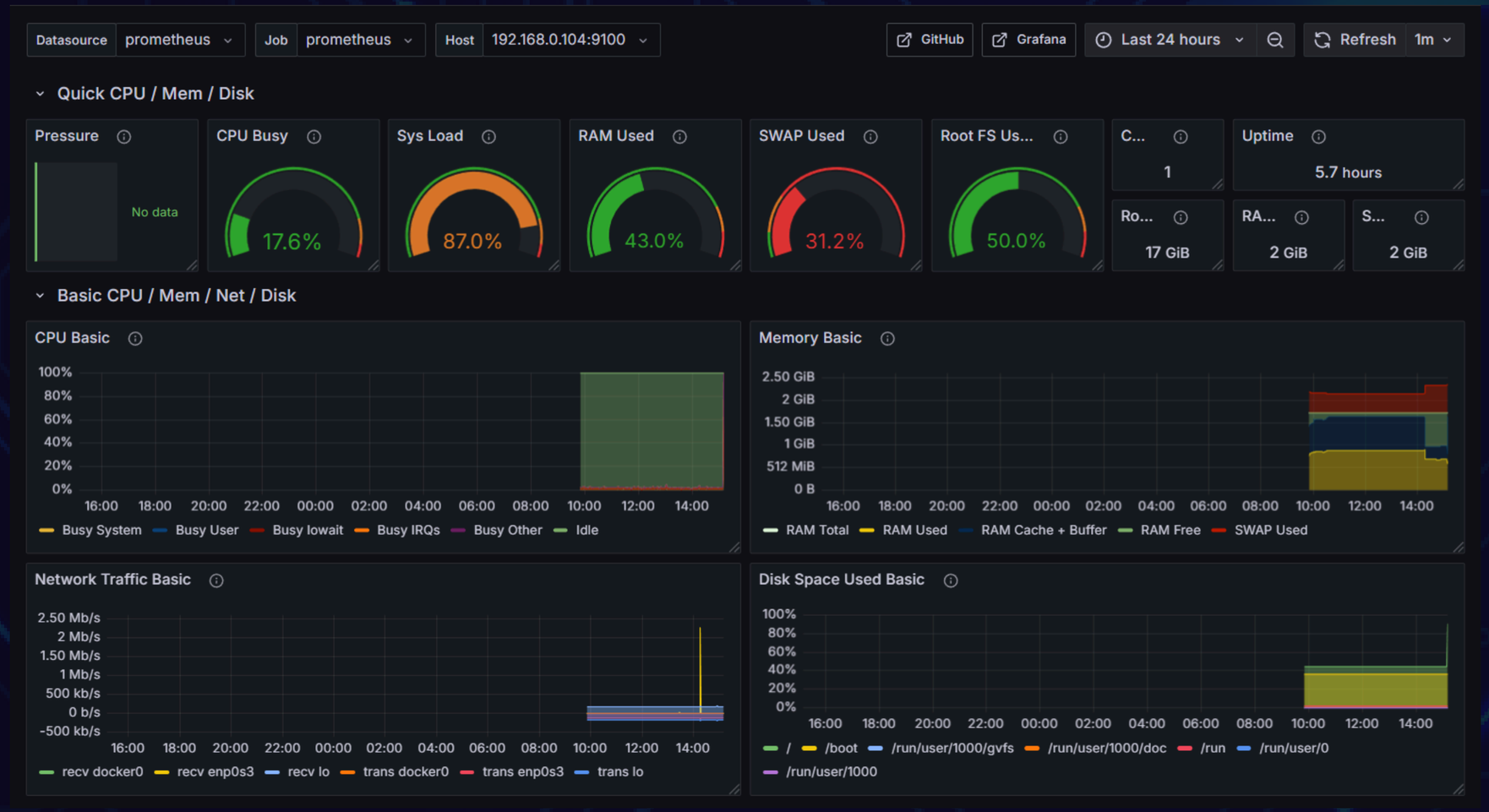


Using Node Exporter Full (**Dashboard ID: 1860**) to monitor CPU, disk, memory, and network usage metrics.

