CMM707 – Cloud Computing

Coursework Report

Name: Avishka Vithanage

RGU ID: 2506663

IIT ID: 20241889

Introduction

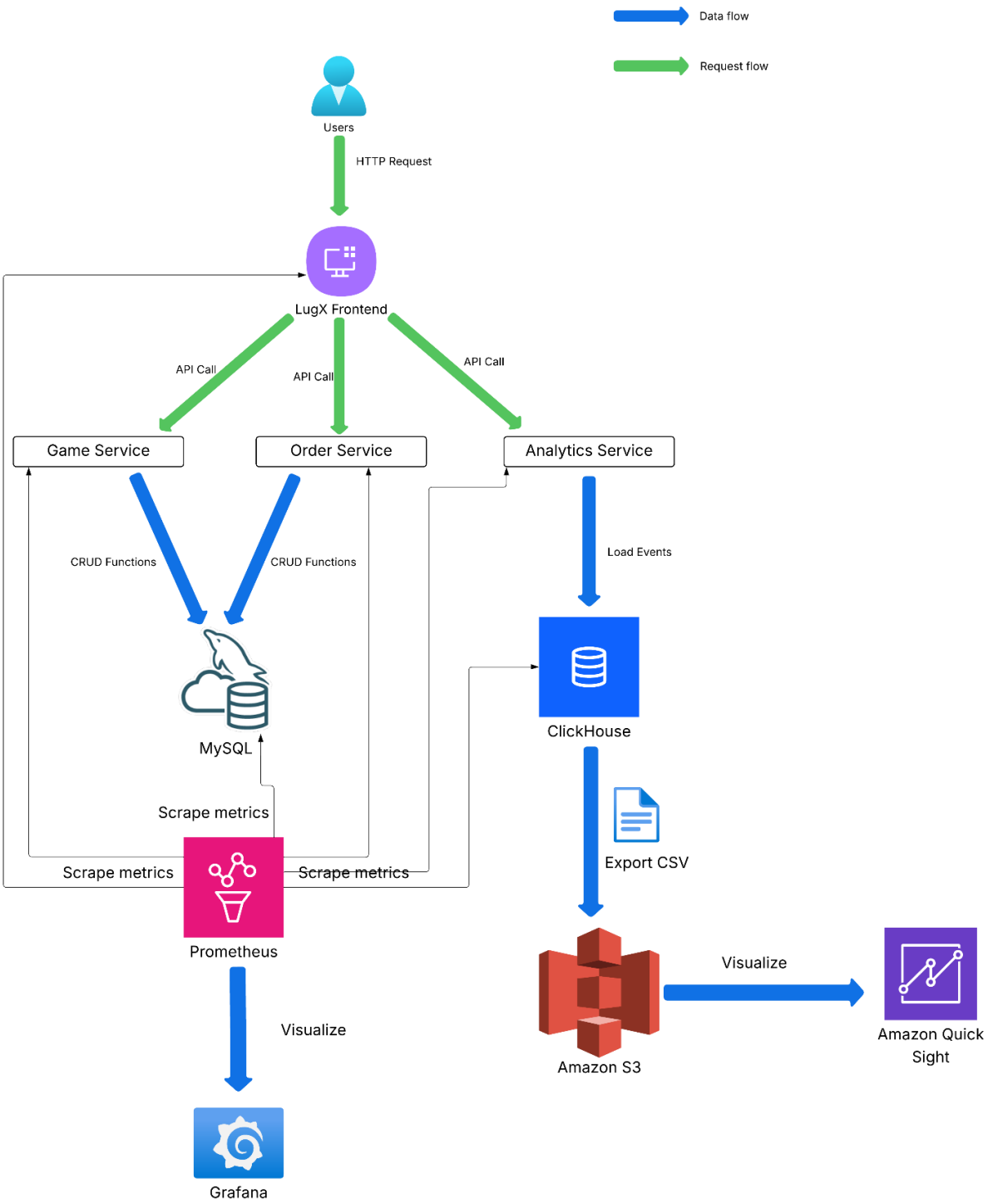
The design and implementation of a microservices-based deployment for the Lugx Gaming platform, created as a component of the CMM707 Cloud Computing course, are presented in this report. In keeping with the module's ability to use both local and cloud-native Kubernetes environments, the solution was created to run exclusively on a locally deployed Kubernetes cluster using Minikube. This strategy avoided the expense and account suspension risks that are occasionally connected to public cloud accounts near assessment deadlines while guaranteeing complete functionality.

The platform architecture consists of multiple containerized microservices:

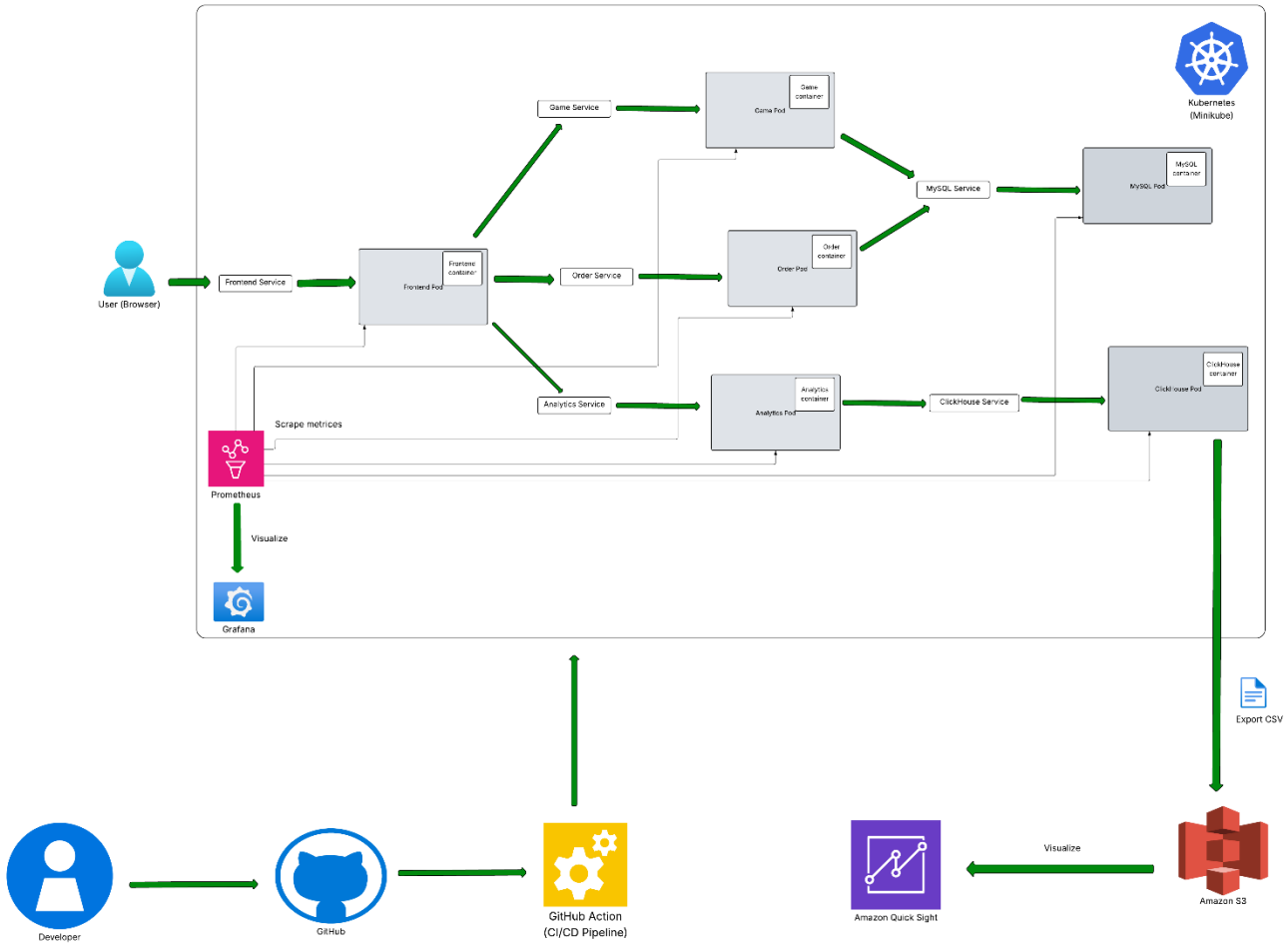
- **Frontend** - a static web application serving the Lugx Gaming user interface.
- **Game Service** - a REST API for storing and updating game details.
- **Order Service** - a REST API for managing orders and cart transactions.
- **Analytics Service** - an API integrated with ClickHouse to capture web analytics events such as page views and clicks.

Docker was used to containerize each microservice, and Kubernetes manifests were used to deploy them to Minikube. Grafana and Prometheus were used to achieve observability, and ClickHouse was used to store and visualize analytics data using AWS QuickSight. Utilizing a CI/CD pipeline with GitHub Actions to automate builds, tests, and deployments to the local cluster, the deployment adhered to best practices for scalability, resilience, and maintainability. To ensure high availability, rolling updates and optional blue-green deployment techniques were included.

Solution Architecture Diagram



Deployment Architecture Diagram



Security and Ethics Challenges in the Lugx Gaming Platform

Security Risks

- Insecure API Endpoints: Public APIs (like /orders, /track) that are not properly authenticated are vulnerable to misuse or illegal access.
- SQL Injection or Input Exploits: Attackers may run malicious SQL or script commands if input is not cleaned up.
- Data Leakage: HTTPS must be used to securely store and transfer sensitive data, such as session IDs and payment-related metadata.
- Improper Secret Handling: It is possible to extract and abuse hard-coded secrets or API tokens from code or Docker files.

