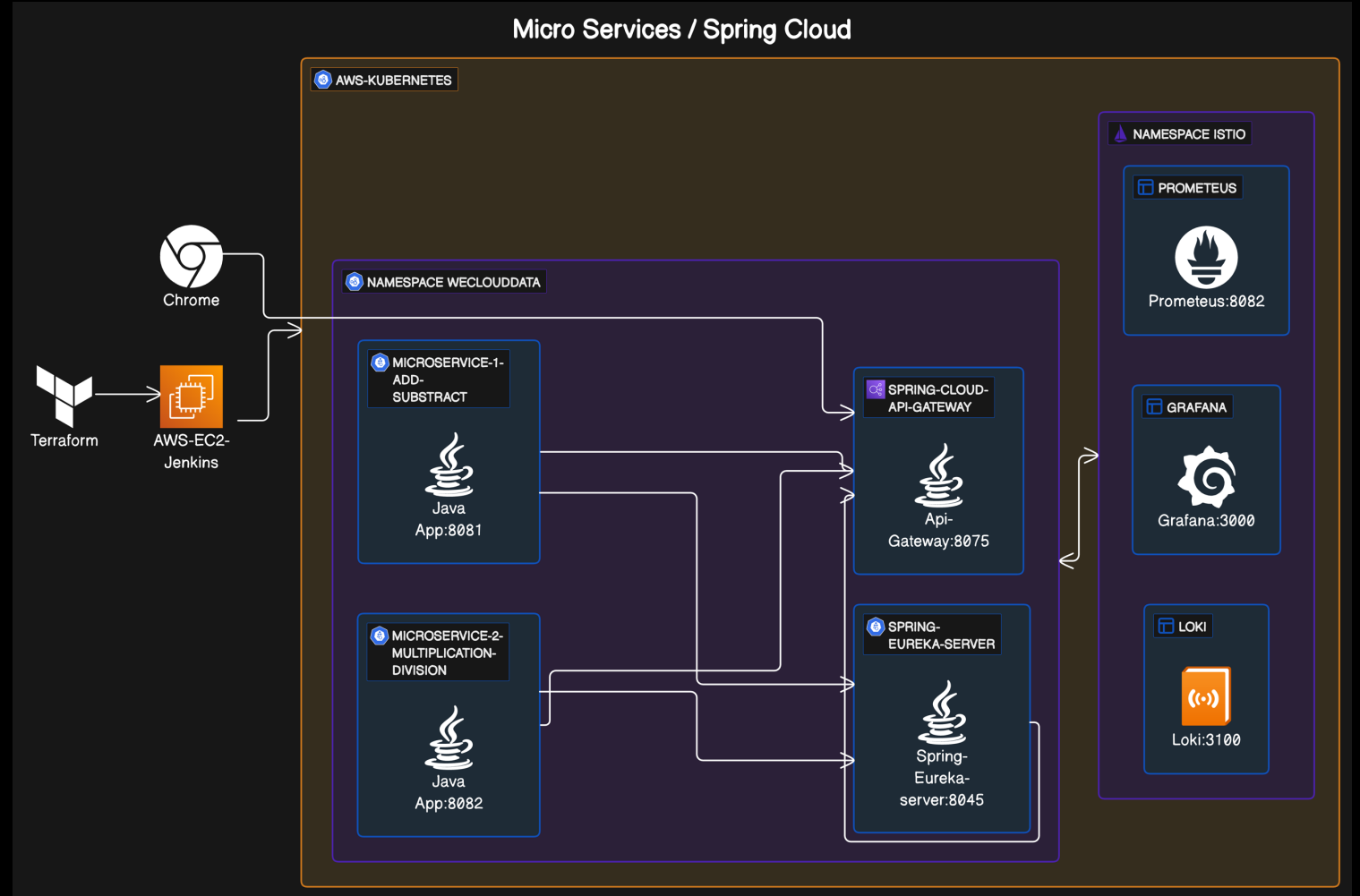




# **SPRING BOOT CLOUD NETFLIX ON KUBERNETES ARCHITECTURE**

# INTRODUCTION

the architecture of a microservices-based system utilizing Spring Boot Cloud Netflix components, deployed on Kubernetes (K8S). The system is designed for high availability, scalability, and robust service discovery and monitoring.





## Architecture Overview

- Infrastructure Setup
  - <https://github.com/maxiplux/Final-Capstone-Project-weclouddata/blob/main/README.md#architecture-overview>
- Kubernetes (K8S)
- Microservices
- API Gateway
- Service Discovery
- Monitoring and Observability



```
brew reinstall tree
➔ Final-Capstone-Project-weclouddata git:(main) tree
.
├── LICENSE
├── README.md
├── api-gateway
│   ├── deployment.yml
│   └── service.yaml
├── aws-eks-cluster
│   ├── cluster.yaml
│   ├── delete-cluster.sh
│   ├── eks-installer.sh
│   └── install-cluster.sh
├── aws-eks-permissions
│   └── main.tf
├── aws-jenkins-machine
│   ├── README
│   ├── docker-compose.yml
│   └── terraform
│       ├── check_status.sh
│       └── main.tf
├── aws-spring-netflix.png
├── deployer.sh
├── docker-compose.yml
├── grafana-dashboard
│   └── spring-boot-dashboard.json
├── loki-dns.yml
├── math-add-subtract
│   ├── deployment.yml
│   └── service.yaml
├── math-division-multiplication
│   ├── deployment.yml
│   └── service.yaml
├── monitoring
│   ├── grafana.yaml
│   ├── jaeger.yaml
│   ├── kiali.yaml
│   ├── loki.yaml
│   └── prometheus.yaml
├── monitoring.sh
├── namespace.yml
├── services-discovery-eureka
│   ├── deployment.yml
│   └── service.yaml
├── tester.py
└── undeployer.sh
```