Using MySQL Overview (**Dashboard ID: 7362**) to monitor MySQL metrics that is Queries Per Second.

{mthree}

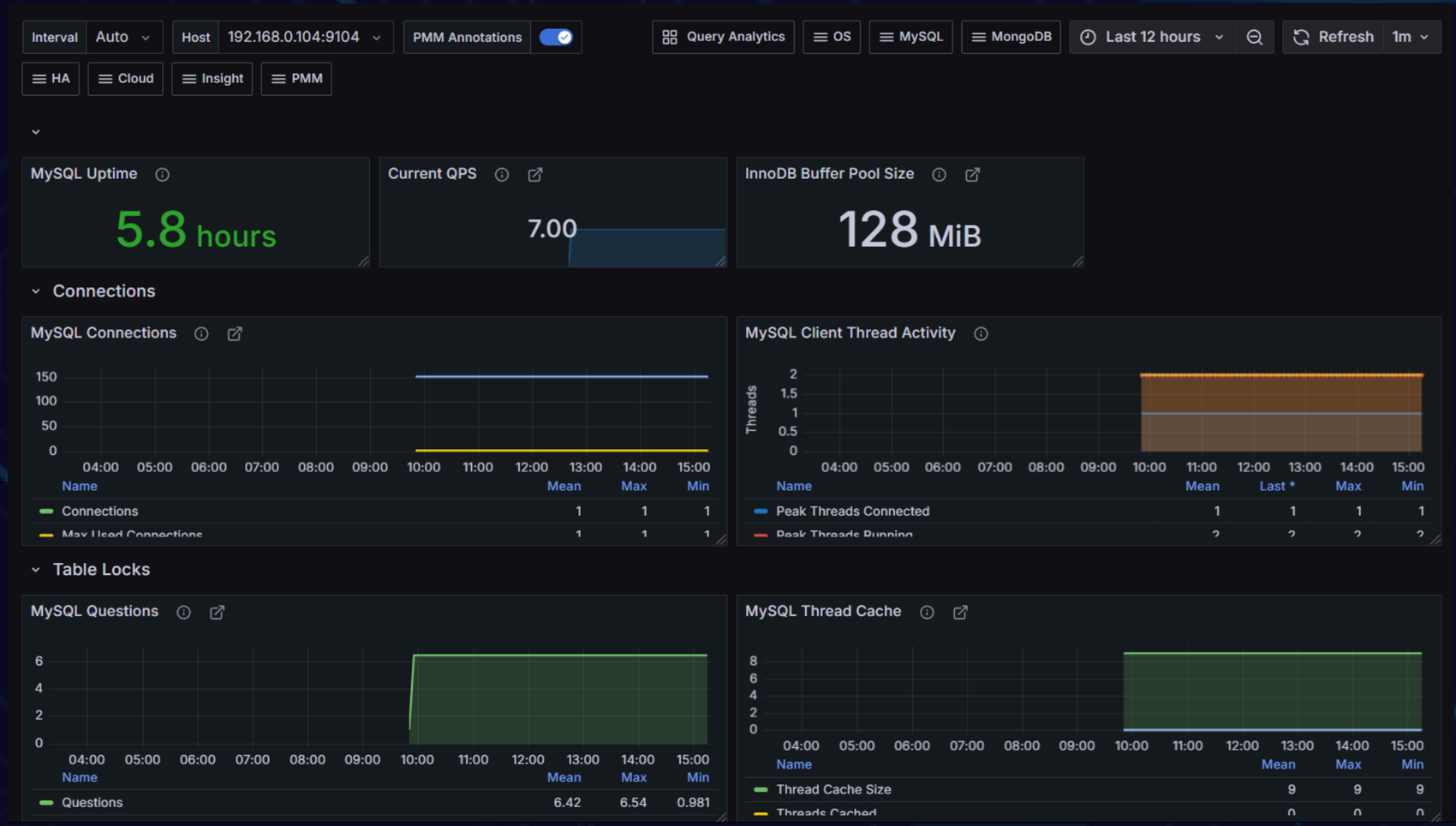
GRAFANA DASHBOARD FOR VISUALIZING METRICS OF FIRST FOUR STRESS TESTS (NODE EXPORTER)



The CPU, memory, and disk usage have all exceeded the 80% threshold, while network usage has surpassed the threshold of 1 MB.