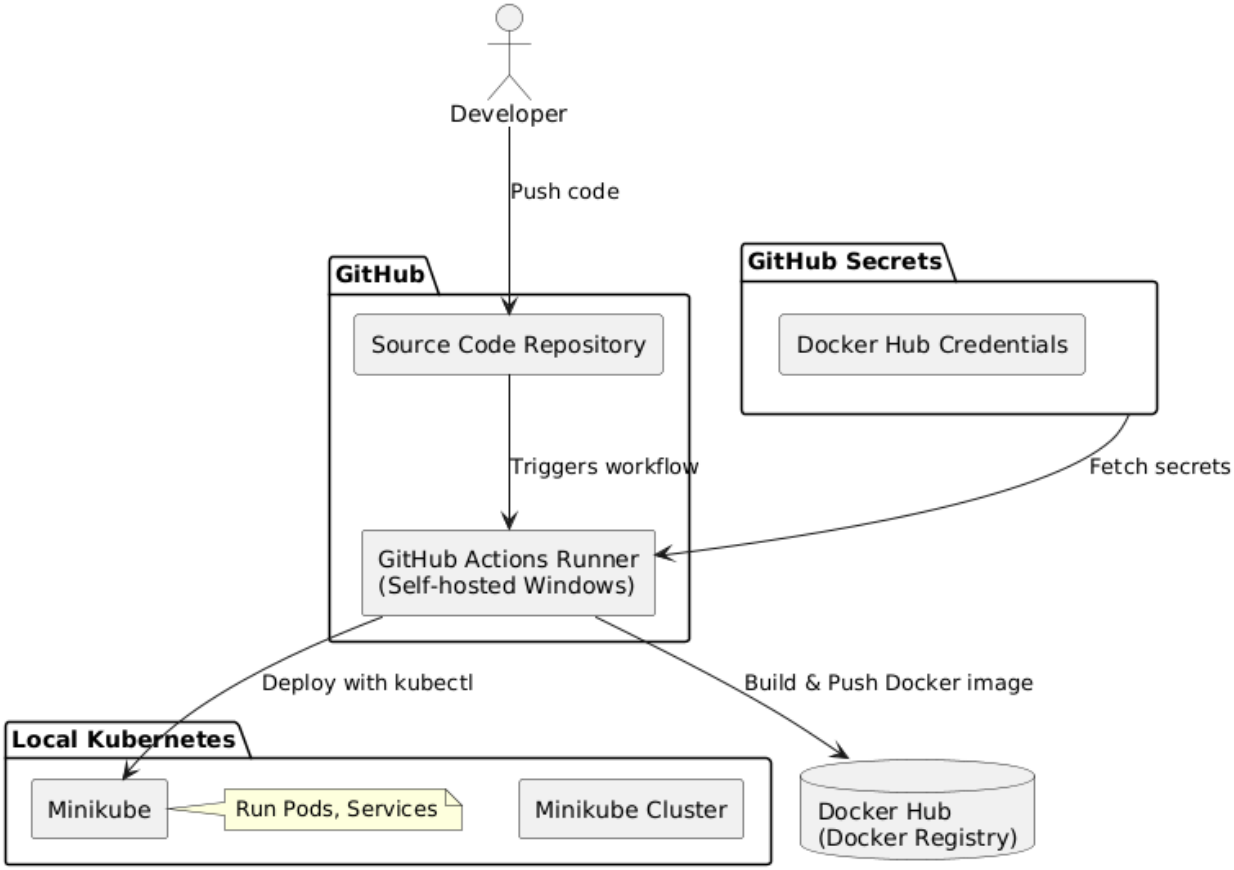
Ethical Concerns

- Privacy Violations: Without clear user consent, tracking user clicks, session length, and scroll depth may violate privacy laws and expectations.
- Over-collection of Data: It may be unethical to record more user behavior than is required, particularly in the absence of a clear goal.
- Data Retention Without Justification: It is unethical to keep web activity logs for an extended period as this raises risks and goes against ethical data minimization guidelines.

Mitigation Strategies

- Secure APIs with appropriate authorization and authentication.
- Use environment variables or Kubernetes Secrets to secure secrets and encrypt all traffic using HTTPS.
- Clearly explain analytics collection in a consent banner or user-facing privacy policy.
- Establish time-based guidelines for analytics log data retention.

CI/CD pipeline Designs Diagram



Process Description

The Lugx Gaming platform's CI/CD pipeline has been set up to operate with a local Kubernetes cluster that is powered by Minikube. The steps are as follows:

1. Trigger: Code pushed to the main branch or manually dispatched via workflow initiates the pipeline.

2. Build: A self-hosted action runner builds Docker images for each microservice (Frontend, Game Service, Order Service, and Analytics Service) after checking out the repository.

3. Push: Using Docker Hub credentials stored in GitHub Secrets, images are tagged and pushed to Docker Hub.

4. Deploy: The runner uses the local kubeconfig on the same Windows machine to run kubectl apply and update Kubernetes manifests.

5. Deployment Strategy: By defaulting to rolling updates, there is no downtime. For safer version switching, a Blue-Green deployment strategy is also supported.

6. Testing: Automated smoke tests examine service health endpoints following deployment. Periodically, integration tests are conducted to confirm functionality.

7. Scheduled Runs: A separate scheduled workflow in GitHub Actions runs the integration test periodically to ensure ongoing reliability.

8. Post-deploy verification: The pipeline checks kubectl rollout status and tails pod logs when needed to confirm a successful rollout.

Security and Ethics in CI/CD and Cloud Deployment

CI/CD and Cloud Security Challenges

• The storage of AWS credentials, database passwords, or tokens in plaintext within docker-compose.yml or CI files (such as GitHub Actions) can lead to significant security breaches.

• Absence of Pipeline Integrity: Malicious code may infiltrate the production environment if the pipeline is not secured by version control permissions or does not enforce commit signatures.

• Misconfigured Ingress or Open Kubernetes Ports: When Kubernetes services are not configured properly, they may be exposed to the outside world, which can lead to brute-force or denial-of-service (DoS) attacks.

• Absence of Audit Logging: Deployments and configuration modifications are not accountable in the absence of audit logs.

CI/CD and Cloud Ethical Concerns

• Abuse of Continuous Testing: If integration tests are periodically started without throttle controls, they may inadvertently overload services or waste cloud resources.

• Deploying Without Verification: When updates are released without user approval, end users may experience a worsening of their experience or encounter bugs.

• Ignoring Cost and Environmental Impact: Over-provisioned cloud resources and inefficient build pipelines lead to excessive energy use, which raises sustainability-related ethical questions.

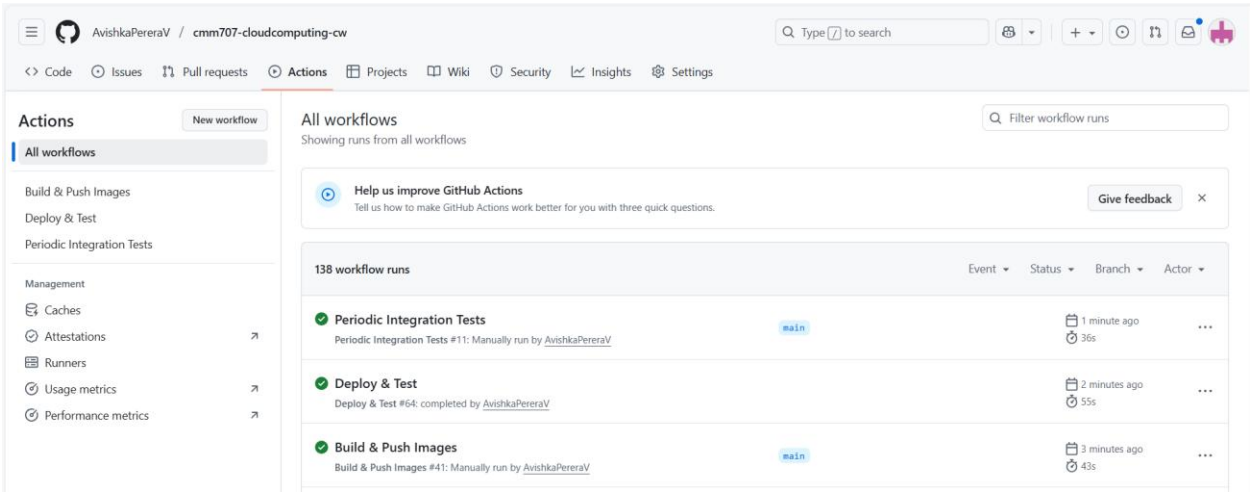
Mitigation Strategies

- Make use of safe secret management solutions, such as AWS Secrets Manager and GitHub Secrets.
- For CI/CD service accounts, apply the least privilege principle.
- To keep track of deployment activities, including audit logging and monitoring.

