AI-driven DevSecOps for Embedded Systems

Rainer Poisel, Stefan Riegler

honeytree Labs

Overview

- Introduction
- Usage Scenarios
- Demo / Results
- Outlook

About the Speakers

- Rainer Poisel
  - Embedded Systems DevSecOps
  - Embedded Systems CI/CT/CD
- Stefan Riegler
  - SW / Embedded / IIoT Security
  - Reverse Engineering / Forensics

Process Automation, Security Testing
(IEC 62443, CRA)

honeytree
Labs

Introduction

- **Original-Title**:
  Secure by Design: Wie KI und DevSecOps die Embedded Systems-Entwicklung verändern

The Scenario:
A Zero-Day That Almost Shipped

- Discovery: Buffer overflow inside the network stack
- Context: 24 hours before customer deployment
- Scope: 10,000+ devices already in field

Enter AI-DevSecOps

LLM + MCP + pytest + labgrid

- Automated vulnerability analysis
- Intelligent test generation
- Distributed hardware testing

The Traditional Response

- Manual code audit
- Manual test creation
- 3-week validation cycle

Business impact:

- $500K penalty clause
- Customer relationship at risk

AI-DevSecOps in Action (1/2)

- **Automated Vulnerability Analysis**
  - LLM scans code, generates vectors & regression suite in **2h** (vs. 2w)
- **Intelligent Test Generation**
  - Prompted fuzzing for `tcp_packet_handler()`
  - pytest suite with edge cases & 95% coverage (vs. 60% manual)

AI-DevSecOps in Action (2/2)

- **Distributed Testing Pipeline**
  - labgrid provisions **40+** devices, fast, parallel test execution
  - Results/Logs aggregated with AI-powered analysis (RAG)
- **Technical Outcome**
  - Vulnerability identified, patched, tested in **4h**
  - Regression suite integrated into CI/CD
  - Zero customer impact

Large Language Models (LLMs)

- Most used type of a probabilistic generative model
- Can understand & generate natural language, code, and structured data
- Key enabler for **automation in DevSecOps**

Model Context Protocol (MCP)

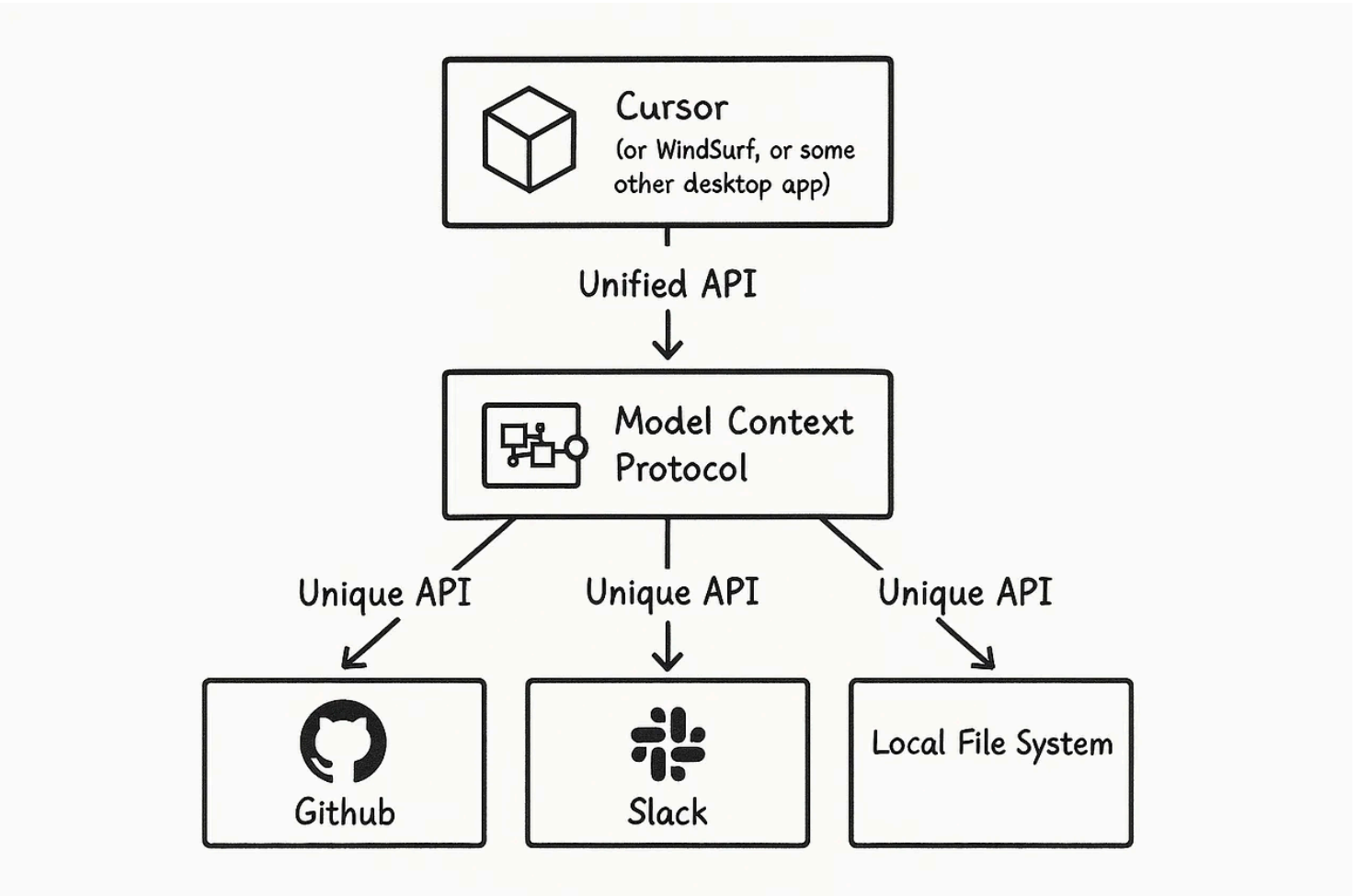


Figure 1: MCP Overview (Addy Osmani - MCP: What It Is and Why It Matters)

honeytree Labs

Model Context Protocol (MCP)

- Open protocol to connect LLMs with tools, data & workflows
- Think of it as **“IPC for AI systems”**
- Ensures reproducibility, interoperability, and integration into pipelines

Note: Still a long way to go in comparison to IPC (authorization, versioning, etc.)