{mthree}

GRAFANA DASHBOARD FOR VISUALIZING METRICS OF MYSQL TESTS (MYSQL EXPORTER)

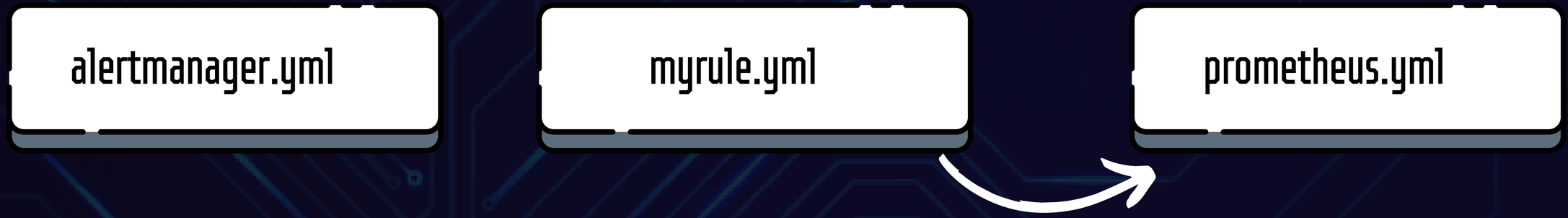


The MySQL stress test has caused the queries per second to exceed the set threshold of 5 QPS.

{mthree}

USING ALERTMANAGER FOR ALERTS VIA EMAILS

Alertmanager manages alerts from Prometheus by routing them to specified email. When an alert is triggered, it sends a formatted notification to the designated email addresses using the configured SMTP server, allowing for real-time awareness of system issues.



Configures Alertmanager to manage alerts from Prometheus, specifying notification routes and receiver details.

Contains custom alert rules for Prometheus, defining when alerts should be triggered based on metric thresholds.

Main configuration file for Prometheus, detailing how to scrape metrics, set job targets, and link to external services like Alertmanager.

{mthree}

ALERTS SENT TO THE USER

1 alert for alertname=DiskUsageHigh

View In Alertmanager

[1] Firing

Labels
alertname = DiskUsageHigh
device = /dev/mapper/cs_vbox-root
fstype = xfs
instance = 192.168.1.6:9100
job = prometheus
mountpoint = /
severity = critical
Annotations
description = Disk usage has exceeded the threshold of 80%. Current usage is at 99.20063797577855%.

1 alert for alertname=CPUUsageHigh

View In Alertmanager

[1] Firing

Labels
alertname = CPUUsageHigh
instance = 192.168.1.6:9100
severity = critical
Annotations
description = CPU usage has exceeded the threshold of 80%.
summary = CPU Usage Alert

1 alert for alertname=MemoryUsageHigh

View In Alertmanager

[1] Firing

Labels
alertname = MemoryUsageHigh
instance = 192.168.1.6:9100
job = prometheus
severity = critical
Annotations
description = Memory usage has exceeded the threshold of 80%.
summary = Memory Usage Alert

1 alert for alertname=NetworkUsageHigh

View In Alertmanager

[1] Firing

Labels
alertname = NetworkUsageHigh
device = lo
instance = 192.168.1.6:9100
job = prometheus
severity = critical
Annotations
description = Network usage has exceeded the threshold of 1 MB.
summary = Network Usage Alert

