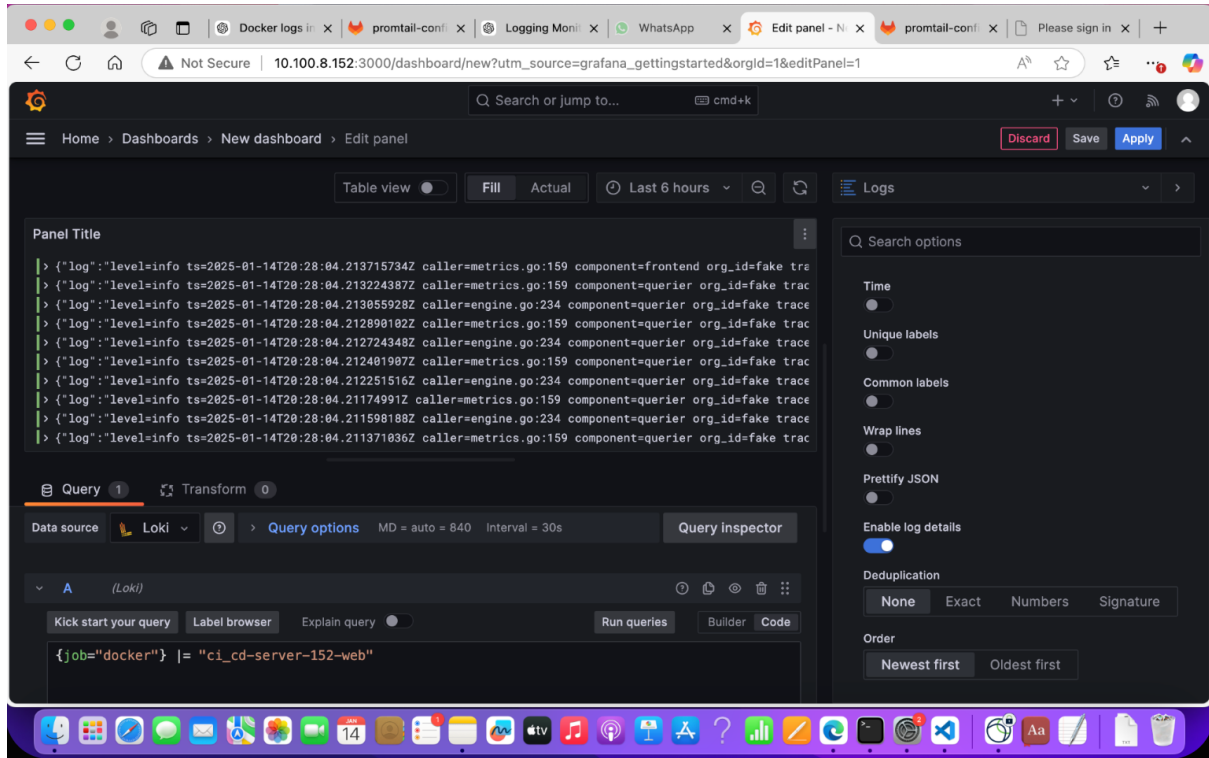


New Features: Logging and Monitoring Stack

This branch introduces a comprehensive logging and monitoring solution to centralize application logs and visualize system health.



Here's a breakdown of what the screenshot shows:

1. **Grafana Dashboard Interface:** The overall layout is that of a Grafana dashboard, a popular open-source platform for analytics and monitoring. It features navigation, a panel title, time range selector ("Last 6 hours"), and query input fields.
2. **Loki Data Source:** In the bottom left, the "Data source" dropdown clearly shows "Loki" selected. This confirms that Grafana is configured to pull log data from your Loki instance.
3. **Log Query:** The query being executed is:


```
{job="docker"} |= "ci_cd-server-152-web"
```
4. **Query Breakdown:**
 - o `{job="docker"}:` This part of the query targets logs that have been labeled with `job="docker"`. As previously documented, Promtail is configured to add this label to logs collected from Docker containers.
 - o `|= "ci_cd-server-152-web":` This is a line filter that searches for log entries containing the string "ci_cd-server-152-web". This string likely corresponds to the container name or a label associated with your Spring Boot application's Docker container, indicating that the query is specifically trying to retrieve logs from your web application.
5. **Displayed Logs:** The "Panel Title" area displays several log lines. While the query explicitly filters for "ci_cd-server-152-web", the displayed log lines show entries like `caller=metrics.go`, `component=frontend`, `caller=engine.go`, `component=querier`, and `org_id=fake trace`. These specific lines suggest that the panel might be showing internal logs from Loki itself or another component of the

monitoring stack (like Grafana's internal metrics), or that the "ci_cd-server-152-web" string might appear in these system-level logs as part of a larger context. In a typical setup, you would expect to see the actual application logs (e.g., Spring Boot INFO/ERROR messages, custom log statements) from your `MyFirstTodoApp` here.

6. **Query Options and Controls:** The right panel offers various options such as "Time", "Unique labels", "Common labels", "Wrap lines", "Prettify JSON", "Enable log details", "Deduplication", and "Order". These allow users to refine their log views, analyze log patterns, and manage how logs are presented.

Significance: This screenshot visually confirms that the logging and monitoring stack you've integrated is operational:

- **Promtail** (implicitly, by providing logs with `job="docker"` labels) is successfully collecting logs from your Docker environment.
- **Loki** is receiving and storing these logs.
- **Grafana** is able to connect to Loki as a data source and query the aggregated logs, providing a centralized interface for troubleshooting and observation.

This demonstrates a critical aspect of the `feature-logs-monitoring` branch: improved observability through centralized log management.

1. Application Logging Configuration (`logging.properties`)

- **Purpose:** This file configures the Java standard logging (JUL) for the Spring Boot application.
- **Details:**
 - `handlers=java.util.logging.ConsoleHandler:` Specifies that log messages should be sent to the console.
 - `java.util.logging.ConsoleHandler.level=INFO:` Sets the minimum logging level for the console handler to `INFO`.
 - `java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter:` Uses a simple, single-line format for log messages.
 - `.level=INFO:` Sets the default logging level for all loggers to `INFO`.

2. Updated Application Dockerfile (`Dockerfile`)

The main application's `Dockerfile` has been updated to integrate with the new logging setup.

- **Logging Properties Inclusion:**

`Dockerfile`

```
COPY logging.properties /app/logging.properties
```

- This line explicitly copies the `logging.properties` file into the `/app` directory within the Docker image.

- **Runtime Logging Configuration:**

Dockerfile

```
ENTRYPOINT ["java", "-Djava.util.logging.config.file=/app/logging.properties", "-jar", "target/MyFirstTodoApp-0.0.1-SNAPSHOT.jar"]
```

- The **ENTRYPOINT** command now includes - `Djava.util.logging.config.file=/app/logging.properties`. This tells the Java Virtual Machine (JVM) to use the provided `logging.properties` file for its logging configuration at runtime, ensuring that the application's console output is formatted and handled as intended, which can then be collected by Promtail.
- **Dependency Installation Updates:**
 - The `apt-get install` command now includes `openjdk-11-jre-headless` which seems redundant as the base image is already `openjdk-17`, but it explicitly ensures a JRE is present.
 - It also adds `--allow-insecure-repositories --allow-releaseinfo-change` to `apt-get update` for potentially broader compatibility in certain network setups, and more aggressive cache cleanup `apt-get clean && rm -rf /var/cache/apt/* /var/lib/apt/lists/* /tmp/* /var/tmp/*`.

3. Loki (Log Aggregation)

Loki is a horizontally scalable, highly available, multi-tenant log aggregation system inspired by Prometheus.