# Components in action

The screenshot displays a Jenkins web interface for a pipeline named 'Java 1'. The browser's address bar shows the URL: `107.21.53.146:8080/blue/organizations/jenkins/Java/detail/math-division-and-multiplication/1/pipeline`. The interface includes a green header bar with the pipeline name and a 'Pipeline' tab. Below the header, a summary section shows the branch 'math-division-and-multiplication' and the commit '5 hours ago'. A horizontal flow diagram illustrates the pipeline stages: Start, Checkout, build and test, Docker Login, Deploy to Docker Registry, Deploy to K8S (highlighted with a blue circle), and End. The 'Deploy to K8S' stage is expanded, showing a list of commands executed in a shell script:

- > `git rev-parse HEAD` — Shell Script (<1s)
- > `sed -i 's/TAG_HERE/36fae57d8992c8663459bbd098fe4bcb1971ea32/g' deployment.yml` — Shell Script (<1s)
- > `kubectl apply -f deployment.yml -n weclouddata` — Shell Script (3s)

Each command is preceded by a green checkmark, indicating successful execution. A 'Restart Deploy to K8S' button is visible in the top right corner of the expanded stage.

# Components in action

Not Secure 107.21.53.146:20001/kiali/console/graph/namespaces/?traffic=grpc%2CgrpcRequest%2Chttp%2ChttpRequest%2Ctcp%2CtcpSent&graphType=versionedApp&namespaces=wec... macOS Sonoma 14.4.1 is available and will be installed later tonight.

Google Podcasts Inbox (15,092) - m... valen la pena AI ASISTANCE AWS Cloud Institut... Amazon Web Servi... AWS devops-bootcamp WeCloudData COLOMBIANOS E... ai Inbox: maxiplux@g...

Kubernetes

Overview

Graph

Applications

Workloads

Services

Istio Config

Distributed Tracing

Mesh

Namespace: weclouddata Traffic Versioned app graph

Display Find... Hide...

Apr 15, 11:01:27 PM ... 11:02:27 PM

```
graph TD; api-gateway[api-gateway latest] --> eureka-server[eureka-server]; math-add-subtract[math-add-subtract latest] --> eureka-server; math-add-subtract --> eureka-server-latest[eureka-server latest]; math-division-multiplication[math-division-multiplication latest] --> eureka-server; math-division-multiplication --> loki-latest[loki latest]; loki[loki] --> loki-latest; eureka-server --> eureka-server-latest;
```

Current Graph

NS weclouddata N/A

5 apps (5 versions)

2 services

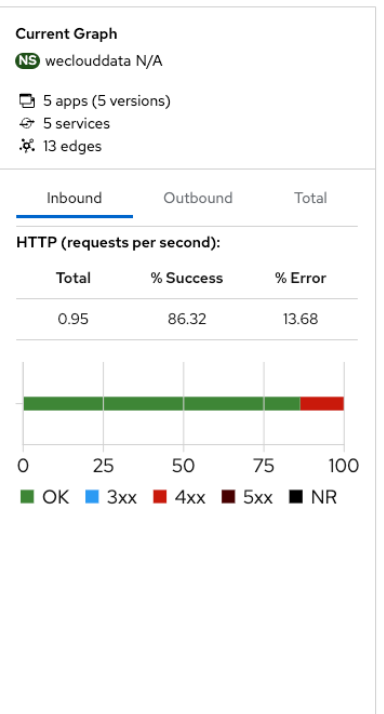
7 edges

Inbound	Outbound	Total
HTTP (requests per second):		
Total	% Success	% Error
1.02	96.08	3.92

0 25 50 75 100

OK 3xx 4xx 5xx NR

>> Hide



# CONCLUSION

This architecture is designed to be resilient and flexible, enabling quick scaling and easy maintenance of the microservices. Monitoring tools ensure the health of the system is continuously observed, allowing for proactive management of the system's operations.



# REFERENCES

- TEMPLATE
  - [https://pptthemes.com/netflix-powerpoint-template/?fbclid=IwAR0DMqpBahQ-QUaXr6FQMUp1\\_fes0LmuOmrASC1w7aF94JHbv-KixOmKOYU\\_aem\\_Af1mG7rdTPdvijodH0rBf4jGsLNVR7jKJ5lVn9OgT50ADfMgF3RCPbw3Gq9hx8AHc8&amp](https://pptthemes.com/netflix-powerpoint-template/?fbclid=IwAR0DMqpBahQ-QUaXr6FQMUp1_fes0LmuOmrASC1w7aF94JHbv-KixOmKOYU_aem_Af1mG7rdTPdvijodH0rBf4jGsLNVR7jKJ5lVn9OgT50ADfMgF3RCPbw3Gq9hx8AHc8&amp)
- MICROSERVICES CODE
  - <https://github.com/maxiplux/k8s-istio-math>
- ISTIO
  - <https://istio.io/latest/>
- SPRING CLOUD
  - <https://spring.io/projects/spring-cloud>

**THANKS**