2 alerts for alertname=MySQLStressTestHighUsage

View In Alertmanager

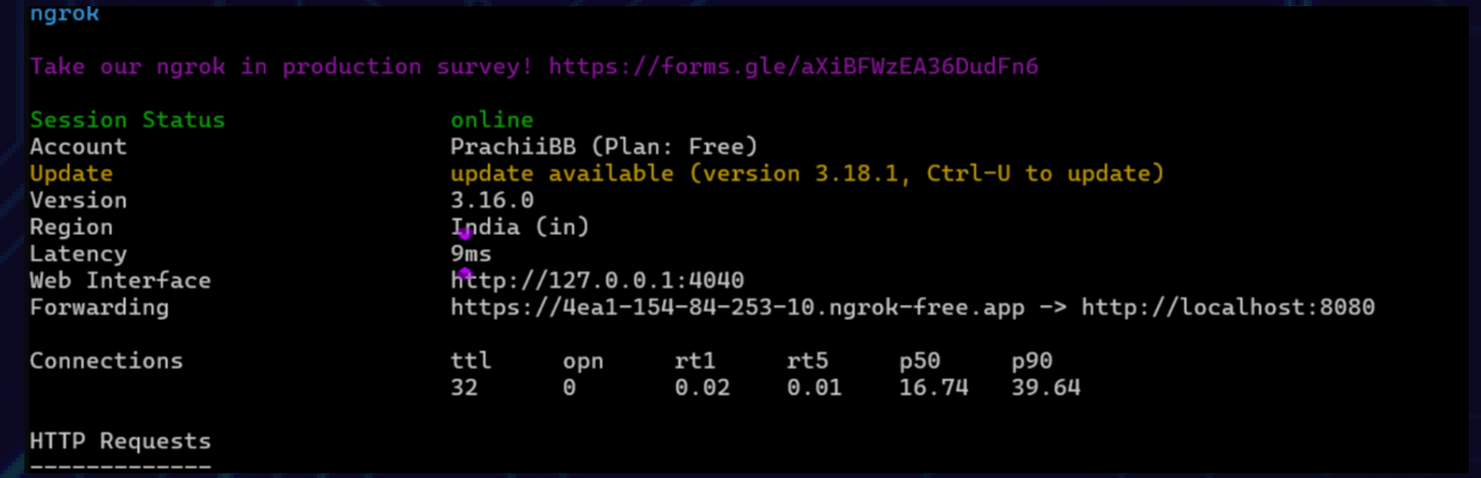
[2] Firing

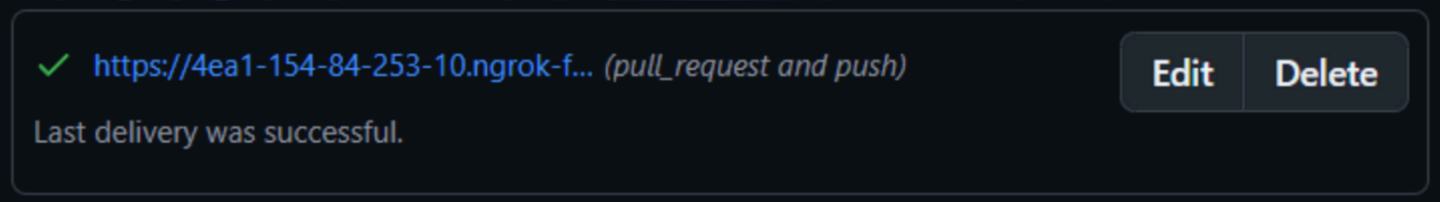
Labels
alertname = MySQLStressTestHighUsage
instance = 192.168.1.64:9105
job = prometheus
severity = critical
Annotations
description = CPU usage has exceeded the threshold
summary = High CPU usage detected in MySQL stress test

{mthree}

VERSION CONTROL & AUTOMATION

- Ngrok creates a secure tunnel to expose the local Jenkins server to the internet, allowing it to receive webhook notifications from GitHub.
- We configured a GitHub webhook that points to the ngrok URL, so whenever there are changes in the repository.
- GitHub sends a POST request to Jenkins, triggering an automatic build process.