- **Loki Dockerfile (`Dockerfile.loki`):**

Dockerfile

```
FROM grafana/loki:2.9.0
USER root
RUN mkdir -p /data /wal /tmp/loki/index /tmp/loki/chunks && \
    chmod -R 777 /data /wal /tmp/loki
USER 10001
COPY loki-config.yaml /etc/loki/local-config.yaml
CMD ["-config.file=/etc/loki/local-config.yaml"]
```

- **Base Image:** Uses the official `grafana/loki:2.9.0` image.
- **Permissions:** Temporarily switches to `root` to create necessary directories (`/data`, `/wal`, `/tmp/loki/index`, `/tmp/loki/chunks`) and set `777` permissions to ensure Loki has write access to its storage locations. It then switches back to Loki's default user for security.
- **Configuration Copy:** Copies the `loki-config.yaml` file into the container at `/etc/loki/local-config.yaml`.
- **Command:** Sets the default command to start Loki using this copied configuration file.
- **Loki Configuration (`loki-config.yaml`):**
 - **`auth_enabled: false`:** Disables authentication for simplicity, suitable for local development or trusted environments.

- **server.http_listen_port: 3100:** Loki listens for incoming log streams (from Promtail) on port 3100.
- **Storage Configuration:** Configures Loki to use `boltdb-shipper` for index storage and `filesystem` for chunk storage. This means logs and their indices will be stored directly on the container's filesystem within the `/tmp/loki` directory.
- **Limits:** Sets various limits, including `reject_old_samples_max_age` (168h or 7 days), defining how long old log samples are rejected.
- **Retention:** `retention_deletes_enabled: false` and `retention_period: 0s` indicate that log retention/deletion is not actively managed by Loki in this configuration.

4. Promtail (Log Collector)

Promtail is an agent that ships the contents of local logs to a private Loki instance or other compatible systems.

- **Promtail Dockerfile (`Dockerfile.promtail`):**

Dockerfile

```
FROM grafana/promtail:2.9.0
COPY promtail-config.yaml /etc/promtail/config.yaml
CMD ["-config.file=/etc/promtail/config.yaml"]
```

- **Base Image:** Uses the official `grafana/promtail:2.9.0` image.
- **Configuration Copy:** Copies `promtail-config.yaml` to `/etc/promtail/config.yaml`.
- **Command:** Sets the default command to start Promtail with this configuration.
- **Promtail Configuration (`promtail-config.yaml`):**
 - **clients.url:** `http://loki:3100/loki/api/v1/push`: Defines the endpoint where Promtail will send collected logs. It targets the `loki` service within the Docker Compose network on port 3100.
 - **scrape_configs:** Defines how Promtail discovers and collects logs.
 - **job_name:** `docker`: A job named `docker` for collecting container logs.
 - **__path__:** `/var/lib/docker/containers/*//*.log`: This crucial setting tells Promtail to look for log files in the standard Docker container log directory. It collects logs from all containers on the host.
 - **labels:** `job: docker`: Adds a `job: docker` label to all collected logs, enabling filtering and querying in Loki.

5. Docker Compose Enhancements (`docker-compose.yml`)

The `docker-compose.yml` file has been significantly expanded to include the logging and monitoring stack.