Retrieval-Augmented Generation (RAG)

- Technique to ground LLMs with **domain-specific knowledge**
- Reduces hallucinations, improves accuracy
- Ideal for **security & testing contexts**

What We Demonstrate Today

- **MCP**: generates test cases automatically from GitLab commits/PRs
- **pytest**: automated testing framework
- **labgrid**: hardware device management for embedded testing

Together: **AI-driven DevSecOps pipeline** that scales, integrates, and hardens security testing

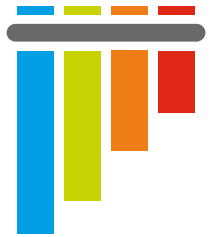
Motivation

- Product Certification Process
  - IEC 62443
  - CRA / EUCC
- Automate “Boring Stuff”
- Increase Efficiency
- Faster Adaption
- Developers vs. Test Engineers

Test Framework

pytest

- Python-based
- 13.1k Stars on GitHub
- Well-known to LLMs
- Easy to use



pytest

labgrid (I)

- Local/Distributed Infrastructure
- Pytest Integration
- CLI/Library Usage
- Drivers
- Strategies
- Virtualization Support

labgrid (II)

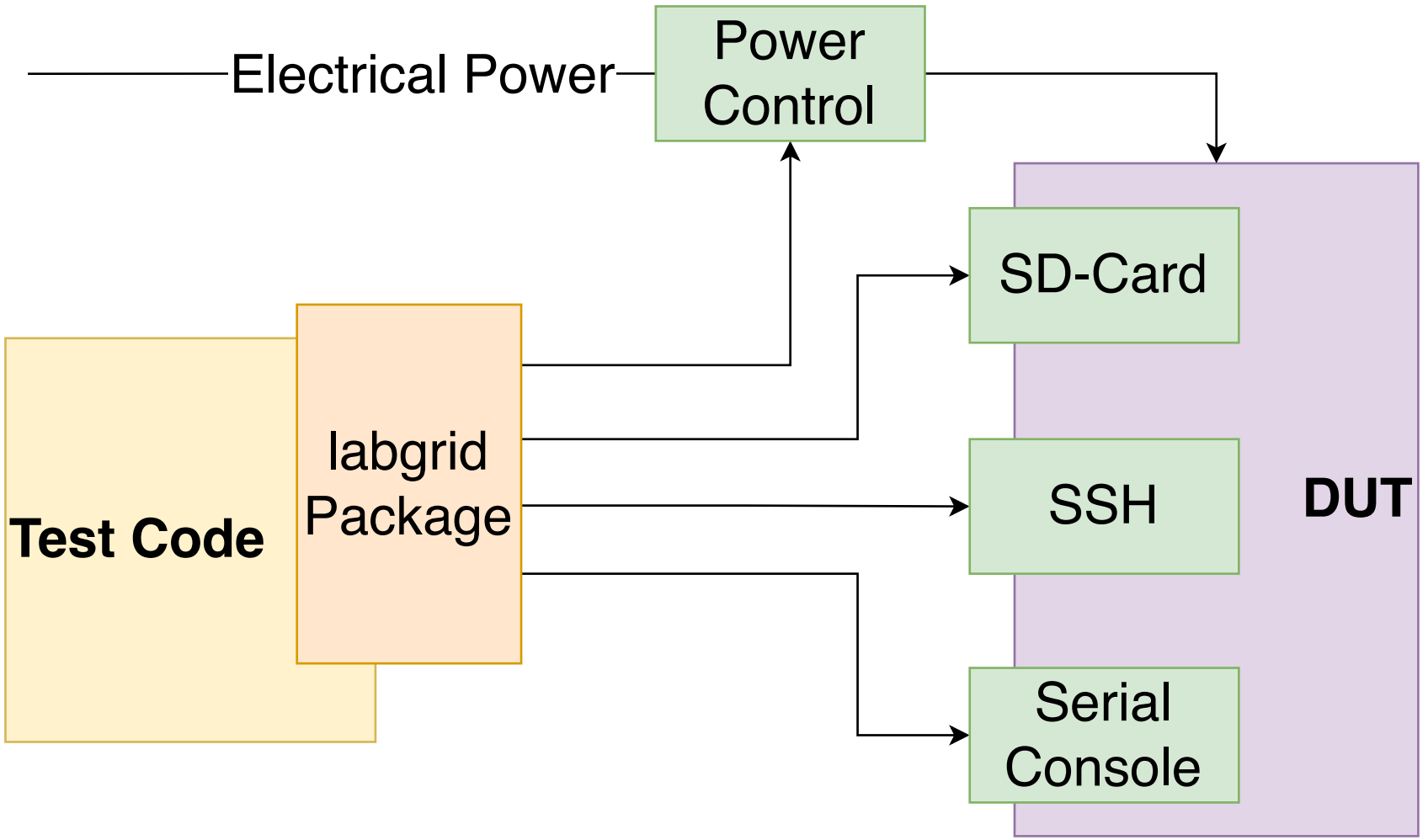


Figure 2: Labgrid Local Architecture

labgrid (III)