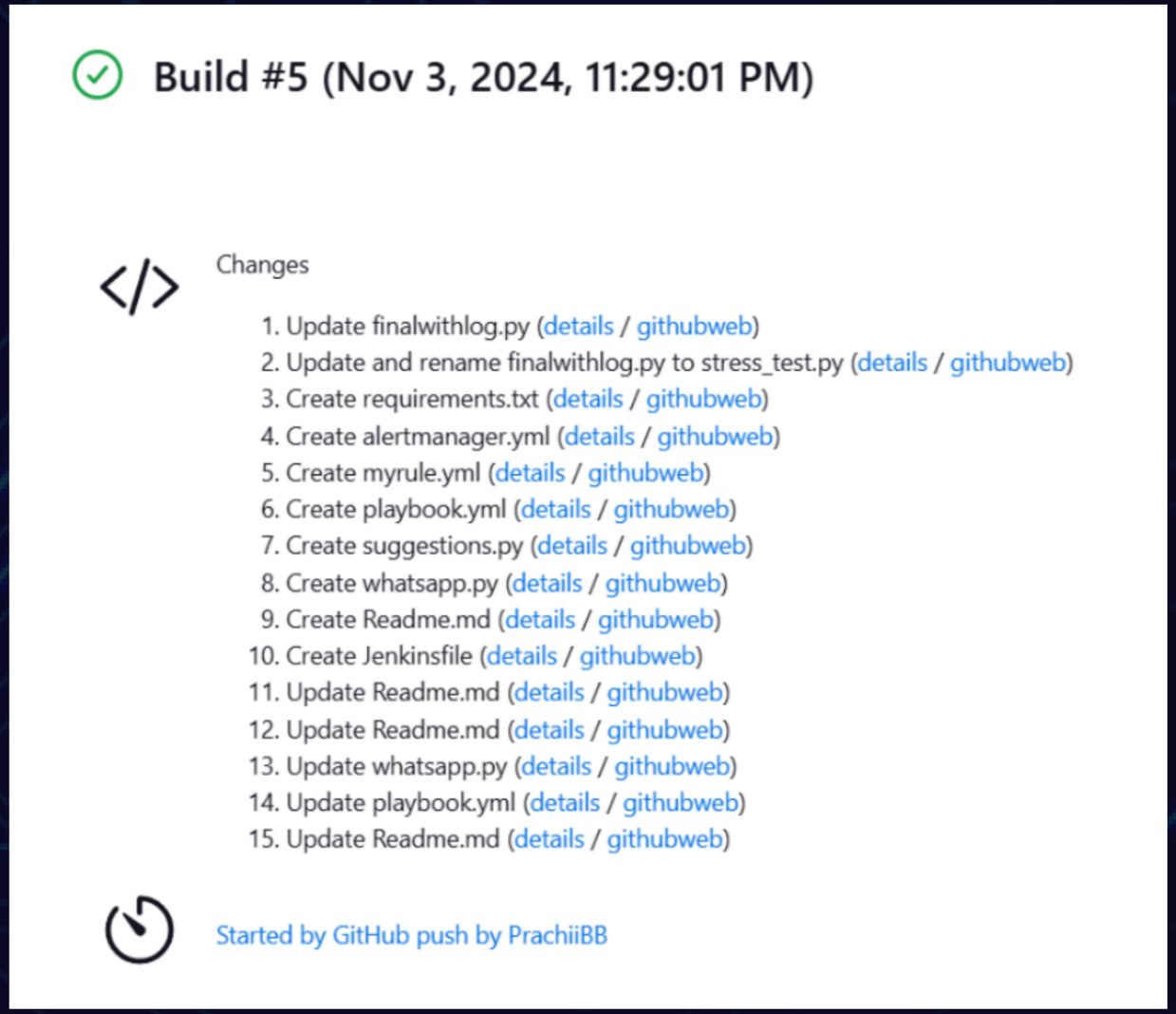


{mthree}

VERSION CONTROL & AUTOMATION

- To enable automatic Jenkins builds upon changes in the GitHub repository, we configured a webhook in GitHub that triggers the Jenkins job whenever a push event occurs. Additionally, the Jenkinsfile defines the build process, ensuring that the pipeline runs smoothly with every change pushed to the repository.

