1. Define the Structure

Use this standard enterprise layout:

Context Engine – Intelligent DevOps Activity Tracker

├— 1. Overview

├— 2. Problem Statement

├— 3. Solution Architecture

├— 4. Key Features

├— 5. Tech Stack

├— 6. Installation / Setup Guide

├— 7. Workflow (How It Works)

├— 8. Data Flow Diagram

├— 9. Future Enhancements

├— 10. References

🏗️ **2. Sample Confluence Documentation (Full Content Below)**

---

1. Overview

Context Engine is an intelligent activity tracking system designed to analyze and optimize DevOps workflows.
It intelligently captures contextual actions, automates reporting, and correlates engineering tasks with business impact.

The engine provides a unified view of “what engineers do” — enabling better productivity insights and operational awareness.

---

2. Problem Statement

DevOps teams operate in complex, distributed environments — context switching between Git, Jenkins, Jira, and multiple dashboards.

Tracking actual engineering activity and its correlation to delivery outcomes remains manual and inconsistent.

Challenges identified:

- Lack of contextual visibility across tools.
- Manual status reporting and syncs.
- No real-time insight into work efficiency.
- Hard to quantify developer productivity.

---

3. Solution Architecture

Context Engine acts as a layer between developer tools and analytics systems.

It:

- Collects real-time events from developer tools (Git commits, Jenkins builds, tickets, etc.).
- Correlates and enriches them with metadata (user, environment, project).
- Sends structured metrics to observability tools or dashboards.
- Generates insights for productivity, automation coverage, and delivery velocity.

🧩 **Architecture Components**

- **Collector Layer:** Gathers activities (via APIs, webhooks, or local agents).
- **Processing Core:** Normalizes and processes contextual data.
- **Insight Engine:** Applies rules and ML models to detect patterns and trends.
- **Dashboard / Exporter:** Sends analytics to Grafana, Prometheus, or Confluence reports.

---

4. Key Features

- 📊 Real-time DevOps activity correlation
- 🧠 Intelligent context mapping
- 🔄 Git + Jenkins + Jira integration (extensible)

- 🧩 Modular architecture for plug-and-play sources
- 📈 Built-in metrics exporter for observability tools
- 🔐 Secure token-based communication between modules

---

5. Tech Stack

Component	Technology
Core Logic	Python / Node.js
Data Layer	SQLite / PostgreSQL
Dashboard	Grafana / Custom HTML
APIs	REST / WebSocket
Hosting	Docker / Kubernetes-ready

---

6. Installation / Setup Guide

Step 1: Clone repository

git clone https://github.com/sainathmitalakar/context-engine.git

cd context-engine

Step 2: Install dependencies

npm install

Step 3: Start the application

npm start

Step 4: Access dashboard

http://localhost:3000

---

7. Workflow (How It Works)

1. Engineer performs actions (commit → build → deploy).

2. Context Engine listens to these events via webhooks or API calls.
3. Data is processed and tagged with metadata (service, environment, time, user).
4. Enriched data is visualized via dashboard or pushed to observability tools.
5. Insights are generated automatically for reporting and optimization.

---

8. Data Flow Diagram (Simplified)

[Engineer Action] → [Webhook/API Listener] → [Context Processor]

↓

[Insight Engine] → [Dashboard / Exporter] → [Stakeholders]

---

9. Future Enhancements

- AI-driven context prediction for DevOps incidents.
- Integration with OpenTelemetry for unified traces.
- Context-aware anomaly detection for CI/CD pipelines.
- Multi-cloud integration layer (AWS, GCP, Azure).

---

10. References

- Internal DevOps Observability Framework
- OpenTelemetry Documentation
- T-Mobile Productivity Insight Framework (inspiration)
- Grafana + Loki + Tempo Stack References

==========

2. Running the CLI (Main Execution)

Run the Context Engine CLI module directly

python -m cli.context_cli sainathmitalakar/context-engine

Run with custom repository

python -m cli.context_cli sainathmitalakar/sainathmitalakar.github.io

Run help command to view available options

python -m cli.context_cli --help

3. Repository Intelligence Commands

Show repository activity

python -m cli.context_cli <repo-path>

Example

python -m cli.context_cli sainathmitalakar/context-engine

Fetch commit logs and workflow analysis

python -m cli.context_cli --analyze

Export report to a file (optional future enhancement)

python -m cli.context_cli --export report.txt

4. Debug & Logging Commands

Run CLI in debug mode (shows internal logs)

python -m cli.context_cli --debug

Run with verbose flag for detailed output

python -m cli.context_cli --verbose

Save output to file for logging

python -m cli.context_cli sainathmitalakar/context-engine > output.log

5. Package Management & Maintenance

Freeze dependencies

pip freeze > requirements.txt

Upgrade pip and dependencies

python -m pip install --upgrade pip

pip install --upgrade -r requirements.txt

Uninstall a package

pip uninstall <package-name>

Check outdated packages

pip list --outdated

6. Development & Testing

Run project locally (test mode)

python cli/context_cli.py sainathmitalakar/context-engine

Execute unit tests (if tests added later)

python -m unittest discover -s tests -p "*.py"

Check Python syntax and linting

python -m py_compile cli/context_cli.py

7. Git + Python Automation

Example automation: commit log extraction via Python

```
python -c "import os; os.system('git log --oneline -5')"


# Run Python script after every commit (can add in hooks)
python cli/context_cli.py --auto
```

8. Cleanup

```
# Deactivate virtual environment
deactivate


# Remove venv
rm -rf venv


# Remove compiled cache
find . -type d -name "__pycache__" -exec rm -r {} +
```

9. Quick Summary Table

Command	Description
python -m cli.context_cli sainathmitalakar/context-engine	Run main CLI
python -m cli.context_cli sainathmitalakar/sainathmitalakar.github.io	Check portfolio repo commits
pip install -r requirements.txt	Install dependencies
pip freeze > requirements.txt	Save dependencies
python -m unittest discover	Run tests

Command	Description
python -m py_compile cli/context_cli.py	Check syntax
deactivate	Exit virtual environment