- **New Services Added:**
 - **loki:**

- build: Uses the `Dockerfile.loki` to build Loki's image.
- ports: - "3100:3100": Exposes Loki's HTTP API port.
- networks: - app-network: Connects Loki to the application network.
- **promtail:**
 - build: Uses the `Dockerfile.promtail` to build Promtail's image.
 - volumes: This is critical:
 - - `/var/lib/docker/containers:/var/lib/docker/containers`: Mounts the host's Docker container log directory into Promtail's container, allowing Promtail to read application logs.
 - - `/var/run/docker.sock:/var/run/docker.sock`: Mounts the Docker daemon socket, enabling Promtail to get metadata (like container names and IDs) for log labeling.
 - depends_on: - loki: Ensures Loki is running before Promtail attempts to send logs.
 - networks: - app-network: Connects Promtail to the application network.
-
- **grafana:**
 - image: `grafana/grafana:9.5.3`: Uses a specific Grafana image version.
 - ports: - "3000:3000": Exposes Grafana's web UI port.
 - depends_on: - loki: Ensures Loki is available for Grafana to connect to as a data source.
 - networks: - app-network: Connects Grafana to the application network.
- **Web Service Logging Driver:**
 - The web service now explicitly configures its logging driver to `loki`. This is important for directing the web container's logs directly to Loki, bypassing the need for Promtail to scrape them from files in this specific case.

YAML

```
logging:
  driver: loki
  options:
    loki-url: "http://loki:3100/loki/api/v1/push"
    loki-external-labels: application=my-first-todo-app,job=containerlogs
```

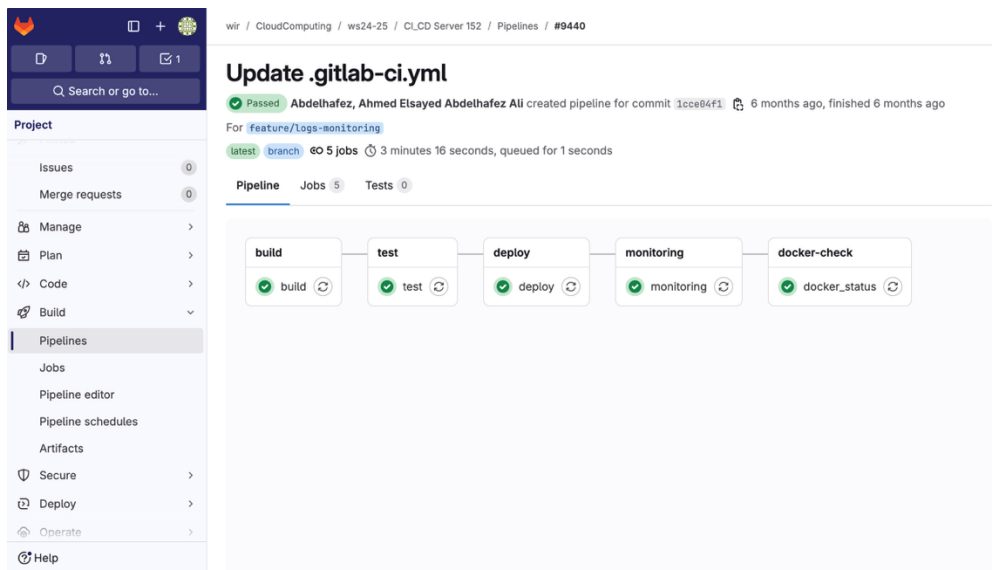
- `loki-url`: Specifies the Loki endpoint.
- `loki-external-labels`: Adds useful labels (`application` and `job`) to logs from the web service, making them easily queryable in Grafana.

6. GitLab CI/CD Pipeline Updates (`.gitlab-ci.yml`)

The GitLab CI pipeline now includes dedicated stages and jobs for managing the monitoring stack.

- **New Stages:**

- **monitoring:** For deploying the Loki, Promtail, and Grafana services.
- **docker-check:** For verifying the status of the deployed Docker containers.



- **Job Additions:**

- **build Stage Job:**
 - Added `docker system prune -f || true` at the beginning of the script. This forcefully removes unused Docker data (containers, images, networks, volumes) before building, ensuring a clean build environment and preventing disk space issues on runners.
- **test Stage Job (Unified test job, no longer split into `unit_tests` and `integration_tests` as in main 3):**
 - This branch consolidates the `unit_tests` and `integration_tests` from main 3 into a single test job.
 - It still includes the `mariadb` service and the wait loop to ensure the database is ready for integration tests.
 - The script now uses `mvn test -e -X`, which provides extended error reporting and debug output.