- This integration facilitates continuous integration by ensuring that the latest code changes are always built and tested without manual intervention.

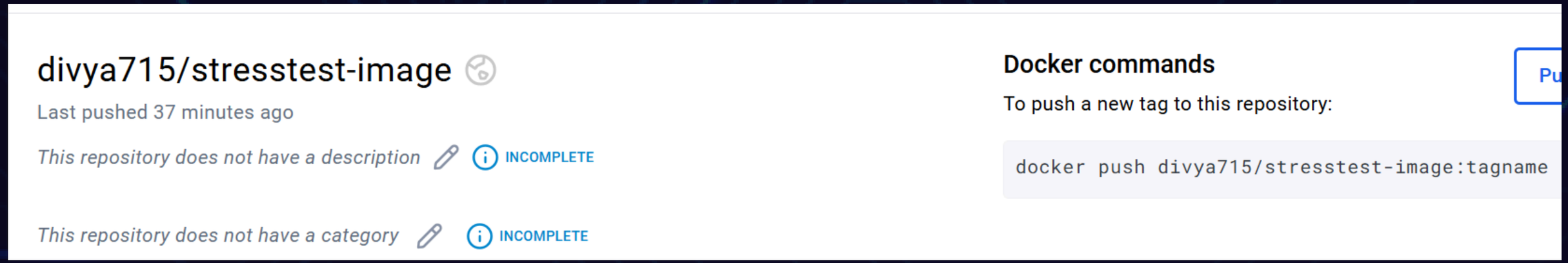


{mthree}

CONTAINERIZATION AND ORCHESTRATION

Containerization with Docker

1. Creating a dockerfile 📝
2. Building the docker image 🔨
3. Running the docker container 🚀
4. Pushing the docker image to dockerhub 🌐