Utilize scheduling and cost optimization tools to handle computation in an ethical manner.

CI/CD Pipeline Scripts

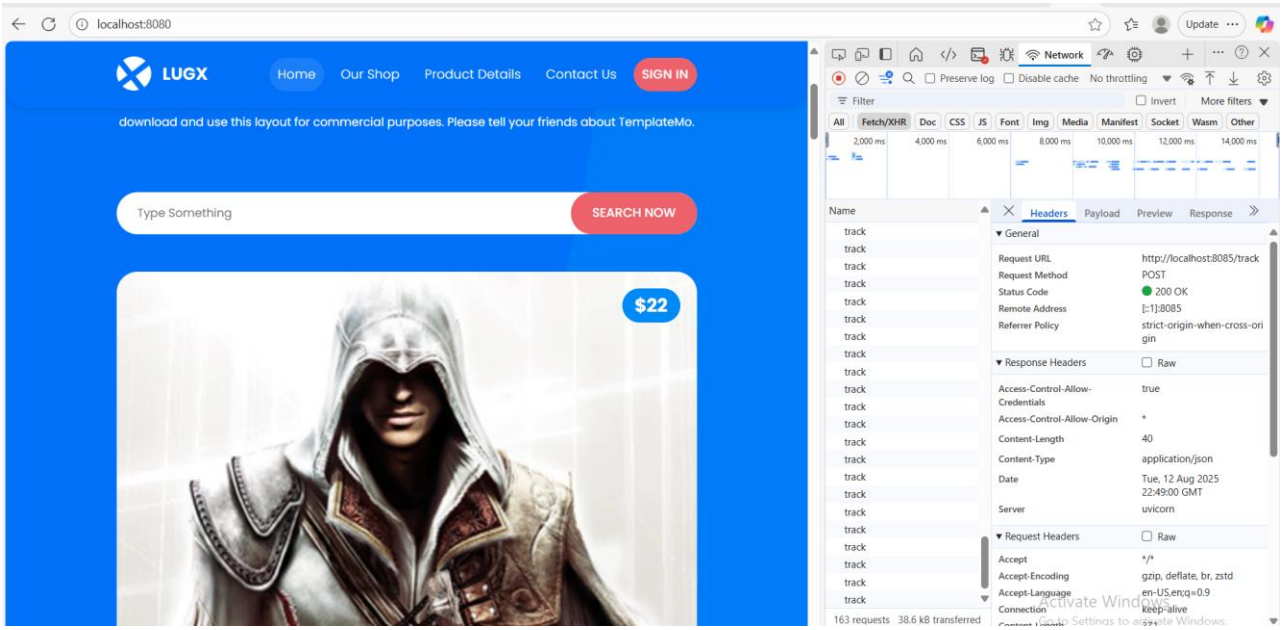
cmm707-cloudcomputing-cw/.github/workflows at main · AvishkaPereraV/cmm707-cloudcomputing-cw