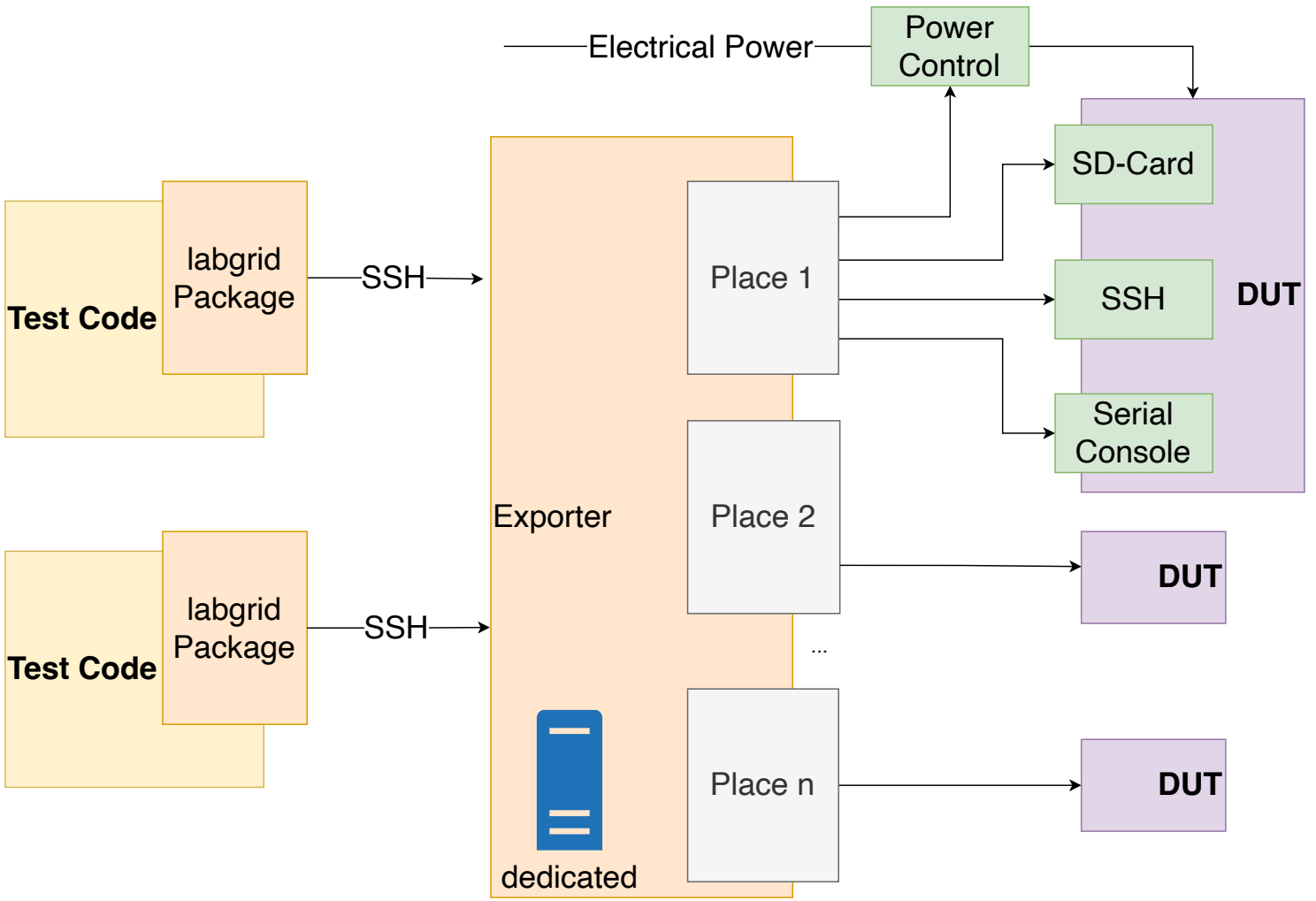


Figure 3: Labgrid Distributed Architecture

honeytree Labs

labgrid: Example (I)

Task: Check that `uname -s` returns the string `Linux`

```python
from labgrid.util.ssh import SSHConnection

def test_uname(ssh_cmd: SSHConnection) -> None:
    stdout = ssh_cmd.run_check("uname -s")
    assert stdout == "Linux"
```

Figure 4: Simple pytest/labgrid sample

labgrid: Example (II)

Not in the code:

- Power on
- Pass bootloader
- Wait for login
- Wait for shell
- Configure DHCP client (if required)
- Determine IP