{mthree}

CONTAINERIZATION AND ORCHESTRATION

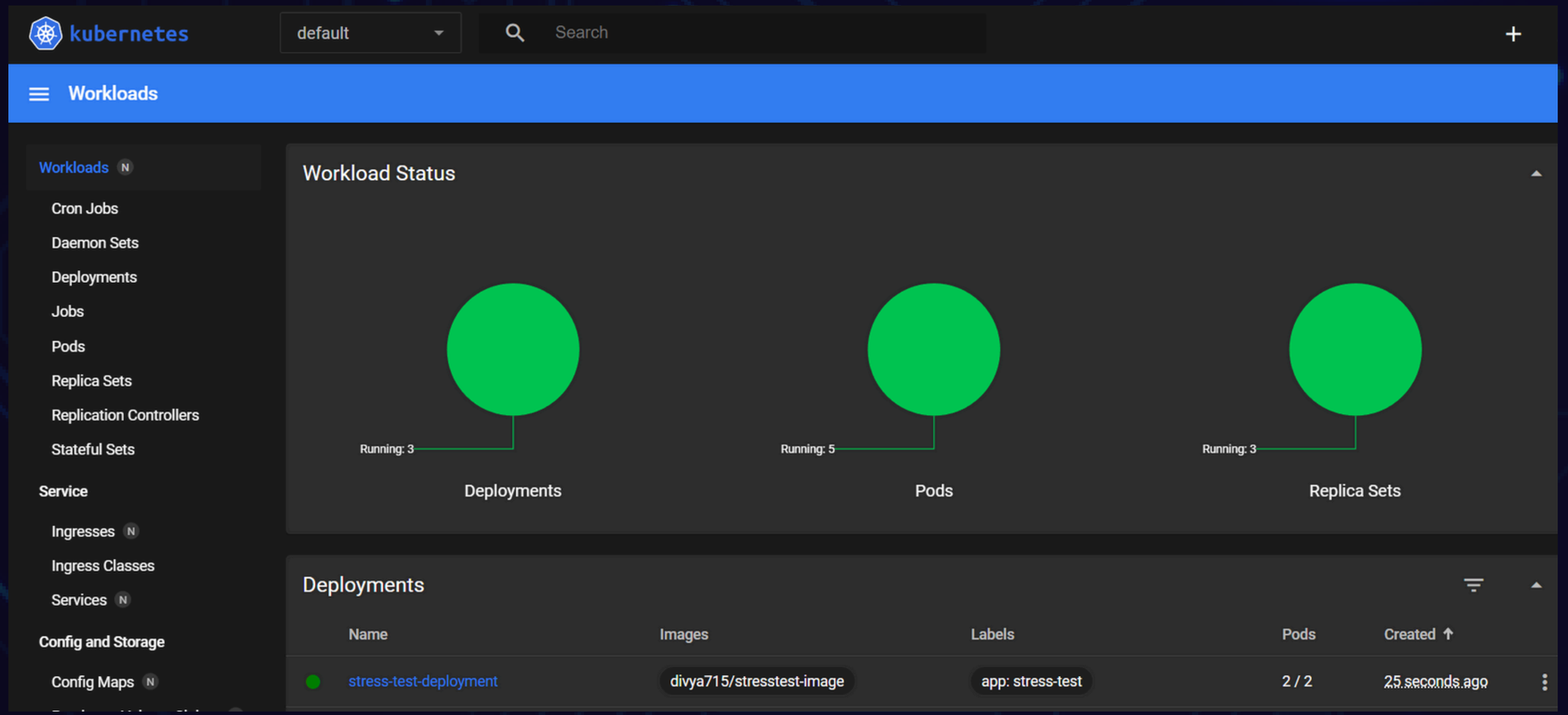
Orchestration with Kubernetes

1. Creating deployment YAML file 📦
2. Creating service YAML file 🌐
3. Deploying to Kubernetes 🚀
4. Monitoring pods in Kubernetes 📊

```
[root@master ~]# kubectl get pods
NAME                                     READY   STATUS    RESTARTS       AGE
stress-test-deployment-6988fdf8fd-2h4pq  1/1     Running   5 (102s ago)   5m24s
stress-test-deployment-6988fdf8fd-5rs2m  1/1     Running   5 (107s ago)   5m24s
```

{mthree}

KUBERNETES DASHBOARD



The Kubernetes dashboard shows the workload status for a stress testing deployment, with 2 replica sets currently running for the stress-test-deployment.

{mthree}

LOG ANALYSIS AND SUGGESTIONS USING CHATGPT API

The logs generated during stress testing from stress_test.log. The logs are analyzed using suggestions.py, which generates a summary of insights in suggestions.txt. Finally, we send these suggestions directly to a WhatsApp account using Twilio, ensuring real-time feedback and analysis.

Python Script:

Logs generated using logging module of Python after stress testing.

Suggestions Script:

Using ChatGpt API to analize the logs and generate sugesstions and analysis.

Whatsapp Script:

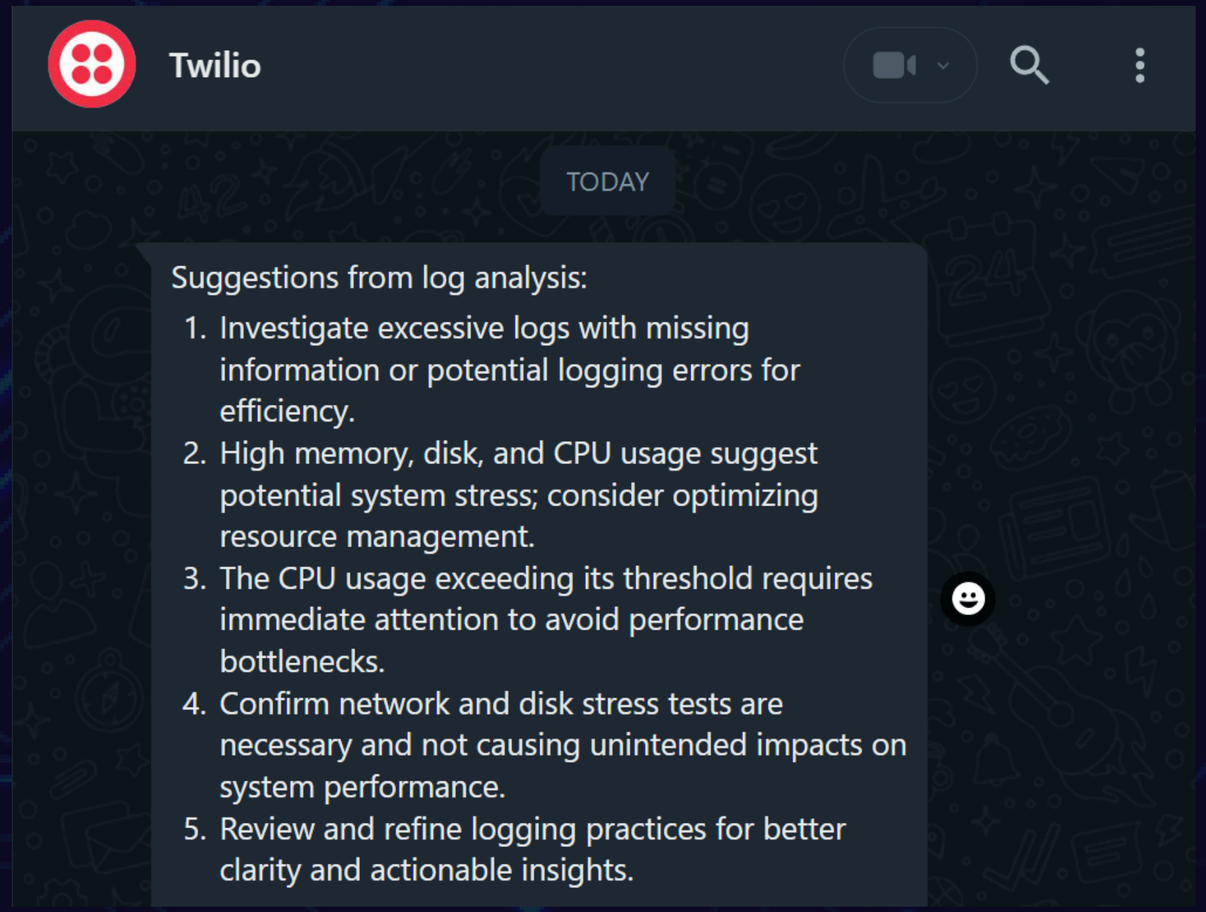
Using Twilio to send the generated analysis and suggestions to the user.

{mthree}

LOGS GENERATED IN STRESS_TEST.LOG USING LOGGING MODULE

```
Memory Usage: 25.5%
Increasing memory stress to exceed threshold.

Memory Usage: 79.8899417792012%
Memory Usage: 79.94629158825019%
Memory Usage: 80.00264139729917%
Memory stress test reached target usage.

Disk Usage: 36.0%
Increasing disk stress to exceed threshold.
Disk Usage: 36.1%
Disk Usage: 36.2%

Disk Usage: 83.3%
Disk stress test reached target usage.

Increasing network stress to attempt to reach threshold.
Network stress test reached target usage.

Initial CPU Usage: 4.0%
Increasing CPU stress to exceed threshold.
CPU Usage: 96.4%
CPU Usage: 100.0%
CPU stress test completed.

- Starting new HTTP connection (1): 192.168.0.104:9104
- http://192.168.0.104:9104 "GET /metrics HTTP/1.1" 200 None
process_cpu_seconds_total: 17.8
- CPU usage exceeds threshold - 17.8 seconds (Threshold: 1.0 seconds)
Exiting
```

MESSAGE RECEIVED ON WHATSAPP WITH SUGGESTIONS USING TWILIO



{mthree}

THANK YOU