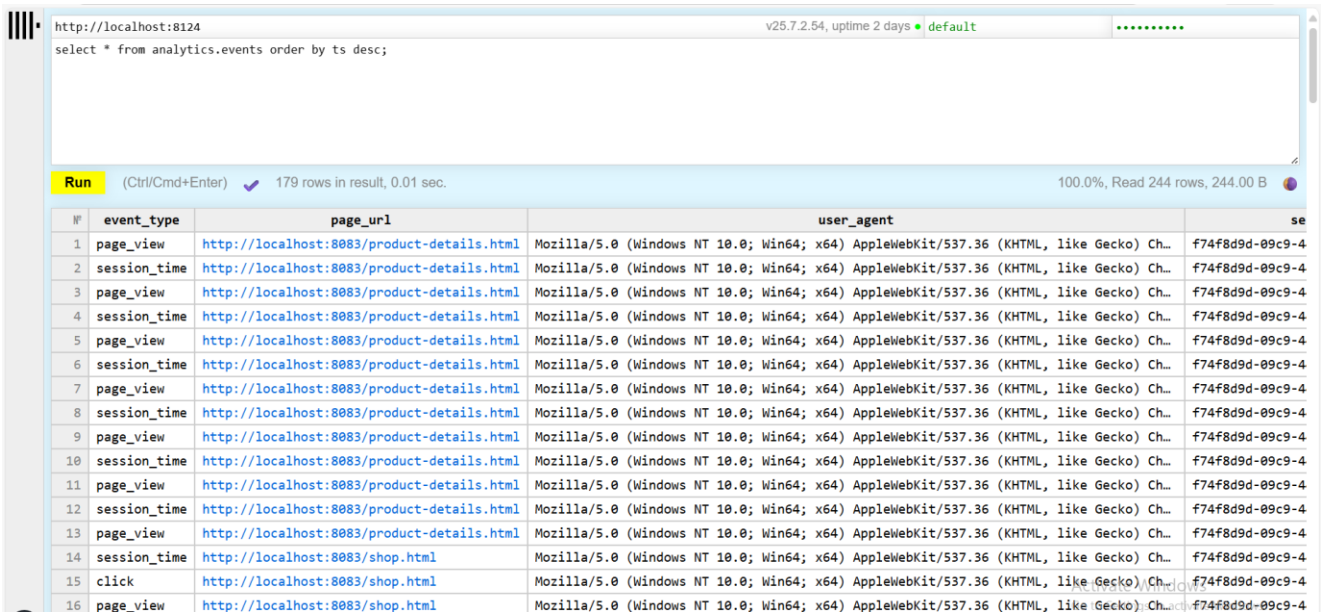


GitHub Repo

AvishkaPereraV/cmm707-cloudcomputing-cw: CMM707 - Cloud Computing Coursework

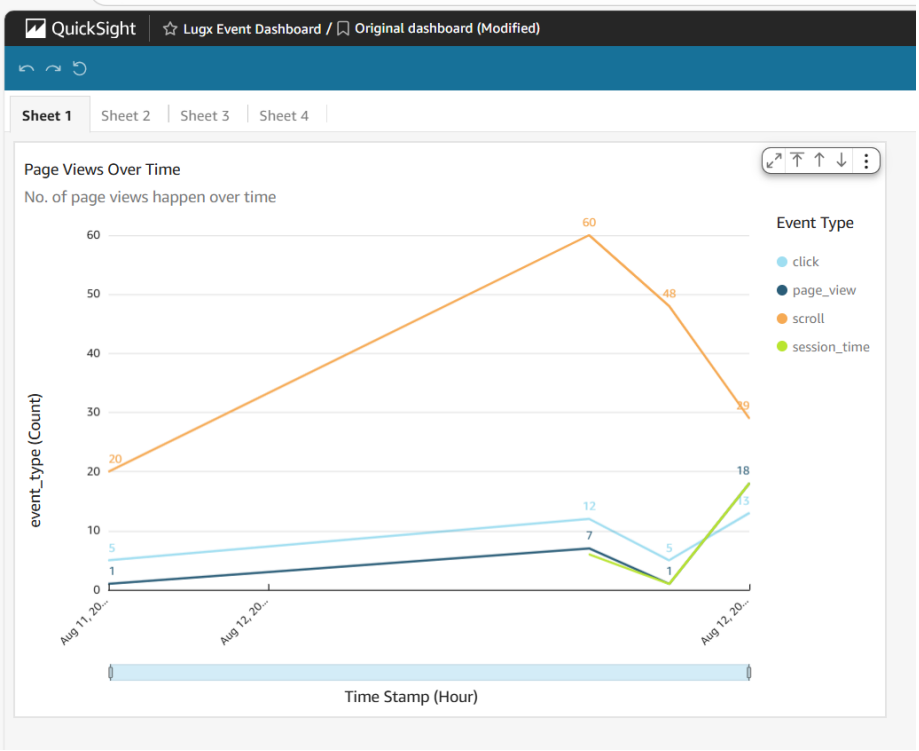
Web analytics + ClickHouse

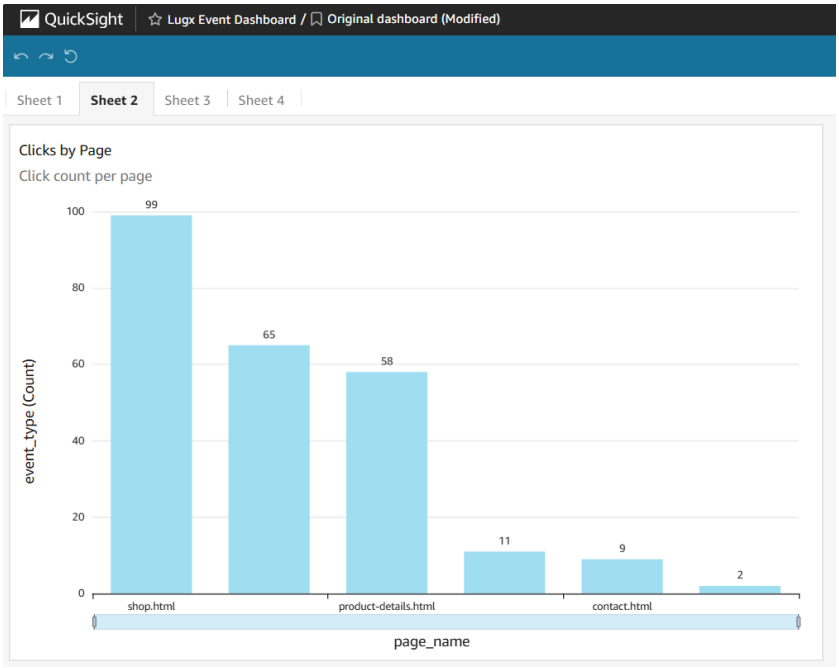


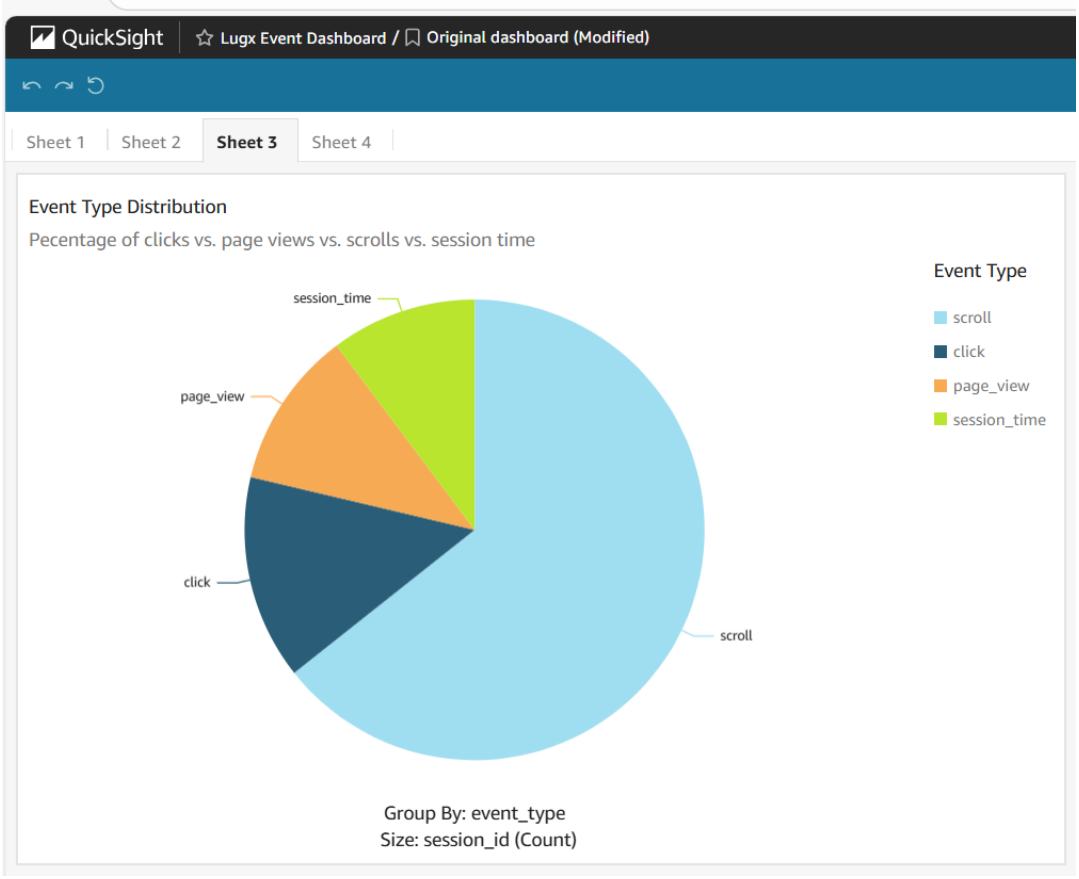


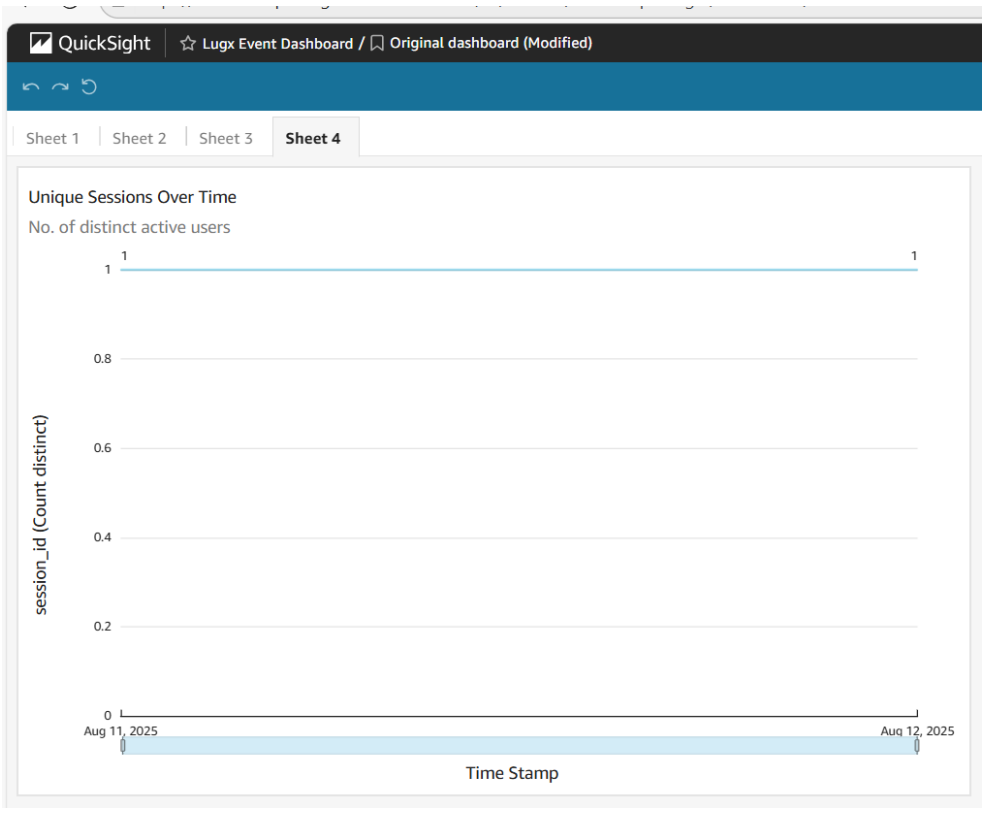
№	event_type	page_url	user_agent	se
1	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
2	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
3	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
4	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
5	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
6	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
7	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
8	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
9	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
10	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
11	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
12	session_time	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
13	page_view	http://localhost:8083/product-details.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
14	session_time	http://localhost:8083/shop.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
15	click	http://localhost:8083/shop.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4
16	page_view	http://localhost:8083/shop.html	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch…	f74f8d9d-09c9-4