Todo Web App with CI/CD PIPELINE

The screenshot displays the GitLab CI/CD interface for a project named 'wird'. The left sidebar shows the project navigation menu with options like Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, and Help. The main area shows the 'Jobs' section for the pipeline #29941. The job 'test' is highlighted, showing its log output. The log output includes debug and info messages, such as 'Closing the fork 1 after saying GoodBye.', 'Results:', 'Tests run: 15, Failures: 0, Errors: 0, Skipped: 0', and 'BUILD SUCCESS'. The right sidebar provides job details: Duration: 54 seconds, Finished: 6 months ago, Queued: 2 seconds, Timeout: 1h (from project), Runner: #231 (x86f5rhu) new, Source: Push, Tags: new runner, Commit: 1cce04f1, Update: .gitlab-ci.yml, Pipeline #9440 (Passed) for feature/logs-monitoring, and Related jobs: test.

Showing last 499.08 KiB of log. [View raw or full log.](#)

Search visible log output

```
3282 [DEBUG] Closing the fork 1 after saying GoodBye.
3283 [INFO]
3284 [INFO] Results:
3285 [INFO]
3286 [INFO] Tests run: 15, Failures: 0, Errors: 0, Skipped: 0
3287 [INFO]
3288 [INFO] -----
3289 [INFO] BUILD SUCCESS
3290 [INFO] -----
3291 [INFO] Total time: 40.874 s
3292 [INFO] Finished at: 2025-01-14T19:59:41Z
3293 [INFO] -----
3294 [DEBUG] Shutting down adapter factory; available factories [file-lock, rwl
ock-local, semaphore-local, noop]; available name mappers [discriminating,
file-gav, file-hgav, file-static, gav, static]
3295 [DEBUG] Shutting down 'file-lock' factory
3296 [DEBUG] Shutting down 'rwlock-local' factory
3297 [DEBUG] Shutting down 'semaphore-local' factory
3298 [DEBUG] Shutting down 'noop' factory
3299 Cleaning up project directory and file based variables
3300 Job succeeded
```

Duration: 54 seconds
Finished: 6 months ago
Queued: 2 seconds
Timeout: 1h (from project) ⓘ
Runner: #231 (x86f5rhu) new
Source: Push
Tags: new runner
Commit: 1cce04f1 ⓘ
Update: .gitlab-ci.yml
Pipeline #9440 ✔ Passed for feature/logs-monitoring ⓘ
test
Related jobs
→ ✔ test

- **monitoring Stage Job:**

- **job: monitoring:** A new job dedicated to bringing up the monitoring stack.
- **script: docker-compose up -d loki promtail grafana:** This command selectively starts only the loki, promtail, and grafana services defined in docker-compose.yml. This allows deploying the monitoring stack independently or as part of the main application deployment.
- **tag: new runner:** Specifies runner tag.

Todo Web App with CI/CD PIPELINE

The screenshot displays the GitHub Actions interface for a CI/CD pipeline. The left sidebar shows the project structure with 'Jobs' selected. The main area shows the execution log of a job, with steps including 'Getting source from Git repository', 'Reinitialized existing Git repository', 'Checking out 1cce04f1 as detached HEAD', 'Skipping Git submodules setup', 'Executing "step_script" stage of the job script', 'Using docker image sha256:ed9a10a5bc310dfdad94ab737c61698d5a5bc8074a039804531452fe19280896 for docker:20.10 with digest docker@sha256:2967f8819c84dd589ed0a023b9d25dcfe7a3c123d5bf784ffbb77edf55335f0c ...', 'mkdir -p ~/.m2', 'echo <settings>...', 'Starting Loki, Promtail, and Grafana...', 'docker-compose up -d loki promtail grafana', and 'Cleaning up project directory and file based variables'. The job status is 'Job succeeded'. The right sidebar shows job details: Duration: 7 seconds, Finished: 6 months ago, Queued: 2 seconds, Timeout: 1h (from project), Runner: #231 (x8Gf5rhu) new, Source: Push, Tags: new runner, Commit: 1cce04f1, Update: .gitlab-ci.yml, Pipeline #9440 Passed for feature/Logs-monitoring, and Related jobs: monitoring.