Automatic Testcase Generation

Components

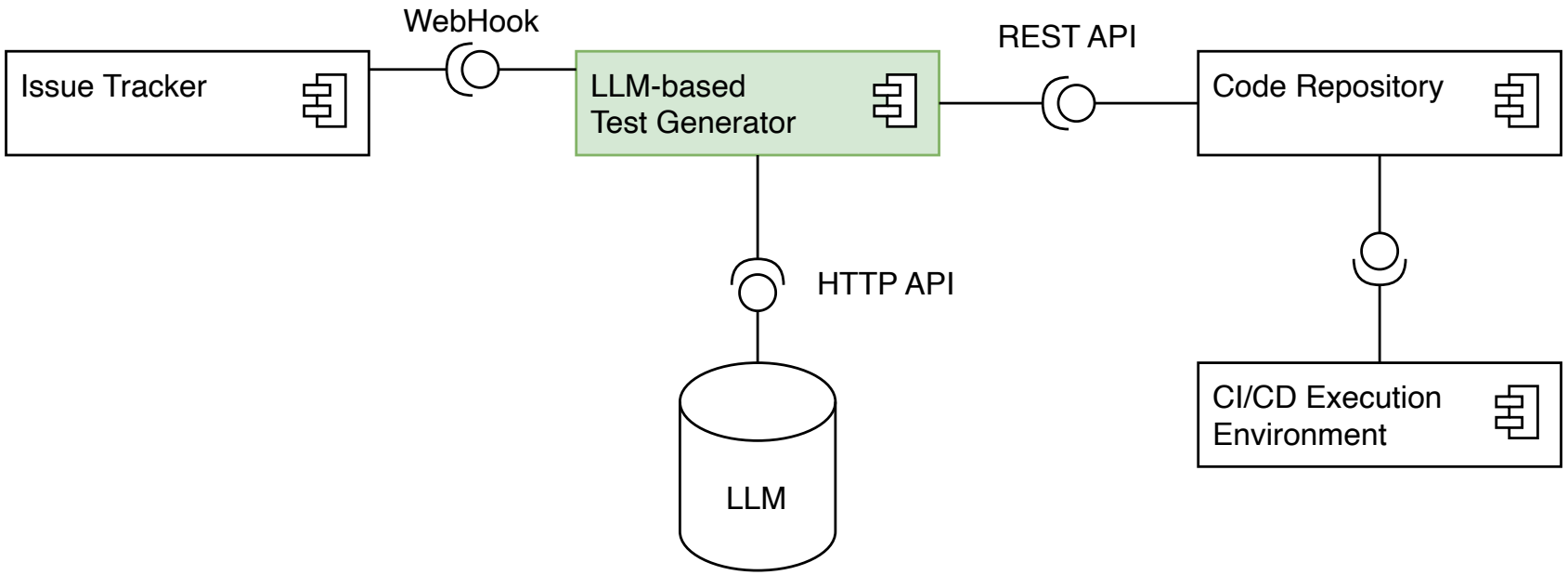


Figure 5: Components Diagram

Test Generation Sequence

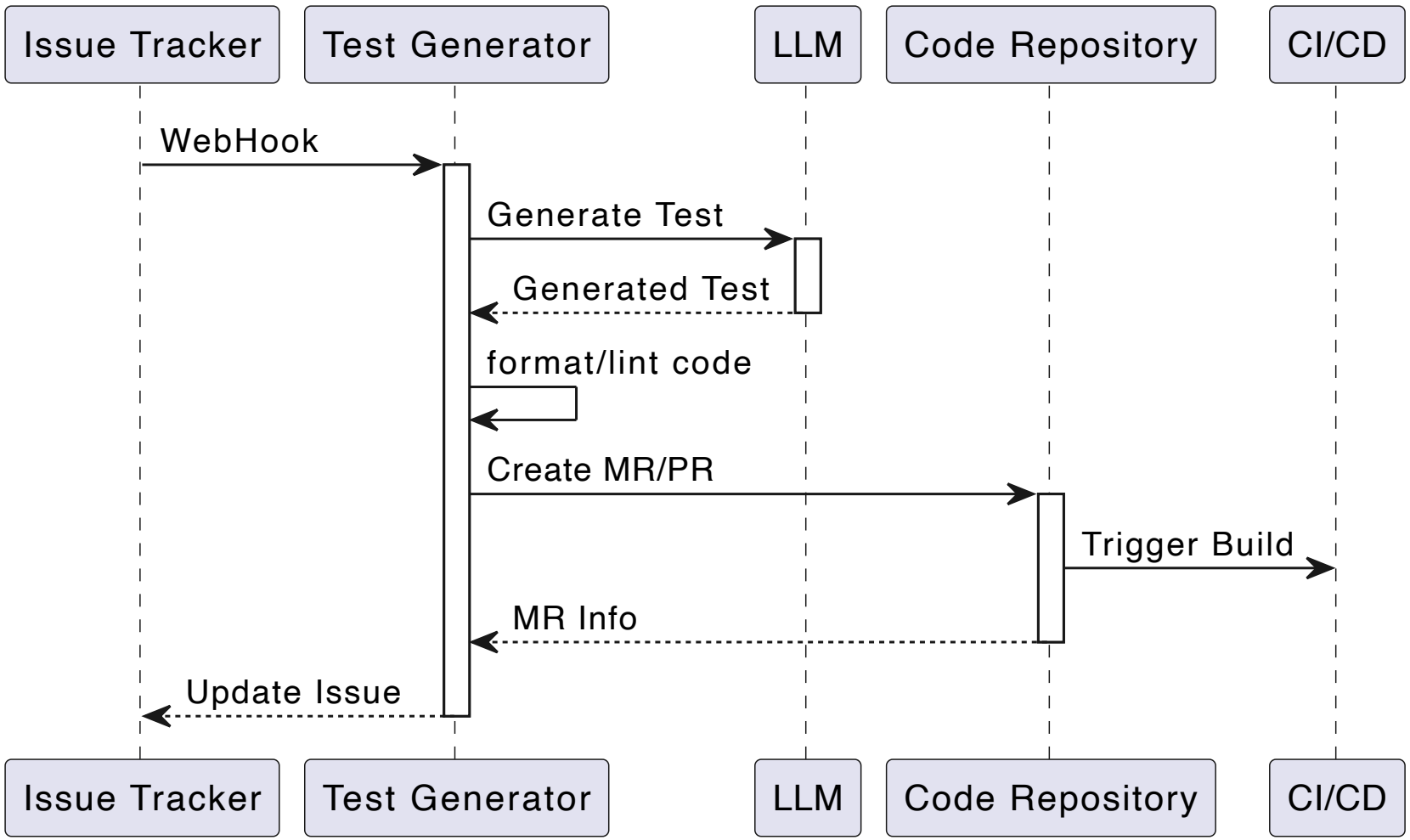


Figure 6: Sequence Diagram

honeytree Labs

Prepared Command Prompt (I)

````
Generate a pytest from the following description in markdown format:
```markdown
{description}
```

Terminology:

< ... >

Important:

< ... >

Code Guidelines:

< ... >
````