ClickHouse + QuickSight Visualizations

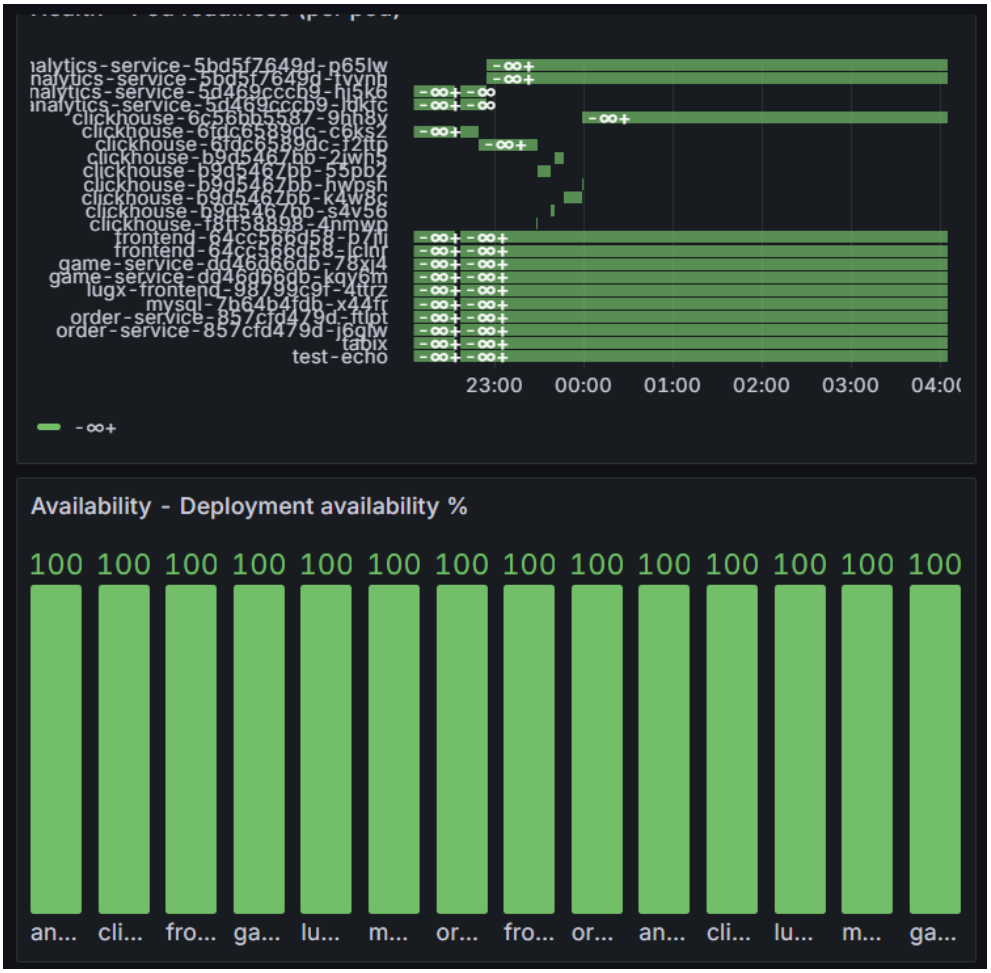


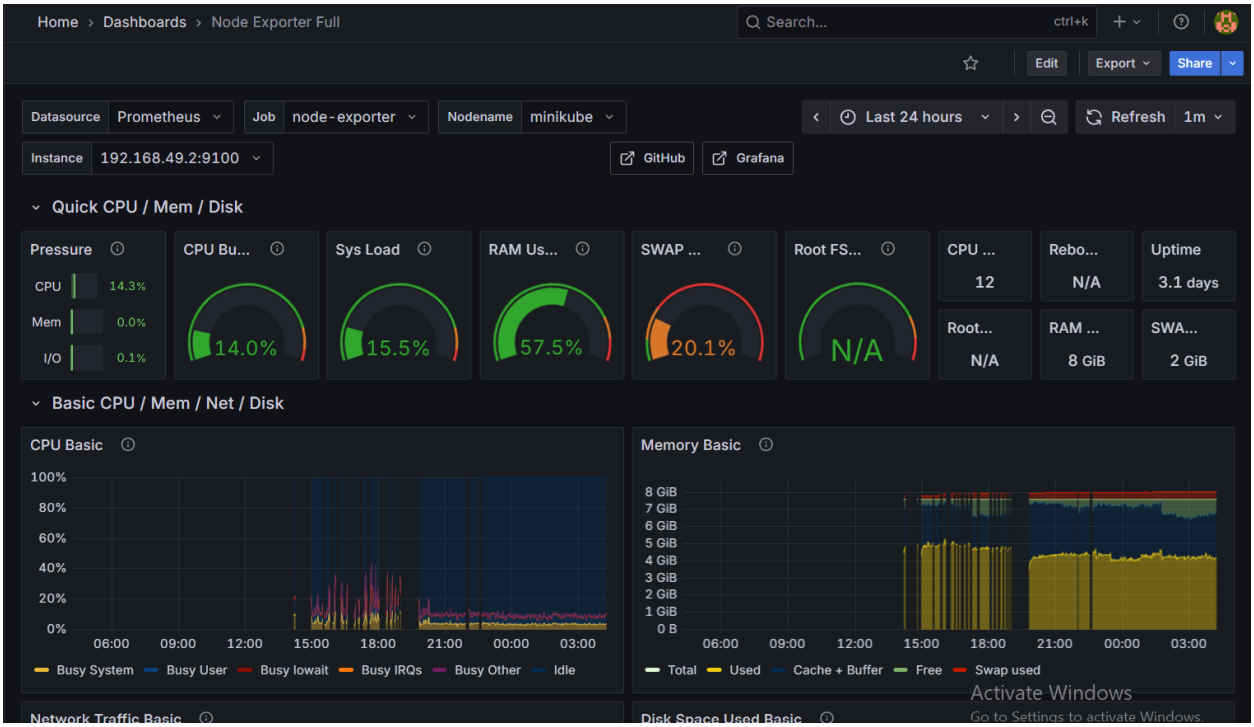


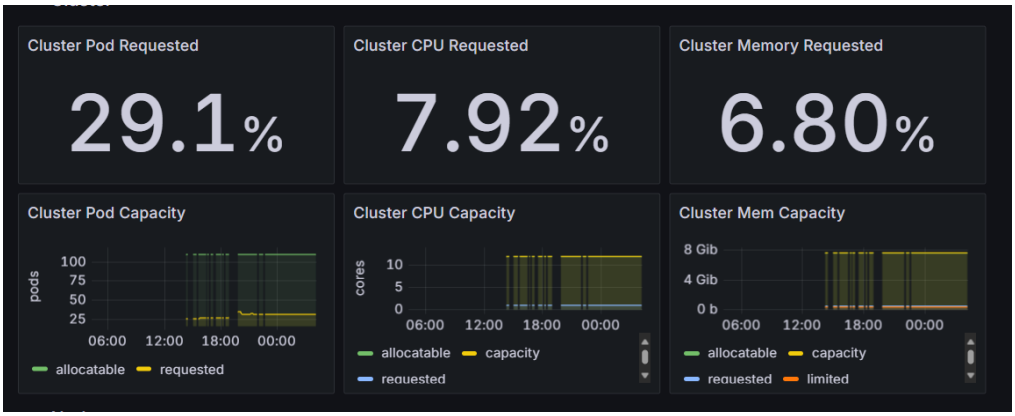

QuickSight
Lugx Event Dashboard / Original dashboard (Modified)
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Event Type Distribution
Pecentage of clicks vs. page views vs. scrolls vs. session time
session_time
page_view
click
scroll
Event Type
scroll
click
page_view
session_time
Group By: event_type
Size: session_id (Count)

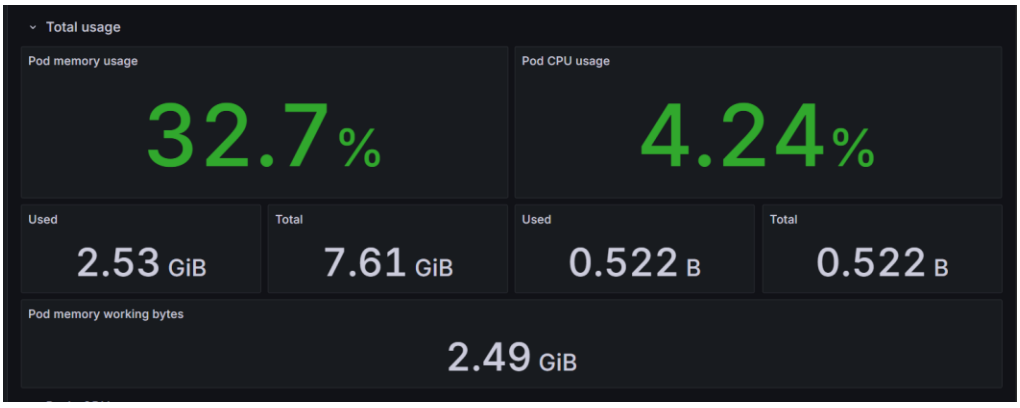

QuickSight
Lugx Event Dashboard / Original dashboard (Modified)
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Unique Sessions Over Time
No. of distinct active users
1
1
1
0.8
0.6
0.4
0.2
0
session_id (Count distinct)
Aug 11, 2025
Aug 12, 2025
Time Stamp