- **docker_status Stage Job:**
 - **job: docker_status:** A new job to verify the health of Docker containers after deployment.
 - **services:** - **docker:dind:** Requires Docker in Docker to run Docker commands.
 - **script:**
 - `docker ps:` Lists currently running containers.
 - Includes a check `if [-z "$(docker ps -q)"]; then ...` to ensure at least one container is running. If no containers are found, it prints an error and exits with a non-zero status, failing

Todo Web App with CI/CD PIPELINE

the CI/CD job. This acts as a basic health check for the Docker services.

- **tags: new runner:** Specifies runner tag.

The screenshot displays the GitLab CI/CD interface for a project named 'ws24-25 / CLCD Server 152'. The left sidebar shows the project navigation menu with options like Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, and Help. The main area shows the job log for the 'docker-check' job, which is part of the 'Pipeline #9440' and has a status of 'Passed'. The log output shows the following commands and their results:

```
52 NAMES
53 cdbdbbd3a4ab ed9a10a5bc31 "docker-entrypoint.s..." Less
54 than a second ago Up Less than a second
55 runner-t3x8gf5r-project-1884-concurrent-0-da2bad6dae257bd-build
56 0d08b9254140 ci_cd-server-152-web "java -Djava.util.lo..." 54 s
57 econds ago Up 53 seconds 0.0.0.0:8080->8080/tcp, :::80
58 80->8080/tcp ci_cd-server-152-web-1
59 5db08a2e2187 grafana/grafana:9.5.3 "/run.sh" 54 s
60 econds ago Up 53 seconds 0.0.0.0:3000->3000/tcp, :::30
61 00->3000/tcp grafana
62 4e55305e5e52 ci_cd-server-152-promtail "/usr/bin/promtail -..." 54 s
63 econds ago Up 53 seconds
64 promtail
65 d5a408aed0e2 ci_cd-server-152-loki "/usr/bin/loki -conf..." 55 s
66 econds ago Up 54 seconds 0.0.0.0:3100->3100/tcp, :::31
67 00->3100/tcp loki
68 ee6770015b88 mariadb:10.5 "docker-entrypoint.s..." 55 s
69 econds ago Up 54 seconds 0.0.0.0:3306->3306/tcp, :::33
70 06->3306/tcp mariadb
71 0dcfcfb348 phpmyadmin/phpmyadmin "/docker-entrypoint..." 55 s
72 econds ago Up 54 seconds 0.0.0.0:8081->8081/tcp, :::8081
73 ->8081/tcp phpmyadmin
74 48f2f8970387 gitlab/gitlab-runner:latest "/usr/bin/dumb-init ..." 5 we
75 eks ago Up 5 days
76 gitlab-runner
77 $ RUNNING_CONTAINERS=$(docker ps -q)
78 $ if [ -z "$(docker ps -q)" ]; then # collapsed multi-line command
79 All containers are running.
80 Cleaning up project directory and file based variables
81 Job succeeded
```

On the right side of the log, there is a summary of the job's performance and configuration:

- Duration: 37 seconds
- Finished: 6 months ago
- Queued: 2 seconds
- Timeout: 1h (from project)
- Runner: #231 (x8Gf5rhu) new
- Source: Push
- Tags: new runner
- Commit: 1cce04f1
- Update: .gitlab-ci.yml
- Pipeline #9440 ✔ Passed for feature/Logs-monitoring
- Related jobs: ✔ docker_status

These new additions dramatically enhance the observability of the application by providing centralized logging and a platform for visualization, all integrated seamlessly into the automated CI/CD pipeline.