Figure 8: Prepared Command Prompt

Prepared Command Prompt (II)

- **Terminology**: Abbreviations
- **Important**:
  - High-Level Test Code Requirements
  - General Structure of generated Code
- **Code Guidelines**:
  - Description of the test infrastructure
  - Libraries/Techniques to follow/avoid

Large Language Model

- Open-Source Models
  - Self-Hosted: Privacy
  - Hosted: Low computing resources available
- Models evaluated:
  - `meta/meta-llama-3.1-405b-instruct`
  - `meta/meta-llama-3-70b-instruct`

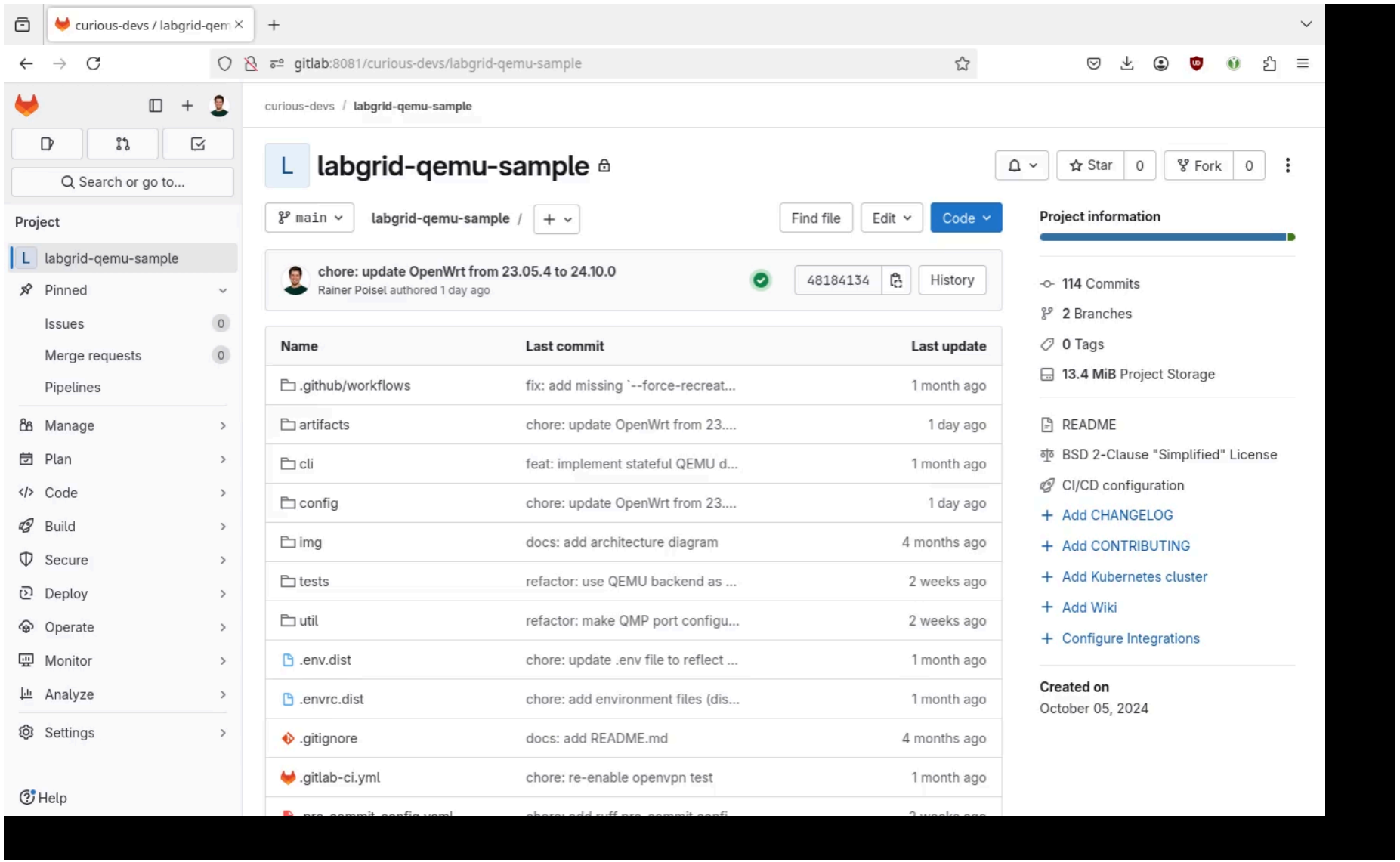
Software Engineering and Architecture Roles

- System/Software Architect
- Security Architect
- Software Developer
- Quality Assurance:
  - Framework/Library Developer
  - Test Developer

Demo: The Test Case

- Determine Listening TCP Ports
  - Expected Listening Ports: `[22, 80, 443]`
- Determine Listening UDP Ports
  - Expected Listening Ports: `[]`

Demo: Workflow



honeytree Labs

MCP: Use-Cases

MCP Labgrid Server: Use-Cases (I)

Examples:

- Autonomous Regression Selector
  - LLM picks relevant test packs from PR changes
  - Runs related tests on DUTs
- Firmware Bisect Bot
  - Given a failing test
  - Agent auto-bisects changes to identify the first bad commit

MCP Labgrid Server: Use-Cases (II)

Further examples:

- No-Code Security
  - Inspired by “No-Code Development”
  - Security Audits and System Analyses w/o Coding
  - Empowering non-technical Stakeholders
  - Proactive Security Culture
- Reverse Logistics
  - Automatic diagnosis of customer problem reports