Prometheus + Grafana Visualizations




Home › Dashboards › Node Exporter Full
Search... ctrl+k
Edit
Export
Share
Datasource Prometheus
Job node-exporter
Nodename minikube
Last 24 hours
Refresh 1m
Instance 192.168.49.2:9100
GitHub
Grafana
Quick CPU / Mem / Disk
Pressure
CPU 14.3%
Mem 0.0%
I/O 0.1%
CPU Bu... 14.0%
Sys Load 15.5%
RAM Us... 57.5%
SWAP ... 20.1%
Root FS... N/A
CPU ... 12
Rebo... N/A
Uptime 3.1 days
Root... N/A
RAM ... 8 GiB
SWA... 2 GiB
Basic CPU / Mem / Net / Disk
CPU Basic
100%
80%
60%
40%
20%
0%
06:00 09:00 12:00 15:00 18:00 21:00 00:00 03:00
Busy System Busy User Busy Iowait Busy IRQs Busy Other Idle
Memory Basic
8 GiB
7 GiB
6 GiB
5 GiB
4 GiB
3 GiB
2 GiB
1 GiB
0 B
06:00 09:00 12:00 15:00 18:00 21:00 00:00 03:00
Total Used Cache + Buffer Free Swap used
Activate Windows
Go to Settings to activate Windows.
Network Traffic Basic
Disk Space Used Basic


Cluster Pod Requested
29.1%
Cluster CPU Requested
7.92%
Cluster Memory Requested
6.80%
Cluster Pod Capacity
pods
100
75
50
25
06:00 12:00 18:00 00:00
allocatable requested
Cluster CPU Capacity
cores
10
5
0
06:00 12:00 18:00 00:00
allocatable capacity
requested
Cluster Mem Capacity
8 Gib
4 Gib
0 b
06:00 12:00 18:00 00:00
allocatable capacity
requested limited


Total usage
Pod memory usage
32.7%
Pod CPU usage
4.24%
Used
2.53 GiB
Total
7.61 GiB
Used
0.522 B
Total
0.522 B
Pod memory working bytes
2.49 GiB

Runbook (Deployment Steps)

1. Prerequisites
    a. Minikube and Docker Desktop are running on the Windows host that runs the self-hosted runner.
    b. GitHub Secrets exist: CLICKHOUSE_USER, CLICKHOUSE_PASSWORD, DOCKERHUB_USERNAME, DOCKERHUB_TOKEN, KUBE_CONFIG_DATA (base64 of kubeconfig).
    c. Kubernetes manifests are committed:
        i. frontend and analytics service: single Deployment with RollingUpdate strategy.
        ii. game service and order service: two Deployments (-blue, -green) and a single Service with a color selector.
    d. Basic smoke tests are wired in the workflow (HTTP health checks via port-forward).
    e. Grafana and Prometheus installed in the cluster for monitoring.
    f. Windows Task Scheduler jobs set up for ClickHouse → S3 export, and QuickSight linked to S3 for reports.

2. Trigger
    a. Developer either pushes to main or manually dispatches the workflow in GitHub Actions.

3. Build & Publish
    a. The workflow checks out the repository on the self-hosted Windows runner.
    b. Docker images are built for frontend, game-service, order-service, analytics-service.
    c. Using Docker Hub credentials from GitHub Secrets, images are tagged and pushed to Docker Hub.

4. Connect to the Cluster
    a. Workflow provisions kubeconfig from GitHub Secrets
    b. Target cluster is local Minikube

5. Deployment
    a. RollingUpdate services (frontend, analytics-service):
        i. Apply/update the Deployment manifests.
        ii. Wait for rollout to complete (readiness/liveness/startup probes govern progress).
    b. Blue-Green services (game-service, order-service):
        i. Identify the current live color from the Service selector.
        ii. Update the inactive color Deployment to the new image and wait until it's Ready.
        iii. Run smoke tests against the inactive color.
        iv. Promote by switching the Service selector to the previously inactive color.

6. Post-Deploy Tests
    a. The workflow performs lightweight smoke tests for each Service (via temporary port-forward) to confirm:
        i. Frontend returns HTTP 200.
        ii. Game/Order health endpoints respond and core list endpoints work
        iii. Analytics /track accepts an event.

7. Visualization & Monitoring
    a. Grafana Dashboards:
        i. Used immediately after deployment to check service health, CPU/memory usage, error rates, and latency.
        ii. Rollout status and probe metrics confirm pods are stable.
    b. QuickSight Reports:
        i. Windows Task Scheduler exports web events from ClickHouse into S3 daily.
        ii. Amazon QuickSight dashboards visualize gameplay trends, orders, and user activity from S3 data.
        iii. Used for verifying that data pipeline and analytics remain intact after deployments.

8. Verification
    a. All deployments show as successful.
    b. Smoke tests and Grafana monitoring show no errors.
    c. QuickSight dashboards display expected event data.

9. Rollback
    a. RollingUpdate services: initiate a Deployment rollback to the previous ReplicaSet if smoke tests or monitoring fail.
    b. Blue-Green services: instantly restore by switching the Service selector back to the previously live color; optionally roll back the failed color's Deployment.

10. Acceptance Criteria
    a. Services updated without downtime.
    b. Grafana shows healthy system metrics.
    c. QuickSight dashboards reflect accurate data post-deployment.
    d. Developer confirms rollout success in GitHub Actions logs, Grafana, and QuickSight.