MCP Labgrid Server: Components

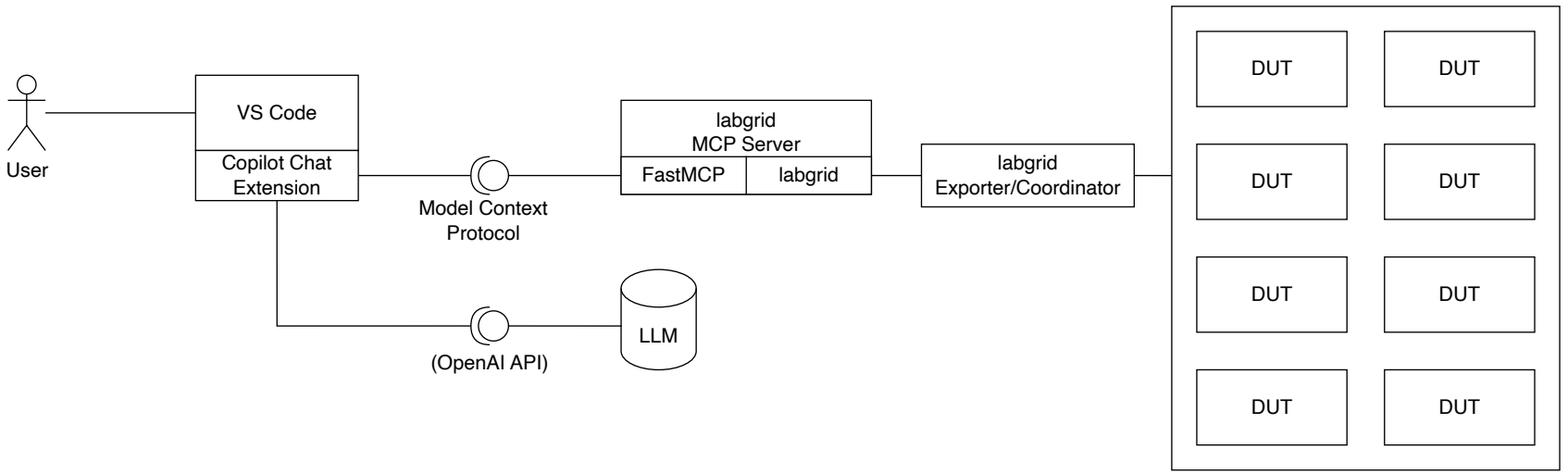


Figure 9: Components Diagram

honeytree Labs

MCP Labgrid Server: Sequence

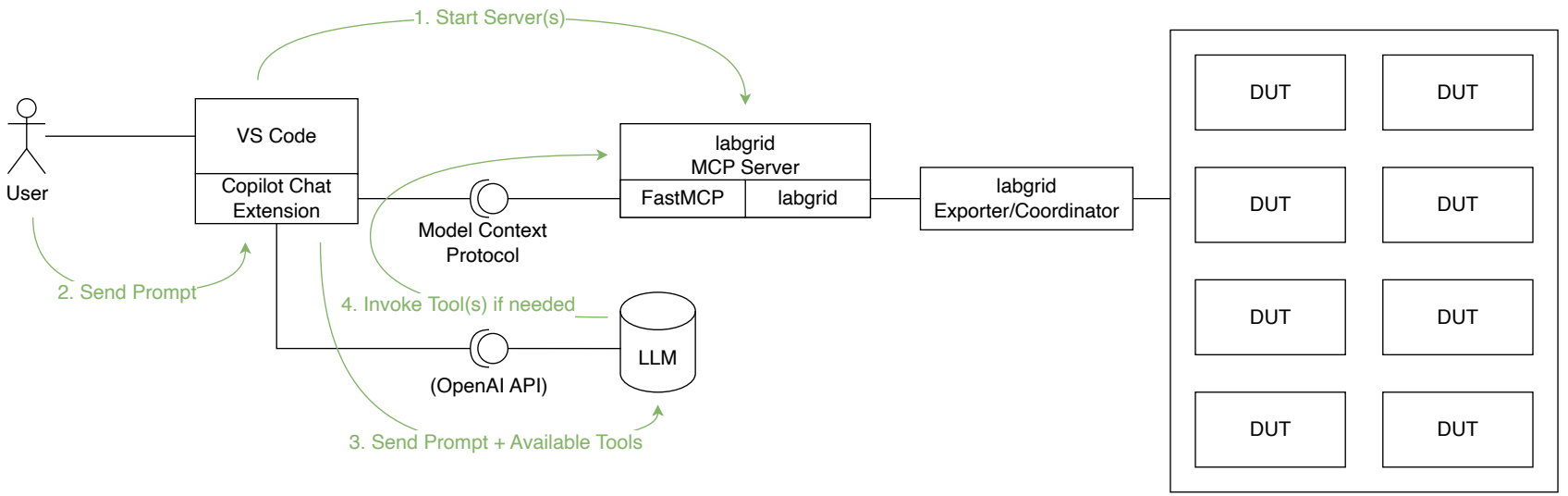
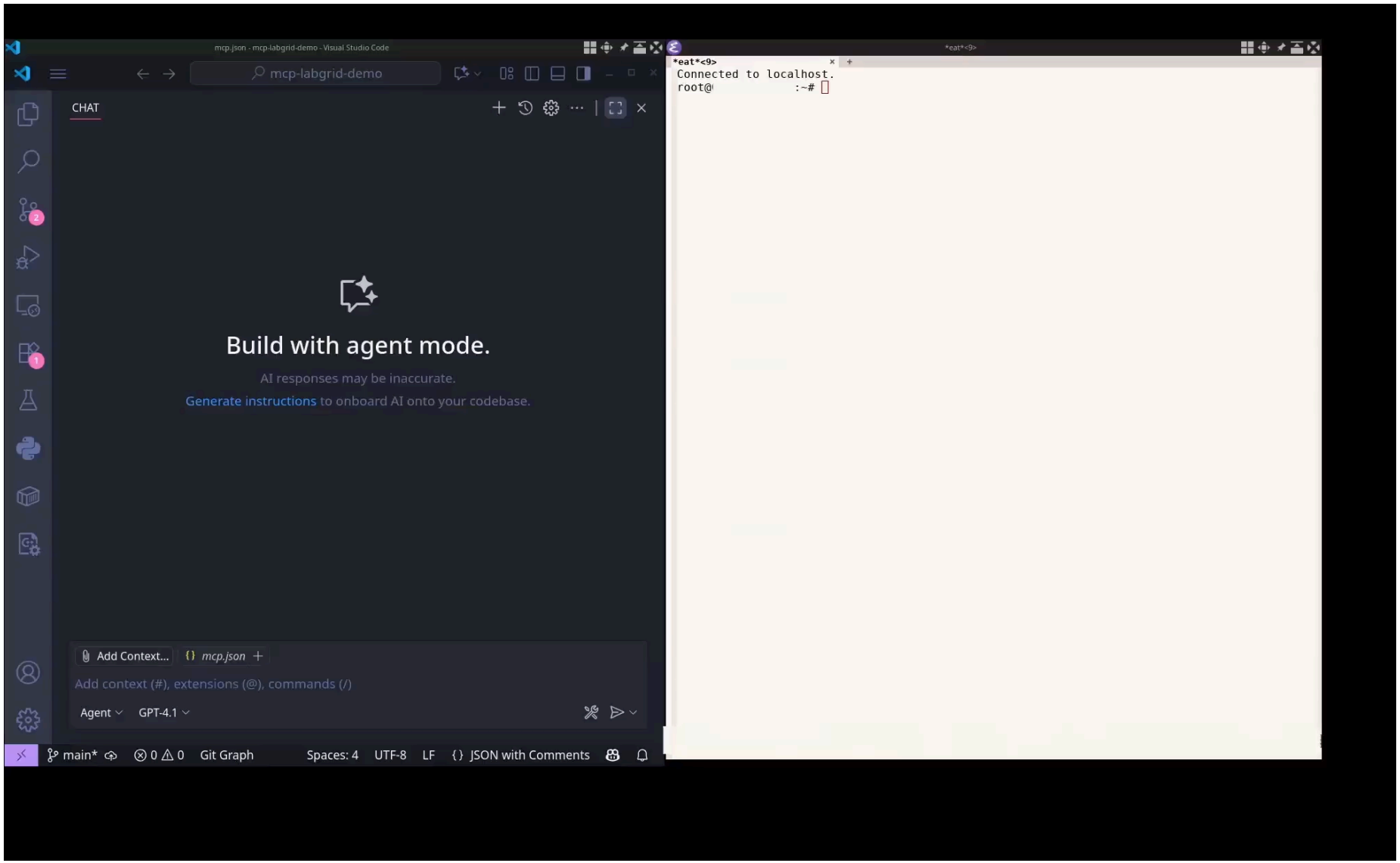


Figure 10: Components Diagram with Sequence

MCP Labgrid Server: Demo



honeytree Labs

AI-based Logfile Analysis

honeytree Labs

Retrieval Augmented Generation

Use-Case: Talk to your log files.

1. **Retrieve**: Find relevant log entries
2. **Augment**: Add context to prompt
3. **Generate**: Use LLM to generate human-readable answer

Retrieval Augmented Generation

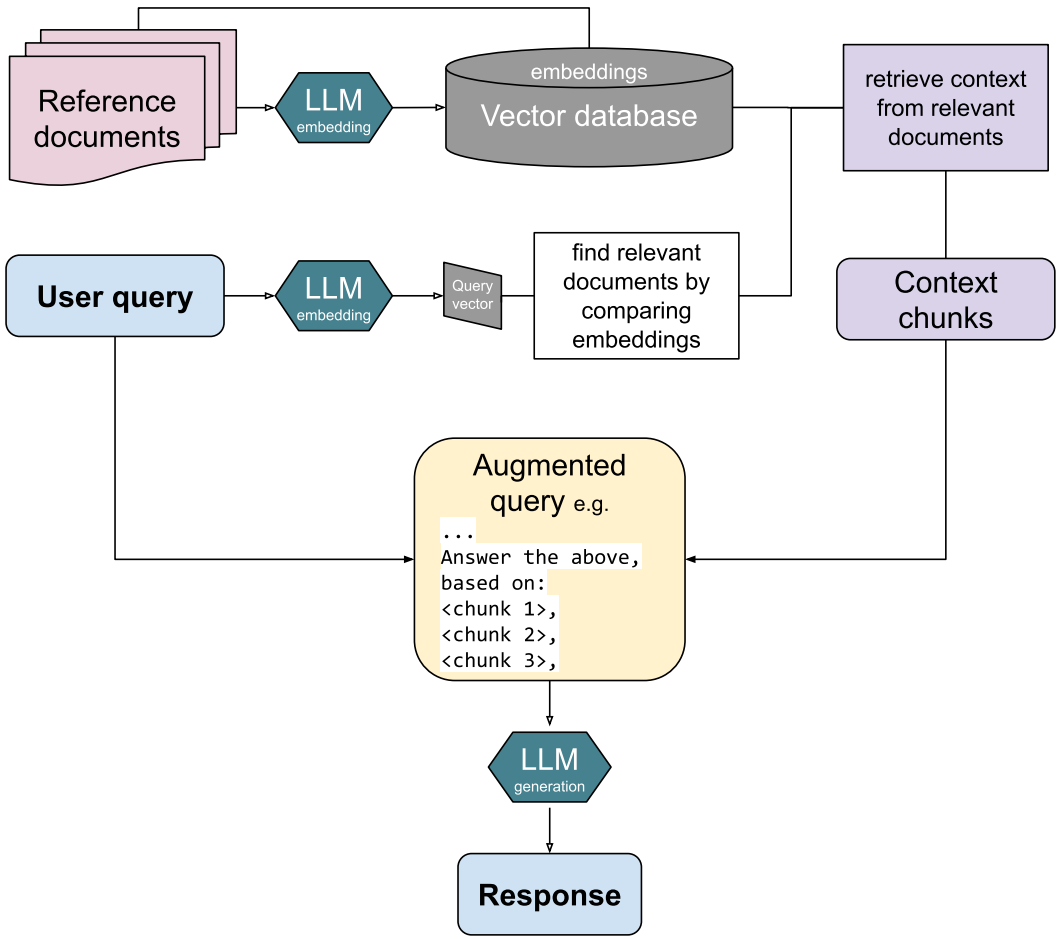


Figure 11: RAG Concept (Wikimedia Commons)

Pattern Matching: Supervised Learning

Use-Case:
Classify test run log files by **known** error conditions.

1. **Collect**: Log files from previous test runs
2. **Classify and Learn**: Determine causes of failures
3. **Apply**: Classify new log files

Pattern Matching: Unsupervised Learning

Use-Case:

Discover novel, **unknown** error patterns

1. Convert logs into embeddings
2. Clustering similar log events together.
3. Outlier clusters represent potential anomalies.

Outcome:

Engineers only analyze anomalous clusters

Future Vision

(Still) Limited Practical Suitability

- Simple Complexity
- Limits of Prompt Engineering
- Hosted LLMs Used

Agent Client Protocol (ACP)

- Standardizes communication between AI agents and clients
- Complements MCP by focusing on agent orchestration & interoperability
- Enables multi-agent, multi-stage DevSecOps workflows

Test Generation: Feedback Loop

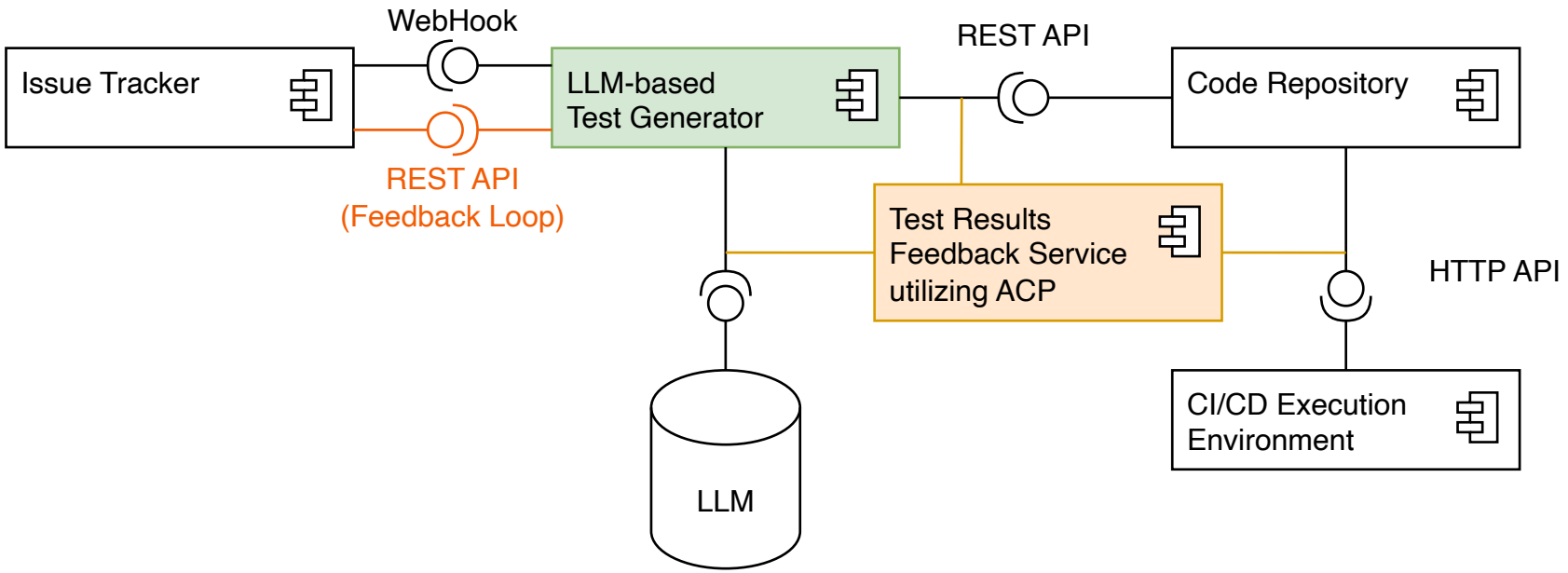


Figure 12: Components with Feedback Loop

Call to Action

- Say Hi 👋
- Connect on LinkedIn:
  - in/rainer-poisel
  - in/stefan-riegler-oe3sha

Addendum

References

pytest

https://pytest.org

labgrid

https://pengutronix.de/en/software/labgrid.html

labgrid QEMU Sample

https://github.com/Embedded-Focus/labgrid-qemu-sample

Our Services

- Embedded CI/CD
- Software Development
- DevSecOps Trainings
- Security Audits
- Reverse Engineering
- Forensics
- Automation
- Modernization
- Supply Chain Security

honeytree Labs